

Travaux Dirigés sur Machine n°5 — C++ : vecteurs et matrices — 3h non noté

L'objectif de ce TDM est de construire des classes `Vector` et `Matrix` pour réaliser des opérations sur les objets mathématiques correspondant. L'application proposée est d'implanter la *factorisation LU* d'une matrice carrée.

Exercice 1. Les classes `Vector` et `Matrix`

Les fichiers `vector.hpp` et `matrix.hpp` donnent les définitions documentées des classes. Les fichiers `vector.cpp` et `matrix.cpp` contiennent les squelettes des méthodes à implanter.

On considère le cas général des matrices rectangulaires. Les opérations comme la multiplication ou l'addition de matrices doivent tenir compte des contraintes de dimension.

Les méthodes suivantes ne s'appliquent que sur des matrices carrées :

- `set_identity`,
- `extract_triangular_lower_diag_one`,
- `extract_triangular_upper_diag`, et
- `extract_diagonal`.

Il doit y avoir tous les `asserts` correspondant aux conditions d'utilisation.

Les méthodes et fonctions `operator` illustrent la définition et l'utilisation d'opérateurs ainsi que la surcharge.

L'archive propose également deux programmes de test pour les deux classes :

- `test_vector.cpp` pour la classe `Vector` et la vérification des différents opérateurs.
- `test_matrix.cpp` pour la classe `Matrix`.

Ces programmes peuvent être complétés pour réaliser d'autres tests selon les besoins.

Exercice 2. Factorisation LU par élimination de Gauss

La factorisation LU d'une matrice carrée A consiste à trouver une matrice triangulaire inférieure L dont la diagonale est à 1 et une matrice triangulaire supérieure U telles que

$$A = LU.$$

La Figure 1(a) montre la forme de la décomposition LU.

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{2,1} & 1 & 0 & 0 \\ l_{3,1} & l_{3,2} & 1 & 0 \\ l_{4,1} & l_{4,2} & l_{4,3} & 1 \end{pmatrix} \times \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ 0 & u_{2,2} & u_{2,3} & u_{2,4} \\ 0 & 0 & u_{3,3} & u_{3,4} \\ 0 & 0 & 0 & u_{4,4} \end{pmatrix} \quad \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ l_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} \\ l_{3,1} & l_{3,2} & u_{3,3} & u_{3,4} \\ l_{4,1} & l_{4,2} & l_{4,3} & u_{4,4} \end{pmatrix}$$

(a) décomposition LU
(b) stockage de L et U

FIGURE 1 – Factorisation LU d'une matrice et stockage.

On voit sur la Figure 1(a) que l'information des deux matrices peut être stockée dans une seule... Dans l'implantation demandée, la matrice initiale est modifiée de manière à contenir en sortie la matrice L (hormis la diagonale où tout est à 1) dans sa partie triangulaire inférieure et dans sa partie triangulaire supérieure, y compris la diagonale la matrice U . Cela est illustré sur la Fig 1(b).

L'archive fournit un fichier `test_factorize.lu.cpp` pour tester la factorisation de matrices.

L'algorithme classique consiste à transformer A progressivement en une matrice triangulaire supérieure en utilisant le pivot de Gauss, les pivots successifs permettant de construire L . Par exemple soit $A = \begin{pmatrix} 2 & 3 & 4 \\ 4 & 9 & 9 \\ 6 & 15 & 18 \end{pmatrix}$, on considère que $U = A$ initialement et $L = I$ (identité).

On part de la diagonale où l'on prend le pivot (le 2 en haut à gauche). On le laisse en place (donc dans U).

On soustrait des multiples de la première ligne de manière à annuler les coefficients de la première colonne en dessous du pivot. Sur l'exemple, on soustrait $\frac{4}{2}(2 \ 3 \ 4)$ de la seconde ligne et on note $\frac{4}{2} = 2$ dans la première colonne de L . De même pour les lignes en dessous. On obtient :

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix} \text{ et } U = \begin{pmatrix} 2 & 3 & 4 \\ 0 & 3 & 1 \\ 0 & 6 & 6 \end{pmatrix} .$$

Le second pivot est à suivre sur la diagonale, ici 3. On fait de même (sans toucher aux colonnes à la gauche du pivot ou les lignes au dessus) et on arrive à :

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{pmatrix} \text{ et } U = \begin{pmatrix} 2 & 3 & 4 \\ 0 & 3 & 1 \\ 0 & 0 & 4 \end{pmatrix} .$$

Tout mis bout à bout et dans une seule matrice, cela correspond au pseudo-code suivant :

Pour d de 1 à n

$pivot = m_{d,d}$

$m_{l,d} /= pivot \ (d < l \leq n)$

$m_{l,c} -= m_{d,c} * m_{l,d} \ (d < l \leq n, d < c \leq n)$

Fin_Pour

On travaille avec l'hypothèse que le pivot n'est jamais nul.

(1) Compléter les tests de `test_factorisation.lu` par des testes sur des matrices plus grandes et/ou avec des valeurs plus importantes dedans. Que constate-on ? Comment l'expliquer ? Comment éviter cela ?