Travaux pratiques #3 — Matrices et vecteurs

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

Les classes `Vector` et `Matrix` — *Spybreak !*

Les fichiers `vector.hpp` et `matrix.hpp` donnent les définitions documentées des classes.

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 :
— `creer_identite`,
— `extraire_triangle_diagonale_inferieure`,
— `extraire_triangle_diagonale_superieure`, et
— `extraire_diagonale`.

Les méthodes et fonction `operator` illustrent la définition d'opérateur ainsi que la surcharge.

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

L'archive propose également deux programmes de test pour les deux classes :
— `test_vector.cpp` pour la classe `Vector`.
— `test_matrix.cpp` pour la classe `Matrix`.
Ils peuvent être complétés pour réaliser d'autres tests selon les besoins.

Factorisation LU par élimination de Gauss — *Gauss-Jordan.*

La factorisation LU d'une matrice carrée A consiste à définir 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 un exemple de décomposition LU.

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

(a) Décomposition LU

$$\begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ l_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} \\ l_{3,1} & l_{3,2} & u_{3,3} & u_{3,4} \\ l_{4,1} & l_{4,2} & l_{4,3} & u_{4,4} \end{pmatrix}$$

(b) stockage de L et U

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

La Figure 1(a) illustre le fait que l'information des deux matrices peut être stockée dans une seule. La matrice initiale est modifiée et en sortie elle contient dans sa partie triangulaire inférieure la matrice L (hormis la diagonale où tout est à 1) et dans sa partie triangulaire supérieure la matrice U diagonale comprise (voir Figure 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).

D'une part de la diagonale où l'on prend le pivot (le 2 en haut à gauche). On le laisse en place (donc dans U) et on soustrait la première ligne de manière à annuler les coefficients de la première colonne en dessous du pivot. On note les coefficients dans la première colonne.

On soustrait $\frac{4}{2}(2\ 3\ 4)$ de la seconde ligne et on note $\frac{4}{2}$ dans le première colonne de L. De même pour les autres lignes. 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
  $pivot = m_{d,d}$
  $m_{l,d}$ /= $pivot$ $(d < l \leqslant n)$
  $m_{l,c}$ -= $m_{d,c} * m_{l,d}$ $(d < l \leqslant n,\ d < c \leqslant n)$
Fin_Pour

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

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 ?