# Travaux dirigés Produit matriciel n°3

**Mathématiques pour l'informatique**

*—IMAC 2—*

▸ **Exercice 1.** ***Multiplication de Strassen***

1. *La méthode de multiplication de matrices de Strassen permet de multiplier des matrices en $O(n^{2,81})$ plutôt qu'en $O(n^3)$. En théorie, à partir de quel ordre la multiplication de 2 matrices carrée cette méthode est-elle 2 fois plus rapide que la méthode standard?*

2. *Codez la méthode de Strassen en C++ en utilisant une fonction récursive. Dans cet exercice, nous nous limiterons aux matrices carrées de taille $2^n \times 2^n$. À chaque récursion, la fonction traite des matrices aux dimensions deux fois plus petites. La condition d'arrêt devrait être quand les matrices ont une taille de 1x1, pourtant, nous allons prendre une limite plus large : si les matrices utilisées ont une taille $n < 100$, alors renvoyez directement leur produit effectué par* `Eigen`*.*

$$\begin{array}{|c|c|} \hline r \\ ae+bg & s \\ af+bh \\ \hline t \\ ce+dg & u \\ cf+dh \\ \hline \end{array} = \begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array} \times \begin{array}{|c|c|} \hline e & f \\ \hline g & h \\ \hline \end{array}$$

*Avec :*

$$\begin{array}{ll} P_1 = a(f-h) & \\ P_2 = (a+b)h & \\ P_3 = (c+d)e & \quad et \\ P_4 = d(g-e) & \\ P_5 = (a+d)(e+h) & \\ P_6 = (b-d)(g+h) & \\ P_7 = (a-c)(e+f) & \end{array} \qquad \begin{array}{l} r = P_5 + P_4 - P_2 + P_6 \\ s = P_1 + P_2 \\ t = P_3 + P_4 \\ u = P_1 + P_5 - P_3 - P_7 \end{array}$$

*Vous pourrez vous aider des fonction de Eigen suivantes :*

- `A.topLeftCorner(rows,cols)`
- `A.topRightCorner(rows,cols)`
- `A.bottomLeftCorner(rows,cols)`
- `A.bottomRightCorner(rows,cols)`

*où* `A` *est une matrice. Vous pouvez tout aussi bien utiliser la commande* `block`*.*

▸ **Exercice 2.** ***Multiplication matricielle : Benchmark***
*Dans cet exercice, nous allons comparer les performances de notre produit matriciel codé au tp précédent, avec Strassen et le produit de la bibliothèque Eigen.*

1. *Faites un test de performance de vitesse entre :*
   - *votre version "trois boucles* `for`*"*
   - *Strassen*
   - *le produit matriciel de Eigen*
2. *Comparez avec la version multi-threadée de Eigen, qui fonctionne de la façon suivante :*
   - *Regardez combien de cœurs a votre machine*
     - *Linux :* `grep -c processor /proc/cpuinfo`
     - *Mac :* `sysctl hw.physicalcpu` *et* `sysctl hw.logicalcpu`
     - *Windows : dites moi comment vous faites*
   - *Installez OpenMP :*
     - *Linux : probablement rien à faire*
     - *Mac :*
       * *si g++ n'est pas installé, installez le :* `brew install gcc`
       * *s'il était déjà installé, vous pouvez le mettre à jour :* `brew upgrade gcc`
       * *quoi qu'il en soit, on veut le numéro de version de g++ en redemandant une installation :* `brew install gcc`
       * `brew install libomp`
     - *Windows : clic droit sur la barre en bas > gestionnaires de taches > performance*
   - *en rajourtant* `-fopenmp` *dans la ligne de compilation.*
     *Sous mac, il faut également rajouter le numéro de version de g++, par exemple :* `g++-10 -Wall -O2 -fopenmp plop.cpp`
   - *en exécutant avec la commande* `OMP_NUM_THREADS=n ./my_program` *plutôt que* `./my_program` *ou bien en ajoutant dans votre code* `Eigen::setNbThreads(n);` *(où n est le nombre de threads que vous voulez lancer).*
   - *Comparez les temps d'exécution avec les autres méthodes.*

▸ **Exercice 3.** ***Vérification du produit matriciel***
*Soit* $\mathbf{C} = \mathbf{AB}$*. Un indice pour détecter les erreus d'un produit matriciel consiste à vérifier que le produit de la matrice* $\mathbf{A}$ *avec le vecteur somme-des-colonnes* $\mathbf{b}$ *de la matrice* $\mathbf{B}$ *est égal au vecteur somme-des-colonnes* $\mathbf{c}$ *de la matrice* $\mathbf{C}$ *:*

$$\mathbf{Ab} = \mathbf{c}$$

*Si* $\mathbf{Ab} \neq \mathbf{c}$*, alors il y a une erreur de calcul.*
***Exemple :***

$$\begin{matrix} \mathbf{C} & = & \mathbf{A} & \mathbf{B} \end{matrix}$$
$$\begin{bmatrix} 2 & 15 \\ 4 & 22 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 0 & 4 \end{bmatrix}$$

$$\mathbf{c} = \begin{pmatrix} 2+15 \\ 4+22 \end{pmatrix} = \begin{pmatrix} 17 \\ 26 \end{pmatrix} \quad \text{et} \quad \mathbf{b} = \begin{pmatrix} 2+3 \\ 0+4 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \end{pmatrix}$$
$$\mathbf{Ab} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{pmatrix} 5 \\ 4 \end{pmatrix} = \begin{pmatrix} 17 \\ 26 \end{pmatrix} = \mathbf{c}$$

*Codez une fonction de vérification du produit matriciel. Vous pourrez vous aider de la fonction* `M.rowwise().sum()`*.*