|  | **Master M1** |
| --- | --- |
| **République Française** | Calcul Haute Performance |
| **UVSQ, Paris Saclay** | *Simulation CHPS* |
|  | ***2021/2022*** |





**Rapport TP2:**

**Calcul Numérique**

**Par :**

**Aicha Maaoui**

**Le 21/11/2021**

# Exercice 6:

1/ Vecteur x (1 ligne, 4 colonnes): x= [2, 4, 6, 8]

2/ Vecteur y (4 ligne, 1 colonnes): y= [7; 1; 8; 2]

3/ Addition: On ne peut pas additionner un vecteur x et une vecteur y . Par conséquent, on utilise la transposition du vecteur y (on peut aussi faire la transposition du vecteur x et le sommer avec le vecteur y).

soit le vecteur z le résultat de la sommation des vecteurs x et : z=x+y' et z= [9, 5, 14, 10]

Multiplication: On peut multiplier un vecteur x et une vecteur y (nombre de colonnes du premier vecteur est celui du nombre de lignes du deuxième). Soit le vecteur s le résultat de la multiplication des vecteur x et y: s=x\*y et s=82.

4/ **Calcul de taille des vecteurs x et y**:

-->size(x)

ans =

1. 4.

-->size(y)

ans =

4. 1.

5/ Norme 2 de x:

norm(x,2)

→ Le résultat est évalué à: 10.954451

6/ **A = matrice à 4 lignes et 3 colonnes**:

A= [1, 3, 7, 13; 1, 5, 9, 71; 8, 10, 50, 45]

Alors A s'écrit:

A =

1. 3. 7. 13.

1. 5. 9. 71.

8. 10. 50. 45.

7/ **Transposée de A**: A'

Alors s'écrit:

1. 1. 8.

3. 5. 10.

7. 9. 50.

13. 71. 45.

8/ Soit A et B deux matrices carrées .

Soit les 2 matrices A et B:

A= [1, 3, 7, 13; 1, 5, 9, 71; 8, 10, 50, 45; 8, 7, 5, 1]

B= [18, 0, 1, 13; 21, 45, 19, 0; 18, 20, 51, 45; 3, 7, 9, 26]

Opérations de bases:

\***Addition** des 2 matrices A et B:

sum=A+B

Le résultat de l’addition des 2 matrices est alors stocké dans la matrice **sum**:

19. 3. 8. 26.

22. 50. 28. 71.

26. 30. 101. 90.

11. 14. 14. 27.

\***Multiplication** des 2 matrices A et B:

mul=A\*B

Le résultat de la multiplication des 2 vecteurs est alors stocké dans la matrice **mul**:

246. 366. 532. 666.

498. 902. 1194. 2264.

1389. 1765. 3153. 3524.

384. 422. 405. 355.

\* **Transposée de A**:

C1=A'

\***Transposée de B**:

C2=B'

\***Multiplication de A et B par un scalaire lambda respectivement**:

lambda=50

A1=lambda\*A

A2=lambda\*B

9/ **Conditionnement de A**:

cond(A)

Le conditionnement de la matrice A est évalué à 95.589276

# → Le conditionnement est bien supérieur à 1.

# Exercice 7:

1/ **Matrice A de taille**

A=rand(3,3)

La matrice générée est alors:

0.7263507 0.2320748 0.8833888

0.1985144 0.2312237 0.6525135

0.5442573 0.2164633 0.3076091

2/ **Vecteur xex**

xex=rand(3,1)

Alors, le vecteur xex généré est:

0.9329616

0.2146008

0.312642

→ xex est un vecteur colonne.

3/ **b=A\*xex**

Alors le vecteur b obtenu est:

1.0036452

0.4388302

0.6503959

4/ **Résolution du système Ax=b**, **le vecteur x calculé** est:

x=(A \ b)

0.9329616

0.2146008

0.312642

5/ On va utiliser la norme 2 dans le calcul des erreurs.

\*Calcul de l’**erreur avant**:norm((xex-x)/xex,2)

→ L’erreur avant est alors: err=1.081D-15

\*Calcul de l’**erreur arrière**: norm(b-A\*x,2)/(norm(A,2)\*norm(x,2))

→ L’erreur arrière est donc: relres=0.

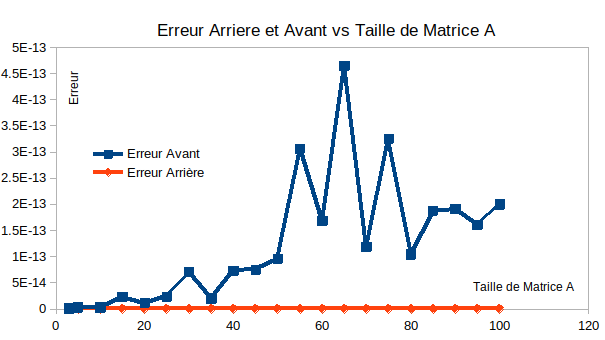
Ce résultat nul est expliqué par le fait que la taille du matrice n est faible.

6/ **\*\* 1er essai: Des matrices A de tailles comprises entre 3 et 100:**

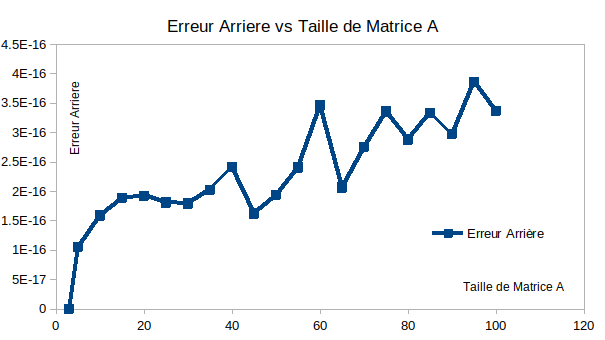
| **Taille de matrice A** | **Erreur Avant** | **Erreur Arrière** |
| --- | --- | --- |
| 3 | 1.081D-15 | 0 |
| 5 | 2.637D-15 | 1.056D-16 |
| 10 | 2.665D-15 | 1.589D-16 |
| 15 | 2.283D-14 | 1.889D-16 |
| 20 | 1.149D-14 | 1.926D-16 |
| 25 | 2.317D-14 | 1.813D-16 |
| 30 | 7.067D-14 | 1.800D-16 |
| 35 | 2.051D-14 | 2.031D-16 |
| 40 | 7.290D-14 | 2.414D-16 |
| 45 | 7.502D-14 | 1.625D-16 |
| 50 | 9.533D-14 | 1.941D-16 |
| 55 | 3.063D-13 | 2.407D-16 |
| 60 | 1.678D-13 | 3.462D-16 |
| 65 | 4.651D-13 | 2.066D-16 |
| 70 | 1.184D-13 | 2.756D-16 |
| 75 | 3.247D-13 | 3.360D-16 |
| 80 | 1.038D-13 | 2.884D-16 |
| 85 | 1.869D-13 | 3.339D-16 |
| 90 | 1.910D-13 | 2.976D-16 |
| 95 | 1.609D-13 | 3.865D-16 |
| 100 | 1.997D-13 | 3.368D-16 |

**Tableau 1: Erreurs avant et arrière en fonction de la taille de matrice A.**

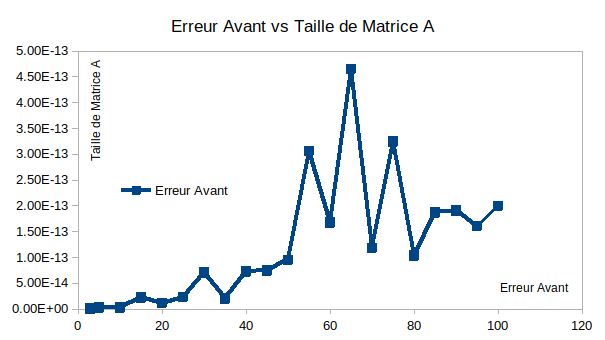
Les courbes des erreurs avant et arrière en fonction de la taille de la matrice A sont illustrées dans les figures 1, 2 et 3.



**Figure 1: Erreur Arrière et Avant vs Taille de Matrice A.**



**Figure 2: Erreur Arrière vs Taille de Matrice A.**



**Figure 3: Erreur Avant vs Taille de Matrice A.**

**\*\* 2eme essai: Taille de matrices n={100, 1000, 10000}**

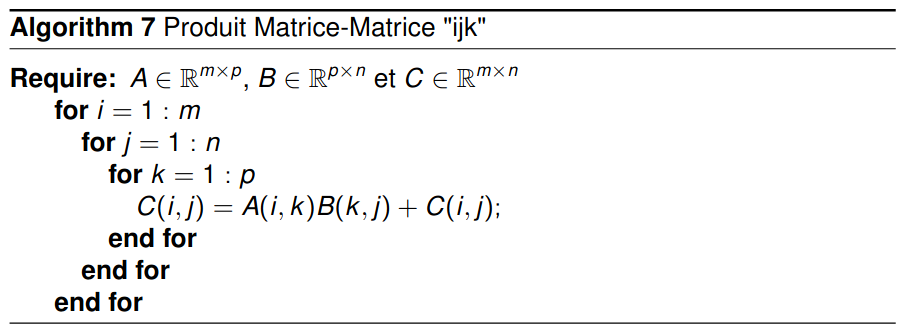
| **Taille de matrice A** | **Erreur Avant** | **Erreur Arrière** |
| --- | --- | --- |
| 100 | 4.253D-13 | 3.158D-16 |
| 1000 | 1.656D-10 | 1.079D-15 |
| 10000 |  |  |

**Conclusion:** L’erreur avant et arrière évoluent d’une manière exponentielle.

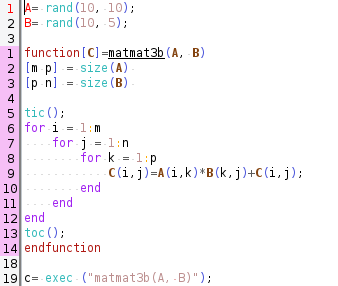
# 

# Exercice 7:

1/ L’algorithme du produit Matrice-Matrice “ijk” à 3 boucles, illustré dans la figure 5, est créée dans une fonction matmat3b (A, B), comme illustré dans la figure 6.

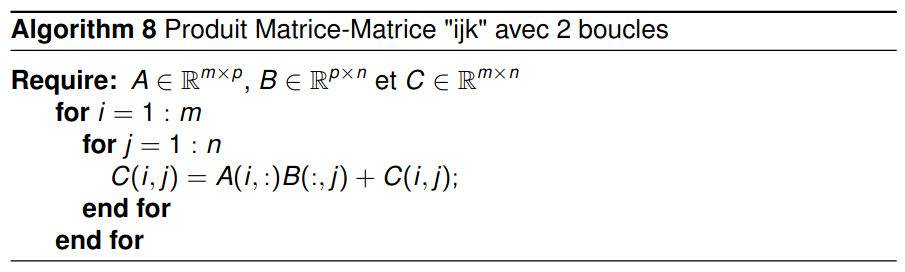


**Figure 5: Produit Matrice-Matrice “ijk” avec 3 boucles.**

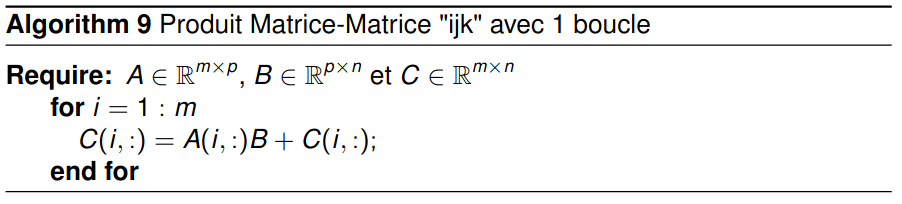


**Figure 6: Programme Scilab de Produit Matrice-Matrice “ijk” avec 3 boucles.**

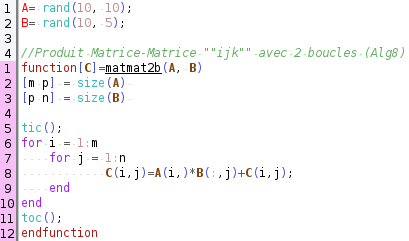
2/ Les algorithmes de calcul des produits Matrice-Matrice “ijk” à 2 boucles et 1 boucle sont illustrés dans les figures 7 et 8. Leurs traductions en Scilab sont illustrées dans les figures 9 et 10.

****

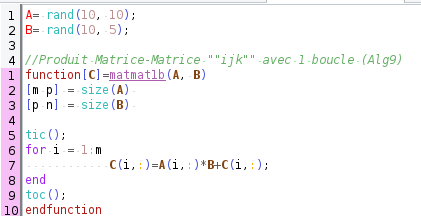
**Figure 7: Algorithme Produit Matrice-Matrice “ijk” avec 2 boucles.**

****

**Figure 8: Algorithme Produit Matrice-Matrice “ijk” avec 1 boucle.**

****

**Figure 9: Programme Scilab de Produit Matrice-Matrice “ijk” avec 2 boucles.**

****

**Figure 10: Programme Scilab de Produit Matrice-Matrice “ijk” avec 1 boucle.**

3/ Le temps de chacun des algorithmes pour des tailles différentes est mesuré avec les fonctions tic et toc dans scilab.