Projet Calcul Scientifique et Analyse de Données
Compte-Rendu de Projet 2

Méthode d'itération sur un sous-espace

François Lauriol, Priscilia Gonthier, Yael Gras
Groupe MN

21 avril 2022

Table des matières

Introduction 2
1 Limitation de la méthode de la puissance 2
2 Extension de la méthode de la puissance pour calculer les vecteurs dominants de l'espace propre 6
3 Subspace_iter_v2 et subspace_iter_v3 : vers un solveur efficace 7
4 Expériences numériques 8

Introduction

1 Limitation de la méthode de la puissance

Question 1 :

Nous commençons d'abord par comparer le temps d'exécution en changeant le type de la matrice :

```
Matrice 200 x 200 - type 1

******* création de la matrice ******

Temps de création de la matrice = 2.500e-01

******* calcul avec eig ******

Temps eig = 2.000e-02
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 2.642e-14]
Qualité des couples propres = [5.352e-16 , 1.033e-13]

Matrice 200 x 200 - type 1

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 5.160e+00
Nombre de valeurs propres pour attendre le pourcentage = 46
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.990e-14]
Qualité des couples propres = [9.977e-09 , 1.422e-08]
```

FIGURE 1 – Matrice 200 × 200 de type 1

```
Matrice 200 x 200 - type 2

******* création de la matrice ******

Temps de création de la matrice = 2.100e-01

******* calcul avec eig ******

Temps eig = 1.000e-02
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 9.059e-08]
Qualité des couples propres = [4.072e-16 , 1.160e-06]

Matrice 200 x 200 - type 2

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 7.000e-02
Nombre de valeurs propres pour attendre le pourcentage = 5
Qualité des valeurs propres (par rapport au spectre de la matrice) = [1.460e-16 , 1.595e-15]
Qualité des couples propres = [9.767e-09 , 1.794e-08]
```

FIGURE 2 – Matrice 200 × 200 de type 2

```
Matrice 200 x 200 - type 3

******* création de la matrice ******

Temps de création de la matrice = 2.200e-01

******* calcul avec eig ******

Temps eig = 0.000e+00
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.641e-11]
Qualité des couples propres = [5.321e-16 , 5.343e-11]

Matrice 200 x 200 - type 3

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 1.500e-01
Nombre de valeurs propres pour attendre le pourcentage = 9
Nombre d'itérations pour chaque couple propre
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.554e-15]
Qualité des couples propres = [9.733e-09 , 1.487e-08]
```

FIGURE 3 – Matrice 200×200 de type 3

```
Matrice 200 x 200 - type 4

******* création de la matrice ******

Temps de création de la matrice = 1.900e-01

******* calcul avec eig ******

Temps eig = 1.000e-02
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 2.609e-14]
Qualité des couples propres = [6.323e-16 , 4.994e-14]

Matrice 200 x 200 - type 4

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 5.090e+00
Nombre de valeurs propres pour attendre le pourcentage = 46
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.998e-14]
Qualité des couples propres = [9.954e-09 , 1.422e-08]
```

FIGURE 4 – Matrice 200×200 de type 4

Nous remarquons sur ces différents résultats que la méthode *eig* est plutôt constante en temps d'exécution sur les différents type de matrice (de l'ordre du centième de seconde) et la qualité des couples propres est réussi car elle reste plutôt constante pour la première valeur avec un ordre de grandeur de 10^{-16} cependant la deuxième valeur est plus variable avec un ordre de grandeur entre 10^{-14} et 10^{-6}. Cette variation se voit surtout sur le type 2 qui a la valeur la plus élevée.

Pour ce qui est de la méthode de la puissance itérée nous pouvons voir qu'elle n'obtient jamais tous les couples propres. Elle est moins rapide sur le type 4 où elle a un temps d'exécution de $5.09s$ alors que les autres types restent dans l'ordre de grandeur du centième. Nous pouvons cependant voir que sur le type 2 et le type 3, moins d'une dizaine de valeur propres ont été trouvé contre 46 pour le type 1 et le type 4. Pour finir, la qualité des couples propres sont constant avec les valeurs en 10^{-9} et 10^{-8}.

Pour conclure, la méthode de la puissance itérée est plus efficace sur un matrice de type 1 mais elle restera constante sur la qualité des couples propres fournis même si elle en fournis peu. La méthode *eig* est quant à elle plus stable sur la durée d'exécution pour fournir toutes les couples propres mais la qualité des couples propres est plus variables que pour la méthode de la puissance itérée. La méthode *eig* est plus efficaces sur les matrices de type 1 et 4.

Nous comparons maintenant le temps d'exécution en changeant la taille de la matrice :

```
Matrice 20 x 20 - type 1

******* création de la matrice ******

Temps de création de la matrice = 1.000e-02

******* calcul avec eig ******

Temps eig = 0.000e+00
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 2.220e-15]
Qualité des couples propres = [3.809e-16 , 6.349e-15]

Matrice 20 x 20 - type 1

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 2.000e-02
Nombre de valeurs propres pour attendre le pourcentage = 5
 Qualité des valeurs propres (par rapport au spectre de la matrice) = [3.740e-16 , 2.132e-15]
 Qualité des couples propres = [9.753e-09 , 1.465e-08]
```

FIGURE 5 – Matrice 20×20 de type 1

```
Matrice 100 x 100 - type 1

******* création de la matrice ******

Temps de création de la matrice = 3.000e-02

******* calcul avec eig ******

Temps eig = 0.000e+00
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 6.550e-15]
Qualité des couples propres = [6.278e-16 , 4.220e-14]

Matrice 100 x 100 - type 1

******* calcul avec la méthode de la puissance itérée ******

Temps puissance itérée = 2.500e-01
Nombre de valeurs propres pour attendre le pourcentage = 23
Nombre d'itérations pour chaque couple propre
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 9.948e-15]
Qualité des couples propres = [9.941e-09 , 1.428e-08]
```

FIGURE 6 – Matrice 100×100 de type 1

```
 Matrice 200 x 200 - type 1

 ******* création de la matrice ******

 Temps de création de la matrice = 2.500e-01

 ******* calcul avec eig ******

 Temps eig = 2.000e-02
 Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 2.642e-14]
 Qualité des couples propres = [5.352e-16 , 1.033e-13]

 Matrice 200 x 200 - type 1

 ******* calcul avec la méthode de la puissance itérée ******

 Temps puissance itérée = 5.160e+00
 Nombre de valeurs propres pour attendre le pourcentage = 46
 Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.990e-14]
 Qualité des couples propres = [9.977e-09 , 1.422e-08]
```

FIGURE 7 – Matrice 200×200 de type 1

```
Matrice 1000 x 1000 - type 1

******* création de la matrice ******

Temps de création de la matrice = 2.988e+01

******* calcul avec eig ******

Temps eig = 2.800e-01
Qualité des valeurs propres (par rapport au spectre de la matrice) = [0.000e+00 , 1.907e-13]
Qualité des couples propres = [6.292e-16 , 7.985e-13]

Matrice 1000 x 1000 - type 1

******* calcul avec la méthode de la puissance itérée ******
puissance_itérée : convergence non atteinte pour un des couples propres
```

FIGURE 8 – Matrice 1000×1000 de type 1

Nous pouvons voir que quelque soit la taille de la matrice, la méthode eig calcul plus rapidement et de manière plus juste les couples propres que la méthode de la puissance itérée. De plus la méthode de la puissance itérée est limitée sur la taille de la matrice car elle n'arrive pas à converger sur une matrice de taille 1000×1000 de type 1.

Ainsi, nous pouvons conclure que la méthode *eig* est plus rapide que l'algorithme de la puissance itérée quelque soit la matrice mais *eig* n'est pas forcément plus juste que la méthode de la puissance itérée.

Question 2 :

Algorithme 1 v2 Vector power method rearanged
Input : Matrix $A \in R^{n \times n}$
Output : (λ_1, v_1) eigenpair associated to the largest (in module) eigenvalue.
$v \in R^n$ given
$z = A \cdot v$
$\beta = v^T \cdot z$
repeat
$y = z$
$v = y/||y||$
$z = A \cdot v$
$\beta_{old} = \beta$
$\beta = v^T \cdot z$
until $|\beta - \beta_{old}|/|\beta_{old}| < \varepsilon$
$\lambda_1 = \beta$ and $v_1 = v$

Question 3 :

La méthode a quelques désavantages, pour une même taille de matrice, le temps de calcul des couples propres varie énormément pour des types différents. Le temps de calcul pour quelques couples propres est aussi plus élevé que le temps de calcul pour toutes les valeurs propres de la fonction eig. De plus pour le type 1 si on utilise une taille de matrice trop importante (par exemple 1000×1000), la méthode ne converge pas.

2 Extension de la méthode de la puissance pour calculer les vecteurs dominants de l'espace propre

2.1 Subspace_iter_v0 : une méthode de base pour calculer un espace propre dominant

Question 4 :

Dans le cas où la question fait référence à l'algorithme 1 (Vector power method avec m vecteurs), nous ne savons pas vers quelle matrice, cet algorithme tend.

Dans le cas où la question fait référence à l'algorithme 2 (Subspace iteration method v0), la matrice V converge vers la matrice de changement de base qui lie les vecteurs propres de H (une matrice qui contient les mêmes valeurs propres que A) aux vecteurs propres de A.

Question 5 :

Nous avons : $A \in \mathbb{R}^{n \times n}$ et $v \in \mathbb{R}^{n \times m}$.
Donc comme $H = v^T \cdot A \cdot v$ on a $H \in \mathbb{R}^{m \times m}$.
H est donc d'une taille raisonnable, comme $m << n$. Ce n'est donc pas un problème de calculer la décomposition spectrale de H.

2.2 Subspace_iter_v1 : version améliorée faisant appel à la projection de Raleigh-Ritz

Question 7 :

Algorithm 4 Subspace iteration method v1 with Raleigh-Ritz projection
Input : Symmetric matrix $A \in R^{n \times n}$, tolerance ϵ, MaxIter (max nb of iterations) and PercentTrace the target percentage of the trace of A
Output : n_{ev} dominant eigenvectors V_{out} and the corresponding eigenvalues Λ_{out}.
Generate an initial set of m orthonormal vectors $V \in R^{n \times m}$; k = 0 ; PercentReached = 0
repeat

$l.55$	$k = k + 1$
$l.57$	Compute Y such that $Y = A \cdot V$
$l.59$	$V \leftarrow$ orthonormalisation of the columns of Y
$l.62$	Rayleigh-Ritz projection applied on matrix A and orthonormal vectors V
$l.64 - 113$	Convergence analysis step : save eigenpairs that have converged and update PercentReached

until (PercentReached > PercentTrace or n_{ev} = m or k > MaxIter)

3 Subspace_iter_v2 et subspace_iter_v3 : vers un solveur efficace

3.1 Approche par bloc (subspace_iter_v2)

Question 8 :

Si on prend $A \in \mathbb{R}^{m \times n}$ et $B \in \mathbb{R}^{n \times p}$ alors le coût en flops du calcul $A \cdot B$ est de $O(2 \cdot m \cdot n \cdot p)$. Dans notre cas, nous avons $A \in \mathbb{R}^{n \times n}$ et nous faisons le calcul $A \cdot A$ donc le coût de cette multiplication est de $O(2 \cdot n^3)$. Ainsi, pour faire A^p, le coût est de $O(2p \cdot n^3)$ car nous faisons p fois la multiplication $A \cdot A$.

Nous savons que $V \in \mathbb{R}^{n \times m}$ avec $m << n$ donc pour faire $A^p \cdot V$ revient à faire une multiplication $\mathbb{R}^{n \times n} \cdot \mathbb{R}^{n \times m}$ ainsi le coût de la dernière multiplication est de $O(2 \cdot m \cdot n^2)$ donc le coût total du calcul est de : $O(2p \cdot n^3 + 2 \cdot m \cdot n^2) = O(2p \cdot n^3)$.

Pour réduire le coût de ce calcul nous devons faire une fois le calcul A^p avant de rentrer dans la boucle puis il nous reste dans la boucle uniquement la multiplication avec V qui coûte $O(2 \cdot m \cdot n^2)$, qui est moins onéreux car $m << n$.

Question 10 :

On remarque que dans un premier temps l'incrémentation de p permet à notre algorithme de converger plus rapidement pour les couples propres nécessitant le plus d'itération, mais si on augmente trop p, cette effet à tendance à augmenter le nombre d'itérations nécessaires pour trouver les premiers couples propres et à empêcher la convergence.
Pour p = 1 : 57 itérations (pour le dernier couple propre)
Pour p = 2 : 31 itérations (pour le dernier couple propre)
Pour p = 3 : 21 itérations (pour le dernier couple propre)
Pour p = 4 : 16 itérations (pour le dernier couple propre)
Pour p = 10 : pas de convergence.

3.2 Méthode de la déflation (subspace_iter_v3)

Question 11 :

La précision des vecteurs est différente selon leur valeur propre, plus leur valeur propre est grande, plus il est facile d'avoir une grande précision concernant leur vecteur propre. On l'observe d'ailleurs dans le critère de convergence d'un vecteur $||A \cdot V_j - \Lambda_j \cdot V_j||/||A|| < \varepsilon$

Question 12 :

La convergence des eigenfaces doit pour sûr être plus rapide, mais moins précise. Du fait que l'on ne cherche qu'à préciser les couples propres non encore déterminés.

4 Expériences numériques

Question 14 :

Les 4 types de matrices que l'on a décidé d'utiliser pour nos tests ont des valeurs propres qui sont respectivement incrémentés de 1 à n, aléatoires, avec leur logarithme uniformément réparti, et enfin répartis de manière uniforme entre 0 et 1.

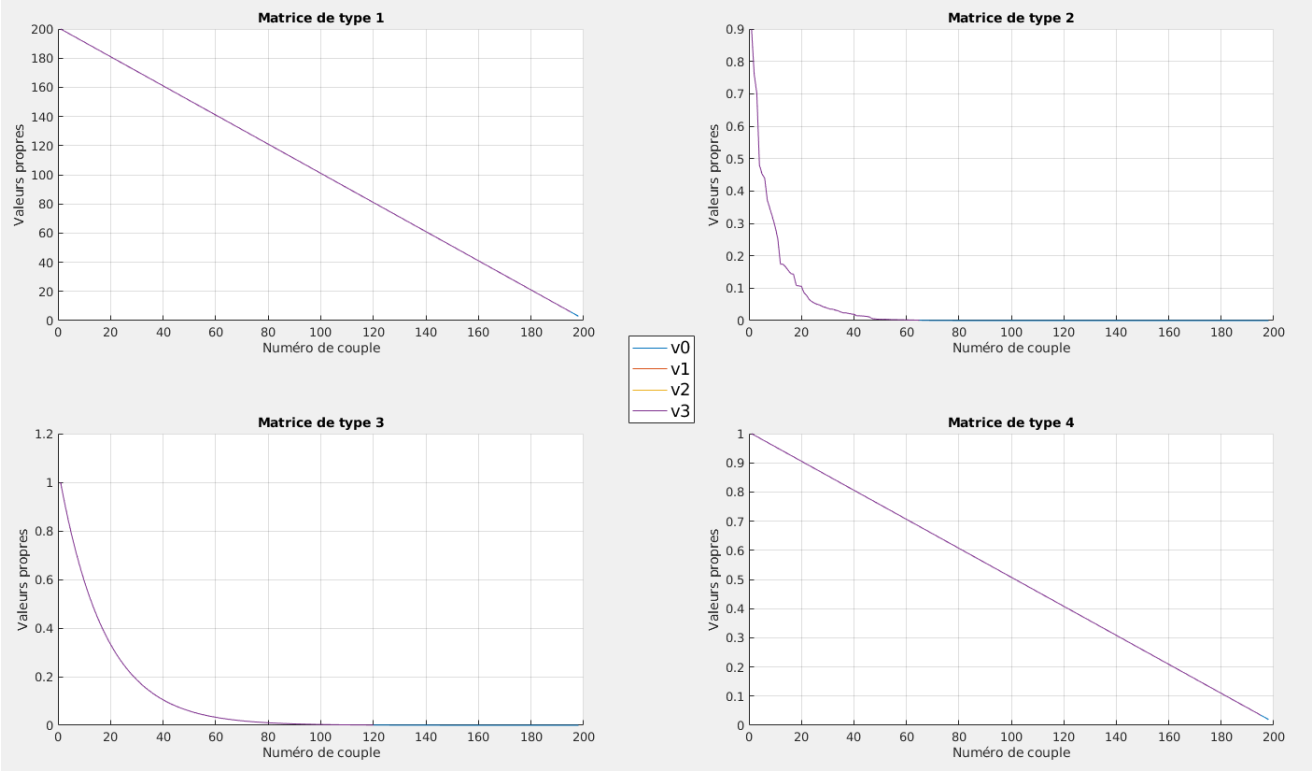

FIGURE 9 – Valeurs propres pour les différents types de matrices

Question 15 :

Pour les matrices de type 1, la méthode subspace_iter_v2 semble être la meilleure combinaison quant au temps et à la précision qu'elle apporte. (même si la version v3 est légèrement plus rapide, la version v2 reste bien plus précise concernant les derniers couples propres)

Pour les matrices de type 2,la méthode subspace_iter_v3 semble être la meilleure combinaison quant au temps et à la précision qu'elle apporte (la version v1 est trop longue).

Pour les matrices de type 3, la méthode subspace_iter_v1 semble être la meilleure combinaison quant au temps et à la précision qu'elle apporte (la version v2 est trop longue par rapport à la précision qu'elle apporte).

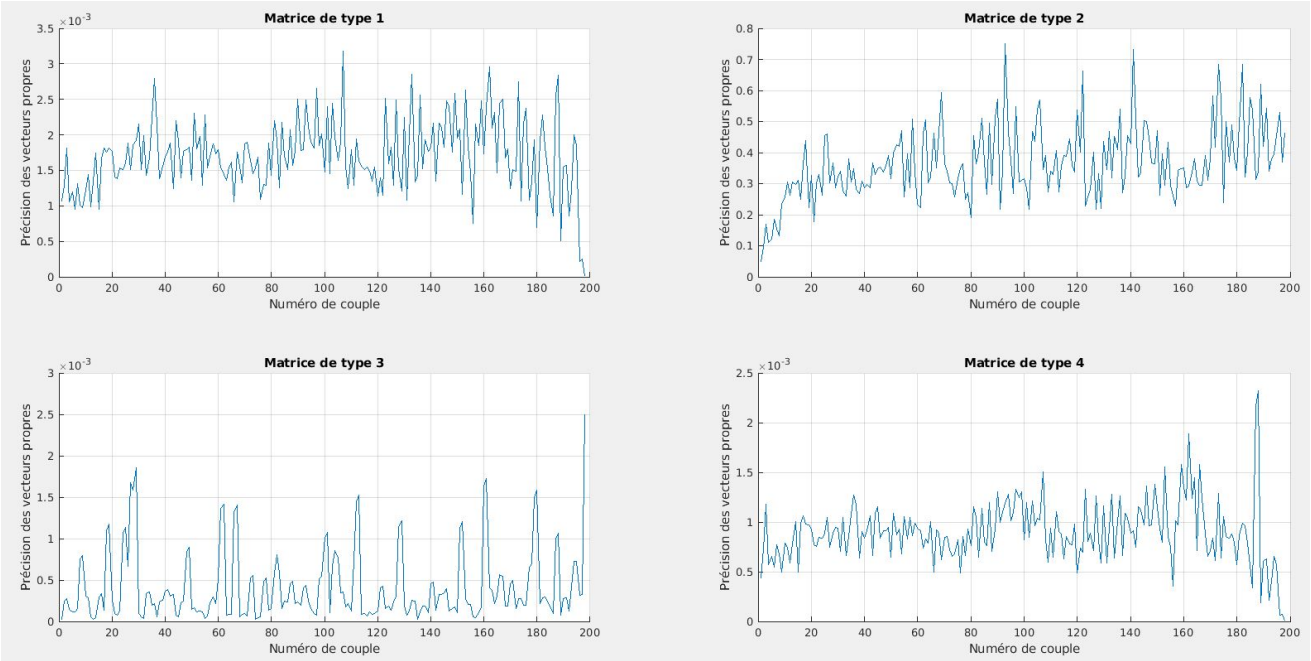

FIGURE 10 – Précision des vecteurs propres pour la version 0

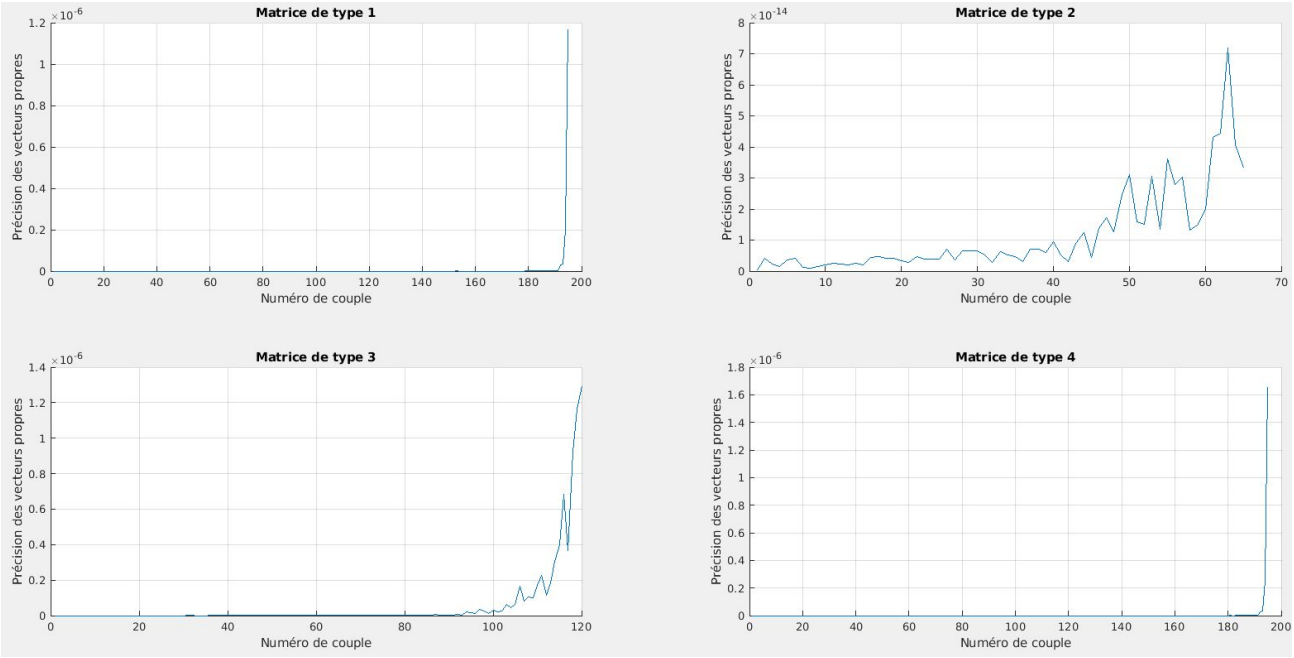

FIGURE 11 – Précision des vecteurs propres pour la version 1

Pour les matrices de type 4, la méthode subspace_iter_v2 semble être la meilleure combinaison quant au temps et à la précision qu'elle apporte pour les grandes matrices mais la méthode subspace_iter_v1 est à priviligier pour les petites matrices quant au temps qu'elle nous fait gagner (la version v3 n'étant pas assez précise pour les temps qu'elle apporte).

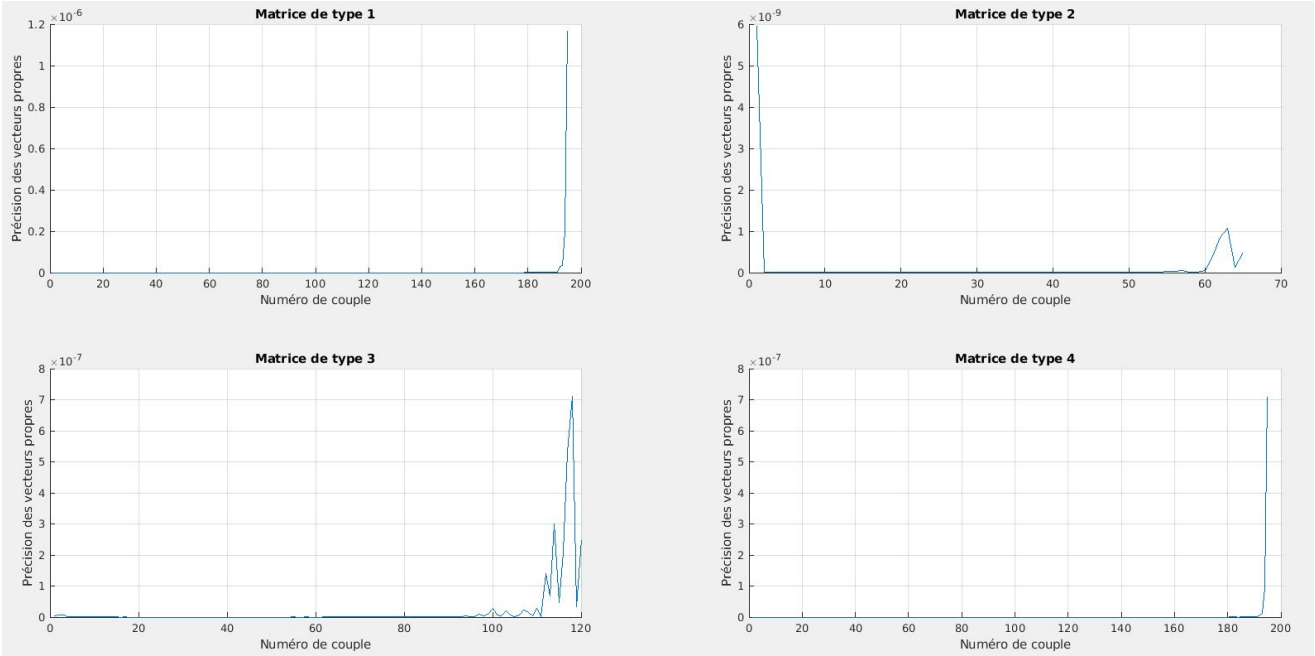

FIGURE 12 – Précision des vecteurs propres pour la version 2

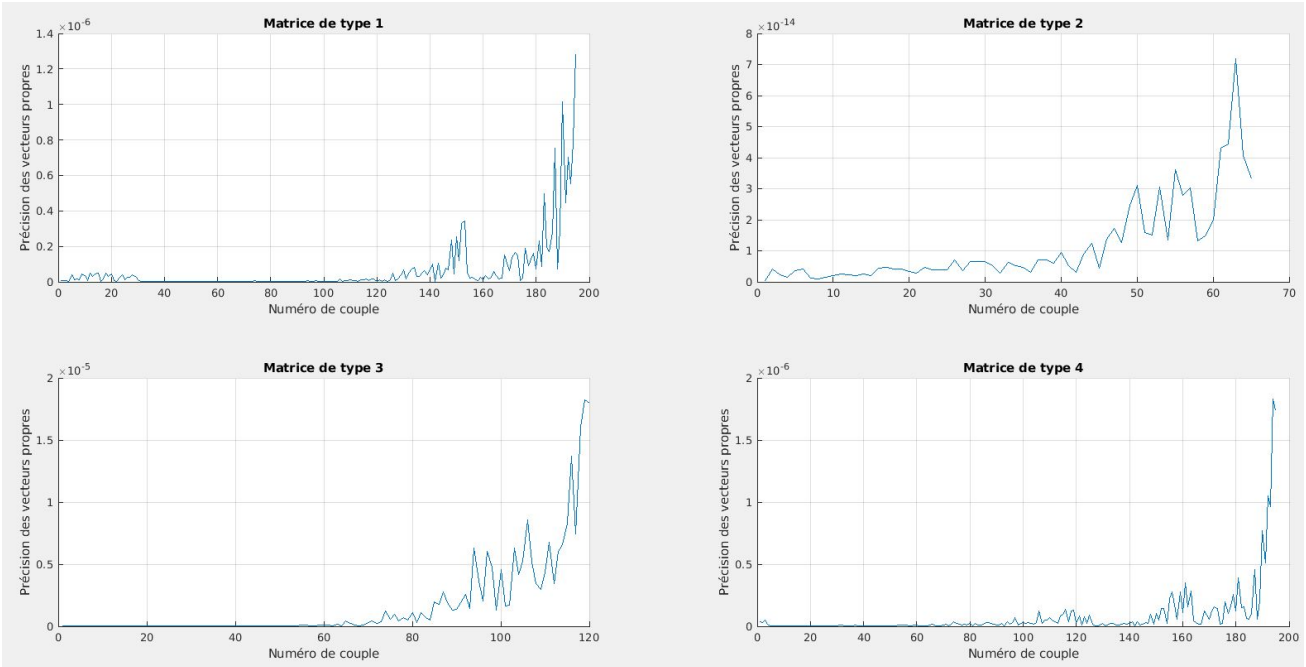

FIGURE 13 – Précision des vecteurs propres pour la version 3

		iter_v0	iter_v1	iter_v2	iter_v3
matrice 200x200	Type 1	6.800e-01	1.060e+00	5.500e-01	5.300e-01
	Type 2	2.400e-01	1.500e-01	6.600e-01	1.400e-01
	Type 3	1.160e+00	1.900e-01	4.400e-01	1.500e-01
	Type 4	8.600e-01	1.290e+00	4.800e-01	6.000e-01
matrice 10x10	Type 1	0.000e+00	1.000e-02	2.000e-02	1.000e-02
	Type 2	1.000e-02	1.000e-02	non conv	2.000e-02
	Type 3	1.000e-02	1.000e-02	3.000e-02	1.000e-02
	Type 4	1.000e-02	1.000e-02	3.000e-02	1.000e-02
matrice 50x50	Type 1	2.000e-02	5.000e-02	5.000e-02	2.000e-02
	Type 2	4.000e-02	6.000e-02	non conv	3.000e-02
	Type 3	1.000e-02	3.000e-02	7.000e-02	3.000e-02
	Type 4	2.000e-02	3.000e-02	6.000e-02	3.000e-02

FIGURE 14 – Temps en secondes de calcul des différentes méthodes subspace_iter