République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



Université Yahia Farès de Médéa

Faculté des Sciences

Département de Mathématiques et de l'Informatique

MÉMOIRE DE FIN D'ÉTUDES

Présenté en vue de l'obtention du Diplôme de MASTER
Spécialité : Recherche opérationnelle.

TITRE

Méthodes numériques pour résoudre l'équation de Lyapunov-Sylvester et ses applications

Présentée par

KHELFAOUI Nour El Houda

Devant le jury :

Mr. BIBI Hamza	*MCB.* Univ. Médéa	Président
Mr. ALIANE Mohamed	*MCA.* Univ. Médéa	Encadreur
Mr. ZEMIR Mohamed	*MCB.* Univ. Médéa	Examinateur

Promotion : 2024-2025

Remerciements

Tout d'abord On aimerait remercier Allah le tout puissant, qui nous a donné la force et la patience d'accomplir ce modeste travail.

Quand le travail est remarquable, que le don est sincère et que la créativité devient une méthode, la gratitude devient un devoir et la reconnaissance, une nécessité. J'adresse mes sincères remerciements et ma profonde reconnaissance à mon honorable professeur ALIANE Mohamed, qui a dirigé ce travail. Son enthousiasme, sa bienveillance et sa compétence m'ont guidé bien au-delà de ce que j'imaginais. J'ai énormément appris grâce à lui. C'est avec beaucoup de joie que je lui exprime ma gratitude pour son dévouement, le temps et les efforts qu'il a consacrés. Je prie Allah de bénir ses efforts et de guider ses pas sur la voie du bien.

Je tiens également à remercier chaleureusement Monsieur BIBI Hamza, président du jury, ainsi que Monsieur Zemir Mohamed, membre du jury, pour l'honneur qu'ils m'ont fait en acceptant d'évaluer ce travail, ce que je considère comme un grand privilège.

Je remercie aussi du fond du cœur tous les membres de ma famille, en particulier mes parents, pour leur soutien constant tout au long de mon parcours universitaire.

Je souhaite aussi remercier tous les enseignants du département de mathématiques et informatique de la faculté des sciences de l'université Yahia Farès Médéa.

Enfin, je ne peux que me remercier moi-même pour les efforts, la patience et la persévérance que j'ai déployés durant toutes ces années, jusqu'à la réalisation de cet accomplissement.

إهداء

لا يطيب الليل الا بشكره ولا يطيب النهار الا بطاعته ولا تطيب اللحظات الا بذكره سبحانه و تعالى.

الحمد لله عند البدء و عند الختام من قال انا لها نالها.

اهدي عملي هذا الى من رباني و كافح من اجلي الى المصباح الذي انار دربي و لمن احمل اسمه بكل افتخار طاب بك العمر يا سيد الرجال و طبت لي عمرا ارجوا من الله ان يمدد في عمرك لترى ثمارا قد حان قطفها...والدي العزيز.

الى ملاكي في الحياة و معنى الحب و قرة عيني و اعز ما املك الى بسمة الحياة وسر الوجود الى من كان دعائها سر نجاحي و حنانها بلسم جراحي الى غاليتي و جنة قلبي التي رافقتني و ارشدتني في كل مشاوير حياتي ...امي الغالية حفظها الله.

الى ضلعي الثابت الذي لا يميل الى من رزقت بهم سندا و ملاذي الاول و الاخير الى من ازاحوا عن طريقي اشواك الفشل الى اخوتي و اخواتي الاعزاء كل باسمه.

الى نفسي التي راهنت على النجاح اصبري و صابري فلا يزال الطريق طويل .

الى من علمني حرفا طيلة مساري الدراسي و لم يبخل بعطاءه , اساتذتي الافاضل كل باسمه و مقامه.

الى رفقاء الدرب من غادرونا و بقيت كلماتهم و قعا في اذاننا.

الى كل من اتسع قلبي لهم و ضاقت هذه الورقة عن ذكرهم ,اهديكم عملي المتواضع عرفانا لكم بالجميل و تقديرا لجهودكم.

و الحمد لله على حسن التمام و الختام

نور الهدى

Résumé

Ce travail porte sur les différentes méthodes numériques développées pour résoudre l'équation de Lyapunov-Sylvester de la forme $AX + XB = C$, qui joue un rôle central dans l'analyse de la stabilité des systèmes dynamiques linéaires. Parmi les approches étudiées, on distingue les méthodes directes basées sur la forme de Kronecker et la vectorisation, les méthodes itératives telles que la méthode de l'algorithme de Bartels-Stewart et la méthode de Smith, ainsi que les techniques basées sur les décompositions de Schur ou de Cholesky. Des variantes plus récentes exploitent les propriétés de faible rang et la structure particulière des matrices dans les grandes dimensions. Ces méthodes sont essentielles dans divers domaines d'application, notamment le contrôle optimal, l'analyse de stabilité, la réduction de modèles, et le traitement de l'image. Le choix de la méthode dépend généralement de la taille du problème, de la nature des matrices et des contraintes de temps de calcul.

Abstract

This work focuses on various numerical methods developed to solve the Lyapunov–Sylvester equation of the form $AX + XB = C$, which plays a central role in the stability analysis of linear dynamical systems. Among the studied approaches are direct methods based on the Kronecker form and vectorization, iterative methods such as the Bartels–Stewart algorithm and the Smith method, as well as techniques based on Schur or Cholesky decompositions. More recent variants exploit low-rank properties and the specific structure of matrices in large-scale problems. These methods are essential in several application fields, including optimal control, stability analysis, model reduction, and image processing. The choice of method generally depends on the problem size, matrix structure, and computational constraints.

Liste des tableaux

1.1 Comparaison des temps CPU pour différentes méthodes de résolution de $LX + XU = C$. 25

2.1 Comparaison des temps CPU pour différentes méthodes de résolution de $AX + XB = C$. 44

Table des figures

1.1 Disques de Gershgorin pour les matrices A et B. 14
1.2 Disques de Gershgorin pour les matrices A et B. 15
1.3 Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $LX + XU = C$. 26
1.4 Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $LX + XU = C$. 26

2.1 Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $AX + XB = C$, avec tol $= 10^{-2}$. 44
2.2 Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $AX + XB = C$, avec tol $= 10^{-4}$. 45
2.3 Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $AX + XB = C$, avec tol $= 10^{-6}$. 45

Table des matières

Introduction 2

1 Equation de Lyapunov-Sylvester 4

I Introduction 4

II Les types de l'équation de Lyapunov-Sylvester 4

III Relation entre l'équation de Lyapunov-Sylvester et les systèmes linéaires $Gx = c$ 5

IV Existence et unicité 10

V Nouvelles méthodes pour calculer le produit matricielle 16

VI Analyse des cas particuliers de l'équation de Lyapunov-Sylvester 18

2 Méthodes numériques pour résoudre l'équation de Lyapunov-Sylvester 27

I Algorithme de Bartels-Stewart 27

II Algorithme de Jordan-Schur 34

III Approche itérative avancée pour la résolution de l'équation de Lyapunov-Sylvester 39

Conclusion 46

Bibliography 47

Introduction

Les équations, sous leurs différentes formes, constituent la pierre angulaire des mathématiques appliquées et de la modélisation scientifique. Parmi ces équations, on trouve les équations algébriques, utilisées pour déterminer les racines des fonctions [12], les équations différentielles ordinaires (EDO) qui modélisent l'évolution temporelle de nombreux phénomènes dynamiques [13], ainsi que les équations aux dérivées partielles (EDP), largement utilisées dans les domaines tels que la mécanique des fluides ou la thermodynamique [11]. De plus, les systèmes linéaires $Gx = c$ jouent un rôle central dans les méthodes numériques appliquées en ingénierie [23].

Toutes ces équations exigent des techniques numériques précises et efficaces. L'équation de Lyapunov-Sylvester compte parmi les équations fondamentales en analyse numérique, en raison de ses nombreuses applications et de son importance théorique. Elle se formule à l'aide de matrices. L'équation de Lyapunov-Sylvester trouve ses racines dans les travaux du mathématicien britannique James Sylvester, au XIX siècle, qui a étudié l'équation Sylvester sous la forme $AX + XB = C$.

Par la suite, le mathématicien russe Aleksandr Lyapunov a formulé une équation spécifique dédiée à l'analyse de la stabilité des systèmes dynamiques, connue sous le nom d'équation de Lyapunov [13]. L'intégration et l'harmonisation de ces deux approches théoriques ont conduit à l'émergence de l'équation de Lyapunov-Sylvester telle qu'elle est reconnue aujourd'hui, et qui occupe une place centrale dans l'étude des systèmes linéaires et leurs applications pratiques dans divers domaines des mathématiques appliquées et de l'ingénierie [8, 16].

L'analyse matricielles constituent des outils fondamentaux dans plusieurs branches des mathématiques appliquées et de l'ingénierie, où l'équation de Lyapunov-Sylvester se présente comme un exemple central de ce type d'équations, à la fois d'un point de vue théorique et pratique [2, 16]. Cette équation apparaît sous la forme :

$$AX + XB = C,$$

où $A \in \mathcal{M}_n(\mathbb{R})$, $B \in \mathcal{M}_m(\mathbb{R})$ et $C \in \mathcal{M}_{n,m}(\mathbb{R})$ sont des matrices données, et l'on cherche à déterminer la matrice inconnue $X \in \mathcal{M}_{n,m}(\mathbb{R})$. Cette équation a suscité un grand intérêt au cours des dernières décennies en raison de la diversité de ses applications dans les domaines du contrôl optimal linéaire, les equations aux dérirée partielles [13], de l'analyse de stabilité, de la théorie des systèmes [16], ainsi que dans des applications en physique et en ingénierie numérique [8].

D'un point de vue théorique, l'équation peut être représentée comme un système linéaire de grande taille en utilisant le produit de Kronecker et la vectorisation [23], mais cette méthode devient coûteuse numériquement lorsqu'il s'agit de grandes matrices, ce qui rend nécessaire le recours à des algorithmes efficaces exploitant la structure particulière de ces équations [11].

La réduction de la complexité algorithmique et l'accélération des opérations de calcul constituent l'un des défis majeurs dans le traitement des matrices de grande dimen-

sion. Ces dernières années, plusieurs méthodes modernes ont été proposées pour relever ce défi, notamment en exploitant les structures particulières des matrices. Dans de nombreuses situations pratiques, les matrices intervenant dans l'équation de Lyapunov-Sylvester sont des matrices creuses, c'est-à-dire que la majorité de leurs éléments sont nuls. Dans ce contexte, nous faisons appel à des méthodes numériques dites *sparse*, qui ont été spécialement développées pour exploiter au mieux cette structure lorsqu'elle est présente. Cette structure creuse représente un avantage qui peut être exploité pour améliorer l'efficacité numérique, accélérer les calculs et réduire la complexité algorithmique des méthodes de résolution [21]. À mesure que la taille des matrices augmente, les algorithmes classiques deviennent inefficaces en raison des coûts élevés de stockage et de calcul, tandis que la structure creuse permet d'éviter la manipulation explicite de matrices denses, tout en maintenant une bonne précision numérique.

Dans ce contexte, plusieurs méthodes numériques ont été proposées, variant selon les principes mathématiques et les structures sur lesquelles elles s'appuient. Parmi les méthodes les plus connues, on trouve l'algorithme Bartels-Stewart [2], qui repose sur la transformation des matrices en forme triangulaire supérieure à l'aide de la décomposition de Schur, en appliquant les transformations de Householder [12], suivies de l'algorithme QR pour l'extraction des valeurs propres et la structure de Hessenberg. Après cette transformation, l'équation résultante est résolue par élimination rétrograde, ce qui rend l'algorithme efficace pour les matrices de petite ou moyenne taille [8].

En revanche, l'algorithme Jordan-Schur [3] repose sur une décomposition différente consistant à transformer les matrices en forme de Jordan, ce qui permet de traiter les cas particuliers comme les valeurs propres multiples ou non diagonalisables, et facilite le traitement numérique dans certains contextes spécifiques [19].

En plus des avancées dans les algorithmes directs, ces dernières années ont vu un intérêt croissant et un développement remarquable de l'utilisation des méthodes itératives pour résoudre l'équation de Lyapunov-Sylvester, en raison de l'efficacité numérique qu'elles offrent, particulièrement dans le cas des matrices de grande dimension[18]. Cette démarche s'inscrit dans l'idée principale qui consiste à développer une approche itérative pour résoudre l'équation de Lyapunov-Sylvester sous la forme $AX + XB = C$, en concevant une approche similaire aux méthodes classiques de Jacobi et de Gauss-Seidel qui sont utilisées pour la résolution des systèmes linéaires sous la forme $Gx = c$. Cette approche repose sur la décomposition des matrices en parties triangulaires et résiduelles, et sur l'exploitation de la structure résultante pour formuler des algorithmes itératifs plus efficaces.

Ce travail vise à étudier les algorithmes numériques utilisés pour résoudre l'équation de Lyapunov-Sylvester, en mettant l'accent sur l'aspect algorithmique et la précision numérique, tout en explorant les possibilités de développement de nouveaux algorithmes itératifs mieux adaptés aux structures matricielles particulières [15]. Il s'agira également d'analyser les conditions nécessaires à la convergence de ces méthodes [7], ainsi que d'évaluer leur performance en termes d'efficacité numérique et de complexité de algoréthmique [1].

Equation de Lyapunov-Sylvester

Chapitre 1

I Introduction

L'équation de Lyapunov-Sylvester est une équation fondamentale dans l'analyse de la stabilité des systèmes linéaires et la conception des systèmes de contrôle, en raison de son importance dans des domaines tels que la théorie du contrôle, les équations différentielles et l'optimisation [13, 2]. Dans ce chapitre, nous concentrons sur l'étude des méthodes de résolution de cette équation en nous basant sur le produit de Kronecker, qui permet de la transformer en un système linéaire classique via la technique de vectorisation, facilitant ainsi son traitement numérique [19, 21]. Nous présentons les aspects théoriques liés à la relation entre l'équation et le produit de Kronecker, ainsi que les conditions d'existence et d'unicité de la solution basées sur l'analyse spectrale des matrices [7]. Nous évaluons également la performance numérique de certains algorithmes, particulièrement dans les cas où des simplifications sont possibles, comme pour les matrices (Triangulaires et Toeplitz, etc.) [2, 8, 3].

II Les types de l'équation de Lyapunov-Sylvester

Soit $\mathbb{K}$ un corps commutatif, l'équation de Lyapunov-Sylvester est définie de la manière suivante :

$$\sum_{i=1}^{n} A_i X B_i = C,$$

où A_i et B_i sont des matrices carrées de taille $n \times n$ et $m \times m$, respectivement et $X, C \in \mathcal{M}_{n\times m}(\mathbb{K})$.

Soient $A \in \mathcal{M}_n(\mathbb{K})$ et $B \in \mathcal{M}_m(\mathbb{K})$ deux matrices carrées de taille $n \times n$ et $m \times m$, respectivement, et $X, C \in \mathcal{M}_{n\times m}(\mathbb{K})$. Dans ce cas, l'équation de Sylvester prend la forme :

— L'équation de Sylvester définie comme suite

$$AX - XB = C. \tag{1.1}$$

— L'équation de Lyapunov est aussi définie par (si $B = A^T$)

$$AX - XA^T = C. \tag{1.2}$$

— L'équation matricielle non linéaire du type

$$AX - XB^T - XDX = C,$$

appellé l'équation de Ricatti .
Le problème est de déterminer les matrices d'inconnues X satisfant les équations précédentes.

III Relation entre l'équation de Lyapunov-Sylvester et les systèmes linéaires $Gx = c$

L'équation de Lyapunov-Sylvester prend la forme (1.1) où A , B et C sont des matrices données et X est la matrice inconnue. Cette équation est liée aux systèmes linéaires $Gx = c$ en la transformant en une forme vectorielle en utilisant la vectorisation. En utilisant le produit de Kronecker, cela permet de résoudre l'équation (1.1) par les méthodes classiques.

1 Produit de kronecker

Le produit de Kronecker est un outil clé dans la résolution de l'équation de Lyapunov-Sylvester car il permet de transformer une équation matricielle en un système linéaire classique, ce qui simplifie les calculs et facilite l'application des méthodes numériques.

Définition 1.1 [23] Soient $A = (a_{i,j})_{\substack{1\le i\le n\\ 1\le j\le m}}$ et $B = (b_{i,j})_{\substack{1\le i\le p\\ 1\le j\le q}}$ deux matrices de taille $m \times n$ et $p \times q$ respectivement. Le produit de Kronecker ou produit tensoriel de A et B noté $A \otimes B$ est la matrice de taille $mp \times nq$ est donnée par :

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix}.$$

Exemple 1

On considère les deux matrices A et B données par :

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \text{ et } B = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}.$$

Le produit de Kronecker est donné explicitement comme suit :

$$A\otimes B=\begin{pmatrix}1\cdot\begin{pmatrix}0&1\\2&3\end{pmatrix}&2\cdot\begin{pmatrix}0&1\\2&3\end{pmatrix}\\3\cdot\begin{pmatrix}0&1\\2&3\end{pmatrix}&4\cdot\begin{pmatrix}0&1\\2&3\end{pmatrix}\\5\cdot\begin{pmatrix}0&1\\2&3\end{pmatrix}&6\cdot\begin{pmatrix}0&1\\2&3\end{pmatrix}\end{pmatrix}=\begin{pmatrix}0&1&0&2\\2&3&4&6\\0&3&0&4\\6&9&8&12\\0&5&0&6\\10&15&12&18\end{pmatrix}.$$

Propriétés 1.1. *Soient A, B, C trois matrices de même taille. On a les propriétés suivantes :*

— ***Distributivité*** *: Le produit de Kronecker est distributif par rapport à l'addition.*

$$(\alpha A+\beta B)\otimes C=\alpha(A\otimes C)+\beta(B\otimes C),$$

$$A\otimes(\alpha B+\beta C)=\alpha(A\otimes B)+\beta(A\otimes C).$$

— ***Associativité*** *:*

$$(A\otimes B)\otimes C=A\otimes(B\otimes C).$$

— ***Rang*** *: Le rang du produit de Kronecker est le produit des rangs des matrices impliquées. C'est-à-dire, si A est une matrice de rang r_1 et B est une matrice de rang r_2, alors le produit de Kronecker $A\otimes B$ est une matrice de rang $r_1\times r_2$.*

— ***Transposition*** *:*

$$(A\otimes B)^T=A^T\otimes B^T.$$

Définition 1.2 [23] Soient A une matrice de taille $m\times n$, notée par $A=[A_1,\dots,A_n]$, les colonnes de A. La vectorisation de A, notée $\mathrm{vec}(A)$, est un vecteur colonne de taille $mn\times 1$ et on écrit :

$$\mathrm{vec}(A)=\begin{bmatrix}A_1\\\vdots\\A_n\end{bmatrix}.$$

2 Transformation de l'équation de Lyapunov-Sylvester en un système linéaire $Gx=c$

La transformation de l'équation linéaire de Lyapunov-Sylvester de la forme (1.1) en un système linéaire de la forme $Gx=c$, consiste à utiliser le produit de Kronecker et la technique de vectorisation.

Nous illustrerons cela à l'aide d'un exemple pratique dans $M_2(\mathbb{K})$. On considère les matrices A, B et C données par :

$$A=\begin{pmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{pmatrix},\quad B=\begin{pmatrix}b_{11}&b_{12}\\b_{21}&b_{22}\end{pmatrix},\quad C=\begin{pmatrix}c_1&c_3\\c_2&c_4\end{pmatrix}.$$

On cherche une matrice X donnée par :

$$X=\begin{pmatrix}x_1&x_3\\x_2&x_4\end{pmatrix}.$$

Comme solution de l'équation (1.1), ce qui donne après remplacement :

$$\begin{pmatrix} a_{11}x_1 + a_{12}x_2 - b_{11}x_1 - b_{21}x_3 & a_{11}x_3 + a_{12}x4 - b_{12}x_1 - b_{22}x_3 \\ a_{21}x_1 + a_{22}x_2 - b_{11}x_2 - b_{21}x_4 & a_{21}x_3 + a_{22}x_4 - b_{12}x_2 - b_{22}x_4 \end{pmatrix} = \begin{pmatrix} c_1 & c_3 \\ c_2 & c_4 \end{pmatrix}.$$

Par identiocation, on obtient le système linéaire

$$\begin{cases} (a_{11} - b_{11})x_1 + a_{12}x_2 - b_{21}x_3 = c_1 \\ -b_{21}x_4 + (a_{22} - b_{11})x_2 + a_{21}x_1 = c_2 \\ -b_{12}x_1 + (a_{11} - b_{22})x_3 + a_{12}x_4 = c_3 \\ -b_{12}x_2 + a_{21}x_3 + (a_{22} - b_{22})x_4 = c_4. \end{cases}$$

Ce qui est donne sous forme matricielle par

$$\underbrace{\begin{pmatrix} a_{11} - b_{11} & a_{12} & -b_{21} & 0 \\ a_{21} & a_{22} - b_{11} & 0 & -b_{21} \\ -b_{12} & 0 & a_{11} - b_{22} & a_{12} \\ 0 & -b_{12} & a_{21} & a_{22} - b_{22} \end{pmatrix}}_{G} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}}_{c}.$$

Nous observons que la matrice de ce système est décomposée comme suit :

$$\begin{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} - b_{11}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & -b_{21}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ -b_{12}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} - b_{22}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix}.$$

La matrice G, est exprimée à l'aide du produit de Kronecker sous la forme :

$$G = (I_n \otimes A - B^T \otimes I_m). \tag{1.3}$$

Nous proposons maintenant une généralisation de cette transformation. Soient $A = (a_{i,j})_{\substack{1 \le i \le m \\ 1 \le j \le m}}$ et $B = (b_{i,j})_{\substack{1 \le i \le n \\ 1 \le j \le n}}$. I_n et I_m sont les matrices identité de tailles $n \times n$ et $m \times m$, respectivement. la fonction vec(X) transforme la matrice X de taille $m \times n$ en un vecteur colonne de taille $mn \times 1$. De la même manière, vec(C) transforme la matrice c de taille $m \times n$ en un vecteur colonne de taille $mn \times 1$, alors l'équation (1.1) se transforme en un système linéaire standard sous la forme :

$$G\text{vec}(X) = \text{vec}(C).$$

Dans ce cas, si la matrice$G$ est inversible, alors l'équation admet une solution unique.

Exemple 2

Soient les matrices suivantes :

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & 3 & 1 \\ 1 & 0 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}, \quad C = \begin{pmatrix} 4 & 8 \\ 7 & 14 \\ 5 & 10 \end{pmatrix} et \quad X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix}.$$

En utilisant l'opérateur de vectorisation pour réécrire les matrices X et C sous forme vectorielle :

$$\text{vec}(X) = \begin{pmatrix} x_{11} \\ x_{21} \\ x_{31} \\ x_{12} \\ x_{22} \\ x_{32} \end{pmatrix}, \quad \text{vec}(C) = \begin{pmatrix} 4 \\ 7 \\ 5 \\ 8 \\ 14 \\ 10 \end{pmatrix}.$$

La matrice G est définie comme suit :

$$G = (I_2 \otimes A - B^T \otimes I_3).$$

On utilise le produit de Kronecker pour trouver la matrice G :

$$G = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ -2 & 0 & 0 & -1 & 1 & 0 \\ 0 & -2 & 0 & 0 & 0 & 1 \\ 0 & 0 & -2 & 1 & 0 & -1 \end{pmatrix}.$$

Nous avons :

$$G \cdot \text{vec}(X) = \text{vec}(C).$$

Si G est inversible, alors :

$$\text{vec}(X) = G^{-1} \cdot \text{vec}(C).$$

Après calculs, la matrice X s'écrit sous la forme suivante :

$$X = \begin{pmatrix} 2 & 34 \\ 2 & 46 \\ 3 & 18 \end{pmatrix}.$$

3 Complexité algorithmique

Définition 1.3 [21] La complexité algorithmique est une mesure permettant d'évaluer l'efficacité d'un algorithme en termes de temps d'exécution et d'espace mémoire requis.

Objectif 1. *L'objectif principal de la complexité algorithmique est de quantifier la difficulté d'un problème et d'anticiper les ressources nécessaires lorsque la taille des données d'entrée augmente. Elle permet également de comparer différents algorithmes et d'optimiser les performances en sélectionnant la méthode la plus efficace.*

On s'intéresse à la complexité du calcul de la solution de l'équation (1.1) en utilisant le produit de Kronecker[23].

— Les deux opérations , $I_n \otimes A$ et $B^T \otimes I_m$ nécessitent $O(n^2m^2)$ opérations.

— Leur addition n'affecte pas la complexité, et on a donc $O(n^2m^2)$ opérations.

— L'inversion de la matrice G nécessite $O((n^2m^2)^3)$ opérations.

— La multiplication de G^{-1} par le vecteur colonne $vec(C)$ nécessite $O(n^2m^2)$ opérations.

— La réorganisation des éléments pour obtenir la matrice X nécessite $O(n^2m^2)$ opérations.

Ainsi, la complexité totale de la résolution de l'équation de Sylvester en utilisant le produit de Kronecker est généralement dominée par $O((n^2m^2)^3)$.

— L'équation de Lyapunov (1.2) sa complexité est dominée par $O((n^2)^3)$.

Probleme 1.1. *Nous visons dans cette étude à déterminer si la matrice G est inversible ou non. Cela se fait en calculant le déterminant de la matrice. De plus, certains types de matrices sont directement inversibles en raison de leurs propriétés structurelles.*

Les différentes méthodes de calcul du déterminant et analyse de leur complexité

1. *Développement par les cofacteurs : Cette méthode repose sur la définition mathématique du déterminant, qui est calculé en développant le long de la première ligne ou colonne :*

$$\det(G) = \sum_{j=1}^{n} (-1)^{1+j} \cdot G_{1j} \cdot \det\Big(M_{1j}\Big),$$

où :

   — *G_{1j} est l'élément situé à la première ligne et à la j colonne de la matrice G.*

   — *M_{1j} est la sous-matrice obtenue en supprimant la première ligne et la j colonne de la matrice G.*

   Cette méthode devient rapidement impraticable pour les matrices de grande dimensions.

2. *Élimination de Gauss :*
   Cette méthode consiste à transformer la matrice en une forme triangulaire supérieure, où le déterminant est donné par :

$$\det(G) = \prod_{i=1}^{n} u_{ii},$$

   où u_{ii} sont les éléments diagonaux de la matrice triangulaire.

3. *Décomposition LU :*
   Cette décomposition est efficace pour les matrices de grande taille, car elle consiste à écrire la matrice G sous la forme d'un produit de deux matrices triangulaires, c'est-à-dire :

$$\det(G) = \det(L) \times \det(U) = \left(\prod_{i=1}^{n} l_{ii}\right) \times \left(\prod_{i=1}^{n} u_{ii}\right),$$

   où :

   — *l_{ii} sont les éléments diagonaux de la matrice triangulaire inférieure L.*

   — *u_{ii} sont les éléments diagonaux de la matrice triangulaire supérieure U.*

La matrice G est inversible si elle appartient à l'un des types suivants :

1. *Matrice diagonale : Une matrice diagonale est inversible si et seulement si tous ses éléments diagonaux sont non nuls.*
2. *Matrice orthogonale : Une matrice orthogonale vérifie $G^T G = I$. Ces matrices sont toujours inversibles, avec $G^{-1} = G^T$.*
3. *Matrice triangulaire : Une matrice triangulaire (supérieure ou inférieure) est inversible si et seulement si tous ses éléments diagonaux sont non nuls.*
4. *Matrice de permutation : Une matrice de permutation est obtenue en permutant les lignes ou les colonnes de la matrice identité. Ces matrices sont toujours inversibles.*
5. *Matrice inversible par blocs : Si une matrice est partitionnée en blocs et que chaque bloc est inversible, alors la matrice globale est inversible.*

IV Existence et unicité

Définition 1.4 [12]

Soit A est une matrice carrée d'ordre n, le spectre de A, noté spect(A), est défini comme suite :

$$\text{spect}(A) = \{\lambda \in \mathbb{C} \mid \det(A - \lambda I_n) = 0\}.$$

où $spect(A)$ est l'ensemble des valeurs propres de la matrice A.

Théorème 1.1. *[23, 19] L'équation matricielle (1.1) admet une solution unique si et seulement si les spectres de A et $-B$ sont disjoints, c'est-à-dire :*

$$spect(A) \cap spect(-B) = \emptyset.$$

Preuve 1. *[23, 19]*

En utilisant le produit de Kronecker, on obtient :

$$Gvec(X) = vec(C),$$

L'existence et l'unicité de $vec(X)$ sont garanties si et seulement si la matrice G est inversible, c'est-à-dire que son spectre ne contient pas zéro.

Il est connu que le spectre de la somme de produits de Kronecker suit la règle :

$$spect(G) = spect(I_n \otimes A - B^T \otimes I_n) = \{\lambda_i - \mu_j \mid \forall \lambda_i \in spect(A) \quad et \; \forall \mu_j \in spect(B)\} i \in \{1,\dots,n\}, j \in \{1,\dots,m\}.$$

En déduire que la matrice G est inversible si et seulement si :

$$\lambda_i - \mu_j \neq 0, \quad \forall \lambda_i \in spect(A), \forall \mu_j \in spect(B) \quad i \in \{1,\dots,n\}, j \in \{1,\dots,m\}.$$

Ce qui est équivalent à :

$$spect(A) \cap spect(-B) = \emptyset.$$

1 Différentes techniques pour déterminer les valeurs propres

Le calcul des valeurs propres constitue l'un des sujets fondamentaux de l'analyse linéaire, en raison de son importance dans l'étude de la stabilité des systèmes linéaires et l'analyse de la structure spectrale des matrices. Au fil des décennies, plusieurs techniques numériques et théoriques ont été développées afin de déterminer ces valeurs avec précision ou d'estimer leur position dans le plan complexe. Ces méthodes varient en termes de précision et de complexité, allant du calcul direct à l'aide du polynôme caractéristique à des estimations approximatives comme le théorème de Gershgorine. Ces outils visent à faciliter l'étude du comportement des matrices, notamment lorsqu'elles sont de grande dimension .

1.1 Polynôme Caractéristique

Le polynôme caractéristique est un outil fondamental en algèbre linéaire permettant de déterminer les valeurs propres d'une matrice carrée. Il est défini pour une matrice A de taille $n \times n$ par l'équation suivante :

$$P(\lambda) = \det(A - \lambda I_n) = 0.$$

où :

— $\det(A - \lambda I_n)$ représente le déterminant de la matrice $A - \lambda I_n$.

— λ est une variable scalaire.

Propriétés 1.2.

1. *Degré du polynôme : Le polynôme caractéristique $P(\lambda)$ est toujours un polynôme de degré n, où n est la taille de la matrice A.*
2. *Valeurs propres : Les racines du polynôme caractéristique.*

Exemple 3

Prenons la matrice suivante :

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 2 & 4 \\ 0 & 0 & 3 \end{pmatrix}.$$

Le polynôme caractéristique est donné par :

$$P(\lambda) = \det(A - \lambda I) = \det \begin{pmatrix} 1-\lambda & 2 & 3 \\ 0 & 2-\lambda & 4 \\ 0 & 0 & 3-\lambda \end{pmatrix}.$$

Ainsi, le polynôme caractéristique devient :

$$P(\lambda) = (3 - \lambda)(\lambda^2 - 3\lambda + 2),$$

Les racines sont : $\lambda = \{1, 2, 3\}$.

1.2 Disque de Gershgorine

Le disque de Gershgorine est un outil important en analyse matricielle, offrant une méthode simple et efficace pour estimer la localisation des valeurs propres d'une matrice carrée sans avoir à les calculer précisément. Il repose sur un théorème puissant qui délimite la répartition des valeurs propres dans le plan complexe, ce qui le rend particulièrement utile pour les matrices de grande taille et complexes, où les méthodes classiques de calcul des valeurs propres deviennent difficiles et coûteuses en termes de calcul.

Définition 1.5 [7] Soit A une matrice carrée de taille $n \times n$, et a_{ij} l'élément de la matrice A à la i-ème ligne et j-ème colonne. Le disque de Gershgorine associé à la ligne i de la matrice A est défini comme un disque dans le plan complexe, de centre a_{ii} (l'élément diagonal de la matrice) et de rayon égal à la somme des valeurs absolues des éléments non diagonaux de la ligne i.

Propriétés 1.3.

1. ***Localisation des Valeurs Propres*** *: Les disques de Gershgorine fournissent une première approximation des valeurs propres de la matrice. D'après le théorème de Gershgorine, chaque disque contient au moins une valeur propre de la matrice A.*
2. ***Cas d'une matrice contenant une ligne nulle (sauf l'élément diagonal)*** *:*
   Si une matrice A contient une ligne où tous les éléments sont nuls sauf l'élément diagonal a_{ii}, alors le disque de Gershgorine correspondant à cette ligne se réduit au point a_{ii}, car son rayon est nul. En revanche, les autres disques conservent leurs rayons définis par la somme des valeurs absolues des éléments hors diagonale. Cela signifie que a_{ii} est nécessairement l'une des valeurs propres de la matrice.

Théorème 1.2. *[7] Le théorème de Gershgorine donne une localisation des valeurs propres de la matrice A. Il indique que toute valeur propre de A se trouve dans au moins un des disques de Gershgorine. Plus précisément :*

— *Soit la matrice $A \in \mathcal{M}_n(\mathbb{R})$ de taille $n \times n$.*

— *Le disque de Gershgorine D_i associé à la ligne i de la matrice A donné par :*

$$D_i = \left\{ \lambda \in \mathbb{C} \,|\, |\lambda - a_{ii}| \leq \sum_{j \neq i}^{n} |a_{ij}| \right\}.$$

Preuve 2. *[7] Soientla matriceA $\in \mathcal{M}_n(\mathbb{R})$, et $\lambda \in \mathbb{R}$ une valeur propre de A, c'est-à-dire qu'il existe un vecteur propre $v = (v_1, v_2, \ldots, v_n)^T$ tel que :*

$$Av = \lambda v.$$

Cela implique que les composantes de Av sont :

$$a_{ii}v_i + \sum_{j \neq i}^{n} a_{ij}v_j = \lambda v_i.$$

De cette équation, on obtient :

$$\sum_{j\neq i}^{n} a_{ij}v_j = (\lambda - a_{ii})v_i.$$

En prenant les valeurs absolues des deux côtés, et en appliquant l'inégalité triangulaire, on obtient :

$$\left|\sum_{j\neq i}^{n} a_{ij}v_j\right| \leq \sum_{j\neq i}^{n} |a_{ij}||v_j|.$$

Cela montre que :

$$|\lambda - a_{ii}| \leq \sum_{j\neq i}^{n} |a_{ij}|.$$

Ainsi, λ se trouve dans un disque centré en a_{ii} de rayon $\sum_{j\neq i}^{n} |a_{ij}|$.

En utilise les disques de Gershgorine pour estimer l'emplacement des valeurs propres d'une matrice carrée. On peux également utiliser cette théorie pour étudier la séparation des spectres de deux matrices qui ne partagent pas de valeurs propres. Lors de l'étude de l'intersection des spectres $\text{spect}(A) \cap \text{spect}(B)$.

Lorsqu'il n'y a pas d'intersection entre les spectres : Si chaque disque de Gershgorine associé à la matrice A est distinct de ceux associés à la matrice B, alors :

$$\left(\bigcup_i D_i(A)\right) \cap \left(\bigcup_j D_j(B)\right) = \emptyset \quad i, j \in \{1, \ldots, n\}.$$

Ce qui implique que :

$$\text{spect}(A) \cap \text{spect}(B) = \emptyset.$$

Lorsque l'intersection est possible : Si certains disques de Gershgorine associés aux matrices A et B se chevauchent, alors une intersection entre ces disques est possible, ce qui peut entraîner une intersection entre les spectres. Dans ce cas, si l'intersection entre les disques n'est pas vide, c'est-à-dire :

$$\left(\bigcup_i D_i(A)\right) \cap \left(\bigcup_j D_j(B)\right) \neq \emptyset \quad i, j \in \{1, \ldots, n\}.$$

Il n'est pas possible de conclure de manière définitive qu'il existe une intersection entre les valeurs propres des matrices A et B sur la base de ces informations seules, car le chevauchement des disques ne garantit pas nécessairement l'intersection des spectres.

Pour illustrer l'application du théorème de Gershgorin, nous présentons deux exemples concrets. Le premier montre une situation où les disques associés à deux matrices ne se chevauchent pas, indiquant une séparation claire des spectres. Le second met en évidence un cas où certains disques se chevauchent, rendant possible une intersection entre les spectres des deux matrices.

Exemple 4

Considérons les deux matrices triangulaires supérieures suivantes :

$$A = \begin{pmatrix} 7.1 & 0.4 & 0 \\ 0 & 4.6 & 0.6 \\ 0 & 0 & 2.2 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 5.9 & 0.3 & 0 \\ 0 & 3.1 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}.$$

Les valeurs propres des matrices triangulaires supérieures A et B sont égales à leurs éléments diagonaux :

— Pour la matrice A :

$$\lambda_1 = 7.1, \quad \lambda_2 = 4.6, \quad \lambda_3 = 2.2.$$

— Pour la matrice B :

$$\mu_1 = 5.9, \quad \mu_2 = 3.1, \quad \mu_3 = 1.$$

Pour appliquer le théorème de Gershgorin, nous construisons les disques associés à chaque ligne de chacune des matrices :

Disques de Gershgorin pour la matrice A :

— Ligne 1 : Disque : $|\lambda - 7.1| \leq 0.4$

— Ligne 2 : Disque : $|\lambda - 4.6| \leq 0.6$

— Ligne 3 : Disque : $|\lambda - 2.2| \leq 0$ (se réduit à un point)

Disques de Gershgorin pour la matrice B :

— Ligne 1 : Disque : $|\lambda - 5.9| \leq 0.3$

— Ligne 2 : Disque : $|\lambda - 3.1| \leq 0.5$

— Ligne 3 : Disque : $|\lambda - 1| \leq 0$ (se réduit à un point)

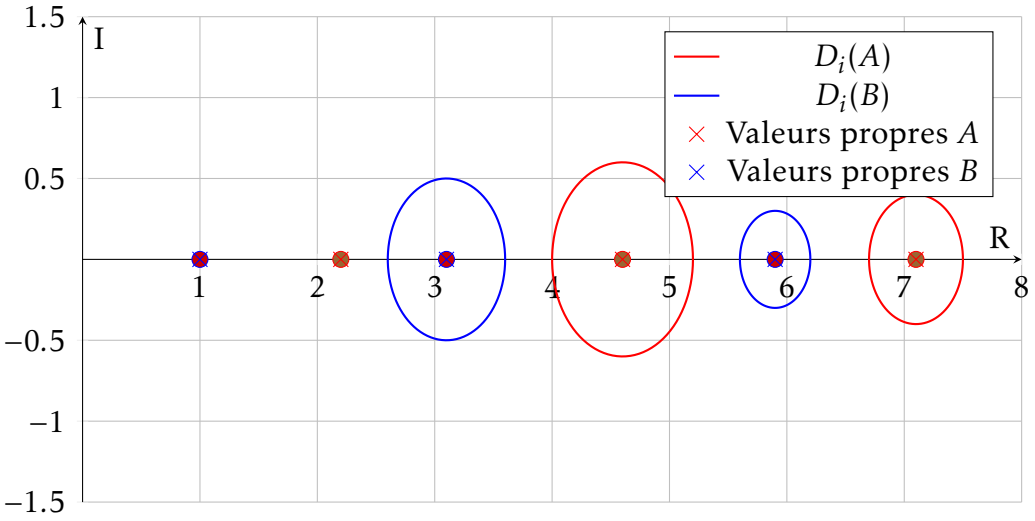


FIGURE 1.1 – Disques de Gershgorin pour les matrices A et B.

Exemple 5

Considérons les deux matrices triangulaires supérieures suivantes :

$$A = \begin{pmatrix} 6 & 2 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & 3 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 5 & -1 & 0 \\ 0 & 2 & 3 \\ 0 & 0 & 1 \end{pmatrix}.$$

Les valeurs propres des matrices triangulaires supérieures A et B sont égales à leurs éléments diagonaux :

— Pour la matrice A :

$$\lambda_1 = 6, \quad \lambda_2 = 4, \quad \lambda_3 = 3.$$

— Pour la matrice B :

$$\mu_1 = 5, \quad \mu_2 = 2, \quad \mu_3 = 1.$$

Pour appliquer le théorème de Gershgorin, nous construisons les disques associés à chaque ligne de chacune des matrices.

Disques de Gershgorin pour la matrice A :

— Ligne 1 : Disque : $|z - 6| \leq 3$

— Ligne 2 : Disque : $|z - 4| \leq 1$

— Ligne 3 : Disque : $|z - 3| \leq 0$ (se réduit à un point)

Disques de Gershgorin pour la matrice B :

— Ligne 1 : Disque : $|z - 5| \leq 1$

— Ligne 2 : Disque : $|z - 2| \leq 3$

— Ligne 3 : Disque : $|z - 1| \leq 0$ (se réduit à un point)

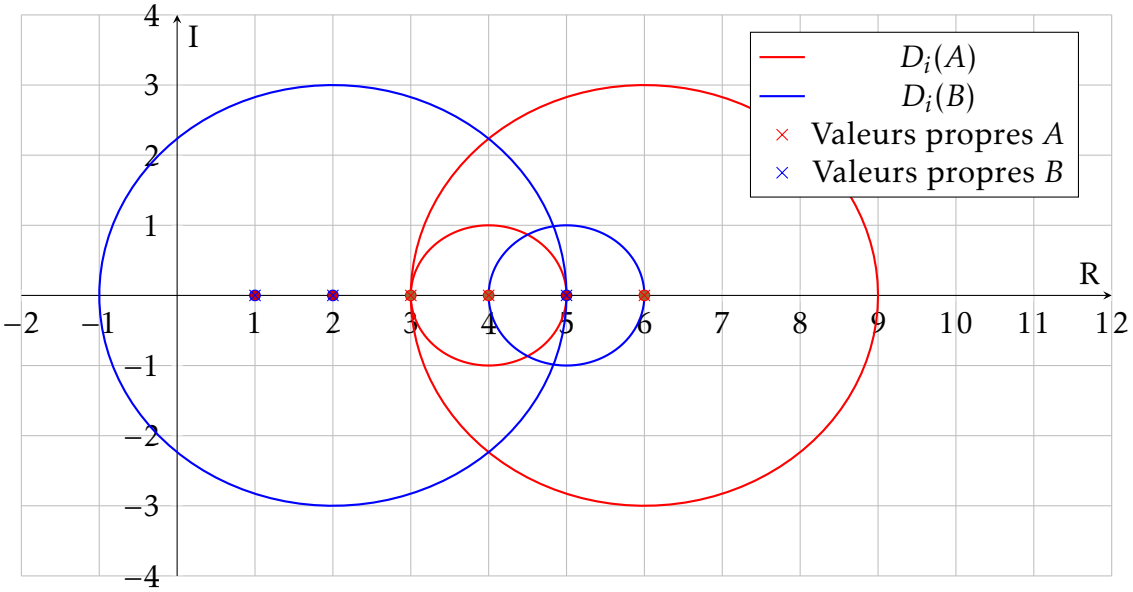


FIGURE 1.2 – Disques de Gershgorin pour les matrices A et B.

V Nouvelles méthodes pour calculer le produit matricielle

Le produit matriciel constitue une opération fondamentale en algèbre linéaire, intervenant dans de nombreux domaines scientifiques et techniques, tels que la résolution des systèmes linéaires, l'analyse de données et les équations différentielles matricielles. Bien que l'algorithme classique, de complexité $\mathcal{O}(n^3)$, soit adapté aux matrices de petite taille, il devient inefficace en pratique lorsqu'il s'agit de grandes matrices, ce qui motive le recours à des algorithmes plus performants sur le plan du coût computationnel.

1 Algorithmes avancés de multiplication matricielle

— **Algorithme de Strassen (1969) :** Réduit la complexité à environ $\mathcal{O}\left(n^{\log_2 7}\right) \approx \mathcal{O}(n^{2.81})$ en divisant les matrices en blocs et en utilisant des additions et soustractions astucieuses. Adapté aux grandes matrices, mais avec des constantes cachées élevées qui peuvent limiter son efficacité pratique[22].

— **Algorithme de Coppersmith-Winograd (1987) :** Complexité théorique de $\mathcal{O}(n^{2.376})$, mais peu utilisé en pratique à cause des constantes énormes[5].

— **Algorithme de Williams-Le Gall (2014) :** Amélioration théorique à $\mathcal{O}(n^{2.3729})$, mais reste principalement d'intérêt théorique[17].

2 Optimisations pratiques pour les calculs modernes

— **Découpage en blocs (Tiling) :** Optimise l'utilisation du cache CPU en divisant les matrices en sous-blocs pour réduire les défauts de cache[9].

— **Parallélisation (CPU/GPU) :** Utilisation d'OpenMP, CUDA (pour Nvidia), ou ROCm (pour AMD) pour exploiter les cœurs multiples et les GPU[14].

— **Bibliothèques optimisées :** BLAS niveau 3 (GEMM) pour des opérations optimisées. Intel MKL, OpenBLAS, cuBLAS (Nvidia), hipBLAS (AMD) : Implémentations ultra-rapides pour CPU/GPU[6].

3 Méthodes adaptées aux matrices creuses

— **Format CSR/CSC :** Stockage compressé adapté aux matrices avec beaucoup de zéros[21].

— **Algorithmes spécifiques :** Méthodes itératives ou multiplication par blocs pour éviter les calculs inutiles[21].

Lorsqu'on travaille avec des matrices de grande taille, on constate souvent que la majorité de leurs éléments sont nuls. Ce type de matrices est appelé matrices creuses.

Définition 1.6 [21] [**Matrice creuse**] Une matrice creuse est une matrice contenant un grand nombre d'éléments nuls, de sorte que les éléments non nuls représentent une faible proportion par rapport à la taille totale. Pour des raisons d'efficacité en stockage et en calcul, ces matrices sont souvent représentées de manière

compacte en ne conservant que les éléments non nuls ainsi que leurs positions.

Ce type de matrices présente un grand intérêt en calcul numérique, notamment pour les systèmes de grande dimension. En effet, les matrices creuses permettent de réduire considérablement l'utilisation de la mémoire et d'accélérer les opérations algébriques telles que la multiplication matricielle ou la résolution de systèmes linéaires.

Après avoir défini les matrices creuses et souligné leur importance dans la réduction de la consommation mémoire ainsi que dans l'accélération des calculs numériques, il est naturel de s'intéresser à leur représentation en programmation.

Dans le logiciel MATLAB, une structure de données dédiée, appelée `sparse`, permet de stocker efficacement les matrices creuses sans mémoriser les éléments nuls.

- Dans MATLAB, la commande `sparse` sert à créer ou convertir une matrice en format creux.
- Cette représentation réduit significativement la consommation mémoire, surtout pour les matrices comportant un grand nombre de zéros.
- Elle contribue aussi à accélérer les opérations numériques telles que la multiplication matricielle ou la résolution de systèmes linéaires.
- Elle est particulièrement utile pour le traitement d'équations de grande dimension, comme les équations de Lyapunov ou de Sylvester.

Dans cet exemple, nous illustrons la différence entre la multiplication classique (dense) et la multiplication optimisée (sparse) de matrices creuses, à l'aide d'un cas simple en dimension 3×3. L'objectif est de réduire le nombre d'opérations inutiles et de comparer les deux approches en termes d'efficacité.

Exemple 6

Considérons les deux matrices creuses suivantes :

$$A = \begin{pmatrix} \mathbf{5} & 0 & 0 \\ 0 & 0 & \mathbf{2} \\ 0 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 & \mathbf{1} \\ 0 & \mathbf{3} & 0 \\ 0 & 0 & \mathbf{4} \end{pmatrix}.$$

Multiplication dense (classique)

On applique la formule classique :

$$c_{i,j} = \sum_{k=1}^{3} a_{i,k} \cdot b_{k,j},$$

On a besoin 27 opérations de multiplication et 18 opérations de addition pour trouver la matrice C :

$$C = \begin{pmatrix} 0 & 0 & \mathbf{5} \\ 0 & 0 & \mathbf{8} \\ 0 & 0 & 0 \end{pmatrix}.$$

Multiplication optimisée (sparse)

On ne conserve que les éléments non nuls :

- **Matrice A** : lignes = [1, 2], colonnes = [1, 3], valeurs = [5, 2].

— **Matrice B** : lignes = [1, 2, 3], colonnes = [3, 2, 3], valeurs = [1, 3, 4].

On multiplie uniquement les paires où l'indice de colonne dans A correspond à l'indice de ligne dans B.

— $A_{1,1} = 5$: on cherche $B_{1,j}$ non nuls $\Rightarrow B_{1,3} = 1$
$\Rightarrow C_{1,3} = 5 \times 1 = 5$

— $A_{2,3} = 2$: on cherche $B_{3,j}$ non nuls $\Rightarrow B_{3,3} = 4$
$\Rightarrow C_{2,3} = 2 \times 4 = 8$

Résultat obtenu avec seulement deux multiplications :

$$C = \begin{pmatrix} 0 & 0 & \mathbf{5} \\ 0 & 0 & \mathbf{8} \\ 0 & 0 & 0 \end{pmatrix}.$$

VI Analyse des cas particuliers de l'équation de Lyapunov-Sylvester

Dans certains cas particuliers, la résolution de cette équation peut être simplifiée grâce à des propriétés spécifiques des matrices utilisées. La simplification des solutions dépend de la structure particulière des matrices, comme la présence d'éléments nuls ou une distribution particulière des éléments dans la matrice. Ces propriétés contribuent à accélérer les calculs et facilitent la recherche des solutions. Cependant, nous présentons certains cas particuliers qui facilitent la résolution de l'équation de Lyapunov-Sylvester, tels que les matrices diagonales, les matrices de Toeplitz tridiagonales et les matrices triangulaires supérieures et inférieures.

Nous allons examiner en détail chaque cas en expliquant ses propriétés et ses principales applications.

1 Cas des matrices diagonales $A = D$ et $B = D'$

Considérons l'équation matricielle suivante.

$$DX + XD' = C. \tag{1.4}$$

où :

— $D \in \mathcal{M}_n(\mathbb{R})$, $D' \in \mathcal{M}_m(\mathbb{R})$sont des matrices diagonales données, et $C \in \mathcal{M}_{n,m}(\mathbb{R})$ est une matrice donnée, $X \in \mathcal{M}_{n,m}(\mathbb{R})$ est la matrice inconnue à déterminer.

L'équation devient (1.4), On peut alors calculer chaque élément selon l'ordre suivant :

$$X_{ij} = \frac{c_{ij}}{d_{ii} + d'_{jj}}, \quad i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}.$$

à condition que $d_{ii} + d'_{jj} \neq 0$.
Avec des matrices de taille 3×3, pour on peut généraliser à une taille n.

$$\underbrace{\begin{pmatrix} d_{11} & 0 & 0 \\ 0 & d_{22} & 0 \\ 0 & 0 & d_{33} \end{pmatrix}\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}}_{DX} + \underbrace{\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}\begin{pmatrix} d'_{11} & 0 & 0 \\ 0 & d'_{22} & 0 \\ 0 & 0 & d'_{33} \end{pmatrix}}_{XD'} = \underbrace{\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}}_{C},$$

L'algorithme utilisé pour résoudre le problème (1.4) est le suivant :

```
1  function X=LyapSylvDiag(D, D',C)
2  input :D,D',C;
3  output :X;
4  n=length(D);
5  m=length(D');
6  X=zeros(n,m);
7  for j=1 :m do
8  |  for i=1 :n do
9  |  |  X(i,j) = C(i,j) / (D(i,i) + D'(j,j));
10 |  end
11 end
```

Exemple 7

Soient les matrices suivantes :

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \quad D' = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 6 \end{pmatrix}, \quad C = \begin{pmatrix} 10 & 6 & 14 \\ 12 & 14 & 32 \\ 14 & 24 & 27 \end{pmatrix} et \quad X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}.$$

la solution d'équation (1.4) est :

$$X = \begin{pmatrix} 2 & 1 & 2 \\ 2 & 2 & 4 \\ 2 & 3 & 3 \end{pmatrix}.$$

2 Matrice de Toeplitz tridiagonale

Définition 1.7 [11] Soient A et B deux matrices tridiagonales de Toeplitz de taille $n \times n$ de la forme suivante :

$$A = \begin{pmatrix} a & b & 0 & \cdots & 0 \\ c & a & b & \cdots & 0 \\ 0 & c & a & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & b \\ 0 & 0 & 0 & c & a \end{pmatrix}, \quad B = \begin{pmatrix} p & q & 0 & \cdots & 0 \\ r & p & q & \cdots & 0 \\ 0 & r & p & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & q \\ 0 & 0 & 0 & r & p \end{pmatrix}.$$

où :

— $a, b, c, p, r, q \in \mathbb{R}$

Les valeurs propres λ_k de A sont définies par :

$$\lambda_k = a + 2\sqrt{bc}\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, 2, \ldots, n$$

Les vecteurs propres θ_k de A sont définies par :

$$\theta_k = \left(\left(\frac{\sqrt{bc}}{b}\right)^j \sin\left(\frac{jk\pi}{n+1}\right)\right)_{j=1,2,\ldots,n}$$

Les valeurs propres μ_k de B sont définies par :

$$\mu_k = p + 2\sqrt{qr}\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, 2, \ldots, n$$

Les vecteurs propres $\mathbf{v}_k$ de B sont définies par :

$$\mathbf{v}_k = \left(\left(\frac{\sqrt{qr}}{q}\right)^j \sin\left(\frac{jk\pi}{n+1}\right)\right)_{j=1,2,\ldots,n}.$$

Les matrices A et B, admettent des décompositions spectrales de la forme :

$$A = UDU^T \quad \text{et} \quad B = VD'V^T,$$

où :

- U et V sont respectivement les matrices formées des vecteurs propres de A et B,
- $D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ la matrice diagonale contenant les valeurs propres de A.
- $D' = \text{diag}(\mu_1, \mu_2, \ldots, \mu_n)$ la matrice diagonale contenant les valeurs propres de B.

On substitue les décompositions de A et B dans l'équation originale :

$$UDU^T X + XVD'V^T = C,$$

On multiplie les deux côtés par U^T à gauche et par V à droite :

$$U^T(UDU^T X)V + U^T(XVD'V^T)V = U^T CV,$$

En utilisant $U^T U = I$ et $V^V = I$, on obtient :

$$D(U^T XV) + (U^T XV)D' = U^T CV,$$

On définit $Y = U^T XV$, et l'équation devient :

$$DY + YD' = U^T CV,$$

comme D et D' sont diagonales, on peut résoudre l'équation élément par élément :

$$Y_{ij} = \frac{(U^T CV)_{ij}}{\lambda_i + \mu_j}, \quad i, j \in \{1, \ldots, n\}.$$

à condition que $\lambda_i + \mu_j \neq 0$.

Après avoir trouvé Y, on utilise la relation $Y = U^T X V$ pour calculer X : $X = UYV^T$.

Exemple 8

Soient Les matrices A, B et C sont définies comme suit :

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}, \quad B = \begin{pmatrix} 3 & 2 & 0 \\ 2 & 3 & 2 \\ 0 & 2 & 3 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Les valeurs propres λ_k de A sont données par :

$$\lambda_k = 2 + 2\cos\left(\frac{k\pi}{4}\right), \quad k = 1, 2, 3$$

On obtient :

$$\lambda_1 = 2 + \sqrt{2}, \quad \lambda_2 = 2, \quad \lambda_3 = 2 - \sqrt{2}$$

La matrice de passage U associée à la matrice A est :

$$U = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{2}}{2} & \frac{1}{2} \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ \frac{1}{2} & -\frac{\sqrt{2}}{2} & \frac{1}{2} \end{pmatrix}.$$

Les valeurs propres μ_k de B sont données par :

$$\mu_k = 3 + 2\cos\left(\frac{k\pi}{4}\right), \quad k = 1, 2, 3$$

On obtient :

$$\mu_1 = 3 + \sqrt{2}, \quad \mu_2 = 3, \quad \mu_3 = 3 - \sqrt{2}$$

La matrice de passage V associée à la matrice B est :

$$V = \begin{pmatrix} \frac{1}{2} & \frac{\sqrt{2}}{2} & \frac{1}{2} \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ \frac{1}{2} & -\frac{\sqrt{2}}{2} & \frac{1}{2} \end{pmatrix}.$$

On calcule $U^T C V$:

$$U^T C V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

On résout l'équation (1.4)et on obtient :

$$Y = \begin{pmatrix} \frac{1}{5+2\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{5} & 0 \\ 0 & 0 & \frac{1}{5-2\sqrt{2}} \end{pmatrix}.$$

On sait que $V^{-1} = V^T$, donc :

$$X = UYV^T.$$

Après calculs, on obtient :

$$X = \begin{pmatrix} \frac{1}{5+2\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{5} & 0 \\ 0 & 0 & \frac{1}{5-2\sqrt{2}} \end{pmatrix}.$$

3 Cas des matrices triangulaires $A = B^T = L$

L'équation matricielle à résoudre est :

$$LX + XL^T = C. \tag{1.5}$$

où :

— $L \in \mathcal{M}_n(\mathbb{R})$ est une matrice triangulaire inférieure connue, $C \in \mathcal{M}_{n,m}(\mathbb{R})$ est une matrice connue et $X \in \mathcal{M}_{n,m}(\mathbb{R})$ est la matrice inconnue à déterminer.

On peut alors calculer chaque élément selon l'ordre suivant :

$$X_{ij} = \frac{c_{ij} - \sum_{k=1}^{i-1} l_{ik}x_{kj} - \sum_{k=1}^{j-1} x_{ik}l_{jk}}{l_{ii} + l_{jj}}, \quad i \in \{1,\dots,n\}, j \in \{1,\dots,m\}.$$

où : $l_{ii} + l_{jj} \neq 0$.

La résolution se fait ligne par ligne : chaque ligne de X est calculée en utilisant les valeurs déjà obtenues des lignes précédentes. Pour mieux comprendre la méthode, nous commençons par un cas simple avec une matrice de taille 3×3, avant de généraliser à une taille n.

$$\underbrace{\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix}\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}}_{LX} + \underbrace{\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}\begin{pmatrix} l_{11} & l_{12} & l_{13} \\ 0 & l_{22} & l_{23} \\ 0 & 0 & l_{33} \end{pmatrix}}_{XL^T} = \underbrace{\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}}_{C},$$

L'algorithme utilisé pour résoudre le problème (1.5) est le suivant :

```
1  function X=LyapLowerMatrix(L,C)
2  input :L,C;
3  output :X;
4  n=length(L);
5  X=zeros(n,n);
6  for j=1 :n do
7  |   for i=1 :n do
8  |   |   somme1 = sum(L(i, 1 :i-1).* X(1 :i-1, j)');
9  |   |   somme2 = sum(X(i, 1 :j-1).* L(j, 1 :j-1));
10 |   |   X(i, j) = (C(i, j) - somme1 - somme2) / (L(i, i) + L(j, j))
11 |   end
12 end
```

4 Cas des matrices triangulaires $A = L$ et $B = U$

L'équation matricielle sous la forme :

$$LX + XU = C. \tag{1.6}$$

est un cas particulier de l'équation de Sylvester $AX + XB = C$ avec :

— L une matrice triangulaire inférieure et U une matrice triangulaire supérieure.

— C une matrice est donnée et X est la matrice inconnue .

Considérons l'équation matricielle (1.6) : La résolution se fait ligne par ligne : chaque ligne de X est calculée en utilisant les valeurs déjà obtenues des lignes précédentes. Pour mieux comprendre la méthode, nous commençons par un cas simple avec une matrice de taille 3×3, avant de généraliser à une taille n.

$$\underbrace{\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix}\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}}_{LX} + \underbrace{\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix}\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}}_{XU} = \underbrace{\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}}_{C},$$

Les différents algorithmes pour résoudre le problème (1.6) sont donnés :

— Sylvester pour les systèmes LU (**S-LUS**).

— Méthode Lyap de MATLAB (**LMM**).

— Sylvester–Produit de Kronecker pour les systèmes LU (**S-KPLUS**).

— Sylvester–Produit de Kronecker pour les systèmes LU (**S-KPLUSS**).

La première algorithme repose sur le développement de l'équation (1.6).

Algorithm 1: S-LUS

```
function X=S-LUS(L,U,C)
input :L,U,C;
output :X;
n=length(L);
X=zeros(n,n);
for j=1 :n do
    for i=1 :n do
        X(i,j)=(C(i,j)-L(i,1 :i-1)*X(1 :i-1,j)-X(1 :j-1,j)'*U(i,1 :j-1))/(L(i,i)+U(j,j));
    end
end
disp(X)
```

Nous utilisons la fonction lyap de MATLAB pour résoudre des équations de Lyapunov.

Algorithm 2: LMM

```
function X=LMM(L,U,C)
input :L,U,C;
output :X;
n=length(L);
X=zeros(n,n);
X=lyap(L,U,-C);
```

Transformation de l'équation de Lyaponuv-Sylvester vers un système linéaire $Gx = c$ à l'aide du produit de Kronecker, puis sa résolution $x = G^{-1}c$.

Algorithm 3: S-KPLUS

```
function X=S-KPLUS(L,U,C)
input :L,U,C;
output :X;
n=length(L);
X=zeros(n,n);
I=eye(n,n);
G=kron(I,L)+kron(U',I);
c=reshape(C,[ ],1);
x=inv(G)*c;
Z=reshape(x,[ ],n);
```

La deuxième algorithme repose sur la transformation de l'équation (1.3) en $Gx = c$ à l'aide du produit de Kronecker, puis la résolution de système ligne par ligne.

Algorithm 4: S-KPLUSS

```
function X=S-KPLUSS(L,U,C)
input :L,U,C;
output :X;
n=length(L);
X=zeros(n,n);
I=eye(n,n);
G=kron(I,L)+kron(U',I);
c=reshape(C,[],1);
x=zeros(n^2,1);
for i=1 :n^2 do
    s=G(i,1 :i-1)*x(1 :i-1);
    x(i)=(c(i)-s)/G(i,i)
end
X=reshape(x,[],n);
```

5 Comparaison Numérique

D'après les figures et le tableau ci-dessous, on compare les temps CPU pour différentes méthodes de résolution de l'équation $LX + XU = C$. Les résultats montrent que les méthodes **S-LUS** et **LMM** offrent de bonnes performances, avec une stabilité et une rapidité remarquables, particulièrement lorsque la taille de la matrice N augmente. Cela est dû à la structure numérique efficace adoptée dans ces deux méthodes, qui permet de traiter les grandes dimensions de manière performante.

En revanche, les méthodes **S-KPLUS** et **S-KPLUSS** rencontrent des difficultés lorsque la taille de la matrice devient grande, malgré l'utilisation d'une structure *sparse* dans leur implémentation, dans le but d'optimiser les performances et de réduire l'utilisation de la mémoire. Les expériences ont montré que ces deux méthodes cessent de fonctionner à partir de $N = 400$, en raison de la complexité croissante des calculs, notamment lors de la formation de matrices de Kronecker de grande taille, qui deviennent denses à certains stades de l'algorithme, ce qui entraîne une perte d'efficacité de la structure *sparse* et une augmentation significative des besoins en calcul et en mémoire.

Table 1.1 – Comparaison des temps CPU pour différentes méthodes de résolution de $LX + XU = C$.

N	CPU_{S-LUS}	CPU_{LMM}	$CPU_{S-KPLUS}$	$CPU_{S-KPLUSS}$
4	4.1×10^{-3}	3.3×10^{-2}	3.9×10^{-4}	3.61×10^{-4}
10	5.1×10^{-3}	3.5×10^{-2}	6.13×10^{-4}	6.1×10^{-3}
50	4.5×10^{-2}	3.9×10^{-2}	2.8×10^{-3}	2.1×10^{-1}
100	1.48×10^{-1}	5.1×10^{-2}	5.27×10^{-2}	2.81
200	5.1×10^{-1}	6.2×10^{-2}	2.1×10^{-1}	73.62
400	2.07	2.6×10^{-1}	12.89	2495.48
600	5.25	8.9×10^{-1}	/	/
800	12.52	2.308	/	/
1000	22.71	4.563	/	/
1500	83.49	24.51	/	/
2000	209.6	68.61	/	/

Ainsi, les méthodes **S-LUS** et **LMM** s'avèrent les plus adaptées aux applications nécessitant le traitement de matrices de grande dimension, grâce à leur efficacité et leur stabilité numérique.

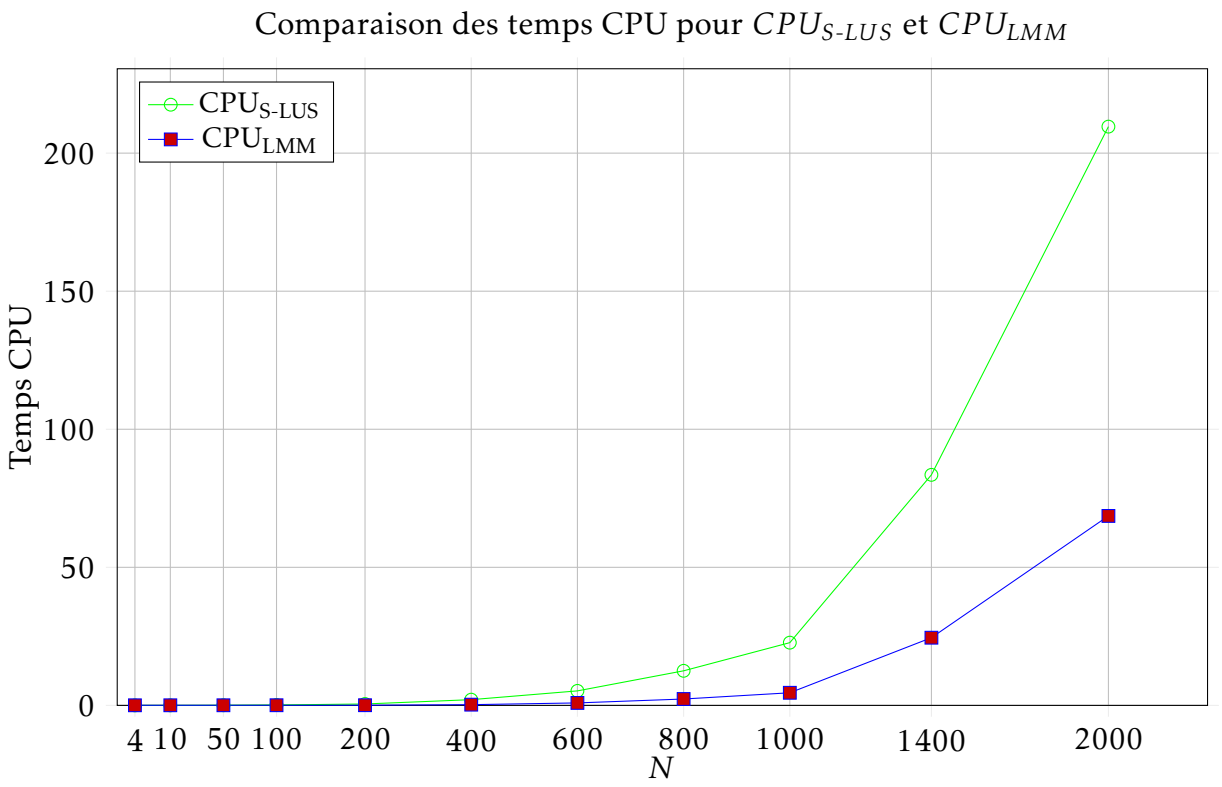


FIGURE 1.3 – Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $LX + XU = C$.

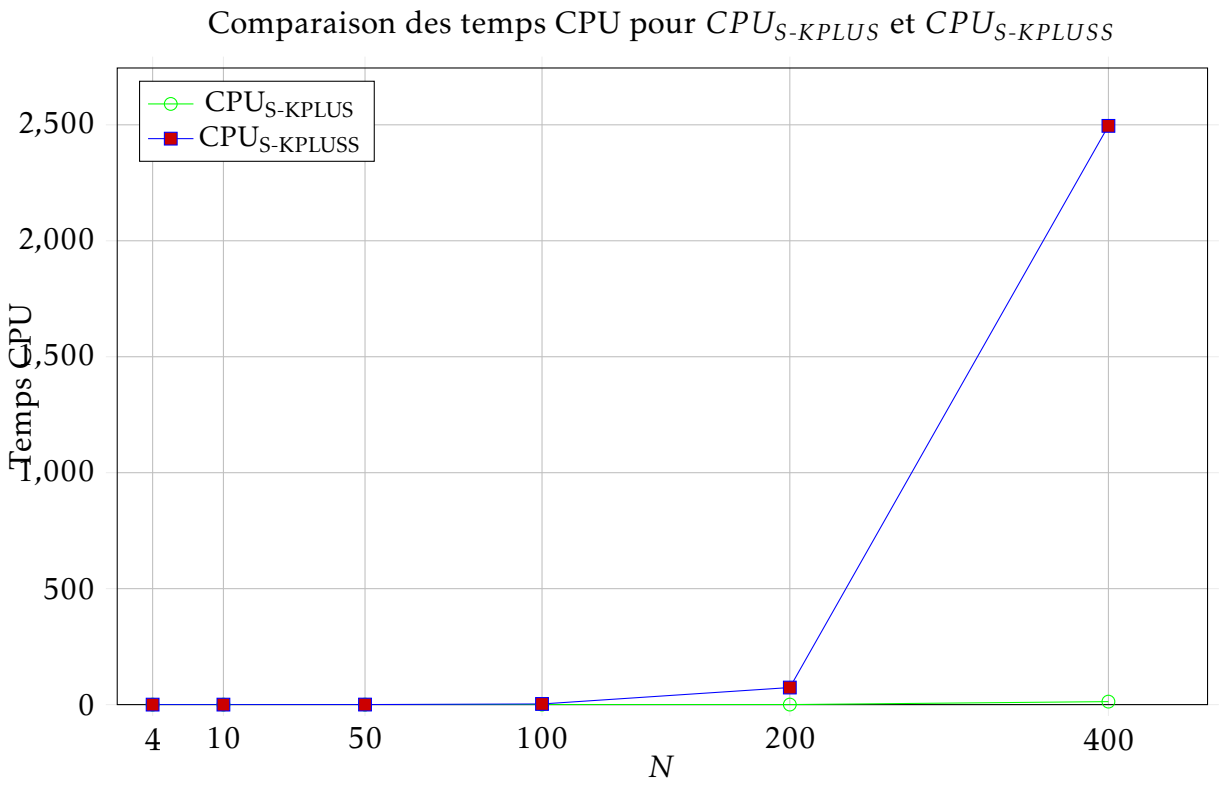


FIGURE 1.4 – Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $LX + XU = C$.

Méthodes numériques pour résouc l'équation de Lyapunov-Sylvester

2
Chapitre

L'équation de Lyapunov-Sylvester est une équation matricielle qui ne possède pas toujours de solution analytique explicite, en particulier lorsque les matrices sont de grande taille ou difficiles à diagonaliser. Pour cette raison, plusieurs méthodes numériques ont été développées en exploitant les propriétés des matrices et les techniques d'analyse numérique [2].

Parmi ces méthodes, l'algorithme de Bartels-Stewart est l'un des plus anciens et des plus utilisés. Cependant, il peut rencontrer des difficultés numériques lorsque les matrices possèdent des valeurs propres répétées ou très proches, ce qui peut entraîner une instabilité numérique et une diminution de la précision de la solution.

Pour surmonter ces limitations, l'algorithme Jordan-Schur a été développé[3] . Il repose sur la décomposition Jordan-Schur afin d'améliorer la stabilité numérique des solutions, notamment lorsque les valeurs propres sont proches. Cette approche permet d'obtenir une précision accrue, rendant l'algorithme plus fiable dans les cas où la méthode de Bartels-Stewart s'avère insuffisante.

I Algorithme de Bartels-Stewart

L'algorithme de Bartels-Stewart est une méthode numérique utilisée pour résoudre l'équation de Lyapunov-Sylvester sous la forme :

$$AX + XB = C.$$

où $A \in \mathcal{M}_n(\mathbb{R})$, $B \in \mathcal{M}_m(\mathbb{R})$ et $C \in \mathcal{M}_{n,m}(\mathbb{R})$ sont des matrices données, et $X \in \mathcal{M}_{n,m}(\mathbb{R})$ est la matrice inconnue. Cet algorithme repose sur la décomposition de Schur pour transformer les matrices A et B en formes triangulaires supérieures (ou quasi-triangulaires dans le cas réel), simplifiant ainsi l'équation en un système triangulaire résoluble par substitution arrière (Backward Substitution). Grâce à sa complexité optimale ($O(n^3)$) et à son utilisation deS techniques avancées comme l'algorithme QR et la transformation de Householder, il améliore la précision et la stabilité numérique, notamment dans les cas où les matrices possèdent des valeurs propres répétées ou très proches.

Définition 2.1 [19] [**Décomposition de Schur**] La décomposition de Schur est une méthode fondamentale en algèbre linéaire numérique qui permet de transformer une matrice carrée en une forme triangulaire (ou quasi-triangulaire dans le cas réel) en utilisant une matrice unitaire (ou orthogonale dans le cas réel). Cette décomposition est particulièrement utile pour simplifier les calculs matriciels, notamment dans la résolution d'équations matricielles comme celles de Lyapunov et Sylvester.
Il existe une décomposition de Schur de la forme :

$$A = QTQ^*,$$

où :

— Q est une matrice unitaire (c'est-à-dire $Q^*Q = I$, où Q^* est la conjuguée transposée de Q),

— T est une matrice triangulaire supérieure (ou quasi-triangulaire dans le cas réel), dont les éléments diagonaux sont les valeurs propres de A.

Dans le cas réel, si $A \in \mathbb{R}^{n\times n}$, la décomposition de Schur réelle s'écrit :

$$A = QTQ^T,$$

où :

— Q est une matrice orthogonale (c'est-à-dire $Q^TQ = I$),

— T est une matrice quasi-triangulaire supérieure.

Exemple 9

Soit la matrice A définie par :

$$A = \begin{pmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 4 \end{pmatrix}.$$

Les valeurs propres de A sont :

$$\lambda_1 = 6, \quad \lambda_2 = 3, \quad \lambda_3 = 3.$$

La décomposition de Schur de A est donnée par :

$$A = QTQ^T.$$

où :

$$Q = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \end{pmatrix}, \quad T = \begin{pmatrix} 6 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{pmatrix}.$$

Exemple 10

Soit la matrice A définie par :

$$A = \begin{pmatrix} 0 & 1 & 0 \\ -2 & -3 & 0 \\ 0 & 0 & 4 \end{pmatrix}.$$

Les valeurs propres de A sont :

$$\lambda_1 = -1, \quad \lambda_2 = -2, \quad \lambda_3 = 4.$$

La décomposition de Schur de A est donnée par :

$$A = QTQ^T.$$

où :

$$Q = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad T = \begin{pmatrix} -1 & 3 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 4 \end{pmatrix}.$$

Définition 2.2 [12] [**Décomposition QR**] La décomposition QR est une factorisation d'une matrice A sous la forme :

$$A = QR,$$

où Q est une matrice orthogonale et R est une matrice triangulaire supérieure. Cette décomposition est un outil fondamental en algèbre linéaire numérique, permettant la résolution stable des systèmes linéaires, le calcul des valeurs propres et l'optimisation numérique.
Il existe plusieurs méthodes pour la calculer :

1. Algorithme de Gram-Schmidt : basé sur l'orthonormalisation des colonnes de A.
2. Transformation de Householder : plus stable numériquement et adaptée aux grandes matrices.
3. Méthode de Givens : efficace pour les systèmes creux.

Définition 2.3 [20] [**Matrice de Hessenberg**] Une matrice de Hessenberg est une matrice carrée ayant une structure particulière qui facilite les calculs numériques, notamment pour le calcul des valeurs propres. Cette structure permet de réduire la complexité computationnelle lors de l'application d'algorithmes numériques, tels que l'algorithme QR.
Les matrices de Hessenberg peuvent être classées en deux types principaux.
Le premier type est la matrice de Hessenberg supérieure, qui prend la forme suivante :

$$H_U = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & \cdots & h_{1n} \\ h_{21} & h_{22} & h_{23} & h_{24} & \cdots & h_{2n} \\ 0 & h_{32} & h_{33} & h_{34} & \cdots & h_{3n} \\ 0 & 0 & h_{43} & h_{44} & \cdots & h_{4n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & h_{nn} \end{pmatrix}.$$

Le second type est la matrice de Hessenberg inférieure, qui prend la forme suivante :

$$H_L = \begin{pmatrix} h_{11} & h_{12} & 0 & 0 & \cdots & 0 \\ h_{21} & h_{22} & h_{23} & 0 & \cdots & 0 \\ h_{31} & h_{32} & h_{33} & h_{34} & \cdots & 0 \\ h_{41} & h_{42} & h_{43} & h_{44} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n1} & h_{n2} & h_{n3} & h_{n4} & \cdots & h_{nn} \end{pmatrix}.$$

Définition 2.4 [10][**Méthode de Householder**] Soit x un vecteur colonne arbitraire de dimension m et $\beta = \|x\|$ où $\|\cdot\|$ désigne la norme euclidienne.
Définissons le vecteur unitaire e_1 comme suit :

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Si x n'est pas colinéaire à βe_1, alors on définit le vecteur u comme :

$$u = x - \beta e_1,$$

et son vecteur normalisé associé :

$$v = \frac{u}{\|u\|},$$

La matrice de Householder associée est alors donnée par :

$$H_1 = I - 2vv^T,$$

où H_1 est une matrice orthogonale symétrique utilisée pour transformer une matrice en une forme triangulaire tout en préservant sa stabilité numérique.

Exemple 11

Soit la matrice A définie par :

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

Nous appliquons l'algorithme de Householder pour transformer A en une matrice triangulaire supérieure R, tout en construisant la matrice orthogonale Q telle que :

$$A = QR,$$

La première colonne de la matrice A :

$$x = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Calcul de β :

$$\beta = \|x\| = \sqrt{1^2 + 1^2 + 0^2} = \sqrt{2}.$$

Calcul du vecteur u :

$$u = x - \beta e_1 = \begin{pmatrix} 1 - \sqrt{2} \\ 1 \\ 0 \end{pmatrix}.$$

Puis, et sa vecteur unitaire :

$$v = \frac{u}{\|u\|} = \frac{1}{\sqrt{4 - 2\sqrt{2}}} \begin{pmatrix} 1 - \sqrt{2} \\ 1 \\ 0 \end{pmatrix}.$$

La matrice de Householder associée est donnée par :

$$H_1 = I - 2vv^T = \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Application de la première transformation :

$$A^{(1)} = H_1 A = \begin{pmatrix} \sqrt{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0 & 1 & 1 \end{pmatrix}.$$

On considère maintenant la deuxième colonne de $A^{(1)}$, à partir de la sous-matrice 2×1 suivante :

$$x' = \begin{pmatrix} -\frac{\sqrt{2}}{2} \\ 1 \end{pmatrix}.$$

Calcul de β' :

$$\beta' = \|x'\| = \sqrt{\left(-\frac{\sqrt{2}}{2}\right)^2 + 1^2} = \sqrt{\frac{2}{4} + 1} = \sqrt{\frac{3}{2}}.$$

Calcul du vecteur u' :

$$u' = x' - \beta' e_1' = \begin{pmatrix} -\frac{\sqrt{2}}{2} \\ 1 \end{pmatrix} - \sqrt{\frac{3}{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{\sqrt{2}}{2} - \sqrt{\frac{3}{2}} \\ 1 \end{pmatrix}.$$

La matrice de Householder correspondante est :

$$H_2' = I - 2v'v'^T = I_2 - 2\frac{u'u'^T}{u'^T u'}.$$

Ce qui donne la matrice H_2' suivante :

$$H_2' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{3} & \frac{2\sqrt{2}+4\sqrt{3}}{18+4\sqrt{6}} \\ 0 & \frac{2\sqrt{2}+4\sqrt{3}}{18+4\sqrt{6}} & \frac{1}{3} \end{pmatrix}.$$

Le produit final donne la matrice triangulaire R :

$$R = H_2 A^{(1)} = \begin{pmatrix} \sqrt{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 0 & \sqrt{\frac{3}{2}} & \frac{1}{\sqrt{6}} \\ 0 & 0 & \frac{2}{\sqrt{3}} \end{pmatrix}.$$

La matrice orthogonale Q est obtenue par le produit :

$$Q = H_1 H_2' = \begin{pmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} & \frac{\sqrt{3}}{3} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{6}}{6} & -\frac{\sqrt{3}}{3} \\ 0 & \frac{\sqrt{6}}{3} & \frac{\sqrt{3}}{3} \end{pmatrix}.$$

1 Résolution par la méthode Bartels-Stewart

On utilise la décomposition de Schur pour transformer les matrices A et B en formes triangulaires supérieures :

$$A = UDU^T, \quad B = VD'V^T$$

où :

— U et V sont des matrices orthogonales ($U^T U = I$ et $V^T V = I$),

— D une matrice triangulaire inférieure, et D' une matrice triangulaire supérieure.

Les matrices D et D' ont la structure suivante :

$$D = \begin{pmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ d_{21} & d_{22} & 0 & \cdots & 0 \\ d_{31} & d_{32} & d_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \cdots & d_{nn} \end{pmatrix}, \quad D' = \begin{pmatrix} d'_{11} & d'_{12} & d'_{13} & \cdots & d'_{1m} \\ 0 & d'_{22} & d'_{23} & \cdots & d'_{2m} \\ 0 & 0 & d'_{33} & \cdots & d'_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d'_{mm} \end{pmatrix}.$$

On substitue la décomposition de Schur dans l'équation d'origine $AX + XB = C$:

$$UDU^T X + XVD'V^T = C,$$

On multiplie l'équation par U^T à gauche et par V à droite :

$$U^T(UDU^T X)V + U^T(XVD'V^T)V = U^T CV,$$

On simplifie l'expression en utilisant $U^T U = I$ et $V^T V = I$:

$$D(U^T XV) + (U^T XV)D' = U^T CV,$$

On définit $Y = U^T XV$ et $\tilde{C} = U^T CV$, ce qui donne l'équation :

$$DY + YD' = \tilde{C},$$

La matrice Y est calculée élément par élément.
Après avoir calculé Y, on retrouve X en utilisant la relation : $X = UYV^T$.

2 Complexité algorithmique

— Dans la méthode Bartels-Stewart, la décomposition de Schur des matrices A et B est calculée en utilisant l'algorithme QR, qui nécessite une complexité de $O(n^3)$ pour chaque matrice[2].

— Comme l'équation de Sylvester implique deux matrices (A et B), la complexité totale de la décomposition de Schur dans ce cas est de $O(n^3)$.

— En revanche, pour l'équation de Lyapunov, où une seule matrice A est décomposée, la complexité de la décomposition de Schur est de $O(n^3)$.

— Après avoir transformé les matrices A et B en une forme quasi triangulaire supérieure, l'équation de Sylvester AX + XB = C est convertie en un système triangulaire en multipliant les matrices pour changer de base, et cette étape nécessite une complexité de $O(n^3)$.

— Dans le cas de l'équation de Lyapunov $AX + XA^T = C$, lorsque A est sous forme quasi triangulaire supérieure de Schur, A^T est également quasi triangulaire, et l'équation est convertie en un système triangulaire en utilisant la multiplication de matrices, ce qui nécessite une complexité de $O(n^3)$.

3 Avantages et inconvénients

Avantages

— Haute précision numérique : La méthode repose sur la *décomposition de Schur*, ce qui la rend stable numériquement lorsqu'on traite des matrices dont les valeurs propres ne sont pas trop proches.

— Efficacité computationnelle : Comparée à d'autres méthodes directes, elle a une *complexité de* $O(n^3)$, ce qui est raisonnable pour des matrices de taille moyenne.

Inconvénients

— Sensibilité aux valeurs propres proches ou répétées : Si les valeurs propres de la matrice sont très proches, cela peut entraîner des erreurs numériques ou nécessiter des ajustements supplémentaires.

— Inefficace pour les matrices à grand taille : En raison de sa complexité computationnelle, cette méthode devient impraticable pour des *matrices de grande taille*, car elle consomme beaucoup de mémoire et de temps de calcul.

— Dépendance à la décomposition de Schur : La *décomposition de Schur* est elle-même coûteuse en calcul, ce qui la rend moins adaptée aux approches itératives ou approximatives nécessitant une rapidité accrue.

Remarque 2.1. *La méthode classique de Bartels-Stewart présente des limitations notables lorsqu'elle est appliquée à des matrices possédant des valeurs propres rapprochées ou multiples, conduisant à des instabilités numériques et à une perte de précision. Pour pallier ces inconvénients, la méthode Jordan-Schur a été proposée, reposant sur une décomposition de Jordan-Schur plutôt que sur la décomposition de Schur traditionnelle. Cette approche améliore significativement la gestion des valeurs propres proches et renforce la stabilité numérique, offrant ainsi une solution plus robuste pour la résolution des équations de Lyapunov et de Sylvester avec une précision accrue.*

II Algorithme de Jordan-Schur

Définition 2.5 [3, 1] L'algorithme de Jordan-Schur est une méthode numérique avancée pour résoudre les équations de Lyapunov et de Sylvester. Il repose sur la décomposition Jordan-Schur, une décomposition combinant Schur et les blocs de Jordan, permettant une gestion efficace des matrices à valeurs propres multiples ou rapprochées. Cette approche améliore la stabilité numérique et réduit les erreurs d'arrondi, offrant ainsi des solutions plus précises et robustes, notamment dans les cas où les méthodes classiques comme Bartels-Stewart sont inefficaces.

Définition 2.6 [18]

[**Blocs de Jordan**] Bloc de Jordan est une sous-matrice carrée associée à une valeur propre d'une matrice carrée. Il représente la structure de la matrice lorsqu'elle est mise sous **forme de Jordan**, qui est une forme canonique simplifiant l'étude des matrices tout en conservant leurs valeurs propres.
La matrice de Jordan est composée de blocs de Jordan $J_k(\lambda)$, où chaque bloc représente une valeur propre λ avec une certaine multiplicité. La forme générale d'une matrice de Jordan est :

$$J = \begin{pmatrix} J_{k_1}(\lambda_1) & 0 & 0 & \dots & 0 \\ 0 & J_{k_2}(\lambda_2) & 0 & \dots & 0 \\ 0 & 0 & J_{k_3}(\lambda_3) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & J_{k_m}(\lambda_m) \end{pmatrix}.$$

où chaque bloc de Jordan $J_k(\lambda)$ est de la forme :

$$J_k(\lambda) = \begin{pmatrix} \lambda & 1 & 0 & \dots & 0 \\ 0 & \lambda & 1 & \dots & 0 \\ 0 & 0 & \lambda & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & \dots & \lambda \end{pmatrix}_{k \times k}.$$

Dans chaque bloc de Jordan :

- La diagonale principale contient uniquement la valeur propre λ.
- La diagonale juste au-dessus de la principale contient des 1.
- Toutes les autres entrées sont nulles.

  Le nombre k représente la taille du bloc de Jordan, qui est lié à la multiplicité des valeurs propres :
- Si la valeur propre λ est répétée k fois mais possède moins de vecteurs propres que k, alors elle forme un **bloc de Jordan de taille** k.
- Lorsque chaque valeur propre a un nombre suffisant de vecteurs propres, les blocs sont de taille 1 et la matrice est diagonalisable.

Définition 2.7 [15] [**Décomposition Jordan-Schur**] La décomposition Jordan-Schur est une généralisation de la décomposition de Schur, permettant de transformer toute matrice carrée A en une matrice quasi-triangulaire J prenant explicitement en compte la structure de Jordan, à l'aide d'une matrice orthogonale ou unitaire U, selon la relation :

$$A = UJU^*,$$

où U^* désigne la matrice adjointe de U. La matrice J est quasi-triangulaire supérieure, avec les valeurs propres de A sur sa diagonale principale, ainsi que des blocs de Jordan, représentant les valeurs propres multiples ou très proches.
Cette décomposition est particulièrement utilisée pour améliorer la précision nu-

mérique lors de la résolution des équations matricielles, telles que l'équation de Lyapunov-Sylvester, en permettant une meilleure prise en compte de la structure spectrale de la matrice.
Dans le cas réel où $A \in M_n(\mathbb{R})$, la décomposition prend la forme suivante :

$$A = UJU^T,$$

où U est une matrice **orthogonale**. Étant donné que A et J sont **similaires**, elles possèdent les mêmes valeurs propres. L'intégration des blocs de **Jordan** permet d'améliorer la stabilité et la précision numérique, en particulier pour les matrices ayant des valeurs propres proches.

Exemple 12

Soit A une matrice carrée 3×3 donnée par :

$$A = \begin{pmatrix} 4 & 1 & 2 \\ 0 & 3 & 1 \\ 1 & 0 & 4 \end{pmatrix}.$$

$U^T AU = J$.
Calcul des valeurs propres :
les racines de polynôme caractéristique : $\det(A - \lambda I) = 0$.

— $\lambda_1 = 3$ (multiplicité algébrique = 2)

— $\lambda_2 = 5$ (multiplicité algébrique = 1)

Forme de Jordan :
Bloc de Jordan pour $\lambda_1 = 3$:

— Un seul bloc de taille 2×2

$$J_2(3) = \begin{pmatrix} 3 & 1 \\ 0 & 3 \end{pmatrix}.$$

Bloc de Jordan pour $\lambda_2 = 5$:

— Bloc 1×1

$$J_2(5) = [5].$$

$$J = \begin{pmatrix} 3 & 1 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{pmatrix}.$$

Matrice de passage U :

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ 0 & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} \end{pmatrix}.$$

1 Résolution par la méthode Jordan-Schur

L'équation de Sylvester s'écrit sous la forme :

$$AX + XB = C.$$

On applique la décomposition **Jordan-Schur** aux matrices A et B :

$$A = UDU^T, \quad B = VD'V^T,$$

où :

— U et V sont des matrices orthogonales ($U^TU = I$ et $V^TV = I$).

— D et D' sont des matrices quasi-triangulaires (formes Jordan-Schur de A et B).

Les matrices D et D' ont la structure suivante :

$$D = \begin{pmatrix} d_{11} & 0 & 0 & \cdots & 0 \\ d_{21} & d_{22} & 0 & \cdots & 0 \\ d_{31} & d_{32} & d_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & d_{n3} & \cdots & d_{nn} \end{pmatrix}, \quad D' = \begin{pmatrix} d'_{11} & d'_{12} & d'_{13} & \cdots & d'_{1m} \\ 0 & d'_{22} & d'_{23} & \cdots & d'_{2m} \\ 0 & 0 & d'_{33} & \cdots & d'_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & d'_{mm} \end{pmatrix}.$$

où les blocs d_{ii} et d'_{ii} sont soit des valeurs propres simples, soit des blocs de Jordan.
On multiplie l'équation originale à gauche par U^T et à droite par V :

$$U^T(AX + XB)V = U^TCV,$$

En utilisant les décompositions Jordan-Schur de A et B, on obtient :

$$U^T(UDU^TX + XVD'V^T)V = U^TCV,$$

Puisque $U^TU = I$ et $V^TV = I$, on simplifie :

$$D(U^TXV) + (U^TXV)D' = U^TCV,$$

On définit :

$$X' = U^TXV, \quad C' = U^TCV,$$

Ainsi, l'équation devient :

$$DX' + X'D' = C',$$

Après avoir résolu pour X', on obtient la solution finale de X en utilisant :

$$X = UX'V^T.$$

2 Complexité algorithmique

— L'analyse Jordan-Schur des matrices A et B nécessite une complexité algorithmique de $O(n^3)$ pour chaque matrice, ce qui rend la complexité totale de cette étape $O(n^3)$.

— La réduction d'une matrice à sa forme de Jordan est une opération numériquement instable et plus coûteuse que la décomposition de Schur, pouvant atteindre une complexité de $O(n^4)$ dans certains cas.

— Lorsqu'on exprime A et B sous forme Jordan-Schur, l'équation de Sylvester $AX + XB = C$ est transformée en un système triangulaire ou quasi-triangulaire via des multiplications matricielles avec une complexité de $O(n^3)$.

— Si le système obtenu est entièrement triangulaire, il peut être résolu par substitution arrière avec une complexité de $O(n^3)$, tandis qu'en présence de blocs de Jordan, la résolution devient plus complexe et peut nécessiter des opérations supplémentaires.

— Lorsque seule la décomposition Jordan-Schur est utilisée sans calcul explicite de la forme de Jordan, la complexité globale reste $O(n^3)$, tandis que si la forme de Jordan est explicitement calculée, la complexité peut s'élever à $O(n^4)$ dans certains cas.

3 Avantages et inconvénients

Avantages :

— Plus efficace que l'algorithme de Bartels-Stewart lorsque les valeurs propres ne sont pas distinctes.

— Contrairement à Bartels-Stewart, qui ne résout que l'équation de Lyapunov, l'algorithme Jordan-Schur peut résoudre les équations de Lyapunov et de Sylvester.

— Plus performant avec les matrices difficiles à trianguliariser.

— Réduit certaines erreurs numériques pour des valeurs propres proches.

Ionvénients :

— Nécessite des calculs supplémentaires, augmentant le temps d'exécution par rapport à Bartels-Stewart.

— Moins couramment utilisée en pratique par rapport à **Bartels-Stewart**, en raison de sa complexité.

III Approche itérative avancée pour la résolution de l'équation de Lyapunov-Sylvester

Définition 2.8 [21, 4] Dans ce travail, nous proposons une nouvelle méthode basée sur une décomposition triangulaire itérative pour résoudre l'équation de Lyapunov-Sylvester $AX+XB=C$. L'idée principale est de décomposer les matrices A et B en une somme de matrices triangulaires (inférieure ou supérieure) et de matrices résiduelles, puis d'exploiter cette structure pour concevoir un processus itératif adapté à la résolution du système résultant. Cette méthode se distingue par sa facilité d'implémentation, son efficacité numérique et sa capacité à traiter des matrices de grande taille, garantissant ainsi une convergence rapide et une grande précision dans les résultats.

Pour résoudre l'équation de Lyapunov-Sylvester :

$$AX + XB = C,$$

nous décomposons A et B en une somme de matrices triangulaires et de termes résiduels. Plus précisément :

$$A = L + E, \quad B = U + F,$$

où :

— L est la partie triangulaire inférieure de A, définie comme suit pour tous $i, j \in \{1, \ldots, n\}$:

$$l_{ij} = \begin{cases} \alpha_{ij}, & \text{si } i \geq j, \\ 0, & \text{sinon.} \end{cases}$$

— E est la partie complémentaire (strictement triangulaire supérieure) de A, définie comme suit pour tous $i, j \in \{1, \ldots, n\}$:

$$e_{ij} = \begin{cases} \alpha_{ij}, & \text{si } i < j, \\ 0, & \text{sinon.} \end{cases}$$

— U est la partie triangulaire supérieure de B, définie comme suit pour tous $i, j \in \{1, \ldots, n\}$:

$$u_{ij} = \begin{cases} \beta_{ij}, & \text{si } i \leq j, \\ 0, & \text{sinon.} \end{cases}$$

— F est la partie complémentaire (strictement triangulaire inférieure) de B, définie comme suit pour tous $i, j \in \{1, \ldots, n\}$:

$$f_{ij} = \begin{cases} \beta_{ij}, & \text{si } i > j, \\ 0, & \text{sinon.} \end{cases}$$

En substituant les décompositions de A et B dans l'équation originale, il vient :

$$(L+E)X + X(U+F) = C.$$

En réorganisant les termes pour isoler les parties principales associées aux matrices triangulaires, il vient :

$$LX + XU = C - EX - XF.$$

Puis on réorganise pour faire apparaître un schéma itératif de point fixe :

$$LX^{(k+1)} + X^{(k+1)}U = C - EX^{(k)} - X^{(k)}F.$$

où :

— $X^{(k)}$: estimation de la solution à l'itération k

— $X^{(k+1)}$: estimation de la solution à l'itération k+1

1. Initialisation : $X^{(0)}$ = donnée
2. Résolution du système triangulaire
3. Test de convergence : $\|X^{(k+1)} - X^{(k)}\| < \epsilon$.

1 Convergence de la méthode itérative

Preuve 3. *[2, 21] Considérons l'équation itérative :*

$$LX^{(k+1)} + X^{(k+1)}U = C - EX^{(k)} - X^{(k)}F. \tag{1}$$

Nous procédons à une analyse spectrale de l'opérateur d'itération. En vectorisant l'équation (1) à l'aide du produit de Kronecker, on obtient :

$$(I \otimes L + U^T \otimes I) \cdot vec(X^{(k+1)}) = vec(C) - (I \otimes E + F^T \otimes I) \cdot vec(X^{(k)}).$$

En isolant $vec(X^{(k+1)})$, on a :

$$vec(X^{(k+1)}) = -(I \otimes L + U^T \otimes I)^{-1}(I \otimes E + F^T \otimes I) \cdot vec(X^{(k)}) + (I \otimes L + U^T \otimes I)^{-1} \cdot vec(C).$$

La matrice d'itération G est définie par :

$$G = -(I \otimes L + U^T \otimes I)^{-1}(I \otimes E + F^T \otimes I).$$

Ainsi, le schéma peut s'écrire sous la forme :

$$vec(X^{(k+1)}) = G \cdot vec(X^{(k)}) + (I \otimes L + U^T \otimes I)^{-1} \cdot vec(C).$$

$(I \otimes L + U^T \otimes I)^{-1} \cdot vec(C)$=*terme constant*
La convergence de la méthode dépend du rayon spectral $\rho(G)$ de la matrice G.

***Condition nécessaire et suffisante de convergence** : la méthode converge si et seulement si :*

$$\rho(G) = \rho\Big((I \otimes L + U^T \otimes I)^{-1}(I \otimes E + F^T \otimes I)\Big) < 1.$$

Une estimation pratique du taux de convergence peut être donnée par :

$$\rho(G) \approx \max_i |\lambda_i(G)|,$$

où $\lambda_i(G)$ désignent les valeurs propres de G.
D'un autre côté, la convergence peut également être garantie si la condition $\|G\| < 1$ est satisfaite, c'est-à-dire si la norme de la matrice G est strictement inférieure à 1.

Par conséquent, deux critères peuvent être utilisés pour assurer la convergence de la méthode :

- *Soit le rayon spectral $\rho(G)$ est strictement inférieur à 1,*
- *Soit la norme matricielle $\|G\|$ est strictement inférieure à 1.*

Preuve 4. *[21] Considérons l'équation itérative :*

$$LX^{(k+1)} + X^{(k+1)}U = C - EX^{(k)} - X^{(k)}F.$$

On définit un opérateur G par :

$$G(X^{(k)}) = T(C - EX^{(k)} - X^{(k)}F),$$

Par conséquent, on peut écrire :

$$X^{(k+1)} = T(C - EX^{(k)} - X^{(k)}F),$$

D'après le théorème du point fixe :

$$X^{(k+1)} = G(X^{(k)}),$$

On étudie la différentielle de G :

$$dG(X)[\Delta X] = -T(E\Delta X + \Delta X F),$$

Ce qui donne en notation de Kronecker :

$$dG = -T \circ (E \otimes I + I \otimes F^{\top}).$$

La convergence est assurée si le rayon spectral de dG est strictement inférieur à 1 :

$$\rho(dG) < 1.$$

Condition Suffisante de Convergence :
Une majoration pratique consiste à estimer la norme subordonnée de dG :

$$\|dG\| \leq \|T\| \cdot (\|E\| + \|F\|) < 1.$$

L'opérateur T est donné par :

$$T = (L \otimes I + I \otimes U^{\top})^{-1},$$

et on peut estimer sa norme par :

$$\|T\| \leq \|(L \otimes I + I \otimes U^{\top})^{-1}\|.$$

Si L et U sont strictement diagonalement dominantes, on a :

$$\|L^{-1}\| \leq \frac{1}{\min_i(|L_{ii}| - \sum_{j \neq i}|L_{ij}|)}, \quad \|U^{-1}\| \leq \frac{1}{\min_i(|U_{ii}| - \sum_{j \neq i}|U_{ij}|)}. \quad i, j \in \{1, \ldots, n\}$$

Ainsi,$\|T\|$ Peut être majorée.

Les différents algorithmes pour résoudre le problème $AX + XB = C$ sont donnés :

— Méthode itérative de Sylvester (**IMS**).

— Méthode Lyap de MATLAB (**LMM**).

— Sylvester–Produit de Kronecker (**S-KPI**).

— Sylvester–Produit de Kronecker (**S-KPL**).

Algorithm 5: IMS

```
1  function IMS
2  input :A,B,C;
3  output :X;
4  n=length(A);
5  X=zeros(n,n);
6  X0=zeros(n,n);
7  L=tril(A);U=triu(A);
8  E=A-L; F=A-U;
9  S=zeros(n,n);
10 nmax=200;
11 for j=1 :n do
12 |  for i=1 :n do
13 |  |  X(i,j)=(C(i,j)+S(i,j)-L(i,1 :i-1)*X(1 :i-1,j)-X(i,1 :j-1)*U(1 :j-
   |  |    1,j))/(L(i,i)+U(j,j));
14 |  end
15 end
16 if max(max(abs(X-X0)))<tol then
17 |  break end
18 end
19 X0=X;
20 S=-E*X0-X0*F;
21 end disp(X)
```

Algorithm 6: LMM

```
1 function X=LMM(A,B,C)
2 input :A,B,C;
3 output :X;
4 n=length(A);
5 X=zeros(n,n);
6 X=lyap(A,B,-C);
```

Algorithm 7: S-KPI

```
function X=S-KPI(A,B,C)
input :A,B,C;
output :X;
n=length(A);
X=zeros(n,n);
I=eye(n,n);
G=kron(I,L)+kron(U',I);
c=reshape(C,[ ],1);
x=inv(G)*c;
Z=reshape(x,[ ],n);
```

Algorithm 8: S-KPL

```
function X=S-KPL(A,B,C)
input :A,B,C;
output :X;
n=length(A);
E=triu(A,1);L=tril(A);
U=triu(B); F=tril(B,-1);
I= eye(n);
G=kron(I,L)+kron(U',I);
M=kron(I,E)+kron(F',I);
c=reshape(C,[],1);
nmax=200;tol=0.0001;x0 = zeros(n*n,1);
x=zeros(n*n,1);
for k=1 :nmax do
    b=c-M*x0; x=G;
    if max(abs(x-x0))<tol then
        break
        end
    end
    x0=x;
    end
end
Y=reshape(x,[],n);
T1=toc
```

2 Comparaison Numérique

D'après les figures et le tableau ci-dessous, on compare les temps CPU pour différentes méthodes de résolution de l'équation $AX + XB = C$, en considérant plusieurs valeurs de la tolérance numérique. Les résultats montrent que les méthodes IMS et LMM sont nettement plus rapides et plus stables, en particulier pour les grandes tailles de matrices.

En ce qui concerne la méthode S-KPL, elle présente des limites de performance lorsque la taille de la matrice N augmente, avec des temps d'exécution très élevés ou une absence de convergence à partir de $N \geq 150$.

TABLE 2.1 – Comparaison des temps CPU pour différentes méthodes de résolution de $AX + XB = C$.

N	tol = 10^{-2}		tol = 10^{-4}		tol = 10^{-6}		CPU_{LMM}	CPU_{S-KPL}
	CPU_{IMS}	CPU_{S-KPI}	CPU_{IMS}	CPU_{S-KPI}	CPU_{IMS}	CPU_{S-KPI}		
20	1.2×10^{-2}	1.4×10^{-4}	2.3×10^{-2}	2.9×10^{-4}	3.4×10^{-2}	3.8×10^{-4}	1.9×10^{-2}	1.4×10^{-2}
40	2.4×10^{-2}	9.3×10^{-4}	4.2×10^{-2}	1.4×10^{-3}	0.1035	2×10^{-3}	2.6×10^{-2}	1.2×10^{-1}
80	4.9×10^{-2}	2.4×10^{-3}	1.09×10^{-1}	1.3×10^{-2}	3.6×10^{-1}	1.3×10^{-2}	3.2×10^{-2}	4.14
100	6.8×10^{-2}	4.2×10^{-4}	1.5×10^{-1}	1.7×10^{-2}	5.1×10^{-1}	2.5×10^{-2}	4.6×10^{-2}	15.89
150	1.8×10^{-1}	1.5×10^{-2}	3.5×10^{-1}	6.4×10^{-2}	8.6×10^{-1}	9.3×10^{-2}	7.1×10^{-2}	/
200	4.9×10^{-1}	3.6×10^{-2}	5.6×10^{-1}	1.5×10^{-1}	1.50	2.1×10^{-1}	14.4×10^{-2}	/
400	2.23	12.5	3.71	11.02	5.67	7.201	1.01	/
700	6.04	/	9.61	/	13.93	/	2.97	/
900	10.94	/	21.14	/	30.32	/	5.12	/
1000	16.65	/	29.76	/	42.63	/	7.002	/

Il convient également de noter que la structure creuse (*sparse*) a été utilisée dans l'implémentation de la méthode S-KPI, dans le but de réduire l'utilisation de la mémoire et d'optimiser les performances. Toutefois, cette méthode cesse de fonctionner à partir de $N = 400$, ce qui reflète ses limites face aux grandes dimensions.
Ainsi, les méthodes IMS et LMM apparaissent comme les plus appropriées pour les applications de grande dimension, en raison de leur efficacité et de leur stabilité numérique.

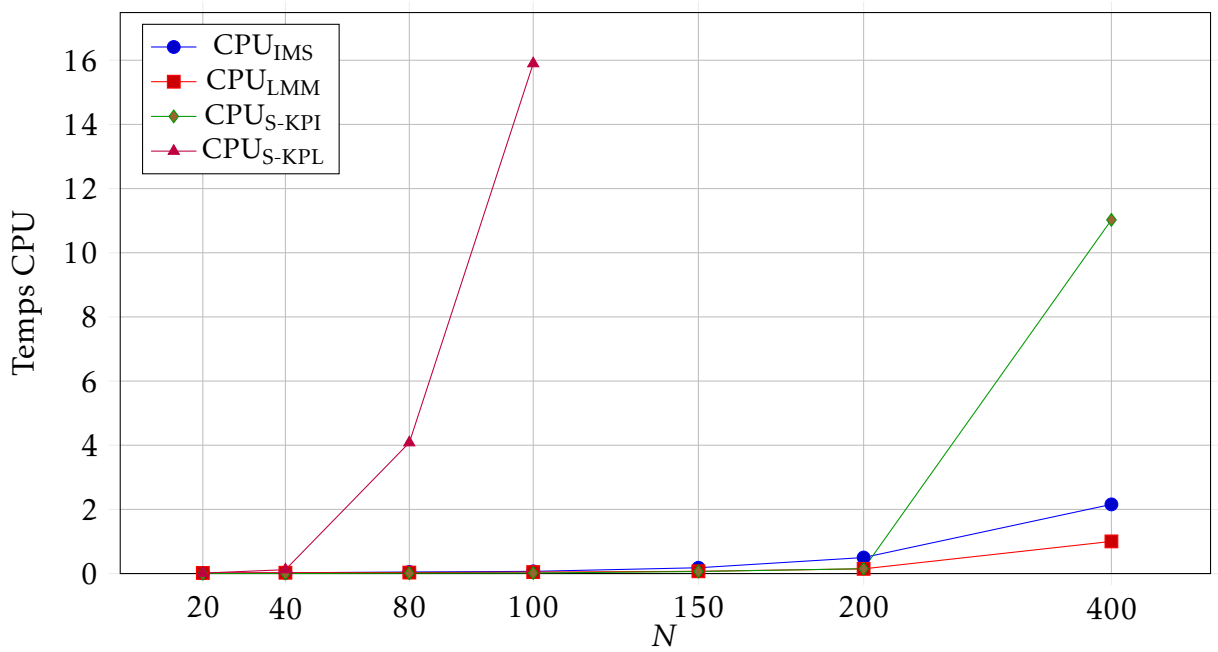


FIGURE 2.1 – Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $AX + XB = C$, avec tol = 10^{-2}.

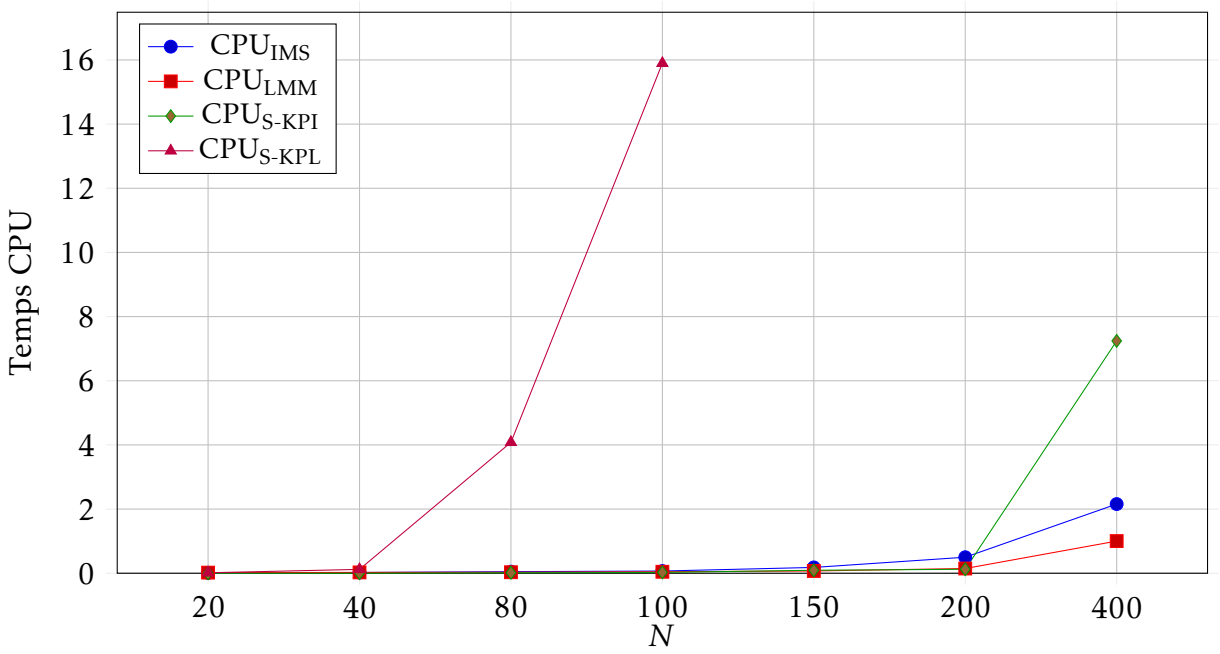


FIGURE 2.2 – Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $AX + XB = C$, avec tol = 10^{-4}.

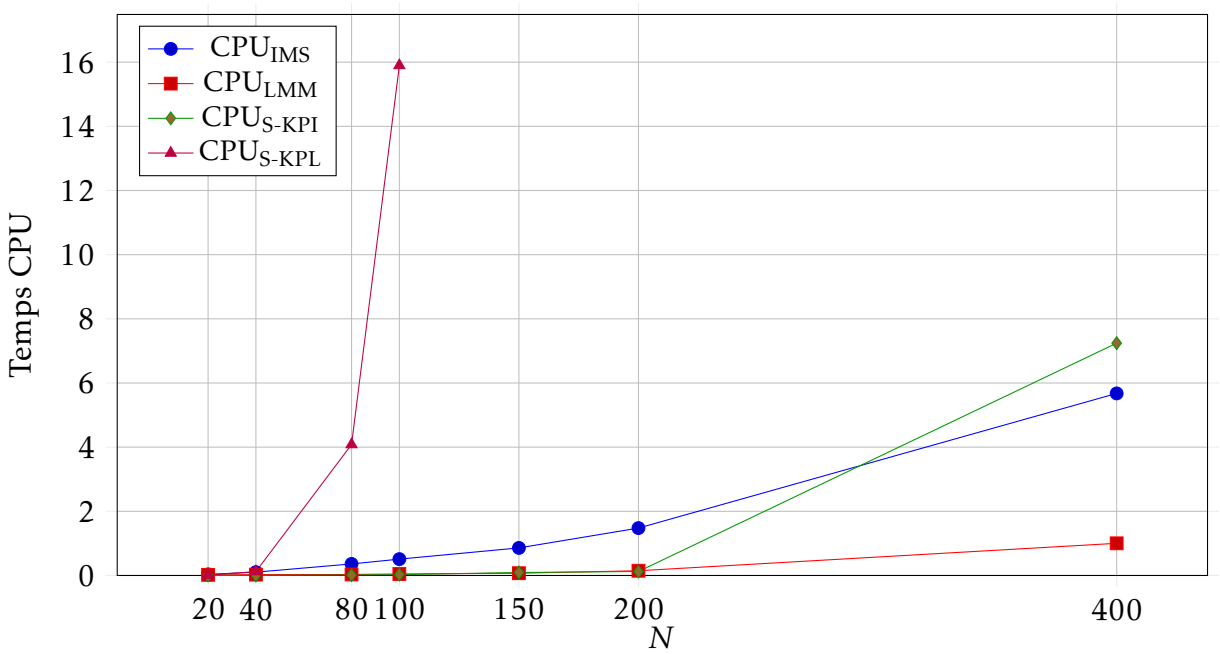


FIGURE 2.3 – Comparaison des temps CPU pour différentes méthodes de résolution de l'équation $AX + XB = C$, avec tol = 10^{-6}.

Conclusion

Ce mémoire a d'abord établi les fondements théoriques de l'équation de Lyapunov-Sylvester, en analysant les conditions d'existence et d'unicité des solutions à travers les propriétés spectrales et le produit de Kronecker. Cette analyse a mis en évidence des cas particuliers (matrices diagonales, matrices de Toeplitz, etc.) permettant des résolutions exactes ou simplifiées grâce à la structure des matrices impliquées.

Par la suite, ce travail a exploré les méthodes numériques pour résoudre l'équation de Lyapunov-Sylvester, en mettant l'accent sur leur efficacité et leur stabilité. Les méthodes classiques (Bartels-Stewart, Jordan-Schur) ont été analysées, révélant leurs limites face aux matrices de grande taille.

Nous développons une approche itérative, basée sur la décomposition triangulaire, pour améliorer les performances computationnelles. Les tests numériques ont confirmé sa rapidité et sa précision, notamment pour des systèmes de grande dimension.

Cette étude contribue ainsi au domaine du calcul matriciel et ouvre des perspectives prometteuses pour des applications en contrôle optimal et en modélisation des systèmes dynamiques. Des travaux futurs pourraient étendre cette méthode à des cas plus complexes, notamment les équations matricielles non linéaires. .

Bibliographie

[1] Barraud, A.Y. (1977). *A numerical algorithm to solve A XA - X = Q*. IEEE Transactions on Automatic Control, AC-22(5), 883-885.

[2] Bartels, R. H., & Stewart, G. W. (1972). *Solution of the Matrix Equation $AX + XB = C$*. Communications of the ACM, **15**(9), 820–826.

[3] Benner, P., & Mehrmann, V. (2005). *A Jordan-Schur Algorithm for Numerical Solution of Sylvester Equations*. Linear Algebra and Its Applications, 404, 44-67.

[4] D. A. Bini, B. Meini, *Numerical Solution of Algebraic Riccati Equations*, SIAM, 2012.

[5] D. Coppersmith et S. Winograd, *Matrix Multiplication via Arithmetic Progressions*, Journal of Symbolic Computation, vol. 9, pp. 251–280, 1987.

[6] J. Dongarra, J. Du Croz, S. Hammarling, et R. Hanson, *An Extended Set of FORTRAN Basic Linear Algebra Subprograms*, ACM Transactions on Mathematical Software, vol. 14, no. 1, pp. 1–17, 1990.

[7] Gershgorin, S. (1931). Über die Abgrenzung der Eigenwerte einer Matrix. *Izvestiya Akademii Nauk SSSR*, **7**, 749-754.

[8] Golub, G.H., Nash, S. & Van Loan, C.F. (1979). *A Hessenberg-Schur method for the problem $AX + XB = C$*. IEEE Transactions on Automatic Control, AC-24(5), 909-913.

[9] K. Goto et R. A. van de Geijn, *Anatomy of High-Performance Matrix Multiplication*, ACM Transactions on Mathematical Software, vol. 34, no. 3, pp. 12 :1–12 :25, 2008.

[10] Grubišić, L. et Kressner, D. (2014). On the eigenvalue decay of solutions to operator Lyapunov equations. *Systems & Control Letters*, **73**, 42–47.

[11] Hammarling, S. J. (1982). Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA Journal of Numerical Analysis*, **2**(3), 303-323.

[12] Householder, A. S. (1964). *The Theory of Matrices in Numerical Analysis*. Blaisdell.

[13] Khalil, H. K. (2015). *Nonlinear Systems* (3ème éd.). Prentice Hall.

[14] D. B. Kirk et W. W. Hwu, *Programming Massively Parallel Processors – A Hands-on Approach*, Morgan Kaufmann, 3e édition, 2016.

[15] Kirrinnis, Fast algorithms for the Sylvester equation AX- XBT= C, Theoretical Computer Science 259.1-2 (2001) : 623-638.

[16] Lancaster, P., & Rodman, L. (1995). *Algebraic Riccati Equations*. Oxford University Press.

[17] F. Le Gall, *Powers of Tensors and Fast Matrix Multiplication*, International Symposium on Symbolic and Algebraic Computation (ISSAC), pp. 296–303, 2014.

[18] Penzl, T. (1998). *Numerical solution of generalized Lyapunov equations*. Advances in Computational Mathematics, 8(1-2), 33-48.

[19] Roth, W. E. (1952). The equations $AX - YB = C$ and $AX - XB = C$ in matrices. *Proceedings of the American Mathematical Society*, **3**(3), 392-396.

[20] Rutherford, D. E. (1932). On the solution of the matrix equation $AX + XB = C$. *Nederl. Akad. Wetensch. Proc. Ser. A*, **35**, 53-59.

[21] Saad, Y. (2003). *Iterative methods for sparse linear systems* (2nd ed.). Society for Industrial and Applied Mathematics (SIAM).

[22] V. Strassen, *Gaussian Elimination is Not Optimal*, Numerische Mathematik, vol. 13, pp. 354–356, 1969.

[23] Trefethen, L. N., & Bau, D. (1997). *Numerical Linear Algebra*. SIAM.