Sur des méthodes préservant les structures d'une classe de matrices structurées

Haithem Ben Kahla

THÈSE DE DOCTORAT EN COTUTELLE

pour obtenir le grade de docteur délivré par

Université Littoral Côte d'Opale - France

et

Université de Tunis El Manar - Tunisie

Spécialité : "Mathématiques Appliquées"

présentée et soutenue publiquement par

Haithem BEN KAHLA

le 14 décembre 2017

Sur des méthodes préservant les structures d'une classe de matrices structurées

Dirigée par : **Skander BELHAJ, Maher MOAKHER et Ahmed SALAM**

Membres du Jury

M. Nabil GMATI,	PR, Université de Tunis El Manar, Tunisie	Président
Mme Michela REDIVO-ZAGLIA,	PR, Université de Padoue, Italie	Rapporteur
M. Maher BERZIG,	MC HDR, Université de Tunis, Tunisie	Rapporteur
M. Claude BREZINSKI,	PR émérite, Université de Lille 1, France	Invité
M. Skander BELHAJ,	MC HDR, Université de Manouba, Tunisie	Examinateur
M. Ahmed SALAM,	MC HDR, Université Littoral Côte d'Opale, France	Directeur
M. Maher MOAKHER,	PR, Université de Tunis El Manar, Tunisie	Co-directeur

Laboratoire de Mathématiques Pures et Appliquées Joseph Liouville (LMPA),
Centre Universitaire de la Mi-Voix 50, rue Ferdinand Buisson, CS 80699 - 62228 Calais Cedex, France
Laboratoire de Modélisation Mathématique et Numérique dans les Sciences de l'Ingénieur (LAMSIN),
Université de Tunis El Manar, École National d'Ingénieurs de Tunis, Campus universitaire BP 37, 1002 Tunis, Tunisie

THESE

Cette thèse a été préparée au sein de deux laboratoires :

LAMSIN :
Laboratoire de Modélisation Mathématique et Numériques dans les Sciences de l'Ingénieur
École Nationale d'Ingénieurs de Tunis
Université Tunis El-Manar
Campus universitaire BP 37
1002 Le Belvédère Tunis
Tunisie
☎ (+216) 71 87 10 22
(+216) 71 87 10 22
lamsin@enit.rnu.tn
http://www.lamsin.rnu.tn

LMPA Joseph Liouville :
Laboratoire de Mathématiques Pures et Appliquées Joseph Liouville
Centre Universitaire de la Mi-Voix
Maison de la Recherche Blaise Pascal
50, rue Ferdinand Buisson
CS 80699
62228 Calais Cedex
France
☎ (+33) (0)3 21 46 55 86
(+33) (0)3 21 46 55 75
secretariat@lmpa.univ-littoral.fr
http://www-lmpa.univ-littoral.fr

Remerciements

La réalisation de cette thèse fut une opportunité merveilleuse de rencontrer et d'échanger avec de nombreuses personnes. Je ne saurais pas les citer toutes sans dépasser le nombre de pages raisonnablement admis dans ce genre de travail. Je reconnais que chacune a, à des degrés divers, mais avec une égale bienveillance, apporté une contribution positive à sa finalisation. Mes dettes de reconnaissance sont, à ce point de vue, énormes à leur égard.

Je vais commencer par remercier avec l'intensité qui se doit les membres de mon jury, à savoir Nabil GMATI, Michela REDIVO-ZAGLIA, Maher BERZIG, Claude BREZINSKI, Skander BELHAJ, Ahmed SALAM et Maher MOAKHER, d'avoir fait d'un doctorant un docteur.

Je tiens en premier lieu à remercier très sincèrement mes directeurs de thèse Monsieur Ahmed SALAM et Monsieur Maher MOAKHER pour leur encadrement, leur disponibilité et leur aide considérable tout au long de mes années de thèse. Je leur suis très reconnaissant pour les nombreuses discussions et suggestions qui m'ont permis d'améliorer mes connaissances, ses remarques et ses critiques constructives ont énormément contribué à améliorer la qualité de la thèse. Je voudrais les remercier pour m'avoir toujours encouragé et pour avoir toujours trouvé les bons mots pour me faire retrouver la confiance en moi-même.

Je pense particulièrement à mon co-encadrant Monsieur Skander BELHAJ pour la finesse de ses attitudes sur le plan aussi bien humain que scientifique. Je lui remercie vivement et chaleureusement de m'avoir fait partager ses connaissances et son expérience. Tous mes remerciements pour sa confiance et son aide précieuse. J'ai été extrêmement sensible à ses qualités humaines d'écoute et de compréhension tout au long de ce travail. Je lui exprimer ma sincère reconnaissance et mon profond respect. Je lui voue une très grande admiration. Il a toujours trouvé le juste équilibre entre la liberté qu'il m'a laissée dans le choix des grandes orientations et dans la détermination des pistes à suivre, d'une part, et un soutien total et sans faille dans les moments délicats, d'autre part. De lui, j'ai toujours reçu non seulement les encouragements dont le doctorant a tant besoin, mais aussi les précieux conseils pratiques que seul un homme, ayant des qualités humaines comme lui, peut amener à prodiguer. Grâce à son approche respectueuse de la personne humaine, je me suis continuellement senti à l'aise. Je lui en sais infiniment gré.

Je tiens à remercier Madame Michela REDIVO-ZAGLIA et Monsieur Maher BERZIG pour avoir accepter d'être rapporteurs de mes travaux de thèse et pour leurs observations

qui m'ont permis d'améliorer la qualité de ce mémoire. Je tiens à leur exprimer mes remerciements pour l'honneur qu'ils me font en participant à ce jury.

Mes sincère remerciements et ma gratitude vont aussi à Monsieur Nabil GMATI pour avoir accepter de juger ce travail et d'en présider le jury.

C'est également avec plaisir que je remercie Monsieur Claude BREZINSKI pour m'avoir fait l'honneur de faire partie de jury.

Un grand merci à tous les membres du LAMSIN et LMPA qui ont été toujours à mes cotés par leurs conseils et leur aide, avec une pensée particulière à Raoudha JELASSI, Salem DRIDI, Isabelle BUCHARD et Carole LAGATIE sans qui je n'aurais jamais réussi à surmonter les difficultés administratives. Je ne pourrais également manquer de remercier en particulier les membres de l'équipe de la Bibliothèque Régionale de Recherche Mathématique à Lille : Hélène DEHAUDT, Omar AOUADI et Catherine GAQUIÈRE.

Je remercie tous mes amis et proches, qui ont contribué de près ou de loin à l'accomplissement de cette thèse. Je remercie mes amis Marwa, Oussama, Fateh, Ahmed, Anis, Wissal, Saber, Asma, Mohamed, Abdullatif, Aya, Meriem, Salwa et etc. pour leurs encouragements et les discussion que nous avons pu partager.

Enfin, je me tourne vers ma famille. Mes pensées affectueuses se dirigent vers mes parents Mounira et Ayache, ma sœur Wafa, mon beau-frère Fakhri, mon adorable neveu Ayoube et mes aimables nièces Alaa et Aya qui m'ont soutenu par leur présence tout au long de cette aventure doctorale. Ils ont toujours été à mes côtés et ont su m'écouter et m'encourager. Ils ont tout mis en œuvre pour que je puisse mener à bien ce travail. J'espère qu'ils retrouveront dans ce travail les valeurs de liberté, d'autonomie et de fidélité qu'ils m'ont fait partager. J'adresse également mes remerciements les plus affectueux à mes tentes Radia, Saida et Fatma pour leur soutien inconditionnel, leurs encouragements sans faille et surtout pour ce qu'ils représentent pour moi. Je ne saurais oublier de remercier Oncle Habibe, Tante Souad, Siwar, Sirine et Ahmed Yassine. Merci pour vous tous pour votre encouragement, votre amour et le support moral que vous m'avez accordé. Que Dieu vous protège et vous donne chance, santé et longue vie. Finalement, je ne me lasserais jamais de remercier ma chère fiancée Nawress qui a toujours été présente à mes côtés. Merci Nawress... Que Dieu te bénisse...

Pour finir, je ne manquerai pas de saluer le lecteur pour la motivation qui l'aura amené à lire le présent document et pour l'intérêt qu'il prêtera, je l'espère, à son contenu.

Haithem BEN KAHLA

À mes parents,
À tout les membres de ma famille,
À tous mes proches et à tous ceux qui m'aiment et j'aime.

Résumé

Les méthodes d'algèbre linéaire classiques, pour le calcul de valeurs et vecteurs propres d'une matrice, ou des approximations de rang inférieurs (low-rank approximations) d'une solution, etc., ne tiennent pas compte des structures de matrices. Ces dernières sont généralement détruites durant le procédé du calcul. Des méthodes alternatives préservant ces structures font l'objet d'un intérêt important par la communauté. Cette thèse constitue une contribution dans ce domaine.

Ainsi l'algorithme SR, qui est un algorithme de type QR, et qui préserve les structures d'une classe importante de matrices, est basé sur la décomposition SR (qui est de type QR) et sur une réduction à une forme condensée : J-Hessenberg forme (de type Hessenberg).

La décomposition SR peut être calculé via l'algorithme de Gram-Schmidt symplectique. Comme dans le cas classique, une perte d'orthogonalité peut se produire. Pour y remédier, nous avons proposé deux algorithmes $RSGSi$ et $RMSGSi$, qui consistent à ré-orthogonaliser deux fois les vecteurs à calculer. La perte de la J-orthogonalité s'est améliorée de manière très significative.

L'étude directe de la propagation des erreurs d'arrondis dans les algorithmes de Gram-Shmidt symplectique est très difficile à effectuer. Nous avons réussi à contourner cette difficulté et donner des majorations pour la perte de la J-orthogonalité et de l'erreur de factorisation.

Une autre façon de calculer la décomposition SR est basée sur les transformations de Householder symplectiques. Celles ci présentent des paramètres libres. Un choix optimal a abouti à l'algorithme $SROSH$. Cependant, ce dernier peut être sujet à une instabilité numérique. Nous avons proposé une version modifiée nouvelle $SRMSH$, qui a l'avantage d'être aussi stable que possible. Une étude approfondie a été faite, présentant les différentes versions : $SRMSH$ et $SRMSH2$.

Dans le but de construire un algorithme SR, d'une complexité d'ordre $\mathcal{O}(n^3)$ où $2n$ est la taille de la matrice, une réduction (appropriée) de la matrice à une forme condensée (J-Hessenberg forme) via des similarités adéquates, est cruciale. Cette réduction peut être effectué via l'algorithme $JHESS$.

Nous avons montré qu'il possible de réduire une matrice sous la forme J-Hessenberg, en se basant exclusivement sur les transformations de Householder symplectiques. Le nouvel algorithme, appelé $JHSH$, est basé sur une adaptation de l'algorithme $SRSH$. D'un point de vue algébrique, cette méthode est l'analogue de la réduction d'une matrice sous la forme Hessenberg, par des transformations de Householder, dans le cas Euclidien. Ce nouveau algorithme peut être aussi sujet à une instabilité numérique. Nous avons réussi à proposer deux nouvelles variantes, aussi stables que possible : $JHMSH$ et $JHMSH2$. Une étude approfondie a été faite. Nous avons constaté que ces algorithmes se comportent d'une manière similaire à l'algorithme $JHESS$.

Une caractéristique importante de tous ces algorithmes ($JHESS$, $JHMSH$, $JHMSH2$) est qu' ils peuvent rencontrer un breakdown fatal, ou un "near breakdown" rendant impossible la suite des calculs, ou débouchant sur une instabilité numérique, privant le résultat final de toute signification. Ce phénomène n'a pas d'équivalent dans le cas Euclidien.

Nous avons réussi à élaborer une stratégie très efficace pour "guérir" le breakdown fatal et traîter le near breakdwon. Les nouveaux algorithmes intégrant cette stratégie sont désignés par $MJHESS$, $MJHSH$, JHM^2SH et JHM^2SH2.

Ces stratégies ont été ensuite été intégrées dans la version implicite de l'algorithme SR lui permettant de surmonter les difficultés rencontrées du fatal breakdown ou du near-breakdown. Rappelons que, sans ces stratégies, l'algorithme SR s'arrête.

Ensuite, nous présenté une variante de l'algorithme QR appliquée aux matrices Hamiltonienne (symétriques ou antisymétriques) qui préserve leur structure pendant la procédure.

Finalement, et dans un autre cadre de matrices structurées, nous avons présenté un algorithme robuste via FFT et la matrice de Hankel, basé sur le calcul approché de plus grand diviseur commun (PGCD) de deux polynômes, pour résoudre le problème de la déconvolution d'images. Plus précisément, nous avons conçu un algorithme pour le calcul du PGCD de deux polynômes bivariés. La nouvelle approche est basée sur un algorithme rapide, de complexité quadratique $\mathcal{O}(n^2)$, pour le calcul du PGCD des polynômes unidimensionnels. La complexité de notre algorithme est $\mathcal{O}(n^2\log(n))$ où la taille des images floues est $n \times n$. Les résultats expérimentaux avec des images synthétiquement floues, illustrent l'efficacité de notre approche.

Mots-clés : Produit scalaire antisymétrique, la préservation de la structure, matrice structurée, les transformations de Householder symplectiques, Gram-Schmidt symplectique, la décomposition SR, la forme de J-Hessenberg, réduction de matrice, l'algorithme SR, le PGCD approché, la matrice de Hankel, la matrice Hamiltonienne, la matrice symplectique, déconvolution d'image floue, restauration d'images.

Abstract

The classical linear algebra methods, for calculating eigenvalues and eigenvectors of a matrix, or lower-rank approximations of a solution, etc., do not consider the structures of matrices. Such structures are usually destroyed in the numerical process. Alternative structure-preserving methods are the subject of an important interest mattering to the community. This thesis establishes a contribution in this field.

Thus the SR algorithm, which is a QR-like algorithm, structure-preserving for a large class of structured matrices, is based on the SR decomposition (which is a QR-like decomposition) and on a reduction to the condensed matrix form : upper J-Hessenberg form (Hessenberg-like).

The SR decomposition is usually implemented via the symplectic Gram-Schmidt algorithm. As in the classical case, a loss of orthogonality can occur. To remedy this, we have proposed two algorithms $RSGSi$ and $RMSGSi$, where the reorthogonalization of a current set of vectors against the previously computed set is performed twice. The loss of J-orthogonality has significantly improved.

A direct rounding error analysis of symplectic Gram-Schmidt algorithm is very hard to accomplish. We managed to get around this difficulty and give the error bounds on the loss of the J-orthogonality and on the factorization.

Another way to implement the SR decomposition is based on symplectic Householder transformations. This algorithm involves some free parameters. An optimal choice of free parameters provided an optimal version of the algorithm $SROSH$. However, the latter may be subject to numerical instability. We have proposed a new modified version $SRMSH$, which has the advantage of being numerically more stable. By a detailed study, we are led to two new variants numerically more stables : $SRMSH$ and $SRMSH2$.

In order to build a SR algorithm of complexity $\mathcal{O}(n^3)$, where $2n$ is the size of the matrix, a reduction to the condensed matrix form (upper J-Hessenberg form) via adequate similarities is crucial. This reduction may be handled via the algorithm $JHESS$.

We have shown that it is possible to perform a reduction of a general matrix, to an upper J-Hessenberg form, based only on the use of symplectic Householder transformations. The new algorithm, which will be called $JHSH$ algorithm, is based on an adaptation of $SRSH$ algorithm. From a linear algebra point of view, $JHSH$ is the analogue in the symplectic case, of the algorithm performing the Hessenberg reduction of a matrix via Householder transformations in the Euclidean case. This new algorithm may also be subject to numerical instability. We are led to two news variants algorithms $JHMSH$ and $JHMSH2$ which are significantly more stable numerically. A detailed study has been done. We found that these algorithms behave quite similarly to $JHESS$ algorithm.

The main drawback of all these algorithms ($JHESS$, $JHMSH$, $JHMSH2$) is that they may encounter fatal breakdowns or may suffer from a severe form of near-breakdowns, causing a brutal stop of the computations, the algorithm breaks down, or leading to a serious numerical instability. This phenomenon has no equivalent in the Euclidean case.

We sketch out a very efficient strategy for curing fatal breakdowns and treating near breakdowns. Thus, the new algorithms incorporating this modification will be referred to as $MJHESS$, $MJHSH$, JHM^2SH and JHM^2SH2.

These strategies were then incorporated into the implicit version of the SR algorithm to overcome the difficulties encountered by the fatal breakdown or near-breakdown. We recall that without these strategies, the SR algorithm breaks.

Next, we presented a new variant of a double structure-preserving QR Algorithm for symmetric\skew-symmetric and Hamiltonian Matrices.

Finally and in another framework of structured matrices, we presented a robust algorithm via FFT and a Hankel matrix, based on computing approximate greatest common divisors (GCD) of polynomials, for solving the problem of blind image deconvolution. Specifically, we designed a specialized algorithm for computing the GCD of bivariate polynomials. The new algorithm is based on the fast GCD algorithm for univariate polynomials, of quadratic complexity $\mathcal{O}(n^2)$ flops. The complexity of our algorithm is $\mathcal{O}(n^2\log(n))$ where the size of blurred images is $n \times n$. The experimental results with synthetically blurred images are included to illustrate the effectiveness of our approach.

Key-words : Indefinite inner product, structure-preserving eigenproblems, structured matrix, symplectic Householder transformations, symplectic Gram-Schmidt, SR decomposition, upper J-Hessenberg form, breakdowns and near-breakdowns, matrix reduction, SR-algorithm, approximate GCD, Hankel matrix, Hamiltonian matrix, symplectic matrix, blind image deconvolution, image restauration.

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Table de notations

Nous résumons ici les notations les plus courantes qui seront utilisées tout au long de ce manuscrit.

Notation	**Explication**
$\mathbb{K}$	Corps arbitraire
$\mathbb{R}$	Corps des nombres réels
$\mathbb{C}$	Corps des nombres complexes
$\mathbb{K}^{n\times p}$	Corps des matrices arbitraires de taille $n \times p$
$\mathbb{R}^{n\times p}$	Corps des matrices réelles de taille $n \times p$
$\mathbb{C}^{n\times p}$	Corps des matrices complexes de taille $n \times p$
$(.,.)$	Le produit scalaire usuel
$(.,.)_J$	Le produit scalaire antisymétrique
A^{T}	La transposée de la matrice A
$A^*,\ A^H$	la transposée conjuguée de la matrice A, $(A^* = \bar{A}^{\mathrm{T}})$
A^J	La J-transposée ou l'adjoint symplectique de la matrice A
e_k	Le $k^{\text{ème}}$ vecteurs de la base canonique
$I = I_n$	La matrice identité de taille $n \times n$
$0 = 0_n$	La matrice nulle de taille $n \times n$
$\bar{z}$	Le conjugué du nombre complexe z
$\mathrm{rang}(A)$	Le rang de la matrice A
$\mathrm{trace}(A)$	La trace de la matrice A
$\det(A)$	Le déterminant de la matrice A
$\mathrm{vect}(a_1, a_2, \ldots, a_n)$	L'espace engendré par la famille $(a_1, a_2, \ldots, a_n)$
$\kappa(A)$, $\mathrm{cond}(A)$	Le nombre de conditionnement de la matrice A
$\mathcal{O}(.)$	La notion de complexité
flop	Opération en arithmétique flottante

Introduction

« *Change is the end result of all true learning* »

Leo Buscaglia (1924-1998)

Des problèmes provenant du contrôle linéaire optimal [6, 15, 33, 41, 82] comme les problèmes LQR, les équations algébriques continues de Riccati, etc., la modélisation de certaines applications industrielles (mécanique quantique, résonance, etc.) débouchent sur le calcul de valeurs/vecteurs propres ou espaces invariants de matrices structurées creuses et de grandes tailles. Une matrice symétrique définie positive est un exemple classique de matrices structurées, pour lesquelles il existe dans la littérature des méthodes de calcul de valeurs/vecteurs propres exploitant d'une manière optimale cette structure, en diminuant le coût et en augmentant l'efficacité.

Le besoin de calculer des valeurs et vecteurs propres des matrices structurées, moins classiques, telles que les matrices Hamiltoniennes, anti-Hamiltoniennes et symplectiques est rencontré dans beaucoup d'applications comme la construction dynamique, macro-économie, chimie quantique, la théorie de contrôle, etc. En effet, de nombreux problèmes de calcul des valeurs propres découlant de la pratique sont structurés en raison des propriétés physiques induites par le problème initial. La structure peut également être introduite par des techniques de discrétisation et de linéarisation. La préservation de cette structure peut aider à préserver les symétries physiquement pertinentes dans les valeurs propres de la matrice et peut améliorer la précision et l'efficacité d'un calcul des valeurs propres.

La méthode QR pourrait ne pas être un outil approprié pour ce genre de calcul. En raison des erreurs d'arrondis inévitables dans l'arithmétique de précision finie, les valeurs propres calculées ne sont généralement pas en paires ou en quadruple bien que les valeurs propres exactes aient cette propriété. Ce problème est dû au fait que les méthodes standards ignorent la structure de la matrice puisqu'elle est traitée comme n'importe quelle autre matrice. En effet, les méthodes de préservation de structure sont souhaitables, afin de préserver les structures [18, 20, 38, 40, 70, 83].

Une matrice réelle $S \in \mathbb{R}^{2n\times 2n}$ est dite symplectique ou J-orthogonale si

$$S^{\mathrm{T}} J S = J,$$

avec la matrice J s'écrit sous la forme :

$$J = \begin{pmatrix} 0_n & I_n \\ -I_n & 0_n \end{pmatrix} \in \mathbb{R}^{2n\times 2n}.$$

Les matrices symplectiques sont inversibles puisque

$$S^{-1} = J^{\mathrm{T}} S^{\mathrm{T}} J$$

et leurs valeurs propres se présentent en paires réciproques. C'est-à-dire en paire $\{\lambda, \lambda^{-1}\}$ si λ est une valeur propre réelle ou imaginaire pure et en quadruple $\{\lambda, \lambda^{-1}, \bar{\lambda}, \bar{\lambda}^{-1}\}$ si λ est une valeur propre complexe avec une partie réelle non nulle.

La solution du problème de calcul des valeurs propres symplectiques a fait l'objet de nombreuses publications au cours des dernières décennies, mais elle n'est pas encore bien connue. Ce problème est utile par exemple pour analyser un certain nombre de problèmes qui se posent dans la théorie du contrôle linéaire pour les systèmes à temps discret. Certains problèmes de calcul des valeurs propres quadratiques par exemple dans la discrétisation par éléments finis dans l'analyse structurelle, la simulation acoustique de matériaux poro-élastiques ou la déformation élastique de matériaux anisotropes peuvent également conduire à des problèmes de calcul des valeurs propres symplectiques. Le problème apparaît également dans d'autres applications [16, 37, 54, 61, 65, 72, 79, 80, 98].

Une matrice réelle $H \in \mathbb{R}^{2n\times 2n}$ est dite Hamiltonienne si

$$HJ = (HJ)^{\mathrm{T}}.$$

En effet, toute matrice Hamiltonienne s'écrit sous la forme :

$$H = \begin{pmatrix} A & G \\ Q & -A^{\mathrm{T}} \end{pmatrix},$$

avec les blocs $A,\ G,\ Q \in \mathbb{R}^{n\times n}$ tels que $G = G^{\mathrm{T}}$ et $Q = Q^{\mathrm{T}}$.

Pareillement, les valeurs propres d'une matrice Hamiltonienne se présentent toujours en paires $\{\lambda, -\lambda\}$ pour les valeurs propres réelles et purement imaginaires et en quadruples $\{\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}\}$ pour les valeurs propres complexes avec une partie réelle non nulle.

Le calcul du sous-espace invariant de la matrice H Hamiltonienne est parmi les applications importantes dans les problèmes linéaires de contrôle quadratique.

L'ensemble de toutes les matrices symplectiques $\mathbb{S}$ forme un groupe multiplicatif : le groupe de Lie et l'ensemble de toutes les matrices Hamiltoniennes $\mathbb{L}$ forme une algèbre de Lie [3, 5, 108]. De plus, les transformations symplectiques de similarité préservent la structure Hamiltonienne.

$$(S^{-1}HS)J = [(S^{-1}HS)J]^{\mathrm{T}}.$$

En outre, la transformée de Cayley transforme une matrice symplectique en une matrice Hamiltonienne et vice versa. Ceci explique la ressemblance entre les spectres des matrices Hamiltoniennes et les spectres des matrices symplectiques. Malgré cette relation, le problème du calcul des valeurs propres d'une matrice symplectiques est beaucoup plus difficile que le problème du calcul des valeurs propres d'une matrice Hamiltonienne

[4, 17, 18, 21, 22, 42, 53, 75, 83, 90, 106, 114]. En particulier, la structure des matrices Hamiltoniennes est explicite alors que la structure des matrices symplectiques est donnée implicitement.

Le problème de la résolution de l'équation algébrique de Riccati [71]

$$-XNX + XA + A^{\mathrm{T}}X + K = O, \tag{1}$$

où les matrices X, N, K, A sont des matrices réelles de taille $n \times n$ avec $K = K^{\mathrm{T}}$ et $N = N^{\mathrm{T}}$, se pose fréquemment dans les problèmes de contrôle optimal [2, 39, 40, 71, 73, 74, 84].

Le problème de contrôle optimal linéaire consiste à trouver une trajectoire de contrôle $\{u(t),\, t \geq 0\}$ tel que la fonction de contrôle

$$u(t) = Hx(t),$$

minimise la fonction

$$J(u) = \int_0^\infty [x^{\mathrm{T}}(t)Kx(t) + u^{\mathrm{T}}(t)Ru(t)]\,\mathrm{d}t,$$

sous la contrainte

$$\begin{cases} \dot{x}(t) = \dfrac{\mathrm{d}}{\mathrm{d}t}x(t) = Ax(t) + Bu(t) \\ x(0) = x_0 \end{cases} \tag{2}$$

Supposons que la matrice $K = K^{\mathrm{T}}$ est semi-définie positive et la matrice $R = R^{\mathrm{T}}$ est définie positive. Soit $K = C^{\mathrm{T}}C$ une décomposition de la matrice K. Supposons que la paire (A, B) est stabilisable, c'est-à-dire si $w^{\mathrm{T}}A = \lambda w^{\mathrm{T}}$ et $\Re\mathrm{e}(\lambda) \geq 0$ avec $w \neq 0$, alors $w^{\mathrm{T}}A \neq 0$. Supposons aussi que la paire (C, A) est détectable, c'est-à-dire si $Ax = \lambda x$ et $\Re\mathrm{e}(\lambda) \geq 0$ avec $x \neq 0$, alors $Cx \neq 0$.

Sous ces hypothèses, l'équation (1) admet une unique solution symétrique définie positive X et la fonction de contrôle optimal s'écrit

$$u(t) = -R^{-1}B^{\mathrm{T}}Xx.$$

Soit M une matrice de taille $2n \times 2n$ par blocs tel que

$$M = \begin{pmatrix} A & N \\ K & -A^{\mathrm{T}} \end{pmatrix},$$

où $N = BR^{-1}B^{\mathrm{T}}$. Cette matrice a la forme d'une matrice Hamiltonienne.

Si les matrices Y, Z et Λ sont des matrices de tailles $n \times n$ telles que nous avons

$$M \begin{bmatrix} Y \\ Z \end{bmatrix} = \begin{bmatrix} Y \\ Z \end{bmatrix} \Lambda,$$

et la matrice Y est inversible, alors

$$X = -ZY^{-1} \tag{3}$$

résout l'équation (1).

Pour chaque valeur propre λ de la matrice M la valeur $-\bar{\lambda}$ est aussi une valeur propre de la matrice M avec la même multiplicité algébrique et géométrique que λ. Dans les hypothèses ci-dessus la matrice M n'a pas de valeur propre avec une partie réelle nulle.

Comme la matrice $\begin{bmatrix} Y \\ Z \end{bmatrix}$ engendre les sous-espaces réels invariants associés aux n valeurs propres avec une partie réelle négative, alors la matrice symétrique semi-définie positive

$$X = -ZY^{-1}$$

existe.

Le sous-espace invariant $\begin{bmatrix} Y \\ Z \end{bmatrix}$ de la matrice M peut être calculé par l'algorithme QR [73]. Mais cette méthode ne peut pas profiter de la structure Hamiltonienne de la matrice M. Elle traitera la matrice M comme n'importe quelle matrice arbitraire de taille $2n \times 2n$. Ainsi, pour préserver la structure Hamiltonienne nous devons utiliser des transformations de similarité symplectiques au lieu des transformations habituelles dans l'algorithme QR.

La présente thèse s'inscrit dans ce cadre et dans le but d'introduire une nouvelle version de l'algorithme SR avec une complexité d'ordre $\mathcal{O}(n^3)$ dans le cas général et aussi pour les matrices Hamiltoniennes. En fait, nous avons implémenté une nouvelle version de l'algorithme SR implicite avec shift en se basant sur les transformations de Householder symplectiques. Nous allons montré qu'il est possible de réduire une matrice sous la forme J-Hessenberg, en se basent exclusivement sur les transformations de Householder symplectiques, qui sont des transvections s'écrivant comme l'identité modifiée par le rajout d'un terme de rang un. Notre algorithme SR est basé aussi sur la décomposition SR à chaque itération qui est une méthode qui préserve la structure et analogue à la décomposition QR dans le cas Euclidien. Du point de vue algébrique, notre méthode dans le cas symplectique est l'analogue à l'algorithme QR via les transformations de Householder dans le cas Euclidien.

Dans un autre cadre des matrices structurées, nous avons proposé un algorithme rapide pour résoudre le problème de la déconvolution d'images. Nous avons introduit un algorithme robuste via FFT (la transformation de Fourier rapide) et la matrice de Hankel, qui est basé sur le calcul approché de plus grand diviseur commun (PGCD) de deux polynômes, pour résoudre le problème de la déconvolution d'images. Plus précisément, nous concevons un algorithme spécialisé pour le calcul du PGCD de deux polynômes à deux variables. La nouvelle approche est basée sur l'algorithme rapide, de complexité quadratique $\mathcal{O}(n^2)$, pour le calcul du PGCD des polynômes unidimensionnels. La complexité de notre algorithme est $\mathcal{O}(n^2 \log(n))$ où la taille des images floues est $n \times n$. Les résultats expérimentaux avec des images synthétiquement floues sont inclus pour illustrer l'efficacité de notre approche. Ce dernier chapitre a fait l'objet d'un article en collaboration avec Skander Belhaj, Marwa Dridi et Maher Moakher intitulé "Blind image deconvolution via Hankel based method for computing the GCD of polynomials" et accepté pour publication dans la revue "Mathematics and Computers in Simulation", (voir [13]).

Le travail présenté se décompose en quatre grandes parties. Le premier chapitre présente le cadre terminologique. La première partie commencera à partir du deuxième chapitre qui s'intéressera aux algorithmes de la décomposition SR symplectique. La deuxième

partie consacrée aux méthodes de réduction sous la forme J-Hessenberg sera présentée dans le troisième chapitre. L'algorithme SR sera présenté dans le quatrième chapitre. Pour finir, le dernier chapitre traitera le problème de la déconvolution d'images en introduisant un algorithme robuste via FFT et la matrice de Hankel basée sur l'inversion rapide de la matrice triangulaire de Toeplitz.

Chapitre 1

Préliminaires

« *When one door closes, another opens; but we often look so long and so regretfully upon the closed door that we do not see the one which has opened for us* »

Alexander Graham Bell
(1847-1922)

Sommaire

Dans ce chapitre, nous introduisons quelques outils qui seront nécessaires tout au long de cette thèse. Nous donnons également certaines définitions et propriétés sur les matrices structurées tels que les matrices Hamiltonienne, anti-Hamiltonienne, symplectique, Hankel, Toeplitz, etc.

1.1 Notations

Nous utilisons les lettres majuscules et minuscules pour désigner les matrices et les vecteurs respectivement. Les lettres minuscules grecques désignent les scalaires. Nous désignons les matrices réelles de taille $n \times k$ par $\mathbb{R}^{n\times k}$, les matrices complexes de taille $n \times k$ par $\mathbb{C}^{n\times k}$. Nous utilisons $\mathbb{K}$ pour désigner $\mathbb{R}$ ou $\mathbb{C}$. La matrice d'identité $n \times n$ sera désignée par $I_{n,n}$ ou bien I_n. Si la dimension de I_n est claire du contexte nous la notons tout simplement I avec $I = [e_1, e_2, \ldots, e_n]$ et e_k est le $k^{\text{ième}}$ vecteur de la base canonique.

Soit la matrice $\mathbb{K}^{n\times k}$, nous désignerons par :

— $a_{i,j}$ l'entrée (i, j) de la matrice A.

— $(a_{i,j})_{i=1,\ldots,n,\ j=1,\ldots,k}$ la matrice A.

— $A_{j,1:k}$ la $j^{\text{ième}}$ ligne de la matrice A.

— $A_{j,l:m}$ les entrées $l, l+1, \ldots, m$ de la $j^{\text{ième}}$ ligne de la matrice A.

— $A_{1:n,j}$ la $j^{\text{ième}}$ colonne de la matrice A.

— $A_{l:m,j}$ les entrées $l, l+1, \ldots, m$ de la $j^{\text{ième}}$ colonne de la matrice A.

— A^{T} la transposée de la matrice $A = (a_{ij})$ avec $A^{\mathrm{T}} = (a_{ji})$.

— A^* ou A^H la transposé conjuguée de la matrice $A = (a_{ij})$ (conjuguée hermitienne) avec $A^H = (\bar{a}_{ji})$.

— A_{ij} le bloc (i, j) ou bien la sous-matrice en position (i, j) de la matrice A.

— vect(A) l'espace engendré par les colonnes de la matrice A.

Le produit scalaire usuel est noté par

$$(x, y) = x^{\mathrm{T}} y,$$

avec $x,\ y \in \mathbb{R}^n$.

Soit $J_{2n} \in \mathbb{R}^{(2n\times 2n)}$ (ou simplement J lorsque la dimension est claire dans le contexte) une matrice réelle définie par

$$J_{2n} = \begin{pmatrix} 0_n & I_n \\ -I_n & 0_n \end{pmatrix}, \tag{1.1}$$

où les matrices 0_n et $I_n \in \mathbb{R}^{(n\times n)}$ sont respectivement la matrice nulle et la matrice identité. La matrice J est symplectique, antisymétrique et orthogonale, c'est-à-dire $J^{\mathrm{T}} = J^{-1} = -J$ avec $\det(J) = 1$ et $J^2 = -J$.

Nous considérons la permutation P ("perfect shuffle") tel que

$$P = [e_1, e_3, \ldots, e_{2n-1}, e_2, \ldots, e_{2n}] \in \mathbb{R}^{2n\times 2n}. \tag{1.2}$$

Ainsi, l'inverse de la matrice P est la matrice

$$P^{\mathrm{T}} = [e_1, e_{n+1}, e_2, e_{n+2}, \ldots, e_n, e_{2n}] \in \mathbb{R}^{2n\times 2n}.$$

De plus, nous avons

$$J = P^{\mathrm{T}} \operatorname{diag}\left(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\right) P = P^{\mathrm{T}} \operatorname{diag}(J_2) P.$$

1.1.1 Le produit scalaire associé à la matrice J

Soit la forme bilinéaire

$$\begin{aligned} \mathbb{R}^{2n} \times \mathbb{R}^{2n} &\longrightarrow \mathbb{R} \\ (x, y) &\longmapsto (x, y)_J \end{aligned}$$

tel que

$$\forall x \in \mathbb{R}^{2n},\ \forall y \in \mathbb{R}^{2n},\ (x, y)_J = (x, Jy) = x^{\mathrm{T}} J y. \tag{1.3}$$

Cette forme bilinéaire (1.3) est non-dégénérée, c'est-à-dire

$$(x, y)_J = 0,\ \forall y \in \mathbb{R}^{2n} \Rightarrow x = 0,$$

et antisymétrique, c'est-à-dire

$$\forall x, y \in \mathbb{R}^{2n},\ (x, y)_J = -(y, x)_J.$$

Définition 1.1.1. *(L'adjoint ou la J-transposée)*

— *La J-transposée x^J d'un vecteur $x \in \mathbb{R}^{2n}$ est définie par :*

$$x^J = x^{\mathrm{T}} J. \tag{1.4}$$

— *La J-transposée d'une matrice $A \in \mathbb{R}^{2n\times 2k}$ réelle est définie par :*

$$A^J = J_{2k}^{\mathrm{T}} A^{\mathrm{T}} J_{2n}, \tag{1.5}$$

tel que

$$\forall x \in \mathbb{R}^{2n},\ \forall y \in \mathbb{R}^{2n},\ (Mx, y)_J = (x, M^J y)_J.$$

Proposition 1.1.1. *Soit A une matrice réelle de taille $2n \times 2m$ tel que*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}.$$

Alors, A^J la J-transposée de la matrice A s'écrit

$$A^J = \begin{pmatrix} A_{22}^{\mathrm{T}} & -A_{12}^{\mathrm{T}} \\ -A_{21}^{\mathrm{T}} & A_{11}^{\mathrm{T}} \end{pmatrix}.$$

Preuve. En effet, nous avons

$$\begin{aligned}
A^J &= J_{2m}^{\mathrm{T}} A^{\mathrm{T}} J_{2n} \\
&= \begin{pmatrix} 0_m & -I_m \\ I_m & 0_m \end{pmatrix} \begin{pmatrix} A_{11}^{\mathrm{T}} & A_{12}^{\mathrm{T}} \\ A_{21}^{\mathrm{T}} & A_{22}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} 0_n & I_n \\ -I_n & 0_n \end{pmatrix} \\
&= \begin{pmatrix} -A_{21}^{\mathrm{T}} & -A_{22}^{\mathrm{T}} \\ A_{11}^{\mathrm{T}} & A_{12}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} 0_n & I_n \\ -I_n & 0_n \end{pmatrix} \\
&= \begin{pmatrix} A_{22}^{\mathrm{T}} & -A_{21}^{\mathrm{T}} \\ -A_{12}^{\mathrm{T}} & A_{11}^{\mathrm{T}} \end{pmatrix}.
\end{aligned}$$

□

Les propriétés de la J-transposée (l'adjoint) par rapport au produit scalaire antisymétrique $(.,.)_J$ sont toutes les analogues de la transposée. En particulier, nous avons les résultats suivants.

Proposition 1.1.2. *Soient les matrices $A \in \mathbb{R}^{2n\times 2k}$, $B \in \mathbb{R}^{2k\times 2p}$ et $C \in \mathbb{R}^{2n\times 2k}$ tels que nous avons les propriétés suivantes :*

$$\begin{aligned}
(AB)^J &= B^J A^J, \\
(A+C)^J &= A^J + C^J, \\
(A^J)^J &= A, \\
(A^{\mathrm{T}})^J &= (A^J)^{\mathrm{T}}, \\
(A^{-1})^J &= (A^J)^{-1}.
\end{aligned}$$

Preuve. D'après la définition (1.5), nous avons

$$(AB)^J = J_{2p}^{\mathrm{T}}(AB)^{\mathrm{T}} J_{2n} = J_{2p}^{\mathrm{T}} B^{\mathrm{T}} A^{\mathrm{T}} J_{2n} = J_{2p}^{\mathrm{T}} B^{\mathrm{T}} J_{2k} J_{2k}^{\mathrm{T}} A^{\mathrm{T}} J_{2n} = B^J A^J.$$

$$(A+C)^J = J_{2k}^{\mathrm{T}}(A+C)^{\mathrm{T}} J_{2n} = J_{2k}^{\mathrm{T}} A^{\mathrm{T}} J_{2n} + J_{2k}^{\mathrm{T}} C^{\mathrm{T}} J_{2n} = A^J + C^J.$$

$$(A^J)^J = J_{2n}^{\mathrm{T}}(A^J)^{\mathrm{T}} J_{2k} = J_{2n}^{\mathrm{T}}(J_{2k}^{\mathrm{T}} A^{\mathrm{T}} J_{2n})^{\mathrm{T}} J_{2k} = A.$$

$$(A^{\mathrm{T}})^J = J_{2n}^{\mathrm{T}}(A^{\mathrm{T}})^{\mathrm{T}} J_{2k} = (J_{2k}^{\mathrm{T}} A^{\mathrm{T}} J_{2n})^{\mathrm{T}} = (A^J)^{\mathrm{T}}.$$

$$(A^{-1})^J = J_{2n}^{\mathrm{T}}(A^{-1})^{\mathrm{T}} J_{2k} = J_{2n}^{-1}(A^{\mathrm{T}})^{-1}(J_{2k}^{\mathrm{T}})^{-1} = (J_{2k}^{\mathrm{T}} A^{\mathrm{T}} J_{2n})^{-1} = (A^J)^{-1}.$$

□

1.2 Les ensembles structurés

Un espace linéaire muni du produit scalaire antisymétrique $(.,.)_J$ est appelé espace symplectique. Les principaux ensembles de matrices structurées associées au produit scalaire antisymétrique $(.,.)_J$ sont :

— le groupe d'automorphisme est défini par :

$$\mathbb{S} = \{S \in \mathbb{R}^{2n\times 2n} \mid (Sx, Sy)_J = (x, y)_J,\ \forall x, y \in \mathbb{R}^{2n}\},$$

où la matrice $S \in \mathbb{S}$ est dite une matrice symplectique. L'ensemble $\mathbb{S}$ est un groupe multiplicatif appelé groupe symplectique.

— L'algèbre de Jordan $\mathbb{J}$ est défini par :

$$\begin{aligned}\mathbb{J} &= \{A \in \mathbb{R}^{2n\times 2n} \mid (Ax, y)_J = (x, Ay)_J,\ \forall x, y \in \mathbb{R}^{2n}\}\\ &= \{A \in \mathbb{R}^{2n\times 2n} \mid A^J = A\},\end{aligned}$$

où la matrice $A \in \mathbb{J}$ est dite une matrice anti-Hamiltonienne. L'ensemble $\mathbb{J}$ est l'espace vectoriel des matrices anti-Hamiltonienne.

— L'algèbre de lie $\mathbb{L}$ est défini par :

$$\begin{aligned}\mathbb{L} &= \{H \in \mathbb{R}^{2n\times 2n} \mid (Hx, y)_J = -(x, Hy)_J,\ \forall x, y \in \mathbb{R}^{2n}\}\\ &= \{H \in \mathbb{R}^{2n\times 2n} \mid H^J = -H\},\end{aligned}$$

où la matrice $H \in \mathbb{L}$ est dite une matrice Hamiltonienne. L'ensemble $\mathbb{L}$ est l'espace vectoriel des matrices Hamiltoniennes.

Définition 1.2.1. *Le groupe symplectique $\mathbb{S}$ préserve la structure par similarités pour des matrices dans $\mathbb{S}$, $\mathbb{J}$ et $\mathbb{L}$, c'est-à-dire*

$$\forall S \in \mathbb{S}, \forall A \in \mathbb{E} \Rightarrow S^{-1}AS \in \mathbb{E}, \text{ où } \mathbb{E} = \mathbb{S}, \mathbb{J} \text{ ou } \mathbb{L}.$$

1.3 La géométrie symplectique

Le groupe d'automorphisme, par rapport au produit scalaire classique est donné par :

$$\mathbb{O} = \{O \in \mathbb{R}^{2n\times 2n} \mid (Ox, Oy) = (x, y),\ \forall x, y \in \mathbb{R}^{2n}\}. \tag{1.6}$$

L'ensemble $\mathbb{O}$ est le groupe des matrices orthogonales. De plus, nous avons

$$O \in \mathbb{O} \Leftrightarrow\ \forall x \in \mathbb{R}^{2n}\ (Ox, Ox) = (x, x). \tag{1.7}$$

Cette propriété n'a pas d'analogue dans le cas du groupe symplectique. En fait, pour toutes matrices $S \in \mathbb{R}^{2n\times 2n}$ symplectiques ou non nous avons :

$$(Sx, Sx)_J = (x, x)_J = 0. \tag{1.8}$$

Dans la suite, nous décrivons brièvement quelques caractéristiques de la géométrie symplectique. Pour plus de détails, voir [5]. Un vecteur $x \in \mathbb{R}^{2n}$ est dit orthogonal au vecteur $y \in \mathbb{R}^{2n}$ par rapport au produit scalaire antisymétrique $(.,.)_J$ si et seulement si

$$(x, y)_J = 0.$$

Nous utilisons le symbole $\perp$ pour désigner l'orthogonalité dans l'espace Euclidien et le symbole $\perp'$ pour désigner l'orthogonalité dans l'espace symplectique.

Soit $\mathscr{L}$ un sous-espace de $\mathbb{R}^{2n}$. L'espace orthogonal au sous-espace $\mathscr{L}$ par rapport au produit scalaire antisymétrique $(.,.)_J$ est défini par :

$$\mathscr{L}^{\perp'} = \{x \in \mathbb{R}^{2n} \mid \forall\, y \in \mathscr{L},\ \ (x,y)_J = 0\}. \tag{1.9}$$

L'espace $\mathscr{L}^{\perp'}$ est le complément symplectique de l'espace $\mathscr{L}$ et qui vérifie :

$$(\mathscr{L}^{\perp'})^{\perp'} = \mathscr{L} \quad \text{et} \quad \dim(\mathscr{L}^{\perp'}) + \dim(\mathscr{L}) = 2n. \tag{1.10}$$

Remarque 1.3.1. *Contrairement au complément orthogonal, la propriété $\mathscr{L}^{\perp} \bigcap \mathscr{L} = \{0\}$ est fausse dans la géométrie symplectique.*

Proposition 1.3.1. *Si $\mathscr{L} = \mathrm{vect}(v)$ avec v est un vecteur non nul quelconque de $\mathbb{R}^{2n}$, alors*

$$\mathscr{L}^{\perp'} \cap \mathscr{L} = \mathscr{L}.$$

Plus généralement, nous distinguons quatre cas dans la géométrie symplectique :

— *L'espace $\mathscr{L}$ est dit symplectique si*

$$\mathscr{L}^{\perp'} \cap \mathscr{L} = \{0\}.$$

C'est vrai si et seulement si la restriction de $(.,.)_J$ dans $\mathscr{L}$ est une forme non dégénérée.

— *L'espace $\mathscr{L}$ est dit isotrope si*

$$\mathscr{L}^{\perp'} \cap \mathscr{L} = \mathscr{L}.$$

C'est vrai si et seulement si la restriction de $(.,.)_J$ dans $\mathscr{L}$ est une forme nulle. Ainsi, tout sous espace de dimension un est isotrope.

— *L'espace $\mathscr{L}$ est dit coisotrope si*

$$\mathscr{L}^{\perp'} \cap \mathscr{L} = \mathscr{L}^{\perp'}.$$

L'espace $\mathscr{L}$ est coisotrope si et seulement si $\mathscr{L}^{\perp'}$ est isotrope. Ainsi, tout sous espace de codimension un est coisotrope.

— *L'espace $\mathscr{L}$ est dit Lagrangien si*

$$\mathscr{L} = \mathscr{L}^{\perp'}.$$

Un espace est Lagrangien si et seulement si $\mathscr{L}$ est à la fois isotrope et coisotrope. En dimension finie, le sous-espace Lagrangien d'un espace vectoriel de dimension $2n$ est un sous-espace isotrope de dimension n.

Tout espace Euclidien E de dimension fini peut s'écrire comme une somme directe d'un sous-espace unidimensionnel et son complément orthogonal :

$$\forall\, v \in E\backslash\{0\},\ \ E = \langle v\rangle \bigoplus \langle v\rangle^{\perp}.$$

Ainsi, l'espace Euclidien $\mathbb{R}^n$ peut être représenté comme somme directe des droites orthogonales, c'est-à-dire

$$\exists v_1 \dots v_n \in \mathbb{R}^n \text{ tel que } \mathbb{R}^n = \bigoplus_{i=1}^{n} vect\{v_i\},\ v_i \perp v_j,\ \forall 1 \leqslant i \neq j \leqslant n.$$

Contrairement au cas Euclidien, l'espace symplectique $\mathbb{R}^{2n}$ ne peut jamais être écrit comme une somme directe de sous-espaces symplectiques unidimensionnels. Cela est dû au fait que tout espace unidimensionnel est isotrope, c'est-à-dire

$$\forall v \in E \quad \langle v \rangle \subset \langle v \rangle^{\perp'}.$$

Cependant, cette décomposition peut être obtenue en utilisant des sous-espaces bidimensionnels (les plans). En fait, pour tout sous espace symplectique $\langle v_1, v_2 \rangle$ de $\mathbb{R}^{2n}$, nous avons

$$\mathbb{R}^{2n} = \langle v_1, v_2 \rangle \bigoplus \langle v_1, v_2 \rangle^{\perp'}.$$

En conséquence, l'espace linéaire symplectique $\mathbb{R}^{2n}$ peut s'écrire comme une somme directe des plans symplectiques, c'est-à-dire

$$\exists v_i \in \mathbb{R}^{2n},\ i = 1, \dots, 2n, \text{ tels que } \mathbb{R}^{2n} = \bigoplus_{i=1}^{n} \langle v_{2i-1}, v_{2i} \rangle,$$

lorsque la matrice $[v_{2i-1}, v_{2i}]$ est une matrice symplectique pour $1 \leqslant i \leqslant n$ et tel que

$$1 \leqslant i \neq j \leqslant n, \quad \langle v_{2i-1}, v_{2i} \rangle \perp' \langle v_{2j-1}, v_{2j} \rangle.$$

1.4 Les matrices structurées

Définition 1.4.1. *Soit la matrice $A \in \mathbb{R}^{n \times n}$. Nous avons les définitions suivantes :*

- *Soit $\lambda \in \mathbb{C}$. Si $\det(A - \lambda I) = 0$, alors λ est une valeur propre de la matrice A.*
- *Le spectre de la matrice A est défini par*

$$\sigma(A) = \{\lambda \in \mathbb{C} \text{ tel que } \det(A - \lambda I) = 0\}.$$

- *Le rayon spectral de la matrice A est défini par*

$$\rho(A) = \max\{|\lambda| \text{ tel que } \lambda \in \sigma(A)\}.$$

- *Les normes sont un outil indispensable dans l'algèbre linéaire numérique. Nous décrivons certaines de leurs propriétés les plus utiles et les plus intéressantes.*
  - ▷ *La norme d'un vecteur est définie par la fonction*

$$\begin{aligned} \|.\| : \mathbb{C}^n &\longrightarrow \mathbb{R} \\ x &\longmapsto \|x\|, \end{aligned}$$

tels que

◇ *la norme* 1

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|.$$

◇ *la norme* 2

$$\|x\|_2 = (x^{\mathrm{T}}x)^{\frac{1}{2}} = \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}}.$$

◇ *la norme* ∞

$$\|x\|_\infty = \max_{1\leq i\leq n} |x_i|.$$

▷ *La norme matricielle est définie par la fonction*

$$\begin{aligned} \|.\| : \mathbb{C}^{m\times n} &\longrightarrow \mathbb{R} \\ A &\longmapsto \|A\|, \end{aligned}$$

tels que

◇ *la norme de Frobenius*

$$\|A\|_F = \left(\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}|^2}\right)^{\frac{1}{2}}.$$

◇ *la norme* 1 *d'une matrice*

$$\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}|.$$

◇ *la norme* ∞ *d'une matrice*

$$\|A\|_\infty = \max_{1\leq i\leq m} \sum_{i=1}^{n} |a_{ij}|.$$

◇ *la norme* 2 *d'une matrice (la norme spectrale)*

$$\|A\|_2 = (\rho(A^*A))^{\frac{1}{2}} = \sigma_{\max}(A),$$

avec $\sigma_{\max}(A)$ *est la plus grande valeur singulière de la matrice* A.

— *Le conditionnement de la matrice* $A \in \mathbb{K}^{n\times n}$ *en utilisant la norme* 2 *est désigné par*

$$\kappa_2(A) = \mathrm{cond}_2(A) = \|A\|_2\|A^{-1}\|_2,$$

et d'une manière générale en utilisant la norme $\|.\|_i$ *le conditionnement de la matrice* A *s'écrit comme suit*

$$\kappa_i(A) = \mathrm{cond}_i(A) = \|A\|_i\|A^{-1}\|_i.$$

— *La trace d'une matrice* A

$$\mathrm{trace}(A) = \sum_{k=1}^{n} a_{kk} = \sum_{k=1}^{n} \lambda_k,$$

avec les λ_k *sont les valeurs propres de la matrice* A.

— *Pour $A \in \mathbb{R}^{n\times n}$, $v \in \mathbb{R}^n$ et $j \in \mathbb{N}$, nous notons par $K(A,v,j)$ la matrice de Krylov de taille $n \times j$ tel que*

$$K(A,v,j) = [v, Av, A^2v, \ldots, A^{j-1}v] \in \mathbb{R}^{n\times j}.$$

Définition 1.4.2. *Soit une matrice $A \in \mathbb{K}^{n\times n}$ dont les coefficients sont notés par $(a_{ij})_{i,j=1,\ldots,n}$. La matrice A est dite :*

— *Symétrique si $A^{\mathrm{T}} = A$.*

— *Orthogonale si $\mathbb{K} = \mathbb{R}$ et $A^{\mathrm{T}}A = AA^{\mathrm{T}} = I_n$.*

— *Unitaire si $\mathbb{K} = \mathbb{C}$ et $A^*A = AA^* = I_n$.*

— *Diagonale si $a_{ij} = 0$ pour $i \neq j$ et $i, j = 1, \ldots, n$ tel que*

$$A = \begin{pmatrix} & & \\ & \ddots & \\ & & \end{pmatrix}.$$

— *Hessenberg si $a_{ij} = 0$ pour $i > j+1$ avec $i, j = 1 \ldots n$ tel que*

$$A = \begin{pmatrix} & & \\ & \ddots & \\ & & \end{pmatrix}.$$

— *Triangulaire supérieure si $a_{ij} = 0$ pour $i > j$ avec $i, j = 1, \ldots, n$ tel que*

$$A = \begin{pmatrix} & & \\ & \ddots & \\ & & \end{pmatrix}.$$

— *Strictement triangulaire supérieure si $a_{ij} = 0$ pour $i \geq j$ avec $i, j = 1, \ldots, n$ tel que*

$$A = \begin{pmatrix} 0 & & \\ & \ddots & \\ & & 0 \end{pmatrix}.$$

— *Tridiagonale si $a_{ij} = 0$ pour $i > j+1$ et $i < j-1$ avec $i, j = 1, \ldots, n$ tel que*

$$A = \begin{pmatrix} & & \\ & \ddots & \\ & & \end{pmatrix}.$$

Nous rappelons que dans le cas Euclidien toute matrice réelle A est semblable à une matrice H de Hessenberg, via une matrice de similarité orthogonale, c'est-à-dire qu'il existe une matrice orthogonale notée Q tel que

$$H = QAQ^{\mathrm{T}}.$$

La matrice H est dite une matrice de Hessenberg supérieure irréductible si elle est une matrice de Hessenberg supérieure avec $a_{i,i-1} \neq 0$ pour $i = 2 \dots n$.

Définition 1.4.3. *Soit $A \in \mathbb{R}^{2n\times 2n}$ une matrice tel que*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

où les blocs $A_{ij} \in \mathbb{R}^{n\times n}$ pour $i, j = 1, 2$, alors nous avons :

— *Une matrice A est dite une matrice de J-Hessenberg supérieure si les blocs A_{11}, A_{21} et A_{22} sont des matrices triangulaires supérieures et le bloc A_{12} est une matrice de Hessenberg supérieure, c'est-à-dire la matrice A est sous la forme*

$$A = \begin{pmatrix} \diagdown & \diagdown\!\diagdown \\ \diagdown & \diagdown \end{pmatrix}.$$

Dans le cas symplectique, l'équivalent de la forme Hessenberg est la forme J-Hessenberg.

— *La matrice A est une matrice de J-Hessenberg supérieure irréductible si la matrice A est une matrice J-Hessenberg, le bloc A_{21} est une matrice non singulière et le bloc A_{12} est une matrice de Hessenberg supérieure irréductible, c'est-à-dire elle n'a aucune entrée zéro dans sa première sous-diagonale.*

— *La matrice A est une matrice J-triangulaire supérieure si les blocs A_{11}, A_{12} et A_{22} sont des matrices triangulaires supérieures et le bloc A_{21} est une matrice strictement triangulaire supérieure, c'est-à-dire elle a une diagonale nulle tel que :*

$$A = \begin{pmatrix} \diagdown & \diagdown \\ 0\ \diagdown & \diagdown \\ \quad 0 & \end{pmatrix}.$$

— *La matrice A est une matrice J-triangulaire inférieure si la matrice A^{T} est une matrice J-triangulaire supérieure tel que :*

$$A = \begin{pmatrix} \diagdown & 0\ \diagdown \\ \diagdown & \quad 0 \end{pmatrix}.$$

— *La matrice A est une matrice J-tridiagonale si les blocs A_{11}, A_{21} et A_{22} sont des matrices diagonales et le bloc A_{12} est une matrice tridiagonale tel que :*

$$A = \begin{pmatrix} \diagdown & \diagdown\!\diagdown\!\diagdown \\ \diagdown & \diagdown \end{pmatrix}.$$

Lemme 1.4.1. *Soit la matrice A, $B \in \mathbb{R}^{2n\times 2n}$ et la matrice P est la permutation (1.2) présentée par $P = [e_1, e_3, \ldots, e_{2n-1}, e_2, e_4, \ldots, e_{2n}] \in \mathbb{R}^{2n\times 2n}$. Alors, nous avons les résultats suivants :*

1. *Si la matrice A est une matrice de J-Hessenberg supérieure (J-Hessenberg supérieure irréductible respectivement), alors la matrice PAP^{T} est une matrice de Hessenberg supérieure (Hessenberg supérieure irréductible respectivement) tel que :*

$$PAP^{\mathrm{T}} = \begin{pmatrix} & \diagdown \\ & \diagdown\!\diagdown \\ \diagdown & \diagdown \end{pmatrix}.$$

2. *Si la matrice A est une matrice J-triangulaire supérieure, alors la matrice PAP^{T} est une matrice triangulaire supérieure tel que*

$$PAP^{\mathrm{T}} = \begin{pmatrix} & \diagdown \\ & \diagdown \\ \diagdown & \diagdown \end{pmatrix}.$$

3. *Si la matrice A est une matrice J-triangulaire, alors la matrice A^{-1} est une matrice J-triangulaire.*
4. *Si les matrices A et B sont deux matrices J-triangulaires, alors la matrice AB est une matrice J-triangulaire.*
5. *Si la matrice A est une matrice de J-Hessenberg et la matrice B est une matrice J-triangulaire, alors les matrices AB et BA sont deux matrices de J-Hessenberg.*

1.4.1 La matrice symplectique

Définition 1.4.4. *Une matrice réelle $S \in \mathbb{R}^{2n\times 2k}$ est dite symplectique si et seulement si elle satisfait la condition*

$$S^J S = I_{2k},$$

ou de façon équivalente

$$S^{\mathrm{T}} J_{2n} S = J_{2k},$$

avec

$$S^J = J_{2k}^{\mathrm{T}} S^{\mathrm{T}} J_{2n}.$$

Proposition 1.4.1.

1. *Soient $A \in \mathbb{R}^{2n\times 2k}$ et $B \in \mathbb{R}^{2k\times 2p}$ deux matrices symplectiques, alors la matrice $AB \in \mathbb{R}^{2n\times 2p}$ est aussi une matrice symplectique.*
2. *Toute matrice $A \in \mathbb{R}^{2n\times 2k}$ symplectique est inversible et la matrice inverse A^{-1} est donnée par son adjoint tel que*

$$A^{-1} = A^J = J_{2k}^{\mathrm{T}} S^{\mathrm{T}} J_{2n}.$$

3. *Soit $A \in \mathbb{R}^{2n\times 2k}$ une matrice symplectique réelle. Alors,*

$$\det(A) = \pm 1.$$

Preuve.

1. Soient A et B deux matrices symplectiques, alors

$$\begin{aligned}(AB)^{\mathrm{T}} J_{2n}(AB) &= B^{\mathrm{T}}(A^{\mathrm{T}} J_{2n} A)B \\ &= B^{\mathrm{T}} J_{2k} B \\ &= J_{2p}.\end{aligned}$$

   Donc, la matrice AB est symplectique.
2. Soit A une matrice symplectique, alors $A^J A = I_{2k}$. Ainsi, $A^{-1} = A^J = J_{2k}^{\mathrm{T}} S^{\mathrm{T}} J_{2n}$.
3. Soit A une matrice symplectique, alors $A^{\mathrm{T}} J A = J$. En passant au déterminant nous avons $\det(A^{\mathrm{T}} J A) = \det(A^{\mathrm{T}})\det(J)\det(A) = \det(J)\det(A)^2 = \det(J)$. Par conséquence, $\det(A) = \pm 1$.

□

Remarque 1.4.1. *Une matrice A est dite triviale si elle est à la fois symplectique et J-triangulaire et a la forme*

$$A = \begin{pmatrix} C^{-1} & F \\ 0 & C \end{pmatrix},$$

avec les blocs C, $F \in \mathbb{R}^{n\times n}$ sont des matrices diagonales.

Les matrices symplectiques peuvent être considérées comme les matrices orthogonales par rapport au produit antisymétrique $(.,.)_J$. Nous rappelons que les matrices symplectiques sont également appelées les matrices J-orthogonales.

Définition 1.4.5. *Soit $A \in \mathbb{R}^{2n\times 2n}$ une matrice. La décomposition SR consiste à écrire la matrice A comme produit de deux matrices tel que*

$$A = SR = S\begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix} = S\begin{pmatrix} \blacksquare & \blacksquare \\ 0 & \blacksquare \end{pmatrix}.$$

sachant que $S \in \mathbb{R}^{2n\times 2n}$ est une matrice symplectique et $R \in \mathbb{R}^{2n\times 2n}$ est une matrice J-triangulaire, c'est-à-dire les blocs R_{11}, R_{12}, R_{22} sont des matrices triangulaires supérieures et le bloc R_{21} est une matrice strictement triangulaire supérieure avec $R_{ij} \in \mathbb{R}^{n\times n}$.

Presque chaque matrice $A \in \mathbb{R}^{2n\times 2n}$ peut être décomposée en SR. La décomposition SR est essentiellement unique au sens symplectique. Nous rappelons que la décomposition SR a été introduite pour la première fois par Della-Dora [47, 48]. Contrairement à la décomposition QR, cette décomposition n'existe pas toujours puisque l'ensemble des matrices qui peuvent être décomposées de cette manière est dense dans $\mathbb{R}^{2n\times 2n}$.

Théorème 1.4.1. *Soit $A \in \mathbb{R}^{2n\times 2n}$ une matrice non singulière. Alors, il existe une matrice symplectique S et une matrice J-triangulaire R telle que $A = SR$ si et seulement si tous les principaux mineurs de dimension paire de la matrice $PA^{\mathrm{T}}JAP^{\mathrm{T}}$ sont non nuls où P est la matrice de permutation (1.2). L'ensemble des matrices de taille $2n \times 2n$ décomposables en SR est dense dans $\mathbb{R}^{2n\times 2n}$.*

Remarque 1.4.2. *Les valeurs propres d'une matrice symplectique A se présentent en paires réciproques (en paire $\{\lambda, \lambda^{-1}\}$ si λ est une valeur propre réelle ou imaginaire pure et en quadruple $\{\lambda, \lambda^{-1}, \bar{\lambda}, \bar{\lambda}^{-1}\}$ si λ est une valeur propre complexe avec une partie réelle non nulle), c'est-à-dire si λ est une valeur propre de la matrice A avec vecteur propre v, alors λ^{-1} est aussi une valeur propre de A avec un vecteur propre à gauche Jv^{T}.*

1.4.2 La matrice Hamiltonienne

Définition 1.4.6. *Une matrice réelle $H \in \mathbb{R}^{2n\times 2n}$ est dite matrice Hamiltonienne si*

$$H^J = -H,$$

c'est-à-dire

$$HJ = (HJ)^{\mathrm{T}}.$$

En effet, toute matrice Hamiltonienne s'écrit sous la forme :

$$H = \begin{pmatrix} A & G \\ Q & -A^{\mathrm{T}} \end{pmatrix} \tag{1.11}$$

avec les blocs G et Q sont deux matrices symétriques $G = G^{\mathrm{T}}$, $Q = Q^{\mathrm{T}}$ et $A,\ G,\ Q \in \mathbb{R}^{n\times n}$.

Proposition 1.4.2.

- *La transposée d'une matrice Hamiltonienne est une matrice Hamiltonienne.*
- *La trace d'une matrice Hamiltonienne est nulle.*
- *Les valeurs propres d'une matrice Hamiltonienne se présentent en paires $\{\lambda, -\lambda\}$ pour les valeurs propres réelles et purement imaginaires et en quadruples $\{\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}\}$ pour les valeurs propres complexes avec une partie réelle non nulle. Autrement dit, le spectre de toute matrice Hamiltonienne est symétrique par rapport aux axes des réels et des imaginaires.*

Remarque 1.4.3.

— *Si A est une matrice Hamiltonienne et S est une matrice Symplectique, alors $S^{-1}AS$ est une matrice Hamiltonienne.*

— *Si une matrice T a la forme Hamiltonienne et J-Hessenberg, alors T est une matrice J-tridiagonale :*

$$T = \left(\begin{array}{cccc|cccc} a_1 & & & & c_1 & b_1 & & \\ & a_2 & & & b_1 & \ddots & \ddots & \\ & & \ddots & & & \ddots & \ddots & b_{n-1} \\ & & & a_n & & & b_{n-1} & c_n \\ \hline q_1 & & & & -a_1 & & & \\ & q_2 & & & & -a_2 & & \\ & & \ddots & & & & \ddots & \\ & & & q_n & & & & -a_n \end{array}\right).$$

— *Si T est une matrice J-tridiagonale (Hamiltonienne et J-Hessenberg), alors*

$$PTP^{\mathrm{T}} = \begin{pmatrix} a_1 & c_1 & 0 & b_1 & 0 & \cdots & \cdots & 0 \\ q_1 & -a_1 & 0 & 0 & 0 & \ddots & & \vdots \\ 0 & b_1 & a_2 & c_2 & 0 & b_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & b_{n-1} \\ \vdots & & & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \ddots & b_{n-1} & a_n & c_n \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & q_n & -a_n \end{pmatrix},$$

avec $P = [e_1, e_3, \ldots, e_{2n-1}, e_2, e_4, \ldots, e_{2n}] \in \mathbb{R}^{2n\times 2n}$ est la permutation de (1.2).

1.4.3 La matrice anti-Hamiltonienne

Définition 1.4.7. *Une matrice réelle $H \in \mathbb{R}^{2n\times 2n}$ est dite matrice anti-Hamiltonienne si*

$$H^J = H,$$

c'est-à-dire

$$HJ = -(HJ)^{\mathrm{T}}.$$

En effet, toute matrice anti-Hamiltonienne s'écrit sous la forme :

$$H = \begin{pmatrix} A & G \\ Q & A^{\mathrm{T}} \end{pmatrix} \tag{1.12}$$

avec les blocs G et Q sont deux matrices antisymétriques $G = -G^{\mathrm{T}}$, $Q = -Q^{\mathrm{T}}$ et les matrices $A,\ G,\ Q \in \mathbb{R}^{n\times n}$.

Proposition 1.4.3.

— *Si la matrice A est une matrice Hamiltonienne, alors la matrice A^2 est une matrice anti-Hamiltonienne.*

— *Si la matrice S est une matrice symplectique, alors la matrice $S+S^{-1}$ est une matrice anti-Hamiltonienne.*

Preuve.

— Soit la matrice $B = A^2$ et comme la matrice A est une matrice Hamiltonienne, alors $A^J = -A$. Ainsi, nous avons $B^J = A^J A^J = A^2 = B$. Donc, $B^J = B$ et par suite la matrice A^2 est une matrice anti-Hamiltonienne.

— Nous avons $(S+S^{-1})^J = S^J + (S^{-1})^J = S^J + S = S + S^{-1}$. Donc, la matrice $S+S^{-1}$ est une matrice anti-Hamiltonienne.

□

1.5 Les transformations de Householder symplectiques

Soit $\mathbb{R}^\nu$ un espace vectoriel tel que ν est pair pour l'espace vectoriel symplectique.

Définition 1.5.1. *Soit la transformation $T : \mathbb{R}^\nu \longrightarrow \mathbb{R}^\nu$ est dite une transvection si elle vérifie*

$$\exists v \in \mathbb{R}^\nu, \forall x \in \mathbb{R}^\nu, T(x) = x + \varphi(x)\, v,$$

avec φ est une forme linéaire.

Une transvection orthogonale T est fréquemment appelée transformation de Householder. Elle est de la forme

$$T = I - 2vv^{\mathrm{T}}, \text{ avec } v \in \mathbb{R}^\nu, v^{\mathrm{T}}v = 1. \tag{1.13}$$

Ce qui nous amène à la définition suivante :

Définition 1.5.2. *Une transvection symplectique est appelée transformation de Householder symplectique.*

Lemme 1.5.1. *Une transvection T est dite symplectique si et seulement si elle est de forme*

$$T = I + cvv^J, \text{ avec } c \in \mathbb{R} \text{ et } v \in \mathbb{R}^\nu \text{ tel que } \nu \text{ est paire.} \tag{1.14}$$

Le vecteur v est appelé la direction de la transformation T.

Preuve. Soit T une transvection symplectique. De plus, T est une isométrie par rapport au produit scalaire antisymétrique $(.,.)_J$. Alors, nous avons

$$\forall x \in \mathbb{R}^{2n},\ \forall y \in \mathbb{R}^{2n}\ (T(x), T(y))_J = (x, y)_J.$$

En simplifiant par $x^J y$ et en utilisant $v^J v = 0$, nous obtenons

$$\forall x \in \mathbb{R}^{2n},\ \forall y \in \mathbb{R}^{2n}\ \varphi(x)\, v^J y = \varphi(y)\, v^J x.$$

Soit $y_0 \in \mathbb{R}^{2n}$ tel que $v^J y_0 \neq 0$. Nous obtenons $\varphi(x) = \dfrac{\varphi(y_0)}{v^J y_0} v^J x$. Ainsi, $\exists c \in \mathbb{R}$ tel que

$$T = I + c v v^J.$$

Réciproquement, $I + c v v^J$ est évidemment une transvection symplectique. □

Proposition 1.5.1. *La transformation de Householder symplectique $T = I + c v v^J$ satisfait $T^J = I - c v v^J$.*

Nous pouvons noter que les expressions algébriques des transvections orthogonales et symplectiques sont fondamentalement différentes.

Dans le cas Euclidien, nous avons le théorème suivant :

Théorème 1.5.1. *Il existe une transvection orthogonale T de direction $v = \dfrac{y-x}{\|y-x\|_2}$ qui transforme x en y si $\|x\|_2 = \|y\|_2$.*

Preuve. Soit T une transvection orthogonale donnée par (1.13) avec une direction $v = \dfrac{y-x}{\|y-x\|_2}$. Comme nous avons $\|v\|_2 = 1$, alors $T(x) = x + 2vv^{\mathrm{T}}$. De plus, nous avons $v^{\mathrm{T}}x = \dfrac{y^{\mathrm{T}}x - \|x\|_2^2}{\|y-x\|_2}$ et la condition $\|x\|_2 = \|y\|_2$ implique que $vv^{\mathrm{T}}x = (y-x)\dfrac{y^{\mathrm{T}}x - \|x\|_2^2}{\|y-x\|_2^2} = -\dfrac{1}{2}(y-x)$. Donc, nous avons le résultat $T(x) = x + 2vv^{\mathrm{T}}x = x + (y-x) = y$. □

Dans le cas symplectique nous avons le théorème suivant :

Théorème 1.5.2. *Il existe une transvection symplectique T qui transforme x en y si $x = y$ ou bien si $x^J y \neq 0$ de direction $y - x$. De plus, dans le cas où $x^J y \neq 0$ la transvection est donnée par*

$$T = I - \frac{1}{x^J y}(y-x)(y-x)^J.$$

Preuve. Dans le cas où $x = y$, le résultat est évident. Nous supposons que $x \neq y$ et nous posons que

$$T = I + c(y-x)(y-x)^J.$$

Pour $y = T(x) = x + c(y-x)(y-x)^J x$, nous avons $y - x = c(y-x)(y-x)^J x = c(y-x)y^J x$. Alors, $c = \dfrac{1}{y^J x} = -\dfrac{1}{x^J y}$. □

Théorème 1.5.3. *Tout vecteur non nul x peut être transformé en un vecteur non nul y par au plus deux transvections symplectiques.*

Preuve. (voir [102]) □

Théorème 1.5.4. *Toute transvection symplectique T est une rotation et* $\det(T) = 1$.

Preuve. Soit la transvection symplectique $T_c = I + c v v^J$. Comme $T_c^J T_c = I$, alors nous aurons $\det(T)^2 = 1$. Donc, nous obtenons $\det(T) = \pm 1$. Puisque $T_c = T_{\frac{c}{2}}^2$, alors $\det(T) = 1$. □

1.5.1 Les transformations de Householder symplectiques optimales

Lemme 1.5.2. *Soit T une transformation de Householder symplectique non triviale, c'est à dire $c \neq 0$ et $v \neq 0$, tel que*

$$T = I + cvv^J$$

alors le conditionnement de cette transformation selon la norme $\|.\|_2$ est :

$$\kappa_2(T) = \frac{2 + c^2\|v\|_2^4 + \sqrt{c^4\|v\|_2^8 + 4c^2\|v\|_2^4}}{2 + c^2\|v\|_2^4 - \sqrt{c^4\|v\|_2^8 + 4c^2\|v\|_2^4}}. \tag{1.15}$$

Dans ce cas, pour minimiser le conditionnement de la transformation T, il suffit de minimiser $c^2\|v\|_2^4$.

Preuve. Soit $T = I + cvv^J$ une transformation de Householder symplectique. L'inverse de cette transformation est $T^{-1} = T^J = J^{\mathrm{T}}T^{\mathrm{T}}J$. Ainsi $\|T^{-1}\|_2 = \|T^{\mathrm{T}}\|_2 = \|T\|_2$. Donc

$$\kappa_2(T) = \|T\|_2\|T^{-1}\|_2 = \|T\|_2^2.$$

Par suite, la minimisation de conditionnement de T est équivalent à la minimisation de la norme de T.

$$\begin{aligned}
T^{\mathrm{T}}T &= (I + cvv^J)^{\mathrm{T}}(I + cvv^J) \\
&= (I + cvv^{\mathrm{T}}J)^{\mathrm{T}}(I + cvv^{\mathrm{T}}J) \\
&= (I + cJ^{\mathrm{T}}vv^{\mathrm{T}})(I + cvv^{\mathrm{T}}J) \\
&= I + cvv^{\mathrm{T}}J + cJ^{\mathrm{T}}vv^{\mathrm{T}} + c^2J^{\mathrm{T}}vv^{\mathrm{T}}vv^{\mathrm{T}}J \\
&= I + cvv^{\mathrm{T}}J + cJ^{\mathrm{T}}vv^{\mathrm{T}} + c^2\|v\|_2^2J^{\mathrm{T}}vv^{\mathrm{T}}J.
\end{aligned}$$

Cette matrice est unitaire et similaire à la matrice

$$A = \begin{bmatrix} 1 & -c\|v\|_2^2 & 0 \\ -c\|v\|_2^2 & 1 + c^2\|v\|_2^4 & 0 \\ 0 & 0 & I \end{bmatrix},$$

avec la matrice unitaire de similarité qui est sous la forme :

$$Q = \begin{bmatrix} \frac{v}{\|v\|_2} & \frac{Jv}{\|v\|_2} & Q' \end{bmatrix}.$$

En effet,

$$\begin{aligned}
QAQ^{\mathrm{T}} &= \begin{bmatrix} \frac{v}{\|v\|_2} & \frac{Jv}{\|v\|_2} & Q' \end{bmatrix} \begin{bmatrix} 1 & -c\|v\|_2^2 & 0 \\ -c\|v\|_2^2 & 1 + c^2\|v\|_2^4 & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \frac{v^{\mathrm{T}}}{\|v\|_2} \\ \frac{v^{\mathrm{T}}J^{\mathrm{T}}}{\|v\|_2} \\ Q'^T \end{bmatrix}, \\
&= \begin{bmatrix} \frac{v}{\|v\|_2} & \frac{Jv}{\|v\|_2} & Q' \end{bmatrix} \begin{bmatrix} \frac{v^{\mathrm{T}}}{\|v\|_2} - c\|v\|_2v^{\mathrm{T}}J^{\mathrm{T}} \\ -c\|v\|_2v^{\mathrm{T}} + \frac{v^{\mathrm{T}}J^{\mathrm{T}}}{\|v\|_2} + c^2\|v\|_2^3v^{\mathrm{T}}J^{\mathrm{T}} \\ Q'^T \end{bmatrix}, \\
&= \frac{1}{\|v\|_2^2}vv^{\mathrm{T}} - cvv^{\mathrm{T}}J^{\mathrm{T}} - cJvv^{\mathrm{T}} + \frac{1}{\|v\|_2^2}Jvv^{\mathrm{T}}J^{\mathrm{T}} + c^2\|v\|_2^2Jvv^{\mathrm{T}}J^{\mathrm{T}} + I, \\
&= I + cvv^{\mathrm{T}}J + cJ^{\mathrm{T}}vv^{\mathrm{T}} + c^2\|v\|_2^2J^{\mathrm{T}}vv^{\mathrm{T}}J.
\end{aligned}$$

Il est clair que la matrice A admet 1 comme valeur propre de multiplicité $2n-1$. Les deux autres valeurs propres : soit $\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix}$ un vecteur propre associe à la valeur propre λ tel que

$$\begin{bmatrix}1 & -c\|v\|_2^2\\ -c\|v\|_2^2 & 1+c^2\|v\|_2^4\end{bmatrix}\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix}=\lambda\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix}.$$

Ceci est équivalent à

$$\begin{cases}\alpha_1 - c\|v\|_2^2\alpha_2 = & \lambda\alpha_1\\ -c\|v\|_2^2\alpha_1+\alpha_2+c^2\|v\|_2^4\alpha_2 = & \lambda\alpha_2\end{cases}.$$

Si $\lambda=1$ alors $\alpha_1=\alpha_2=0$. Si non, si $\lambda\neq 0$ nous aurons

$$\begin{cases}\alpha_1 = & \dfrac{c\|v\|_2^2}{(1-\lambda)}\alpha_2\\ -c\|v\|_2^2\alpha_1+\alpha_2+c^2\|v\|_2^4\alpha_2 = & \lambda\alpha_2\end{cases}.$$

D'où,

$$\begin{aligned}-c\|v\|_2^2\alpha_1+\alpha_2+c^2\|v\|_2^4\alpha_2&=\lambda\alpha_2\\ -\frac{1}{(1-\lambda)}c^2\|v\|_2^4\alpha_2+\alpha_2+c^2\|v\|_2^4\alpha_2&=\lambda\alpha_2\\ -\frac{1}{(1-\lambda)}c^2\|v\|_2^4+\alpha_2+c^2\|v\|_2^4-\lambda&=0\\ -c^2\|v\|_2^4+1-\lambda+c^2\|v\|_2^4-\lambda c^2\|v\|_2^4-\lambda+\lambda^2&=0\\ \lambda^2-(2+c^2\|v\|_2^4)\lambda+1&=0.\end{aligned}$$

Ainsi,

$$\begin{aligned}\Delta&=(2+c^2\|v\|_2^4)^2-4,\\ &=c^4\|v\|_2^8+4c^2\|v\|_2^4.\end{aligned}$$

Donc,

$$\begin{aligned}\lambda_1&=\frac{2+c^2\|v\|_2^4-\sqrt{c^4\|v\|_2^8+4c^2\|v\|_2^4}}{2} &<1,\\ \lambda_2&=\frac{2+c^2\|v\|_2^4+\sqrt{c^4\|v\|_2^8+4c^2\|v\|_2^4}}{2} &>1.\end{aligned}$$

Ainsi, sachant que 1 est une valeur propre de multiplicité $2n-2$, et comme de plus λ_1 et λ_2 sont deux valeurs singuliers tel que $\lambda_2\geq\lambda_1$, alors

$$\kappa_2(T)=\frac{\lambda_2}{\lambda_1}=\frac{2+c^2\|v\|_2^4+\sqrt{c^4\|v\|_2^8+4c^2\|v\|_2^4}}{2+c^2\|v\|_2^4-\sqrt{c^4\|v\|_2^8+4c^2\|v\|_2^4}}.$$

Soit la fonction

$$t(x)=\frac{2+x+\sqrt{x^2+4x}}{2+x-\sqrt{x^2+4x}},$$

où $x \geq 0$.

La dérivée de cette fonction est :

$$t'(x) = \frac{8}{\sqrt{x^2+4x}[2+x-\sqrt{x^2+4x}]^2},$$

c'est d'une fonction strictement positive pour tout $x \geq 0$. Le minimum de $t(x)$ est atteint pour $x = 0$.

Donc, pour minimiser

$$\kappa_2(T) = \frac{2 + c^2\|v\|_2^4 + \sqrt{c^4\|v\|_2^8 + 4c^2\|v\|_2^4}}{2 + c^2\|v\|_2^4 - \sqrt{c^4\|v\|_2^8 + 4c^2\|v\|_2^4}},$$

il suffit de minimiser $c^2\|v\|_2^4$. □

Remarque 1.5.1. *Les valeurs propres λ_1 et λ_2 vérifient que $\lambda_1\lambda_2 = 1$.*

1.6 Les matrices des transformations élémentaires symplectiques

En fait, nous utilisons trois types des transformations :

Le premier type c'est

$$H(k,w) = \begin{pmatrix} \mathrm{diag}(I_{k-1,}, P) & 0 \\ 0 & \mathrm{diag}(I_{k-1}, P) \end{pmatrix},$$

avec

$$P = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w,\ w \in \mathbb{R}^{n-k+1}.$$

Cette transformation $H(k,w)$ est la somme directe de deux matrices $n \times n$ de Householder ordinaires [116]. Nous notons par $H(k,w)$ la transformation de Householder symplectique au sens de Van Loan. Remarquons que

$$H(k,w)e_i = e_i,$$

pour $i = 1,\ldots,k-1$ et pour $i = n+1,\ldots,n+k-1$.

Le deuxième type est

$$J(k,c,s) = \begin{pmatrix} C & S \\ -S & C \end{pmatrix},$$

avec

$$c^2 + s^2 = 1,$$

et

$$C = \mathrm{diag}(I_{k-1}, c, I_{n-k}) = I_n + (c-1)e_k e_k^{\mathrm{T}},$$
$$S = \mathrm{diag}(0_{k-1}, s, 0_{n-k}) = se_k e_k^{\mathrm{T}},$$

pour $k \in \{1,\ldots,n\}$, $s,c \in \mathbb{R}$ et $C,S \in \mathbb{R}^{n\times n}$.

Cette transformation $J(k,c,s)$ est une transformation de Givens symplectique, c'est une rotation $2n\times 2n$ ordinaire de Givens dans les plans k et $n+k$ [116]. Nous notons par $J(k,c,s)$ la rotation de Givens symplectique au sens de Van Loan.

Le troisième type est la matrice transformation de Gauss symplectique

$$G(k,v) = \begin{pmatrix} D & V \\ 0 & D^{-1} \end{pmatrix},$$

où la matrice V est une matrice de taille $n\times n$ où seulement deux entrées non nulles dans les positions $(k,k-1)$ et $(k-1,k)$

$$V = \left(\frac{v}{(1+v^2)^{\frac{1}{4}}}\right)(e_{k-1}e_k^{\mathrm{T}} + e_k e_{k-1}^{\mathrm{T}}),$$

la matrice D est une matrice diagonale de taille $n\times n$ qui est différente de la matrice identité I seulement par les positions diagonales $k-1$ et k

$$D = I_n + \left(\frac{v}{(1+v^2)^{\frac{1}{4}}} - 1\right)(e_{k-1}^{\mathrm{T}}e_{k-1} + e_k e_k^{\mathrm{T}}).$$

Notons que

$$G(k,v)^{-1} = \begin{pmatrix} D^{-1} & -V \\ 0 & D \end{pmatrix}.$$

Remarque 1.6.1. *Les matrices $J(k,c,s)$ et $H(k,w)$ sont deux matrices orthogonales et symplectiques. La matrice $G(k,v)$ est une matrice symplectique mais non orthogonale et*

$$\kappa_2(G(k,u)) = (l+v^2)^{\frac{1}{2}} + |v|.$$

Notons que Bunse-Gerstner et Mehrmann dans [40] ont référé aux matrices des transformations élémentaires $J(k,c,s)$, $H(k,w)$ et $G(k,v)$ par algorithme J, algorithme H et algorithme G.

Les matrices $J(k,c,s)$ seront utilisées pour mettre un zéro dans une entrée de la moitié inférieure d'un vecteur.

Algorithme 1.1 Algorithme J

Étant donné k tel que $1 \le k \le n$. La matrice $J(k,c,s)$ annule l'entrée de la position $n+k$ du vecteur $a \in \mathbb{R}^{2n}$, c'est-à-dire si $b = J(k,c,s)a$, alors $b_{n+k} = 0$.

1: **Début**
2: $\alpha = \sqrt{a_k^2 + a_{n+k}^2}$;
3: **Si** $\alpha = 0$ **alors**
4: $\quad c = 1$;
5: $\quad s = 0$;
6: **Sinon**
7: $\quad c = \dfrac{a_k}{\alpha}$;
8: $\quad s = \dfrac{a_{n+k}}{\alpha}$;
9: **Fin Si**
10: **Fin**

Les matrices $H(k,w)$ sont utilisées pour éliminer des entrées dans la moitié supérieure d'un vecteur.

Algorithme 1.2 Algorithme H

Étant donné k tel que $1 \leq k \leq n$ et un vecteur $a \in \mathbb{R}^{2n}$. Cet algorithme détermine $w = (w_k, \ldots, w_n)^{\mathrm{T}} \in \mathbb{R}^{n-k+1}$ tel que si $b = H(k,w)a$, alors $b_i = 0$ pour $i \in \{k+1, \ldots, n\}$.

1: **Début**
2: $\alpha = \sqrt{a_k^2 + \ldots + a_n^2}$;
3: $w_k = a(k) + sign(a(k))\alpha$;
4: **Pour** $i = k+1, \ldots, n$ **faire**
5: $\quad w_i = a_i$;
6: **Fin Pour**
7: **Fin**

Les matrices $G(k,v)$ sont utilisées pour mettre un zéro dans les entrées qui ne peuvent pas être annihilées par des matrices d'éliminations symplectiques orthogonales. Les matrices $G(k,v)$ sont utilisées pour éliminer une entrée dans la moitié supérieure d'un vecteur spécial.

Algorithme 1.3 Algorithme G

Étant donné k tel que $1 \leq k \leq n$ et un vecteur $a \in \mathbb{R}^{2n}$ avec $a_{n+k-1} = 0$ seulement si $a_k = 0$. Cet algorithme détermine v tel que si $b = G(k,v)$ alors $b_k = 0$.

1: **Début**
2: **Si** $(a_k == 0)$ **alors**
3: $\quad v = 0$;
4: **Sinon**
5: $\quad v = -\dfrac{a_k}{a_{n+k-1}}$;
6: **Fin Si**
7: **Fin**

1.7 Les matrices de Toeplitz, Hankel et Bézout

Définition 1.7.1. *La matrice $T = \left[t_{i,j}\right]$ est une matrice de Toeplitz si $t_{i,j} = t_{i+k,j+k}$ pour tout k positif, c'est-à-dire les entrées de la matrice T sont constantes sur les diagonales descendantes. La matrice de Toeplitz est donc complètement définie par sa première ligne et sa première colonne. La matrice de Toeplitz est sous la forme :*

$$T = (t_{i-j})_{i,j=0}^{n-1} = \begin{pmatrix} t_0 & t_{-1} & \cdots & \cdots & \cdots & t_{1-n} \\ t_1 & t_0 & \ddots & & & \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & t_{-1} \\ t_{n-1} & t_{n-2} & \cdots & \cdots & t_1 & t_0 \end{pmatrix}.$$

Définition 1.7.2. *La matrice $H = \left[h_{i,j}\right]$ est une matrice de Hankel, si $h_{i,j} = h_{i-k,j+k}$ pour tout k, c'est-à-dire les entrées de la matrice H sont constantes sur les antidiagonales. La matrice de Hankel est complètement définie par sa première ligne et sa dernière colonne. La matrice de Hankel est sous la forme :*

$$H = (h_{i+j})_{i,j=0}^{n-1} = \begin{pmatrix} h_0 & h_1 & h_2 & \cdots & \cdots & h_{n-1} \\ h_1 & h_2 & \iddots & \iddots & \iddots & h_n \\ h_2 & \iddots & \iddots & \iddots & \iddots & \vdots \\ \vdots & \iddots & \iddots & \iddots & \iddots & \vdots \\ \vdots & \iddots & \iddots & \iddots & \iddots & \vdots \\ h_{n-1} & h_n & \cdots & \cdots & \cdots & h_{2n-1} \end{pmatrix}.$$

Corollaire 1.7.1. *Une matrice carrée de Hankel est une matrice symétrique.*

Définition 1.7.3. *Une matrice de Fourier d'ordre n est sous la forme :*

$$F_n = \frac{1}{\sqrt{n}}(\omega_n^{ij})_{0\leq i,j\leq n-1} = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & \cdots & \cdots & 1 \\ 1 & \omega_n & \cdots & \omega_n^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \omega_n^{n-1} & \cdots & \omega_n^{(n-1)^2} \end{pmatrix}.$$

avec ω_n est une racine $n^{ième}$ de l'unité.

Définition 1.7.4. *La matrice de Frobenius (ou bien la matrice compagnon) associée au polynôme unitaire $u(x) = \sum_{i=0}^{m} u_i x^i = x^m + \sum_{i=0}^{m-1} u_i x^i$ est sous la forme :*

$$F_u = \begin{pmatrix} 0 & \cdots & \cdots & 0 & -u_0 \\ 1 & \ddots & & \vdots & -u_1 \\ 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & 1 & -u_{m-1} \end{pmatrix}.$$

1.7.1 La matrice de Bézout associée à deux polynômes

Dans ce paragraphe nous présentons quelques définitions de la matrice du Bézout et ses propriétés :

Définition 1.7.5. *Soient $u(x) = \sum_{i=0}^{n} u_i x^i$ et $v(x) = \sum_{i=0}^{m} v_i x^i$ deux polynômes avec $n \geq m$. Nous appelons matrice de Bézout associée aux deux polynômes $u(x)$ et $v(x)$ la matrice symétrique :*

$$B(u,v) = \begin{pmatrix} b_{0,0} & \cdots & b_{0,n-1} \\ \vdots & & \vdots \\ b_{n-1,0} & \cdots & b_{n-1,n-1} \end{pmatrix}, \tag{1.16}$$

avec les entrées $b_{i,j}$ sont définis par l'expression de Cayley tel que :

$$b(x,y) = \frac{u(x)v(y) - u(y)v(x)}{x-y} = \sum_{i,j=0}^{n-1} b_{i,j} x^i y^j, \tag{1.17}$$

$$= [1, x, x^2, \cdots, x^{n-1}] B(u,v) [1, y, y^2, \cdots, y^{n-1}]^{\mathrm{T}}. \tag{1.18}$$

D'où $B(u,v) = (b_{i,j})_{i,j}$.

Définition 1.7.6. *La représentation matricielle : la matrice de Bézout est représentée par :*

$$B(u,v) = uH(u_1,\ldots,u_n)uT(v_0,\ldots,v_{n-1}) - uH(v_1,\ldots,v_n)uT(u_0,\ldots,u_{n-1}), \tag{1.19}$$

$$B(u,v) = \begin{pmatrix} u_1 & \cdots & u_n \\ \vdots & \iddots & \\ u_n & & 0 \end{pmatrix} \begin{pmatrix} v_0 & \cdots & v_{n-1} \\ & \ddots & \vdots \\ 0 & & v_0 \end{pmatrix} - \begin{pmatrix} v_1 & \cdots & v_n \\ \vdots & \iddots & \\ v_n & & 0 \end{pmatrix} \begin{pmatrix} u_0 & \cdots & u_{n-1} \\ & \ddots & \vdots \\ 0 & & u_0 \end{pmatrix},$$

telle que pour $m < n$ les coefficients $v_{m+1}, \ldots, v_n$ sont supposés nuls.

Remarque 1.7.1. *Nous remarquons que :*

$$B(u,1) = uH(u_1,\ldots,u_n) = \begin{pmatrix} u_1 & \cdots & u_n \\ \vdots & \iddots & \\ u_n & & 0 \end{pmatrix}, \tag{1.20}$$

$$B(u,x^n) = -lH(u_0,\ldots,u_{n-1}) = -\begin{pmatrix} 0 & & u_0 \\ & \iddots & \vdots \\ u_0 & \cdots & u_{n-1} \end{pmatrix}. \tag{1.21}$$

Définition 1.7.7. *Les composantes de la matrice de Bézout $B(u,v)$ sont récursivement calculées avec l'équation suivante :*

$$b_{i,j+1} = b_{i+1,j} + u_i v_j - u_j v_i,$$

tels que $b_{n+1,k} = b_{0,k} = 0$ pour tout i, j et $v_{m+1} = \cdots = v_n = 0$.

Proposition 1.7.1. *Nous avons les propriétés suivantes :*

1. *La matrice $B(u,v)$ est une matrice symétrique d'ordre N où $N = \max(n,m)$.*
2. *$B(u,v) = -B(v,u)$.*
3. *$B(u,v)$ est linéaire par rapport aux polynômes u et v tels que :*

$$B(\alpha u + \beta f, v) = \alpha B(u,v) + \beta B(f,v),$$

$$B(u, \alpha v + \beta g) = \alpha B(u,v) + \beta B(u,g),$$

pour tout $f(x)$ et $g(x)$ deux polynômes et pour tout α et β deux scalaires.

Remarque 1.7.2. *Si $n = m$ alors $B(u,v) = B(u(x), u(x) \bmod (v(x)))$.*

Proposition 1.7.2. *Soient $u(x)$ et $v(x)$ deux polynômes, alors nous avons les propriétés suivantes :*

1. *la matrice $B(u,v)$ est inversible si et seulement si les polynômes $u(x)$ et $v(x)$ sont premiers entre eux.*
$$\dim(\ker(B(u,v))) = \deg(\mathrm{PGCD}(u,v)). \tag{1.22}$$
2. *La matrice du Bézout associée aux polynômes $u(x)$ et $v(x)$ peut être factorisée comme suit :*
$$B(u,v) = B(u,1)\,v(F_u). \tag{1.23}$$
où $v(F_u) = \sum_{i=0}^{m} v_i (F_u)^m$ et $u(x)$ est supposée être unitaire avec F_u la matrice de Frobenius associée au polynôme unitaire $u(x)$.

1.7.2 La matrice de Hankel associée à deux polynômes

Dans ce paragraphe nous présentons quelques définitions de la matrice du Hankel et ses propriétés :

Définition 1.7.8. *Soient $u(x) = \sum_{i=0}^{n} u_i x^i$, $v(x) = \sum_{i=0}^{m} v_i x^i$ deux polynômes avec $n \geq m$. Nous appelons matrice du Hankel associée aux polynômes $u(x)$ et $v(x)$ la matrice symétrique*

$$H(u,v) = H(h_0, \cdots, h_{2n-2}) = \begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \iddots & h_n \\ \vdots & \iddots & \iddots & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{pmatrix}, \tag{1.24}$$

telle que les h_i sont définies par

$$R(x) = \frac{v(x)}{u(x)} = \sum_{i=0}^{\infty} h_i x^{-i-1}. \tag{1.25}$$

La définition originale est définie pour $n > m$. Mais pour travailler avec des polynômes du même degré, nous utilisons la remarque suivante pour étendre la définition à $n \geq m$:

Remarque 1.7.3. *Dans le cas où $n = m$, soient*

$$\tilde{v}(x) = v(x) - \frac{v_m}{u_n} u(x)$$

et

$$R(x) = \frac{v(x)}{u(x)} = \frac{v_m}{u_n} + \frac{\tilde{v}(x)}{u(x)} = h_0 + \sum_{i=1}^{\infty} h_i x^{-i},$$

avec $h_0 = \frac{v_m}{u_n}$. Ainsi, la matrice de Hankel associée aux deux polynômes $u(x)$ et $v(x)$ est sous la forme :

$$H(u,v) = \begin{pmatrix} h_1 & h_2 & \cdots & h_n \\ h_2 & h_3 & \iddots & h_{n+1} \\ \vdots & \iddots & \iddots & \vdots \\ h_n & h_{n+1} & \cdots & h_{2n-1} \end{pmatrix}.$$

Par suite, $H(u,v) = H(u, v \bmod(u)) = H(u,\tilde{v})$.

Les coefficients de la matrice de Hankel $H(u,v)$ sont liés aux coefficients des deux polynômes $u(x)$ et $v(x)$ par les systèmes linéaires suivants :

$$\begin{pmatrix} u_n & & & \\ u_{n-1} & u_n & & \\ u_{n-2} & u_{n-1} & u_n & \\ \vdots\ddots & \ddots & \ddots & \ddots \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} v_{n-1} \\ v_{n-2} \\ v_{n-3} \\ \vdots \end{pmatrix}, \tag{1.26}$$

avec la convention que $u_j = v_j = 0$ pour $j < 0$ et $v_i = 0$ pour $i > m$.

Ou bien

$$\begin{pmatrix} h_1 & h_2 & \cdots & h_n \\ h_2 & h_3 & \iddots & h_{n+1} \\ \vdots & \iddots & \iddots & \vdots \\ h_n & h_{n+1} & \cdots & h_{2n-1} \end{pmatrix} \begin{pmatrix} u_0 \\ \vdots \\ \vdots \\ u_{n-1} \end{pmatrix} = -u_n \begin{pmatrix} h_{n+1} \\ \vdots \\ \vdots \\ h_{2n} \end{pmatrix}. \tag{1.27}$$

Ou bien

$$\begin{pmatrix} v_{n-1} \\ v_{n-2} \\ \vdots \\ v_0 \end{pmatrix} = \begin{pmatrix} h_1 & 0 & \cdots & 0 \\ h_2 & h_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ h_n & \cdots & \cdots & h_1 \end{pmatrix} \begin{pmatrix} u_n \\ u_{n-1} \\ \vdots \\ u_1 \end{pmatrix}. \tag{1.28}$$

Ainsi, si la matrice de Hankel est une matrice triangulaire inférieure

$$H = \begin{pmatrix} 0 & \cdots & 0 & h_n \\ \vdots & \iddots & \iddots & h_{n+1} \\ 0 & \iddots & \iddots & \vdots \\ h_n & h_{n+1} & \cdots & h_{2n-1} \end{pmatrix},$$

alors

$$\begin{pmatrix} v_{n-1} \\ v_{n-2} \\ \vdots \\ v_0 \end{pmatrix} = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots \\ h_n & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} u_n \\ u_{n-1} \\ \vdots \\ u_1 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ h_n u_n \end{pmatrix}.$$

Par suite, $v(x)$ est constante. Soit alors $v(x) = v_0 = 1$ d'où $u_n = \frac{1}{h_n}$ et

$$H = H(u,1) = \begin{pmatrix} 0 & \cdots & 0 & h_n \\ \vdots & \iddots & \iddots & h_{n+1} \\ 0 & \iddots & \iddots & \vdots \\ h_n & h_{n+1} & \cdots & h_{2n-1} \end{pmatrix}.$$

1.8 Conclusion

Dans ce chapitre, nous avons présenté quelques outils nécessaires qui seront utiles pour la suite. Nous avons aussi rappelé quelques définitions et certaines propriétés des

matrices structurées telles que les matrices symplectique, Hamiltonienne, anti-Hamiltonienne, J-Hessenberg, Hankel, Bézout, etc.

Chapitre 2

La décomposition SR

« Tell me and I forget, teach me and I may remember, involve me and I learn »

Benjamin Franklin (1706-1790)

Sommaire

2.1 Introduction

La décomposition SR est une étape fondamentale pour résoudre de nombreux problèmes dans l'algèbre linéaire où les méthodes de préservation de la structure sont souhaitables.

Rappelons que la décomposition SR d'une matrice réelle $A \in \mathbb{R}^{2m\times 2n}$, avec $m \geq n$ et $\text{rang}(A) = 2n$, consiste à écrire cette matrice comme produit de deux matrices S et R tel que $A = SR$, avec la matrice S est une matrice symplectique (J-orthogonale, $S^J S = I$) et la matrice R est J-triangulaire : $R = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix}$, tel que les blocs R_{11}, R_{12}, R_{22} sont des matrices triangulaires supérieures et R_{21} est une matrice strictement triangulaire supérieure. Ainsi,

$$A = SR = S \begin{pmatrix} \ddots & \ddots \\ 0 \;\ddots & \ddots \\ \ddots\;0 & \ddots \end{pmatrix}.$$

La décomposition SR a été initialement introduite par Della-Dora [47, 48]. Cette décomposition est impliquée dans des nombreuses méthodes de préservation de la structure afin de préserver les propriétés géométriques de certaines fonctions matricielles ou dans les méthodes de type QR symplectiques pour résoudre le problème symplectique de calcul des valeurs et/ou des vecteurs propres d'une classe importante de matrices structurées, telles que les matrices Hamiltoniennes, anti-Hamiltoniennes et symplectiques [15, 18, 19, 52, 90, 114, 116].

Nous pouvons voir la décomposition SR comme l'analogue de la décomposition QR dans le cas Euclidien, mais en considérant l'espace symplectique : c'est-à-dire l'espace linéaire muni d'un produit scalaire antisymétrique $(.,.)_J$ au lieu du produit scalaire usuel. Le groupe orthogonal correspondant, avec respect au produit scalaire antisymétrique, est appelé le groupe symplectique. Contrairement au cas Euclidien, ce groupe n'est pas compact. Une détermination numérique de la forme canonique pour les matrices symplectiques a été introduite par Godunov et Sadkane dans [59].

Les méthodes générales de type QR, dans lesquelles les décompositions QR sont remplacées par d'autres décompositions, ont été étudiées par plusieurs auteurs [47, 52, 81, 116]. Ces décomposition doivent satisfaire plusieurs conditions pour conduire à un processus de calcul raisonnable. Deux types de décompositions symplectiques pour les matrices arbitraires de taille paires $2n \times 2n$ répondent à la plupart de ces exigences, voir [38]. La première version d'entre elles est la décomposition SR unitaire tel que $A = SR$, où la matrice S est une matrice symplectique et en plus unitaire et la matrice $R = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix}$, où les blocs R_{11} et R_{21} sont des matrices triangulaires supérieures et la matrice R_{21} a une diagonale nulle. L'autre version est la décomposition SR tel que $A = SR$ où la matrice S est une matrice symplectique et la matrice R est une matrice J-triangulaire supérieure. Cette dernière a déjà été introduite par Della-Dora dans [47] et elle peut en effet servir comme

une base pour les méthodes de type QR. Une version modifiée de cette méthode générale a été étudiée par Mehrmann dans [81]. La décomposition SR préserve la structure symplectique et permet de développer des implémentations rapides et efficaces.

Dans le cas Euclidien, le calcul de la décomposition QR est géré par deux types d'algorithmes. Le premier type est effectué par un processus d'orthogonalisation de type Gram–Schmidt [27, 28, 29, 30]. Ainsi, l'algorithme est plus sensible aux erreurs numériques alors que la version modifiée améliore sa performance [7, 30, 46]. Le deuxième type est effectué par des transformations de Householder [60, 97, 117, 118]. Puisque ce dernier est obtenu par un produit de matrices orthogonales, alors l'algorithme est plus précis.

Dans le cas symplectique, la décomposition SR peut être effectuée par deux types d'algorithmes. La première classe via l'algorithme de Gram–Schmidt symplectique (SGS) [99, 107]. Ces algorithmes et leurs versions modifiées sont habituellement impliquées dans les méthodes de sous-espaces de Krylov [18, 19] qui préservent la structure, pour les matrices structurées, creuses et de grandes tailles. La deuxième classe est construite à partir d'une variété de transformations élémentaires symplectiques. Chaque choix, de telles transformations, conduit à une décomposition SR correspondante. Comme ces transformations élémentaires sont assez hétérogènes, la décomposition SR est considérablement influencée par leur choix.

Dans la littérature, les transformations élémentaires symplectiques impliquées dans la décomposition SR peuvent être divisées en deux sous-ensembles. Le premier sous-ensemble est constitué de deux types de transformations symplectiques et orthogonales à la fois, introduites dans [90, 114] pour calculer une décomposition de Schur ou l'approximation de toutes les valeurs propres d'une matrice Hamiltonienne; et d'une troisième transformation symplectique, mais non orthogonale, a été proposée par Bunse-Gerstner et Mehrmann dans [40]. En effet, Bunse-Gerstner a montrée dans [38] que la décomposition SR d'une matrice générale ne pouvait pas être réalisée en utilisant uniquement les transformations ci-dessus à la fois orthogonales et symplectiques. Pour corriger cette lacune, la troisième transformation symplectique, non orthogonale a été ajoutée. L'algorithme $SRDECO$, basé sur ces trois transformations, a été proposé par Bunse-Gerstner et Mehrmann dans [40], ce qui généralise la méthode de Paige et Van Loan [90]. L'algorithme $SRDECO$ est destiné pour être utilisé dans l'algorithme SR, seulement il est très coûteux.

De point de vue algébrique, la décomposition SR par l'algorithme $SRDECO$ ne correspond pas à l'analogue de la décomposition QR via Householder, parce que l'algorithme $SRDECO$ implique des transformations qui ne sont pas une modification élémentaires de l'identité par une matrice de rang un, voir [5, 60].

Le deuxième sous-ensemble est constitué par des transformations analogiques aux transformations de Householder dans l'espace linéaire symplectique. Ces transformations, qui sont juste une modification de l'identité par une matrice de rang un, sont appelées les transformations de Householder symplectiques. L'algorithme $SRSH$ pour calculer la décomposition SR a été proposé dans [102] en utilisant des transformations de Householder symplectiques. Contrairement à la décomposition QR de Householder, l'algorithme $SRSH$ comporte des paramètres libres et les avantages proviennent de ce fait. Ces paramètres peuvent être déterminés de manière optimale en fournissant une version optimale $SROSH$ de cet algorithme. La décomposition SR par l'algorithme $SRSH$ ou $SROSH$ correspond à l'analogue de la décomposition QR via Householder.

Les aspects computationnels et les comparaisons numériques entre les algorithmes *SGS* et *SROSH* ont clairement montré la supériorité de l'algorithme *SROSH* par rapport à l'algorithme *SGS* ; et aussi que les algorithmes *SROSH* et *SRDECO* se comportent de façon similaire, sauf quand l'algorithme *SRDECO* sera instable numériquement ou même quand il s'arrête sans donner une décomposition *SR*.

Presque toute matrice A peut être décomposée, $A = SR$, en un produit d'une matrice S symplectique et une matrice R J-triangulaire supérieure. Contrairement à la décomposition QR, la décomposition SR n'existe pas toujours. En effet, l'ensemble des matrices, qui peuvent être factorisées de cette manière, est dense dans $\mathbb{R}^{2n\times 2n}$ [38]. Alors que la décomposition QR est généralement considérée pour les matrices dans $\mathbb{R}$ et $\mathbb{C}$, la décomposition SR n'est généralement pas pris en compte pour les matrices complexes $A \in \mathbb{C}^{2n\times 2n}$. Cela est dû au fait que l'ensemble des matrices, qui ont une décomposition SR ($A = SR$), où $S^H JS = J$ ou $S^H JS = -J$, n'est pas dense dans $\mathbb{C}^{2n\times 2n}$ [38].

En outre, soit A une matrice inversible. Il existe une matrice S symplectique et une matrice R J-triangulaire supérieure tel que $A = SR$ si et seulement si tous les mineurs principaux dominants de dimension paire de la matrice $PA^{\mathrm{T}}JAP^{\mathrm{T}}$ sont non nuls.

Si la décomposition SR existe pour une matrice A, alors les autres décompositions SR de la matrice A peuvent être construites à partir de celle-ci en passant des facteurs triviaux, c'est-à-dire des facteurs symplectiques et J-triangulaires à la fois. Autrement dit, si la matrice D est une matrice triviale, $\tilde{S} = SD^{-1}$ et $\tilde{R} = DR$, alors $A = \tilde{S}\tilde{R}$ est une autre décomposition SR de la matrice A. Donc, si la matrice A est inversible, alors c'est la seule façon de créer d'autres décompositions SR. En d'autres termes, la décomposition SR est unique dans ce sens.

2.2 Gram–Schmidt symplectique

Dans le cas classique, soit A une matrice de $\mathbb{R}^{n\times m}$ tel que $\mathrm{rang}(A) = m$. L'algorithme de Gram–Schmidt classique (*GS*) calcule la décomposition :

$$A = QR,$$

où la matrice $Q = [q_1, \ldots, q_m] \in \mathbb{R}^{n\times m}$ a des colonnes orthonormées et la matrice R est une matrice triangulaire supérieure. Ce processus d'orthogonalisation est un élément crucial dans les méthodes d'Arnoldi et de Lanczos [60, 97].

Dans le cas symplectique, son analogue est la décomposition SR à travers l'algorithme de Gram–Schmidt symplectique qui a été introduite par Salam [99]. C'est un processus d'orthogonalisation de type Gram–Schmidt d'un ensemble des vecteurs relatifs au produit scalaire antisymétrique $(.,.)_J$ avec préservation de la structure. Il existe deux versions de l'algorithme de Gram–Schmidt symplectique *SGS*, la version classique *SGS* (parfois notée *CSGS*) et la version modifiée *MSGS*. Dans l'arithmétique exacte, les deux algorithmes sont équivalents, mais ils ont des comportements numériques très différents. Comme dans le cas Euclidien, l'algorithme de Gram–Schmidt symplectique modifiée *MSGS* est plus stable que la version classique.

Soit A une matrice telle que $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$, alors l'algorithme de Gram–Schmidt symplectique calcule la décomposition :

$$A = SR,$$

tel que la matrice $S = [v_1, v_2, \ldots, v_{2p}] \in \mathbb{R}^{2n\times 2p}$ est une matrice symplectique et la matrice $R \in \mathbb{R}^{2n}$ est une matrice J-triangulaire supérieure. En effet, l'algorithme dans sa version colonne consiste à calculer une séquence des matrices :

$$A = A^{(1)}, A^{(2)}, A^{(3)}, \ldots, A^{(p+1)} = S,$$

où la matrice $A^{(j)}$ intermédiaire est sous la forme :

$$A^{(j)} = (v_1, \ldots, v_{j-1}, a_j^{(j)}, \ldots, a_p^{(j)}, v_{p+1} \ldots, v_{p+j-1}, a_{p+j}^{(j)}, \ldots, a_{2p}^{(j)}).$$

Ainsi, les matrices S et R sont générées deux colonnes par deux colonnes. À la $j^{\text{ième}}$ étape de l'algorithme, les deux colonnes $[a_j^{(j)}, a_{p+j}^{(j)}]$ se transforment aux deux colonnes $[v_j, v_{p+j}]$ par la factorisation SR élémentaire ESR

$$[a_j^{(j)}, a_{p+j}^{(j)}] = [v_j, v_{p+j}] \begin{pmatrix} r_{j,j} & r_{j,p+j} \\ 0 & r_{p+j,p+j} \end{pmatrix}.$$

2.2.1 La factorisation SR élémentaire (ESR)

La factorisation SR élémentaire ESR consiste à écrire deux colonne arbitraire $[a_1, a_2]$ comme produit de deux colonnes symplectique $[v_1, v_2]$ et une matrice de taille 2×2 tel que

$$[a_1, a_2] = [v_1, v_2] \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}.$$

Soit $\mathscr{L}$ un espace engendré par les deux vecteurs u_1 et u_2 de $\mathbb{R}^{2n}$ tel que nous notons $\mathscr{L} = \text{vect}\{u_1, u_2\}$. L'espace $\mathscr{L}$ de $\mathbb{R}^{2n}$ est un espace non-isotrope si et seulement si $(u_1, u_2)_J \neq 0$. La matrice $[u_1, u_2] \in \mathbb{R}^{2n\times 2}$ est une matrice symplectique si et seulement si $u_1^{\mathrm{T}} J u_2 = 1$.

Soit la matrice $A_1 = [a_1, a_2] \in \mathbb{R}^{2n\times 2}$, et supposons que l'espace vect(A_1) est non-isotrope. Alors il existe deux matrices : une matrice $V_1 = [v_1, v_2] \in \mathbb{R}^{2n\times 2}$ symplectique et une matrice $R_1 = \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix} \in \mathbb{R}^{2\times 2}$ triangulaire supérieure, telles que

$$A_1 = V_1 R_1. \tag{2.1}$$

Nous référons à cette décomposition (2.1) en tant que la factorisation SR élémentaire de la matrice A_1 et nous la nommons ESR.

Cette factorisation n'est pas unique puisque les coefficients $r_{11} \neq 0$ et r_{12} peuvent être choisis arbitrairement en vérifiant que $r_{11} r_{22} = a_1^{\mathrm{T}} J a_2$.

Pour une matrice $A_1 \in \mathbb{R}^{2n\times 2}$, l'algorithme qui calcule cette factorisation SR élémentaire ESR en satisfaisant l'équation (2.1) est le suivant :

Algorithme 2.1 Algorithme de la factorisation SR élémentaire ESR

Cet algorithme calcule la décomposition SR élémentaire ESR de la matrice $A_1 = [a_1, a_2] \in \mathbb{R}^{2n\times 2}$ tel que $A_1 = V_1 R_1$ avec la matrice $V_1 = [v_1, v_2] \in \mathbb{R}^{2n\times 2}$ est une matrice symplectique et la matrice $R_1 \in \mathbb{R}^{2\times 2}$ est une matrice triangulaire supérieure sous la forme $R_1 = \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}$.

ENTRÉE(S) : Deux vecteurs qui forment la matrice $A_1 = [a_1, a_2] \in \mathbb{R}^{2n\times 2}$.

SORTIE(S) : Deux matrices $V_1 = [v_1, v_2] \in \mathbb{R}^{2n\times 2}$ et $R_1 \in \mathbb{R}^{2\times 2}$ respectivement symplectique et triangulaire supérieure telles que $A_1 = V_1 R_1$.

1: Fonction $[V_1, R_1] = ESR(A_1)$
2: $[den, dep] = size(A_1)$;
3: $n = den/2$;
4: $J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
5: $V_1 = zeros(den, dep)$;
6: $R_1 = zeros(2,2)$;
7: Choisir r_{11} arbitrairement tel que $r_{11} \neq 0$
8: $v_1 = a_1 / r_{11}$;
9: Choisir r_{12} arbitrairement
10: $w = a_2 - r_{12} v_1$;
11: $r_{22} = v_1^{\mathrm{T}} J w$;
12: **Si** $r_{22} \neq 0$ (vect(A_1) non-isotrope) **alors**
13: $v_2 = w / r_{22}$;
14: **Fin Si**
15: **Fin**

Nous notons qu'il y a des différentes versions de la factorisation SR élémentaire ESR proviennent des différents choix des paramètres libres. J. Della-Dora (1975) dans [48], a utilisé la décomposition SR élémentaire qui correspond aux choix $r_{11} = 1$, $r_{12} = 0$. A. Salam (2005) dans [99] a désigné par $ESR1$, la version de la factorisation SR élémentaire ESR, où les paramètres libres $r_{11} = \|a_1\|_2$ et r_{12} reste arbitraire. La deuxième version $ESR2$ est une amélioration de $ESR1$, avec $r_{11} = \|a_1\|_2$ et $r_{12} = v_1^{\mathrm{T}} a_2$. La troisième version $ESR3$, où $r_{11} = \sqrt{|a_1^{\mathrm{T}} J a_2|}$ et r_{12} arbitraire.

2.2.2 Gram–Schmidt symplectique classique (SGS)

Soit la matrice $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$. L'algorithme de Gram–Schmidt symplectique classique calcule la décomposition SR de la matrice A tel que

$$A = SR,$$

avec $S = [s_1, s_2, \ldots, s_{2p}] \in \mathbb{R}^{2n\times 2p}$ est une matrice symplectique et $R \in \mathbb{R}^{2p}$ est une matrice J-triangulaire supérieure.

La version colonne de l'algorithme de Gram–Schmidt symplectique classique est la suivante :

Algorithme 2.2 Algorithme de Gram–Schmidt symplectique classique *SGS* (version colonne)

Cet algorithme calcule la décomposition SR de la matrice $A \in \mathbb{R}^{2n\times 2p}$ tel que $A = SR$ où la matrice S est une matrice symplectique et la matrice R est une matrice J-triangulaire supérieure.

ENTRÉE(S) : La matrice $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$.

SORTIE(S) : Deux matrices $S = [s_1, s_2, \ldots, s_{2p}] \in \mathbb{R}^{2n\times 2p}$ et $R \in \mathbb{R}^{2p\times 2p}$ respectivement symplectique et J-triangulaire supérieure telles que $A = SR$.

Fonction $[S, R] = SGS(A)$
$[den, dep] = size(A)$;
$n = den/2$;
$p = dep/2$;
$J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
$J_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$;
$R = zeros(den, dep)$;
$S = zeros(den, dep)$;
$W_1 = [A(:,1),\ A(:,p+1)]$;
Si $A(:,1)^{\mathrm{T}} \times J \times A(:,p+1) \neq 0$ (W_1 non-isotrope) **alors**
  $[S(:,[1,p+1]),\ R([1,p+1],[1,p+1])] = ESR(A(1:den,[1,p+1]))$;
Sinon
  Arrêter
Fin Si
Pour $j = 2 : p$ **faire**
  Pour $i = 1 : j-1$ **faire**
    $R([i,p+i],[j,p+j]) = J_1^{\mathrm{T}} \times S(:,[i,p+i])^{\mathrm{T}} \times J \times A(:,[j,p+j])$;
  Fin Pour
  $W_j = A(:,[j,p+j]) - \sum_{i=1}^{j-1} S(:,[i,p+i]) \times R([i,p+i],[j,p+j])$;
  Si $W_j(:,1)^{\mathrm{T}} \times J \times W_j(:,2) \neq 0$ (W_j non-isotrope) **alors**
    $[S(:,[j,p+j]),\ R([j,p+j],[j,p+j])] = ESR(W_j)$;
  Sinon
    Arrêter
  Fin Si
Fin Pour
Fin

Notons que *SGS*1 (*SGS*2, *SGS*3, respectivement) désigne la version *SGS* de l'algorithme de Gram–Schmidt symplectique classique dans lequel la factorisation SR élémentaire ESR est remplacée par $ESR1$ ($ESR2$, $ESR3$, respectivement).

2.2.3 Gram–Schmidt symplectique modifié (MSGS)

L'algorithme de Gram–Schmidt classique *SGS* est instable numériquement. Généralement, il y a une perte de la J-orthogonalité comme dans le cas Euclidien. Un réarrange-

ment du calcul, conduit à la version améliorée l'algorithme : Gram–Schmidt symplectique modifié noté par *MSGS*.

La version ligne de l'algorithme de Gram–Schmidt symplectique modifié consiste à calculer une suite des matrices $A = A^{(1)}, A^{(2)}, A^{(3)}, \ldots, A^{(p+1)} = S$, sachant que la matrice $A^{(j)}$ est sous la forme :

$$A^{(j)} = [s_1, \ldots, s_{j-1}, a_j^{(j)}, \ldots, a_p^{(j)}, s_{p+1} \ldots, s_{p+j-1}, a_{p+j}^{(j)}, \ldots, a_{2p}^{(j)}].$$

Les colonnes $a_j^{(j)}, \ldots, a_p^{(j)}, a_{p+j}^{(j)}, \ldots, a_{2p}^{(j)}$ sont déjà J-orthogonales par rapport aux colonnes $s_1, \ldots, s_{j-1}, s_{p+1} \ldots, s_{p+j-1}$. Ces dernières sont des colonnes finales de la matrice S. Dans la $j^{\text{ième}}$ étape, les deux colonnes $[s_j, s_{p+j}]$ sont calculées par la factorisation SR élémentaire ESR tels que

$$[a_j^{(j)}, a_{p+j}^{(j)}] = [s_j, s_{p+j}] \begin{pmatrix} r_{j,j} & r_{j,p+j} \\ 0 & r_{p+j,p+j} \end{pmatrix}.$$

Puis, les colonnes restantes $a_j^{(j)}, \ldots, a_p^{(j)}, a_{p+j}^{(j)}, \ldots, a_{2p}^{(j)}$ sont J-orthogonalisées par rapport à $[s_j, s_{p+j}]$, c'est-à-dire

$$[a_k^{(j+1)}, a_{p+k}^{(j+1)}] = [a_k^{(j)}, a_{p+k}^{(j)}] - [s_j, s_{p+j}] \begin{pmatrix} r_{j,k} & r_{j,p+k} \\ r_{p+j,k} & r_{p+j,p+k} \end{pmatrix},$$

avec

$$\begin{pmatrix} r_{j,k} & r_{j,p+k} \\ r_{p+j,k} & r_{p+j,p+k} \end{pmatrix} = [s_j, s_{p+j}]^J [a_k^{(j)}, a_{p+k}^{(j)}], \quad \text{pour } k = j+1, \ldots, n.$$

Il est intéressant de noter que la matrice $A^{(j)}$ peut être écrite sous la forme

$$A^{(j)} = A^{(j+1)} R_j,$$

avec la matrice R_j possède les mêmes lignes j et $p+j$ comme la matrice J-triangulaire supérieure R, mais elle correspond à la matrice unitaire ailleurs. Ainsi, à la dernière étape, nous obtenons la décomposition

$$A = A^{(1)} = A^{(2)} R_1 = A^{(3)} R_2 R_1 = \ldots = A^{(p+1)} R_p \ldots R_1 = SR,$$

avec la matrice S est une matrice symplectique par construction et la matrice R est une matrice J-triangulaire supérieure.

La version ligne de l'algorithme de Gram–Schmidt symplectique modifié est la suivante :

Algorithme 2.3 Algorithme de Gram–Schmidt symplectique modifié *MSGS* (version ligne)

Cet algorithme calcule la décomposition SR de la matrice $A \in \mathbb{R}^{2n\times 2p}$ tel que $A = SR$ où la matrice S est une matrice symplectique et la matrice R est une matrice J-triangulaire supérieure.

ENTRÉE(S) : La matrice $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$.

SORTIE(S) : Deux matrices $S = [s_1, s_2, \ldots, s_{2p}] \in \mathbb{R}^{2n\times 2p}$ et $R \in \mathbb{R}^{2p\times 2p}$ respectivement symplectique et J-triangulaire supérieure telles que $A = SR$.

Fonction $[S, R] = MSGS(A)$
$[den, dep] = size(A)$;
$n = den/2$;
$p = dep/2$;
$J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
$J_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$;
$R = zeros(den, dep)$;
$S = zeros(den, dep)$;
Pour $j = 1 : n$ **faire**
$\%W_1 = [A(:,1),\ A(:,p+1)]$;
  Si $A(:,j)^{\mathrm{T}} \times J \times A(:,p+j) \neq 0$ (W_j non-isotrope) **alors**
    $[S(:,[j,p+j]),\ R([j,p+j],[j,p+j])] = ESR(A(:,[j,p+j]))$;
  Sinon
    Arrêter
  Fin Si
  Pour $k = j+1 : n$ **faire**
    $R([j,p+j],[k,p+k]) = J_1^{\mathrm{T}} \times S(:,[j,p+j])^{\mathrm{T}} \times J \times A(:,[k,p+k])$;
    $A(:,[k,p+k]) = A(:,[k,p+k]) - S(:,[j,p+j]) \times R([j,p+j],[k,p+k])$;
  Fin Pour
Fin Pour
Fin

La version colonne de l'algorithme de Gram–Schmidt symplectique modifié *MSGS* est obtenue en échangeant l'ordre des calculs dans l'algorithme 2.3 afin que la matrice R soit progressivement obtenu deux colonnes par deux colonnes. Ce qui nous ramène à l'algorithme suivant dans sa version colonne :

Algorithme 2.4 Algorithme de Gram–Schmidt symplectique modifié $MSGS$ (version colonne)

Cet algorithme calcule la décomposition SR de la matrice $A \in \mathbb{R}^{2n\times 2p}$ tel que $A = SR$ où la matrice S est une matrice symplectique et la matrice R est une matrice J-triangulaire supérieure.

ENTRÉE(S) : La matrice $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$.

SORTIE(S) : Deux matrices $S = [s_1, s_2, \ldots, s_{2p}] \in \mathbb{R}^{2n\times 2p}$ et $R \in \mathbb{R}^{2p\times 2p}$ respectivement symplectique et J-triangulaire supérieure telles que $A = SR$.

1: Fonction $[S, R] = MSGS(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
6: $J_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$;
7: $R = zeros(den, dep)$;
8: $S = zeros(den, dep)$;
9: $W_1 = [A(:,1),\ A(:,p+1)]$;
10: **Si** $A(:,1)^{\mathrm{T}} \times J \times A(:,p+1) \neq 0$ (W_1 non-isotrope) **alors**
11: $[S(:,[1,p+1]),\ R([1,p+1],[1,p+1])] = ESR(W_1)$;
12: **Sinon**
13: Arrêter
14: **Fin Si**
15: **Pour** $j = 2 : p$ **faire**
16: **Pour** $i = 1 : j-1$ **faire**
17: $R([i,p+i],[j,p+j]) = J_1^{\mathrm{T}} \times S(:,[i,p+i])^{\mathrm{T}} \times J \times A(:,[j,p+j])$;
18: $W_j = A(:,[j,p+j]) - S(:,[i,p+i]) \times R([i,p+i],[j,p+j])$;
19: **Fin Pour**
20: **Si** $W_j(:,1)^{\mathrm{T}} \times J \times W_j(:,2) \neq 0$ (W_j non-isotrope) **alors**
21: $[S(:,[j,p+j]),\ R([j,p+j],[j,p+j])] = ESR(W_j)$;
22: **Sinon**
23: Arrêter
24: **Fin Si**
25: **Fin Pour**
26: **Fin**

Remarquons que l'algorithme $MSGS1$ ($MSGS2$, $MSGS3$, respectivement) désigne la version $MSGS$ de l'algorithme de Gram–Schmidt symplectique modifié dans lequel la factorisation SR élémentaire ESR est remplacée par $ESR1$ ($ESR2$, $ESR3$, respectivement).

Nous notons que l'algorithme de Gram–Schmidt symplectique modifié $MSGS$ dans ses deux versions colonne ou ligne a le même coût $\mathcal{O}(np^2)$. Les résultats numériques de deux versions sont comparables. Les deux versions produisent les mêmes résultats numériques avec les mêmes erreurs d'arrondis.

La différence entre l'algorithme SGS et l'algorithme $MSGS$ est que dans la version mo-

difiée la projection $[s_i, s_{p+i}]\begin{bmatrix} r_{i,j} & r_{i,p+j} \\ r_{p+i,j} & r_{p+i,p+j} \end{bmatrix}$ est immédiatement soustraite des deux colonnes courantes $[a_j, a_{p+j}]$ une fois le produit scalaire indéfini $[s_i, s_{p+i}]^J[a_j, a_{p+j}]$ est calculées (voir les lignes 16, 17 de l'algorithme 2.3 et les lignes 17, 18 de l'algorithme 2.4). Cependant, comme déjà mentionné précédemment, même si l'algorithme de Gram–Schmidt symplectique classique *SGS* et l'algorithme de Gram–Schmidt symplectique modifié *MSGS* sont mathématiquement équivalents, ils ont des comportements numériques très différents.

2.2.4 Résultats numériques

Dans les exemples ci-dessous, nous allons présenté une comparaison de rapport de la perte de J-orthogonalité et l'erreur absolue de la décomposition SR via l'algorithme de Gram–Schmidt symplectique $MSGS2$ avec $\sqrt{\kappa_2(A^JA)}$ et $\sqrt{\|A^JA\|_2}$ respectivement. En effet,

$$\frac{\left\|I - S^JS\right\|_2}{\kappa_2(A^JA)} \leq \varepsilon_1,$$

et

$$\frac{\|A - SR\|_2}{\sqrt{\|A^JA\|_2}} \leq \varepsilon_2.$$

Ces majorations sont très fines et c'est très difficile pour le moment de démontrer ces résultats théoriquement. En fait, ils méritent plus d'investigations et feront l'objet d'un travail à venir. Les exemples suivants prouvent numériquement ces résultats. Nous avons utilisé l'algorithme $MSGS2$ via la transformation élémentaire $ESR2$.

Exemple 2.2.1. *Dans cet exemple, nous prenons une matrice aléatoire A de taille $2n \times 2n$ telle que $A = rand(2n)$. Dans le tableau 2.1, nous présentons dans les deux premières colonnes la perte de J-orthogonalité et l'erreur absolue de la décomposition SR via l'algorithme $MSGS2$, puis dans les deux dernières colonnes la comparaison avec $\sqrt{\kappa_2(A^JA)}$ et $\sqrt{\|A^JA\|_2}$, respectivement.*

$2n$	$\left\|I - S^JS\right\|_2$	$\|A - SR\|_2$	$\frac{\left\|I-S^JS\right\|_2}{\sqrt{\kappa_2(A^JA)}}$	$\frac{\|A-SR\|_2}{\sqrt{\|A^JA\|_2}}$
10	$2.778227e-14$	$1.141352e-15$	$2.279520e-15$	$5.337267e-16$
12	$4.552833e-15$	$8.859565e-16$	$3.123375e-16$	$2.934378e-16$
14	$3.714892e-14$	$9.550775e-16$	$1.241824e-15$	$3.725937e-16$
16	$9.472045e-14$	$1.951878e-15$	$5.184198e-15$	$6.709442e-16$
18	$3.135370e-13$	$3.182066e-15$	$9.245536e-15$	$9.255411e-16$
20	$1.676059e-13$	$2.872195e-15$	$9.586040e-15$	$9.313757e-16$
22	$2.036772e-13$	$2.773160e-15$	$9.719581e-15$	$7.453637e-16$
24	$4.505086e-13$	$4.325070e-15$	$1.198175e-14$	$9.991685e-16$
26	$4.033913e-12$	$1.268641e-14$	$2.110076e-13$	$3.080962e-15$
28	$7.169826e-12$	$1.829421e-14$	$1.148212e-13$	$3.650348e-15$
30	$1.515253e-11$	$1.303545e-14$	$6.458780e-13$	$2.784771e-15$
32	$1.701451e-12$	$1.028750e-14$	$1.439645e-14$	$1.950676e-15$

$2n$	$\|I - S^J S\|_2$	$\|A - SR\|_2$	$\frac{\|I-S^J S\|_2}{\sqrt{\kappa_2(A^J A)}}$	$\frac{\|A-SR\|_2}{\sqrt{\|A^J A\|_2}}$
34	$1.038683e-12$	$5.842093e-15$	$3.506724e-14$	$1.175783e-15$
36	$3.889220e-11$	$2.321684e-14$	$4.727317e-13$	$3.928278e-15$
38	$1.382190e-11$	$1.700541e-14$	$3.508044e-13$	$2.824444e-15$
40	$2.972035e-11$	$4.190310e-14$	$7.643429e-13$	$6.422285e-15$
42	$6.798994e-12$	$3.616688e-14$	$9.771534e-14$	$6.272007e-15$
44	$1.140465e-10$	$2.522382e-14$	$7.986842e-13$	$3.976627e-15$
46	$2.240403e-11$	$2.435215e-14$	$4.309188e-13$	$4.135919e-15$
48	$8.087628e-11$	$2.677528e-14$	$7.277089e-13$	$3.612570e-15$
50	$2.812247e-12$	$1.861417e-14$	$4.383341e-14$	$2.715005e-15$
52	$2.704368e-11$	$2.726110e-14$	$2.774043e-13$	$3.938440e-15$
54	$5.007757e-11$	$2.930749e-14$	$3.372220e-13$	$3.698948e-15$
56	$1.386341e-11$	$1.849616e-14$	$1.205285e-13$	$2.495525e-15$
58	$4.238597e-10$	$5.009153e-14$	$8.456004e-12$	$6.011019e-15$
60	$1.308539e-10$	$3.621644e-14$	$1.756340e-12$	$4.202345e-15$
62	$3.131928e-10$	$1.061028e-13$	$1.238298e-12$	$1.274508e-14$
64	$1.559199e-10$	$4.848794e-14$	$1.752463e-12$	$5.774761e-15$
66	$1.822717e-10$	$5.655282e-14$	$2.951418e-12$	$6.603007e-15$
68	$1.028708e-10$	$5.528851e-14$	$2.044379e-12$	$5.991214e-15$
70	$6.172274e-10$	$7.893311e-14$	$3.921378e-12$	$8.689253e-15$
72	$4.709477e-10$	$8.874115e-14$	$1.015753e-12$	$9.723950e-15$
74	$1.060537e-10$	$5.595833e-14$	$1.312967e-12$	$5.939292e-15$
76	$4.099725e-08$	$5.379400e-13$	$3.859308e-10$	$5.895765e-14$
78	$3.411352e-09$	$2.544616e-13$	$7.473639e-12$	$2.620247e-14$
80	$1.631161e-09$	$1.634163e-13$	$1.271911e-11$	$1.642790e-14$
82	$5.280397e-11$	$7.087051e-14$	$3.277464e-13$	$6.725807e-15$
84	$8.107981e-09$	$3.570929e-13$	$5.936622e-11$	$3.387143e-14$
86	$1.678155e-10$	$7.541498e-14$	$4.154631e-13$	$7.193127e-15$
88	$8.514081e-10$	$1.612639e-13$	$4.342928e-12$	$1.452360e-14$
90	$6.129730e-10$	$1.427392e-13$	$3.259150e-12$	$1.312587e-14$
92	$7.229107e-10$	$1.348801e-13$	$4.432470e-12$	$1.243933e-14$
94	$7.391016e-10$	$1.519676e-13$	$6.341387e-12$	$1.363400e-14$
96	$1.605787e-09$	$3.248849e-13$	$8.429832e-12$	$2.725135e-14$
98	$6.141053e-10$	$1.971038e-13$	$1.085730e-11$	$1.712551e-14$
100	$5.431989e-09$	$2.360141e-13$	$3.936356e-11$	$1.926131e-14$
102	$5.538027e-10$	$1.360254e-13$	$4.495491e-12$	$1.141446e-14$
104	$4.729025e-10$	$6.018928e-14$	$2.422432e-12$	$4.912927e-15$
106	$2.761664e-10$	$1.099373e-13$	$2.308214e-12$	$8.607192e-15$

TABLEAU 2.1 – La perte de J-orthogonalité et l'erreur absolue de la décomposition SR pour la matrice $A = rand(2n)$ via l'algorithme Gram–Schmidt symplectique modifié $MSGS2$ de l'exemple 2.2.1

Exemple 2.2.2. *Dans cet exemple, nous prenons une matrice A mal conditionnée de taille $2n \times 2n$ telle que $A = Pascal(2n)$. Nous comparons la perte de J-orthogonalité et l'erreur absolue avec $\sqrt{\kappa_2(A^J A)}$ et $\sqrt{\|A^J A\|_2}$, respectivement de la décomposition SR via l'algorithme MSGS2 dans le tableau 2.2.*

$2n$	$\frac{\|I-S^JS\|_2}{\sqrt{\kappa_2(A^JA)}}$	$\frac{\|A-SR\|_2}{\sqrt{\|A^JA\|_2}}$
10	$8.024810e-16$	$1.032346e-15$
12	$9.061303e-17$	$3.571795e-15$
14	$1.933898e-14$	$1.823824e-15$
16	$1.600923e-13$	$8.737769e-15$
18	$6.596920e-12$	$8.732076e-15$
20	$3.456839e-11$	$2.612938e-14$
22	$1.141926e-11$	$2.976639e-14$
24	$4.505905e-12$	$1.616225e-14$
26	$4.855978e-13$	$6.131477e-14$
28	$3.595324e-14$	$1.171254e-13$
30	$2.157750e-14$	$8.033088e-14$
32	$1.366745e-15$	$3.202818e-13$
34	$3.470493e-17$	$2.062405e-13$

(a) Pour $2n$ de 10 jusqu'à 34

$2n$	$\frac{\|I-S^JS\|_2}{\sqrt{\kappa_2(A^JA)}}$	$\frac{\|A-SR\|_2}{\sqrt{\|A^JA\|_2}}$
36	$2.581543e-17$	$7.541128e-13$
38	$3.847825e-16$	$8.748867e-13$
40	$3.920629e-19$	$1.835898e-12$
42	$3.568690e-20$	$4.067951e-12$
44	$3.065806e-20$	$3.673544e-12$
46	$2.954762e-21$	$1.900495e-12$
48	$9.320458e-22$	$1.165014e-11$
50	$2.827825e-23$	$8.411726e-12$
52	$2.168174e-23$	$2.731371e-11$
54	$1.533508e-24$	$3.455703e-11$
56	$1.013009e-25$	$1.458120e-10$
58	$1.369439e-26$	$1.032969e-10$
60	$1.455934e-26$	$1.954161e-10$

(b) Pour $2n$ de 36 Jusqu'à 60

TABLEAU 2.2 – La perte de J-orthogonalité et l'erreur absolue de la décomposition SR pour la matrice $A = Pascal(2n)$ via l'algorithme Gram–Schmidt symplectique modifié $MSGS2$ de l'exemple 2.2.2

2.3 Gram–Schmidt symplectique avec ré-J-orthogonalisation

Après l'étude de la décomposition SR via l'algorithme de Gram–Schmidt classique SGS et modifié $MSGS$ dans le cas symplectique [99, 101, 107] et la décomposition QR via l'algorithme de Gram–Schmidt classique CGS et modifié MGS dans le cas Euclidien [1, 7, 27, 29, 30, 31, 36, 46, 60, 63, 66, 93, 95, 96], nous proposons deux algorithmes de Gram–Schmidt symplectique avec la ré-orthogonalisation symplectique ($RSGS$ et $MRSGS$) en s'inspirant de l'approche de L. Giraud , J. Langou et M. Rozložník (2005) dans [58], introduite dans le cas Euclidien. La ré-orthogonalisation symplectique pourrait être en principe appliquée plusieurs fois, mais la répétition d'une seule étape de la J-orthogonalisation est suffisante pour préserver la J-orthogonalité des vecteurs calculés. Donc, nous considérons juste des algorithmes où l'orthogonalisation symplectique d'un plan courant contre l'ensemble précédemment calculé est effectuée exactement deux fois seulement. Les tests numériques montrent l'efficacité et une meilleure stabilité numérique de la nouvelle méthode avec la ré-J-orthogonalisation.

2.3.1 Gram–Schmidt symplectique classique avec ré-J-orthogonalisation (RSGS)

Soit la matrice $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$. L'algorithme de Gram–Schmidt symplectique classique avec ré-J-orthogonalisation calcule la décomposition SR de la matrice A tel que

$$A = SR,$$

avec la matrice $S = [s_1, s_2, \ldots, s_{2p}] \in \mathbb{R}^{2n\times 2p}$ est une matrice symplectique et la matrice $R \in \mathbb{R}^{2p}$ est une matrice J-triangulaire supérieure.

La version colonne de l'algorithme de Gram–Schmidt symplectique classique avec ré-J-orthogonalisation *RSGS* est la suivante :

Algorithme 2.5 Algorithme *RSGS* de Gram–Schmidt symplectique classique avec ré-J-orthogonalisation (version colonne)

Cet algorithme calcule la décomposition SR de la matrice $A \in \mathbb{R}^{2n\times 2p}$ tel que $A = SR$ où la matrice S est une matrice symplectique et la matrice R est une matrice J-triangulaire supérieure.

ENTRÉE(S) : La matrice $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$.

SORTIE(S) : Deux matrices $S = [s_1, s_2, \ldots, s_{2p}] \in \mathbb{R}^{2n\times 2p}$ et $R \in \mathbb{R}^{2p\times 2p}$ respectivement symplectique et J-triangulaire supérieure telles que $A = SR$.

1: Fonction $[S, R] = RSGS(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
6: $J_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$;
7: $R = zeros(den, dep)$;
8: $S = zeros(den, dep)$;
9: $W_1 = [A(:,1),\ A(:,p+1)]$;
10: **Si** $A(:,1)^{\mathrm{T}} \times J \times A(:,p+1) \neq 0$ (W_1 non-isotrope) **alors**
11: $[S(:,[1,p+1]),\ R([1,p+1],[1,p+1])] = ESR(A(1:den,[1,p+1]))$;
12: **Sinon**
13: Arrêter
14: **Fin Si**
15: **Pour** $j = 2 : p$ **faire**
% La ré-J-orthogonalisation
16: **Pour** $k = 1 : 2$ **faire**
17: **Pour** $i = 1 : j-1$ **faire**
18: $R([i,p+i],[j,p+j]) = J_1^{\mathrm{T}} \times S(:,[i,p+i])^{\mathrm{T}} \times J \times A(:,[j,p+j])$;
19: **Fin Pour**
20: $W_j = A(:,[j,p+j]) - \sum_{i=1}^{j-1} S(:,[i,p+i]) \times R([i,p+i],[j,p+j])$;
21: **Fin Pour**
22: **Si** $W_j(:,1)^{\mathrm{T}} \times J \times W_j(:,2) \neq 0$ (W_j non-isotrope) **alors**
23: $[S(:,[j,p+j]),\ R([j,p+j],[j,p+j])] = ESR(W_j)$;
24: **Sinon**
25: Arrêter
26: **Fin Si**
27: **Fin Pour**
28: **Fin**

Afin de préserver la J-orthogonalité des vecteurs calculés, nous répétons l'étape de

la J-orthogonalisation deux fois seulement. Cette étape de la ré-J-orthogonalisation est entre la ligne 16 et la ligne 21.

Nous notons que l'algorithme $RSGS1$ ($RSGS2$, $RSGS3$, respectivement) désigne la version $RSGS$ de l'algorithme de Gram–Schmidt symplectique classique avec ré-J-orthogonalisation dans lequel la factorisation SR élémentaire ESR est remplacée par $ESR1$ ($ESR2$, $ESR3$, respectivement).

2.3.2 Gram–Schmidt symplectique modifié avec ré-J-orthogonalisation (MRSGS)

La version ligne de l'algorithme de Gram–Schmidt symplectique modifié avec ré-J-orthogonalisation $MRSGS$ est la suivante :

Algorithme 2.6 Algorithme $MRSGS$ de Gram–Schmidt symplectique modifié avec ré-J-orthogonalisation (version ligne)

Cet algorithme calcule la décomposition SR de la matrice $A \in \mathbb{R}^{2n\times 2p}$ tel que $A = SR$ où la matrice S est une matrice symplectique et la matrice R est une matrice J-triangulaire supérieure.

ENTRÉE(S) : La matrice $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$.

SORTIE(S) : Deux matrices $S = [s_1, s_2, \ldots, s_{2p}] \in \mathbb{R}^{2n\times 2p}$ et $R \in \mathbb{R}^{2p\times 2p}$ respectivement symplectique et J-triangulaire supérieure telles que $A = SR$.

1: Fonction $[S, R] = MRSGS(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
6: $J_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$;
7: $R = zeros(den, dep)$;
8: $S = zeros(den, dep)$;
9: **Pour** $j = 1 : n$ **faire**
   $\%W_1 = [A(:, 1),\ A(:, p+1)]$;
10: **Si** $A(:, j)^{\mathrm{T}} \times J \times A(:, p+j) \neq 0$ (W_j non-isotrope) **alors**
11: $[S(:, [j, p+j]),\ R([j, p+j], [j, p+j])] = ESR(A(:, [j, p+j]))$;
12: **Sinon**
13: Arrêter
14: **Fin Si**
15: **Pour** $i = 1 : 2$ **faire**
16: **Pour** $k = j+1 : n$ **faire**
17: $R([j, p+j], [k, p+k]) = J_1^{\mathrm{T}} \times S(:, [j, p+j])^{\mathrm{T}} \times J \times A(:, [k, p+k])$;
18: $A(:, [k, p+k]) = A(:, [k, p+k]) - S(:, [j, p+j]) \times R([j, p+j], [k, p+k])$;
19: **Fin Pour**
20: **Fin Pour**
21: **Fin Pour**
22: **Fin**

Pour préserver la J-orthogonalité des vecteurs calculés, nous répétons l'étape de la J-orthogonalisation deux fois seulement. Cette étape de la ré-J-orthogonalisation est entre la ligne 15 et la ligne 20.

La version colonne de l'algorithme de Gram–Schmidt symplectique modifié avec ré-J-orthogonalisation $MRSGS$ est la suivante :

Algorithme 2.7 Algorithme $MRSGS$ de Gram–Schmidt symplectique modifié avec ré-J-orthogonalisation (version colonne)

Cet algorithme calcule la décomposition SR de la matrice $A \in \mathbb{R}^{2n\times 2p}$ tel que $A = SR$ où la matrice S est une matrice symplectique et la matrice R est une matrice J-triangulaire supérieure.

ENTRÉE(S) : La matrice $A = [a_1, a_2, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$.

SORTIE(S) : Deux matrices $S = [s_1, s_2, \ldots, s_{2p}] \in \mathbb{R}^{2n\times 2p}$ et $R \in \mathbb{R}^{2p\times 2p}$ respectivement symplectique et J-triangulaire supérieure telles que $A = SR$.

1: Fonction $[S, R] = MRSGS(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
6: $J_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$;
7: $R = zeros(den, dep)$;
8: $S = zeros(den, dep)$;
9: $W_1 = [A(:,1),\ A(:,p+1)]$;
10: **Si** $A(:,1)^{\mathrm{T}} \times J \times A(:,p+1) \neq 0$ (W_1 non-isotrope) **alors**
11: $[S(:,[1,p+1]),\ R([1,p+1],[1,p+1])] = ESR(W_1)$;
12: **Sinon**
13: Arrêter
14: **Fin Si**
15: **Pour** $j = 2 : p$ **faire**
16: **Pour** $k = 1 : 2$ **faire**
17: **Pour** $i = 1 : j-1$ **faire**
18: $R([i,p+i],[j,p+j]) = J_1^{\mathrm{T}} \times S(:,[i,p+i])^{\mathrm{T}} \times J \times A(:,[j,p+j])$;
19: $W_j = A(:,[j,p+j]) - S(:,[i,p+i]) \times R([i,p+i],[j,p+j])$;
20: **Fin Pour**
21: **Fin Pour**
22: **Si** $W_j(:,1)^{\mathrm{T}} \times J \times W_j(:,2) \neq 0$ (W_j non-isotrope) **alors**
23: $[S(:,[j,p+j]),\ R([j,p+j],[j,p+j])] = ESR(W_j)$;
24: **Sinon**
25: Arrêter
26: **Fin Si**
27: **Fin Pour**
28: **Fin**

Pour conserver la J-orthogonalité des vecteurs calculés, nous répétons l'étape de la J-

orthogonalisation deux fois seulement. Cette étape de la ré-J-orthogonalisation est entre la ligne 16 et la ligne 21.

Remarquons que l'algorithme $MRSGS1$ ($MRSGS2$, $MRSGS3$, respectivement) désigne la version $MRSGS$ de l'algorithme de Gram–Schmidt symplectique modifié dans lequel la factorisation SR élémentaire ESR est remplacée par $ESR1$ ($ESR2$, $ESR3$, respectivement).

2.3.3 Résultats numériques

Dans cet exemple nous prenons une matrice A de taille $2n \times 2n$ de type Pascal pour comparer la perte de J-orthogonalité et l'erreur absolue de la décomposition SR via les algorithmes SGS, $MSGS$, $RSGS$ et $MRSGS$ pour les transformations élémentaires $ESR1$, $ESR2$ et $ESR3$.

A	$Pascal(6)$	$Pascal(8)$	$Pascal(10)$	$Pascal(12)$	$Pascal(14)$	$Pascal(16)$
$\kappa_2(A)$	$1.1079e+05$	$2.0645e+07$	$4.1552e+09$	$8.7639e+11$	$1.9076e+14$	$4.2469e+16$

TABLEAU 2.3 – Le conditionnement de la matrice $A = Pascal(2n)$

$2n$	$\|I - S^J S\|_2^{SGS1}$	$\|I - S^J S\|_2^{MSGS1}$	$\|I - S^J S\|_2^{RSGS1}$	$\|I - S^J S\|_2^{MRSGS1}$
6	$9.7012e-13$	$7.8104e-13$	$1.1696e-15$	$6.5038e-16$
8	$3.1177e-08$	$4.3199e-11$	$1.5097e-14$	$1.2973e-14$
10	$2.8014e-04$	$6.8979e-09$	$2.4219e-14$	$7.4469e-15$
12	1.9698	$3.5929e-07$	$2.1514e-12$	$1.1392e-13$
14	2.1050	$6.8904e-04$	$2.1567e-11$	$4.9721e-12$
16	53.2962	$6.7000e-03$	$2.5478e-06$	$3.4844e-11$

TABLEAU 2.4 – La perte de J-orthogonalité de la décomposition SR via $SGS1$, $MSGS1$, $RSGS1$ et $MRSGS1$

$2n$	$\|A - SR\|_2^{SGS1}$	$\|A - SR\|_2^{MSGS1}$	$\|A - SR\|_2^{RSGS1}$	$\|A - SR\|_2^{MRSGS1}$
6	$8.9076e-15$	$2.8491e-14$	$2.6041e-14$	$2.6004e-14$
8	$5.6823e-13$	$4.5512e-13$	$4.7735e-13$	$1.1657e-12$
10	$2.2004e-11$	$8.3543e-12$	$2.9968e-12$	$8.1192e-12$
12	$1.6673e-10$	$1.4652e-10$	$1.3494e-10$	$6.9440e-11$
14	$1.8787e-09$	$5.4953e-10$	$6.6667e-10$	$2.4057e-09$
16	$1.7305e-08$	$1.8339e-08$	$4.3498e-08$	$2.1482e-08$

TABLEAU 2.5 – L'erreur absolue de la décomposition SR via $SGS1$, $MSGS1$, $RSGS1$ et $MRSGS1$

$2n$	$\|I - S^J S\|_2^{SGS2}$	$\|I - S^J S\|_2^{MSGS2}$	$\|I - S^J S\|_2^{RSGS2}$	$\|I - S^J S\|_2^{MRSGS2}$
6	$4.1182e-12$	$4.0933e-13$	$3.0033e-16$	$4.5558e-16$
8	$1.9669e-07$	$2.0908e-10$	$3.9456e-16$	$6.5182e-16$
10	$5.1000e-03$	$1.9040e-08$	$9.3301e-16$	$9.9909e-16$
12	4.2601	$1.4009e-06$	$1.5539e-15$	$1.9093e-15$
14	14.8226	$5.1644e-05$	$2.4906e-11$	$4.3186e-15$
16	104.9010	$5.7000e-03$	$1.6216e-07$	$1.2447e-14$

TABLEAU 2.6 – La perte de J-orthogonalité de la décomposition *SR* via *SGS*2, *MSGS*2, *RSGS*2 et *MRSGS*2

$2n$	$\|A - SR\|_2^{SGS2}$	$\|A - SR\|_2^{MSGS2}$	$\|A - SR\|_2^{RSGS2}$	$\|A - SR\|_2^{MRSGS2}$
6	$2.5244e-14$	$3.4552e-14$	$4.5418e-14$	$2.6004e-14$
8	$5.7072e-13$	$4.9291e-13$	$1.1239e-12$	$1.1657e-12$
10	$3.9779e-12$	$4.3537e-12$	$1.1247e-11$	$8.1192e-12$
12	$1.7566e-10$	$1.4058e-10$	$2.7780e-10$	$6.9440e-11$
14	$1.1558e-09$	$6.8283e-10$	$1.1307e-09$	$2.4057e-09$
16	$2.2758e-08$	$3.1535e-08$	$5.8564e-08$	$2.1482e-08$

TABLEAU 2.7 – L'erreur absolue de la décomposition *SR* via *SGS*2, *MSGS*2, *RSGS*2 et *MRSGS*2

$2n$	$\|I - S^J S\|_2^{SGS3}$	$\|I - S^J S\|_2^{MSGS3}$	$\|I - S^J S\|_2^{RSGS3}$	$\|I - S^J S\|_2^{MRSGS3}$
6	$1.5404e-12$	$1.1247e-12$	$2.6191e-15$	$4.0610e-16$
8	$8.2336e-08$	$1.2097e-10$	$1.6127e-14$	$4.5090e-14$
10	$7.9030e-04$	$4.1587e-08$	$5.5011e-14$	$4.4277e-14$
12	$3.7950e-01$	$1.7138e-06$	$7.7078e-13$	$4.4817e-12$
14	9.4204	$1.4000e-03$	$3.9971e-11$	$1.1603e-11$
16	600.9414	$2.6470e-01$	$4.0615e-06$	$9.4430e-11$

TABLEAU 2.8 – La perte de J-orthogonalité de la décomposition *SR* via *SGS*3, *MSGS*3, *RSGS*3 et *MRSGS*3

D'après ces tests numériques sur la perte de J-orthogonalité et l'erreur absolue de la décomposition *SR*, il s'est avéré que l'algorithme de Gram–Schmidt symplectique avec ré-orthogonalisation est plus stable et conserve la J-orthogonalité dans ces deux versions *RSGS* et *MRSGS*. Nous remarquons que l'algorithme *MRSGS* présente un avantage si-

$2n$	$\|A-SR\|_2^{SGS3}$	$\|A-SR\|_2^{MSGS3}$	$\|A-SR\|_2^{SRGS3}$	$\|A-SR\|_2^{MRSGS3}$
6	$1.0211e-14$	$3.3440e-14$	$1.7815e-14$	$3.4173e-14$
8	$5.2193e-13$	$4.7279e-13$	$2.2900e-13$	$2.9461e-13$
10	$8.3275e-12$	$8.4975e-12$	$1.6969e-11$	$1.1531e-11$
12	$1.3505e-10$	$6.5604e-11$	$2.5004e-10$	$1.7605e-10$
14	$1.0953e-09$	$5.6571e-10$	$2.2142e-09$	$3.4233e-09$
16	$2.0750e-08$	$1.5198e-08$	$3.6519e-08$	$4.8390e-08$

TABLEAU 2.9 – L'erreur absolue de la décomposition SR via $SGS3$, $MSGS3$, $RSGS3$ et $MRSGS3$

gnificatif par rapport à l'algorithme $RSGS$ pour la J-orthogonalité. Nous pouvons refaire la ré-J-orthogonalisation plusieurs fois, mais une seule étape de ré-J-orthogonalisation est suffisante pour préserver la J-orthogonalité des vecteurs calculés. Le choix des paramètres libres dans la factorisation SR élémentaire joue un rôle très important. Ainsi, la version $RSGS2$ a un avantage par rapport à $RSGS1$ et $RSGS3$; aussi la version $MRSGS2$ a un avantage par rapport à $MRSGS1$ et $MRSGS3$. La ré-J-orthogonalisation n'intervient pas sur l'erreur absolue de la décomposition SR et les différentes versions de Gram–Schmidt symplectique fournissent des résultats très semblables.

2.3.4 Conclusion

La préservation de la J-orthogonalité est très importante pour les méthodes de conservation de la structure. En fait, les algorithmes de Gram–Schmidt classique ou modifié, dans le cas Euclidien ou symplectique, sont des procédures bien connues et utilisées pour l'orthogonalisation des colonnes d'une matrice donnée. Lorsqu'ils sont appliqués sur des matrices mal conditionnées, alors l'orthogonalité peut être perdue. Dans ce sens, en s'inspirant de l'approche de L. Giraud, J. Langou, et M. Rozložník [58] introduite dans le cas Euclidien, nous avons proposé une technique de ré-J-orthogonalisation pour l'algorithme de Gram–Schmidt symplectique dans ces deux versions (la version classique $SGSi$ et la version modifiée $MSGSi$) afin d'avoir les versions $RSGSi$ et $MRSGSi$. En réalité, La ré-J-orthogonalisation pourrait être en principe appliqué plusieurs fois, mais une seule étape de ré-J-orthogonalisation est suffisante pour préserver la J-orthogonalité des vecteurs calculés. Donc, nous considérons juste des algorithmes où la ré-J-orthogonalisation d'un plan courant contre l'ensemble précédemment calculé est effectuée exactement deux fois seulement. Nous avons montré la pertinence du choix des paramètres libres impliqués dans l'algorithme de Gram–Schmidt symplectique, qui a une incidence sur la préservation de la J-orthogonalité. En effet, il s'est avéré que l'algorithme de Gram–Schmidt symplectique avec ré-J-orthogonalisation, en utilisant la factorisation SR élémentaire $ESR2$, via l'algorithme $RSGS2$ et via la version modifiée $MRSGS2$ sont plus stables et conservent la J-orthogonalité mieux que les versions $RSGS1$, $RSGS3$, $MRSGS1$ et $MRSGS3$. Nous notons que les versions modifiées $MRSGSi$ sont légèrement meilleurs que les versions classiques. Les algorithmes $RSGS1$, $RSGS2$, $RSGS3$, $MRSGS1$, $MRSGS2$, et $MRSGS3$ sont

testés sur une variété d'exemples numériques qui nous permettent de confirmer que la J-orthogonalité est conservée, alors que pour *SGS*1, *SGS*2, *SGS*3, *MSGS*1, *MSGS*2 et *MSGS*3 la J-orthogonalité est souvent complètement perdu. Nous remarquons aussi que la ré-J-orthogonalisation n'intervient pas sur l'erreur absolue de la décomposition *SR* via l'algorithme de Gram–Schmidt symplectique.

2.4 La décomposition SR via les transformations de Householder symplectiques

La décomposition *SR* via les transformations de Householder symplectiques est l'analogue de la décomposition *QR* via les transformations de Householder dans le cas Euclidien. Contrairement du cas Euclidien, dans le cas symplectique il y a des paramètres libres qui intervient à chaque itération. En effet, cette décomposition est basée sur deux types des transformations de Householder symplectiques. Soient $\{e_1, \ldots, e_{2n}\}$ la base canonique de $\mathbb{R}^{2n\times 2n}$, $[a,b] \in \mathbb{R}^{2n\times 2}$ et ρ, μ et ν trois scalaires arbitraires. La décomposition *SR* utilise deux transformations de Householder symplectiques tels que :

$$T_1(a) = \rho e_1,$$
$$T_2(e_1) = e_1$$

et

$$T_2(T_1(b)) = \mu e_1 + \nu e_{n+1}.$$

Le fait que $T_2 T_1$ est une isométrie symplectique nous donne la condition nécessaire suivante :

$$a^J b = (T_2 T_1(a))^J (T_2 T_1(b)) = \rho\nu.$$

Ceci nous conduit au théorème sur lequel est basée la décomposition *SR* via les transformations de Householder symplectiques.

Théorème 2.4.1. *Soient $[a,b] \in \mathbb{R}^{2n\times 2}$ et ρ, μ deux scalaires arbitraires satisfaisant*

$$\rho\nu = a^J b$$

et soient

$$c_1 = -\frac{1}{a^J \rho e_1}, \qquad v_1 = \rho e_1 - a,$$
$$c_2 = -\frac{1}{(T_1(b))^J(\mu e_1 + \nu e_{n+1})}, \qquad v_2 = \mu e_1 + \nu e_{n+1} - T_1(b).$$

Alors, nous avons les transformations de Householder symplectiques

$$T_1 = I + c_1 v_1 v_1^J, \tag{2.2}$$
$$T_2 = I + c_2 v_2 v_2^J, \tag{2.3}$$

vérifient que

$$T_1(a) = \rho e_1,$$
$$T_2(e_1) = e_1,$$
$$T_2(T_1(b)) = \mu e_1 + \nu e_{n+1}.$$

2.4.1 L'algorithme SRSH

Nous décrivons ici les étapes de l'algorithme *SRSH* pour effectuer la décomposition *SR* via les transformations de Householder symplectiques. Cette décomposition est souvent utilisée pour les matrices carrées. Toutefois, pour certaines applications, il est nécessaire de faire la décomposition pour les matrices non carrées. Nous présentons ici la version générale.

Soit la matrice de départ $A = [a_1, ..., a_p, a_{p+1}, ..., a_{2p}] \in \mathbb{R}^{2n\times 2p}$. Soient ρ, μ et ν des scalaires arbitraires satisfaisant

$$\rho\nu = a_1^J a_{p+1}.$$

La première étape :

La première étape de l'algorithme *SRSH* est de trouver la transformation de Householder symplectique T_1 pour annuler les entrées de la première colonne de la position 2 jusqu'à la position $2n$, c'est-à-dire trouver la constante c_1 et le vecteur v_1 par le Théorème 2.4.1, tel que

$$T_1(a_1) = \rho e_1 = A^{(1)}_{11} e_1.$$

L'action de T_1 sur A est

$$A^{(1)} = T_1 A = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:p)} & A^{(1)}_{(1,p+1)} & A^{(1)}_{(1,p+2:2p)} \\ 0 & A^{(1)}_{(2:n,2:p)} & A^{(1)}_{(2:n,p+1)} & A^{(1)}_{(2:n,p+2:2p)} \\ 0 & A^{(1)}_{(n+1,2:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2:2p)} \\ 0 & A^{(1)}_{(n+2:2n,2:p)} & A^{(1)}_{(n+2:2n,p+1)} & A^{(1)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

□

La deuxième étape :

Ensuite, la deuxième étape consiste à trouver la transformation de Householder symplectique T_2 pour mettre des zéros dans la $(p+1)^{\text{ième}}$ colonne de la matrice $A^{(1)}$ de la position 2 jusqu'à la position n et de la position $n+2$ jusqu'à la position $2n$, c'est-à-dire trouver la constante c_2 et le vecteur v_2 par le Théorème 2.4.1, tel que

$$T_2(e_1) = e_1$$

et

$$T_2 T_1(a_{p+1}) = A^{(2)}_{(1,p+1)} e_1 + A^{(1)}_{(n+1,p+1)} e_{n+1}.$$

L'action de T_2 sur $T_1 A$ est

$$A^{(2)} = T_2 T_1 A = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2:p)} & A^{(2)}_{(1,p+1)} & A^{(2)}_{(1,p+2:2p)} \\ 0 & A^{(2)}_{(2:n,2:p)} & 0 & A^{(2)}_{(2:n,p+2:2p)} \\ 0 & A^{(1)}_{(n+1,2:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2:2p)} \\ 0 & A^{(2)}_{(n+2:2n,2:p)} & 0 & A^{(2)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

Puisque la composante $(n+1)^{\text{ième}}$ de l'entrée du vecteur v_2 est nulle ($v_2(n+1)=0$), alors pour n'importe quel vecteur $x \in \mathbb{R}^{2n}$ la transformation T_2 ne modifie pas la composante $(n+1)^{\text{ième}}$ du vecteur $T_2(x)$.

Ces deux étapes impliquent deux paramètres libres $\rho = A^{(1)}_{(1,1)}$ et $\mu = A^{(1)}_{(1,p+1)}$.

□

La troisième étape :

Dans cette étape, nous voulons annuler les entrées de la position 3 jusqu'à la position n et de la position $n+2$ jusqu'à la position $2n$ de la deuxième colonne de la matrice $A^{(2)}$.

Soit la sous matrice $\tilde{A}^{(2)}$ obtenue en supprimant la première ligne et la $(n+1)^{\text{ième}}$ ligne et la première colonne et la $(p+1)^{\text{ième}}$ colonne de la matrice $A^{(2)}$, c'est-à-dire

$$\tilde{A}^{(2)} = \begin{bmatrix} A^{(2)}_{(2:n,2:p)} & A^{(2)}_{(2:n,p+2:2p)} \\ A^{(2)}_{(n+2:2n,2:p)} & A^{(2)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

Cette étape consiste à répéter la première étape sur la nouvelle matrice réduite $\tilde{A}^{(2)}$. En d'autres termes, nous choisissons une transformation de Householder symplectique $\tilde{T}_3$, ce qui signifie que nous devons calculer un scalaire réel $c_3 \in \mathbb{R}$ et un vecteur

$$\tilde{v}_3 = \begin{bmatrix} u_3 \\ \hline w_3 \end{bmatrix} \in \mathbb{R}^{2n-2},$$

avec $u_3 \in R^{n-1}$, $w_3 \in R^{n-1}$ tel que $\tilde{T}_3 = I_{2n-2} + c_3 \tilde{v}_3 \tilde{v}_3^J$. Cette transformation annule les entrées de la position 2 jusqu'à la position $n-1$ et de la position n jusqu'à la position $2n-2$ de la première colonne de la matrice $\tilde{A}^{(2)}$, sachant que

$$\tilde{T}_3 \tilde{A}^2(:,1) = A^{(3)}_{(2,2)} e_1 \in \mathbb{R}^{2n-2},$$

avec $A^{(3)}_{(2,2)}$ est un scalaire arbitraire non nul. En effet, la transformation $\tilde{T}_3$ correspond à la transformation de Householder symplectique T_1 dans le Théorème 2.4.1.

Par conséquent, nous obtenons

$$\tilde{A}^{(3)} = \tilde{T}_3 \tilde{A}^{(2)} = \begin{bmatrix} A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:p)} & A^{(3)}_{(2,p+2)} & A^{(3)}_{(2,p+3:2p)} \\ 0 & A^{(3)}_{(3:n,3:p)} & A^{(3)}_{(3:n,p+2)} & A^{(3)}_{(3:n,p+3:2p)} \\ 0 & A^{(3)}_{(n+2,3:p)} & A^{(3)}_{(n+2,p+2)} & A^{(3)}_{(n+2,p+3:2p)} \\ 0 & A^{(3)}_{(n+3:2n,3:p)} & A^{(3)}_{(n+3:2n,p+2)} & A^{(3)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

Soit $T_3 = I_{2n} + c_3 v_3 v_3^J$ la transformation de Householder symplectique où le vecteur de direction

$$v_3 = \begin{bmatrix} 0 \\ \hline u_3 \\ \hline 0 \\ \hline w_3 \end{bmatrix} \in \mathbb{R}^{2n},$$

tel que sur ces entrefaites, nous avons :

$$A^{(3)} = T_3 A^{(2)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(2)}_{(1,3:p)} & A^{(2)}_{(1,p+1)} & A^{(2)}_{(1,p+2)} & A^{(2)}_{(1,p+3:2p)} \\ 0 & A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:p)} & 0 & A^{(3)}_{(2,p+2)} & A^{(3)}_{(2,p+3:2p)} \\ 0 & 0 & A^{(3)}_{(3:n,3:p)} & 0 & A^{(3)}_{(3:n,p+2)} & A^{(3)}_{(3:n,p+3:2p)} \\ 0 & A^{(1)}_{(n+1,2)} & A^{(1)}_{(n+1,3:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2)} & A^{(1)}_{(n+1,p+3:2p)} \\ 0 & 0 & A^{(3)}_{(n+2,3:p)} & 0 & A^{(3)}_{(n+2,p+2)} & A^{(3)}_{(n+2,p+3:2p)} \\ 0 & 0 & A^{(3)}_{(n+3:2n,3:p)} & 0 & A^{(3)}_{(n+3:2n,p+2)} & A^{(3)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

La transformation T_3 préserve la première et la $(n+1)^{\text{ième}}$ lignes et la première et la $(p+1)^{\text{ième}}$ colonnes de la matrice $A^{(2)}$.

□

La quatrième étape :

Maintenant, nous répétons la deuxième étape à la matrice $\tilde{A}^{(3)} = \tilde{T}_3 \tilde{A}^{(2)}$ pour mettre des zéros dans les entrées de la position 3 jusqu'à la position n et de la position $n+3$ jusqu'à la position $2n$ de la $(p+2)^{\text{ième}}$ colonne de la matrice $A^{(3)}$. En effet, nous cherchons la transformation de Householder symplectique $\tilde{T}_4$ pour annuler les entrées de la $p^{\text{ième}}$ colonne de la matrice $\tilde{A}^{(3)}$ de la position 2 jusqu'à la position $n-2$ et de la positon $n+2$ jusqu'à la position $2n-2$, c'est-à-dire cherchons à trouver la constante c_4 et le vecteur

$$\tilde{v}_4 = \begin{bmatrix} u_4 \\ \hline w_4 \end{bmatrix} \in \mathbb{R}^{2n-2},$$

avec $u_4 \in R^{n-1}$, $w_4 \in R^{n-1}$ tel que $\tilde{T}_4 = I_{2n-2} + c_4 \tilde{v}_4 \tilde{v}_4^J$ (la transformation $\tilde{T}_4$ correspond à la transformation T_2 dans le Théorème 2.4.1), tel que

$$\tilde{T}_4(e_1) = e_1$$

et

$$\tilde{T}_4 \tilde{T}_3(\tilde{A}^{(2)}_{(:,p+2)}) = A^{(4)}_{(2,p+2)} e_1 + A^{(3)}_{(n+2,p+2)} e_{n+1},$$

avec

$$\tilde{A}^{(2)}(1,:)^J \tilde{A}^{(2)}(p,:) = A^{(3)}_{(2,2)} A^{(3)}_{(n+2,p+2)}.$$

Ainsi, nous avons

$$\tilde{A}^{(4)} = \tilde{T}_4 \tilde{A}^{(3)} = \begin{bmatrix} A^{(3)}_{(2,2)} & A^{(4)}_{(2,3:p)} & A^{(4)}_{(2,p+2)} & A^{(4)}_{(2,p+3:2p)} \\ 0 & A^{(4)}_{(3:n,3:p)} & 0 & A^{(4)}_{(3:n,p+3:2p)} \\ 0 & A^{(3)}_{(n+2,3:p)} & A^{(3)}_{(n+2,p+2)} & A^{(3)}_{(n+2,p+3:2p)} \\ 0 & A^{(4)}_{(n+3:2n,3:p)} & 0 & A^{(4)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

De même, soit la transformation de Householder symplectique $T_4 = I_{2n} + c_4 v_4 v_4^J$ avec

$$v_4 = \begin{bmatrix} 0 \\ u_4 \\ \hline 0 \\ w_4 \end{bmatrix} \in \mathbb{R}^{2n}.$$

L'action de T_4 sur $A^{(3)}$ est

$$A^{(4)} = T_4 A^{(3)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(2)}_{(1,3:p)} & A^{(2)}_{(1,p+1)} & A^{(2)}_{(1,p+2)} & A^{(2)}_{(1,p+3:2p)} \\ 0 & A^{(3)}_{(2,2)} & A^{(4)}_{(2,3:p)} & 0 & A^{(4)}_{(2,p+2)} & A^{(4)}_{(2,p+3:2p)} \\ 0 & 0 & A^{(4)}_{(3:n,3:p)} & 0 & 0 & A^{(4)}_{(3:n,p+3:2p)} \\ 0 & A^{(1)}_{(n+1,2)} & A^{(1)}_{(n+1,3:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2)} & A^{(1)}_{(n+1,p+3:2p)} \\ 0 & 0 & A^{(3)}_{(n+2,3:p)} & 0 & A^{(3)}_{(n+2,p+2)} & A^{(3)}_{(n+2,p+3:2p)} \\ 0 & 0 & A^{(4)}_{(n+3:2n,3:p)} & 0 & 0 & A^{(4)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

La transformation T_4 ne modifie pas la première ligne, la $(n+1)^{\text{ième}}$ ligne et la $(n+2)^{\text{ième}}$ ligne et la première colonne, la deuxième colonne et la $(p+1)^{\text{ième}}$ colonne de la matrice $A^{(3)}$.

La troisième et la quatrième étapes de l'algorithme impliquent deux paramètres libres $A^{(3)}_{(2,2)}$ et $A^{(3)}_{(n+2,p+2)}$ qui correspondent aux paramètres libres ρ et μ du Théorème 2.4.1, respectivement.

□

Ainsi dans la suite, à chaque fois nous répétons la première et la deuxième étapes jusqu'à la dernière étape c'est-à-dire la $2p^{\text{ième}}$ étape.

À la fin, nous obtenons

$$T_{2p} T_{2p-1} \dots T_4 T_3 T_2 T_1 A = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} = R \in \mathbb{R}^{2n \times 2p},$$

avec les blocs R_{11}, R_{12}, R_{22} sont des matrices triangulaires supérieurs et le bloc R_{21} est une matrice strictement triangulaire supérieure. La matrice R est appelée J-triangulaire supérieure. Nous avons

$$T_{2p} T_{2p-1} \dots T_4 T_3 T_2 T_1 A = \begin{bmatrix} \diagdown & \diagdown \\ 0 \diagdown & \diagdown \\ \quad 0 & \end{bmatrix} = R.$$

Par suite, nous avons $A = SR$ avec la matrice $S = T_1^J T_2^J \dots T_{2p-1}^J T_{2p}^J$ est une matrice symplectique et la matrice R est une matrice J-triangulaire.

Nous proposons ici l'algorithme dans sa version générale, pour calculer la décomposition SR d'une matrice A, via des transformations de Householder symplectiques. C'est algorithme de $SRSH$:

Algorithme 2.8 Algorithme *SRSH*

Cet algorithme calcule la décomposition SR d'une matrice A tel que $A = SR$ avec la matrice $S = T_1^J T_2^J \dots T_{2p-1}^J T_{2p}^J$ est une matrice symplectique. La matrice A est remplacée par la matrice J-triangulaire R.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2p}$ et $S = I_{2n}$.

SORTIE(S) : R la matrice sous la forme J-triangulaire et $S = T_1^J T_2^J \dots T_{2p-1}^J T_{2p}^J$ la matrice symplectique.

1: Fonction $[S,R] = SRSH(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $S = eye(den)$;
6: **Pour** $j = 1 : p$ **faire**
7: $J = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
8: $ro = [j : n, n+j : 2n]$;
9: $co = [j : p, p+j : 2p]$;
% Calculer la constant c et le vecteur v_1 de la transformation de Householder symplectique de type un tel que $T_1 = I + c v_1 v_1^J$.
10: $[c, v_1] = sh1(A(ro, j))$;
% Mise à jour de la matrice A.
11: $A(ro, co) = A(ro, co) + c \times v_1 \times (v_1^{\mathrm{T}} \times J \times A(ro, co))$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
12: $S(:, co) = S(:, co) - (S(:, co) \times (c \times v_1)) \times v_1^{\mathrm{T}} \times J$;
% Mise à jour de la matrice S^J implicitement (si nous avons besoin).
%$S(ro, 2 : end) = S(ro, 2 : end) + c \times (v_1 \times v_1^{\mathrm{T}}) \times J \times S(ro, 2 : end)$;
% Calculer la constant c et le vecteur v_2 de la transformation de Householder symplectique de type deux tel que $T_2 = I + c v_2 v_2^J$.
13: $[c, v2] = sh2(A(ro, p+j))$;
% Mise à jour de la matrice A.
14: $A(ro, co) = A(ro, co) + c \times v_2 \times (v_2^{\mathrm{T}} \times J \times A(ro, co))$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
15: $S(:, co) = S(:, co) - (S(:, co) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;
% Mise à jour de la matrice S^J implicitement (si nous avons besoin).
%$S(ro, 2 : end) = S(ro, 2 : end) + c \times (v_2 \times v_2^{\mathrm{T}}) \times J \times S(ro, 2 : end)$;
16: **Fin Pour**
17: $R = A$
18: **Fin**

Dans cet algorithme *SRSH* de décomposition *SR*, nous calculons les transformations de Householder symplectiques de type un $sh1$ et de type deux $sh2$ via les algorithmes suivants.

L'algorithme de la transformation de Householder symplectique de type un est donné par :

Algorithme 2.9 Algorithme de la transformation de Householder symplectique $sh1$

Cet algorithme calcule la constante $c_1 \in \mathbb{R}$ et un vecteur directeur $v_1 \in \mathbb{R}^{2n}$ tel que la transformation de Householder symplectique de type un soit sous la forme $T_1 = I_{2n} + c_1 v_1 v_1^J$ en vérifiant que $T_1(a) = \rho e_1$. ρ est un paramètre libre.

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$.

SORTIE(S) : Une constante $c_1 \in \mathbb{R}$ et un vecteur $v_1 \in \mathbb{R}^{2n}$.

Fonction $[c_1, v_1] = sh1(a)$
$den =$ taille (a);
$n = den/2$;
$J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
Si $a_{n+1} == 0$ **alors**
  Stop division par zéro
Sinon
  Choisir le paramètre ρ
  $v_1 = a$;
  $v_1(1) = a(1) - \rho$
  $c_1 = \dfrac{1}{\rho a_{n+1}}$;
  $\%T_1 = I + c_1 v_1 v_1^J$;
Fin Si
Fin

Pareillement, l'algorithme de la transformation de Householder symplectique de type deux est comme suit :

Algorithme 2.10 Algorithme de la transformation de Householder symplectique $sh2$

Cet algorithme calcule la constante $c_2 \in \mathbb{R}$ et un vecteur $v_2 \in \mathbb{R}^{2n}$ tel que la transformation de Householder symplectique soit sous la forme $T_2 = I_{2n} + c_2 v_2 v_2^J$ en vérifiant que $T_2(e_1) = e_1$ et $T_2(u) = \mu e_1 + \nu e_{n+1}$. μ est un paramètre libre.

ENTRÉE(S) : Un vecteur $u \in \mathbb{R}^{2n}$

SORTIE(S) : Une constante $c_2 \in \mathbb{R}$ et un vecteur $v_2 \in \mathbb{R}^{2n}$

Fonction $[c_2, v_2] = sh2(u)$
$den =$ taille (u);
$n = den/2$;
$J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
Si $n == 1$ **alors**
  $c_2 = 0$;
  $v_2 = 0_{2n}$;
  $\%T_2 = I_{2n}$;
Sinon
  Choisir le paramètre ξ non nul
  $\nu = u(n+1)$;
  $\mu = u(1) + \xi$
  $aux = \nu \times \xi$
  Si $aux == 0$ **alors**
    Stop division par zéro
  Sinon
    $v_2 = u$;
    $v_2(1) = -\xi$;
    $v_2(n+1) = 0$;
    $c = \dfrac{1}{u(n+1)(u(1)-\mu)}$;
  $\%T_2 = I + c_2 v_2 v_2^J$;
  Fin Si
Fin Si
Fin

Il est important de noter que, contrairement à la décomposition QR via les transformations de Householder, l'algorithme $SRSH$ implique des paramètres libres. En outre, à chaque itération j de l'algorithme $SRSH$, deux des trois paramètres peuvent être choisis librement.

2.4.2 L'algorithme SROSH

Du point de vue algébrique, l'algorithme $SRSH$ est l'analogue, dans le cas Euclidien, à l'algorithme QR via les transformations de Householder. Contrairement au cas Euclidien, l'algorithme $SRSH$ comporte deux paramètres libres à chaque étape, et les transformations de Householder symplectiques impliquées ne sont pas orthogonales. Afin d'obtenir un algorithme plus stable numériquement que possible, les paramètres libres seront choisis de sorte que les transformations de Householder symplectiques utilisées dans

la décomposition *SR* aient des conditionnements minimaux selon la norme $\|.\|_2$ pour prendre l'avantage de ces paramètres libres d'une manière optimale. En d'autres termes, l'algorithme *SROSH* correspond à un choix efficace et optimal des paramètres libres utilisés dans l'algorithme *SRSH*. De ce fait, nous devons minimiser le conditionnement des transformations de Householder symplectiques afin de les optimiser. Pour plus de détails voir les paragraphes 1.5 et 1.5.1 dans le premier chapitre.

L'algorithme *SROSH* est basé sur le résultat suivant :

Théorème 2.4.2. *Soit* $[a,b] \in \mathbb{R}^{2n\times 2}$. *Soient*

$$\rho = sign(a_1)\|a\|_2, \qquad c_1 = -\frac{1}{\rho a^J e_1}, \qquad v_1 = \rho e_1 - a,$$

tel que

$$T_1 = I + c_1 v_1 v_1^J.$$

Alors, T_1 *a un conditionnement minimal selon la norme* $\|.\|_2$ *et vérifie que*

$$T_1(a) = \rho e_1. \tag{2.4}$$

Soit u *un vecteur tel que* $u = T_1(b)$ *et* u_i *sa* $i^{ième}$ *composante. Soient*

$$\nu = u_{n+1}, \qquad \xi = \|u - u_1 e_1 - u_{n+1} e_{n+1}\|_2, \qquad \mu = u_1 \pm \xi,$$

$$c_2 = -\frac{1}{\pm \xi u_{n+1}}, \qquad v_2 = \mu e_1 + u_{n+1} e_{n+1} - u,$$

tel que

$$T_2 = I + c_2 v_2 v_2^J.$$

Alors, T_2 *a un conditionnement minimal selon la norme* $\|.\|_2$ *et vérifie que*

$$T_2(e_1) = e_1, \tag{2.5}$$

$$T_2(u) = \mu e_1 + \nu e_{n+1}. \tag{2.6}$$

Les transformations T_1 *et* T_2 *sont deux transformations de Householder symplectiques optimales.*

Remarque 2.4.1. *Le vecteur* v_2 *se distingue du vecteur* u *uniquement par la première composante et la* $(n+1)^{ième}$ *composante et satisfait* $v_2(n+1) = 0$. *Cela va être pris en considération lors du stockage de vecteur* v_2.

La fonction $osh1$ à partir d'un vecteur a calcule le coefficient c_1 et le vecteur v_1 de la transformation de Householder symplectique optimale T_1. De même, la fonction $osh2$ à partir d'un vecteur u calcule le coefficient c_2 et le vecteur v_2 de la transformation de Householder symplectique optimale T_2.

La normalisation du vecteur v_1 de la transformation Householder symplectique optimal de la fonction $osh1$ de sorte que $v_1(1) = 1$, permet de stocker $v_1(2:2n)$ où les zéros ont été introduits dans le vecteur a, c'est-à-dire dans $a(2:2n)$ (voir la version de $osh1$ avec normalisation dans le chapitre suivant).

L'algorithme qui calcule la transformation de Householder symplectique optimale de type un est le suivant :

Algorithme 2.11 Algorithme de la transformation de Householder symplectique optimale $osh1$

Cet algorithme calcule la constante $c_1 \in \mathbb{R}$ et un vecteur $v_1 \in \mathbb{R}^{2n}$ tel que $v_1(1) = 1$ et la transformation de Householder symplectique optimale soit sous la forme $T_1 = I + c_1 v_1 v_1^J$ en vérifiant que $T_1(a) = \rho e_1$.

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$.

SORTIE(S) : Une constante $c_1 \in \mathbb{R}$ et un vecteur $v_1 \in \mathbb{R}^{2n}$.

1: Fonction $[c_1, v_1] = osh1(a)$
2: $den =$ taille (a);
3: $n = den/2$;
4: $J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
5: **Si** $a_{n+1} == 0$ **alors**
6: Stop division par zéro
7: **Sinon**
8: $\rho = sign(a_1)\|a\|_2$;
9: $v_1 = a$;
10: $v_1(1) = a(1) - \rho$
11: $c_1 = \dfrac{1}{\rho a_{n+1}}$;
$\%T_1 = I + c_1 v_1 v_1^J$;
12: **Fin Si**
13: **Fin**

D'une manière similaire, nous pouvons normaliser le vecteur v_2 de la transformation Householder symplectique optimale de la fonction $osh2$ de sorte que $v_2(1) = 1$ (voir la version de $osh2$ avec normalisation dans le chapitre suivant). De plus, puisque $v_2(n+1) = 0$ alors $v_2(2:2n)$ peut être stocké où les zéros ont été introduits dans le vecteur u. Ce stockage n'est pas possible si la transformation de Householder symplectique utilisée n'est pas optimale puisque dans ce cas, $v_2(n+1)$ n'est pas nécessairement nul.

Ainsi, l'algorithme qui calcule la transformation de Householder symplectique optimale de type deux est le suivant :

Algorithme 2.12 Algorithme de la transformation de Householder symplectique optimale $osh2$

Cet algorithme calcule la constante $c_2 \in \mathbb{R}$ et un vecteur $v_2 \in \mathbb{R}^{2n}$ tel que l'entrée $(n+1)^{\text{ième}}$ du vecteur v_2 est nulle ($v_2(n+1) = 0$), et la transformation de Householder symplectique optimale soit sous la forme $T_2 = I + c_2 v_2 v_2^J$ en vérifiant que $T_2(e_1) = e_1$ et $T_2(u) = \mu e_1 + \nu e_{n+1}$.

ENTRÉE(S) : Un vecteur $u \in \mathbb{R}^{2n}$.

SORTIE(S) : Une constante $c_2 \in \mathbb{R}$ et un vecteur $v_2 \in \mathbb{R}^{2n}$.

Fonction $[c_2, v_2] = osh2(u)$
$den =$ taille (u);
$n = den/2$;
$\%J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
Si $n == 1$ **alors**
  $c_2 = 0$;
  $v_2 = 0_{2n}$;
  $\%T_2 = I_{2n}$;
Sinon
  $I = [2 : n, n+2 : den]$;
  $\xi = \|u(I)\|_2$;
  Si $\xi == 0$ **alors**
    $c_2 = 0$;
    $v_2 = 0_{2n}$;
    $\%T_2 = I_{2n}$;
  Sinon
    $\nu = u(n+1)$;
    $\%\mu = u(1) + \xi$; pas besoin de calculer μ
    Si $\nu == 0$ **alors**
      Stop division par zéro
    Sinon
      $v_2 = -\dfrac{u}{\xi}$;
      $v_2(1) = 1$;
      $v_2(n+1) = 0$;
      $c = \dfrac{\xi}{u(n+1)}$;
      $\%T_2 = I + c_2 v_2 v_2^J$;
    Fin Si
  Fin Si
Fin Si
Fin

Remarque 2.4.2. *Dans l'algorithme précédent de* $osh2$ *nous pouvons prendre la variable* $\xi = -\|u(I)\|_2$ *au lieu de* $\xi = \|u(I)\|_2$.

Ce qui nous ramène à proposer l'algorithme dans sa version générale, pour calculer la décomposition SR d'une matrice A, via des transformations de Householder symplec-

tiques optimales. L'algorithme de $SROSH$ est alors comme suit :

Algorithme 2.13 Algorithme $SROSH$

Cet algorithme calcule la décomposition SR une matrice A tel que $A = SR$ avec la matrice $S = T_1^J T_2^J \dots T_{2p-1}^J T_{2p}^J$ est une matrice symplectique. La matrice A est remplacées par la matrice J-triangulaire R.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2p}$ et $S = I_{2n}$.

SORTIE(S) : R la matrice sous la forme J-triangulaire et $S = T_1^J T_2^J \dots T_{2p-1}^J T_{2p}^J$ la matrice symplectique.

1: Fonction $[S,R] = SROSH(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $S = eye(den)$;
6: **Pour** $j = 1 : p$ **faire**
7: $J = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
8: $ro = [j : n, n+j : 2n]$;
9: $co = [j : p, p+j : 2p]$;
% Calculer la constant c et le vecteur v_1 de la transformation de Householder symplectique optimale de type un tel que $T_1 = I + c v_1 v_1^J$.
10: $[c, v_1] = osh1(A(ro, j))$;
% Mise à jour de la matrice A.
11: $A(ro, co) = A(ro, co) + c \times v_1 \times (v_1^{\mathrm{T}} \times J \times A(ro, co))$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
12: $S(:, co) = S(:, co) - (S(:, co) \times (c \times v_1)) \times v_1^{\mathrm{T}} \times J$;
% Mise à jour de la matrice S^J implicitement (si nous avons besoin).
%$S(ro, 2 : end) = S(ro, 2 : end) + c \times (v_1 \times v_1^{\mathrm{T}}) \times J \times S(ro, 2 : end)$;
% Calculer la constant c et le vecteur v_2 de la transformation de Householder symplectique optimale de type deux tel que $T_2 = I + c v_2 v_2^J$.
13: $[c, v2] = osh2(A(ro, p+j))$;
% Mise à jour de la matrice A.
14: $A(ro, co) = A(ro, co) + c \times v_2 \times (v_2^{\mathrm{T}} \times J \times A(ro, co))$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
15: $S(:, co) = S(:, co) - (S(:, co) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;
% Mise à jour de la matrice S^J implicitement (si nous avons besoin).
%$S(ro, 2 : end) = S(ro, 2 : end) + c \times (v_2 \times v_2^{\mathrm{T}}) \times J \times S(ro, 2 : end)$;
16: **Fin Pour**
17: $R = A$
18: **Fin**

2.5 La décomposition SR via l'algorithme SRMSH

Dans ce qui suit, nous allons présenter un nouvel algorithme $SRMSH$ qui calcule la décomposition SR. En effet, nous allons modifier l'algorithme $SROSH$ en remplaçons la

moitié des transformations de Householder symplectiques, non forcément orthogonales, par d'autres transformations élémentaires, qui ont l'avantage d'être symplectiques et orthogonales. Autrement dit, nous remplaçons les transformations de Householder symplectiques optimales de type un T_1 par des transformations de Givens symplectiques et des transformations de Householder symplectiques au sens de Van Loan [114].

En fait, nous utilisons deux types des transformations pour remplacer les transformations T_1.

Le premier type est

$$H(k,w) = \begin{pmatrix} \mathrm{diag}(I_{k-1,}, P) & 0 \\ 0 & \mathrm{diag}(I_{k-1}, P) \end{pmatrix},$$

avec

$$P = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w,\ w \in \mathbb{R}^{n-k+1}.$$

Cette transformation $H(k,w)$ est la somme directe de deux matrices $n \times n$ de Householder ordinaires [116]. Nous notons par $H(k,w)$ la transformation de Householder symplectique au sens de Van Loan. Remarquons que

$$H(k,w)e_i = e_i$$

pour $i = 1,\ldots,k-1$ et pour $i = n+1,\ldots,n+k-1$.

Le deuxième type est

$$J(k,c,s) = \begin{pmatrix} C & S \\ -S & C \end{pmatrix},$$

avec $c^2 + s^2 = 1$, et

$$C = \mathrm{diag}(I_{k-1}, c, I_{n-k})$$
$$S = \mathrm{diag}(0_{k-1}, s, 0_{n-k}).$$

Cette transformation $J(k,c,s)$ est une transformation de Givens symplectique, basée sur une rotation $2n \times 2n$ ordinaire de Givens dans les plans k et $n+k$ [116]. Nous notons par $J(k,c,s)$ la rotation de Givens symplectique au sens de Van Loan.

Remarquons que pour $i \neq k$ et $i \neq n+k$, nous avons $J(k,c,s)e_i = e_i$. De plus,

$$J(k,c,s)e_k = ce_k - se_{n+k}$$

et

$$J(k,c,s)e_{n+k} = se_k + ce_{n+k}.$$

La transformation $J(k,c,s)$ ne modifie que la $k^{\text{ième}}$ ligne et $(n+k)^{\text{ième}}$ ligne de la matrice à la quelle elle a été multiplié.

Les transformations de Householder et les rotations de Givens au sens de Van Loan [114] sont à la fois orthogonales et symplectiques.

Les étapes de l'algorithme $SRMSH$ sont comme suit :
Soit la matrice $A = [a_1, \ldots, a_p, a_{p+1}, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$ et posons que $A^{(0)} = A$.

La première étape :

Au début de l'algorithme $SRMSH$, nous voulons annuler les entrées de la première colonne de la position 2 jusqu'à la position $2n$. D'abord, nous appliquons la rotation de Givens symplectique au sens de Van Loan (Algorithme 2.15) à la matrice $A^{(0)}$ pour annuler les entrées de la position $2n$ jusqu'à la position $n+1$. En effet, pour $k = n, \ldots, 1$ nous calculons les constantes c et s de la matrice de rotation $J(k,c,s)$ pour mettre un zéro dans l'entrée de la position $n+k$ de la première colonne de la matrice $J(k,c,s)A^{(0)}$. Pour ce faire, nous appliquons algorithme J (vlg 2.15) avec le vecteur $v = A^{(0)}e_1$ pour déterminer la matrice de rotation $J(n,c,s)$. Puis, nous actualisons la matrice $A^{(0)}$ par la matrice $A^{(0)} = J(n,c,s)A^{(0)}$, ce qui modifie uniquement la $n^{\text{ième}}$ et la $2n^{\text{ième}}$ lignes de la matrice $A^{(0)}$. Ainsi de suite jusqu'à $k = 1$, avec le vecteur $v = A^{(0)}e_1$ nous calculons la matrice de rotation $J(1,c,s)$ et nous actualisons la matrice $A^{(0)}$ par la matrice $A^{(0)} = J(1,c,s)A^{(0)}$, ce qui modifie uniquement la première et la $(n+1)^{\text{ième}}$ lignes de la matrice $A^{(0)}$. De ce fait, la matrice $A^{(0)}$ est finalement actualisée par la nouvelle matrice $A'^{(1)} = J(1,c,s)\ldots J(n,c,s)A^{(0)}$. Donc, nous avons

$$A'^{(1)} = J(1,c,s)\ldots J(n,c,s)A^{(0)} = \begin{bmatrix} A'^{(1)}_{(1,1)} & A'^{(1)}_{(1,2:p)} & A'^{(1)}_{(1,p+1)} & A'^{(1)}_{(1,p+2:2p)} \\ A'^{(1)}_{(2:n,1)} & A'^{(1)}_{(2:n,2:p)} & A'^{(1)}_{(2:n,p+1)} & A'^{(1)}_{(2:n,p+2:2p)} \\ 0 & A'^{(1)}_{(n+1,2:p)} & A'^{(1)}_{(n+1,p+1)} & A'^{(1)}_{(n+1,p+2:2p)} \\ 0 & A'^{(1)}_{(n+2:2n,2:p)} & A'^{(1)}_{(n+2:2n,p+1)} & A'^{(1)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

Après, afin de créer des zéros dans les positions souhaitées, nous utilisons la transformation de Householder symplectique au sens de Van Loan (Algorithme 2.16). Nous calculons la constante β et le vecteur w de la transformation tel que :

$$H(1,w) = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w = I - \beta ww^{\mathrm{T}}.$$

La multiplication de la matrice $A'^{(1)}$ à gauche par la matrice $H(1,w)$ crée des zéros dans les entrées de la première colonne de la position 2 jusqu'à la position n.

$$A^{(1)} = H(1,w)A'^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:p)} & A^{(1)}_{(1,p+1)} & A^{(1)}_{(1,p+2:2p)} \\ 0 & A^{(1)}_{(2:n,2:p)} & A^{(1)}_{(2:n,p+1)} & A^{(1)}_{(2:n,p+2:2p)} \\ 0 & A^{(1)}_{(n+1,2:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2:2p)} \\ 0 & A^{(1)}_{(n+2:2n,2:p)} & A^{(1)}_{(n+2:2n,p+1)} & A^{(1)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

En général, chaque ligne de la matrice $A'^{(1)}$ est affectée par cette transformation, mais les zéros déjà créés dans la partie inférieure de la première colonne sont conservés.

□

La deuxième étape :

Ensuite, comme dans la deuxième étape de l'algorithme *SROSH*, nous utilisons la matrice $T_2 = I + c_2 v_2 v_2^J$ de la transformation de Householder symplectique de type deux pour mettre des zéros dans la $(p+1)^{\text{ième}}$ colonne de la matrice $A^{(1)}$ de la position 2 jusqu'à la position n et de la position $n+2$ jusqu'à la position $2n$, c'est-à-dire nous cherchons la constante c_2 et le vecteur v_2 qui sont définis par le Théorème 2.4.1.

Ainsi, nous avons

$$A^{(2)} = T_2 A^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2:p)} & A^{(2)}_{(1,p+1)} & A^{(2)}_{(1,p+2:2p)} \\ 0 & A^{(2)}_{(2:n,2:p)} & 0 & A^{(2)}_{(2:n,p+2:2p)} \\ 0 & A^{(1)}_{(n+1,2:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2:2p)} \\ 0 & A^{(2)}_{(n+2:2n,2:p)} & 0 & A^{(2)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

Puisque la composante $(n+1)^{\text{ième}}$ de l'entrée du vecteur v_2 est nulle (c'est-à-dire $v_2(n+1) = 0$), alors pour n'importe quel vecteur $x \in \mathbb{R}^{2n}$ la transformation T_2 ne modifie pas la composante $(n+1)^{\text{ième}}$ du vecteur $T_2(x)$. Ainsi, T_2 ne modifie pas la $(n+1)^{\text{ième}}$ ligne de la matrice $A^{(1)}$.

□

La troisième étape :

Dans cette étape le but est d'annuler les entrées de la position 3 jusqu'à la position n et de la position $n+2$ jusqu'à la position $2n$ de la deuxième colonne de la matrice $A^{(2)}$.

Soit $\tilde{A}^{(2)}$ la sous matrice obtenu en supprimant la première et la $(n+1)^{\text{ième}}$ lignes et la première et la $(p+1)^{\text{ième}}$ colonnes de la matrice $A^{(2)}$, c'est-à-dire

$$\tilde{A}^{(2)} = \begin{bmatrix} A^{(2)}_{(2:n,2:p)} & A^{(2)}_{(2:n,p+2:2p)} \\ A^{(2)}_{(n+2:2n,2:p)} & A^{(2)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

Nous répétons ici la première étape à la matrice $\tilde{A}^{(2)}$. En d'autres termes, pour $k = n-2, \ldots, 0$ nous calculons la matrice $\tilde{J}(k,c,s)$ de la rotation de Givens symplectique au sens de Van Loan pour mettre un zéro dans l'entrée de la position $n+k$ de la première colonne de la matrice $\tilde{A}^{(2)}$.

Ainsi, nous avons

$$\tilde{A}'^{(3)} = \tilde{J}(0,c,s)\ldots\tilde{J}(n-2,c,s)\tilde{A}^{(2)} = \begin{bmatrix} A'^{(3)}_{(2,2)} & A'^{(3)}_{(2,3:p)} & A'^{(3)}_{(2,p+2)} & A'^{(3)}_{(2,p+3:2p)} \\ A'^{(3)}_{(3:n,2)} & A'^{(3)}_{(3:n,3:p)} & A'^{(3)}_{(3:n,p+2)} & A'^{(3)}_{(3:n,p+3:2p)} \\ 0 & A'^{(3)}_{(n+2,3:p)} & A'^{(3)}_{(n+2,p+2)} & A'^{(3)}_{(n+2,p+3:2p)} \\ 0 & A'^{(3)}_{(n+3:2n,3:p)} & A'^{(3)}_{(n+3:2n,p+2)} & A'^{(3)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

Autrement dit, pour $k = n, \ldots, 2$ nous utilisons la matrice $J(k,c,s)$ de la rotation de Givens symplectique au sens de Van Loan pour annuler l'entrée de la position $n+k$ de la

deuxième colonne de la matrice $A^{(2)}$. Donc, nous actualisons la matrice $A^{(2)}$ par la matrice $A'^{(3)} = J(2,c,s)\dots J(n,c,s)A^{(2)}$. Par conséquent, nous avons :

$$A'^{(3)} = J(2,c,s)\dots J(n,c,s)A^{(2)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(2)}_{(1,3:n)} & A^{(2)}_{(1,n+1)} & A^{(2)}_{(1,n+2)} & A^{(2)}_{(1,n+3:2n)} \\ 0 & A'^{(3)}_{(2:n,2)} & A'^{(3)}_{(2:n,3:n)} & 0 & A'^{(3)}_{(2:n,n+2)} & A'^{(3)}_{(2:n,n+3:2n)} \\ 0 & A^{(1)}_{(n+1,2)} & A^{(1)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(1)}_{(n+1,n+2)} & A^{(1)}_{(n+1,n+3:2n)} \\ 0 & 0 & A'^{(3)}_{(n+2:2n,3:n)} & 0 & A'^{(3)}_{(n+2:2n,n+2)} & A'^{(3)}_{(n+2:2n,n+3:2n)} \end{bmatrix}.$$

Ensuite, pour terminer l'annulation du deuxième colonne de la matrice $A'^{(3)}$, nous utilisons la transformation de Householder symplectique au sens de Van Loan tel que $H(2,w) = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w$. Ainsi, des zéros apparaissent de la position 3 jusqu'à la position n dans la deuxième colonne de la matrice $A'^{(3)}$.

$$A^{(3)} = H(2,w)A'^{(3)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(2)}_{(1,3:p)} & A^{(2)}_{(1,p+1)} & A^{(2)}_{(1,p+2)} & A^{(2)}_{(1,p+3:2p)} \\ 0 & A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:p)} & 0 & A^{(3)}_{(2,p+2)} & A^{(3)}_{(2,p+3:2p)} \\ 0 & 0 & A^{(3)}_{(3:n,3:p)} & 0 & A^{(3)}_{(3:n,p+2)} & A^{(3)}_{(3:n,p+3:2p)} \\ 0 & A^{(1)}_{(n+1,2)} & A^{(1)}_{(n+1,3:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2)} & A^{(1)}_{(n+1,p+3:2p)} \\ 0 & 0 & A^{(3)}_{(n+2,3:p)} & 0 & A^{(3)}_{(n+2,p+2)} & A^{(3)}_{(n+2,p+3:2p)} \\ 0 & 0 & A^{(3)}_{(n+3:2n,3:p)} & 0 & A^{(3)}_{(n+3:2n,p+2)} & A^{(3)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

Dans cette étape, les multiplications par les matrices des transformations de Householder symplectiques ou les matrices des rotations de Givens symplectiques au sens de Van Loan ne modifient pas les zéros déjà créés et ne modifient pas la première et la $(n+1)^{\text{ième}}$ lignes et la première et la $(p+1)^{\text{ième}}$ colonnes de la matrice $A^{(2)}$.

□

La quatrième étape :

Cette étape consiste à répéter la deuxième étape, c'est-à-dire à travers une transformation de Householder symplectique de type deux $T_4 = I + c_4 v_4 v_4^J$ nous annulons les entrées de la $(p+2)^{\text{ième}}$ colonne de la matrice $A^{(3)}$ de la position 3 jusqu'à la position n et de la position $n+3$ jusqu'à la position $2n$. En d'autres termes, nous appliquons $\tilde{T}_4 = I + c_4 \tilde{v}_4 \tilde{v}_4^J$ à la sous matrice $\tilde{A}^{(3)}$ tel que cette dernière est obtenue en supprimant la première ligne et la $(n+1)^{\text{ième}}$ ligne et la première colonne et la $(p+1)^{\text{ième}}$ colonne de la matrice $A^{(3)}$:

$$\tilde{A}^{(3)} = \begin{bmatrix} A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:p)} & A^{(3)}_{(2,p+2)} & A^{(3)}_{(2,p+3:2p)} \\ 0 & A^{(3)}_{(3:n,3:p)} & A^{(3)}_{(3:n,p+2)} & A^{(3)}_{(3:n,p+3:2p)} \\ 0 & A^{(3)}_{(n+2,3:p)} & A^{(3)}_{(n+2,p+2)} & A^{(3)}_{(n+2,p+3:2p)} \\ 0 & A^{(3)}_{(n+3:2n,3:p)} & A^{(3)}_{(n+3:2n,p+2)} & A^{(3)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

La multiplication de la matrice $\tilde{A}^{(3)}$ par la matrice $\tilde{T}_4$ annule les entrées de la position 2 jusqu'à la position $n-1$ et de la position $n+1$ jusqu'à la position $2n-2$ de la $p^{\text{ième}}$ colonne tel que :

$$\tilde{A}^{(4)} = \tilde{T}_4 \tilde{A}^{(3)} = \begin{bmatrix} A^{(3)}_{(2,2)} & A^{(4)}_{(2,3:p)} & A^{(4)}_{(2,p+2)} & A^{(4)}_{(2,p+3:2p)} \\ 0 & A^{(4)}_{(3:n,3:p)} & 0 & A^{(4)}_{(3:n,p+3:2p)} \\ 0 & A^{(3)}_{(n+2,3:p)} & A^{(3)}_{(n+2,p+2)} & A^{(3)}_{(n+2,p+3:2p)} \\ 0 & A^{(4)}_{(n+3:2n,3:p)} & 0 & A^{(4)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

Ainsi, nous avons

$$A^{(4)} = T_4 A^{(3)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(2)}_{(1,3:p)} & A^{(2)}_{(1,p+1)} & A^{(2)}_{(1,p+2)} & A^{(2)}_{(1,p+3:2p)} \\ 0 & A^{(3)}_{(2,2)} & A^{(4)}_{(2,3:p)} & 0 & A^{(4)}_{(2,p+2)} & A^{(4)}_{(2,p+3:2p)} \\ 0 & 0 & A^{(4)}_{(3:n,3:p)} & 0 & 0 & A^{(4)}_{(3:n,p+3:2p)} \\ 0 & A^{(1)}_{(n+1,2)} & A^{(1)}_{(n+1,3:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2)} & A^{(1)}_{(n+1,p+3:2p)} \\ 0 & 0 & A^{(3)}_{(n+2,3:p)} & 0 & A^{(3)}_{(n+2,p+2)} & A^{(3)}_{(n+2,p+3:2p)} \\ 0 & 0 & A^{(4)}_{(n+3:2n,3:p)} & 0 & 0 & A^{(4)}_{(n+3:2n,p+3:2p)} \end{bmatrix}.$$

La transformation T_4 ne modifie pas la première ligne, la $(n+1)^{\text{ième}}$ ligne et la $(n+2)^{\text{ième}}$ ligne et la première colonne, la deuxième colonne et la $(p+1)^{\text{ième}}$ colonne de la matrice $A^{(3)}$.

□

Ainsi de suite jusqu'à la fin, à chaque fois nous répétons la première et la deuxième étapes. Ensuite, nous réduisons la taille de la matrice $A^{(2j)}$ avec $j = 1, \ldots, p$ en éliminant la première ligne et la $(n+1)^{\text{ième}}$ ligne et la première colonne et la $(p+1)^{\text{ième}}$ colonne.

Finalement, nous obtenons

$$A = SR = S \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} = S \begin{bmatrix} \blacktriangle & \blacktriangle \\ 0 \blacktriangle & \blacktriangle \\ 0 & \end{bmatrix}.$$

avec les blocs R_{11}, R_{12}, $R_{22} \in \mathbb{R}^{n\times p}$ sont des matrices triangulaires supérieurs et le bloc $R_{21} \in \mathbb{R}^{n\times p}$ est une matrice strictement triangulaire supérieure. La matrice $R \in \mathbb{R}^{2n\times 2p}$ est une matrice J-triangulaire supérieure. La matrice $S \in \mathbb{R}^{2n\times 2n}$ est une matrice symplectique.

Nous présentons ici l'algorithme *SRMSH*, pour calculer la décomposition *SR* d'une matrice $A \in \mathbb{R}^{2n\times 2p}$.

Algorithme 2.14 Algorithme $SRMSH$

Cet algorithme calcule la décomposition SR une matrice $A \in \mathbb{R}^{2n\times 2p}$ tel que $A = SR$ avec la matrice $S \in \mathbb{R}^{2n\times 2n}$ est une matrice symplectique. La matrice A est remplacées par la matrice J-triangulaire $R \in \mathbb{R}^{2n\times 2p}$.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2p}$ et $S = I_{2n}$.

SORTIE(S) : R la matrice sous la forme J-triangulaire et S la matrice symplectique.

1: Fonction $[S,R] = SRMSH(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $S = eye(den)$;
6: **Pour** $j = 1 : p$ **faire**
7: $J = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
8: $ro = [j : n, n+j : 2n]$;
9: $co = [j : p, p+j : 2p]$;
% Calculer les constantes c et s de la matrice $J(k,c,s)$ de la rotation de Givens symplectique au sens de Van Loan afin d'annuler les entrées souhaitées dans la partie inférieure de la $j^{\text{ième}}$ colonne.
10: **Pour** $k = n : -1 : j$ **faire**
11: $[c,s] = vlg(k, A(:,j))$;
% Mise à jour de la matrice A.
12: $vlA = A(k,co)$;
13: $A(k,co) = c \times A(k,co) + s \times A(n+k,co)$;
14: $A(n+k,co) = -s \times vlA + c \times A(n+k,co)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
15: $vcS = S(:,k)$;
16: $S(:,k) = c \times S(:,k) + s \times S(:,n+k)$;
17: $S(:,n+k) = -s \times vcS + c \times S(:,n+k)$;
18: **Fin Pour**
% Calculer la constante β et le vecteur w de la matrice $H(j,w)$ de la transformation de Householder symplectique au sens de Van Loan afin d'annuler les entrées souhaitées dans la partie supérieure de la $j^{\text{ième}}$ colonne.
19: **Si** $j \leq n-1$ **alors**
20: $[\beta, w] = vlh(j, A(:,j))$;
% Mise à jour de la matrice A.
21: $A(j:n,co) = A(j:n,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j:n,co)$;
22: $A(j+n:den,co) = A(j+n:den,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+n:den,co)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
23: $S(:,j:n) = S(:,j:n) - \beta \times S(:,j:n) \times (w \times w^{\mathrm{T}})$;
24: $S(:,n+j:den) = S(:,n+j:den) - \beta \times S(:,n+j:den) \times (w \times w^{\mathrm{T}})$;
25: **Fin Si**
% Calculer la constant c et le vecteur v_2 de la transformation de Householder symplectique optimale de type deux tel que $T_2 = I + c v_2 v_2^J$ pour mettre des zéros dans les entrées souhaitées de la $(p+j)^{\text{ième}}$ colonne.
26: $[c, v2] = osh2(A(ro, p+j))$;

La suite de l'algorithme *SRMSH*

% Mise à jour de la matrice A.

27: $A(ro, co) = A(ro, co) + c \times v_2 \times (v_2^{\mathrm{T}} \times J \times A(ro, co))$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

28: $S(:, ro) = S(:, ro) - (S(:, ro) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;

29: **Fin Pour**

30: $R = A$

31: **Fin**

Dans l'algorithme *SRMSH* nous utilisons la rotation de Givens symplectique au sens de Van Loan vlg (Algorithme *J*) qui calcule les constantes c et s de la matrice $J(k, c, s)$ pour mettre un zéro dans l'entrée de la position $n + k$ de la colonne j avec $k = n, \ldots, j$. L'algorithme vlg est le suivant :

Algorithme 2.15 Algorithme de la rotation de Givens symplectique au sens de Van Loan vlg

Cet algorithme calcule les constantes c et s de la matrice $J(k, c, s)$ de la rotation de Givens symplectique au sens de Van Loan avec

$$J(k,c,s) = \begin{pmatrix} I_{k-1} & & & & & \\ & c & & & s & \\ & & I_{n-k} & & & \\ & & & I_{k-1} & & \\ & -s & & & c & \\ & & & & & I_{n-k} \end{pmatrix}, \text{ où } c^2 + s^2 = 1.$$

La matrice $J(k, c, s)$ annule l'entrée de la position $n + k$ du vecteur a, c'est-à-dire si $b = J(k, c, s)a$, alors $b(n + k) = 0$.

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$ et une constante k.

SORTIE(S) : Les constantes c et s de la matrice de rotation de Givens symplectique au sens de Van Loan tel que $c^2 + s^2 = 1$ et l'entrée de la position $n+k$ du vecteur $J(k, c, s)a$ est zéro.

1: Fonction $[c, s] = vlg(k, a)$

2: $den = length(a)$;

3: $n = den/2$;

4: $r = \sqrt{a(k)^2 + a(n+k)^2}$;

5: **Si** $r = 0$ **alors**

6: $c = 1$;

7: $s = 0$;

8: **Sinon**

9: $c = \dfrac{a_k}{r}$;

10: $s = \dfrac{a_{n+k}}{r}$;

11: **Fin Si**

12: **Fin**

Aussi dans l'algorithme $SRMSH$, nous utilisons la transformation de Householder symplectique au sens de Van Loan vlh (Algorithme H). Ce dernier calcule la constante β et le vecteur $w = (w_1, \ldots, w_{n-k+1})^{\mathrm{T}} \in \mathbb{R}^{n-k+1}$ de la matrice $H(k,w)$ pour mettre des zéros de la position $k+1$ jusqu'à la position n. L'algorithme vlh est le suivant :

Algorithme 2.16 Algorithme de la transformation de Householder symplectique au sens de Van Loan vlh

Cet algorithme calcule la constante β et le vecteur w de la matrice $H(k,w)$ de la transformation de Householder symplectique au sens de Van Loan avec

$$H(k,w) = \begin{pmatrix} \mathrm{diag}(I_{k-1}, P) & 0 \\ 0 & \mathrm{diag}(I_{k-1}, P) \end{pmatrix},$$

où $P = I_{n-k+1} + \beta w w^{\mathrm{T}} = I_{n-k+1} + 2ww^{\mathrm{T}}/w^{\mathrm{T}}w$ et $w = (w_1, \ldots, w_{n-k+1})^{\mathrm{T}} \in \mathbb{R}^{n-k+1}$. La matrice $H(k,w)$ annule les entrées aux positions $k+1, \ldots, n$ du vecteur a, c'est-à-dire si $b = H(k,w)a$, alors $b(k+1:n) = 0$.

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$ et une constante k.

SORTIE(S) : La constante β et le vecteur w de la matrice de transformation de Householder symplectique au sens de Van Loan tel que $c^2 + s^2 = 1$ et l'entrée de la position $n+k$ du vecteur $J(k,c,s)a$ est zéro.

1: Fonction $[\beta, w] = vlh(k,a)$
2: $den = length(a)$;
3: $n = den/2$;
4: $r_1 = \sum_{i=2}^{n-k+1} a(i+k-1)^2$;
5: $r = \sqrt{a(k)^2 + r_1}$;
6: $w_1 = a(k) + sign(a(k))r$;
7: **Pour** $k = 2, \ldots, n-k+1$ **faire**
8: $\quad w_i = a_{i+k-1}$;
9: **Fin Pour**
10: $r = w_1^2 + r_1$;
11: **Si** $r \neq 0$ **alors**
12: $\quad \beta = \dfrac{2}{r}$;
13: **Sinon**
14: $\quad beta = 0$;
15: **Fin Si**
$\%P = I + \beta w w^{\mathrm{T}}$ tel que $(H(k,w)a)_i = 0$ pour $i = k+1, \ldots, n$.
16: **Fin**

2.6 La décomposition SR via l'algorithme SRMSH2

Dans ce paragraphe, nous présentons un autre nouvel algorithme de décomposition SR via $SRMSH2$. Cet algorithme est une version modifiée de l'algorithme $SRMSH$. En effet, nous allons modifier les étapes paires. Au lieu d'utiliser la transformation de Householder symplectique optimale de type deux seulement pour mettre des zéros dans la

colonne $p+j$, nous allons utiliser d'abord la rotation de Givens symplectique au sens de Van Loan pour la partie inférieure du vecteur. Puis, la transformation de Householder symplectique au sens de Van Loan pour la partie supérieure du vecteur. Enfin, pour annuler la dernière entrée dans la position $(j+1)$ du vecteur $(p+j)$, nous utilisons la transformation de Householder symplectique optimale de type deux.

Ainsi, dans la suite, nous donnons une brève description des étapes d'algorithme $SRMSH2$:

Soit la matrice $A = [a_1, \ldots, a_p, a_{p+1}, \ldots, a_{2p}] \in \mathbb{R}^{2n\times 2p}$ et posons que $A^{(0)} = A$.

La première étape :

Cette étape est identique à la première étape de l'algorithme $SRMSH$. Nous voulons mettre des zéros dans la première colonne de la matrice $A^{(0)}$ de la position 2 jusqu'à la position $2n$.

D'abord, nous commençons par la partie inférieure de la première colonne. Nous utilisons la rotation de Givens symplectique au sens de Van Loan pour annuler les entrées de la position $2n$ jusqu'à la position $(n+1)$. Ainsi, nous calculons les matrices $J(k,c,s)$ pour $k = n, \ldots, 1$ afin de mettre un zéro dans l'entrée de la position $(n+k)$ de la première colonne de la matrice $A^{(0)}$. Par conséquent, la matrice de départ est actualisée par la matrice $A'^{(1)} = J(1,c,s)\ldots J(n,c,s)A^{(0)}$. Ainsi,

$$A'^{(1)} = J(1,c,s)\ldots J(n,c,s)A^{(0)} = \begin{bmatrix} A'^{(1)}_{(1,1)} & A'^{(1)}_{(1,2:p)} & A'^{(1)}_{(1,p+1)} & A'^{(1)}_{(1,p+2:2p)} \\ A'^{(1)}_{(2:n,1)} & A'^{(1)}_{(2:n,2:p)} & A'^{(1)}_{(2:n,p+1)} & A'^{(1)}_{(2:n,p+2:2p)} \\ 0 & A'^{(1)}_{(n+1,2:p)} & A'^{(1)}_{(n+1,p+1)} & A'^{(1)}_{(n+1,p+2:2p)} \\ 0 & A'^{(1)}_{(n+2:2n,2:p)} & A'^{(1)}_{(n+2:2n,p+1)} & A'^{(1)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

Ensuite, pour la partie supérieure de la première colonne de la matrice $A'^{(1)}$ nous utilisons la transformation de Householder symplectique au sens de Van Loan $H(1,w)$. Nous calculons la constante β et le vecteur w de la transformation tel que

$$H(1,w) = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w = I - \beta ww^{\mathrm{T}},$$

pour crée des zéros dans les entrées des positions $2,\ldots,n$ de la première colonne.

Donc, nous avons

$$A^{(1)} = H(1,w)A'^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:p)} & A^{(1)}_{(1,p+1)} & A^{(1)}_{(1,p+2:2p)} \\ 0 & A^{(1)}_{(2:n,2:p)} & A^{(1)}_{(2:n,p+1)} & A^{(1)}_{(2:n,p+2:2p)} \\ 0 & A^{(1)}_{(n+1,2:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2:2p)} \\ 0 & A^{(1)}_{(n+2:2n,2:p)} & A^{(1)}_{(n+2:2n,p+1)} & A^{(1)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

□

La deuxième étape :

Ensuite, nous modifions cette étape par rapport à la deuxième étape de l'algorithme $SRMSH$ en appliquant deux transformations symplectiques et orthogonales au sens de

Van Loan puis nous utilisons la transformation de Householder symplectique de type deux pour mettre des zéros dans la $(p+1)^{\text{ième}}$ colonne de la matrice $A^{(1)}$ de la position 2 jusqu'à la position n et de la position $n+2$ jusqu'à la position $2n$.

En effet, nous commençons en premier lieu par annuler le bas de la $(p+1)^{\text{ième}}$ colonne en utilisant la rotation de Givens symplectique au sens de Van Loan. Ainsi, nous calculons les constantes c et s de la matrice de rotation $J(k,c,s)$, afin de créer un zéro dans l'entrée de la position $(n+k, p+1)$ avec $k = n, \ldots, 2$.

Donc, nous avons

$$A'^{(2)} = J(2,c,s)\ldots J(n,c,s)A^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:p)} & A^{(1)}_{(1,p+1)} & A^{(1)}_{(1,p+2:2p)} \\ 0 & A'^{(2)}_{(2:n,2:p)} & A'^{(2)}_{(2:n,p+1)} & A'^{(2)}_{(2:n,p+2:2p)} \\ 0 & A^{(1)}_{(n+1,2:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2:2p)} \\ 0 & A'^{(2)}_{(n+2:2n,2:p)} & 0 & A'^{(2)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

Ainsi, les transformations $J(2,c,s) \times \ldots \times J(n,c,s)$ de Givens symplectiques au sens de Van Loan ne modifient pas la première et la $(n+1)^{\text{ième}}$ lignes de la matrice $A^{(1)}$.

Après, nous annulons en second lieu les entrées de la position 3 jusqu'à la position n en appliquant la transformation de Householder symplectique au sens de Van Loan

$$H(2,w) = I - \beta w w^{\mathrm{T}}.$$

En conséquence, nous avons

$$A''^{(2)} = H(2,w)A'^{(2)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:p)} & A^{(1)}_{(1,p+1)} & A^{(1)}_{(1,p+2:2p)} \\ 0 & A''^{(2)}_{(2,2:p)} & A''^{(2)}_{(2,p+1)} & A''^{(2)}_{(2,p+2:2p)} \\ 0 & A''^{(2)}_{(3:n,2:p)} & 0 & A''^{(2)}_{(3:n,p+2:2p)} \\ 0 & A^{(1)}_{(n+1,2:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2:2p)} \\ 0 & A''^{(2)}_{(n+2:2n,2:p)} & 0 & A''^{(2)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

Pour finir, il nous reste à annuler l'entrée $A''^{(2)}_{(2,p+1)}$ par la transformation de Householder symplectique de type deux $T_2 = I + c_2 v_2 v_2^J$. En effet, la $(p+1)^{\text{ième}}$ colonne de la matrice $A''^{(2)}$ est sous la forme suivante :

$$A''^{(2)}(:,p+1) = \overbrace{\begin{bmatrix} A^{(1)}_{(1,p+1)} \\ A''^{(2)}_{(2,p+1)} \\ 0 \\ \hline A^{(1)}_{(n+1,p+1)} \\ 0 \end{bmatrix}}^{\text{la } (p+1)^{\text{ième}} \text{ colonne de } A''^{(2)}} \quad \begin{matrix} \} & (1) \\ \} & (1) \\ \} & (n-2) \\ \hline \} & (1) \\ \} & (n-1) \end{matrix}.$$

Ainsi, pour calculer la constante c_2 et le vecteur v_2 nous pouvons utiliser la transformation de Householder symplectique T_2 avec un vecteur de dimension 4 au-lieu de $2n$ parce que les restes des entrées du vecteur $A''^{(2)}(:,p+1)$ sont déjà nulles. Donc, nous utilisons le vecteur suivant :

$$\begin{bmatrix} A^{(1)}_{(1,p+1)} \\ \hline A''^{(2)}_{(2,p+1)} \\ \hline A^{(1)}_{(n+1,p+1)} \\ 0 \end{bmatrix} \begin{matrix} \} & (1) \\ \} & (1) \\ \hline \} & (1) \\ \} & (1) \end{matrix}.$$

De ce fait, nous avons :

$$A^{(2)} = T_2 A''^{(2)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2:p)} & A^{(2)}_{(1,p+1)} & A^{(2)}_{(1,p+2:2p)} \\ 0 & A^{(2)}_{(2,2:p)} & 0 & A^{(2)}_{(2,p+2:2p)} \\ 0 & A''^{(2)}_{(3:n,2:p)} & 0 & A''^{(2)}_{(3:n,p+2:2p)} \\ 0 & A^{(1)}_{(n+1,2:p)} & A^{(1)}_{(n+1,p+1)} & A^{(1)}_{(n+1,p+2:2p)} \\ 0 & A''^{(2)}_{(n+2:2n,2:p)} & 0 & A''^{(2)}_{(n+2:2n,p+2:2p)} \end{bmatrix}.$$

La multiplication de la matrice $A''^{(2)}$ par la transformation T_2 modifie la première ligne et la deuxième ligne seulement de la position 2 jusqu'à la position $2n$.

□

Ainsi de suite jusqu'à l'étape $(2p)^{\text{ième}}$, à chaque fois nous répétons la première et la deuxième étapes et nous déminions à la fin de ces deux étapes la taille de la matrice en supprimant la première et la $(n+1)^{\text{ième}}$ lignes et la première et la $(p+1)^{\text{ième}}$ colonnes.

Finalement, nous obtenons

$$A = SR = S \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} = S \begin{bmatrix} \blacktriangle & \blacktriangle \\ 0 & \blacktriangle \end{bmatrix},$$

avec les blocs R_{11}, R_{12}, R_{22} sont des matrices triangulaires supérieures et le bloc R_{21} est une matrice strictement triangulaire supérieure. La matrice $R \in \mathbb{R}^{2n\times 2p}$ est une matrice J-triangulaire supérieure. La matrice $S \in \mathbb{R}^{2n\times 2n}$ est une matrice symplectique.

Cette description nous ramène à l'algorithme *SRMSH2* pour la décomposition *SR* d'une matrice $A \in \mathbb{R}^{2n\times 2p}$:

Algorithme 2.17 Algorithme $SRMSH2$

Cet algorithme calcule la décomposition SR de la matrice $A \in \mathbb{R}^{2n\times 2p}$ de telle sorte que $A = SR$ avec $S \in \mathbb{R}^{2n\times 2n}$ est une matrice symplectique et la matrice A est remplacée par la matrice J-triangulaire $R \in \mathbb{R}^{2n\times 2p}$.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2p}$ et $S = I_{2n}$.

SORTIE(S) : La matrice R sous la forme J-triangulaire. La matrice S qui est une matrice symplectique. Tel que $A = SR$.

1: Fonction $[S,R] = SRMSH2(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $S = eye(den)$;
6: $J2 = [zeros(2), eye(2); -eye(2), zeros(2)]$;
7: **Pour** $j = 1 : p$ **faire**
8: $ro1 = [j : j+1, n+j : n+j+1]$;
9: $co = [j : p, p+j : 2p]$;
% Mettre des zéros dans les positions souhaitées de la $j^{\text{ième}}$ colonne :
% Calculer les constantes c et s de la matrice $J(k,c,s)$ de la rotation de Givens symplectique au sens de Van Loan dans l'intention d'annuler les entrées désirées dans la partie inférieure de la $j^{\text{ième}}$ colonne.
10: **Pour** $k = n : -1 : j$ **faire**
11: $[c,s] = vlg(k, A(:,j))$;
% Mise à jour de la matrice A.
12: $vlA = A(k,co)$;
13: $A(k,co) = c \times A(k,co) + s \times A(n+k,co)$;
14: $A(n+k,co) = -s \times vlA + c \times A(n+k,co)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
15: $vcS = S(:,k)$;
16: $S(:,k) = c \times S(:,k) + s \times S(:,n+k)$;
17: $S(:,n+k) = -s \times vcS + c \times S(:,n+k)$;
18: **Fin Pour**
% Calculer la constante β et le vecteur w de la matrice $H(j,w)$ de la transformation de Householder symplectique au sens de Van Loan dans le but d'annuler les entrées désirées dans la partie supérieure de la $j^{\text{ième}}$ colonne.
19: **Si** $j \leq n-1$ **alors**
20: $[\beta, w] = vlh(j, A(:,j))$;
% Mise à jour de la matrice A.
21: $A(j:n,co) = A(j:n,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j:n,co)$;
22: $A(j+n:den,co) = A(j+n:den,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+n:den,co)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
23: $S(:,j:n) = S(:,j:n) - \beta \times S(:,j:n) \times (w \times w^{\mathrm{T}})$;
24: $S(:,n+j:den) = S(:,n+j:den) - \beta \times S(:,n+j:den) \times (w \times w^{\mathrm{T}})$;
25: **Fin Si**

La suite de l'algorithme $SRMSH2$

% Mettre des zéros dans les positions souhaitées de la $(p+j)^{\text{ième}}$ colonne :

Si $j \leq n-1$ **alors**

% Calculer les constantes c et s de la matrice $J(k,c,s)$ de la rotation de Givens symplectique au sens de Van Loan dans l'intention d'annuler les entrées désirées dans la partie inférieure de la $(p+j)^{\text{ième}}$ colonne.

Pour $k = n:-1:j+1$ **faire**

$[c,s] = vlg(k, A(:, p+j))$;

% Mise à jour de la matrice A.

$vlA = A(k, co)$;

$A(k, co) = c \times A(k, co) + s \times A(n+k, co)$;

$A(n+k, co) = -s \times vlA + c \times A(n+k, co)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$vcS = S(:, k)$;

$S(:, k) = c \times S(:, k) + s \times S(:, n+k)$;

$S(:, n+k) = -s \times vcS + c \times S(:, n+k)$;

Fin Pour

Si $j \leq n-2$ **alors**

% Calculer la constante β et le vecteur w de la matrice $H(j+1, w)$ de la transformation de Householder symplectique au sens de Van Loan dans le but d'annuler les entrées désirées dans la partie supérieure de la $(p+j)^{\text{ième}}$ colonne.

$[\beta, w] = vlh(j+1, A(:, p+j))$;

% Mise à jour de la matrice A.

$A(j+1:n, co) = A(j+1:n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1:n, co)$;

$A(j+1+n:den, co) = A(j+1+n:den, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1+n:den, co)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$S(:, j+1:n) = S(:, j+1:n) - \beta \times S(:, j+1:n) \times (w \times w^{\mathrm{T}})$;

$S(:, n+j+1:den) = S(:, n+j+1:den) - \beta \times S(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;

Fin Si

% Calculer la constant c et le vecteur v_2 de la transformation de Householder symplectique optimale de type deux tel que $T_2 = I + c v_2 v_2^J$ pour mettre des zéros dans l'entrée $(j+1)$ de la $(p+j)^{\text{ième}}$ colonne.

$[c, v2] = osh2(A(ro1, p+j))$;

% Mise à jour de la matrice A.

$A(ro1, co) = A(ro1, co) + c \times v_2 \times (v_2^{\mathrm{T}} \times J \times A(ro1, co))$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$S(:, ro1) = S(:, ro1) - (S(:, ro1) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;

Fin Si

Fin Pour

$R = A$

Fin

Dans l'algorithme $SRMSH2$, nous utilisons la rotation de Givens symplectique au sens de Van Loan vlg via l'algorithme 2.15 (Algorithme J) qui calcule les constantes c

et s de la matrice $J(k,c,s)$ pour mettre un zéro dans l'entrée de la position $(n+k)$ dans les colonnes j et $(p+j)$ avec $k=n,\ldots,j$. Aussi dans cet l'algorithme, nous utilisons la transformation de Householder symplectique au sens de Van Loan vlh via l'algorithme 2.16 (Algorithme H) qui calcule le vecteur $w=(w_1,\ldots,w_{n-k+1})^{\mathrm{T}}\in\mathbb{R}^{n-k+1}$ et la constante β de la matrice $H(k,w)$ pour mettre des zéros de la position $(k+1)$ jusqu'à la position n dans la $j^{\text{ième}}$ colonne et la $(p+j)^{\text{ième}}$ colonne.

2.7 La décomposition SR via l'algorithme SRDECO

Un algorithme appelé *SRDECO* de décomposition *SR* d'une matrice A arbitraire de taille $(2n\times 2n)$ a été présenté par Bunse-Gerstner et Mehrmann dans [40]. En fait, la construction de l'algorithme *SRDECO* a été basée sur les transformations de Givens symplectiques, les transformations de Householder symplectiques qui ont été introduites par Van Loan [114] et complétées par les transformations de Gauss symplectiques [40]. Sachant que les deux premières transformations de Givens symplectiques et de Householder symplectiques sont orthogonales, tandis que les dernières ne le sont pas. De point de vue algèbre linéaire, la décomposition *SR* via l'algorithme *SRDECO* ne correspond pas à l'analogue de la décomposition *QR* via Householder.

Maintenant, nous allons commencer par rappeler les étapes de la construction d'une décomposition *SR* d'une matrice A arbitraire de taille $2n\times 2n$ via l'algorithme *SRDECO*. Avant de donner l'algorithme correspondant, Nous illustrerons les deux premières étapes d'élimination.

Soit la matrice $A=[a_1,\ldots,a_n,a_{n+1},\ldots,a_{2n}]\in\mathbb{R}^{2n\times 2n}$ et posons que $A^{(0)}=A$.

La première étape :

L'algorithme commence par mettre des zéros dans la première colonne de la matrice $A^{(0)}$. À travers les transformations de Givens symplectiques au sens de Van Loan (Algorithme J), nous annulons les éléments de la position $2n$ jusqu'à la position $(n+1)$ de la première colonne. Pour cela, nous calculons les matrices $J(k,c,s)$ avec $k=n,\ldots,1$ afin d'annuler l'entrée de la position $(n+k)$. La nouvelle matrice est sous la forme

$$A'^{(1)}=J(1,c,s)\ldots J(n,c,s)A^{(0)}=\begin{bmatrix} A'^{(1)}_{(1,1)} & A'^{(1)}_{(1,2:n)} & A'^{(1)}_{(1,n+1)} & A'^{(1)}_{(1,n+2:2n)}\\ A'^{(1)}_{(2:n,1)} & A'^{(1)}_{(2:n,2:n)} & A'^{(1)}_{(2:n,n+1)} & A'^{(1)}_{(2:n,n+2:2n)}\\ 0 & A'^{(1)}_{(n+1,2:n)} & A'^{(1)}_{(n+1,n+1)} & A'^{(1)}_{(n+1,n+2:2n)}\\ 0 & A'^{(1)}_{(n+2:2n,2:n)} & A'^{(1)}_{(n+2:2n,n+1)} & A'^{(1)}_{(n+2:2n,n+2:2n)}\end{bmatrix}.$$

En utilisant la transformation de Householder symplectique au sens de Van Loan (Algorithme H), nous annulons les éléments de la position 2 jusqu'à la position n dans la première colonne de la matrice $A'^{(1)}$. Pour effectuer cela, nous calculons la constante β et le vecteur w de la transformation tel que :

$$H(1,w)=I-2ww^{\mathrm{T}}/w^{\mathrm{T}}w=I-\beta ww^{\mathrm{T}}.$$

Nous actualisons la matrice $A'^{(1)}$ par la matrice

$$A^{(1)} = H(1,w)A'^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(1)}_{(1,n+2:2n)} \\ 0 & A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1)} & A^{(1)}_{(2:n,n+2:2n)} \\ 0 & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(1)}_{(n+1,n+2:2n)} \\ 0 & A^{(1)}_{(n+2:2n,2:n)} & A^{(1)}_{(n+2:2n,n+1)} & A^{(1)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

□

La deuxième étape :

La deuxième étape consiste à mettre des zéros dans la $(n+1)^{\text{ième}}$ colonne de la matrice $A^{(0)}$. Nous annulons les éléments de la position $2n$ jusqu'à la position $(n+1)$ de la $(n+1)^{\text{ième}}$ colonne, en utilisant les transformations de Givens symplectiques au sens de Van Loan (Algorithme J). En effet, nous nous calculons les matrices $J(k,c,s)$ pour annuler l'entrée de la position $(n+k)$ avec $k = n,\ldots,1$.

Donc, nous avons

$$A'^{(2)} = J(2,c,s)\ldots J(n,c,s)A^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(1)}_{(1,n+2:2n)} \\ 0 & A'^{(2)}_{(2:n,2:n)} & A'^{(2)}_{(2:n,n+1)} & A'^{(2)}_{(2:n,n+2:2n)} \\ 0 & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(1)}_{(n+1,n+2:2n)} \\ 0 & A'^{(2)}_{(n+2:2n,2:n)} & 0 & A'^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

Les transformations de Givens symplectiques ne modifient pas ni les zéros déjà créés ni la première ligne et la $(n+1)^{\text{ième}}$ ligne de la matrice $A^{(1)}$.

Ensuite, nous appliquons la transformation de Householder symplectique au sens de Van Loan

$$H(2,w) = I - \beta w w^{\mathrm{T}}$$

pour annuler les entrées de la position 3 jusqu'à la position n. Donc, nous avons

$$A''^{(2)} = H(2,w)A'^{(2)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(1)}_{(1,n+2:2n)} \\ 0 & A''^{(2)}_{(2,2:n)} & A''^{(2)}_{(2,n+1)} & A''^{(2)}_{(2,n+2:2n)} \\ 0 & A''^{(2)}_{(3:n,2:n)} & 0 & A''^{(2)}_{(3:n,n+2:2n)} \\ 0 & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(1)}_{(n+1,n+2:2n)} \\ 0 & A''^{(2)}_{(n+2:2n,2:n)} & 0 & A''^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

Enfin, nous devons détruire l'entrée de la position $(2,n+1)$. Nous ne pouvons pas utiliser ni les transformations de Givens symplectiques ni les transformations de Householder symplectiques au sens de Van Loan parce que dans le cas contraire nous remplirons l'entrée de la position $(n+2,n+1)$ ou les entrées des positions $(2,1)$ et $(n+2,n+1)$, respectivement.

Dans le cas où l'entrée de la position $(n+1,n+1)$ est non nulle, nous pouvons appliquer la transformation de Gauss symplectique via l'Algorithme G (Algorithme sgt) avec le vecteur $v = A''^{(2)}e_4$. Ainsi, nous obtenons la matrice $G(2,v)$.

Donc, nous avons maintenant

$$A^{(2)} = G(2,v)A''^{(2)} = \begin{bmatrix} A^{(2)}_{(1,1)} & A^{(2)}_{(1,2:n)} & A^{(2)}_{(1,n+1)} & A^{(2)}_{(1,n+2:2n)} \\ 0 & A^{(2)}_{(2,2:n)} & 0 & A^{(2)}_{(2,n+2:2n)} \\ 0 & A''^{(2)}_{(3:n,2:n)} & 0 & A''^{(2)}_{(3:n,n+2:2n)} \\ 0 & A^{(2)}_{(n+1,2:n)} & A^{(2)}_{(n+1,n+1)} & A^{(2)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2,2:n)} & 0 & A^{(2)}_{(n+2,n+2:2n)} \\ 0 & A''^{(2)}_{(n+3:2n,2:n)} & 0 & A''^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

L'Algorithme G modifie la première ligne, la deuxième ligne, la $(n+1)^{\text{ième}}$ ligne et la $(n+2)^{\text{ième}}$ ligne de la matrice $A''^{(2)}$.

Remarquons que si l'entrée de la position $(2, n+1)$ est différente de zéro mais l'entrée de la position $(n+1, n+1)$ est nulle, alors la matrice A n'a pas de décomposition SR et par suite l'algorithme s'arrête.

□

Ainsi de suite, à chaque fois nous répétons les deux étapes précédentes et nous déminions la taille de la matrice en supprimant la première et la $(n+1)^{\text{ième}}$ lignes et colonnes. Enfin, nous obtenons $A = SR$ avec R est une matrice J-triangulaire supérieure et S est une matrice symplectique.

L'algorithme $SRDECO$ de la décomposition SR d'une matrice A arbitraire de taille $(2n \times 2n)$ est le suivant :

Algorithme 2.18 Algorithme $SRDECO$

Cet algorithme calcule la décomposition SR de la matrice $A \in \mathbb{R}^{2n\times 2n}$ de telle sorte que $A = SR$ avec $S \in \mathbb{R}^{2n\times 2n}$ est une matrice symplectique et la matrice A est remplacée par la matrice J-triangulaire $R \in \mathbb{R}^{2n\times 2n}$, si cette décomposition existe. Dans le cas où la matrice A n'a pas de décomposition SR, alors l'algorithme s'arrête.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2p}$ et $S = I_{2n}$.

SORTIE(S) : La matrice R sous la forme J-triangulaire. La matrice S qui est une matrice symplectique. Tel que $A = SR$.

1: Fonction $[S, R] = SRDECO(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $S = eye(den)$;
6: **Pour** $j = 1 : p$ **faire**
7: $co = [j : p, p + j : 2p]$;
% Mettre des zéros dans les positions souhaitées de la $j^{\text{ième}}$ colonne :
% Calculer les constantes c et s de la matrice $J(k, c, s)$ de la rotation de Givens symplectique au sens de Van Loan dans l'intention d'annuler les entrées désirées dans la partie inférieure de la $j^{\text{ième}}$ colonne.
8: **Pour** $k = n : -1 : j$ **faire**
9: $[c, s] = vlg(k, A(:, j))$;
% Mise à jour de la matrice A.
10: $vlA = A(k, co)$;

La suite de l'algorithme *SRDECO*

$A(k,co) = c \times A(k,co) + s \times A(n+k,co)$;
$A(n+k,co) = -s \times vlA + c \times A(n+k,co)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$vcS = S(:,k)$;
$S(:,k) = c \times S(:,k) + s \times S(:,n+k)$;
$S(:,n+k) = -s \times vcS + c \times S(:,n+k)$;
Fin Pour
% Calculer la constante β et le vecteur w de la matrice $H(j,w)$ de la transformation de Householder symplectique au sens de Van Loan dans le but d'annuler les entrées désirées dans la partie supérieure de la $j^{\text{ième}}$ colonne.
Si $j \leq n-1$ **alors**
$[\beta, w] = vlh(j, A(:,j))$;
% Mise à jour de la matrice A.
$A(j:n,co) = A(j:n,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j:n,co)$;
$A(j+n:den,co) = A(j+n:den,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+n:den,co)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$S(:,j:n) = S(:,j:n) - \beta \times S(:,j:n) \times (w \times w^{\mathrm{T}})$;
$S(:,n+j:den) = S(:,n+j:den) - \beta \times S(:,n+j:den) \times (w \times w^{\mathrm{T}})$;
Fin Si
% Mettre des zéros dans les positions souhaitées de la $(n+j)^{\text{ième}}$ colonne :
Si $j \leq n-1$ **alors**
% Calculer les constantes c et s de la matrice $J(k,c,s)$ de la rotation de Givens symplectique au sens de Van Loan dans l'intention d'annuler les entrées désirées dans la partie inférieure de la $(n+j)^{\text{ième}}$ colonne.
Pour $k = n:-1:j+1$ **faire**
$[c,s] = vlg(k, A(:,n+j))$;
% Mise à jour de la matrice A.
$vlA = A(k,co)$;
$A(k,co) = c \times A(k,co) + s \times A(n+k,co)$;
$A(n+k,co) = -s \times vlA + c \times A(n+k,co)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$vcS = S(:,k)$;
$S(:,k) = c \times S(:,k) + s \times S(:,n+k)$;
$S(:,n+k) = -s \times vcS + c \times S(:,n+k)$;
Fin Pour
Si $j \leq n-2$ **alors**
% Calculer la constante β et le vecteur w de la matrice $H(j+1,w)$ de la transformation de Householder symplectique au sens de Van Loan dans le but d'annuler les entrées désirées dans la partie supérieure de la $(n+j)^{\text{ième}}$ colonne.
$[\beta, w] = vlh(j+1, A(:,n+j))$;
% Mise à jour de la matrice A.
$A(j+1:n,co) = A(j+1:n,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1:n,co)$;
$A(j+1+n:den,co) = A(j+1+n:den,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1+n:den,co)$;

La suite de l'algorithme *SRDECO*

% Mise à jour de la matrice S implicitement (si nous avons besoin).

38: $S(:, j+1:n) = S(:, j+1:n) - \beta \times S(:, j+1:n) \times (w \times w^{\mathrm{T}})$;

39: $S(:, n+j+1:den) = S(:, n+j+1:den) - \beta \times S(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;

40: **Fin Si**

% Calculer les constantes c et d de la matrice $G(k,v)$ de la transformation de Gauss symplectique afin de mettre des zéros dans l'entrée $(j+1)$ de la $(n+j)^{\text{ième}}$ colonne.

41: **Si** $((A(j+1, n+j) \neq 0)$ et $(A(n+j, n+j) == 0))$ **alors**

42: Arrêter l'algorithme (il n'y a pas de décomposition SR pour A).

43: **Sinon**

44: $[c,d] = sgt(A(:, n+j), j+1)$;

% Mise à jour de la matrice A.

45: $A(j,:) = c \times A(j,:) + d \times A(n+j+1,:)$;

46: $A(j+1,:) = c \times A(j+1,:) + d \times A(n+j,:)$;

47: $A(n+j,:) = c^{-1} \times A(n+j,:)$;

48: $A(n+j+1,:) = c^{-1} \times A(n+j+1,:)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

49: $S(:, n+j) = -d \times S(:, j+1) + c \times S(:, n+j)$;

50: $S(:, n+j+1) = -d \times S(:, j) + c \times S(:, n+j+1)$;

51: $S(:, j) = c^{-1} \times S(:, j)$;

52: $S(:, j+1) = c^{-1} \times S(:, j+1)$;

53: **Fin Si**

54: **Fin Si**

55: **Fin Pour**

56: $R = A$

57: **Fin**

Dans cet algorithme *SRDECO* nous avons utilisé la transformation de Gauss symplectique sgt (Algorithme G) qui calcule les constantes c et d de la matrice $G(k,v)$ pour mettre un zéro dans l'entrée de la position $(j+1)$ dans la $(n+j)^{\text{ième}}$ colonne. L'algorithme sgt est donné par :

Algorithme 2.19 Algorithme de la transformation de Gauss symplectique sgt

Cet algorithme calcule les constantes c et d de la matrice $G(k,c,d) = G(k,v)$ de la transformation de Gauss symplectique avec

$$G(k,v) = \begin{pmatrix} I_{k-2} & & & & & & & & \\ & c & & & & & & d & \\ & & c & & & & d & & \\ & & & I_{n-k} & & & & & \\ & & & & I_{k-2} & & & & \\ & & & & & c^{-1} & & & \\ & & & & & & c^{-1} & & \\ & & & & & & & & I_{n-k} \end{pmatrix}$$

où étant donné k ($1 \leq k \leq n$) et un vecteur $a \in \mathbb{R}^{2n}$, où $a_{n+k-1} = 0$ si et seulement si $a_k = 0$, l'algorithme suivant détermine les constates c et d tel que si $G(k,c,d)\,a = b$ alors $b_k = 0$.

La suite de l'algorithme de la transformation de Gauss symplectique sgt

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$ et une constante k.
SORTIE(S) : Les constantes c et d de la matrice $G(k,c,d)$ de transformation de Gauss symplectique.

Fonction $[c,d] = sgt(a,k)$
$den = length(a)$;
$n = den/2$;
Si $((a(k) == 0)$ et $(a(n+k-1 == 0))$ **alors**
  Arrêter l'algorithme.
Sinon
  Si $(a(k) == 0)$ **alors**
    $\nu = 0$;
  Sinon
    $\nu = -a(k)/a(n+k-1)$;
  Fin Si
  $c = \sqrt[4]{1+\nu^2}$;
  $d = \nu \times c$;
  % Le conditionnement de la matrice $G(k,c,d)$ (si nous avons besoin).
  $\%\kappa = \sqrt{1+\nu^2} + |\nu|$;
Fin Si
Fin

Cet algorithme *SRDECO*, de la décomposition *SR* de Angelika Bunse-Gerstner et Volker Mehrmann [40], a une instabilité numérique. L'origine de cette instabilité est clairement identifiée : elle survient lorsqu'une transformation $G(k,\nu)$ de Gauss symplectique, qui n'est pas orthogonale, est rencontrée. En effet, quand la matrice $G(k,\nu)$ est appliquée pour éliminer l'entrée de la position $k^{\text{ième}}$ d'un vecteur a et si l'entrée a_{n+k-1} est très petite par rapport à l'entrée a_k, alors $\nu = -\dfrac{a_k}{a_{n+k-1}}$ devient très grande. Par conséquent, la matrice $G(k,\nu)$ devient mal-conditionnée sachant que son conditionnement s'écrit : $\kappa_2(G(k,\nu)) = \sqrt{1+\nu^2} + |\nu|$. Donc, l'algorithme *SRDECO* sera instable numériquement et même il échoue ; dans ce cas il n'aura pas de décomposition *SR* pour la matrice A.

Notre nouvel algorithme *SRMSH2* de la décomposition *SR* remédies aux instabilités numériques de l'algorithme *SRDECO*. En outre, nous pouvons voir l'algorithme *SRMSH2* comme une version modifiée de l'algorithme *SRDECO* dans lequel nous remplaçons les transformations de Gauss symplectiques par les transformations de Householder symplectiques de type deux.

2.8 Résultats numériques

Nous proposons ici quelques exemples numériques dans des situations différentes, ce qui nous permet de faire la comparaison entre les différents algorithmes. Nous comparons la perte de la J-orthogonalité et l'erreur absolue de la décomposition *SR*.

Exemple 2.8.1. *Dans cet exemple, nous prenons une matrice A de taille $2n \times 2n$ telle que $A = Pascal(2n)$. Puis, nous appliquons à cette dernière les algorithmes SRDECO, SRMSH, SRMSH2 et SROSH.*

$2n$	La perte de J-Orthogonalité $\|I - S^J S\|_2$			
	SRDECO	*SRMSH*	*SRMSH2*	*SROSH*
4	$4.9187e-16$	$3.0260e-16$	$2.7746e-16$	$2.5322e-16$
6	$1.1906e-15$	$8.7662e-16$	$1.7813e-15$	$7.3212e-16$
8	$1.8792e-15$	$1.8524e-15$	$1.7427e-15$	$2.9899e-14$
10	$2.0450e-15$	$2.0232e-15$	$5.2375e-15$	$7.0806e-14$
12	$5.2632e-15$	$8.6536e-15$	$1.9973e-14$	$1.4156e-08$
14	$1.1398e-14$	$6.7548e-14$	$4.7555e-14$	$1.6448e-11$
16	$5.1172e-14$	$1.1601e-13$	$1.0313e-13$	$7.3073e-10$
18	$2.4254e-14$	$3.8304e-13$	$6.7605e-13$	$1.8471e-09$

TABLEAU 2.10 – La perte de J-orthogonalité de la décomposition SR de la matrice $A = Pascal(2n)$ via les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH*

$2n$	L'erreur absolue de la réduction $\|A - SR\|_2$			
	SRDECO	*SRMSH*	*SRMSH2*	*SROSH*
4	$1.3146e-14$	$8.3477e-15$	$8.3768e-15$	$1.4865e-15$
6	$1.3215e-13$	$8.2041e-14$	$1.2978e-13$	$2.2680e-13$
8	$1.1740e-12$	$2.2313e-12$	$1.4626e-12$	$2.9295e-12$
10	$1.9155e-11$	$1.4932e-11$	$2.5801e-11$	$4.1395e-11$
12	$6.4563e-10$	$2.9852e-10$	$3.0539e-10$	$4.4348e-08$
14	$4.9791e-09$	$5.4522e-09$	$5.0939e-09$	$3.7373e-08$
16	$5.8830e-08$	$5.9826e-08$	$6.5018e-08$	$3.4990e-07$
18	$9.6741e-07$	$1.0267e-06$	$1.0815e-06$	$3.3385e-06$

TABLEAU 2.11 – L'erreur absolue de la décomposition SR de la matrice $A = Pascal(2n)$ via les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH*

Dans l'Exemple 2.8.1, nous remarquons que, pour la matrice $A = Pascal(2n)$, les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH* fournissent des résultats raisonnablement très semblables avec un petit désavantage pour l'algorithme *SROSH*. La perte de la J-orthogonalité et l'erreur dans la décomposition SR, pour les différents algorithmes, sont présentées dans le Tableau 2.10 et le Tableau 2.11, respectivement.

Exemple 2.8.2. *Dans cet exemple, nous prenons une matrice A de taille $2n \times 2n$ telle que $A = rand(2n)$. Puis, nous appliquons à cette dernière les algorithmes SRDECO, SRMSH, SRMSH2 et SROSH.*

$2n$	La perte de J-Orthogonalité $\|I - S^J S\|_2$			
	SRDECO	*SRMSH*	*SRMSH2*	*SROSH*
8	$1.2241e-15$	$1.7885e-15$	$2.9616e-15$	$8.2483e-15$
10	$8.4262e-15$	$5.9949e-14$	$8.2671e-14$	$2.5664e-13$
12	$5.5863e-15$	$3.3405e-14$	$4.4051e-14$	$2.3591e-11$
14	$3.1964e-14$	$8.0824e-13$	$3.9842e-13$	$2.3202e-12$
16	$1.1135e-14$	$2.5081e-14$	$3.2348e-14$	$3.1093e-12$
18	$2.6492e-14$	$1.9372e-12$	$7.0526e-13$	$1.3444e-11$
20	$1.6493e-14$	$4.5311e-13$	$8.3591e-13$	$9.5135e-10$
22	$6.4256e-14$	$1.8804e-12$	$4.3414e-12$	$1.5161e-09$

TABLEAU 2.12 – La perte de J-orthogonalité de la décomposition SR de la matrice $A = rand(2n)$ via les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH*

$2n$	L'erreur absolue de la réduction $\|A - SR\|_2$			
	SRDECO	*SRMSH*	*SRMSH2*	*SROSH*
8	$1.6499e-15$	$2.4184e-15$	$1.8322e-15$	$4.0711e-15$
10	$3.5192e-15$	$6.1972e-15$	$6.3972e-15$	$9.2264e-14$
12	$1.2944e-14$	$1.3647e-14$	$8.7209e-15$	$1.4552e-12$
14	$1.2887e-14$	$8.2082e-14$	$4.6269e-14$	$4.9120e-13$
16	$3.4408e-15$	$4.6350e-15$	$8.6635e-15$	$1.6625e-12$
18	$1.1579e-14$	$1.6906e-13$	$2.5665e-14$	$3.5581e-12$
20	$1.8281e-14$	$2.0731e-13$	$1.6150e-14$	$7.4008e-09$
22	$3.9904e-14$	$2.8081e-13$	$6.5138e-14$	$2.7094e-10$

TABLEAU 2.13 – L'erreur absolue de la décomposition SR de la matrice $A = rand(2n)$ via les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH*

Dans l'Exemple 2.8.2, nous remarquons aussi que pour la matrice $A = rand(2n)$ les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH* fournissent des résultats raisonnablement très semblables avec un petit désavantage pour l'algorithme *SROSH*. La perte de la J-orthogonalité et l'erreur dans la décomposition SR, pour les différents algorithmes, sont présentées dans le Tableau 2.12 et le Tableau 2.13, respectivement.

Exemple 2.8.3. *Dans cet exemple, nous prenons une matrice A de taille $2n \times 2n$ telle que*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}.$$

Sachant que chaque bloc A_{ij} est de taille $n \times n$ avec

$$A_{11} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}, A_{12} = \begin{pmatrix} 1 & & & & \\ 0.01 & 1 & & & \\ & \ddots & \ddots & & \\ & & 0.01 & 1 & \\ & & & 0.01 & 1 \end{pmatrix}, A_{21} = \begin{pmatrix} 1 & & & & \\ 1 & 1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & 1 & \\ & & & 1 & 1 \end{pmatrix}$$

et $A_{22} = \begin{pmatrix} e^{\frac{1}{2}} & & & & \\ & e^{\frac{2}{2}} & & & \\ & & \ddots & & \\ & & & e^{\frac{n-1}{2}} & \\ & & & & e^{\frac{n}{2}} \end{pmatrix}$.

Puis, nous appliquons à cette matrice A les algorithmes SRDECO, SRMSH, SRMSH2 et SROSH.

$2n$	La perte de J-Orthogonalité $\|I - S^J S\|_2$			
	SRDECO	*SRMSH*	*SRMSH2*	*SROSH*
20	$3.0065e-15$	$4.7320e-15$	$5.9728e-15$	$2.0241e-15$
22	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$1.3031e-15$
24	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$1.3031e-15$
26	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$1.3031e-15$
28	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$2.6935e-15$
30	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$2.8098e-15$
32	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$5.2430e-15$
34	$1.5726e-15$	$3.1068e-15$	$2.3385e-12$	$4.6879e-15$
36	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$3.5034e-14$
38	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$3.4659e-14$
40	$1.5726e-15$	$3.1068e-15$	$5.4974e-15$	$8.9884e-14$

TABLEAU 2.14 – La perte de J-orthogonalité de la décomposition SR de la matrice A via les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH*

Dans l'Exemple 2.8.3, nous remarquons aussi que pour la matrice A les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH* fournissent des résultats raisonnablement très semblables. La perte de la J-orthogonalité et l'erreur dans la décomposition SR, pour les

$2n$	L'erreur absolue de la réduction $\|A - SR\|_2$			
	SRDECO	*SRMSH*	*SRMSH2*	*SROSH*
20	$3.4335e-14$	$2.6720e-14$	$2.2870e-14$	$1.5017e-14$
22	$5.0466e-14$	$8.0533e-14$	$1.0527e-13$	$3.7174e-14$
24	$1.7501e-13$	$1.4238e-13$	$1.9510e-13$	$1.8580e-13$
26	$3.0157e-13$	$2.5726e-13$	$2.4957e-13$	$2.4646e-13$
28	$4.9370e-13$	$4.4890e-13$	$4.1654e-13$	$9.3066e-13$
30	$5.9785e-13$	$6.0880e-13$	$6.0874e-13$	$7.7500e-13$
32	$1.2407e-12$	$1.1383e-12$	$1.4439e-12$	$1.0314e-10$
34	$2.0213e-12$	$2.3350e-12$	$2.3385e-12$	$1.0412e-10$
36	$2.3328e-12$	$3.6446e-12$	$5.0825e-12$	$7.0664e-10$
38	$3.3066e-12$	$4.3147e-12$	$7.1038e-12$	$6.9971e-10$
40	$4.9845e-12$	$1.0552e-11$	$8.2649e-12$	$7.0307e-10$

TABLEAU 2.15 – L'erreur absolue de la décomposition SR de la matrice A via les algorithmes *SRDECO*, *SRMSH*, *SRMSH2* et *SROSH*

différents algorithmes, sont présentées dans le Tableau 2.14 et le Tableau 2.15, respectivement.

Nous Remarquons aussi que l'injection des transformations de Givens symplectiques et les transformations de Householder symplectiques au sens de Van Loan avant l'application des transformations de Householder symplectiques optimale $osh2$ dans l'algorithme *SRMSH2* ne fait que renforcer la précision par rapport à l'algorithme *SRMSH*.

2.9 Étude d'erreur du MSGS par l'équivalence entre MSGS et SR via Householder symplectique

Dans le cas Euclidien, Åke Björck, Christopher C. Paige dans leur article [30] ont montré que la décomposition QR d'une matrice A de taille $m \times n$ via l'algorithme de Gram–Schmidt modifié *MGS* est numériquement équivalente à celle résultant des transformations du Householder appliquées à la matrice A augmentée par une matrice carrée des zéros de taille $n \times n$ en haut. Ce n'est pas seulement vrai en théorie, mais aussi en présence des erreurs d'arrondi. Cette observation fascinante est à l'origine due à Charles Sheffield et a été communiquée par Gene Golub. Ceci est expliqué d'une manière claire et simple dans [30], puis combiné avec un résultat d'erreur d'arrondissement bien connu pour montrer que la matrice triangulaire supérieure R de l'algorithme *MGS* est aussi précise que la matrice R à partir de l'autre décomposition QR via les transformations du Householder. La structure spéciale du produit des transformations du Householder est

dérivée, puis utilisée pour expliquer et majorer la perte d'orthogonalité dans l'algorithme *MGS*. Enfin, cette équivalence numérique est utilisée pour montrer comment la perte d'orthogonalité dans l'algorithme *MGS* peut être retrouvée en général.

Dans le cas symplectique, A. Salam et E. Al-Aidarous dans [101] ont montré que la décomposition *SR* d'une matrice A via l'algorithme de Gram–Schmidt symplectique modifié *MSGS* est mathématiquement et numériquement équivalente à la décomposition *SR* via les transformations du Householder symplectiques (Algorithme *SRSH*) appliquées à une matrice augmentée. Cette matrice est obtenue à partir de la matrice A en ajoutant deux blocs de zéros au sommet de la première moitié et au sommet de la seconde moitié de la matrice A.

Dans ce paragraphe, nous prévoyons utiliser ces résultats importants pour la dérivation d'une analyse d'erreur de l'algorithme *MSGS* (en se basant sur l'équivalence entre les algorithmes *MSGS* et *SRSH*). En fait, les tentatives directes pour établir l'analyse des erreurs d'arrondi de l'algorithme *MSGS* ont été révélées très difficiles à réaliser [107]. Du coup, nous nous attendons à ce que l'analyse de l'erreur d'arrondi de l'algorithme *MSGS* devienne plus accessible pour l'étudier. En outre, nous affirmons que ces résultats seraient utiles pour dériver élégamment et plus facilement des majorations plus fines que celles dans [107], et aussi pour établir que la perte de J-orthogonalité produite lors de l'application de l'algorithme *MSGS* se produit d'une manière prévisible.

Soit la matrice

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \in \mathbb{R}^{2m\times 2n},$$

avec les blocs $A_{ij} \in \mathbb{R}^{m\times n}$ pour $i, j \in \{1,2\}$.

La matrice A est augmentée avec un premier bloc de deux matrices carrées nulles au-dessus de la première moitié de la matrice A et un deuxième bloc de deux matrices carrées nulles au-dessus de la seconde moitié de la matrice A tel que :

$$A' = [a'_1, \ldots, a'_n, a'_{n+1}, \ldots, a'_{2n}] = \begin{pmatrix} 0_{nn} & 0_{nn} \\ A_{11} & A_{12} \\ 0_{nn} & 0_{nn} \\ A_{21} & A_{22} \end{pmatrix} \in \mathbb{R}^{2(m+n)\times 2n}.$$

L'algorithme de la décomposition *SR* via les transformations du Householder symplectiques à la matrice A' effectue les mêmes calculs et dans le même ordre que ceux de l'algorithme de Gram–Schmidt symplectique modifié *MSGS*. Nous rappelons aussi que cet algorithme consiste à construire des transformations du Householder symplectiques $P_1, P_2, \ldots, P_n$ de telle sorte que la matrice $P^J = P_n \ldots P_2 P_1$ transforme la matrice A' comme suit :

$$P^J A' = P_n \ldots P_2 P_1 \begin{pmatrix} 0_{nn} & 0_{nn} \\ A_{11} & A_{12} \\ 0_{nn} & 0_{nn} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} \\ 0_{mn} & 0_{mn} \\ R_{21} & R_{22} \\ 0_{mn} & 0_{mn} \end{pmatrix} = R' \in \mathbb{R}^{2(m+n)\times 2n}.$$

Les matrices $R_{ij} \in \mathbb{R}^{n\times n}$ pour $i, j \in \{1,2\}$ sont des matrices triangulaires supérieures et la matrice R_{21} est une matrice strictement triangulaire supérieure. Les transformations

du Householder symplectiques ont la forme $P_k = I + W_k \Sigma_k^J W_k^J$ avec $W_k \in \mathbb{R}^{2(m+n)\times 2}$ et $\Sigma_k^J \in \mathbb{R}^{2\times 2}$ en satisfaisant la condition

$$\text{rang}(W_k) = 1 \text{ ou } W_k^J W_k \det(\Sigma_k) + \text{trace}(\Sigma_k) = 0.$$

Nous pouvons voir cette transformation P_k comme le produit de deux transformations du Householder symplectiques, $T_1 = I + c_1 v_1 v_1^J$ et $T_2 = I + c_2 v_2 v_2^J$ avec $c_j \in \mathbb{R}$ et $v_j \in \mathbb{R}^{2(m+n)}$ pour $j \in \{1,2\}$. Nous avons

$$T_1 T_2 = I + W \Sigma^J W^J,$$

où $W = [v_1, v_2] \in \mathbb{R}^{2(m+n)\times 2}$ et $\Sigma^J = \begin{pmatrix} 0 & c_1 \\ -c_2 & -c_1 c_2 v_1^J v_2 \end{pmatrix} \in \mathbb{R}^{2\times 2}$.

En conséquence, dans la $k^{\text{ième}}$ étape, nous pouvons réécrire la transformation P_k du Householder symplectique comme suit :

$$P_k = I - U_k^{'} \times U_k^{'J},$$

ou bien pour la sous matrice $\tilde{A}$ (voir les étapes de l'algorithme *SRSH* ou *SROSH*) de la façon suivante :

$$\tilde{P}_k = I - \tilde{U}_k^{'} \times \tilde{U}_k^{'J},$$

sachant que

$$U_k^{'} = \begin{bmatrix} -e_k & 0 \\ v_k^u & v_{k+n}^u \\ 0 & -e_k \\ v_k^l & v_{k+n}^l \end{bmatrix}, \quad \tilde{U}_k^{'} = \begin{bmatrix} -e_1^{(k)} & 0 \\ v_k^u & v_{k+n}^u \\ 0 & -e_1^{(k)} \\ v_k^l & v_{k+n}^l \end{bmatrix} \quad \text{et} \quad U_k = \begin{bmatrix} v_k & v_{k+n} \end{bmatrix}.$$

Le vecteur $e_1^{(k)}$ représente le premier vecteur canonique de $\mathbb{R}^{n-k+1}$ et le vecteur e_k représente le $k^{\text{ième}}$ vecteur canonique de $\mathbb{R}^n$. Et pour $j = k+1, \ldots, n$, nous avons

$$\begin{matrix} 1 \\ n-k \\ m \\ 1 \\ n-k \\ m \end{matrix} \begin{pmatrix} r_{k,j} & r_{k,n+j} \\ 0 & 0 \\ a_j^{(k+1)^u} & a_{n+j}^{(k+1)^u} \\ r_{n+k,j} & r_{n+k,n+j} \\ 0 & 0 \\ a_j^{(k+1)^l} & a_{n+j}^{(k+1)^l} \end{pmatrix} = \tilde{P}_k \begin{pmatrix} 0 & 0 \\ a_j^{(k)^u} & a_{n+j}^{(k)^u} \\ 0 & 0 \\ a_j^{(k)^l} & a_{n+j}^{(k)^l} \end{pmatrix} \begin{matrix} n-k+1 \\ m \\ n-k+1 \\ m \end{matrix}. \tag{2.7}$$

Les matrices de deux transformations de Householder symplectiques P_k et $\tilde{P}_k$ sont anti-Hamiltoniennes, c'est-à-dire $P_k^J = P_k$ et $\tilde{P}_k^J = \tilde{P}_k$. Cette propriété correspond, dans le cas Euclidien, au fait que les transformations de Householder sont symétriques. En outre, les matrices $U_k^{'}$ et U_k satisfont :

$$U_k^{'J} U_k^{'} = 2I_2, \quad U_k^J U_k = I_2 \quad \text{et} \quad U_k^{'J} U_l^{'} = U_l^{'J} U_k^{'} = 0 \quad \text{pour} \quad k \neq l.$$

Ainsi, nous avons :

$$P_k = I - \begin{bmatrix} -e_k & 0 \\ v_k^u & v_{k+n}^u \\ 0 & -e_k \\ v_k^l & v_{k+n}^l \end{bmatrix} \times \begin{bmatrix} -e_k^{\mathrm{T}} & v_{k+n}^{l\,\mathrm{T}} & 0 & -v_{k+n}^{u\,\mathrm{T}} \\ 0 & -v_k^{l\,\mathrm{T}} & -e_k^{\mathrm{T}} & v_k^{u\,\mathrm{T}} \end{bmatrix}.$$

Nous pouvons voir la relation (2.7) comme suit :

$$\begin{bmatrix} r_{k,j} & r_{k,n+j} \\ r_{n+k,j} & r_{n+k,n+j} \end{bmatrix} = U_k^J a_j^{(k)} \quad \text{et} \quad \begin{bmatrix} a_j^{(k+1)} & a_{j+n}^{(k+1)} \end{bmatrix} = T_k \begin{bmatrix} a_j^{(k)} & a_{j+n}^{(k)} \end{bmatrix},$$

pour $j = k+1, \ldots, n$ avec $T_k = I - U_k \times U_k^J$.

Théorème 2.9.1. *Soit la matrice* $E_k = \begin{bmatrix} e_k & 0 \\ 0 & e_k \end{bmatrix}$*, avec le vecteur* e_k *représente le* $k^{ième}$ *vecteur canonique de* $\mathbb{R}^n$*. Soit* Z *la matrice de permutation définie par*

$$Z = \begin{pmatrix} I_n & 0 & 0 & 0 \\ 0 & 0 & I_m & 0 \\ 0 & I_n & 0 & 0 \\ 0 & 0 & 0 & I_m \end{pmatrix}.$$

Alors, nous pouvons réécrire la matrice P *de la façons suivante :*

$$P = P_1 P_2 \ldots P_{n-1} P_n = I - \sum_{k=1}^{n} U_k^{'} U_k^{'J},$$

et nous avons

$$Z^{\mathrm{T}} P Z = \begin{pmatrix} 0 & S^J \\ S & I - SS^J \end{pmatrix}.$$

Preuve. En effet, comme la matrice Z est orthogonale, alors nous avons

$$Z^{\mathrm{T}} P_k Z = I - Z^{\mathrm{T}} U_k^{'} U_k^{'J} Z.$$

Sachant que $E_k = \begin{bmatrix} e_k & 0 \\ 0 & e_k \end{bmatrix}$, $E_k^J = E_k^{\mathrm{T}} \begin{bmatrix} e_k^{\mathrm{T}} & 0 \\ 0 & e_k^{\mathrm{T}} \end{bmatrix}$, alors $E_k^J E_k = I$, $E_k^J E_j = 0$ avec $k \neq j$.
D'autre part,

$$U_k^{'J} Z = \begin{bmatrix} -e_k^{\mathrm{T}} & 0 & v_{k+n}^{l\,\mathrm{T}} & -v_{k+n}^{u\,\mathrm{T}} \\ 0 & -e_k^{\mathrm{T}} & -v_k^{l\,\mathrm{T}} & v_k^{u\,\mathrm{T}} \end{bmatrix} = \begin{bmatrix} -E_k^J & U_k^J \end{bmatrix}.$$

De plus,

$$Z^{\mathrm{T}} U_k^{'} = \begin{bmatrix} -e_k & 0 \\ 0 & -e_k \\ v_k^u & v_{k+n}^u \\ v_k^l & v_{k+n}^l \end{bmatrix} = \begin{bmatrix} -E_k \\ U_k \end{bmatrix}.$$

Par suite, nous aurons

$$\begin{aligned} Z^{\mathrm{T}} P_k Z &= Z^{\mathrm{T}}(I - U_k^{'} U_k^{'J}) Z \\ &= I - Z^{\mathrm{T}} U_k^{'} U_k^{'J} Z \\ &= I - \begin{bmatrix} -E_k \\ U_k \end{bmatrix} \times \begin{bmatrix} -E_k^J & U_k^J \end{bmatrix} \\ &= I - \begin{pmatrix} E_k E_k^J & -E_k U_k^J \\ -U_k E_k^J & U_k U_k^J \end{pmatrix} \\ &= \begin{pmatrix} I - E_k E_k^J & E_k U_k^J \\ U_k E_k^J & I - U_k U_k^J \end{pmatrix}. \end{aligned}$$

Donc,

$$\begin{aligned} Z^{\mathrm{T}} P Z &= Z^{\mathrm{T}}(I - \sum_{k=1}^{n} U_k^{'} U_k^{'J}) Z \\ &= I - \sum_{k=1}^{n} Z^{\mathrm{T}} U_k^{'} U_k^{'J} Z \\ &= I - \sum_{k=1}^{n} \begin{pmatrix} E_k E_k^J & -E_k U_k^J \\ -U_k E_k^J & U_k U_k^J \end{pmatrix} \\ &= \begin{pmatrix} I - \sum_{k=1}^{n} E_k E_k^J & \sum_{k=1}^{n} E_k U_k^J \\ \sum_{k=1}^{n} U_k E_k^J & I - \sum_{k=1}^{n} U_k U_k^J \end{pmatrix}. \end{aligned}$$

Comme $\sum_{k=1}^{n} E_k E_k^J = I, \sum_{k=1}^{n} U_k E_k^J = S, \sum_{k=1}^{n} E_k U_k^J = S^J$ et $\sum_{k=1}^{n} U_k U_k^J = SS^J$, alors nous aurons

$$Z^{\mathrm{T}} P Z = \begin{pmatrix} 0 & S^J \\ S & I - SS^J \end{pmatrix}.$$

□

Du coup pour résumer, soit la matrice $A \in \mathbb{R}^{2m \times 2n}$ tel que $\mathrm{rang}(A) = 2n$. Soit la matrice augmentée $A' \in \mathbb{R}^{2(m+n) \times 2n}$ qui est obtenue par l'addition d'un premier bloc de deux matrices carrées nulles $0_{nn} \in \mathbb{R}^{n \times n}$ au-dessus de la première moitié de la matrice A et d'un deuxième bloc de deux matrices carrées nulles $0_{nn} \in \mathbb{R}^{n \times n}$ au-dessus de la seconde moitié de la matrice A. Nous considérons deux décompositions SR, qui sont équivalentes : la décomposition SR via les transformations du Householder symplectiques pour la matrice A' et l'autre décomposition SR via Gram–Schmidt symplectique modifié $MSGS$ pour la matrice A tels que les matrices symplectiques $P \in \mathbb{R}^{2(m+n) \times 2(m+n)}$ et $S_1 \in \mathbb{R}^{2m \times 2m}$ avec $S \in \mathbb{R}^{2m \times 2n}$.

$$A = S_1 \begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} S & \dot{S} \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix} = SR.$$

$$A' = PR' \iff \begin{pmatrix} 0 \\ A \end{pmatrix} = Z^{\mathrm{T}} A' = Z^{\mathrm{T}} P (Z Z^{\mathrm{T}}) R' = Z^{\mathrm{T}} P Z \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}.$$

Comme ça, nous aurons

$$\begin{pmatrix}0\\A\end{pmatrix} = Z^{\mathrm{T}} P Z \begin{pmatrix}\tilde{R}\\0\end{pmatrix} = \begin{pmatrix}P_{11} & P_{12}\\P_{21} & P_{22}\end{pmatrix}\begin{pmatrix}\tilde{R}\\0\end{pmatrix} = \begin{pmatrix}P_{11}\tilde{R}\\P_{21}\tilde{R}\end{pmatrix}.$$

Puisque la matrice rang$(A) = 2n$, alors la matrice P_{11} est une matrice nulle, la matrice $P_{21} \in \mathbb{R}^{2m\times 2n}$ est une matrice symplectique et nous avons

$$A = SR = P_{21}\tilde{R},$$

$$A^J A = R^J R = \tilde{R}^J \tilde{R}.$$

Par l'équivalence entre les deux décompositions SR et par analogie, nous avons alors $R = \tilde{R}$ et $S_1 = P_{21}$.

Dans ce que suit, en utilisant l'équivalence entre la décomposition SR d'une matrice $A \in \mathbb{R}^{2m\times 2n}$ via l'algorithme de Gram–Schmidt symplectique modifié $MSGS$ et la décomposition SR via les transformations du Householder symplectiques appliquées à une matrice augmentée $A' \in \mathbb{R}^{2(m+n)\times 2n}$, nous présentons un nouveau théorème sur l'analyse de l'erreur d'arrondi de l'algorithme $MSGS$ ainsi que la perte de la J-orthogonalité produite lors de l'application cet algorithme.

Théorème 2.9.2. *Soit la matrice* $A = \begin{pmatrix}A_{11} & A_{12}\\A_{21} & A_{22}\end{pmatrix} \in \mathbb{R}^{2m\times 2n}$. *Nous augmentons cette matrice en ajoutant deux blocs de zéros au sommet de la première moitié et au sommet de la seconde moitié de la matrice A pour avoir la matrice augmentée*

$$A' = \begin{pmatrix}0_{nn} & 0_{nn}\\A_{11} & A_{12}\\0_{nn} & 0_{nn}\\A_{21} & A_{22}\end{pmatrix} \in \mathbb{R}^{2(m+n)\times 2n}.$$

Soit $Z \in \mathbb{R}^{2(m+n)\times 2(m+n)}$ *une matrice de permutation définie par :*

$$Z = \begin{pmatrix}I_n & 0 & 0 & 0\\0 & 0 & I_m & 0\\0 & I_n & 0 & 0\\0 & 0 & 0 & I_m\end{pmatrix}.$$

Alors nous avons :

$$\begin{pmatrix}E_1\\A+E_2\end{pmatrix} = Z^{\mathrm{T}}\bar{A}' = Z^{\mathrm{T}} P Z \begin{pmatrix}\bar{R}\\0\end{pmatrix},$$

$$\bar{P} = \begin{pmatrix}P_{11} & (I-P_{11})\bar{S}^J\\\bar{S}(I-P_{11}) & I-\bar{S}(I-P_{11})\bar{S}^J\end{pmatrix},$$

$$\bar{P} = P + E.$$

Tels que :

$$\|E_1\|_2 \leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}\|A\|_2 \prod_{i=1}^{m}\kappa_2(T_i), \tag{2.8}$$

$$\|E_2\|_2 \leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}\left(1+\frac{1}{1-3m\tilde{\gamma}_n}\right)\|A\|_2\prod_{i=1}^{m}\kappa_2(T_i), \tag{2.9}$$

$$\|\bar{P}\|_2 \leq 1+\frac{1}{1-3m\tilde{\gamma}_n}\left(2+\left(\frac{1}{1-3m\tilde{\gamma}_n}+3m\tilde{\gamma}_n k_2(A)\right)\prod_{i=1}^{m}\|T_i\|_2\right)\prod_{i=1}^{m}\|T_i\|_2, \tag{2.10}$$

$$\|E\|_2 \leq \|E_1'\|_2+\|E_2'\|_2, \tag{2.11}$$

$$\|E_1'\|_2 \leq 2\frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}k_2(A)\prod_{i=1}^{m}k_2(T_i)\left(1+\prod_{i=1}^{m}\|T_i\|_2\right)^2, \tag{2.12}$$

$$\|E_2'\|_2 \leq \left(2+\frac{2-3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}\prod_{i=1}^{m}\|T_i\|_2\right)\frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}\prod_{i=1}^{m}\|T_i\|_2, \tag{2.13}$$

$$\|P_{11}\|_2 \leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}k_2(A)\prod_{i=1}^{m}k_2(T_i). \tag{2.14}$$

Où nous désignons $A = SR$ la décomposition SR de la matrice A via MSGS, $A' = PR'$ la décomposition SR de la matrice A' via les transformations du Householder symplectiques et par les matrices $\bar{S}$, $\bar{R}$ et $\bar{P}$ les versions calculées des matrices S, R, P respectivement (c'est-à-dire d'une manière générale $\bar{M} = M + \Delta M$). Nous utilisons la notation

$$\gamma_n \leq \frac{cku}{1-cku},$$

(voir [63, p. 63]) avec c est le plus petit entier constant et u la précision machine tel que $cku \leq 1$ où $k \in \mathbb{N}^*$.

Preuve. En effet, nous avons

$$A' = PR',$$

$$A' = \begin{pmatrix} 0_{nn} & 0_{nn} \\ A_{11} & A_{12} \\ 0_{nn} & 0_{nn} \\ A_{21} & A_{22} \end{pmatrix} = P\begin{pmatrix} R_{11} & R_{12} \\ 0_{mn} & 0_{mn} \\ R_{21} & R_{22} \\ 0_{mn} & 0_{mn} \end{pmatrix}.$$

Ainsi,

$$Z^{\mathrm{T}}A' = Z^{\mathrm{T}}\begin{pmatrix} 0_{nn} & 0_{nn} \\ A_{11} & A_{12} \\ 0_{nn} & 0_{nn} \\ A_{21} & A_{22} \end{pmatrix} = Z^{\mathrm{T}}P\begin{pmatrix} R_{11} & R_{12} \\ 0_{mn} & 0_{mn} \\ R_{21} & R_{22} \\ 0_{mn} & 0_{mn} \end{pmatrix}$$

$$= (Z^{\mathrm{T}}PZ)(Z^{\mathrm{T}}\begin{pmatrix} R_{11} & R_{12} \\ 0_{mn} & 0_{mn} \\ R_{21} & R_{22} \\ 0_{mn} & 0_{mn} \end{pmatrix}).$$

De plus, nous avons d'une part

$$Z^{\mathrm{T}}R' = Z^{\mathrm{T}}\begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \\ R_{21} & R_{22} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

et de l'autre part nous avons

$$\begin{aligned} Z^{\mathrm{T}}\begin{pmatrix} 0 & 0 \\ A_{11} & A_{12} \\ 0 & 0 \\ A_{21} & A_{22} \end{pmatrix} &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \\ &= \begin{pmatrix} 0 & S^J \\ S & I - SS^J \end{pmatrix} Z^{\mathrm{T}}\begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \\ R_{21} & R_{22} \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & S^J \\ S & I - SS^J \end{pmatrix}\begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}. \end{aligned}$$

Donc, nous pouvons écrire

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 0 & S^J \\ S & I - SS^J \end{pmatrix}\begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

D'où,

$$\begin{pmatrix} 0 \\ A \end{pmatrix} = Z^{\mathrm{T}}A' = Z^{\mathrm{T}}PZ\begin{pmatrix} R \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & S^J \\ S & I - SS^J \end{pmatrix}\begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Ainsi, par analogie avec l'étude d'erreur dans le cas Euclidien, nous avons

$$\begin{pmatrix} E_1 \\ A + E_2 \end{pmatrix} = Z^{\mathrm{T}}\bar{A}' = Z^{\mathrm{T}}PZ\begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}.$$

Par suite, nous rappelons la matrice $Z^{\mathrm{T}}P_kZ$ où la matrice P_k est la matrice de la transformation du Householder symplectique appliquée à la $k^{\text{ième}}$ étape de la décomposition

SR, de telle sorte que $P_k^J P_k = I$.

$$\begin{aligned} Z^{\mathrm{T}} P_k Z = I - Z^{\mathrm{T}} U_k^{'} U_k^{'J} Z &= I - \begin{bmatrix} -E_k \\ U_k \end{bmatrix} \begin{bmatrix} -E_k^J & U_k^J \end{bmatrix} \\ &= I - \begin{pmatrix} E_k E_k^J & -E_k U_k^J \\ -U_k E_k^J & U_k U_k^J \end{pmatrix} \\ &= \begin{pmatrix} I - E_k E_k^J & E_k U_k^J \\ U_k E_k^J & I - U_k U_k^J \end{pmatrix} \\ &= \begin{pmatrix} I - E_k E_k^J & E_k U_k^J \\ U_k E_k^J & T_k \end{pmatrix}. \end{aligned}$$

Alors, nous avons

$$Z^{\mathrm{T}} P_k Z \begin{pmatrix} E_j \\ 0 \end{pmatrix} = \begin{pmatrix} I - E_k E_k^J & E_k U_k^J \\ U_k E_k^J & I - U_k U_k^J \end{pmatrix} \begin{pmatrix} E_j \\ 0 \end{pmatrix} = \begin{pmatrix} E_j \\ 0 \end{pmatrix}.$$

De même,

$$Z^{\mathrm{T}} P_k Z \begin{pmatrix} E_k \\ 0 \end{pmatrix} = \begin{pmatrix} I - E_k E_k^J & E_k U_k^J \\ U_k E_k^J & I - U_k U_k^J \end{pmatrix} \begin{pmatrix} E_k \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ U_k \end{pmatrix} = \begin{pmatrix} 0 \\ I \end{pmatrix} U_k.$$

Nous adapterons la notation suivante :

$$\begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} = Z^{\mathrm{T}} P Z = Z^{\mathrm{T}} P_1 P_2 \dots P_{n-1} P_n Z.$$

Soit $1 \leq j \leq n$, si $j \neq k$ alors nous avons

$$Z^{\mathrm{T}} P_k Z \begin{pmatrix} E_j \\ 0 \end{pmatrix} = \begin{pmatrix} E_j \\ 0 \end{pmatrix}, \text{ bien que } Z^{\mathrm{T}} P_k Z \begin{pmatrix} E_k \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ U_k \end{pmatrix} = \begin{pmatrix} 0 \\ I \end{pmatrix} U_k.$$

Alors,

$$
\begin{aligned}
\begin{pmatrix} P_{11} \\ P_{21} \end{pmatrix} E_j &= \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} E_j \\ 0 \end{pmatrix} \\
&= Z^{\mathrm{T}} P Z \begin{pmatrix} E_j \\ 0 \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 \dots P_j \dots P_n Z \begin{pmatrix} E_j \\ 0 \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 \dots P_j Z \begin{pmatrix} E_j \\ 0 \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 \dots P_{j-1} Z Z^{\mathrm{T}} P_j Z \begin{pmatrix} E_j \\ 0 \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 \dots P_{j-1} Z \begin{pmatrix} 0 \\ I \end{pmatrix} U_k \\
&= Z^{\mathrm{T}} P_1 \dots P_{j-2} Z Z^{\mathrm{T}} P_{j-1} Z \begin{pmatrix} 0 \\ I \end{pmatrix} U_k \\
&= Z^{\mathrm{T}} P_1 \dots P_{j-2} Z \begin{pmatrix} E_{j-1} U_{j-1}^J \\ T_{j-1} \end{pmatrix} U_j \\
&= Z^{\mathrm{T}} P_1 \dots P_{j-3} Z Z^{\mathrm{T}} P_{j-2} Z \begin{pmatrix} E_{j-1} U_{j-1}^J \\ T_{j-1} \end{pmatrix} U_j \\
&= Z^{\mathrm{T}} P_1 \dots P_{j-3} Z \begin{pmatrix} I - E_{j-2} E_{j-2}^J & E_{j-2} U_{j-2}^J \\ U_{j-2} E_{j-2}^J & T_{j-2} \end{pmatrix} \begin{pmatrix} E_{j-1} U_{j-1}^J \\ T_{j-1} \end{pmatrix} U_j
\end{aligned}
$$

$$
\begin{aligned}
&= Z^{\mathrm{T}} P_1 \dots P_{j-3} Z \begin{pmatrix} E_{j-1} U_{j-1}^J + E_{j-2} U_{j-2}^J T_{j-1} \\ T_{j-2} T_{j-1} \end{pmatrix} U_j \\
&= \quad \vdots \\
&= \begin{pmatrix} \sum_{i=1}^{j-1} (E_i U_i^J T_{i+1} \cdots T_{j-1} U_j) \\ T_1 \cdots T_{j-2} T_{j-1} U_j \end{pmatrix} \\
&= \begin{pmatrix} U_1^J T_2 \cdots T_{j-1} U_j \\ U_2^J T_3 \cdots T_{j-1} U_j \\ \vdots \\ U_{j-2}^J T_{j-1} U_j \\ U_{j-1}^J U_j^J \\ 0 \\ \vdots \\ 0 \\ T_1 T_2 \cdots T_{j-1} U_j \end{pmatrix} = \begin{pmatrix} \dfrac{P_{11}}{P_{21}} \end{pmatrix} E_j = \begin{pmatrix} \dfrac{p_{11}}{p_{21}} \end{pmatrix} \equiv \begin{pmatrix} \pi_{1j} \\ \pi_{2j} \\ \vdots \\ \pi_{j-2,j} \\ \pi_{j-1,j} \\ \pi_{jj} \\ \vdots \\ \dfrac{\pi_{nj}}{p_{2j}} \end{pmatrix}.
\end{aligned}
$$

Ce qui nous donne les blocs P_{11} et P_{21} de la matrice $Z^{\mathrm{T}}PZ = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}$.

De plus,

$$\begin{aligned}
\begin{pmatrix} P_{12} \\ P_{22} \end{pmatrix} &= \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} 0 \\ I \end{pmatrix} \\
&= Z^{\mathrm{T}} P Z \begin{pmatrix} 0 \\ I \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 P_2 \dots P_{n-1} P_n Z \begin{pmatrix} 0 \\ I \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 P_2 \dots P_{n-1} Z Z^{\mathrm{T}} P_n Z \begin{pmatrix} 0 \\ I \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 P_2 \dots P_{n-1} Z \begin{pmatrix} E_n U_n^J \\ T_n \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 P_2 \dots P_{n-2} Z Z^{\mathrm{T}} P_{n-1} Z \begin{pmatrix} E_n U_n^J \\ T_n \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 P_2 \dots P_{n-2} Z \begin{pmatrix} I - E_{n-1} E_{n-1}^J & E_{n-1} U_{n-1}^J \\ U_{n-1} E_{n-1}^J & I - U_{n-1} U_{n-1}^J \end{pmatrix} \begin{pmatrix} E_n U_n^J \\ T_n \end{pmatrix} \\
&= Z^{\mathrm{T}} P_1 P_2 \dots P_{n-2} Z \begin{pmatrix} E_n U_n^J + E_{n-1} U_{n-1}^J T_n \\ T_{n-1} T_n \end{pmatrix} \\
&= \quad \vdots
\end{aligned}$$

$$\begin{aligned}
&= \begin{pmatrix} \sum_{i=1}^{n} (E_i U_i^J T_{i+1} T_{i+2} \cdots T_{n-1} T_n) \\ T_1 \cdots T_n \end{pmatrix} \\
&= \begin{pmatrix} U_1^J T_2 T_3 \cdots T_n \\ U_2^J T_3 T_4 \cdots T_n \\ \vdots \\ U_{n-1}^J T_n \\ U_n^J \\ T_1 T_2 \cdots T_n \end{pmatrix}.
\end{aligned}$$

Ce qui nous donne les blocs P_{12} et P_{22} de la matrice $Z^{\mathrm{T}}PZ = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}$.

Donc, d'après ce qui précède nous avons

$$\begin{aligned}
P_{21}E_j &= T_1T_2T_3\cdots T_{j-1}U_j \\
&= (I-U_1U_1^J)T_2T_3\cdots T_{j-1}U_j \\
&= T_2T_3\cdots T_{j-1}U_j - U_1\pi_{1j} \\
&= T_3T_4\cdots T_{j-1}U_j - U_1\pi_{1j} - U_2\pi_{2j} \\
&= T_{j-1}U_j - U_1\pi_{1j} - U_2\pi_{2j} - \cdots - U_{j-2}\pi_{j-2,j} \\
&= U_j - U_1\pi_{1j} - U_2\pi_{2j} - \cdots - U_{j-1}\pi_{j-1,j} \\
&= SE_j - SP_{11}E_j.
\end{aligned}$$

Par suite, nous avons le bloc P_{21} de la matrice $Z^{\mathrm{T}}PZ$:

$$P_{21} = S(I-P_{11}). \tag{2.15}$$

De plus, nous avons

$$\begin{aligned}
E_i^JP_{12} &= U_i^JT_{i+1}\cdots T_{n-1}T_n \\
&= U_i^JT_{i+1}\cdots T_{n-1} - U_i^JT_{i+1}\cdots T_{n-1}U_nU_n^J \\
&= U_i^JT_{i+1}\cdots T_{n-1} - \pi_{i,n}U_n^J \\
&= U_i^JT_{i+1}\cdots T_{n-2} - \pi_{i,n-1}U_{n-1}^J - \pi_{i,n}U_n^J \\
&= U_i^JT_{i+1} - \pi_{i,i+2}U_{i+2}^J - \cdots - \pi_{i,n}U_n^J \\
&= U_i^J - \pi_{i,i+1}U_{i+1}^J - \cdots - \pi_{i,n}U_n^J \\
&= E_i^JS^J - E_i^JP_{11}S^J.
\end{aligned}$$

Ainsi, nous avons le bloc P_{12} de la matrice $Z^{\mathrm{T}}PZ$:

$$P_{12} = (I-P_{11})S^J. \tag{2.16}$$

Aussi, nous avons

$$\begin{aligned}
P_{22} &= T_1T_2\cdots T_n \\
&= T_1T_2\cdots T_{n-1} - T_1T_2\cdots T_{n-1}U_nU_n^J \\
&= T_1T_2\cdots T_{n-1} - P_{21}E_nU_n^J \\
&= T_1T_2\cdots T_{n-2} - P_{21}E_{n-1}U_{n-1}^J - P_{21}E_nU_n^J \\
&= I - U_1U_1^J - P_{21}E_2U_2^J - P_{21}E_3U_3^J - \cdots - P_{21}E_nU_n^J \\
&= I - P_{21}(E_1U_1^J + E_2U_2^J + \cdots + E_nU_n^J) \\
&= I - P_{21}S^J \\
&= I - S(I-P_{11})S^J.
\end{aligned}$$

Ainsi, nous avons le bloc P_{22} de la matrice $Z^{\mathrm{T}}PZ$:

$$P_{22} = I - S(I-P_{11})S^J. \tag{2.17}$$

Donc, le bloc $P_{11} = 0$ si et seulement si $S^J S = I$, c'est-à-dire si et seulement si la matrice S est une matrice symplectique.

Enfin, si nous notons par $\bar{P}$ la matrice P calculée, alors d'après (2.16), (2.15) et (2.17) nous aurons :

$$Z^{\mathrm{T}} \bar{P} Z = \begin{pmatrix} P_{11} & (I - P_{11})\bar{S}^J \\ \bar{S}(I - P_{11}) & I - \bar{S}(I - P_{11})\bar{S}^J \end{pmatrix}. \tag{2.18}$$

Avec,

$$Z^{\mathrm{T}} P Z = \begin{pmatrix} 0 & S^J \\ S & I - SS^J \end{pmatrix}. \tag{2.19}$$

Nous rappelons que nous avons

$$\begin{pmatrix} 0 & 0 \\ A_{11} & A_{12} \\ 0 & 0 \\ A_{21} & A_{22} \end{pmatrix} = P \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \\ R_{21} & R_{22} \\ 0 & 0 \end{pmatrix}.$$

Donc,

$$Z^{\mathrm{T}} \begin{pmatrix} 0 & 0 \\ A_{11} & A_{12} \\ 0 & 0 \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = Z^{\mathrm{T}} P Z Z^{\mathrm{T}} \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \\ R_{21} & R_{22} \\ 0 & 0 \end{pmatrix} = Z^{\mathrm{T}} P Z \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

$$\begin{pmatrix} 0 \\ A \end{pmatrix} = Z^{\mathrm{T}} P Z \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Il est clair que les matrices $Z^{\mathrm{T}} P Z$ et $Z^{\mathrm{T}} \bar{P} Z$ sont des matrices J-orthogonales. Les $2n$ premières colonnes $Z^{\mathrm{T}} \bar{P}^{(2n)} Z$ de la matrice $Z^{\mathrm{T}} \bar{P} Z$ sont symplectiques. Ainsi :

$$Z^{\mathrm{T}} \bar{P}^{(2n)} Z = \begin{pmatrix} P_{11} \\ P_{12} \end{pmatrix} = \begin{pmatrix} P_{11} \\ \bar{S} - \bar{S} P_{11} \end{pmatrix}$$

et

$$(Z^{\mathrm{T}} \bar{P}^{(2n)} Z)^J Z^{\mathrm{T}} \bar{P}^{(2n)} Z = I.$$

En effet, nous avons l'équation

$$\begin{pmatrix} E_1 \\ A + E_2 \end{pmatrix} = Z^{\mathrm{T}} \bar{P} Z \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix},$$

en ce cas, nous remplaçons la matrice $Z^{\mathrm{T}} \bar{P} Z$ par la matrice $Z^{\mathrm{T}} \bar{P}^{(2n)} Z$. Autrement dit, d'après (2.18) et ces deux équations précédentes, nous aurons

$$Z^{\mathrm{T}} \bar{P}^{(2n)} Z \bar{R} = \begin{pmatrix} P_{11} \\ \bar{S}(I - P_{11}) \end{pmatrix} \bar{R} = \begin{pmatrix} P_{11}\bar{R} \\ \bar{S}(I - P_{11})\bar{R} \end{pmatrix} = \begin{pmatrix} P_{11}\bar{R} \\ \bar{S}(\bar{R} - P_{11}\bar{R}) \end{pmatrix} = \begin{pmatrix} E_1 \\ \bar{S}(\bar{R} - E_1) \end{pmatrix}.$$

Donc, nous avons

$$A + E_2 = \bar{S}(\bar{R} - E_1)$$

et par suite

$$E_2 + \bar{S} E_1 = A - \bar{S}\bar{R}.$$

Pour la suite de la démonstration, nous avons besoin du lemme suivant

Lemme 2.9.1. *Si $X_j + \Delta X_j \in \mathbb{R}^{m\times m}$ tel que $||\Delta X_j||_F \leq \gamma_j ||X_j||_2$ pour tout j, alors*

$$\| \prod_{j=0}^{p} (X_j + \Delta X_j) - \prod_{j=0}^{p} X_j \|_F \leq (\prod_{j=0}^{p} X_j(1+\gamma_j) - 1) \prod_{j=0}^{p} \|X_j\|_2. \tag{2.20}$$

(voir [63, p. 73]).

Nous pouvons voir que la matrice A_{k+1} dans la décomposition SR via les transformations de Householder symplectiques comme une suite de matrices données par $A_{k+1} = T_k A_k$, avec $k = 1 : r$, où $A_1 = A \in \mathbb{R}^{2n\times 2m}$ et $T_k = I - U_k \times U_k^J$ une transformation du Householder symplectique. Notons que chaque T_k est le produit de deux transformations du Householder symplectiques de types un (T_1') et deux (T_2') de telle sorte que $T_k = T_2' T_1'$ avec $T_1' = I + c_1 v_1 v_1^J$ et $T_2' = I + c_2 v_2 v_2^J$, tel que $S = T_r T_{r-1} \dots T_1$. De plus, nous supposons que $r\tilde{\gamma}_{2n} < \frac{1}{2}$. La matrice A_{r+1} satisfait $A_{k+1} = T_r T_{r-1} \cdots T_2 T_1 A$. Donc, la matrice calculée

$$\bar{A}_{k+1} = (T_r + \Delta T_r)(T_{r-1} + \Delta T_{r-1}) \cdots (T_2 + \Delta T_2)(T_1 + \Delta T_1) A.$$

Sachant que chaque T_k satisfait $\|\Delta T_k\|_F \leq 3\tilde{\gamma}_n \|T_k\|_2$ (voir [100]).

En utilisant l'équation (2.20) de la Lemme 2.9.1, nous obtenons

$$\bar{A}_{k+1} = S(A + \Delta A),$$

où

$$\begin{aligned} \|S\Delta A\|_2 &\leq ((1+3\tilde{\gamma}_n)^r - 1) \prod_{i=1}^{r} \|T_i\|_2 \|A\|_2 \\ &\leq \frac{3r\tilde{\gamma}_n}{1-3r\tilde{\gamma}_n} \prod_{i=1}^{r} \|T_i\|_2 \|A\|_2. \end{aligned} \tag{2.21}$$

Alors,

$$\|\Delta A\|_2 = \|S^J S \Delta A\|_2 \leq \|S^J\|_2 \|S\Delta A\|_2 \leq \|S\Delta A\|_2 \prod_{i=1}^{r} \|T_i^{-1}\|_2.$$

En utilisant (2.21), nous avons

$$\begin{aligned} \|\Delta A\|_2 &\leq \left(\frac{3r\tilde{\gamma}_n}{1-3r\tilde{\gamma}_n} \prod_{i=1}^{r} \|T_i\|_2 \|A\|_2) \right) \left(\prod_{i=1}^{r} \|T_i^{-1}\|_2 \right) \\ &\leq \frac{3r\tilde{\gamma}_n}{1-3r\tilde{\gamma}_n} \left(\prod_{i=1}^{r} \|T_i\|_2 \right) \left(\prod_{i=1}^{r} \|T_i^{-1}\|_2 \right) \|A\|_2 \\ &\leq \frac{3r\tilde{\gamma}_n}{1-3r\tilde{\gamma}_n} \prod_{i=1}^{r} \kappa_2(T_i) \|A\|_2. \end{aligned} \tag{2.22}$$

Du coup, pour $A + \Delta A = S\bar{R}$ nous avons

$$\|\Delta A\|_2 \leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} \|A\|_2 \prod_{i=1}^{m} \|T_i\|_2. \tag{2.23}$$

En outre, pour $A_1 = I$ dans (2.21) et (2.22), nous avons $\bar{S} = S(I + \Delta I)$. D'où,

$$\|\bar{S} - S\|_2 = \|S\Delta I\|_2 \leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} \prod_{i=1}^{m} \|T_i\|_2.$$

Dans ce cas, si $E_2 = 0$, alors $\bar{R} = R$. Comme nous avons $A + E_2 = \bar{S}(\bar{R} - E_1)$, alors $A = \bar{S}(R - E_1)$. Ainsi, $E_1 = -(A - \bar{S}R)$.

$$\|E_1\|_2 = \|A - \bar{S}R\|_2 = \|A - \bar{S}R + SR - SR\|_2 = \| - \bar{S}R + SR\|_2 = \|(\bar{S} - S)R\|_2.$$

Donc,

$$\begin{aligned}
\|E_1\|_2 &= \|(\bar{S} - S)R\|_2 \\
&\leq \|(\bar{S} - S)\|_2 \|R\|_2 \\
&\leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} \prod_{i=1}^{m} \|T_i\|_2 \|S^J\|_2 \|A\|_2 \\
&\leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} \|A\|_2 \prod_{i=1}^{m} \|T_i\|_2 \prod_{i=1}^{m} \|T_i^{-1}\|_2 \\
&\leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} \|A\|_2 \prod_{i=1}^{m} \kappa_2(T_i). \qquad (2.24)
\end{aligned}$$

Aussi,

$$\|\bar{R}\|_2 \leq \frac{1}{1-3m\tilde{\gamma}_n} \|A\|_2 \prod_{i=1}^{m} \|T_i\|_2.$$

Par suite,

$$\begin{aligned}
\|A - \bar{S}\bar{R}\|_2 &= \|(A - S\bar{R})((S - \bar{S})\bar{R})\|_2 \\
&\leq \|A - S\bar{R}\|_2 \|S - \bar{S}\|_2 \|\bar{R}\|_2 \\
&\leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} \left(1 + \frac{1}{1-3m\tilde{\gamma}_n}\right) \|A\|_2 \prod_{i=1}^{m} \kappa_2(T_i). \qquad (2.25)
\end{aligned}$$

De plus, nous avons

$$E_2 = \bar{S}(\bar{R} - E_1) - A = \bar{S}\bar{R} - A - \bar{S}E_1,$$

et dans le cas où $E_1 = 0$ nous aurons

$$E_2 = \bar{S}\bar{R} - A,$$

alors,

$$\begin{aligned}
\|E_2\|_2 &= \|A - \bar{S}\bar{R}\|_2 \\
&\leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} \left(1 + \frac{1}{1-3m\tilde{\gamma}_n}\right) \|A\|_2 \prod_{i=1}^{m} \kappa_2(T_i). \qquad (2.26)
\end{aligned}$$

De plus, nous avons $A = S(I - P_{11})R = S\bar{R}$

$A_{r+1} = S(I - P_{11})A = S(A + \Delta A)$ où $\Delta A = -SP_{11}A$ et comme

$$\|SP_{11}A\|_2 = \|S\Delta A\|_2 \leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} \prod_{i=1}^{m} \|T_i\|_2 \|A\|_2 \tag{2.27}$$

Donc,

$$\begin{aligned}
\|P_{11}\|_2 &= \|S^J SP_{11}AA^{-1}\|_2 \\
&\leq \|S^J\|_2 \|SP_{11}A\|_2 \|A^{-1}\|_2 \\
&\leq \prod_{i=1}^{r} \|T_i^{-1}\|_2 \frac{3r\tilde{\gamma}_n}{1-3r\tilde{\gamma}_n} \|A\|_2 \prod_{i=1}^{r} \|T_i\|_2 \|A^{-1}\|_2 \\
&\leq \frac{3r\tilde{\gamma}_n}{1-3r\tilde{\gamma}_n} k_2(A) \prod_{i=1}^{r} k_2(T_i).
\end{aligned}$$

Ainsi,

$$\|P_{11}\|_2 \leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} k_2(A) \prod_{i=1}^{m} k_2(T_i). \tag{2.28}$$

Comme $P_{21} = \bar{S}(I - P_{11})$, alors

$$\begin{aligned}
\|P_{21}\|_2 = \|\bar{S}(I - P_{11})\|_2 &= \|\bar{S}\|_2 \|I - P_{11}\|_2 \leq \|\bar{S}\|_2 \\
&\leq \frac{1}{1-3m\tilde{\gamma}_n} \prod_{i=1}^{m} \|T_i\|_2.
\end{aligned}$$

De même, $P_{12} = (I - P_{11})\bar{S}^J$

$$\begin{aligned}
\|P_{12}\|_2 = \|(I - P_{11})\bar{S}^J\|_2 &= \|\bar{S}^J\|_2 \|I - P_{11}\|_2 \leq \|\bar{S}^J\|_2 \\
&\leq \frac{1}{1-3m\tilde{\gamma}_n} \prod_{i=1}^{m} \|T_i\|_2.
\end{aligned}$$

Aussi, $P_{22} = I + \bar{S}(I - P_{11})\bar{S}^J$:

$$\begin{aligned}
\|P_{22}\|_2 &= \|I + \bar{S}(I - P_{11})\bar{S}^J\|_2 \\
&= 1 + \|\bar{S}(I - P_{11})\bar{S}^J\|_2 \\
&= 1 + \|\bar{S}\|_2 \|(I - P_{11})\|_2 \|\bar{S}^J\|_2 \\
&= 1 + \|\bar{S}\|_2 \|\bar{S}^J\|_2 \\
&= 1 + \|\bar{S}\|_2^2 \\
&= 1 + \frac{1}{(1-3m\tilde{\gamma}_n)^2} \prod_{i=1}^{m} \kappa_2(T_i).
\end{aligned}$$

Par suite,

$$\begin{aligned}
\|\bar{P}\|_2 &\leq \|P_{11}\|_2 + \|P_{12}\|_2 + \|P_{21}\|_2 + \|P_{22}\|_2 \\
&\leq \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n} k_2(A) \prod_{i=1}^{m} k_2(T_i) + \frac{2}{1-3m\tilde{\gamma}_n} \prod_{i=1}^{m} \|T_i\|_2 + 1 + \frac{1}{(1-3m\tilde{\gamma}_n)^2} \prod_{i=1}^{m} \kappa_2(T_i) \\
&\leq 1 + \frac{1}{1-3m\tilde{\gamma}_n} \left(2 + \left(\frac{1}{1-3m\tilde{\gamma}_n} + 3m\tilde{\gamma}_n k_2(A) \right) \prod_{i=1}^{m} \|T_i\|_2 \right) \prod_{i=1}^{m} \|T_i\|_2.
\end{aligned}$$

$$Z^{\mathrm{T}}(\bar{P}' - P')Z = \begin{pmatrix} P_{11} & (I-P_{11})S^J \\ S(I-P_{11}) & I-S(I-P_{11})S^J \end{pmatrix} - \begin{pmatrix} 0 & S^J \\ S & I-SS^J \end{pmatrix} \tag{2.29}$$

$$= \begin{pmatrix} P_{11} & -P_{11}S^J \\ -SP_{11} & SP_{11}S^J \end{pmatrix} \tag{2.30}$$

Pour $E_1 = 0$, nous savons que $P_{11} = 0$, ainsi

$$Z^{\mathrm{T}}(\bar{P} - P)Z = \begin{pmatrix} 0 & \bar{S}^J \\ \bar{S} & I-\bar{S}\bar{S}^J \end{pmatrix} - \begin{pmatrix} 0 & S^J \\ S & I-SS^J \end{pmatrix} \tag{2.31}$$

$$= \begin{pmatrix} 0 & (\bar{S}-S)^J \\ \bar{S}-S & -(\bar{S}-S)(\bar{S}+S)^J \end{pmatrix}. \tag{2.32}$$

D'où,

$$\begin{aligned} \|E_2'\|_2 &= \|\bar{P} - P\|_2 \\ &= \|Z^{\mathrm{T}}(\bar{P} - P)Z\|_2 \\ &\leq \|\bar{S} - S\|_2 + \|(\bar{S} - S)^J\|_2 + \|\bar{S} - S\|_2 \|(\bar{S} + S)^J\|_2 \\ &\leq (2 + 2\|S\|_2 + \|\bar{S} - S\|_2)\|\bar{S} - S\|_2 \\ &\leq (2 + 2\|S\|_2 + \|S\Delta I\|_2)\|S\Delta I\|_2 \\ &\leq (2 + 2\prod_{i=1}^{m}\|T_i\|_2 + \frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}\prod_{i=1}^{m}\|T_i\|_2)\frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}\prod_{i=1}^{m}\|T_i\|_2 \\ &\leq (2 + \frac{2-3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}\prod_{i=1}^{m}\|T_i\|_2)\frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}\prod_{i=1}^{m}\|T_i\|_2. \end{aligned} \tag{2.33}$$

Puis, pour $E_2 = 0$, nous savons que $\bar{S} = S$, ainsi

$$\begin{aligned} Z^{\mathrm{T}}(\bar{P} - P)Z &= \begin{pmatrix} P_{11} & (I-P_{11})S^J \\ S(I-P_{11}) & I-S(I-P_{11})S^J \end{pmatrix} - \begin{pmatrix} 0 & S^J \\ S & I-SS^J \end{pmatrix} \\ &= \begin{pmatrix} P_{11} & -P_{11}S^J \\ -SP_{11} & SP_{11}S^J \end{pmatrix} \\ &= \begin{pmatrix} I & 0 \\ 0 & S \end{pmatrix}\begin{pmatrix} P_{11} & -P_{11} \\ -P_{11} & P_{11} \end{pmatrix}\begin{pmatrix} I & 0 \\ 0 & S^J \end{pmatrix}. \end{aligned} \tag{2.34}$$

$$\begin{aligned} \|E_1'\|_2 &= \|\bar{P} - P\|_2 \\ &= \|Z^{\mathrm{T}}(\bar{P} - P)Z\|_2 \\ &\leq 2\|P_{11}\|_2(1 + \|S\|_2)^2 \\ &\leq 2\frac{3m\tilde{\gamma}_n}{1-3m\tilde{\gamma}_n}k_2(A)\prod_{i=1}^{m}k_2(T_i)(1 + \prod_{i=1}^{m}\|T_i\|_2)^2. \end{aligned} \tag{2.35}$$

Ainsi, nous avons trouvé le résultat à chercher. □

2.10 Conclusion

La décomposition symplectiques SR (par un processus de type Gram–Schmidt, par de transformation de Householder, par de décomposition de Schur, etc.,) en utilisant un produit scalaire antisymétrique $(.,.)_J$ est une étape essentielle dans les méthodes de réduction de modèle ou dans les méthodes de conservation de la structure où la préservation de la J-orthogonalité est très importante.

La décomposition SR peut être calculé via l'algorithme de Gram-Schmidt symplectique. Comme dans le cas classique, une perte d'orthogonalité peut se produire. Pour y remédier, nous avons proposé une technique de ré-J-orthogonalisation. Nous avons introduit deux algorithmes $RSGSi$ et $RMSGSi$, qui consistent à ré-orthogonaliser deux fois seulement les vecteurs à calculer. La perte de la J-orthogonalité s'est améliorée de manière très significative. Nous avons montré la pertinence du choix des paramètres libres impliqués dans l'algorithme de Gram–Schmidt symplectique, qui a une incidence sur la préservation de la J-orthogonalité. Nous remarquons aussi que la ré-J-orthogonalisation n'intervient pas sur l'erreur absolue de la décomposition.

Une autre façon de calculer la décomposition SR est basée sur les transformations de Householder symplectiques. Celles ci présentent des paramètres libres. Un choix optimal a abouti à l'algorithme $SROSH$. Cependant, ce dernier peut être sujet à une instabilité numérique. Nous avons proposé deux nouvelles versions modifiées $SRMSH$ et $SRMSH2$, qui ont l'avantage d'être plus stable numériquement. Une étude approfondie a été faite, présentant les différentes versions $SRMSH$ et $SRMSH2$.

L'étude directe de la propagation des erreurs d'arrondis dans les algorithmes de Gram-Shmidt symplectique est très difficile à effectuer. Nous avons réussi à contourner cette difficulté et des majorations pour la perte de la J-orthogonalité et de l'erreur de factorisation ont été données en se basant sur l'équivalence entre la décomposition SR via les transformations de Householder symplectiques et la décomposition SR via Gram–Schmidt symplectique modifié $MSGS$.

Chapitre 3

Réduction d'une matrice sous forme J-Hessenberg

« *In the middle of difficulty lies opportunity* »

Albert Einstein (1879-1955)

Sommaire

3.1 Introduction

Dans le but de construire un algorithme SR, qui est un algorithme de type QR, pour calculer les valeurs propres et vecteurs propres d'une matrice, une réduction de la matrice sous la forme J-Hessenberg supérieure est cruciale. Cela est dû au fait que l'algorithme final que nous recherchons devrait avoir une complexité d'ordre $\mathcal{O}(n^3)$. En effet, les matrices symplectiques peuvent servir à transformer une matrice A en une forme de J-Hessenberg. La relation entre cette transformation sous forme J-Hessenberg et la factorisation SR est tout à fait analogue à la relation entre la réduction sous la forme Hessenberg et la factorisation QR dans le cas classique (Euclidien).

Bunse-Gerstner et Mehrmann [40] ont présenté un algorithme de réduction d'une matrice arbitraire sous la forme J-Hessenberg supérieure appelé $JHESS$, en utilisant dans ce but, les transformations de Givens symplectiques, les transformations de Householder symplectiques et les transformations de Gauss symplectiques. Les deux premières transformations (Givens et Householder) sont orthogonales, tandis que les derniers ne le sont pas. L'algorithme $JHESS$ est basée sur une adaptation de l'algorithme de décomposition SR via $SRDECO$. De point de vue algèbre linéaire, la décomposition de SR via l'algorithme $SRDECO$ ne correspond pas à l'analogue de décomposition QR via Householder. Ainsi, la réduction sous forme J-Hessenberg $JHESS$ n'est pas analogue à la réduction sous la forme Hessenberg. En fait, sa construction a été basée sur les transformations orthogonales et symplectiques, qui ont été introduits par Van Loan [114], complétées par une transformation symplectique et non orthogonale [40]. La décomposition SR analogue à QR via Householder a été introduite en [102], c'est-à-dire la décomposition via l'algorithme $SRSH$ dans sa forme générale ou bien via sa forme optimale $SROSH$.

Dans ce paragraphe, nous allons montrer qu'il possible de réduire une matrice sous la forme J-Hessenberg, en utilisant exclusivement les transformations de Householder symplectiques [104], telles qu'elles sont introduites dans [102]. Ces transformations sont des transvections, et s'écrivent comme l'identité, modifiée par le rajout d'un terme de rang 1.

De point de vue algébrique, elles sont dans le cas symplectique, les analogues des transformations de Householder dans le cas Euclidien.

De ce fait, l'algorithme que nous allons construire sera, d'un point de vue algébrique, exactement l'analogue dans le cas symplectique, de l'algorithme faisant la réduction d'une matrice arbitraire, sous la forme Hessenberg, via les transformations de Householder, du cas Euclidien.

Dans un premier temps, nous allons proposer une version générale de l'algorithme ($JHSH$), c'est-à-dire, la question du choix des paramétrés libres ne sera pas abordée.

Des tests numériques, montrant la pertinence de ce choix seront donnés ($JHOSH$). Ce choix garantit une meilleure préservation de la J-orthogonalité, aussi bien, que la factorisation.

Cependant, cet algorithme, comme sa version générale, est basé exclusivement sur des transformations de Householder symplectiques. Bien qu'elle soient juste une modification de l'identité par une matrice de rang 1, peut arriver qu'elles soient mal conditionnées, d'autant plus qu'elles ne sont pas forcement orthogonales.

Dans la partie suivante, nous allons montrer comment on peut remplace la moitié

de ces transformations de Householder symplectiques, non forcément orthogonales, par d'autres transformations élémentaires, qui ont l'avantage d'être symplectiques et orthogonales ($JHMSH$).

Par la suite, pour l'autre moitié, nous avons limité l'intervention de la transformations de Householder symplectiques à l'annulation d'une seule entrée au lieu de toute la colonne ($JHMSH2$).

L'algorithme obtenu est donc numériquement aussi stable que possible (les transformations orthogonales), où les paramètres libres si ont été choisi soigneusement. Des tests numériques, montrant la supériorité de cette dernière version par rapport aux précédentes sont données.

Puis, nous rappelons l'algorithme $JHESS$ de la réduction sous la forme J-Hessenberg de [40].

Nous comparons ensuite notre algorithme dans sa dernière version, à celui donné en [40].

L'algorithme présenté en [40], utilise pour moitié les mêmes transformations symplectiques et orthogonales que notre algorithme. Mais, il utilise pour l'autre moitié, des transformations de Gauss symplectiques [40, p 1106]. Le premier inconvénient de cet algorithme est que ces transformations peuvent être généralement mal conditionnes [40], (voir aussi tests numériques).

Le second inconvénient, qui est majeur, réside dans la façon proposée en [40] pour y remédier, dans le contexte de l'algorithme SR, dans sa version implicite.

En effet, il est proposé, lorsque le problème est rencontré à l'itération consistant à calculer la matrice $A_{i+1} = S_i^{-1} A_i S_i$, où la matrice A_i est la matrice courante, (elle a la forme J-Hessenberg) et S_i la matrice de similarité, (elle est symplectique), choisir une transformation de similarité exceptionnelle sous la forme $S_i = I - ww^{\mathrm{T}}J$ où le vecteur w est un vecteur aléatoire, sachant que $\|w\| = 1$. Remarquons d'abord que ce choix $I - ww^{\mathrm{T}}J$ est une transformation de Householder symplectique au sens du [102]. Un choix aléatoire du vecteur w a pour conséquence de détruire la structure J-Hessenberg de la matrice A_i : en effet la matrice A_i est de J-Hessenberg, la matrice $A_{i+1} = S_i^{-1} A_i S_i$ ne le sera plus avec un choix aléatoire de w, avec $S_i = I - ww^{\mathrm{T}}J$. Cela a pour conséquence de "doubler" le coût de l'algorithme SR implicite, à chaque fois nous avons recours à une telle itération, puisque le coût majeur de SR algorithme implicite, provient essentiellement de la première étape qui consiste à réduire une matrice sous la forme J-Hessenberg. Dans des cas pessimiste, nous pouvons s'attendre à un coût de $\mathcal{O}(n^4)$ d'opérations arithmétiques.

Notre dernière version de l'algorithme, ne présente pas ces inconvénients, et construire de ce fait, une amélioration significative, et une alternative efficace.

Enfin, l'étude approfondie à propos les aspects numériques nos nouveaux algorithmes ainsi que l'algorithme $JHESS$ et ces problèmes d'instabilités numériques (breakdowns\near-breakdowns) nous amène à décrire une stratégie très efficace pour corriger et pré-dicter de tels problèmes. De ce fait, nous sommes amenés à un choix algorithmique, que nous avons mis en œuvre dans les versions modifiées de tous les algorithmes. De cette manière, l'algorithme $JHESS$ incorporant cette modification sera nommé comme $MJHESS$. Pareillement, les versions modifiées de l' algorithme $JHSH$ et ses différentes variantes $JHOSH$, $JHMSH$, $JHMSH2$ seront nommés $MJHSH$, JHM^2SH et JHM^2SH2 respectivement. Notre stratégie est certifiée par des exemples numériques.

3.2 La réduction sous la forme J-Hessenberg via les transformations de Householder symplectiques

3.2.1 La réduction J-Hessenberg via l'algorithme JHSH

Dans cette partie, nous allons proposer un nouvel algorithme appelé $JHSH$ qui réduit une matrice arbitraire de taille $2n \times 2n$ sous la forme J-Hessenberg. Cet algorithme utilise des transformations de Householder symplectiques et il est basé sur une adaptation de la décomposition SR via $SRSH$ ou $SROSH$. Dans le cas symplectique, la réduction J-Hessenberg via $JHSH$ est la version analogue de la réduction Hessenberg via $hess$ qui adapte la décomposition QR par des transformations de Householder dans le cas classique.

Rappelons que dans le choix de deux transformations de Householder symplectiques T_1 et T_2, nous avons trois paramètres libres : Soient $\{e_1, e_2, \ldots, e_{2n}\}$ une base canonique de $\mathbb{R}^{2n\times 2n}$, $[a,b] \in \mathbb{R}^{2n\times 2}$ et trois scalaires arbitraires ρ, μ, ν tels que

$$T_1(a) = \rho e_1 \tag{3.1}$$

et

$$T_2(e_1) = e_1 \tag{3.2}$$
$$T_2(T_1(b)) = \mu e_1 + \nu e_{n+1}. \tag{3.3}$$

Comme les transformations T_1, T_2 sont des isométries symplectiques, alors

$$\langle T_2(T_1(a)), T_2(T_1(b))\rangle_J = \langle T_1(a), T_1(b)\rangle_J = \langle a,b\rangle_J = a^J b \tag{3.4}$$

et

$$\langle T_2(T_1(a)), T_2(T_1(b))\rangle_J = \langle \rho e_1, \mu e_1 + \nu e_{n+1}\rangle_J = \langle a,b\rangle_J = \rho\nu. \tag{3.5}$$

Donc,

$$\rho\nu = a^J b. \tag{3.6}$$

De plus,

$$\langle T_2(a), T_2(e_1)\rangle_J = \langle a, e_1\rangle_J = a^J e_1 = -a_{n+1}. \tag{3.7}$$

Ainsi,

$$\nu = a_{n+1}. \tag{3.8}$$

Alors, à cette étape il n'y a que deux paramètres libres.

Théorème 3.2.1. *Soient $[a,b] \in \mathbb{R}^{2n\times 2}$ et ρ, μ deux scalaires arbitraires et $\nu = a_{n+1}$ tels que :*

$$\begin{aligned}
&\rho\nu = a^J b,\\
&c_1 = -\frac{1}{a^J \rho e_1}, && v_1 = \rho e_1 - a,\\
&c_2 = -\frac{1}{(T_1(b))^J(\mu e_1 + \nu e_{n+1})}, && v_2 = \mu e_1 + \nu e_{n+1} - T_1(b).
\end{aligned}$$

Alors,

$$T_1 = I + c_1 v_1 v_1^J, \tag{3.9}$$

$$T_2 = I + c_2 v_2 v_2^J, \tag{3.10}$$

vérifient que

$$\begin{aligned} T_1(a) &= \rho e_1, \\ T_2(T_1(a)) &= T_1(a), \\ T_2(T_1(b)) &= \mu e_1 + \nu e_{n+1}. \end{aligned}$$

Remarque 3.2.1. *Puisque la composante* $(n+1)^{ième}$ *de l'entrée du vecteur* v_2 *est nulle* ($v_2(n+1) = 0$)*, alors pour n'importe quel vecteur* $x \in \mathbb{R}^{2n}$ *la transformation* T_2 *ne modifie pas la composante* $(n+1)^{ième}$ *du vecteur* $T_2(x)$.

Théorème 3.2.2. *Soit* $v \in \mathbb{R}^{2n}$ *un vecteur avec la partition* $v = [0^{\mathrm{T}}, u^{\mathrm{T}}, 0^{\mathrm{T}}, w^{\mathrm{T}}]^{\mathrm{T}}$*, de sorte que* $[u, w] \in \mathbb{R}^{(n-i)\times 2}$ *pour* $1 \le i \le n-1$ *et soit* $\tilde{v} \in \mathbb{R}^{2(n-i)}$ *avec la partition* $\tilde{v} = [u^{\mathrm{T}}, w^{\mathrm{T}}]^{\mathrm{T}}$*. Considérons les transformations de Householder symplectiques*

$$\begin{aligned} T &= I + c v v^J, \\ \tilde{T} &= I + c \tilde{v} \tilde{v}^J, \end{aligned}$$

alors, $\forall x_1 \in \mathbb{R}^i$, $\forall y_1 \in \mathbb{R}^i$, $\forall x \in \mathbb{R}^{n-i}$, $\forall y \in \mathbb{R}^{n-i}$*, nous avons*

$$T \begin{bmatrix} x_1 \\ x \\ y_1 \\ y \end{bmatrix} = \begin{bmatrix} x_1 \\ x' \\ y_1 \\ y' \end{bmatrix}, \tag{3.11}$$

avec

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}.$$

Preuve. Nous avons :

$$\begin{aligned} v^J \begin{bmatrix} x_1 \\ x \\ y_1 \\ y \end{bmatrix} &= v^{\mathrm{T}} J \begin{bmatrix} x_1 \\ x \\ y_1 \\ y \end{bmatrix} \\ &= [0^{\mathrm{T}}, u^{\mathrm{T}}, 0^{\mathrm{T}}, w^{\mathrm{T}}] J \begin{bmatrix} x_1 \\ x \\ y_1 \\ y \end{bmatrix} \\ &= u^{\mathrm{T}} y - w^{\mathrm{T}} x \\ &= [u^{\mathrm{T}}, w^{\mathrm{T}}] J \begin{bmatrix} x \\ y \end{bmatrix} \\ &= \tilde{v}^J \begin{bmatrix} x \\ y \end{bmatrix}. \end{aligned}$$

Donc,

$$T\begin{bmatrix} x_1 \\ x \\ y_1 \\ y \end{bmatrix} = \begin{bmatrix} x_1 \\ x' \\ y_1 \\ y' \end{bmatrix}$$
$$= \begin{bmatrix} x_1 \\ x \\ y_1 \\ y \end{bmatrix} + c\begin{bmatrix} 0 \\ u \\ 0 \\ w \end{bmatrix}[u^{\mathrm{T}}, w^{\mathrm{T}}]J\begin{bmatrix} x \\ y \end{bmatrix}.$$

En conséquence, il est clair que

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + c\begin{bmatrix} u \\ w \end{bmatrix}[u^{\mathrm{T}}, w^{\mathrm{T}}]J\begin{bmatrix} x \\ y \end{bmatrix}$$
$$= \begin{bmatrix} x \\ y \end{bmatrix} + c\begin{bmatrix} u \\ w \end{bmatrix}[u^{\mathrm{T}}, w^{\mathrm{T}}]J\begin{bmatrix} x \\ y \end{bmatrix}$$
$$= (I + c\tilde{v}\tilde{v}^J)\begin{bmatrix} x \\ y \end{bmatrix}$$
$$= \tilde{T}\begin{bmatrix} x \\ y \end{bmatrix},$$

avec

$$T\begin{bmatrix} x_1 \\ 0 \\ y_1 \\ 0 \end{bmatrix} = \begin{bmatrix} x_1 \\ 0 \\ y_1 \\ 0 \end{bmatrix}.$$

□

Remarque 3.2.2. *Le théorème 3.2.2 reste valable dans le cas où nous remplaçons la transformation de Householder symplectique T par $T^J = I - cvv^J$.*

Les théorèmes 3.2.1 et 3.2.2 constituent le principal outil sur lequel est basée notre réduction d'une matrice sous la forme J-Hessenberg via les transformations de Householder symplectiques.

D'abord, nous allons commencer par une description détaille des étapes de l'algorithme. Soit la matrice $A = [a_1, \ldots, a_n, a_{n+1}, \ldots, a_{2n}] \in \mathbb{R}^{2n\times 2n}$ et posons que $A^{(0)} = A$.

La première étape :

Au début, choisissons une transformation de Householder symplectique H_1 pour annuler les entrées de la position 2 jusqu'à position n et de la position $n+2$ jusqu'à la position $2n$ de la première colonne de la matrice A. Cette transformation de Householder symplectique s'écrit sous la forme

$$H_1 = I + c_1 v_1 v_1^J$$

tels que $c_1 \in \mathbb{R}$ et $v_1 \in \mathbb{R}^{2n}$. Le vecteur v_1 est le vecteur directeur de H_1. La transformation H_1 correspond à la transformation T_2, définit dans le théorème 3.2.1. Comme $H_1 e_1 = e_1$, alors nous aurons $v_1^J e_1 = v_1^{\mathrm{T}} J e_1 = 0$. Ainsi, l'entrée de la position $(n+1)^{\text{ième}}$ de vecteur v_1 est égale à zéro. Il s'ensuit que pour tout vecteur x, l'entrée de la position $(n+1)^{\text{ième}}$ de vecteur $H_1 x$ préserve. La direction v_1 de la transformation H_1 est donnée par :

$$v_1 = A^{(1)}_{1,1} e_1 + a_1 (n+1) e_{n+1} - a_1$$

où $A^{(1)}_{1,1}$ est un scalaire arbitraire. De plus, nous avons $H_1^J e_1 = e_1$, ce qui implique que la première colonne de H_1 et de H_1^J est égale à e_1. En conséquence, la multiplication de la matrice A à gauche par H_1 préserve la $(n+1)^{\text{ième}}$ ligne et crée les zéros désirés dans la première colonne. Nous obtenons

$$A'^{(1)} = H_1 A = \begin{bmatrix} A^{(1)}_{(1,1)} & A'^{(1)}_{(1,2:n)} & A'^{(1)}_{(1,n+1:2n)} \\ 0 & A'^{(1)}_{(2:n,2:n)} & A'^{(1)}_{(2:n,n+1:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(0)}_{(n+1,2:n)} & A^{(0)}_{(n+1,n+1:2n)} \\ 0 & A'^{(1)}_{(n+2:2n,2:n)} & A'^{(1)}_{(n+2:2n,n+1:2n)} \end{bmatrix}.$$

Dans cette étape, $A^{(1)}_{(1,1)}$ est un paramètre libre.

La multiplication de $H_1 A$ à droite par H_1^J préserve la première colonne de $H_1 A H_1^J$ et nous obtenons :

$$A^{(1)} = H_1 A H_1^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1:2n)} \\ 0 & A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1:2n)} \\ 0 & A^{(1)}_{(n+2:2n,2:n)} & A^{(1)}_{(n+2:2n,n+1:2n)} \end{bmatrix}.$$

□

La deuxième étape :

Ensuite, l'étape suivante consiste à choisir une deuxième transformation de Householder symplectique H_2 pour mettre des zéros dans les entrées de la position 3 jusqu'à la position n et de la position $n+2$ jusqu'à la position $2n$ de la $(n+1)^{\text{ième}}$ colonne de la matrice $A^{(1)}$.

Ainsi, nous prenons la sous matrice

$$\tilde{A}^{(1)} = \begin{bmatrix} A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1:2n)} \\ A^{(1)}_{(n+2:2n,2:n)} & A^{(1)}_{(n+2:2n,n+1:2n)} \end{bmatrix}.$$

Elle est obtenue de la matrice $A^{(1)}$ en supprimant la première colonne, la première ligne et la $(n+1)^{\text{ième}}$ ligne. Soit $A^{(2)}_{(2,n+1)}$ un scalaire arbitraire non nul $(A^2_{(2,n+1)} \neq 0)$.

Soit la transformation $\tilde{H}_2 = I_{2n-2} + c_2 \tilde{v}_2 \tilde{v}_2^J$, donnée par le théorème 3.2.1 tel que :

$$\tilde{v}_2 = \begin{bmatrix} u_2 \\ \hline w_2 \end{bmatrix} = A^{(2)}_{(2,n+1)} e_1 - \tilde{A}^{(1)}_{(:,n)} \in \mathbb{R}^{(2n-2)},$$

où $u_2 \in \mathbb{R}^{n-1}$, $w_2 \in \mathbb{R}^{n-1}$, alors nous obtenions :

$$\tilde{A}'^{(2)} = \tilde{H}_2 \tilde{A}^{(1)} = \begin{bmatrix} \tilde{A}'^{(2)}_{(2,2:n)} & \tilde{A}^{(2)}_{(2,n+1)} & \tilde{A}'^{(2)}_{(2,n+2:2n)} \\ \tilde{A}'^{(2)}_{(3:n,2:n)} & 0 & \tilde{A}'^{(2)}_{(3:n,n+2:2n)} \\ \tilde{A}'^{(2)}_{(n+2:2n,2:n)} & 0 & \tilde{A}'^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

La transformation $\tilde{H}_2$ correspond à la transformation de Householder symplectique de type un, c'est-à-dire la transformation T_1 du théorème 3.2.1. Soit $H_2 = I_{2n} + c_2 v_2 v_2^J$, avec

$$v_2 = \begin{bmatrix} 0 \\ \hline u_2 \\ \hline 0 \\ w_2 \end{bmatrix} \in \mathbb{R}^{2n},$$

alors H_2 est une transformation de Householder symplectique. En utilisant le théorème 3.2.2, nous aurons

$$A'^{(2)} = H_2 A^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(1)}_{(1,n+2:2n)} \\ 0 & A'^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A'^{(2)}_{(2,n+2:2n)} \\ 0 & A'^{(2)}_{(3:n,2:n)} & 0 & A'^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(1)}_{(n+1,n+2:2n)} \\ 0 & A'^{(2)}_{(n+2,2:n)} & 0 & A'^{(2)}_{(n+2,n+2:2n)} \\ 0 & A'^{(2)}_{(n+3:2n,2:n)} & 0 & A'^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

La transformation H_2 préserve la première et la $(n+1)^{\text{ième}}$ lignes de la matrice $H_2 A^{(1)}$. Elle ne modifie pas aussi la première colonne de $H_2 A^{(1)}$ et crée les zéros souhaitées dans la $(n+1)^{\text{ième}}$ colonne de la matrice $A'^{(2)}$.

Ensuite, nous multiplions la matrice $\tilde{A}'^{(2)} = \tilde{H}_2 \tilde{A}^{(1)}$ à droite par la transformation $\tilde{H}_2^J = I_{2n-2} - c_2 \tilde{v}_2 \tilde{v}_2^J$. Ainsi,

$$\tilde{A}'^{(2)} \tilde{H}_2^J = \tilde{H}_2 \tilde{A}^{(1)} \tilde{H}_2^J = \begin{bmatrix} A^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A^{(2)}_{(2,n+2:2n)} \\ A^{(2)}_{(3:n,2:n)} & 0 & A^{(2)}_{(3:n,n+2:2n)} \\ A^{(2)}_{(n+2:2n,2:n)} & 0 & A^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

Cela correspond à la multiplication de $H_2 A^{(1)}$ à droite par la transformation de Householder symplectique H_2^J. Cette multiplication ne modifie pas la première et la $(n+1)^{\text{ième}}$ colonnes de la matrice $H_2 A^{(1)} H_2^J$. Donc,

$$A^{(2)} = H_2 A^{(1)} H_2^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2:2n)} \\ 0 & A^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A^{(2)}_{(2,n+2:2n)} \\ 0 & A^{(2)}_{(3:n,2:n)} & 0 & A^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(2)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2,2:n)} & 0 & A^{(2)}_{(n+2,n+2:2n)} \\ 0 & A^{(2)}_{(n+3:2n,2:n)} & 0 & A^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

Comme $H_2 e_1 = e_1$ et $H_2 e_{n+1} = e_{n+1}$, alors la première colonne des matrices H_2 et H_2^J est égale à e_1. Aussi, la $(n+1)^{\text{ième}}$ colonne des matrices H_2 et H_2^J est égale à e_{n+1}. □

La troisième étape :

Dans cette étape, nous voulons annuler les entrées de la position 3 jusqu'à la position n et de la position $n+3$ jusqu'à la position $2n$ de la deuxième colonne de la matrice $A^{(2)}$.

Soit la sous matrice $\tilde{A}^{(2)}$ obtenu en supprimant la première et la $(n+1)^{\text{ième}}$ lignes et colonnes de la matrice $A^{(2)}$, c'est-à-dire

$$\tilde{A}^{(2)} = \begin{bmatrix} A^{(2)}_{(2:n,2:n)} & A^{(2)}_{(2:n,n+2:2n)} \\ A^{(2)}_{(n+2:2n,2:n)} & A^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

Cette étape consiste à répéter la première étape sur la nouvelle matrice réduite $\tilde{A}^{(2)}$. En d'autres termes, nous choisissons une transformation de Householder symplectique $\tilde{H}_3$, ce qui signifie que nous devons calculer un scalaire réel $c_3 \in \mathbb{R}$ et un vecteur

$$\tilde{v}_3 = \begin{bmatrix} u_3 \\ \hline w_3 \end{bmatrix} \in \mathbb{R}^{2n-2},$$

avec $u_3 \in R^{n-1}$, $w_3 \in R^{n-1}$ tel que $\tilde{H}_3 = I_{2n-2} + c_3 \tilde{v}_3 \tilde{v}_3^J$. Cette transformation annule les entrées de la position 2 jusqu'à la position $n-1$ et de la position $n+1$ jusqu'à la position $2n-2$ de la première colonne de la matrice $\tilde{A}^{(2)}$, sachant que $\tilde{H}_3 e_1 = e_1 \in \mathbb{R}^{2n-2}$. En effet, la transformation $\tilde{H}_3$ correspond à la transformation de Householder symplectique T_2 dans le théorème 3.2.1. La direction de $\tilde{H}_3$ est donnée par le vecteur

$$\tilde{v}_3 = A^{(3)}_{(2,2)} e_1 + A^{(2)}_{(n+2,2)} e_n - \tilde{A}^{(2)}_{(:,1)} \in \mathbb{R}^{2n-2},$$

avec $A^{(3)}_{(2,2)}$ est un scalaire arbitraire non nul. En multipliant la matrice $\tilde{A}^{(2)}$ à gauche par $\tilde{H}_3$, cette dernière préserve les entrées de la $n^{\text{ième}}$ ligne de la matrice $H_3 \tilde{A}^{(2)}$. Par conséquent, nous obtenons

$$\tilde{A}'^{(3)} = \tilde{H}_3 \tilde{A}^{(2)} = \begin{bmatrix} A^{(3)}_{(2,2)} & A'^{(3)}_{(2,3:n)} & A'^{(3)}_{(2,n+2:2n)} \\ 0 & A'^{(3)}_{(3:n,3:n)} & A'^{(3)}_{(3:n,n+2:2n)} \\ A^{(2)}_{(n+2,2)} & A^{(2)}_{(n+2,3:n)} & A^{(2)}_{(n+2,n+2:2n)} \\ 0 & A'^{(3)}_{(n+3:2n,3:n)} & A'^{(3)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

Remarquons ici, que l'entrée de la position $n^{\text{ième}}$ du vecteur $\tilde{v}_3$ est nulle, $\tilde{v}_3(n) = 0$.

Soit $H_3 = I_{2n} + c_3 v_3 v_3^J$ la transformation de Householder symplectique où le vecteur de direction

$$v_3 = \begin{bmatrix} 0 \\ \hline u_3 \\ \hline 0 \\ \hline w_3 \end{bmatrix} \in \mathbb{R}^{2n}.$$

Les entrées des positions 1, $(n+1)$ et $(n+2)$ du vecteur v_3 sont nulles. De ce fait, la transformation H_3 préserve les lignes 1, $(n+1)$ et $(n+2)$ de la matrice $H_3A^{(2)}$ après la multiplication à gauche et vérifie que $H_3e_1 = e_1$, $H_3e_2 = e_2$ et $H_3e_{n+1} = e_{n+1}$. Par conséquent, H_3 ne modifie pas la première et la $(n+1)^{\text{ième}}$ colonnes de la matrice $H_3A^{(2)}$. La transformation H_3 annule les entrées de la position 3 jusqu'à la position n et de la position $n+3$ jusqu'à la position $2n$ de la deuxième colonne de la matrice $A^{(2)}$.

Sur ces entrefaites, nous aurons :

$$A'^{(3)} = H_3A^{(2)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(2)}_{(1,3:n)} & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2:2n)} \\ 0 & A^{(3)}_{(2,2)} & A'^{(3)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A'^{(3)}_{(2,n+2:2n)} \\ 0 & 0 & A'^{(3)}_{(3:n,3:n)} & 0 & A'^{(3)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2)} & A^{(2)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(2)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2,2)} & A^{(2)}_{(n+2,3:n)} & 0 & A^{(2)}_{(n+2,n+2:2n)} \\ 0 & 0 & A'^{(3)}_{(n+3:2n,3:n)} & 0 & A'^{(3)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

La transformation $H_3^J = I_{2n} - c_3v_3v_3^J$ ne modifie pas les colonnes 1, 2 et $(n+1)$ de la matrice $H_3A^{(2)}$ après la multiplication à droite, puisque nous avons $H_3^Je_1 = e_1$, $H_3^Je_2 = e_2$ et $H_3^Je_{n+1} = e_{n+1}$.

De cette manière , nous obtiendrons :

$$A^{(3)} = H_3A^{(2)}H_3^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(3)}_{(1,3:n)} & A^{(1)}_{(1,n+1)} & A^{(3)}_{(1,n+2:2n)} \\ 0 & A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2:2n)} \\ 0 & 0 & A^{(3)}_{(3:n,3:n)} & 0 & A^{(3)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2)} & A^{(3)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(3)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2,2)} & A^{(3)}_{(n+2,3:n)} & 0 & A^{(3)}_{(n+2,n+2:2n)} \\ 0 & 0 & A^{(3)}_{(n+3:2n,3:n)} & 0 & A^{(3)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

□

La quatrième étape :

Maintenant, nous voulons mettre des zéros dans les entrées de la position 4 jusqu'à la position n et de la position $n+3$ jusqu'à la position $2n$ de la $(n+2)^{\text{ième}}$ colonne de la matrice $A^{(3)}$.

Soit la sous matrice $\tilde{A}^{(3)}$ définit par :

$$\tilde{A}^{(3)} = \begin{bmatrix} A^{(3)}_{(3:n,3:n)} & A^{(3)}_{(3:n,n+2:2n)} \\ A^{(3)}_{(n+3:2n,3:n)} & A^{(3)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

Nous obtenons cette sous matrice en supprimant les $1^{\text{ère}}$, $2^{\text{ième}}$ et $(n+2)^{\text{ième}}$ lignes et les $1^{\text{ère}}$, $2^{\text{ième}}$ et $(n+1)^{\text{ième}}$ colonnes de la matrice $A^{(3)}$.

La quatrième étape consiste à répéter la deuxième étape sur la nouvelle matrice réduite $\tilde{A}^{(3)}$. Autrement dit, nous devons choisir une transformation de Householder symplectique $\tilde{H}_4$, c'est-à-dire nous devons calculer un scalaire réel $c_4 \in \mathbb{R}$ et un vecteur directeur

$$\tilde{v}_4 = \left[\frac{u_4}{w_4}\right] = A^{(4)}_{(3,n+2)} e_1 - \tilde{A}^{(3)}_{(:,n-1)} \in \mathbb{R}^{2n-4},$$

avec $u_4 \in R^{n-2}$, $w_4 \in R^{n-2}$, tels que la transformation de Householder symplectique soit définie par : $\tilde{H}_4 = I_{2n-4} + c_4 \tilde{v}_4 \tilde{v}_4^J$. En fait, cette transformation correspond à la transformation de Householder symplectique de type un T_1 du théorème 3.2.1.

Puis, nous appliquons $\tilde{H}_4$ à la matrice $\tilde{A}_3$:

$$\tilde{A}'^{(4)} = \tilde{H}_4 \tilde{A}^{(3)} = \begin{bmatrix} \tilde{A}'^{(4)}_{(3,3:n)} & A^{(4)}_{(3,n+2)} & \tilde{A}'^{(4)}_{(3,n+3:2n)} \\ \tilde{A}'^{(4)}_{(4:n,3:n)} & 0 & \tilde{A}'^{(4)}_{(4:n,n+3:2n)} \\ \tilde{A}'^{(4)}_{(n+3:2n,3:n)} & 0 & \tilde{A}'^{(4)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

Le coefficient $A^{(4)}_{(3,n+2)}$, scalaire réel non nul, est choisi arbitrairement.

De même, soit la transformation de Householder symplectique $H_4 = I_{2n} + c_4 v_4 v_4^J$, avec le vecteur directeur définit par

$$v_4 = \begin{bmatrix} 0 \\ 0 \\ u_4 \\ \hline 0 \\ 0 \\ w_4 \end{bmatrix} \in \mathbb{R}^{2n}.$$

Cette transformation préserve les 1ère, 2ème, $(n+1)$ème et $(n+2)$ème lignes et les 1ère, 2ème et $(n+1)$ème colonnes de la matrice $A'^{(4)} = H_4 A^{(3)}$. En même temps temps, elle crée les zéros désirées dans la $(n+2)$ème colonne de la matrice $A'^{(4)}$.

Ainsi, nous aurons

$$A'^{(4)} = H_4 A^{(3)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(3)}_{(1,3:n)} & A^{(1)}_{(1,n+1)} & A^{(3)}_{(1,n+2)} & A^{(3)}_{(1,n+3:2n)} \\ 0 & A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(3)}_{(2,n+3:2n)} \\ 0 & 0 & A'^{(4)}_{(3,3:n)} & 0 & A^{(4)}_{(3,n+2)} & A'^{(4)}_{(3,n+3:2n)} \\ 0 & 0 & A'^{(4)}_{(4:n,3:n)} & 0 & 0 & A'^{(4)}_{(4:n,n+3:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2)} & A^{(3)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(3)}_{(n+1,n+2)} & A^{(3)}_{(n+1,n+3:2n)} \\ 0 & A^{(2)}_{(n+2,2)} & A^{(3)}_{(n+2,3:n)} & 0 & A^{(3)}_{(n+2,n+2)} & A^{(3)}_{(n+2,n+3:2n)} \\ 0 & 0 & A'^{(4)}_{(n+3:2n,3:n)} & 0 & 0 & A'^{(4)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

Puisque nous avons $H_4^J e_i = e_i$ pour $i = 1, 2, n+1, n+2$, alors H_4^J ne modifie pas la première, la deuxième, la $(n+1)$ème et la $(n+2)$ème colonnes de la matrice $H_4 A^{(3)}$ après la multiplication à droite par la matrice H_4^J.

Nous appliquons H_4^J à la matrice $A'^{(4)}$:

$$A^{(4)} = A'^{(4)} H_4^J = H_4 A^{(3)} H_4^J =$$
$$\begin{bmatrix}
A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(4)}_{(1,3:n)} & A^{(1)}_{(1,n+1)} & A^{(3)}_{(1,n+2)} & A^{(4)}_{(1,n+3:2n)} \\
0 & A^{(3)}_{(2,2)} & A^{(4)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(4)}_{(2,n+3:2n)} \\
0 & 0 & A^{(4)}_{(3,3:n)} & 0 & A^{(4)}_{(3,n+2)} & A^{(4)}_{(3,n+3:2n)} \\
0 & 0 & A^{(4)}_{(4:n,3:n)} & 0 & 0 & A^{(4)}_{(4:n,n+3:2n)} \\
A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2)} & A^{(4)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(3)}_{(n+1,n+2)} & A^{(4)}_{(n+1,n+3:2n)} \\
0 & A^{(2)}_{(n+2,2)} & A^{(4)}_{(n+2,3:n)} & 0 & A^{(3)}_{(n+2,n+2)} & A^{(4)}_{(n+2,n+3:2n)} \\
0 & 0 & A^{(4)}_{(n+3:2n,3:n)} & 0 & 0 & A^{(4)}_{(n+3:2n,n+3:2n)}
\end{bmatrix}.$$

□

Ainsi dans la suite, à chaque fois nous répétons la première et la deuxième étapes jusqu'à la $j^{\text{ième}}$ étape.

La $j^{\text{ième}}$ étape :

D'abord, supposons que l'entier $j \in \mathbb{N}$ est un entier impaire, ainsi nous pouvons l'écrire sous la forme $j = 2j' - 1$. La $j^{\text{ième}}$ étape est maintenant claire. Au début, nous devons chercher la transformation de Householder symplectique $H_j = H_{2j'-1}$, c'est-à-dire trouvons un scalaire $c_{2j'-1} \in \mathbb{R}$ et un vecteur $v_{2j'-1} \in \mathbb{R}^{2n}$ tels que

$$H_{2j'-1} = I_{2n} + c_{2j'-1} v_{2j'-1} v^J_{2j'-1}.$$

Cette Transformation $H_{2j'-1}$ correspond à la transformation de Householder symplectique T_2 dans le théorème 3.2.1. Nous appliquons cette transformation à la matrice $A^{(j-1)} = A^{(2j'-2)}$ tel que $A'^{(2j'-1)} = H_{2j'-1} A^{(2j'-2)}$. H_j annule les entrées de la position $(j'+1)$ jusqu'à la position n et de la position $(n+j'+1)$ jusqu'à la position $2n$ de la $j'^{\text{ième}}$ colonne de la matrice $A'^{(j)}$. De même, cette transformation ne modifiée pas les lignes $1,\ldots,j'-1$, les lignes $n+1,\ldots,n+j'-1$, les colonnes $1,\ldots,j'-1$ et les colonnes $n+1,\ldots,n+j'-1$ de la matrice $H_j A^{(j-1)}$.

Le vecteur directeur $v_{2j'-1} \in \mathbb{R}^{2n}$ a la structure

$$v_{2j'-1} = \begin{bmatrix} 0 \\ \hline u_{2j'-1} \\ \hline 0 \\ w_{2j'-1} \end{bmatrix} \begin{matrix} \} & (j'-1) \\ \} & (n-j'+1) \\ \} & (j'-1) \\ \} & (n-j'+1) \end{matrix},$$

avec $u_{2j'-1} \in \mathbb{R}^{n-j'+1}$ et $w_{2j'-1} \in \mathbb{R}^{n-j'+1}$.

Remarquons que la première composante du vecteur $w_{2j'-1}$ est nulle. Ainsi, pour $i = 1,\ldots,j'$ et pour $i = n+1,\ldots,n+j'-1$, nous avons $H_{2j'-1} e_i = e_i$.

Par suite, la $j'^{\text{ième}}$ colonne de $H_{2j'-1} A^{(2j'-2)}(:,j')$ est transformée sous la forme suivante :

$$A'^{(j)}(:,j') = H_{2j'-1}A^{(2j'-2)}(:,j') = \overbrace{\begin{bmatrix} A^{(2j'-2)}(1:j'-1,j') \\ A^{(2j'-1)}(j',j') \\ 0 \\ \hline A^{(2j'-2)}(n+1:n+j',j') \\ 0 \end{bmatrix}}^{\text{la } j'^{\text{ième}} \text{ colonne de } A'^{(j)}} \begin{matrix} \} & (j'-1) \\ \} & (1) \\ \} & (n-j') \\ \hline \} & (j') \\ \} & (n-j') \end{matrix},$$

$A^{(2j'-1)}(j',j')$ est un paramètre libre.

D'après le théorème 3.2.2, soit la transformation de Householder symplectique $\tilde{H}_{2j'-1} = I_{2(n-j'+1)} + c_{2j'-1}\tilde{v}_{2j'-1}\tilde{v}^J_{2j'-1}$ sachant que le vecteur directeur $\tilde{v}_{2j'-1}$ est sous la forme

$$\tilde{v}_{2j'-1} = \begin{bmatrix} u_{2j'-1} \\ \hline w_{2j'-1} \end{bmatrix} \in \mathbb{R}^{2\alpha_{j'}},$$

tel que $u_{2j'-1} \in \mathbb{R}^{\alpha_{j'}}$ et $w_{2j'-1} \in \mathbb{R}^{\alpha_{j'}}$ où $\alpha_{j'} = n-j'+1$.

La sous matrice $\tilde{A}^{(2j'-2)}$ est obtenue en supprimant les lignes $1,\ldots,j'-1$, les lignes $n+1,\ldots,n+j'-1$, les colonnes $1,\ldots,j'-1$ et les colonnes $n+1,\ldots,n+j'-1$ de la matrice $A^{(2j'-2)}$. Aussi bien que $\tilde{A}^{(2j'-2)}(:,j')$ la $j'^{\text{ième}}$ colonne de la matrice $\tilde{A}^{(2j'-2)}$, est obtenue de la colonne $A^{(2j'-2)}(:,j')$ en supprimant les lignes $1,\ldots,j'-1$ et les lignes $n+1,\ldots,n+j'-1$.

Évidemment, nous obtenons la relation suivante

$$\tilde{H}_{(2j'-1)}\tilde{A}^{(2j'-2)}(:,j') = A^{(2j'-1)}(j',j')e_1 + A^{(2j'-2)}(n+j',j')e_{\alpha_{j'}+1}.$$

Les coefficients $A^{(2j'-1)}(j',j')$ et $A^{(2j'-2)}(n+j',j')$ sont les paramètres libres qui coïncident avec les paramètres μ et ν respectivement du théorème 3.2.1. Sachant que ici, nous notons par e_1 et $e_{\alpha_{j'}+1}$ le premier et $(\alpha_{j'}+1)^{\text{ième}}$ vecteurs de la base canonique de $\mathbb{R}^{2\alpha_{j'}}$.

Enfin, nous multiplions la matrice $H_jA^{(j-1)}$ à droite par $H^J_{2j'-1}$ pour avoir la matrice $A^{(j)} = A'^{(j)}H^J_j = H_jA^{(j-1)}H^J_j = H_{2j'-1}A^{(2j'-2)}H^J_{2j'-1}$ sous la forme souhaitée. La transformation $H^J_{2j'-1}$ préserve les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'-1$. □

La $(j+1)^{\text{ième}}$ étape :

De même dans cette étape, nous voulons annuler les entrées de la position $j'+2$ jusqu'à la position n et de la position $n+j'+1$ jusqu'à la position $2n$ de la colonne $(n+j')^{\text{ième}}$. À cet égard, nous cherchons une transformation de Householder symplectique, c'est-à-dire nous calculons un scalaire $c_{2j'}$ et un vecteur $v_{2j'} \in \mathbb{R}^{2n}$ tel que la transformation s'écrit $H_j = H_{2j'} = I_{2n} + c_{2j'}v_{2j'}v^J_{2j'}$ où $j = 2j'-1$. De cette manière, la transformation $H_{2j'}$ préserve les lignes $1,\ldots,j'$, les lignes $n+1,\ldots,n+j'$, les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'-1$ de la matrice $A'^{(2j')} = H_{2j'}A^{(2j'-1)}$. Le vecteur directeur $v_{2j'} \in \mathbb{R}^{2n}$ a la structure

$$v_{2j'} = \begin{bmatrix} 0 \\ u_{2j'} \\ \hline 0 \\ w_{2j'} \end{bmatrix} \begin{matrix} \} & (j') \\ \} & (n-j') \\ \} & (j') \\ \} & (n-j') \end{matrix},$$

avec $u_{2j'} \in \mathbb{R}^{n-j'}$ et $w_{2j'} \in \mathbb{R}^{n-j'}$.

Du coup, $H_{2j'}e_i = e_i$ pour $i = 1,\ldots,j'$ et pour $i = n+1,\ldots,n+j'$. Après la multiplication à gauche de la matrice $A^{(j)}$ par H_{j+1} pour avoir la matrice $A'^{(j+1)} = H_{2j'}A^{(2j'-1)}$, la $(n+j')^{\text{ième}}$ colonne de $H_{2j'}A^{(2j'-1)}(:,n+j')$ est transformée sous la forme suivante :

$$A'^{(j+1)}(:,n+j') = \overbrace{\left[\begin{array}{c} A^{(2j'-1)}(1:j',n+j') \\ A^{(2j')}(j'+1,n+j') \\ 0 \\ \hline A^{(2j'-1)}(n+1:n+j',n+j') \\ 0 \end{array}\right]}^{\text{la } (n+j)'^{\text{ième}} \text{ colonne de } A'^{(2j')}} \begin{array}{l} \} \quad (j') \\ \} \quad (1) \\ \} \quad (n-j'-1) \\ \hline \} \quad (j') \\ \} \quad (n-j') \end{array},$$

$A^{(2j')}(j'+1,n+j')$ est un paramètre libre.

Selon le théorème 3.2.2, soit la transformation de Householder symplectique tel que $\tilde{H}_{2j'} = I_{2(n-j')} + c_{2j'}\tilde{v}_{2j'}\tilde{v}^J_{2j'}$ avec le vecteur $\tilde{v}_{2j'}$ est sous la forme

$$\tilde{v}_{2j'} = \left[\begin{array}{c} u_{2j'} \\ \hline w_{2j'} \end{array}\right] \in \mathbb{R}^{2\beta_{j'}},$$

tel que $u_{2j'} \in \mathbb{R}^{\beta_{j'}}$ et $w_{2j'} \in \mathbb{R}^{\beta_{j'}}$ où $\beta_{j'} = n - j'$.

La sous matrice $\tilde{A}^{(2j'-1)}$ est obtenue en supprimant les lignes $1,\ldots,j'$, les lignes $n+1,\ldots,n+j'$, les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'-1$ de la matrice $A^{(2j'-1)}$. De même que $\tilde{A}^{(2j'-1)}(:,n+j')$ la $(n+j')^{\text{ième}}$ colonne de la matrice $\tilde{A}^{(2j'-1)}$, est obtenue de la colonne $A^{(2j'-1)}(:,n+j')$ en supprimant les lignes $1,\ldots,j'$ et les lignes $n+1,\ldots,n+j'$.

Nous obtenons aisément la relation suivante

$$\tilde{A}^{(2j')}(:,n+j') = \tilde{H}_{(2j')}\tilde{A}^{(2j'-1)}(:,n+j') = A^{(2j')}(j'+1,n+j')e_1.$$

Le coefficient $A^{(2j')}(j'+1,n+j')$ est le paramètre libre qui coïncide avec le paramètre ρ du théorème 3.2.1. Nous notons ici par e_1 le premier vecteur de la base canonique de $\mathbb{R}^{2\beta_{j'}}$.

À la fin, la multiplication de la matrice $H_{j+1}A^{(j)}$ à droite par H^J_{j+1} pour avoir la matrice $A^{(j+1)} = A'^{(j+1)}H^J_{j+1} = H_{j+1}A^{(j)}H^J_{j+1} = H_{2j'}A^{(2j'-1)}H^J_{2j'}$ sous la forme souhaitée, préserve les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'$. En effet, nous avons $H^J_{2j'}e_i = e_i$ pour $i = 1,\ldots,j'$ et pour $i = n+1,\ldots,n+j'$. □

Ainsi, notons que chaque deux étapes j et $j+1$ font intervenir deux paramètres libres $A^{(2j'-1)}(j',j')$ et $A^{(2j')}(j'+1,n+j')$. Ces paramètres sont trouvés dans les transformations de Householder symplectiques $H_{2j'-1}$ et $H_{2j'}$, ou d'une façon équivalente dans les transformations $\tilde{H}_{2j'-1}$ et $\tilde{H}_{2j'}$.

La dernière étape ($(2n-2)^{\text{ième}}$ étape) :

En fin, à la dernière étape, nous aurons la forme J-Hessenberg $H \in \mathbb{R}^{2n\times 2n}$ de la matrice A tel que :

$$\begin{aligned} H &= H_{2n-2}H_{2n-1}\dots H_2H_1A(H_{2n-2}H_{2n-1}\dots H_2H_1)^J \\ &= \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \in \mathbb{R}^{2n\times 2n}, \end{aligned}$$

sachant que les blocs H_{11}, H_{21} et H_{22} sont des matrices triangulaires supérieures et le bloc H_{12} est une matrice de Hessenberg supérieure.

Les entrées de la diagonale de la matrice H_{11} sont les paramètres libres (μ) de la forme $A^{(2j'-1)}(J',J')$, c'est-à-dire $H_{11}(J',J') = A^{(2j'-1)}(J',J')$ pour $J' = 1,\dots,n$. Également, les entrées de la sous-diagonale de la matrice H_{12} sont aussi les paramètres libres (ρ) de la forme $A^{(2j')}(J'+1,n+J')$, c'est-à-dire $H_{12}(J',J'+1) = A^{(2j')}(J'+1,n+J')$ pour $J' = 1,\dots,n-1$.

Ainsi, nous avons $A = S^J HS$ avec S la matrice de la transformation symplectique tel que $S = H_{2n-2}H_{2n-1}\dots H_2H_1$, la matrice S^J l'inverse de la matrice S, c'est-à-dire $S^{-1} = S^J$ tel que $S^J = (H_{2n-2}H_{2n-1}\dots H_2H_1)^J = H_1^J H_2^J \dots H_{2n-1}^J H_{2n-2}^J$ et la matrice réduite H de J-Hessenberg est sous la forme

$$H = \begin{pmatrix} \diagdown & \diagdown \\ \diagdown & \diagdown \end{pmatrix}.$$

□

3.2.2 Cas d'une matrice Hamiltonienne :

Dans le cas où la matrice de départ $A = [a_1,\dots,a_n,a_{n+1},\dots,a_{2n}] \in \mathbb{R}^{2n\times 2n}$ est une matrice Hamiltonienne, nous aurons des zéros supplémentaires qui apparaissent. Dans ce qui suit nous allons ajouter à la description précédente pour le cas d'une matrice aléatoire cet paragraphe dont le quel nous précisons seulement les emplacements de ces zéros.

La première étape :

Au commencement, la multiplication à gauche par la transformation de Householder symplectique H_1 annule les entrées de la position 2 jusqu'à position n et de la position $n+2$ jusqu'à la position $2n$ de la première colonne de la matrice A. Ainsi, nous aurons :

$$A'^{(1)} = H_1A = \begin{bmatrix} A^{(1)}_{(1,1)} & A'^{(1)}_{(1,2:n)} & A'^{(1)}_{(1,n+1)} & A'^{(1)}_{(1,n+2:2n)} \\ 0 & A'^{(1)}_{(2:n,2:n)} & A'^{(1)}_{(2:n,n+1)} & A'^{(1)}_{(2:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(0)}_{(n+1,2:n)} & A^{(0)}_{(n+1,n+1)} & A^{(0)}_{(n+1,n+2:2n)} \\ 0 & A'^{(1)}_{(n+2:2n,2:n)} & A'^{(1)}_{(n+2:2n,n+1)} & A'^{(1)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

Puisque la matrice A est Hamiltonienne, alors des zéros supplémentaires apparaissent, après la multiplication à droite par H_1^J, dans la $(n+1)^{\text{ième}}$ ligne de la matrice $A^{(1)}$ de

la position 2 jusqu'à position n et de la position $n+2$ jusqu'à la position $2n$. Donc, nous obtenons :

$$A^{(1)} = H_1 A H_1^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(1)}_{(1,n+2:2n)} \\ 0 & A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1)} & A^{(1)}_{(2:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & A^{(1)}_{(n+1,n+1)} & 0 \\ 0 & A^{(1)}_{(n+2:2n,2:n)} & A^{(1)}_{(n+2:2n,n+1)} & A^{(1)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

□

La deuxième étape :

De même, La multiplication à gauche par la transformation de Householder symplectique H_2 crée les zéros désirés dans les entrées de la position 3 jusqu'à la position n et de la position $n+2$ jusqu'à la position $2n$ de la $(n+1)^{\text{ième}}$ colonne de la matrice $A^{(1)}$. D'où, nous obtenons :

$$A'^{(2)} = H_2 A^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(1)}_{(1,n+2:2n)} \\ 0 & A'^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A'^{(2)}_{(2,n+2:2n)} \\ 0 & A'^{(2)}_{(3:n,2:n)} & 0 & A'^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & A^{(1)}_{(n+1,n+1)} & 0 \\ 0 & A'^{(2)}_{(n+2,2:n)} & 0 & A'^{(2)}_{(n+2,n+2:2n)} \\ 0 & A'^{(2)}_{(n+3:2n,2:n)} & 0 & A'^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

La multiplication de la matrice $H_2 A^{(1)}$ à droite par la transformation de Householder symplectique H_2^J crée des zéros dans les entrées de la première ligne de la matrice $A'^{(2)}$ de la position 2 jusqu'à position n et de la position $n+3$ jusqu'à la position $2n$. Ainsi, nous obtenons :

$$A^{(2)} = H_2 A^{(1)} H_2^J = \begin{bmatrix} A^{(1)}_{(1,1)} & 0 & 0 & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2)} & 0 \\ 0 & A^{(2)}_{(2,2)} & A^{(2)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A^{(2)}_{(2,n+2)} & A^{(2)}_{(2,n+3:2n)} \\ 0 & A^{(2)}_{(3:n,2)} & A^{(2)}_{(3:n,3:n)} & 0 & A^{(2)}_{(3:n,n+2)} & A^{(2)}_{(3:n,n+3:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & 0 & A^{(1)}_{(n+1,n+1)} & 0 & 0 \\ 0 & A^{(2)}_{(n+2,2)} & A^{(2)}_{(n+2,3:n)} & 0 & A^{(2)}_{(n+2,n+2)} & A^{(2)}_{(n+2,n+3:2n)} \\ 0 & A^{(2)}_{(n+3:2n,2)} & A^{(2)}_{(n+3:2n,3:n)} & 0 & A^{(2)}_{(n+3:2n,n+2)} & A^{(2)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

□

La troisième étape :

Dans cette étape, nous annulons les entrées de la position 3 jusqu'à la position n et de la position $n+3$ jusqu'à la position $2n$ de la deuxième colonne de la matrice $A^{(2)}$ par la transformation H_3. Par conséquent, nous aurons :

$$A'^{(3)} = H_3 A^{(2)} =$$
$$\begin{bmatrix} A^{(1)}_{(1,1)} & 0 & 0 & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2)} & 0 \\ 0 & A^{(3)}_{(2,2)} & A'^{(3)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A'^{(3)}_{(2,n+2)} & A'^{(3)}_{(2,n+3:2n)} \\ 0 & 0 & A'^{(3)}_{(3:n,3:n)} & 0 & A'^{(3)}_{(3:n,n+2)} & A'^{(3)}_{(3:n,n+3:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & 0 & A^{(1)}_{(n+1,n+1)} & 0 & 0 \\ 0 & A^{(2)}_{(n+2,2)} & A^{(2)}_{(n+2,3:n)} & 0 & A^{(2)}_{(n+2,n+2)} & A^{(2)}_{(n+2,n+3:2n)} \\ 0 & 0 & A'^{(3)}_{(n+3:2n,3:n)} & 0 & A'^{(3)}_{(n+3:2n,n+2)} & A'^{(3)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

La transformation H_3^J annule les entrées de la position 3 jusqu'à la position n et de la position $n+3$ jusqu'à la position $2n$ de la $(n+2)^{\text{ième}}$ ligne de la matrice $A^{(2)}$ après la multiplication à droite. De cette manière , nous obtenons :

$$A^{(3)} = H_3 A^{(2)} H_3^J =$$
$$\begin{bmatrix} A^{(1)}_{(1,1)} & 0 & 0 & A^{(1)}_{(1,n+1)} & A^{(3)}_{(1,n+2)} & 0 \\ 0 & A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(3)}_{(2,n+3:2n)} \\ 0 & 0 & A^{(3)}_{(3:n,3:n)} & 0 & A^{(3)}_{(3:n,n+2)} & A^{(3)}_{(3:n,n+3:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & 0 & A^{(1)}_{(n+1,n+1)} & 0 & 0 \\ 0 & A^{(2)}_{(n+2,2)} & 0 & 0 & A^{(3)}_{(n+2,n+2)} & 0 \\ 0 & 0 & A^{(3)}_{(n+3:2n,3:n)} & 0 & A^{(3)}_{(n+3:2n,n+2)} & A^{(3)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

□

La quatrième étape :

À travers la multiplication à gauche par la transformation H_4, nous mettons des zéros dans les entrées de la position 4 jusqu'à la position n et de la position $n+3$ jusqu'à la position $2n$ de la $(n+2)^{\text{ième}}$ colonne de la matrice $A^{(3)}$. Ainsi, nous aurons

$$A'^{(4)} = H_4 A^{(3)} =$$
$$\begin{bmatrix} A^{(1)}_{(1,1)} & 0 & 0 & A^{(1)}_{(1,n+1)} & A^{(3)}_{(2,n+2)} & 0 \\ 0 & A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:2n)} & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(3)}_{(2,n+3:2n)} \\ 0 & 0 & A'^{(4)}_{(3,3:n)} & 0 & A^{(4)}_{(3,n+2)} & A'^{(4)}_{(3,n+3:2n)} \\ 0 & 0 & A'^{(4)}_{(4:n,3:n)} & 0 & 0 & A'^{(4)}_{(4:n,n+3:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & 0 & A^{(1)}_{(n+1,n+1)} & 0 & 0 \\ 0 & A^{(2)}_{(n+2,2)} & 0 & 0 & A^{(3)}_{(n+2,n+2)} & 0 \\ 0 & 0 & A'^{(4)}_{(n+3:2n,3:n)} & 0 & 0 & A'^{(4)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

Puis, nous annulons les entrées de la position 3 jusqu'à la position n et de la position $n+4$ jusqu'à la position $2n$ de la deuxième ligne de la matrice $H_4 A^{(3)}$ en multipliant par la matrice H_4^J :

$$A^{(4)} = A'^{(4)} H_4^J = H_4 A^{(3)} H_4^J =$$

$$\begin{bmatrix}
A^{(1)}_{(1,1)} & 0 & 0 & A^{(1)}_{(1,n+1)} & A^{(3)}_{(2,n+2)} & 0 & 0 \\
0 & A^{(3)}_{(2,2)} & 0 & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(4)}_{(2,n+3)} & 0 \\
0 & 0 & A^{(4)}_{(3,3:n)} & 0 & A^{(4)}_{(3,n+2)} & A^{(4)}_{(3,n+3)} & A^{(4)}_{(3,n+4:2n)} \\
0 & 0 & A^{(4)}_{(4:n,3:n)} & 0 & 0 & A^{(4)}_{(4:n,n+3)} & A^{(4)}_{(4:n,n+4:2n)} \\
A^{(0)}_{(n+1,1)} & 0 & 0 & A^{(1)}_{(n+1,n+1)} & 0 & 0 & 0 \\
0 & A^{(2)}_{(n+2,2)} & 0 & 0 & A^{(3)}_{(n+2,n+2)} & 0 & 0 \\
0 & 0 & A^{(4)}_{(n+3:2n,3:n)} & 0 & 0 & A^{(4)}_{(n+3:2n,n+3)} & A^{(4)}_{(n+3:2n,n+4:2n)}
\end{bmatrix}.$$

□

Ainsi dans la suite, à chaque fois nous répétons la première et la deuxième étapes jusqu'à la fin, c'est-à-dire jusqu'à la $(2n-2)^{\text{ième}}$ étape.

La $(2n-2)^{\text{ième}}$ étape :

À la fin, nous obtenons une matrice sous la forme J-tridiagonale c'est-à-dire la matrice est à la fois Hamiltonienne et J-Hessenberg :

$$A^{(2n-2)} = A'^{(2n-2)} H_{2n-2}^J = H_{2n-2} A^{(2n-3)} H_{2n-2}^J =$$

$$\begin{bmatrix}
A^{(1)}_{(1,1)} & & & & A^{(1)}_{(1,n+1)} & A^{(3)}_{(2,n+2)} & & \\
& A^{(3)}_{(2,2)} & & & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(4)}_{(2,n+3)} & \\
& & \ddots & & & \ddots & \ddots & \ddots \\
& & & A^{(2n-2)}_{(n,n)} & & & A^{(2n-2)}_{(n,2n-1)} & A^{(2n-2)}_{(n,2n)} \\
A^{(0)}_{(n+1,1)} & & & & A^{(1)}_{(n+1,n+1)} & & & \\
& A^{(2)}_{(n+2,2)} & & & & A^{(3)}_{(n+2,n+2)} & & \\
& & \ddots & & & & \ddots & \\
& & & A^{(2n-2)}_{(2n,n)} & & & & A^{(2n-2)}_{(2n,2n)}
\end{bmatrix}.$$

□

Nous proposons ici l'algorithme dans sa version générale, pour calculer la réduction d'une matrice sous la forme J-Hessenberg, via des transformations de Householder symplectiques. C'est algorithme de $JHSH$.

Algorithme 3.1 Algorithme $JHSH$

Cet algorithme réduit une matrice A sous la forme J-Hessenberg tel que $H = S^J AS$ avec la matrice $S = H_1^J H_2^J \dots H_{2n-1}^J H_{2n-2}^J$ est une matrice symplectique. La matrice A est remplacées par la matrice H de J-Hessenberg.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2n}$ et $S = I_{2n}$.

SORTIE(S) : H la Matrice réduite sous la forme J-Hessenberg et S la matrice de la transformation symplectique.

1: Fonction $[S, H] = JHSH(A)$

2: $[den, dep] = size(A)$;

La suite de l'algorithme $JHSH$

3: $n = den/2$;
4: $p = dep/2$;
5: $S = eye(den)$;
6: **Pour** $j = 1 : p-1$ **faire**
7: $J = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
8: $ro = [j : n, n+j : 2n]$;
9: $co = [j : p, p+j : 2p]$;
% Calculer la constant c et le vecteur v_1 de la transformation de Householder symplectique de type deux (T_2) tel que $H_1 = I + cv_1v_1^J$.
10: $[c, v_1] = sh2(A(ro, j))$;
% Mise à jour de la matrice A.
11: $A(ro, co) = A(ro, co) + c \times v_1 \times (v_1^{\mathrm{T}} \times J \times A(ro, co))$;
12: $A(:, co) = A(:, co) - (A(:, co) \times (c \times v_1)) \times v_1^{\mathrm{T}} \times J$;
% Mise à jour de S implicitement (si nous avons besoin).
13: $S(:, co) = S(:, co) - (S(:, co) \times (c \times v_1)) \times v_1^{\mathrm{T}} \times J$;
% Mise à jour de S^J implicitement (si nous avons besoin).
%$S(ro, 2 : end) = S(ro, 2 : end) + c \times (v_1 \times v_1^{\mathrm{T}}) \times J \times S(ro, 2 : end)$;
14: $J = [zeros(n-j), eye(n-j); -eye(n-j), zeros(n-j)]$;
15: $ro = [j+1 : n, n+j+1 : 2n]$;
16: $co = [j : p, p+j : 2p]$;
17: $co1 = [j+1 : p, p+j+1 : 2p]$;
% Calculer la constant c et le vecteur v_2 de la transformation de Householder symplectique de type un (T_1) tel que $H_2 = I + cv_2v_2^J$.
18: $[c, v2] = sh1(A(ro, p+j))$;
% Mise à jour de la matrice A.
19: $A(ro, co) = A(ro, co) + c \times v_2 \times (v_2^{\mathrm{T}} \times J \times A(ro, co))$;
20: $A(:, co1) = A(:, co1) - (A(:, co1) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;
% Mise à jour de S implicitement (si nous avons besoin).
21: $S(:, co1) = S(:, co1) - (S(:, co1) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;
% Mise à jour de S^J implicitement (si nous avons besoin).
%$S(ro, 2 : end) = S(ro, 2 : end) + c \times (v_2 \times v_2^{\mathrm{T}}) \times J \times S(ro, 2 : end)$;
22: **Fin Pour**
23: $H = A$
24: **Fin**

Dans cet algorithme nous calculons les transformations de Householder symplectiques de type un $sh1$ et deux $sh2$ via les algorithmes suivantes :

L'algorithme de la transformation de Householder symplectique de type un :

Algorithme 3.2 Algorithme $sh1$

Cet algorithme calcule la constante $c_1 \in \mathbb{R}$ et un vecteur $v_1 \in \mathbb{R}^{2n}$ tel que $v_1(1) = 1$. La transformation de Householder symplectique de type un est sous la forme $T_1 = I_{2n} + c_1 v_1 v_1^J$ en vérifiant que $T_1(a) = \rho e_1$ avec ρ est un paramètre libre.

La suite de l'algorithme $sh1$

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$.
SORTIE(S) : Une constante $c_1 \in \mathbb{R}$ et un vecteur $v_1 \in \mathbb{R}^{2n}$.

Fonction $[c_1, v_1] = sh1(a)$
$den =$ taille (a);
$n = den/2$;
$J = \begin{pmatrix} 0_n & I_n \\ -I_n & 0_n \end{pmatrix}$;
Choisir le paramètre ρ
$aux = a_1 - \rho$;
Si $aux == 0$ **alors**
    $c_1 = 0$;
    $v_1 = 0_{2n}$;
    $\%T_1 = I_{2n}$;
Sinon Si $a_{n+1} == 0$ **alors**
    Stop division par zéro
Sinon
    $v_1 = \dfrac{a}{aux}$;
    $c_1 = \dfrac{aux^2}{\rho a_{n+1}}$;
    $v_1(1) = 1$
    $\%T_1 = I + c_1 v_1 v_1^J$;
Fin Si
Fin

Pareillement, l'algorithme de la transformation de Householder symplectique de type deux :

Algorithme 3.3 Algorithme $sh2$

Cet algorithme calcule la constante $c_2 \in \mathbb{R}$ et un vecteur $v_2 \in \mathbb{R}^{2n}$ tel que la transformation de Householder symplectique est sous la forme $T_2 = I_{2n} + c_2 v_2 v_2^J$ en vérifiant que $T_2(e_1) = e_1$ et $T_2(u) = \mu e_1 + \nu e_{n+1}$ avec μ est un paramètre libre.

ENTRÉE(S) : Un vecteur $u \in \mathbb{R}^{2n}$.
SORTIE(S) : Une constante $c_2 \in \mathbb{R}$ et un vecteur $v_2 \in \mathbb{R}^{2n}$.

Fonction $[c_2, v_2] = sh2(u)$
$den =$ taille (u);
$n = den/2$;
$J = \begin{pmatrix} 0_n & I_n \\ -I_n & 0_n \end{pmatrix}$;
Si $n == 1$ **alors**
    $c_2 = 0$;
    $v_2 = 0_{2n}$;
    $\%T_1 = I_{2n}$;

La suite de l'algorithme $sh2$

Sinon
  Choisir le paramètre μ
  $v = u(n+1)$;
  Si $v == 0$ **alors**
    Stop division par zéro
  Sinon
    $v_2 = \mu e_1 + v e_{n+1} - u$;
    $v_2(1) = 1$;
    $v_2(n+1) = 0$;
    $c = \dfrac{1}{u(n+1)(u(1)-\mu)}$;
  $\% T_2 = I + c_2 v_2 v_2^J$;
  Fin Si
Fin Si
Fin

L'algorithme $JHSH$ implique des paramètres libre à chaque étape. Dans ce sens nous implémentons l'algorithme $JHOSH$, qui correspond au choix optimal de ces paramètres.

Remarque 3.2.3.

1. *Dans le cas où cet algorithme est appliqué à une matrice Hamiltonienne, alors la matrice réduite qui en résulte sera sous la forme J-tridiagonale, c'est-à-dire :*

$$A = \begin{pmatrix} \diagdown & \diagup\!\!\diagup\!\!\diagup \\ \diagdown & \diagdown \end{pmatrix}.$$

2. *Grâce à la structure Hamiltonienne, les blocs* $(1,1)$ *et* $(2,2)$ *d'une matrice sous la forme Hamiltonienne J-Hessenberg sont identiques* $(A_{11} = -A_{22})$*, tandis que le bloc* $(1,2)$ *est symétrique*$(A_{12}^{\mathrm{T}} = A_{12})$*. Une matrice Hamiltonienne J-Hessenberg peut être représentée par* $4n-1$ *paramètres réels. Par conséquent, toute matrice Hamiltonienne peut être représentée par* $4n-1$ *paramètres réels ainsi que* $2n^2$ *paramètres nécessaires pour représenter la matrice de la transformation symplectique sous la forme Hamiltonienne J-Hessenberg.*
3. *Toute matrice Hamiltonienne peut être réduite en un nombre fini d'étapes sous la forme d'une matrice Hamiltonienne J-Hessenberg.*

3.2.3 La réduction J-Hessenberg via l'algorithme JHOSH

Du point de vue algébrique, l'algorithme $JHSH$ est l'analogue, dans le cas Euclidien, à l'algorithme effectuant la réduction d'une matrice sous la forme Hessenberg par des transformations de Householder. Contrairement au cas Euclidien, l'algorithme $JHSH$ comporte deux paramètres libres à chaque étape, et les transformations de Householder

symplectiques impliquées ne sont pas orthogonales. Dans la suite, nous montrons comment nous pouvons prendre l'avantage de ces paramètres libres d'une manière optimale. Afin d'obtenir un algorithme plus stable numériquement que possible, les paramètres libres seront choisis de sorte que les transformations de Householder symplectiques utilisées dans la réduction sous la forme J-Hessenberg aient des conditionnements minimaux selon la norme deux $\|.\|_2$. En d'autres termes, l'algorithme $JHOSH$ correspond à un choix efficace et optimale des paramètres libres utilisés dans l'algorithme $JHSH$. En effet, nous allons rappeler les résultats qui nous permettent de rendre les transformations utilisées plus optimales. Les conditionnements des transvections vont jouer un rôle très important. De ce fait, nous devons minimiser le conditionnement des transformations de Householder symplectiques afin de les optimiser. Pour plus de détails voir les paragraphes 1.5 et 1.5.1 dans le chapitre préliminaire.

Si $T = I + cvv^J$ est une transformation de Householder symplectique non triviale, c'est-à-dire $c \neq 0$ et $v \neq 0$, alors le conditionnement de cette transformation selon la norme deux $\|.\|_2$ est :

$$\kappa_2(T) = \frac{2 + c^2\|v\|_2^4 + \sqrt{c^4\|v\|_2^8 + 4c^2\|v\|_2^4}}{2 + c^2\|v\|_2^4 - \sqrt{c^4\|v\|_2^8 + 4c^2\|v\|_2^4}}. \tag{3.12}$$

Dans ce cas, pour minimiser le conditionnement de la transformation T il suffit de minimiser $c^2\|v\|_2^4$.

Une transformation de Householder symplectique de type un, c'est-à-dire la transformation T_1 qui transforme la première colonne de la matrice en une colonne colinéaire à e_1, a un conditionnement minimal selon la norme deux $\|.\|_2$ si $\rho = sign(a_1)\|a\|_2$ sachant que a_1 est la première composante du vecteur a.

Supposons que $u = T_1(b)$ tel que pour $i = 1,\ldots,2n$, u_i est la $i^{\text{ième}}$ composante du vecteur u. De cette manière, la condition $\rho v = a^J b$ est équivalente à $v = u_{n+1}$.

Donc pour minimiser le conditionnement de T_2, nous devons choisir le paramètre libre μ de façon optimale. Soit $\xi = \sqrt{u_2^2 + \ldots + u_n^2 + u_{n+2}^2 + \ldots + u_{2n}^2} = \|u - u_1 e_1 - u_{n+1}e_{n+1}\|_2$ Par suite, La transformation de Householder symplectique de type deux a un conditionnement minimal selon la norme deux $\|.\|_2$ si $\mu = u_1 \pm \xi$.

D'où le théorème sur lequel basé le choix des paramètres libres.

Théorème 3.2.3. *Soit* $[a,b] \in \mathbb{R}^{2n\times 2}$. *Soient*

$$\rho = sign(a_1)\|a\|_2, \qquad c_1 = -\frac{1}{\rho a^J e_1}, \qquad v_1 = \rho e_1 - a,$$

tel que

$$T_1 = I + c_1 v_1 v_1^J.$$

Alors T_1 *a un conditionnement minimal selon la norme deux* $\|.\|_2$ *et vérifie que*

$$T_1(a) = \rho e_1. \tag{3.13}$$

Soit u un vecteur tel que $u = T_1(b)$ et u_i sa $i^{ième}$ composante. Soient

$$\nu = u_{n+1}, \qquad \xi = \|u - u_1 e_1 - u_{n+1} e_{n+1}\|_2, \qquad \mu = u_1 \pm \xi,$$
$$c_2 = -\frac{1}{\pm \xi u_{n+1}}, \qquad v_2 = \mu e_1 + u_{n+1} e_{n+1} - u,$$

tel que

$$T_2 = I + c_2 v_2 v_2^J.$$

Alors T_2 a un conditionnement minimal selon la norme deux $\|.\|_2$ et vérifie que

$$T_2(e_1) = e_1, \tag{3.14}$$
$$T_2(u) = \mu e_1 + \nu e_{n+1}. \tag{3.15}$$

Les transformations T_1 et T_2 sont deux transformations de Householder symplectiques optimales.

Remarque 3.2.4. *Les vecteurs u et v_2 sont différés seulement par la premier et la $(n+1)^{ième}$ composantes.*

Pour ces choix des paramètres libres, nous nous référons à T_1 comme la première transformation de Householder symplectique optimale ($osh1$). De même, nous nous référons à T_2 comme la deuxième transformation de Householder symplectique optimale ($osh2$).

L'algorithme qui calcule la transformation de Householder symplectique optimale de type un est le suivant :

Algorithme 3.4 Algorithme $osh1$

Cet algorithme calcule la constante $c_1 \in \mathbb{R}$ et un vecteur $v_1 \in \mathbb{R}^{2n}$ tel que $v_1(1) = 1$ et la transformation de Householder symplectique optimale est sous la forme $T_1 = I + c_1 v_1 v_1^J$ en vérifiant que $T_1(a) = \rho e_1$.

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$.

SORTIE(S) : Une constante $c_1 \in \mathbb{R}$ et un vecteur $v_1 \in \mathbb{R}^{2n}$.

1: Fonction $[c_1, v_1] = osh1(a)$
2: $den =$ taille (a);
3: $n = den/2$;
4: $\rho = sign(a_1)\|a\|_2$;
5: $aux = a_1 - \rho$;
6: **Si** $aux == 0$ **alors**
7: $\quad c_1 = 0$;
8: $\quad v_1 = 0_{2n}$;
9: $\quad \%T_1 = I_{2n}$;

La suite de l'algorithme $osh1$

Sinon Si $a_{n+1} == 0$ **alors**
  Stop division par zéro
Sinon
  $v_1 = \dfrac{a}{aux}$;
  $c = \dfrac{aux^2}{\rho a_{n+1}}$;
  $v_1(1) = 1$
  $\%T_1 = I + c_1 v_1 v_1^J$;
Fin Si
Fin

De même, l'algorithme qui calcule la transformation de Householder symplectique optimale de type deux est le suivant :

Algorithme 3.5 Algorithme $osh2$

Cet algorithme calcule la constante $c_2 \in \mathbb{R}$ et un vecteur $v_2 \in \mathbb{R}^{2n}$ tel que la transformation de Householder symplectique optimale soit sous la forme $T_2 = I + c_2 v_2 v_2^J$ en vérifiant que $T_2(e_1) = e_1$ et $T_2(u) = \mu e_1 + \nu e_{n+1}$.

ENTRÉE(S) : Un vecteur $u \in \mathbb{R}^{2n}$.

SORTIE(S) : Une constante $c_2 \in \mathbb{R}$ et un vecteur $v_2 \in \mathbb{R}^{2n}$.

Fonction $[c_2, v_2] = osh2(u)$
$den =$ taille (u);
$n = den/2$;
Si $n == 1$ **alors**
  $c_2 = 0$;
  $v_2 = 0_{2n}$;
  $\%T_1 = I_{2n}$;
Sinon
  $I = [2 : n, n+2 : den]$;
  $\xi = \|u(I)\|_2$;
  Si $\xi == 0$ **alors**
    $c_2 = 0$;
    $v_2 = 0_{2n}$;
    $\%T_1 = I_{2n}$;
  Sinon
    $\nu = u(n+1)$;
    $\%\mu = u(1) + \xi$; pas besoin de calculer μ
    Si $\nu == 0$ **alors**
      Stop division par zéro
    Sinon
      $v_2 = -\dfrac{u}{\xi}$;

La suite de l'algorithme $osh2$

$v_2(1) = 1$;
$v_2(n+1) = 0$;
$c = \dfrac{\xi}{u(n+1)}$;
$\%T_2 = I + c_2 v_2 v_2^J$;
Fin Si
Fin Si
Fin Si
Fin

Remarque 3.2.5. *Dans l'algorithme précédent, nous pouvons choisir* $\xi = -\|u(I)\|_2$ *à la place de* $\xi = \|u(I)\|_2$.

La version optimale de l'algorithme $JHSH$ est appelée algorithme $JHOSH$. En effet, nous avons remplacer les transformations $sh1$ par $osh1$ et $sh2$ par $osh2$. L'algorithme $JHOSH$ est donné comme suit :

Algorithme 3.6 Algorithme JHOSH :

Cet algorithme réduit une matrice A sous la forme J-Hessenberg tel que $H = S^J AS$ avec $S = H_1^J H_2^J \dots H_{2n-1}^J H_{2n-2}^J$ est une matrice symplectique. La matrice A est remplacées par la matrice J-Hessenberg H.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2n}$ et $S = I_{2n}$.

SORTIE(S) : H la Matrice réduite sous la forme J-Hessenberg et S la matrice de transformation symplectique.

Fonction $[S,H] = JHOSH(A)$
$[den, dep] = size(A)$;
$n = den/2$;
$p = dep/2$;
$S = eye(den)$;
Pour $j = 1 : p-1$ **faire**
  $J = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
  $ro = [j : n, n+j : 2n]$;
  $co = [j : p, p+j : 2p]$;
  % Calculer la constant c et le vecteur v_1 de la transformation de Householder symplectique de type deux (T_2) tel que $H_1 = I + cv_1 v_1^J$.
  $[c, v1] = osh2(A(ro, j))$;
  % Mise à jour de la matrice A.
  $A(ro, co) = A(ro, co) + c \times v_1 \times (v_1^{\mathrm{T}} \times J \times A(ro, co))$;
  $A(:, co) = A(:, co) - (A(:, co) \times (c \times v_1)) \times v_1^{\mathrm{T}} \times J$;
  % Mise à jour de S implicitement (si nous avons besoin).
  $S(:, co) = S(:, co) - (S(:, co) \times (c \times v_1)) \times v_1^{\mathrm{T}} \times J$;
  % Mise à jour de S^J implicitement (si nous avons besoin).
  $\%S(ro, 2 : end) = S(ro, 2 : end) + c \times (v_1 \times v_1^{\mathrm{T}}) \times J \times S(ro, 2 : end)$;

La suite de l'algorithme $JHOSH$

$J = [zeros(n-j), eye(n-j); -eye(n-j), zeros(n-j)]$;
$ro = [j+1:n, n+j+1:2n]$;
$co = [j:p, p+j:2p]$;
$co1 = [j+1:p, p+j+1:2p]$;
% Calculer la constant c et le vecteur v_2 de la transformation de Householder symplectique de type un (T_1) tel que $H_2 = I + c v_2 v_2^J$.
$[c, v2] = osh1(A(ro, p+j))$;
% Mise à jour de la matrice A.
$A(ro, co) = A(ro, co) + c \times v_2 \times (v_2^{\mathrm{T}} \times J \times A(ro, co))$;
$A(:, co1) = A(:, co1) - (A(:, co1) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;
% Mise à jour de S implicitement (si nous avons besoin).
$S(:, co1) = S(:, co1) - (S(:, co1) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;
% Mise à jour de S^J implicitement (si nous avons besoin).
$\%S(ro, 2:end) = S(ro, 2:end) + c \times (v_2 \times v_2^{\mathrm{T}}) \times J \times S(ro, 2:end)$;
Fin Pour
$H = A$;
Fin

Nous avons vu que les transformations de Householder symplectiques utilisées dans l'algorithme $JHOSH$ ont des conditionnements minimaux selon la norme deux $\|.\|_2$. Donc numériquement l'algorithme $JHOSH$ présente un avantage important par rapport l'algorithme $JHSH$. Cependant, toutes ces transformations de Householder symplectiques ne sont pas orthogonaux. Il est bien connu qu'il n'est pas possible de construire une décomposition SR (respectivement la réduction sous la forme J-Hessenberg) en utilisant seulement les transformations symplectiques et orthogonaux (voir [40]).

3.3 La réduction sous la forme J-Hessenberg via l'algorithme JHMSH

Dans cette partie, nous allons montrer comment nous pouvons remplace la moitié de ces transformations de Householder symplectiques, non forcément orthogonales, par d'autres transformations élémentaires, qui ont l'avantage d'être symplectiques et orthogonales. En effet, nous allons modifié notre algorithme $JHOSH$ en remplaçons les transformations de Householder symplectiques optimales de type un par des transformations de Givens et des transformations de Householder au sens de Van Loan [114]. En réalité, nous allons remplacer les transformations H_{2j} par deux types des transformations.

Le premier type c'est

$$H(k,w) = \begin{pmatrix} \mathrm{diag}(I_{k-1,}, P) & 0 \\ 0 & \mathrm{diag}(I_{k-1}, P) \end{pmatrix},$$

avec

$$P = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w,\ w \in \mathbb{R}^{n-k+1}.$$

La transformation $H(k,w)$ est tout simplement la somme directe de deux matrices $n\times n$ de Householder ordinaires [116]. Nous appelons $H(k,w)$ la transformations de Householder au sens de Van Loan.

Le deuxième type est

$$J(k,c,s)=\begin{pmatrix} C & S \\ -S & C \end{pmatrix},$$

avec $c^2+s^2=1$, et

$$\begin{aligned} C &= \mathrm{diag}(I_{k-1}, c, I_{n-k}) \\ S &= \mathrm{diag}(0_{k-1}, s, 0_{n-k}). \end{aligned}$$

La transformation $J(k,c,s)$ est une transformation de Givens, c'est une rotation $2n\times 2n$ ordinaire de Givens dans les plans k et $n+k$ [116]. Nous nous référons à $J(k,c,s)$ par la rotations de Givens au sens de Van Loan.

Les transformations de Householder et de Givens au sens de Van Loan sont à la fois orthogonales et symplectiques. Notons que pour $i\neq k$ et $i\neq n+k$, nous avons $J(k,c,s)e_i=e_i$. Nous avons aussi, $J(k,c,s)e_k=ce_k-se_{n+k}$ et $J(k,c,s)e_{n+k}=se_k+ce_{n+k}$. C'est pourquoi $J(k,c,s)$ ne modifié pas toutes les lignes de la matrice, à la quelle la matrice $J(k,c,s)$ a été multiplié, sauf la $k^{\text{ième}}$ et $(n+k)^{\text{ième}}$ lignes. Il est évident aussi que $H(k,w)e_i=e_i$ pour $i=1,\ldots,k-1$ et pour $i=n+1,\ldots,n+k-1$.

La modification des étapes paires de l'algorithme $JHOSH$ est comme suit.

Soit la matrice $A=[a_1,\ldots,a_n,a_{n+1},\ldots,a_{2n}]\in\mathbb{R}^{2n\times 2n}$ et posons que $A^{(0)}=A$.

La première étape :

En premier lieu, comme dans l'algorithme de $JHOSH$ nous cherchons une transformation de Householder symplectique H_1 pour annuler les entrées de la position 2 jusqu'à position n et de la position $n+2$ jusqu'à la position $2n$ de la première colonne de la matrice A. Cette transformation $H_1=I-c_1v_1v_1^J$ est tels que $c_1\in\mathbb{R}$ et $v_1\in\mathbb{R}^{2n}$. Le vecteur v_1 est le vecteur directeur de H_1. La transformation H_1 correspond à la transformation T_2, définit dans le théorème 3.2.1. De plus, $H_1e_1=e_1$, alors $v_1^Je_1=v_1^{\mathrm{T}}Je_1=0$. Par suite, l'entrée de la position $(n+1)^{\text{ième}}$ de vecteur v_1 est égale à zéro. Ainsi, pour tout vecteur x, l'entrée de la position $(n+1)^{\text{ième}}$ de vecteur H_1x est préserve. Le vecteur v_1 est donnée par :

$$v_1=A^{(1)}_{1,1}e_1+a_1(n+1)e_{n+1}-a_1$$

avec $A^{(1)}_{1,1}$ est un scalaire arbitraire. D'autre part, nous avons $H_1^Je_1=e_1$, ce qui entraîne que la première colonne de H_1 et de H_1^J est égale à e_1. De cette manière, la multiplication de la matrice A à gauche par H_1 préserve la $(n+1)^{\text{ième}}$ ligne et crée les zéros désirés dans la première colonne. Ainsi, nous obtenons

$$A'^{(1)}=H_1A=\begin{bmatrix} A^{(1)}_{(1,1)} & A'^{(1)}_{(1,2:n)} & A'^{(1)}_{(1,n+1:2n)} \\ 0 & A'^{(1)}_{(2:n,2:n)} & A'^{(1)}_{(2:n,n+1:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(0)}_{(n+1,2:n)} & A^{(0)}_{(n+1,n+1:2n)} \\ 0 & A'^{(1)}_{(n+2:2n,2:n)} & A'^{(1)}_{(n+2:2n,n+1:2n)} \end{bmatrix}.$$

Puis, la multiplication de $H_1 A$ à droite par H_1^J préserve la première colonne de $H_1 A H_1^J$ et nous aurons :

$$A^{(1)} = H_1 A H_1^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1:2n)} \\ 0 & A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1:2n)} \\ 0 & A^{(1)}_{(n+2:2n,2:n)} & A^{(1)}_{(n+2:2n,n+1:2n)} \end{bmatrix}.$$

□

La deuxième étape :

Maintenant, pour créer les zéros souhaitées dans la $(n+1)^{\text{ième}}$ colonne et en conservant la première colonne inchangée, nous utiliserons des transformations de Van Loan, au lieu de la transformation de Householder symplectique optimale de type un H_2. Nous calculons les transformations de Givens au sens de Van Loan $J(k,c,s)$ pour $k = n,\ldots,2$, tels qu'un zéro est créé dans l'entrée de position $n+k$ dans la $(n+1)^{\text{ième}}$ colonne de la matrice $J(k,c,s)A^{(1)}$. La première colonne ainsi que les zéros déjà créés dans la courante $(n+1)^{\text{ième}}$ colonne de la matrice $A^{(1)}$ restent inchangées. La première et la $(n+1)^{\text{ième}}$ colonne de la matrice $J(k,c,s)A^{(1)}$ sont préservées lorsque cette dernière est multipliée à droite par la matrice $J(k,c,s)^{\mathrm{T}}$. Ensuite, la matrice $A^{(1)}$ est actualisée par la matrice $A'^{(2)} = J(2,c,s)\ldots J(n,c,s)A^{(1)}J(n,c,s)^{\mathrm{T}}\ldots J(2,c,s)^{\mathrm{T}}$. Ainsi, les entrées aux positions $n+2,\ldots,2n$ dans la $(n+1)^{\text{ième}}$ colonne de la matrice $A'^{(2)}$ sont toutes des zéros. D'où,

$$\begin{aligned} A'^{(2)} &= J(2,c,s)\ldots J(n,c,s)A^{(1)}J(n,c,s)^{\mathrm{T}}\ldots J(2,c,s)^{\mathrm{T}} \\ &= \begin{bmatrix} A^{(1)}_{(1,1)} & A'^{(2)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A'^{(2)}_{(1,n+2:2n)} \\ 0 & A'^{(2)}_{(2,2:n)} & A'^{(2)}_{(2,n+1)} & A'^{(2)}_{(2,n+2:2n)} \\ 0 & A'^{(2)}_{(3:n,2:n)} & A'^{(2)}_{(3:n,n+1)} & A'^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A'^{(2)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A'^{(2)}_{(n+1,n+2:2n)} \\ 0 & A'^{(2)}_{(n+2,2:n)} & 0 & A'^{(2)}_{(n+2,n+2:2n)} \\ 0 & A'^{(2)}_{(n+3:2n,2:n)} & 0 & A'^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}. \end{aligned}$$

Après, nous calculons le vecteur w de la transformation de Householder au sens de Van Loan où $H(2,w) = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w$, de sorte que la multiplication de la matrice $A'^{(2)}$ à gauche par la matrice $H(2,w)$ crée des zéros dans les entrées des positions $3,\ldots,n$ de la $(n+1)^{\text{ième}}$ colonne. La première colonne de la matrice $A'^{(2)}$ ainsi que les zéros déjà créés restent inchangés. La transformation $H(2,w)^{\mathrm{T}}$ préserve la première et la $(n+1)^{\text{ième}}$ co-

lonne de la matrice $A^{(2)} = H(2,w)A'^{(2)}H(2,w)^{\mathrm{T}}$. Ainsi,

$$A^{(2)} = H(2,w)A'^{(2)}H(2,w)^{\mathrm{T}}$$
$$= \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2:2n)} \\ 0 & A^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A^{(2)}_{(2,n+2:2n)} \\ 0 & A^{(2)}_{(3:n,2:n)} & 0 & A^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(2)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2,2:n)} & 0 & A^{(2)}_{(n+2,n+2:2n)} \\ 0 & A^{(2)}_{(n+3:2n,2:n)} & 0 & A^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

□

La $j^{\text{ième}}$ étape :

Dans la $j^{\text{ième}}$ étape où $j = 2j'-1 \in \mathbb{N}$, nous créons les zéros souhaitées dans la colonne j', par la transformation de Householder symplectique $H_j = H_{2j'-1}$. Autrement dit, c'est comme dans la $j^{\text{ième}}$ étape de l'algorithme $JHOSH$, c'est-à-dire nous cherchons un scalaire $c_{2j'-1} \in \mathbb{R}$ et un vecteur $v_{2j'-1} \in \mathbb{R}^{2n}$ tels que $H_{2j'-1} = I_{2n} + c_{2j'-1}v_{2j'-1}v^J_{2j'-1}$. $H_{2j'-1}$ correspond à la transformation de Householder symplectique T_2 dans le théorème 3.2.3. Nous appliquons cette transformation à la matrice $A^{(j-1)} = A^{(2j'-2)}$ pour avoir $H_{2j'-1}A^{(2j'-2)}$. La matrice H_j annule les entrées de la position $(j'+1)$ jusqu'à la position n et de la position $(n+j'+1)$ jusqu'à la position $2n$ de la $j'^{\text{ième}}$ colonne de la matrice $H_{2j'-1}A^{(2j'-2)}$. En effet, cette transformation ne modifiée pas les lignes $1,\ldots,j'-1$, les lignes $n+1,\ldots,n+j'-1$, les colonnes $1,\ldots,j'-1$ et les colonnes $n+1,\ldots,n+j'-1$ de la matrice $H_jA^{(j-1)}$.

Le vecteur directeur $v_{2j'-1} \in \mathbb{R}^{2n}$ a la structure suivante

$$v_{2j'-1} = \begin{bmatrix} 0 \\ u_{2j'-1} \\ 0 \\ w_{2j'-1} \end{bmatrix} \begin{matrix} \} & (j'-1) \\ \} & (n-j'+1) \\ \} & (j'-1) \\ \} & (n-j'+1) \end{matrix},$$

avec $u_{2j'-1} \in \mathbb{R}^{n-j'+1}$ et $w_{2j'-1} \in \mathbb{R}^{n-j'+1}$.

Notons que la $(n+j')^{\text{ième}}$ composante du vecteur $v_{2j'-1}$ est nulle. Donc, pour $i = 1,\ldots,j'$ et pour $i = n+1,\ldots,n+j'-1$, nous avons $H_{2j'-1}e_i = e_i$.

De ce fait, la $j'^{\text{ième}}$ colonne de $H_{2j'-1}A^{(2j'-2)}(:,j')$ est transformée sous la forme suivante :

$$H_{2j'-1}A^{(2j'-2)}(:,j') = \begin{bmatrix} \overbrace{\begin{matrix} A^{(2j'-2)}(1:j'-1,j') \\ A^{(2j'-1)}(j',j') \\ 0 \end{matrix}}^{\text{la } j'^{\text{ième}} \text{ colonne de } A'^{(j)}} \\ A^{(2j'-2)}(n+1:n+j',j') \\ 0 \end{bmatrix} \begin{matrix} \} & (j'-1) \\ \} & (1) \\ \} & (n-j') \\ \} & (j') \\ \} & (n-j') \end{matrix}.$$

Soit la transformation de Householder symplectique

$$\tilde{H}_{2j'-1} = I_{2(n-j'+1)} + c_{2j'-1}\,\tilde{v}_{2j'-1}\,\tilde{v}^J_{2j'-1},$$

avec le vecteur directeur $\tilde{v}_{2j'-1}$ tel que

$$\tilde{v}_{2j'-1} = \begin{bmatrix} u_{2j'-1} \\ \hline w_{2j'-1} \end{bmatrix} \in \mathbb{R}^{2\alpha_{j'}},$$

avec $u_{2j'-1} \in \mathbb{R}^{\alpha_{j'}}$ et $w_{2j'-1} \in \mathbb{R}^{\alpha_{j'}}$ où $\alpha_{j'} = n - j' + 1$.

La matrice $\tilde{A}^{(2j'-2)}$ est obtenue en supprimant les lignes $1,\ldots,j'-1$, les lignes $n+1,\ldots,n+j'-1$, les colonnes $1,\ldots,j'-1$ et les colonnes $n+1,\ldots,n+j'-1$ de la matrice $A^{(2j'-2)}$. De même, $\tilde{A}^{(2j'-2)}(:,j')$ la $j'^{\text{ième}}$ colonne de la matrice $\tilde{A}^{(2j'-2)}$, est obtenue de la colonne $A^{(2j'-2)}(:,j')$ en supprimant les lignes $1,\ldots,j'-1$ et les lignes $n+1,\ldots,n+j'-1$.

Donc, nous obtenons

$$\tilde{H}_{(2j'-1)}\tilde{A}^{(2j'-2)}(:,j') = A^{(2j'-1)}(j',j')e_1 + A^{(2j'-2)}(n+j',j')e_{\alpha_{j'}+1}.$$

Les coefficients $A^{(2j'-1)}(j',j')$ et $A^{(2j'-2)}(n+j',j')$ sont les paramètres libres qui coïncident avec les paramètres μ et ν respectivement du théorème 3.2.3. Nous notons ici par e_1 et $e_{\alpha_{j'}+1}$ le premier et $(\alpha_{j'}+1)^{\text{ième}}$ vecteurs de la base canonique de $\mathbb{R}^{2\alpha_{j'}}$.

Dernièrement, nous multiplions la matrice $H_j A^{(j-1)}$ à droite par $H^J_{2j'-1}$ pour avoir la matrice $A^{(j)} = A'^{(j)}H^J_j = H_j A^{(j-1)}H^J_j = H_{2j'-1}A^{(2j'-2)}H^J_{2j'-1}$ sous la forme désirée. La transformation $H^J_{2j'-1}$ préserve les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'-1$. □

La $(j+1)^{\text{ième}}$ étape :

Maintenant dans cette étape, nous souhaitons créer les zéros désirés dans la $(n+j')^{\text{ième}}$ colonne ($j = 2j'-1 \in \mathbb{N}$) en utilisant des rotations de Givens au sens de Van Loan, au lieu de la transformation de Householder symplectique optimale $H_{2j'}$. Ainsi, pour $k = n,\ldots,j'+1$, nous calculons $J(k,c,s)$ afin de créer un zéro dans l'entrée de la position $n+k$ de la $(n+j')^{\text{ième}}$ colonne de la matrice $J(k,c,s)A^{(2j'-1)}$. Cette multiplication ne modifiée pas les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'-1$ ainsi que les zéros déjà créés dans la courante $(n+j')^{\text{ième}}$ colonne de la matrice $A^{(2j'-1)}$. Également, la multiplication de la matrice $J(k,c,s)A^{(2j'-1)}$ à droite par la matrice $J(k,c,s)^{\mathrm{T}}$ ne modifiée pas les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'-1$ de la matrice $J(k,c,s)A^{(2j'-1)}$. En conséquence, la matrice $A^{(2j'-1)}$ est ensuite actualisée par la matrice

$$A'^{(2j')} = J(j'+1,c,s)\ldots J(n,c,s)A^{(2j'-1)}J(n,c,s)^{\mathrm{T}}\ldots J(j'+1,c,s)^{\mathrm{T}}.$$

Ainsi les entrées aux positions $n+j'+1,\ldots,2n$ de la $(n+j')^{\text{ième}}$ colonne de la matrice $A'^{(2j')}$ sont des zéros.

De cette manière, la $(n+j')^{\text{ième}}$ colonne de $A'^{(2j')}(:,n+j')$ est transformée sous la forme

suivante :

$$A'^{(2j')}(:,n+j') = \overbrace{\begin{bmatrix} A^{(2j'-1)}(1:j',n+j') \\ A'^{(2j')}(j'+1:n,n+j') \\ \hline A^{(2j'-1)}(n+1:n+j',n+j') \\ 0 \end{bmatrix}}^{\text{la } (n+j')^{\text{ième}} \text{ colonne de } A'^{(2j')}} \begin{matrix} \} & (j') \\ \} & (n-j') \\ \hline \} & (j') \\ \} & (n-j') \end{matrix}.$$

Puis, nous calculons le vecteur w de la transformation de Householder au sens de Van Loan $H(j',w) = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w$ afin de créer des zéros dans les entrées aux positions $j'+2,\ldots,n$ de la $(n+j')^{\text{ième}}$ colonne de la matrice $A'^{(2j')}$ après la multiplication à gauche par la matrice $H(j',w)$. Cette multiplication ne modifiée pas les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'-1$ ainsi que les zéros déjà créés dans la $(n+j')^{\text{ième}}$ colonne de la matrice $A'^{(2j')}$. Aussi après la multiplication à droite, la matrice $H(j,w)^{\mathrm{T}}$ ne modifiée pas les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'$ de la matrice actualisée $A^{(2j')} = H(j',w)A'^{(2j')}H(j',w)^{\mathrm{T}}$.

De cette façon, la $(n+j')^{\text{ième}}$ colonne de $A^{(2j')}(:,n+j')$ est transformée sous la forme suivante :

$$A^{(2j')}(:,n+j') = \overbrace{\begin{bmatrix} A^{(2j'-1)}(1:j',n+j') \\ A^{(2j')}(j'+1,n+j') \\ 0 \\ \hline A^{(2j'-1)}(n+1:n+j',n+j') \\ 0 \end{bmatrix}}^{\text{la } (n+j')^{\text{ième}} \text{ colonne de } A^{(2j')}} \begin{matrix} \} & (j') \\ \} & (1) \\ \} & (n-j'-1) \\ \hline \} & (j') \\ \} & (n-j') \end{matrix}.$$

□

Ainsi de la même manière, nous répétons les deux étapes précédentes c'est-à-dire la $j^{\text{ième}}$ et la $(j+1)^{\text{ième}}$ étapes jusqu'à ce que $j+1 = 2n-2$ afin d'obtenir la forme réduite souhaitée.

3.3.1 Cas d'une matrice Hamiltonienne :

De la même manière, dans le cas où $A = [a_1,\ldots,a_n,a_{n+1},\ldots,a_{2n}] \in \mathbb{R}^{2n\times 2n}$ la matrice de départ est une matrice Hamiltonienne, nous aurons des zéros supplémentaires qui apparaissent. Ainsi, dans cette paragraphe nous précisons les emplacements de ces zéros supplémentaires sachant que la description précédente pour le cas d'une matrice aléatoire reste encore valable pour le cas d'une matrice Hamiltonienne.

La première étape :

Au début, nous annulons les entrées de la position 2 jusqu'à position n et de la position $n+2$ jusqu'à la position $2n$ de la première colonne de la matrice A par la transforma-

tion de Householder symplectique H_1. Ainsi, nous aurons :

$$A'^{(1)} = H_1 A = \begin{bmatrix} A^{(1)}_{(1,1)} & A'^{(1)}_{(1,2:n)} & A'^{(1)}_{(1,n+1)} & A'^{(1)}_{(1,n+2:2n)} \\ 0 & A'^{(1)}_{(2:n,2:n)} & A'^{(1)}_{(2:n,n+1)} & A'^{(1)}_{(2:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(0)}_{(n+1,2:n)} & A^{(0)}_{(n+1,n+1)} & A^{(0)}_{(n+1,n+2:2n)} \\ 0 & A'^{(1)}_{(n+2:2n,2:n)} & A'^{(1)}_{(n+2:2n,n+1)} & A'^{(1)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

Après la multiplication à droite par H_1^J, les zéros apparaissent dans la $(n+1)^{\text{ième}}$ ligne de la matrice $A^{(1)}$ de la position 2 jusqu'à position n et de la position $n+2$ jusqu'à la position $2n$. Donc, nous obtenons :

$$A^{(1)} = H_1 A H_1^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(1)}_{(1,n+2:2n)} \\ 0 & A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1)} & A^{(1)}_{(2:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & A^{(1)}_{(n+1,n+1)} & 0 \\ 0 & A^{(1)}_{(n+2:2n,2:n)} & A^{(1)}_{(n+2:2n,n+1)} & A^{(1)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

□

La deuxième étape :

Nous calculons les rotations de Givens au sens de Van Loan $J(k,c,s)$ pour $k = n,\ldots,2$, tels qu'un zéro est créé dans l'entrée de la position $n+k$ dans la $(n+1)^{\text{ième}}$ colonne de la matrice $J(k,c,s)A^{(1)}$ c'est-à-dire $J(k,c,s)A^{(1)}(n+k,n+1) = 0$. Ensuite, nous multiplions la matrice $J(k,c,s)A^{(1)}$ à droite par la matrice $J(k,c,s)^{\mathrm{T}}$. Ainsi, un zéro est créé dans l'entrée de la position k de la première ligne c'est-à-dire $J(k,c,s)A^{(1)}J(k,c,s)^{\mathrm{T}}(1,k) = 0$. Pour $k = n,\ldots,2$, la matrice $A^{(1)}$ est actualisée par la matrice

$$A'^{(2)} = J(2,c,s)\ldots J(n,c,s)A^{(1)}J(n,c,s)^{\mathrm{T}}\ldots J(2,c,s)^{\mathrm{T}}.$$

De ce fait, les entrées aux positions $n+2,\ldots,2n$ dans la $(n+1)^{\text{ième}}$ colonne de la matrice $A'^{(2)}$ sont toutes des zéros. Aussi, les entrées aux positions $2,\ldots,n$ de la première ligne de la matrice $A'^{(2)}$ sont toutes des zéros. D'où, nous obtenons

$$\begin{aligned} A'^{(2)} &= J(2,c,s)\ldots J(n,c,s)A^{(1)}J(n,c,s)^{\mathrm{T}}\ldots J(2,c,s)^{\mathrm{T}} \\ &= \begin{bmatrix} A^{(1)}_{(1,1)} & 0 & A^{(1)}_{(1,n+1)} & A'^{(2)}_{(1,n+2:2n)} \\ 0 & A'^{(2)}_{(2,2:n)} & A'^{(2)}_{(2,n+1)} & A'^{(2)}_{(2,n+2:2n)} \\ 0 & A'^{(2)}_{(3:n,2:n)} & A'^{(2)}_{(3:n,n+1)} & A'^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & A^{(1)}_{(n+1,n+1)} & 0 \\ 0 & A'^{(2)}_{(n+2,2:n)} & 0 & A'^{(2)}_{(n+2,n+2:2n)} \\ 0 & A'^{(2)}_{(n+3:2n,2:n)} & 0 & A'^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}. \end{aligned}$$

Après, la multiplication de la matrice $A'^{(2)}$ à gauche par la matrice de la transformation de Householder au sens de Van Loan $H(2,w)$, crée des zéros dans les entrées des positions $3,\ldots,n$ de la $(n+1)^{\text{ième}}$ colonne. La multiplication à droite par la transformation $H(2,w)^{\mathrm{T}}$

crée des zéros dans les entrées des positions $n+3,\ldots,2n$ de la première ligne de la matrice $A^{(2)} = H(2,w)A'^{(2)}H(2,w)^{\mathrm{T}}$. Ainsi, nous avons

$$A^{(2)} = H(2,w)A'^{(2)}H(2,w)^{\mathrm{T}}$$
$$= \begin{bmatrix} A^{(1)}_{(1,1)} & 0 & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2)} & A^{(2)}_{(1,n+3:2n)} \\ 0 & A^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A^{(2)}_{(2,n+2)} & A^{(2)}_{(2,n+3:2n)} \\ 0 & A^{(2)}_{(3:n,2:n)} & 0 & A^{(2)}_{(3:n,n+2)} & A^{(2)}_{(3:n,n+3:2n)} \\ A^{(0)}_{(n+1,1)} & 0 & A^{(1)}_{(n+1,n+1)} & 0 & 0 \\ 0 & A^{(2)}_{(n+2,2:n)} & 0 & A^{(2)}_{(n+2,n+2)} & A^{(2)}_{(n+2,n+3:2n)} \\ 0 & A^{(2)}_{(n+3:2n,2:n)} & 0 & A^{(2)}_{(n+3:2n,n+2)} & A^{(2)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

□

Ainsi de la même façon dans la suite, à chaque fois nous répétons ces deux étapes jusqu'à la fin, c'est-à-dire jusqu'à la $(2n-2)^{\text{ième}}$ étape.

La $(2n-2)^{\text{ième}}$ étape :

En dernier ressort, nous obtenons une matrice réduite sous la forme J-tridiagonale c'est-à-dire que la matrice est à la fois Hamiltonienne et J-Hessenberg :

$$A^{(2n-2)} = A'^{(2n-2)}H^J_{2n-2} = H_{2n-2}A^{(2n-3)}H^J_{2n-2} =$$
$$\begin{bmatrix} A^{(1)}_{(1,1)} & & & & A^{(1)}_{(1,n+1)} & A^{(3)}_{(2,n+2)} & & \\ & A^{(3)}_{(2,2)} & & & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(4)}_{(2,n+3)} & \\ & & \ddots & & & \ddots & \ddots & \ddots \\ & & & A^{(2n-2)}_{(n,n)} & & & A^{(2n-2)}_{(n,2n-1)} & A^{(2n-2)}_{(n,2n)} \\ A^{(0)}_{(n+1,1)} & & & & A^{(1)}_{(n+1,n+1)} & & & \\ & A^{(2)}_{(n+2,2)} & & & & A^{(3)}_{(n+2,n+2)} & & \\ & & \ddots & & & & \ddots & \\ & & & A^{(2n-2)}_{(2n,n)} & & & & A^{(2n-2)}_{(2n,2n)} \end{bmatrix}.$$

□

Par suite, l'algorithme $JHMSH$ de la réduction d'une matrice arbitraire de taille $2n \times 2n$ sous la forme de J-Hessenberg est le suivant :

Algorithme 3.7 Algorithme $JHMSH$

Cet algorithme réduit une matrice A sous la forme J-Hessenberg $H = S^J AS$ avec S est une matrice symplectique.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2n}$ et $S = I_{2n}$.

SORTIE(S) : H la Matrice réduite sous la forme J-Hessenberg et S la matrice de transformation symplectique.

1: Fonction $[S,H] = JHMSH(A)$
2: $[den, dep] = size(A)$;

La suite de l'algorithme $JHMSH$

3: $n = den/2$;
4: $p = dep/2$;
5: $S = eye(den)$;
6: **Pour** $j = 1, \ldots, n-1$ **faire**
7: $J_{2(n-j+1)} = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
8: $ro = [j : n, n+j : 2n]$;
9: $co = [j : p, p+j : 2p]$;
% Nous mettons des zéros dans $j^{\text{ième}}$ colonne de la matrice A en appliquant la transformation de Householder symplectique optimale osh_2 pour calculer la constante c et le vecteur v de la matrice de transformation $T_{osh_2} = I + cvv^J$.
10: $[c, v] = osh_2(A(ro, j))$
% Mise à jour de la matrice A.
11: $A(ro, co) = A(ro, co) + c \times v \times (v^{\mathrm{T}} \times J_{2(n-j+1)} \times A(ro, co))$;
12: $A(:, co) = A(:, co) - (A(:, co) \times (c \times v)) \times v^{\mathrm{T}} \times J_{2(n-j+1)}$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
13: $S(:, co) = S(:, co) - (S(:, co) \times (c \times v)) \times v^{\mathrm{T}} \times J_{2(n-j+1)}$;
14: **Pour** $k = n, n-1, \ldots, j+1$ **faire**
% Nous mettons un zéro à l'entrée de la position $(n+k, n+j)$ de la matrice A en appliquant l'algorithme vlg pour déterminer les constantes c et s de la matrice $J(k, c, s)$.
15: $[c, s] = vlg(k, A(:, n+j))$;
% Mise à jour de la matrice A.
16: $vlA = A(k, :)$;
17: $A(k, :) = c \times A(k, :) + s \times A(n+k, :)$;
18: $A(n+k, :) = -s \times vlA + c \times A(n+k, :)$;
19: $vcA = A(:, k)$;
20: $A(:, k) = c \times A(:, k) + s \times A(:, n+k)$;
21: $A(:, n+k) = -s \times vcA + c \times A(:, n+k)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
22: $vcS = S(:, k)$;
23: $S(:, k) = c \times S(:, k) + s \times S(:, n+k)$;
24: $S(:, n+k) = -s \times vcS + c \times S(:, n+k)$;
25: **Fin Pour**
26: **Si** $j \leq n-2$ **alors**
% Nous mettons des zéros aux entrées des positions $(j+2, n+j), (j+3, n+j), \ldots, (n, n+j)$ de la matrice A en appliquant l'algorithme vlh pour déterminer la constante β et le vecteur w de la matrice $H(j+1, w)$.
27: $[\beta, w] = vlh(j+1, A(:, n+j))$;
% Mise à jour de la matrice A.
28: $A(j+1 : n, co) = A(j+1 : n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1 : n, co)$;
29: $A(j+1+n : 2n, co) = A(j+1+n : 2n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1+n : 2n, co)$;
30: $A(:, j+1 : n) = A(:, j+1 : n) - \beta \times A(:, j+1 : n) \times (w \times w^{\mathrm{T}})$;
31: $A(:, n+j+1 : 2n) = A(:, n+j+1 : 2n) - \beta \times A(:, n+j+1 : n) \times (w \times w^{\mathrm{T}})$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
32: $S(:, j+1 : n) = S(:, j+1 : n) - \beta \times S(:, j+1 : n) \times (w \times w^{\mathrm{T}})$;

La suite de l'algorithme $JHMSH$

$S(:, n+j+1:2n) = S(:, n+j+1:2n) - \beta \times S(:, n+j+1:2n) \times (w \times w^{\mathrm{T}})$;
Fin Si
Fin Pour
$H = A$;
Fin

L'algorithme de la rotation de Givens au sens de Van Loan qui calcule les constantes c et s de la matrice $J(k,c,s)$ est le suivant :

Algorithme 3.8 Algorithme vlg

Cet algorithme calcule les constantes c et s de la matrice $J(k,c,s)$ de la rotation de Givens au sens de Van Loan avec

$$J(k,c,s) = \begin{pmatrix} I_{k-1} & & & & & \\ & c & & & s & \\ & & I_{n-k} & & & \\ & & & I_{k-1} & & \\ & -s & & & c & \\ & & & & & I_{n-k} \end{pmatrix}, \text{ où } c^2 + s^2 = 1.$$

La matrice $J(k,c,s)$ annule l'entrée de la position $n+k$ du vecteur a, c'est-à-dire si $b = J(k,c,s)a$, alors $b(n+k) = 0$.

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$.

SORTIE(S) : Les constantes c et s de la matrice de rotation de Givens au sens de Van Loan tel que $c^2 + s^2 = 1$ et l'entrée de la position $n+k$ du vecteur $J(k,c,s)a$ est nulle.

Fonction $[c,s] = vlg(k,a)$
$den = length(a)$;
$n = den/2$;
$r = \sqrt{a(k)^2 + a(n+k)^2}$;
Si $r = 0$ **alors**
$c = 1$;
$s = 0$;
Sinon
$c = \dfrac{a_k}{r}$;
$s = \dfrac{a_{n+k}}{r}$;
Fin Si
Fin

L'algorithme de la transformation de Householder au sens de Van Loan qui calcule la constante β et le vecteur $w = (w_1, \ldots, w_{n-k+1})^{\mathrm{T}} \in \mathbb{R}^{n-k+1}$ de la matrice $H(k,w)$ est le suivant :

Algorithme 3.9 Algorithme vlh :

Cet algorithme calcule la constante β et le vecteur $w = (w_1, \ldots, w_{n-k+1})^{\mathrm{T}} \in \mathbb{R}^{n-k+1}$ de la matrice $H(k,w)$ de la transformation de Householder au sens de Van Loan avec $H(k,w) = \begin{pmatrix} \mathrm{diag}(I_{k-1}, P) & 0 \\ 0 & \mathrm{diag}(I_{k-1}, P) \end{pmatrix}$ où $P = I_{n-k+1} + \beta w w^{\mathrm{T}} = I_{n-k+1} + 2ww^{\mathrm{T}}/w^{\mathrm{T}}w$. La matrice $H(k,w)$ annule les entrées aux positions $k+1, \ldots, n$ du vecteur a, c'est-à-dire si $b = H(k,w)a$, alors $b(k+1:n) = 0$.

ENTRÉE(S) : Un vecteur $a \in \mathbb{R}^{2n}$.

SORTIE(S) : La constantes β et le vecteur w de la matrice de transformation de Householder au sens de Van Loan tel que pour $b = H(k,w)a$, nous avons $b(k+1:n) = 0$.

1: Fonction $[\beta, w] = vlh(k,a)$
2: $den = length(a)$;
3: $n = den/2$;
4: $r_1 = \sum_{i=2}^{n-k+1} a(i+k-1)^2$;
5: $r = \sqrt{a(k)^2 + r_1}$;
6: $w_1 = a(k) + sign(a(k))r$;
7: **Pour** $k = 2, \ldots, n-k+1$ **faire**
8: $\quad w_i = a_{i+k-1}$;
9: **Fin Pour**
10: $r = w_1^2 + r_1$;
11: **Si** $r \neq 0$ **alors**
12: $\quad \beta = \dfrac{2}{r}$;
13: **Sinon**
14: $\quad beta = 0$;
15: **Fin Si**
$\%P = I + \beta w w^{\mathrm{T}}$ tel que $(H(k,w)a)_i = 0$ pour $i = k+1, \ldots, n$.
16: **Fin**

3.4 L'algorithme de réduction sous la forme J-Hessenberg JHMSH2

L'algorithme de la réduction d'une matrice sous forme J-Hessenberg $JHESS$ de Bunse-Gerstner et Mehrmann a une instabilité numérique. L'origine de cette dernière est clairement identifiée vu qu'elle survient lorsqu'une transformation de Gauss $G(k,v)$ est rencontrée avec un conditionnement très grand. Avec l'objectif de rendre $JHESS$ plus stable numériquement, nous avons proposé une modification de cet algorithme pour remédies à ses instabilités numériques. En outre, la version de l'algorithme $JHESS$ modifié va correspondre à notre algorithme $JHMSH$ modifié c'est-à-dire l'algorithme $JHMSH2$. En réalité, nous allons limiter l'intervention de la transformation de Householder symplectique optimale osh_2, dans l'algorithme $JHMSH$, à l'annulation de l'entrée de la position $(i+1, i)$ de la matrice seulement au lieu d'annuler toutes les entrées de la colonne i de la position $i+1$ à n et de $n+i+1$ à $2n$. Nous annulons les autres entrées de la colonne via les transformations de Givens et les transformations de Householder au sens de Van

Loan. Cette modification dans l'algorithme $JHMSH$ va correspondre au remplacement de la transformation de Gauss par la transformation de Householder symplectique optimale dans l'algorithme $JHESS$. En effet, la nouvelle version modifiée basée seulement sur trois transformations : les transformations de Givens symplectiques (algorithme J), les transformations de Householder symplectiques au sens de Van Loan (algorithme H) et les transformations Householder symplectiques optimales (algorithme osh_2).

De même, avant de donner l'algorithme correspondant, nous allons présenter une description des étapes de cette réduction sous la forme J-Hessenberg d'une matrice A arbitraire de taille $2n \times 2n$.

Soit la matrice $A = [a_1, \ldots, a_n, a_{n+1}, \ldots, a_{2n}] \in \mathbb{R}^{2n\times 2n}$ et posons que $A^{(0)} = A$.

La première étape :

Premièrement, nous voulons annuler les entrées de la position 2 jusqu'à position n et de la position $n+2$ jusqu'à la position $2n$ de la première colonne de la matrice A. Au lieu d'utiliser une transformation de Householder symplectique optimale de type deux (T_2), nous utilisons les transformations de Van Loan. Au commencement, nous calculons les rotations de Givens au sens de Van Loan $J(k,c,s)$ pour $k = n, \ldots, 2$ afin de créer un zéro dans l'entrée de position $n+k$ dans la première colonne de la matrice $J(k,c,s)A^{(0)}$. Puis, nous multiplions cette dernière à droite par la matrice $J(k,c,s)^{\mathrm{T}}$. La multiplication à droite ne modifiée pas la première colonne de la matrice $J(k,c,s)A^{(0)}$. Donc, nous aurons la matrice

$$A'^{(1)} = J(2,c,s)\ldots J(n,c,s)A^{(0)}J(n,c,s)^{\mathrm{T}}\ldots J(2,c,s)^{\mathrm{T}}.$$

Ainsi, les entrées aux positions $n+2, \ldots, 2n$ de la première colonne de la matrice $A'^{(1)}$ sont toutes nulles. D'où, nous obtenons

$$\begin{aligned} A'^{(1)} &= J(2,c,s)\ldots J(n,c,s)A^{(0)}J(n,c,s)^{\mathrm{T}}\ldots J(2,c,s)^{\mathrm{T}} \\ &= \begin{bmatrix} A^{(0)}_{(1,1)} & A'^{(1)}_{(1,2:n)} & A^{(0)}_{(1,n+1)} & A'^{(1)}_{(1,n+2:2n)} \\ A'^{(1)}_{(2:n,1)} & A'^{(1)}_{(2:n,2:n)} & A'^{(1)}_{(2:n,n+1)} & A'^{(1)}_{(2:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A'^{(1)}_{(n+1,2:n)} & A^{(0)}_{(n+1,n+1)} & A'^{(1)}_{(n+1,n+2:2n)} \\ 0 & A'^{(1)}_{(n+2:2n,2:n)} & A'^{(1)}_{(n+2:2n,n+1)} & A'^{(1)}_{(n+2:2n,n+2:2n)} \end{bmatrix}. \end{aligned}$$

Ensuite, nous cherchons à mettre des zéros dans les entrées aux positions $2, \ldots, n$ de la première colonne. Par conséquent, nous cherchons le vecteur w de la transformation de Householder au sens de Van Loan tel que $H(2,w) = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w$, de sorte que la multiplication de la matrice $A'^{(1)}$ à gauche par la matrice $H(2,w)$ crée des zéros dans les entrées des positions $3, \ldots, n$ de la première colonne. La multiplication à gauche et à droite par la matrice $H(2,w)$ et $H(2,w)^{\mathrm{T}}$ respectivement ne modifiée pas les zéros déjà créés. Ainsi, nous actualisons la matrice $A'^{(1)}$ pour avoir la matrice $A''^{(1)} = H(2,w)A'^{(1)}H(2,w)^{\mathrm{T}}$,

tel que

$$A''^{(1)} = H(2,w)A'^{(1)}H(2,w)^{\mathrm{T}} = \begin{bmatrix} A^{(0)}_{(1,1)} & A''^{(1)}_{(1,2:n)} & A^{(0)}_{(1,n+1)} & A''^{(1)}_{(1,n+2:2n)} \\ A''^{(1)}_{(2,1)} & A''^{(1)}_{(2,2:n)} & A''^{(1)}_{(2,n+1)} & A''^{(1)}_{(2,n+2:2n)} \\ 0 & A''^{(1)}_{(3:n,2:n)} & A''^{(1)}_{(3:n,n+1)} & A''^{(1)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A''^{(1)}_{(n+1,2:n)} & A^{(0)}_{(n+1,n+1)} & A''^{(1)}_{(n+1,n+2:2n)} \\ 0 & A''^{(1)}_{(n+2:2n,2:n)} & A''^{(1)}_{(n+2:2n,n+1)} & A''^{(1)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

Pour terminer la mise à zéro de la première colonne de la matrice A il faut annuler l'entrée de la position $(2,1)$ en utilisant une transformation de Householder symplectique optimale de type deux (T_2) avec l'algorithme osh_2. En effet, nous calculons la constante $c_1 \in \mathbb{R}$ et le vecteur $v_1 \in \mathbb{R}^{2n}$ tel que $H_1 = I - c_1 v_1 v_1^J$. La transformation H_1 correspond à la transformation T_2. Comme $H_1 e_1 = e_1$, alors $v_1^J e_1 = v_1^{\mathrm{T}} J e_1 = 0$. Ainsi, l'entrée de la position $(n+1)^{\text{ième}}$ de vecteur v_1 est nulle. Donc, H_1 préserve la $(n+1)^{\text{ième}}$ ligne. La multiplication de la matrice $A''^{(1)}$ à gauche par H_1 préserve la $(n+1)^{\text{ième}}$ ligne et crée le zéro désirés dans la première colonne. Puis, la multiplication de $H_1 A''^{(1)}$ à droite par H_1^J préserve la première colonne de $H_1 A''^{(1)} H_1^J$ et nous aurons :

$$A^{(1)} = H_1 A''^{(1)} H_1^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1:2n)} \\ 0 & A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1:2n)} \\ 0 & A^{(1)}_{(n+2:2n,2:n)} & A^{(1)}_{(n+2:2n,n+1:2n)} \end{bmatrix}.$$

□

La deuxième étape :

Cette étape correspond exactement à la deuxième étape de l'algorithme précédent $JHMSH$. En effet, nous utilisons les transformations de Van Loan pour créer les zéros souhaitées dans la $(n+1)^{\text{ième}}$ colonne et en conservant la première colonne inchangée. Après l'application des transformations de Givens, nous aurons

$$\begin{aligned} A'^{(2)} &= J(2,c,s)\dots J(n,c,s)A^{(1)}J(n,c,s)^{\mathrm{T}}\dots J(2,c,s)^{\mathrm{T}} \\ &= \begin{bmatrix} A^{(1)}_{(1,1)} & A'^{(2)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A'^{(2)}_{(1,n+2:2n)} \\ 0 & A'^{(2)}_{(2,2:n)} & A'^{(2)}_{(2,n+1)} & A'^{(2)}_{(2,n+2:2n)} \\ 0 & A'^{(2)}_{(3:n,2:n)} & A'^{(2)}_{(3:n,n+1)} & A'^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A'^{(2)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A'^{(2)}_{(n+1,n+2:2n)} \\ 0 & A'^{(2)}_{(n+2,2:n)} & 0 & A'^{(2)}_{(n+2,n+2:2n)} \\ 0 & A'^{(2)}_{(n+3:2n,2:n)} & 0 & A'^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}. \end{aligned}$$

Ensuite, nous appliquons la transformation de Householder au sens de Van Loan.

Ainsi,

$$A^{(2)} = H(2,w)A'^{(2)}H(2,w)^{\mathrm{T}} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2:2n)} \\ 0 & A^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A^{(2)}_{(2,n+2:2n)} \\ 0 & A^{(2)}_{(3:n,2:n)} & 0 & A^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(2)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2,2:n)} & 0 & A^{(2)}_{(n+2,n+2:2n)} \\ 0 & A^{(2)}_{(n+3:2n,2:n)} & 0 & A^{(2)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

□

La $j'^{\text{ième}}$ étape :

Dans cette étape, avec $j = 2j'-1 \in \mathbb{N}$, nous créons les zéros souhaitées dans la colonne j'. Comme dans la première étape, nous utilisons les rotations de Givens au sens de Van Loan pour annuler le bas de la $j'^{\text{ième}}$ colonne. C'est-à-dire créer des zéros dans les entrées aux positions $n+j'+1,\ldots,2n$ de la colonne j'. Ainsi, nous obtenions la $j'^{\text{ième}}$ colonne de la matrice

$$A'^{(2j'-1)} = J(j'+1,c,s)\ldots J(n,c,s)A^{(2j'-1)}J(n,c,s)^{\mathrm{T}}\ldots J(j'+1,c,s)^{\mathrm{T}}$$

sous la forme suivante :

$$A'^{(2j'-1)}(:,j') = \begin{array}{l} \overbrace{\begin{bmatrix} A^{(2j'-2)}(1:j',j') \\ A'^{(2j'-1)}(j'+1:n,j') \\ \hline A^{(2j'-2)}(n+1:n+j',j') \\ 0 \end{bmatrix}}^{\text{la } (j')^{\text{ième}} \text{ colonne de } A'^{(2j'-1)}} \\ \end{array} \begin{array}{ll} \} & (j') \\ \} & (n-j') \\ \hline \} & (j') \\ \} & (n-j') \end{array}.$$

Puis, nous cherchons à créer des zéros dans les entrées aux positions $j'+2,\ldots,n$ de la $(j')^{\text{ième}}$ colonne de la matrice $A'^{(2j'-1)}$ via la transformation de Householder au sens de Van Loan $H(2j'-1,w) = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w$. La multiplication à gauche ne modifiée pas les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'-1$ ainsi que les zéros déjà créés dans la $(j')^{\text{ième}}$ colonne de la matrice $A'^{(2j'-1)}$. Aussi la multiplication à droite par la matrice $H(2j'-1,w)^{\mathrm{T}}$ ne modifiée pas les colonnes $1,\ldots,j'$ et les colonnes $n+1,\ldots,n+j'$ de la matrice actualisée

$$A''^{(2j'-1)} = H(2j'-1,w)A'^{(2j'-1)}H(2j'-1,w)^{\mathrm{T}}.$$

Donc, la $j'^{\text{ième}}$ colonne de la matrice $A''^{(2j'-1)}$ est transformée sous la forme suivante :

$$A''^{(2j'-1)}(:,j') = \begin{array}{l} \overbrace{\begin{bmatrix} A^{(2j'-2)}(1:j',j') \\ A''^{(2j'-1)}(j'+1,j') \\ 0 \\ \hline A^{(2j'-2)}(n+1:n+j',j') \\ 0 \end{bmatrix}}^{\text{la } (j')^{\text{ième}} \text{ colonne de } A''^{(2j'-1)}} \\ \end{array} \begin{array}{ll} \} & (j') \\ \} & (1) \\ \} & (n-j'-1) \\ \hline \} & (j') \\ \} & (n-j') \end{array}.$$

Pour finir cette étape, nous annulons $A''^{(2j'-1)}(j'+1, j')$ par la transformations de Householder symplectique optimale de type deux (T_2). Ainsi, nous cherchons un scalaire $c_j \in \mathbb{R}$ et un vecteur $v_j \in \mathbb{R}^{2n}$ tels que $H_j = I_{2n} + c_j v_j v_j^J$. Nous appliquons cette transformation à la matrice $A''^{(2j'-1)}$ pour avoir $H_j A''^{(2j'-1)}$. La matrice H_j annule l'entrée de la position $(j'+1)$ de la $j'^{\text{ième}}$ colonne de la matrice $A''^{(2j'-1)}$. En effet, cette transformation ne modifiée pas les lignes $1,\ldots,j'-1$, les lignes $n+1,\ldots,n+j'-1$, les colonnes $1,\ldots,j'-1$ et les colonnes $n+1,\ldots,n+j'-1$ ainsi que les zéros déjà créés dans la $(j')^{\text{ième}}$ colonne de la matrice $A''^{(2j'-1)}$. Par suite, nous aurons

$$A^{(2j'-1)} = H_j A''^{(2j'-1)} H_j^J$$

et la $j'^{\text{ième}}$ colonne devient sous la forme suivante :

$$A^{(2j'-1)}(:, j') = \overbrace{\left[\begin{array}{c} A^{(2j'-2)}(1:j', j') \\ 0 \\ \hline A^{(2j'-2)}(n+1:n+j', j') \\ 0 \end{array}\right]}^{\text{la } (j')^{\text{ième}} \text{ colonne de } A^{(2j'-1)}} \begin{array}{ll} \} & (j') \\ \} & (n-j') \\ \hline \} & (j') \\ \} & (n-j') \end{array}.$$

□

La $(j+1)^{\text{ième}}$ étape :

De même, cette $(j+1)^{\text{ième}}$ étape est identique à la $(j+1)^{\text{ième}}$ étape de l'algorithme $JHMSH$. Nous créerons des zéros dans la $(n+j')^{\text{ième}}$ colonne aux positions $j'+1,\ldots,n$ et $j',\ldots,2n$ (avec $j = 2j'-1 \in \mathbb{N}$) en utilisant des rotations de Givens et transformation de Householder au sens de Van Loan. De cette façon, la $(n+j')^{\text{ième}}$ colonne de $A^{(2j')}(:, n+j')$ est transformée sous la forme suivante :

$$A^{(2j')}(:, n+j') = \overbrace{\left[\begin{array}{c} A^{(2j'-1)}(1:j', n+j') \\ A^{(2j')}(j'+1, n+j') \\ 0 \\ \hline A^{(2j'-1)}(n+1:n+j', n+j') \\ 0 \end{array}\right]}^{\text{la } (n+j')^{\text{ième}} \text{ colonne de } A^{(2j')}} \begin{array}{ll} \} & (j') \\ \} & (1) \\ \} & (n-j') \\ \hline \} & (j') \\ \} & (n-j') \end{array}.$$

□

Nous répétons les deux étapes précédentes (la $j^{\text{ième}}$ et la $(j+1)^{\text{ième}}$ étapes) jusqu'à ce que $j+1 = 2n-2$ afin d'obtenir la forme réduite souhaitée.

Du coup, l'algorithme $JHMSH2$ de la réduction d'une matrice arbitraire de taille $2n \times 2n$ sous la forme de J-Hessenberg est le suivant :

Algorithme 3.10 Algorithme $JHMSH2$

Cet algorithme réduit une matrice A sous la forme J-Hessenberg $H = S^J AS$ avec S est une matrice symplectique. La matrices A est remplacée par la matrice J-Hessenberg H.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2n}$ et $S = I_{2n}$.

La suite de l'algorithme $JHMSH2$

SORTIE(S) : H la Matrice réduite sous la forme J-Hessenberg et S la matrice de transformation symplectique.

Fonction $[S,H] = JHMSH2(A)$
$[den, dep] = size(A)$;
$n = den/2$;
$p = dep/2$;
$S = eye(den)$;
Pour $j = 1,\ldots,n-1$ **faire**
  $J_{2(n-j+1)} = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
  $ro = [j:n, n+j:2\times n]$;
  $co = [j:p, p+j:2\times p]$;
  Pour $k = n, n-1,\ldots,j+1$ **faire**
  % Nous mettons un zéro à l'entrée de la position $(n+k, j)$ de la matrice A en appliquant l'algorithme vlg pour déterminer les constantes c et s de la matrice $J(k,c,s)$.
    $[c,s] = vlg(k, A(:,j))$;
  % Mise à jour de la matrice A.
    $vlA = A(k,:)$;
    $A(k,:) = c\times A(k,:) + s\times A(n+k,:)$;
    $A(n+k,:) = -s\times vlA + c\times A(n+k,:)$;
    $vcA = A(:,k)$;
    $A(:,k) = c\times A(:,k) + s\times A(:,n+k)$;
    $A(:,n+k) = -s\times vcA + c\times A(:,n+k)$;
  % Mise à jour de la matrice S implicitement (si nous avons besoin).
    $vcS = S(:,k)$;
    $S(:,k) = c\times S(:,k) + s\times S(:,n+k)$;
    $S(:,n+k) = -s\times vcS + c\times S(:,n+k)$;
  Fin Pour
  Si $j \le n-2$ **alors**
  % Nous mettons des zéros dans la $j^{\text{ième}}$ colonne de la matrice A aux entrées des positions $(j+2,j),(j+3,j),\ldots,(n,j)$ en appliquant l'algorithme vlh pour déterminer la constante β et le vecteur w de la matrice de transformation $H(j+1,w)$.
    $[\beta, w] = vlh(j+1, A(:,j))$;
  % Mise à jour de la matrice A.
    $A(j+1:n, co) = A(j+1:n, co) - \beta\times(w\times w^{\mathrm{T}})\times A(j+1:n, co)$;
    $A(j+1+n:2n, co) = A(j+1+n:2n, co) - \beta\times(w\times w^{\mathrm{T}})\times A(j+1+n:2n, co)$;
    $A(:, j+1:n) = A(:, j+1:n) - \beta\times A(:, j+1:n)\times(w\times w^{\mathrm{T}})$;
    $A(:, n+j+1:2n) = A(:, n+j+1:2n) - \beta\times A(:, n+j+1:n)\times(w\times w^{\mathrm{T}})$;
  % Mise à jour de la matrice S implicitement (si nous avons besoin).
    $S(:, j+1:n) = S(:, j+1:n) - \beta\times S(:, j+1:n)\times(w\times w^{\mathrm{T}})$;
    $S(:, n+j+1:2n) = S(:, n+j+1:2n) - \beta\times S(:, n+j+1:2n)\times(w\times w^{\mathrm{T}})$;
  Fin Si
  % Nous mettons un zéro à l'entrée de la position $(j+1, j)$ de la matrice A en appliquant la transformation de Householder symplectique optimale $T_{osh_2} = I + cvv^{\mathrm{T}}J$.
  $[c, v] = osh_2(A(ro, j))$

La suite de l'algorithme $JHMSH2$

% Mise à jour de la matrice A.

$A(ro, co) = A(ro, co) + c \times v \times (v^{\mathrm{T}} \times J_{2(n-j+1)} \times A(ro, co))$;

$A(:, co) = A(:, co) - (A(:, co) \times (c \times v)) \times v^{\mathrm{T}} \times J_{2(n-j+1)}$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$S(:, co) = S(:, co) - (S(:, co) \times (c \times v)) \times v^{\mathrm{T}} \times J_{2(n-j+1)}$;

Pour $k = n, n-1, \ldots, j+1$ **faire**

% Nous mettons un zéro à l'entrée de la position $(n+k, n+j)$ de la matrice A en appliquant l'algorithme vlg pour déterminer les constantes c et s de la matrice $J(k, c, s)$.

$[c, s] = vlg(k, A(:, n+j))$;

% Mise à jour de la matrice A.

$vlA = A(k, :)$;

$A(k, :) = c \times A(k, :) + s \times A(n+k, :)$;

$A(n+k, :) = -s \times vlA + c \times A(n+k, :)$;

$vcA = A(:, k)$;

$A(:, k) = c \times A(:, k) + s \times A(:, n+k)$;

$A(:, n+k) = -s \times vcA + c \times A(:, n+k)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$vcS = S(:, k)$;

$S(:, k) = c \times S(:, k) + s \times S(:, n+k)$;

$S(:, n+k) = -s \times vcS + c \times S(:, n+k)$;

Fin Pour

Si $j \leq n-2$ **alors**

% Nous mettons des zéros aux entrées des positions $(j+2, n+j), (j+3, n+j), \ldots, (n, n+j)$ de la matrice A en appliquant l'algorithme vlh pour déterminer la constante β et le vecteur w de la matrice $H(j+1, w)$.

$[\beta, w] = vlh(j+1, A(:, n+j))$;

% Mise à jour de la matrice A.

$A(j+1 : n, co) = A(j+1 : n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1 : n, co)$;

$A(j+1+n : 2n, co) = A(j+1+n : 2n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1+n : 2n, co)$;

$A(:, j+1 : n) = A(:, j+1 : n) - \beta \times A(:, j+1 : n) \times (w \times w^{\mathrm{T}})$;

$A(:, n+j+1 : 2n) = A(:, n+j+1 : 2n) - \beta \times A(:, n+j+1 : n) \times (w \times w^{\mathrm{T}})$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$S(:, j+1 : n) = S(:, j+1 : n) - \beta \times S(:, j+1 : n) \times (w \times w^{\mathrm{T}})$;

$S(:, n+j+1 : 2n) = S(:, n+j+1 : 2n) - \beta \times S(:, n+j+1 : 2n) \times (w \times w^{\mathrm{T}})$;

Fin Si

Fin Pour

$H = A$;

Fin

3.5 La réduction sous la forme J-Hessenberg via l'algorithme JHESS

Bunse-Gerstner et Mehrmann ont présenté l'algorithme $JHESS$ permettant la réduction d'une matrice arbitraire sous forme J-Hessenberg. En effet, la construction de l'algorithme $JHESS$ a été basée sur les transformations de Givens symplectiques (via l'algorithme J), les transformations de Householder symplectiques (via l'algorithme H) et les transformations de Gauss symplectiques (via l'algorithme G).

Nous présentons la réduction sous forme J-Hessenberg d'une matrice A arbitraire de taille $2n\times 2n$ comme cela a été fait dans [40]. Avant de donner l'algorithme correspondant, nous allons illustrer la première étape d'élimination sur un exemple de taille 6×6. Dans cet exemple, nous notons que si la matrice A est Hamiltonienne alors il y a des zéros supplémentaires qui sont obtenus.

Soit la matrice

$$A = \begin{pmatrix} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{pmatrix}.$$

Nous utilisons les transformations de Givens symplectiques pour mettre des zéros dans les entrées des positions $(6,1)$, $(5,1)$. Avec $x = Ae_1$, nous déterminons la matrice $J(3,c,s)$ et nous actualisons la matrice A par $A := J(3,c,s)\ A\ J(3,c,s)^{\mathrm{T}}$.

Cas d'une matrice arbitraire :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Cas d'une matrice Hamiltonienne :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & \mathrm{x} & 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Ensuite, avec $x = Ae_1$ nous appliquons alors la matrice $J(2,c,s)$ et nous actualisons la matrice $A := J(2,c,s)\ A\ J(2,c,s)^{\mathrm{T}}$. Étant donné que les première colonnes des matrices $J(3,c,s)$ et $J(2,c,s)$ sont colinéaires au vecteur e_1, la multiplication par ces matrices à droite n'affecte pas les premières colonnes de la matrice $J(3,c,s)\ A$ et $J(2,c,s)\ A$, respectivement. Maintenant, la matrice A est sous la forme suivante :

Cas d'une matrice arbitraire :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Cas d'une matrice Hamiltonienne :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & 0 & 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Puis, nous utilisons les transformations de Householder symplectiques au sens de Van Loan avec $x = Ae_1$, pour générer la matrice $H(2,w)$ qui remet à zéro l'entrée de la position $(3,1)$. Nous actualisons la matrice $A := H(2,w)\ A\ H(2,w)^{\mathrm{T}}$:

Cas d'une matrice arbitraire :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Cas d'une matrice Hamiltonienne :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & 0 & 0 & \mathrm{x} & \mathrm{x} & 0 \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Afin de préserver les zéros déjà créés dans la première colonne, pour la mise à jour de XAX^{-1} nous devons utiliser les matrices X^{-1} dont la première colonne est un multiple de vecteur e_1.

Pour terminer la mise à zéro de la première colonne de la matrice A il faut annuler l'entrée de la position $(2,1)$ en utilisant les transformations de Gauss symplectiques.

En outre, nous calculons avec $x = Ae_1$ la matrice $G(2,v)$ et nous actualisons la matrice A par $A := G(2,v)\ A\ G(2,v)^{-1}$. L'annihilation de l'entrée de la position $(2,1)$ est possible lorsque l'entrée de la position $(4,1)$ est non nulle.

Ainsi, nous aurons :

Cas d'une matrice arbitraire :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Cas d'une matrice Hamiltonienne :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & 0 & 0 & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Par la suite, nous cherchons à éliminer les entrées dans les positions $(6,4)$ et $(5,4)$ par les transformations de Givens symplectiques. Avec $x = Ae_4$, nous calculons la matrice $J(3,c,s)$ pour avoir $A := J(3,c,s)AJ(3,c,s)^{\mathrm{T}}$:

Cas d'une matrice arbitraire :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & 0 & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Cas d'une matrice Hamiltonienne :

$$A = \left(\begin{array}{ccc|ccc} \mathrm{x} & \mathrm{x} & 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ \hline \mathrm{x} & 0 & 0 & \mathrm{x} & 0 & 0 \\ 0 & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ 0 & \mathrm{x} & \mathrm{x} & 0 & \mathrm{x} & \mathrm{x} \end{array}\right).$$

Après, nous actualisons la matrice A par $A := J(2,c,s)\ A\ J(2,c,s)^{\mathrm{T}}$ et nous aurons :

Cas d'une matrice arbitraire :

$$A = \left(\begin{array}{ccc|ccc} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \\ \hline x & x & x & x & x & x \\ 0 & x & x & 0 & x & x \\ 0 & x & x & 0 & x & x \end{array}\right).$$

Cas d'une matrice Hamiltonienne :

$$A = \left(\begin{array}{ccc|ccc} x & 0 & 0 & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \\ \hline x & 0 & 0 & x & 0 & 0 \\ 0 & x & x & 0 & x & x \\ 0 & x & x & 0 & x & x \end{array}\right).$$

Nous notons que la première et la quatrième colonnes des matrices $J(3,c,s)^{\mathrm{T}}$ et $J(2,c,s)^{\mathrm{T}}$ sont colinéaires aux vecteurs e_1 et e_4 respectivement. D'où la multiplication avec ces matrices à droite n'affecte pas les zéros que nous avons déjà obtenus.

Finalement, nous éliminons l'entrée de la position $(3,4)$ par les transformations de Householder symplectiques au sens de Van Loan. En effet, avec $x = Ae_4$ nous aurons $A := H(2,w)\ A\ H(2,w)$:

Cas d'une matrice arbitraire :

$$A = \left(\begin{array}{ccc|ccc} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & 0 & x & x \\ \hline x & x & x & x & x & x \\ 0 & x & x & 0 & x & x \\ 0 & x & x & 0 & x & x \end{array}\right).$$

Cas d'une matrice Hamiltonienne :

$$A = \left(\begin{array}{ccc|ccc} x & 0 & 0 & x & x & 0 \\ 0 & x & x & x & x & x \\ 0 & x & x & 0 & x & x \\ \hline x & 0 & 0 & x & 0 & 0 \\ 0 & x & x & 0 & x & x \\ 0 & x & x & 0 & x & x \end{array}\right).$$

L'algorithme consiste à poursuivre les mêmes étapes précédentes afin de créer des zéros dans les positions désirées pour les colonnes 3 et 5 de la matrice A.

Enfin, la matrice A sera sous la forme suivante :

Cas d'une matrice arbitraire :

$$A = \left(\begin{array}{ccc|ccc} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & x & 0 & x & x \\ \hline x & x & x & x & x & x \\ 0 & x & x & 0 & x & x \\ 0 & 0 & x & 0 & 0 & x \end{array}\right).$$

Cas d'une matrice Hamiltonienne :

$$A = \left(\begin{array}{ccc|ccc} x & 0 & 0 & x & x & 0 \\ 0 & x & 0 & x & x & x \\ 0 & 0 & x & 0 & x & x \\ \hline x & 0 & 0 & x & 0 & 0 \\ 0 & x & 0 & 0 & x & 0 \\ 0 & 0 & x & 0 & 0 & x \end{array}\right).$$

Remarquons que toutes les matrices de transformation dans cet algorithme ont une première colonne colinéaire au vecteur e_1.

Pour la réduction d'une matrice arbitraire de taille $2n \times 2n$ sous la forme J-Hessenberg par une matrice symplectique, dont la première colonne est colinéaire au vecteur e_1, nous avons l'algorithme suivant :

Algorithme 3.11 Algorithme $JHESS$

L'algorithme effectue la réduction d'une matrice A sous la forme d'une matrice J-Hessenberg $H = S^J AS$, où la matrice S est une matrice symplectique. Cette réduction est possible sous une certaines conditions. Si la réduction n'existe pas, alors l'algorithme s'arrête.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2n}$ et $S = I_{2n}$.

SORTIE(S) : H la Matrice réduite sous la forme J-Hessenberg et S la matrice de transformation symplectique tel que $H = S^J HS$.

Fonction $[H,S] = JHESS(A)$
$[den, dep] = size(A)$;
$n = den/2$;
$S = eye(den)$;
Pour $j = 1,\ldots,n-1$ **faire**
  $co = [j:n, n+j:den]$;
  % Mettre des zéros dans les positions souhaiter de la $j^{\text{ième}}$ colonne :
  % Nous calculons les constantes c et s de la matrice $J(k,c,s)$ de la rotation de Givens symplectique au sens de Van Loan dans l'intention d'annuler les entrées désirer dans la partie inférieure de la $j^{\text{ième}}$ colonne.
  Pour $k = n:-1:j+1$ **faire**
  % Nous mettons un zéro à l'entrée de la position $(n+k, j)$ de la matrice A.
    $[c,s] = vlg(k, A(:,j))$;
  % Mise à jour de la matrice A.
    $vlA = A(k,co)$;
    $A(k,co) = c \times A(k,co) + s \times A(n+k,co)$;
    $A(n+k,co) = -s \times vlA + c \times A(n+k,co)$;
    $vcA = A(:,k)$;
    $A(:,k) = c * A(:,k) + s * A(:,n+k)$;
    $A(:,n+k) = -s * vcA + c * A(:,n+k)$;
  % Mise à jour de la matrice S implicitement (si nous avons besoin).
    $vcS = S(:,k)$;
    $S(:,k) = c \times S(:,k) + s \times S(:,n+k)$;
    $S(:,n+k) = -s \times vcS + c \times S(:,n+k)$;
  Fin Pour
  % Nous calculons la constante β et le vecteur w de la matrice $H(j,w)$ de la transformation de Householder symplectique au sens de Van Loan dans le but d'annuler les entrées désirer dans la partie supérieure de la $j^{\text{ième}}$ colonne.
  Si $j \le n-2$ **alors**
  % Nous mettons dans la $j^{\text{ième}}$ colonne de la matrice A des zéros aux entrées des positions $(j+2,j),(j+3,j),\ldots,(n,j)$ de la matrice A.
    $[\beta, w] = vlh(j+1, A(:,j))$;
  % Mise à jour de la matrice A.
    $A(j+1:n,co) = A(j+1:n,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1:n,co)$;
    $A(n+j+1:den,co) = A(n+j+1:den,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(n+j+1:den,co)$;
    $A(:,j+1:n) = A(:,j+1:n) - \beta \times A(:,j+1:n) \times (w \times w^{\mathrm{T}})$;
    $A(:,n+j+1:den) = A(:,n+j+1:den) - \beta \times A(:,n+j+1:den) \times (w \times w^{\mathrm{T}})$;
  % Mise à jour de la matrice S implicitement (si nous avons besoin).

La suite de l'algorithme $JHESS$

$S(:, j+1:n) = S(:, j+1:n) - \beta \times S(:, j+1:n) \times (w \times w^{\mathrm{T}})$;
$S(:, n+j+1:den) = S(:, n+j+1:den) - \beta \times S(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;
Fin Si
Si $(A(j+1, j) \neq 0)$ et $(A(n+j, j) = 0)$ **alors**
Arrêter l'algorithme, la réduction n'est pas possible
Sinon
% Nous calculons les constantes c et d de la matrice $G(k, v)$ de la transformation de Gauss symplectique afin de mettre des zéros dans l'entrée $(j+1)$ de la $j^{\text{ième}}$ colonne.
$[c, d] = sgt(A(:, j), j+1)$;
% Mise à jour de la matrice A.
$A(j, :) = c \times A(j, :) + d \times A(n+j+1, :)$;
$A(j+1, :) = c \times A(j+1, :) + d \times A(n+j, :)$;
$A(n+j, :) = c^{-1} \times A(n+j, :)$;
$A(n+j+1, :) = c^{-1} \times A(n+j+1, :)$;
$A(:, n+j) = -d \times A(:, j+1) + c \times A(:, n+j)$;
$A(:, n+j+1) = -d \times A(: j) + c \times A(:, n+j+1)$;
$A(:, j) = c^{-1} \times A(:, j)$;
$A(: j+1) = c^{-1} \times A(:, j+1)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$S(:, n+j) = -d \times S(:, j+1) + c \times S(:, n+j)$;
$S(:, n+j+1) = -d \times S(:, j) + c \times S(:, n+j+1)$;
$S(:, j) = c^{-1} \times S(:, j)$;
$S(:, j+1) = c^{-1} \times S(:, j+1)$;
Fin Si
% Mettre des zéros dans les positions souhaiter de la $(n+j)^{\text{ième}}$ colonne :
% Calculer les constantes c et s de la matrice $J(k, c, s)$ de la rotation de Givens symplectique au sens de Van Loan dans l'intention d'annuler les entrées désirer dans la partie inférieure de la $(n+j)^{\text{ième}}$ colonne.
Pour $k = n : -1 : j+1$ **faire**
% Nous mettons un zéro à l'entrée de la position $(n+k, n+j)$ de la matrice A.
$[c, s] = vlg(k, A(:, n+j))$;
% Mise à jour de la matrice A.
$vlA = A(k, co)$;
$A(k, co) = c \times A(k, co) + s \times A(n+k, co)$;
$A(n+k, co) = -s \times vlA + c \times A(n+k, co)$;
$vcA = A(:, k)$;
$A(:, k) = c * A(:, k) + s * A(:, n+k)$;
$A(:, n+k) = -s * vcA + c * A(:, n+k)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$vcS = S(:, k)$;
$S(:, k) = c \times S(:, k) + s \times S(:, n+k)$;
$S(:, n+k) = -s \times vcS + c \times S(:, n+k)$;
Fin Pour

La suite de l'algorithme $JHESS$

Si $j \leq n-2$ **alors**
% Nous calculons la constante β et le vecteur w de la matrice $H(j+1,w)$ de la transformation de Householder symplectique au sens de Van Loan dans le but d'annuler les entrées désirer dans la partie supérieure de la $(n+j)^{\text{ième}}$ colonne.
$[\beta, w] = vlh(j+1, A(:, n+j))$;
% Mise à jour de la matrice A.
$A(j+1:n, co) = A(j+1:n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1:n, co)$;
$A(n+j+1:den, co) = A(n+j+1:den, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(n+j+1:den, co)$;
$A(:, j+1:n) = A(:, j+1:n) - \beta \times A(:, j+1:n) \times (w \times w^{\mathrm{T}})$;
$A(:, n+j+1:den) = A(:, n+j+1:den) - \beta \times A(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$S(:, j+1:n) = S(:, j+1:n) - \beta \times S(:, j+1:n) \times (w \times w^{\mathrm{T}})$;
$S(:, n+j+1:den) = S(:, n+j+1:den) - \beta \times S(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;
Fin Si
Fin Pour
$H = A$;
Fin

Remarque 3.5.1. *Dans le cas où cet algorithme est appliqué à une matrice Hamiltonienne, alors la forme de la matrice réduite qui en résulte sera J-tridiagonale, c'est-à-dire :*

$$H = \begin{pmatrix} \diagdown & \diagdown\!\!\diagdown\!\!\diagdown \\ \diagdown & \diagdown \end{pmatrix}.$$

Dans l'algorithme $JHESS$, la réduction sous la forme J-Hessenberg est réalisée par une série de transformations de similarité comportant : des transformations de Givens symplectiques et des transformations de Householder symplectiques au sens de Van Loan ; ces deux dernières sont connues pour être des transformations stables numériquement et orthogonales, ainsi que des transformations de Gauss symplectiques qui ne sont pas orthogonales.

Dans l'algorithme $JHESS$, Bunse-Gerstner et Mehrmann ont appliqué l'algorithme G ($G(k,v)$) pour éliminer l'entre de la position $k^{\text{ième}}$ d'un vecteur x. Si l'entrée x_{n+k-1} est très petite par rapport à x_k, alors $v = -\dfrac{x_k}{x_{n+k-1}}$ devient très grande. Par conséquent, la matrice G devient mal conditionne avec $\kappa_2(G(k,v)) = (1+v^2)^{\frac{1}{2}} + |v|$. Ainsi, l'algorithme $JHESS$ sera instable numériquement et même il peut s'arrêter et il n'aura pas de réduction sous la forme J-Hessenberg.

Remarque 3.5.2. *La réduction sous la forme J-Hessenberg est une étape essentiel dans l'algorithme SR. Lorsqu'il y a un échec de l'algorithme G à un certain niveau ou lorsque le conditionnement $\kappa_2(G(k,v)) = (1+v^2)^{\frac{1}{2}} + |v|$ est très grand, Bunse-Gerstner et Mehrmann ont proposé de choisir une certaine tolérance pour la valeur de v afin de surveiller la stabilité numérique. Si dans $k^{ième}$ étape $|v|$ est plus grand que cette tolérance donnée, alors ils arrêtent l'algorithme à cette étape.*

3.6 Breakdowns et near-breakdowns

Dans la littérature, à notre connaissance, seulement l'algorithme $JHESS$ est utilisé pour effectuer la réduction d'une matrice sous la forme J-Hessenberg, avec des transformations symplectiques. Lorsque nous appliquons l'algorithme $JHESS$, comme présenté par Bunse-Gerstner et Mehrmann dans [40], à une matrice A, alors dans certains cas l'algorithme peut rencontrer une instabilité numérique fatale ("breakdowns") ou bien il peut être très proche d'une telle instabilité ("near-breakdowns"). Ainsi, dans le premier cas l'algorithme $JHESS$ s'arrête. Autrement dit, les problèmes d'instabilités peuvent survenir non seulement parce que l'algorithme s'arrête, mais aussi dans ces cas, où nous sommes proche d'une telle rupture. Par conséquence, nous ne pouvons pas obtenir une réduction sous la forme de J-Hessenberg. En outre, dans ce cas l'algorithme SR ne peut pas être exécuté. Une telle rupture fatale n'est pas nécessairement incurable.

En fait, l'algorithme présenté en [40], utilise des transformations de Gauss symplectiques [40, p 1106], qui ne sont pas orthogonales. Le premier inconvénient de cet algorithme est que ces transformations peuvent être généralement mal conditionnes. Pour contrôler la stabilité numérique, Bunse-Gerstner et Mehrmann ont choisi une certaine tolérance pour la valeur de v dans l'algorithme G (l'algorithme sgt 2.19). Si dans une étape d'itération $|v|$ est plus grand que cette tolérance donnée, alors ils arrêtent cette itération spéciale.

Le second inconvénient, qui est majeur, réside dans la façon proposée en [40] pour y remédier, dans le contexte de l'algorithme SR uniquement, dans sa version implicite. En effet, il est proposé, lorsque le problème est rencontré, de calculer la matrice $A_{k+1} = S_k^{-1} A_k S_k$, sachant que la matrice A_k est la matrice courante sous la forme J-Hessenberg et la matrice de similarité S_k est symplectique. La transformation de similarité exceptionnelle est choisie sous la forme $S_k = I - ww^{\mathrm{T}} J$ tel que le vecteur w est un vecteur aléatoire avec $\|w\| = 1$.

Remarquons d'abord que ce choix $I - ww^{\mathrm{T}} J$ est une transformation de Householder symplectique au sens du [102]. Un choix aléatoire du vecteur w a pour conséquence de détruire la structure J-Hessenberg de la matrice A_k : en effet la matrice A_k est de J-Hessenberg, la matrice $A_{k+1} = S_k^{-1} A_k S_k$ ne le sera plus avec un choix aléatoire de w, avec $S_k = I - ww^{\mathrm{T}} J$. Cela a pour conséquence de doubler le coût de l'algorithme SR implicite, à chaque fois que nous avons recours à une telle itération, puisque le coût majeur de l'algorithme SR implicite, provient essentiellement de la première étape qui consiste à réduire une matrice sous la forme J-Hessenberg. Dans des cas pessimiste, nous pouvons s'attendre à un coût de $\mathcal{O}(n^4)$ d'opérations arithmétiques.

C'est pourquoi une étude approfondie et détaillée à propos les aspects numériques des nouveaux algorithmes ainsi que l'algorithme $JHESS$, les problèmes d'instabilités\d'être proche d'instabilités numériques (breakdowns\near-breakdowns) et leur prédiction, les différentes stratégies des réparations pour ces problèmes, etc., est nécessaire.

Sur ces entrefaites, il s'avère que l'algorithme $JHSH$ et ses différentes variantes $JHMSH$ et $JHMSH2$ peuvent aussi rencontrer ces types de problèmes comme l'algorithme $JHESS$. De tels problèmes d'instabilités se produisent exactement dans la même condition pour tous ces algorithmes. En d'autres termes, l'origine de l'instabilité est clairement identifiée : elle survient dans le $(k-1)^{\text{ième}}$ vecteur x de la matrice A, lorsque l'entrée

de la position $(n+k-1)$ est nulle ou bien très petite par rapport à l'entrée de la position k, qui nous voulons l'annuler.

Par exemple, pour $k = 2$, la première colonne de la matrice A aura l'une des deux formes suivantes :

Cas proche de l'instabilité numérique :	Cas d'instabilité numérique :
$x = \begin{pmatrix} x_1 \\ x_2 \\ 0 \\ \vdots \\ 0 \\ \hline \varepsilon \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.	$x = \begin{pmatrix} x_1 \\ x_2 \\ 0 \\ \vdots \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.

Cela donne lieu à des questions très importantes concernant par exemple les différentes stratégies pour remédies à ses instabilités numériques, lorsque c'est possible. Ainsi, nous décrivons dans [105] une façon très efficace pour corriger un tel problème, comme certifié par les exemples numériques. Pour plus de détails voir dans l'annexe B. Dans la suite nous allons présenté les algorithmes seulement.

3.6.1 L'algorithme JHM^{2}SH

L'algorithme JHM^2SH, la version modifiée de l'algorithme $JHMSH$, de la réduction d'une matrice arbitraire de taille $2n \times 2n$ sous la forme de J-Hessenberg est le suivant :

Algorithme 3.12 Algorithme JHM^2SH

Cet algorithme réduit une matrice A sous la forme J-Hessenberg $H = S^J AS$ avec S est une matrice symplectique.

ENTRÉE(S) : $A \in \mathbb{R}^{2n\times 2n}$ et $S = I_{2n}$.

SORTIE(S) : H la Matrice réduite sous la forme J-Hessenberg et S la matrice de transformation symplectique.

1: Fonction $[S, H] = JHM^2SH(A)$
2: $[den, dep] = size(A)$;
3: $n = den/2$;
4: $p = dep/2$;
5: $S = eye(den)$;
6: $j = 1$;
7: **Tant que** $(j \leqslant n-1)$ **faire**
8: $J_{2(n-j+1)} = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
9: $ro = [j : n, n+j : 2n]$;
10: $co = [j : p, p+j : 2p]$;

La suite de l'algorithme JHM^2SH

% Le test de near breakdown et breakdown.

Tant que $((A(j+1,j) \neq 0)$ et $(A(n+j,j) == 0))$ ou $(|\frac{A(j+1,j)}{A(n+j,j)}| \geqslant 10^8)$ **faire**

$[\beta, w] = vlh(j, A(:,j))$;

% Mise à jour de la matrice A.

$A(j:n,:) = A(j:n,:) - beta \times (w \times w^{\mathrm{T}}) \times A(j:n,:)$;

$A(j+n:den,:) = A(j+n:den,:) - beta \times (w \times w^{\mathrm{T}}) \times A(j+n:den,:)$;

$A(:,j:n) = A(:,j:n) - beta \times A(:,j:n) \times (w \times w^{\mathrm{T}})$;

$A(:,n+j:den) = A(:,n+j:den) - beta \times A(:,n+j:den) \times (w \times w^{\mathrm{T}})$;

% Mise à jour de S implicitement (si nous avons besoin).

$S(:,j:n) = S(:,j:n) - beta \times S(:,j:n) \times (w \times w^{\mathrm{T}})$;

$S(:,n+j:den) = S(:,n+j:den) - beta \times S(:,n+j:den) \times (w \times w^{\mathrm{T}})$;

% L'annulation de l'entrée de la position $(j, n+j-1)$.

$[\beta, w] = vlh(j, A(:,n+j-1))$;

% Mise à jour de la matrice A.

$A(j:n,co) = A(j:n,co) - beta \times (w \times w^{\mathrm{T}}) \times A(j:n,co)$;

$A(j+n:den,co) = A(j+n:den,co) - beta \times (w \times w^{\mathrm{T}}) \times A(j+n:den,co)$;

$A(:,j:n) = A(:,j:n) - beta \times A(:,j:n) \times (w \times w^{\mathrm{T}})$;

$A(:,n+j:den) = A(:,n+j:den) - beta \times A(:,n+j:den) \times (w \times w^{\mathrm{T}})$;

% Mise à jour de S implicitement (si nous avons besoin).

$S(:,j:n) = S(:,j:n) - beta \times S(:,j:n) \times (w \times w^{\mathrm{T}})$;

$S(:,n+j:den) = S(:,n+j:den) - beta \times S(:,n+j:den) \times (w \times w^{\mathrm{T}})$;

Fin Tant que

% Nous mettons des zéros dans $j^{\text{ième}}$ colonne de la matrice A en appliquant la transformation de Householder symplectique optimale osh_2 $T_{osh_2} = I + cvv^J$.

$[c, v] = osh_2(A(ro, j))$

% Mise à jour de la matrice A.

$A(ro,co) = A(ro,co) + c \times v \times (v^{\mathrm{T}} \times J_{2(n-j+1)} \times A(ro,co))$;

$A(:,co) = A(:,co) - (A(:,co) \times (c \times v)) \times v^{\mathrm{T}} \times J_{2(n-j+1)}$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$S(:,co) = S(:,co) - (S(:,co) \times (c \times v)) \times v^{\mathrm{T}} \times J_{2(n-j+1)}$;

Pour $k = n, n-1, \ldots, j+1$ **faire**

% Nous mettons un zéro à l'entrée de la position $(n+k, n+j)$ de la matrice A en appliquant l'algorithme vlg.

$[c, s] = vlg(k, A(:,n+j))$;

% Mise à jour de la matrice A.

$vlA = A(k,:)$;

$A(k,:) = c \times A(k,:) + s \times A(n+k,:)$;

$A(n+k,:) = -s \times vlA + c \times A(n+k,:)$;

$vcA = A(:,k)$;

$A(:,k) = c \times A(:,k) + s \times A(:,n+k)$;

$A(:,n+k) = -s \times vcA + c \times A(:,n+k)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$vcS = S(:,k)$;

$S(:,k) = c \times S(:,k) + s \times S(:,n+k)$;

La suite de l'algorithme JHM^2SH

$S(:, n+k) = -s \times vcS + c \times S(:, n+k)$;
Fin Pour
Si $j \leq n-2$ **alors**
% Nous mettons des zéros aux entrées des positions $(j+2, n+j), (j+3, n+j), \ldots, (n, n+j)$ de la matrice A en appliquant l'algorithme vlh.
$[\beta, w] = vlh(j+1, A(:, n+j))$;
% Mise à jour de la matrice A.
$A(j+1:n, co) = A(j+1:n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1:n, co)$;
$A(j+1+n:2n, co) = A(j+1+n:2n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1+n:2n, co)$;
$A(:, j+1:n) = A(:, j+1:n) - \beta \times A(:, j+1:n) \times (w \times w^{\mathrm{T}})$;
$A(:, n+j+1:2n) = A(:, n+j+1:2n) - \beta \times A(:, n+j+1:n) \times (w \times w^{\mathrm{T}})$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$S(:, j+1:n) = S(:, j+1:n) - \beta \times S(:, j+1:n) \times (w \times w^{\mathrm{T}})$;
$S(:, n+j+1:2n) = S(:, n+j+1:2n) - \beta \times S(:, n+j+1:2n) \times (w \times w^{\mathrm{T}})$;
Fin Si
$j = j+1$;
Fin Tant que
$H = A$;
Fin

3.6.2 L'algorithme JHM²SH2

L'algorithme JHM^2SH2, la version modifiée de l'algorithme $JHMSH2$, de la réduction d'une matrice arbitraire de taille $2n \times 2n$ sous la forme de J-Hessenberg est le suivant :

Algorithme 3.13 Algorithme JHM^2SH2

Cet algorithme réduit une matrice A sous la forme J-Hessenberg $H = S^J AS$ avec S est une matrice symplectique. La matrices A est remplacée par la matrice J-Hessenberg H.

ENTRÉE(S) : $A \in \mathbb{R}^{2n \times 2n}$ et $S = I_{2n}$.

SORTIE(S) : H la Matrice réduite sous la forme J-Hessenberg et S la matrice de transformation symplectique.

Fonction $[S, H] = JHM^2SH2(A)$
$[den, dep] = size(A)$;
$n = den/2$;
$p = dep/2$;
$S = eye(den)$;
$nbd = 0$;
$j = 1$;
Tant que $j \leqslant n-1$ **faire**
Si $nbd == 1$ **alors**
$nbd = 0$;
$j = j-1$;
Fin Si

La suite de l'algorithme JHM^2SH2 :

$J_{2(n-j+1)} = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
$ro = [j:n, n+j:2\times n]$;
$co = [j:p, p+j:2\times p]$;
Pour $k = n, n-1, \ldots, j+1$ **faire**
% Nous mettons un zéro à l'entrée de la position $(n+k, j)$ de la matrice A en appliquant l'algorithme vlg.
  $[c, s] = vlg(k, A(:, j))$;
% Mise à jour de la matrice A.
  $vlA = A(k,:)$;
  $A(k,:) = c\times A(k,:) + s\times A(n+k,:)$;
  $A(n+k,:) = -s\times vlA + c\times A(n+k,:)$;
  $vcA = A(:,k)$;
  $A(:,k) = c\times A(:,k) + s\times A(:,n+k)$;
  $A(:,n+k) = -s\times vcA + c\times A(:,n+k)$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
  $vcS = S(:,k)$;
  $S(:,k) = c\times S(:,k) + s\times S(:,n+k)$;
  $S(:,n+k) = -s\times vcS + c\times S(:,n+k)$;
Fin Pour
Si $j \leq n-2$ **alors**
% Nous mettons des zéros dans la $j^{\text{ième}}$ colonne de la matrice A aux entrées des positions $(j+2, j), (j+3, j), \ldots, (n, j)$, en appliquant l'algorithme vlh.
  $[\beta, w] = vlh(j+1, A(:, j))$;
% Mise à jour de la matrice A.
  $A(j+1:n, co) = A(j+1:n, co) - \beta\times(w\times w^{\mathrm{T}})\times A(j+1:n, co)$;
  $A(j+1+n:2n, co) = A(j+1+n:2n, co) - \beta\times(w\times w^{\mathrm{T}})\times A(j+1+n:2n, co)$;
  $A(:, j+1:n) = A(:, j+1:n) - \beta\times A(:, j+1:n)\times(w\times w^{\mathrm{T}})$;
  $A(:, n+j+1:2n) = A(:, n+j+1:2n) - \beta\times A(:, n+j+1:n)\times(w\times w^{\mathrm{T}})$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
  $S(:, j+1:n) = S(:, j+1:n) - \beta\times S(:, j+1:n)\times(w\times w^{\mathrm{T}})$;
  $S(:, n+j+1:2n) = S(:, n+j+1:2n) - \beta\times S(:, n+j+1:2n)\times(w\times w^{\mathrm{T}})$;
Fin Si
% Le test de near breakdown et breakdown.
Si $((A(j+1, j) \neq 0)$ et $(A(n+j, j) == 0))$ ou $(|\frac{A(j+1,j)}{A(n+j,j)}| \geqslant 10^8)$ **alors**
  $[\beta, w] = vlh(j, A(:, j))$;
% Mise à jour de la matrice A.
  $A(j:n,:) = A(j:n,:) - beta\times(w\times w^{\mathrm{T}})\times A(j:n,:)$;
  $A(j+n:den,:) = A(j+n:den,:) - beta\times(w\times w^{\mathrm{T}})\times A(j+n:den,:)$;
  $A(:, j:n) = A(:, j:n) - beta\times A(:, j:n)\times(w\times w^{\mathrm{T}})$;
  $A(:, n+j:den) = A(:, n+j:den) - beta\times A(:, n+j:den)\times(w\times w^{\mathrm{T}})$;
% Mise à jour de S implicitement (si nous avons besoin).
  $S(:, j:n) = S(:, j:n) - beta\times S(:, j:n)\times(w\times w^{\mathrm{T}})$;
  $S(:, n+j:den) = S(:, n+j:den) - beta\times S(:, n+j:den)\times(w\times w^{\mathrm{T}})$;

La suite de l'algorithme JHM^2SH2 :

% L'annulation de l'entrée de la position $(j, n+j-1)$.

45: $[\beta, w] = vlh(j, A(:, n+j-1))$;

% Mise à jour de la matrice A.

46: $A(j:n, co) = A(j:n, co) - beta \times (w \times w^{\mathrm{T}}) \times A(j:n, co)$;

47: $A(j+n:den, co) = A(j+n:den, co) - beta \times (w \times w^{\mathrm{T}}) \times A(j+n:den, co)$;

48: $A(:, j:n) = A(:, j:n) - beta \times A(:, j:n) \times (w \times w^{\mathrm{T}})$;

49: $A(:, n+j:den) = A(:, n+j:den) - beta \times A(:, n+j:den) \times (w \times w^{\mathrm{T}})$;

% Mise à jour de S implicitement (si nous avons besoin).

50: $S(:, j:n) = S(:, j:n) - beta \times S(:, j:n) \times (w \times w^{\mathrm{T}})$;

51: $S(:, n+j:den) = S(:, n+j:den) - beta \times S(:, n+j:den) \times (w \times w^{\mathrm{T}})$;

52: $nbd = 1$;

53: **Sinon**

% Nous mettons un zéro à l'entrée de la position $(j+1, j)$ de la matrice A en appliquant la transformation de Householder symplectique optimale $T_{osh_2} = I + cvv^{\mathrm{T}}J$.

54: $[c, v] = osh_2(A(ro, j))$

% Mise à jour de la matrice A.

55: $A(ro, co) = A(ro, co) + c \times v \times (v^{\mathrm{T}} \times J_{2(n-j+1)} \times A(ro, co))$;

56: $A(:, co) = A(:, co) - (A(:, co) \times (c \times v)) \times v^{\mathrm{T}} \times J_{2(n-j+1)}$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

57: $S(:, co) = S(:, co) - (S(:, co) \times (c \times v)) \times v^{\mathrm{T}} \times J_{2(n-j+1)}$;

58: **Fin Si**

59: **Si** $nbd == 0$ **alors**

60: **Pour** $k = n, n-1, \ldots, j+1$ **faire**

% Nous mettons un zéro à l'entrée de la position $(n+k, n+j)$ de la matrice A en appliquant l'algorithme vlg.

61: $[c, s] = vlg(k, A(:, n+j))$;

% Mise à jour de la matrice A.

62: $vlA = A(k, :)$;

63: $A(k, :) = c \times A(k, :) + s \times A(n+k, :)$;

64: $A(n+k, :) = -s \times vlA + c \times A(n+k, :)$;

65: $vcA = A(:, k)$;

66: $A(:, k) = c \times A(:, k) + s \times A(:, n+k)$;

67: $A(:, n+k) = -s \times vcA + c \times A(:, n+k)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

68: $vcS = S(:, k)$;

69: $S(:, k) = c \times S(:, k) + s \times S(:, n+k)$;

70: $S(:, n+k) = -s \times vcS + c \times S(:, n+k)$;

71: **Fin Pour**

72: **Si** $j \leq n-2$ **alors**

% Nous mettons des zéros aux entrées des positions $(j+2, n+j), (j+3, n+j), \ldots, (n, n+j)$ de la matrice A en appliquant l'algorithme vlh.

73: $[\beta, w] = vlh(j+1, A(:, n+j))$;

% Mise à jour de la matrice A.

74: $A(j+1:n, co) = A(j+1:n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1:n, co)$;

La suite de l'algorithme JHM^2SH2 :

$A(j+1+n:2n,co) = A(j+1+n:2n,co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1+n:2n,co)$;
$A(:,j+1:n) = A(:,j+1:n) - \beta \times A(:,j+1:n) \times (w \times w^{\mathrm{T}})$;
$A(:,n+j+1:2n) = A(:,n+j+1:2n) - \beta \times A(:,n+j+1:2n) \times (w \times w^{\mathrm{T}})$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$S(:,j+1:n) = S(:,j+1:n) - \beta \times S(:,j+1:n) \times (w \times w^{\mathrm{T}})$;
$S(:,n+j+1:2n) = S(:,n+j+1:2n) - \beta \times S(:,n+j+1:2n) \times (w \times w^{\mathrm{T}})$;
Fin Si
Fin Si
$j = j+1$;
Fin Tant que
$H = A$;
Fin

3.6.3 L'algorithme MJHESS

L'algorithme $MJHESS$, la version modifiée de l'algorithme $JHESS$, de la réduction d'une matrice arbitraire de taille $2n \times 2n$ sous la forme de J-Hessenberg est le suivant :

Algorithme 3.14 Algorithme $MJHESS$

L'algorithme effectue la réduction d'une matrice A sous la forme d'une matrice J-Hessenberg $H = S^J AS$, où la matrice S est une matrice symplectique.

ENTRÉE(S) : $A \in \mathbb{R}^{2n \times 2n}$ et $S = I_{2n}$.

SORTIE(S) : H la Matrice réduite sous la forme J-Hessenberg et S la matrice de transformation symplectique tel que $H = S^J AS$.

Fonction $[H,S] = MJHESS(A)$
$[den, dep] = size(A)$;
$n = den/2$;
$S = eye(den)$;
$nbd = 0$;
$j = 1$;
Tant que $j \leqslant n-1$ **faire**
Si $nbd == 1$ **alors**
$nbd = 0$;
$j = j-1$;
Fin Si
$co = [j:n, n+j:den]$;
% Mettre des zéros dans les positions souhaiter de la $j^{\text{ième}}$ colonne :
% Nous calculons les constantes c et s de la matrice $J(k,c,s)$ dans l'intention d'annuler les entrées désirer dans la partie inférieure de la $j^{\text{ième}}$ colonne.
Pour $k = n:-1:j+1$ **faire**
% Nous mettons un zéro à l'entrée de la position $(n+k, j)$ de la matrice A.
$[c,s] = vlg(k, A(:,j))$;

La suite de l'algorithme $MJHESS$:

% Mise à jour de la matrice A.

$vlA = A(k, co)$;

$A(k, co) = c \times A(k, co) + s \times A(n+k, co)$;

$A(n+k, co) = -s \times vlA + c \times A(n+k, co)$;

$vcA = A(:, k)$;

$A(:, k) = c * A(:, k) + s * A(:, n+k)$;

$A(:, n+k) = -s * vcA + c * A(:, n+k)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$vcS = S(:, k)$;

$S(:, k) = c \times S(:, k) + s \times S(:, n+k)$;

$S(:, n+k) = -s \times vcS + c \times S(:, n+k)$;

Fin Pour

% Nous calculons la constante β et le vecteur w de la matrice $H(j, w)$ dans le but d'annuler les entrées désirer dans la partie supérieure de la $j^{\text{ième}}$ colonne.

Si $j \leq n-2$ **alors**

% Nous mettons dans la $j^{\text{ième}}$ colonne de la matrice A des zéros aux entrées des positions $(j+2, j), (j+3, j), \ldots, (n, j)$ de la matrice A.

$[\beta, w] = vlh(j+1, A(:, j))$;

% Mise à jour de la matrice A.

$A(j+1:n, co) = A(j+1:n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1:n, co)$;

$A(n+j+1:den, co) = A(n+j+1:den, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(n+j+1:den, co)$;

$A(:, j+1:n) = A(:, j+1:n) - \beta \times A(:, j+1:n) \times (w \times w^{\mathrm{T}})$;

$A(:, n+j+1:den) = A(:, n+j+1:den) - \beta \times A(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$S(:, j+1:n) = S(:, j+1:n) - \beta \times S(:, j+1:n) \times (w \times w^{\mathrm{T}})$;

$S(:, n+j+1:den) = S(:, n+j+1:den) - \beta \times S(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;

Fin Si

% Le test de near breakdown et breakdown.

Si $((A(j+1, j) \neq 0)$ et $(A(n+j, j) == 0))$ ou $(|\frac{A(j+1,j)}{A(n+j,j)}| \geqslant 10^8)$ **alors**

$[\beta, w] = vlh(j, A(:, j))$;

% Mise à jour de la matrice A.

$A(j:n, :) = A(j:n, :) - beta \times (w \times w^{\mathrm{T}}) \times A(j:n, :)$;

$A(j+n:den, :) = A(j+n:den, :) - beta \times (w \times w^{\mathrm{T}}) \times A(j+n:den, :)$;

$A(:, j:n) = A(:, j:n) - beta \times A(:, j:n) \times (w \times w^{\mathrm{T}})$;

$A(:, n+j:den) = A(:, n+j:den) - beta \times A(:, n+j:den) \times (w \times w^{\mathrm{T}})$;

% Mise à jour de S implicitement (si nous avons besoin).

$S(:, j:n) = S(:, j:n) - beta \times S(:, j:n) \times (w \times w^{\mathrm{T}})$;

$S(:, n+j:den) = S(:, n+j:den) - beta \times S(:, n+j:den) \times (w \times w^{\mathrm{T}})$;

% L'annulation de l'entrée de la position $(j, n+j-1)$.

$[\beta, w] = vlh(j, A(:, n+j-1))$;

% Mise à jour de la matrice A.

$A(j:n, co) = A(j:n, co) - beta \times (w \times w^{\mathrm{T}}) \times A(j:n, co)$;

$A(j+n:den, co) = A(j+n:den, co) - beta \times (w \times w^{\mathrm{T}}) \times A(j+n:den, co)$;

$A(:, j:n) = A(:, j:n) - beta \times A(:, j:n) \times (w \times w^{\mathrm{T}})$;

La suite de l'algorithme $MJHESS$:

$A(:, n+j:den) = A(:, n+j:den) - beta \times A(:, n+j:den) \times (w \times w^{\mathrm{T}})$;

% Mise à jour de S implicitement (si nous avons besoin).

$S(:, j:n) = S(:, j:n) - beta \times S(:, j:n) \times (w \times w^{\mathrm{T}})$;

$S(:, n+j:den) = S(:, n+j:den) - beta \times S(:, n+j:den) \times (w \times w^{\mathrm{T}})$;

$nbd = 1$;

Sinon

% Nous calculons les constantes c et d de la matrice $G(k, \nu)$ de la transformation de Gauss symplectique afin de mettre des zéros dans l'entrée $(j+1)$ de la $j^{\text{ième}}$ colonne.

$[c, d] = sgt(A(:, j), j+1)$;

% Mise à jour de la matrice A.

$A(j, :) = c \times A(j, :) + d \times A(n+j+1, :)$;

$A(j+1, :) = c \times A(j+1, :) + d \times A(n+j, :)$;

$A(n+j, :) = c^{-1} \times A(n+j, :)$;

$A(n+j+1, :) = c^{-1} \times A(n+j+1, :)$;

$A(:, n+j) = -d \times A(:, j+1) + c \times A(:, n+j)$;

$A(:, n+j+1) = -d \times A(: j) + c \times A(:, n+j+1)$;

$A(:, j) = c^{-1} \times A(:, j)$;

$A(: j+1) = c^{-1} \times A(:, j+1)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$S(:, n+j) = -d \times S(:, j+1) + c \times S(:, n+j)$;

$S(:, n+j+1) = -d \times S(:, j) + c \times S(:, n+j+1)$;

$S(:, j) = c^{-1} \times S(:, j)$;

$S(:, j+1) = c^{-1} \times S(:, j+1)$;

Fin Si

% Mettre des zéros dans les positions souhaiter de la $(n+j)^{\text{ième}}$ colonne :

% Calculer les constantes c et s de la matrice $J(k, c, s)$ dans l'intention d'annuler les entrées désirer dans la partie inférieure de la $(n+j)^{\text{ième}}$ colonne.

Si $nbd == 0$; **alors**

Pour $k = n : -1 : j+1$ **faire**

% Nous mettons un zéro à l'entrée de la position $(n+k, n+j)$ de la matrice A.

$[c, s] = vlg(k, A(:, n+j))$;

% Mise à jour de la matrice A.

$vlA = A(k, co)$;

$A(k, co) = c \times A(k, co) + s \times A(n+k, co)$;

$A(n+k, co) = -s \times vlA + c \times A(n+k, co)$;

$vcA = A(:, k)$;

$A(:, k) = c * A(:, k) + s * A(:, n+k)$;

$A(:, n+k) = -s * vcA + c * A(:, n+k)$;

% Mise à jour de la matrice S implicitement (si nous avons besoin).

$vcS = S(:, k)$;

$S(:, k) = c \times S(:, k) + s \times S(:, n+k)$;

$S(:, n+k) = -s \times vcS + c \times S(:, n+k)$;

Fin Pour

La suite de l'algorithme $MJHESS$:

Si $j \leq n-2$ **alors**
% Nous calculons la constante β et le vecteur w de la matrice $H(j+1,w)$ dans le but d'annuler les entrées désirer dans la partie supérieure de la $(n+j)^{\text{ième}}$ colonne.
$[\beta, w] = vlh(j+1, A(:, n+j))$;
% Mise à jour de la matrice A.
$A(j+1:n, co) = A(j+1:n, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(j+1:n, co)$;
$A(n+j+1:den, co) = A(n+j+1:den, co) - \beta \times (w \times w^{\mathrm{T}}) \times A(n+j+1:den, co)$;
$A(:, j+1:n) = A(:, j+1:n) - \beta \times A(:, j+1:n) \times (w \times w^{\mathrm{T}})$;
$A(:, n+j+1:den) = A(:, n+j+1:den) - \beta \times A(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;
% Mise à jour de la matrice S implicitement (si nous avons besoin).
$S(:, j+1:n) = S(:, j+1:n) - \beta \times S(:, j+1:n) \times (w \times w^{\mathrm{T}})$;
$S(:, n+j+1:den) = S(:, n+j+1:den) - \beta \times S(:, n+j+1:den) \times (w \times w^{\mathrm{T}})$;
Fin Si
Fin Si
$j = j+1$;
Fin Tant que
$H = A$;
Fin

3.7 Résultats numériques

Nous proposons ici quelques exemples numériques dans des situations différentes, ce qui nous permet de faire la comparaison entre les différents algorithmes. Nous comparons la perte de la J-orthogonalité et l'erreur absolue de la réduction sous la forme J-Hessenberg.

Exemple 3.7.1. *Dans cet exemple nous prenons une matrice A de taille $2n \times 2n$ telle que $A = Pascal(2n)$. Puis, nous appliquons à cette dernière les algorithmes $JHESS$, $JHMSH$, $JHMSH2$ et $JHOSH$.*

Dans l'exemple 3.7.1, nous remarquons que, pour la matrice $A = Pascal(2n)$, les algorithmes $JHESS$, $JHMSH$, $JHMSH2$ et $JHOSH$ fournissent des résultats raisonnablement très semblables. La perte de la J-orthogonalité et l'erreur dans la réduction sous la forme J-Hessenberg, pour les différents algorithmes, sont présentées dans le tableau 3.1 et le tableau 3.2 respectivement.

Exemple 3.7.2. *Dans cet exemple nous prenons une matrice A de taille $2n \times 2n$ telle que $A = randn(2n)$. Ensuite, nous appliquons à cette dernière les algorithmes $JHESS$, $JHMSH$, $JHMSH2$ et $JHOSH$.*

Dans l'exemple 3.7.2, nous pouvons observer que les algorithmes $JHESS$, $JHMSH$ et $JHMSH2$ fournissent des résultats raisonnablement semblables, avec un désavantage significatif pour l'algorithme $JHOSH$. Nous Remarquons aussi que l'injection des transformations de Givens symplectiques et les transformations de Householder symplectiques

$2n$	La perte de J-Orthogonalité $\|I - S^J S\|_2$			
	$JHESS$	$JHMSH$	$JHMSH2$	$JHOSH$
4	$4.5386e-16$	$3.1889e-16$	$4.5386e-16$	$2.7894e-16$
6	$8.1225e-16$	$1.0762e-15$	$1.5073e-15$	$1.0240e-15$
8	$2.0280e-15$	$4.2141e-15$	$8.3650e-15$	$1.7292e-12$
10	$3.7131e-15$	$1.8613e-14$	$1.3963e-14$	$2.1527e-13$
12	$5.9575e-15$	$2.4844e-13$	$1.2110e-13$	$9.0008e-13$
14	$1.2189e-14$	$6.4269e-13$	$8.9480e-13$	$4.1346e-12$

TABLEAU 3.1 – La perte de J-orthogonalité de la réduction J-Hessenberg de la matrice $A = Pascal(2n)$ via les algorithmes $JHESS$, $JHMSH$, $JHMSH2$ et $JHOSH$

$2n$	L'erreur absolue de la réduction $\|H - S^{-1}AS\|_2$			
	$JHESS$	$JHMSH$	$JHMSH2$	$JHOSH$
4	$5.4200e-15$	$1.6193e-14$	$5.4200e-15$	$7.7225e-15$
6	$1.1295e-13$	$4.1480e-13$	$2.3103e-13$	$3.1425e-13$
8	$6.5367e-12$	$4.8847e-11$	$3.0102e-11$	$8.7564e-09$
10	$1.1217e-10$	$1.1529e-09$	$8.1251e-10$	$1.3984e-08$
12	$2.7057e-09$	$7.9695e-08$	$1.4269e-07$	$6.0044e-07$
14	$1.1313e-07$	$7.0409e-06$	$3.7869e-06$	$2.1995e-05$

TABLEAU 3.2 – L'erreur absolue de la réduction J-Hessenberg de la matrice $A = Pascal(2n)$ via les algorithmes $JHESS$, $JHMSH$, $JHMSH2$ et $JHOSH$

au sens de Van Loan avant l'application des transformations de Householder symplectiques optimale $osh2$ dans l'algorithme $JHMSH2$ ne fait que renforcer la précision par rapport à l'algorithme $JHMSH$. La perte de la J-orthogonalité et l'erreur dans la réduction sous la forme J-Hessenberg, pour les différents algorithmes, sont affichées dans le tableau 3.3 et le tableau 3.4 respectivement.

Exemple 3.7.3. *Dans cet exemple nous prenons une matrice A de taille $2n \times 2n$ telle que $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$. Sachant que chaque bloc A_{ij} est de taille $n \times n$ avec*

$$A_{11} = \begin{pmatrix} 1 & & & & \\ 2 & 1 & & & \\ & \ddots & \ddots & & \\ & & 2 & 1 & \\ & & & 2 & 1 \end{pmatrix}, \quad A_{12} = \begin{pmatrix} 1 & 2 & & & \\ 2 & 1 & 2 & & \\ & \ddots & \ddots & \ddots & \\ & & 2 & 1 & 2 \\ & & & 2 & 1 \end{pmatrix}, \quad A_{21} = \begin{pmatrix} 0 & 2 & & & \\ 0 & 1 & \ddots & & \\ & & \ddots & \ddots & \\ & & & 1 & 2 \\ & & & & 1 \end{pmatrix}$$

$2n$	La perte de J-Orthogonalité $\|I - S^J S\|_2$			
	$JHESS$	$JHMSH$	$JHMSH2$	$JHOSH$
4	$2.2377e-16$	$5.0453e-16$	$1.5701e-16$	$1.3878e-16$
6	$1.2362e-15$	$1.0314e-14$	$7.2445e-15$	$7.8665e-15$
8	$1.1262e-15$	$4.4185e-15$	$2.0318e-15$	$8.2489e-15$
10	$5.5159e-15$	$6.6951e-14$	$6.1371e-14$	$2.3061e-13$
12	$8.3091e-15$	$8.3005e-13$	$1.2185e-13$	$7.7104e-13$
14	$5.5932e-14$	$4.5568e-13$	$3.8058e-13$	$8.9718e-11$
16	$1.4082e-14$	$2.0836e-13$	$9.2822e-14$	$5.9120e-12$
18	$2.8530e-14$	$1.5159e-12$	$4.1867e-13$	$1.8129e-10$
20	$1.5660e-13$	$8.1831e-11$	$2.4944e-11$	$1.9407e-09$
22	$1.6207e-14$	$3.0169e-13$	$3.8383e-13$	$8.4138e-11$
24	$6.5797e-14$	$9.3572e-12$	$8.3027e-12$	$8.0934e-09$
26	$1.2295e-13$	$3.9518e-11$	$5.5587e-12$	$1.0048e-05$
28	$4.5993e-14$	$1.6715e-12$	$1.1331e-12$	$1.6731e-09$
30	$6.1491e-13$	$5.9323e-11$	$1.1910e-11$	$2.8166e-05$

TABLEAU 3.3 – La perte de J-orthogonalité de la réduction J-Hessenberg de la matrice $A = randn(2n)$ via les algorithmes $JHESS$, $JHMSH$, $JHMSH2$ et $JHOSH$

$$\text{et } A_{22} = \begin{pmatrix} 1 & & & & \\ 3 & 1 & & & \\ & \ddots & \ddots & & \\ & & 3 & 1 & \\ & & & 3 & 1 \end{pmatrix}.$$

Puis, nous appliquons à cette matrice A les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2.

Dans l'exemple 3.7.3, lorsque l'algorithme $JHESS$ est appliqué à la matrice A alors il aura un breakdowns (une instabilité numérique fatale) et l'algorithme s'arrête depuis la première étape. Les résultats numériques de cet exemple montrent l'efficacité de notre stratégie pour remédier à cette instabilité numérique fatale. Ainsi, les algorithmes $MJHESS$, JHM^2SH et JHM^2SH2 fournissent des résultats semblables, avec un petit avantage pour l'algorithme $MJHESS$. La perte de la J-orthogonalité et l'erreur dans la réduction sous la forme J-Hessenberg, pour les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2, sont données dans le tableau 3.5 et le tableau 3.6 respectivement.

Exemple 3.7.4. *Dans cet exemple nous prenons une matrice A Hamiltonienne de taille $2n \times 2n$ telle que $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$. Sachant que chaque bloc A_{ij} est de taille $n \times n$ avec*

$2n$	L'erreur absolue de la réduction $\left\|H - S^{-1}AS\right\|_2$			
	$JHESS$	$JHMSH$	$JHMSH2$	$JHOSH$
4	$7.6284e-16$	$1.7280e-15$	$2.1593e-15$	$1.4299e-15$
6	$1.1399e-14$	$1.3724e-13$	$1.4030e-13$	$4.4936e-13$
8	$5.4087e-15$	$2.7576e-14$	$2.6325e-14$	$8.9172e-14$
10	$4.1767e-14$	$2.6532e-12$	$4.1763e-13$	$1.2819e-11$
12	$4.9776e-14$	$7.1119e-12$	$9.9413e-13$	$2.0655e-11$
14	$1.7671e-13$	$1.3752e-11$	$3.8978e-12$	$6.4185e-09$
16	$1.2971e-13$	$1.4069e-12$	$1.8731e-12$	$2.1820e-10$
18	$1.7410e-13$	$6.4203e-12$	$8.1253e-12$	$3.1054e-08$
20	$1.6234e-12$	$3.3818e-09$	$1.5008e-10$	$7.7719e-07$
22	$1.2996e-13$	$7.0366e-12$	$9.0303e-12$	$2.4491e-09$
24	$7.4530e-13$	$3.1000e-10$	$1.5243e-10$	$7.4156e-07$
26	$1.2377e-12$	$1.2405e-09$	$5.7954e-11$	$6.3300e-02$
28	$7.0871e-13$	$1.8094e-11$	$9.4246e-12$	$4.9546e-07$
30	$3.9641e-12$	$1.4573e-09$	$1.4503e-10$	$1.1500e-02$

TABLEAU 3.4 – L'erreur absolue de la réduction J-Hessenberg de la matrice $A = randn(2n)$ via les algorithmes $JHESS$, $JHMSH$, $JHMSH2$ et $JHOSH$

$A_{11} = -A_{22}^{\mathrm{T}}$, *les blocs* A_{12} *et* A_{12} *sont deux matrices symétriques :* $A_{12} = A_{12}^{\mathrm{T}}$, $A_{21} = A_{21}^{\mathrm{T}}$

tel que $A_{11} = \begin{pmatrix} 1 & & & & \\ 2 & 1 & & & \\ & \ddots & \ddots & & \\ & & 2 & 1 & \\ & & & 2 & 1 \end{pmatrix}$, $A_{12} = \begin{pmatrix} 1 & 2 & & & \\ 2 & 1 & 2 & & \\ & \ddots & \ddots & \ddots & \\ & & 2 & 1 & 2 \\ & & & 2 & 1 \end{pmatrix}$, $A_{21} = \begin{pmatrix} 0 & 0 & & & \\ 0 & 1 & 3 & & \\ & 3 & \ddots & \ddots & \\ & & \ddots & 1 & 3 \\ & & & 3 & 1 \end{pmatrix}$

et $A_{22} = \begin{pmatrix} -1 & -2 & & & \\ & -1 & \ddots & & \\ & & \ddots & -2 & \\ & & & -1 & -2 \\ & & & & -1 \end{pmatrix}$.

Par la suite, nous appliquons à la matrice Hamiltonienne A les algorithmes JHESS, MJHESS, JHM^2*SH et JHM*^{2}*SH2.*

Dans l'exemple 3.7.4, de même comme pour l'exemple 3.7.3, nous remarquons que lorsque l'algorithme $JHESS$ est appliqué à la matrice Hamiltonienne A, il aura un break-downs (une instabilité numérique fatale) et l'algorithme s'arrête depuis la première étape. C'est-à-dire dans ce cas, il n'y aura pas une réduction sous la forme J-Hessenberg Ha-

n	La perte de J-Orthogonalité $\|I - S^J S\|_2$			
	$JHESS$	$MJHESS$	JHM^2SH	JHM^2SH2
2	échoue	$1.0717e-15$	$2.5168e-16$	$3.1402e-16$
3	échoue	$1.6767e-15$	$1.0412e-15$	$9.7146e-16$
4	échoue	$1.0717e-15$	$3.1015e-15$	$3.6572e-15$
5	échoue	$5.5610e-15$	$2.8250e-14$	$3.3284e-14$
6	échoue	$5.2871e-15$	$4.1918e-14$	$4.3812e-14$
7	échoue	$1.3446e-14$	$2.0709e-13$	$1.1965e-13$
8	échoue	$1.8294e-14$	$1.7497e-12$	$7.4477e-13$
9	échoue	$3.8698e-13$	$1.2988e-10$	$5.8035e-11$
10	échoue	$2.4877e-13$	$4.8062e-10$	$1.1476e-10$
11	échoue	$3.2961e-13$	$6.6942e-10$	$1.7784e-10$
12	échoue	$4.2368e-13$	$4.5165e-10$	$1.7250e-10$
13	échoue	$9.8990e-13$	$7.9908e-10$	$2.9785e-10$
14	échoue	$1.4096e-12$	$7.6406e-10$	$1.7497e-10$
15	échoue	$8.8834e-13$	$1.7248e-09$	$1.9073e-10$
16	échoue	$8.3168e-13$	$6.9530e-10$	$1.9133e-10$
17	échoue	$9.0923e-13$	$1.9515e-09$	$2.1889e-10$
18	échoue	$1.4143e-12$	$1.1824e-09$	$6.2781e-10$
19	échoue	$3.0133e-12$	$3.6906e-09$	$2.2293e-10$
20	échoue	$3.0854e-12$	$2.8172e-09$	$2.6019e-10$
21	échoue	$2.4744e-12$	$1.5606e-08$	$8.6765e-10$
22	échoue	$1.1614e-12$	$1.0522e-09$	$2.4081e-10$
23	échoue	$1.2597e-12$	$3.8242e-09$	$2.6805e-10$
24	échoue	$2.0548e-12$	$1.1119e-09$	$4.8392e-10$
25	échoue	$3.0479e-12$	$3.9755e-09$	$4.2710e-10$
26	échoue	$3.8862e-12$	$1.8132e-09$	$1.4496e-09$
27	échoue	$4.3655e-12$	$1.2417e-08$	$1.1257e-09$
28	échoue	$4.3449e-12$	$2.2564e-09$	$1.1255e-09$
29	échoue	$9.6426e-12$	$3.9904e-08$	$2.3791e-09$
30	échoue	$5.3754e-12$	$1.6554e-09$	$5.4776e-10$

TABLEAU 3.5 – La perte de J-orthogonalité de la réduction J-Hessenberg de la matrice A via les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2

n	L'erreur absolue de la réduction $\left\|H - S^{-1}AS\right\|_2$			
	$JHESS$	$MJHESS$	JHM^2SH	JHM^2SH2
2	échoue	$8.9915e-15$	$1.0361e-15$	$1.0262e-15$
3	échoue	$3.9086e-15$	$1.0623e-14$	$5.6678e-15$
4	échoue	$8.9915e-15$	$6.3153e-14$	$2.9172e-14$
5	échoue	$2.5502e-14$	$1.4279e-13$	$6.8545e-14$
6	échoue	$5.1641e-14$	$2.5845e-13$	$1.6997e-13$
7	échoue	$4.9408e-14$	$2.7021e-12$	$5.7755e-13$
8	échoue	$1.2355e-13$	$1.0972e-11$	$3.5435e-12$
9	échoue	$1.2932e-12$	$1.0461e-09$	$3.8219e-10$
10	échoue	$2.4578e-12$	$3.4164e-09$	$7.1532e-10$
11	échoue	$2.7462e-12$	$4.7274e-09$	$5.7041e-10$
12	échoue	$6.5230e-12$	$1.1306e-08$	$8.0399e-10$
13	échoue	$7.3703e-12$	$7.4063e-09$	$1.7637e-09$
14	échoue	$1.1269e-11$	$8.3607e-09$	$1.0158e-09$
15	échoue	$7.3380e-12$	$1.1932e-08$	$9.8201e-10$
16	échoue	$6.7075e-12$	$5.6770e-09$	$1.1922e-09$
17	échoue	$8.2522e-12$	$1.4054e-08$	$1.2598e-09$
18	échoue	$3.4928e-11$	$1.4967e-07$	$5.7161e-09$
19	échoue	$2.1809e-11$	$2.5400e-08$	$1.4194e-09$
20	échoue	$6.7873e-11$	$1.2725e-07$	$2.0413e-09$
21	échoue	$3.9748e-11$	$2.6936e-07$	$5.1208e-09$
22	échoue	$6.5786e-11$	$1.1047e-08$	$1.9222e-09$
23	échoue	$6.3465e-11$	$2.1954e-08$	$1.6025e-09$
24	échoue	$6.2066e-11$	$5.6800e-08$	$3.2751e-09$
25	échoue	$2.3255e-11$	$2.2816e-08$	$2.6839e-09$
26	échoue	$2.9622e-11$	$3.2416e-08$	$1.0678e-08$
27	échoue	$1.0102e-10$	$1.0768e-07$	$1.0010e-08$
28	échoue	$1.4271e-10$	$1.4462e-07$	$8.2262e-09$
29	échoue	$5.3487e-11$	$6.3257e-07$	$4.1958e-08$
30	échoue	$4.6021e-11$	$5.9380e-08$	$4.0406e-09$

TABLEAU 3.6 – L'erreur absolue de la réduction J-Hessenberg de la matrice A via les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2

n	La perte de J-Orthogonalité $\|I - S^J S\|_2$			
	$JHESS$	$MJHESS$	JHM^2SH	JHM^2SH2
2	échoue	$2.6809e-16$	$1.3843e-16$	$2.7756e-17$
3	échoue	$9.1518e-16$	$2.1967e-15$	$4.1153e-15$
4	échoue	$3.6585e-15$	$3.1724e-14$	$1.1623e-14$
5	échoue	$1.3451e-14$	$5.5639e-13$	$4.5393e-13$
6	échoue	$3.2002e-15$	$1.3229e-14$	$3.1824e-14$
7	échoue	$1.7497e-14$	$1.9456e-13$	$2.9018e-13$
8	échoue	$1.1440e-14$	$2.4182e-13$	$9.1255e-14$
9	échoue	$4.7591e-14$	$7.0030e-12$	$4.6008e-12$
10	échoue	$9.8212e-14$	$6.7908e-11$	$1.8421e-11$
11	échoue	$2.5071e-13$	$1.2746e-10$	$3.6111e-11$
12	échoue	$3.0863e-13$	$1.6379e-09$	$1.1448e-10$
13	échoue	$2.3432e-12$	$5.7401e-09$	$1.8386e-09$
14	échoue	$1.5649e-12$	$5.9220e-09$	$2.7826e-09$
15	échoue	$1.2911e-11$	$1.1198e-07$	$1.5282e-08$
16	échoue	$1.7852e-11$	$3.2853e-07$	$1.9260e-07$
17	échoue	$5.2827e-11$	$1.0707e-06$	$1.9526e-07$
18	échoue	$2.1702e-10$	$2.2014e-04$	$2.2887e-05$
19	échoue	$6.5499e-10$	$7.0710e-05$	$2.0118e-05$
20	échoue	$5.6016e-09$	$7.9995e-04$	$4.0086e-05$

TABLEAU 3.7 – La perte de J-orthogonalité de la réduction J-Hessenberg de la matrice A via les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2

miltonienne et par conséquent, l'algorithme SR ne peut pas être exécuté. Cependant, une telle rupture fatale n'est pas insurmontable. Les résultats numériques de cet exemple montrent l'efficacité de notre stratégie pour remédier à cette instabilité numérique fatale dans le cas d'une matrice Hamiltonienne. Ainsi, les algorithmes $MJHESS$, $JHMSH$ et $JHMSH2$ fournissent des résultats semblables, avec un petit avantage pour l'algorithme $MJHESS$. La perte de la J-orthogonalité et l'erreur dans la réduction sous la forme J-Hessenberg, pour les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2, sont exhibées dans le tableau 3.7 et le tableau 3.8 respectivement.

n	L'erreur absolue de la réduction $\|H - S^{-1}AS\|_2$			
	$JHESS$	$MJHESS$	JHM^2SH	JHM^2SH2
2	échoue	$1.2230e-15$	$3.4732e-16$	$7.5047e-16$
3	échoue	$4.6309e-15$	$1.4123e-14$	$9.5826e-15$
4	échoue	$1.1179e-14$	$1.0235e-13$	$1.1283e-13$
5	échoue	$1.0634e-13$	$2.2678e-12$	$1.4082e-12$
6	échoue	$2.0835e-14$	$1.6308e-13$	$1.8500e-13$
7	échoue	$3.6429e-13$	$4.2300e-12$	$5.7276e-12$
8	échoue	$5.3612e-14$	$2.6360e-12$	$1.2184e-12$
9	échoue	$4.2431e-13$	$2.8308e-11$	$6.0019e-11$
10	échoue	$5.5556e-13$	$1.8128e-10$	$4.2484e-11$
11	échoue	$6.2363e-12$	$1.2132e-09$	$1.3393e-10$
12	échoue	$3.2918e-12$	$5.6804e-09$	$1.0683e-09$
13	échoue	$1.7487e-11$	$4.3477e-07$	$5.7596e-09$
14	échoue	$1.2069e-11$	$1.1117e-07$	$1.1405e-08$
15	échoue	$1.2035e-10$	$8.4815e-07$	$2.1596e-07$
16	échoue	$2.0077e-10$	$3.6979e-06$	$8.2332e-07$
17	échoue	$1.8192e-09$	$1.4713e-05$	$3.9805e-06$
18	échoue	$8.0165e-09$	$1.3000e-03$	$4.6621e-04$
19	échoue	$2.2317e-07$	$1.5000e-03$	$4.0607e-04$
20	échoue	$2.5767e-06$	$4.1000e-03$	$6.8321e-04$

TABLEAU 3.8 – L'erreur absolue de la réduction J-Hessenberg de la matrice A via les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2

Exemple 3.7.5. *Nous considérons un exemple de breakdown*

$$A = \begin{pmatrix} 1 & 5 & 7 & 9 & 5 & 1 & 1 & 3 & 1 & 3 & 7 & 2 \\ 0 & 1 & 4 & 6 & 1 & 2 & 2 & 1 & 5 & 4 & 3 & 5 \\ 0 & 0 & 1 & 2 & 3 & 2 & 0 & 0 & 1 & 2 & 5 & 3 \\ 0 & 0 & 2 & 1 & 9 & 8 & 0 & 0 & 2 & 1 & 2 & 4 \\ 0 & 0 & 0 & 2 & 1 & 3 & 0 & 0 & 5 & 2 & 1 & 2 \\ 0 & 0 & 0 & 4 & 2 & 1 & 0 & 0 & 4 & 3 & 2 & 1 \\ 1 & 4 & 7 & 2 & 1 & 3 & 1 & 7 & 6 & 1 & 6 & 7 \\ 0 & 1 & 9 & 3 & 5 & 1 & 0 & 1 & 4 & 5 & 8 & 3 \\ 0 & 0 & 0 & 2 & 7 & 9 & 0 & 0 & 1 & 3 & 4 & 5 \\ 0 & 0 & 0 & 1 & 2 & 8 & 0 & 0 & 3 & 1 & 7 & 3 \\ 0 & 0 & 0 & 2 & 1 & 2 & 0 & 0 & 4 & 3 & 1 & 2 \\ 0 & 0 & 0 & 9 & 3 & 1 & 0 & 0 & 1 & 2 & 3 & 1 \end{pmatrix}.$$

$2n$	La perte de J-Orthogonalité $\left\|I - S^J S\right\|_2$			
	$JHESS$	$MJHESS$	JHM^2SH	JHM^2SH2
12	échoue	$1.8553e-15$	$5.0842e-15$	$6.6428e-15$

TABLEAU 3.9 – La perte de J-orthogonalité de la réduction J-Hessenberg de l'exemple 3.7.5 via les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2

$2n$	L'erreur absolue de la réduction $\left\|H - S^{-1}AS\right\|_2$			
	$JHESS$	$MJHESS$	JHM^2SH	JHM^2SH2
12	échoue	$3.2709e-14$	$3.8777e-13$	$2.7653e-13$

TABLEAU 3.10 – L'erreur absolue de la réduction J-Hessenberg de l'exemple 3.7.5 via les algorithmes $JHESS$, $MJHESS$, JHM^2SH et JHM^2SH2

Via l'algorithme $MJHESS$, nous avons $H_{MJHESS} = S^J_{MJHESS} A S_{MJHESS}$ tel que :

$$H_{MJHESS} = \begin{pmatrix}
1 & 5 & -27.1818 & -12.9177 & 9.8049 & 11.1982 & 1 & 3 & -12.4164 & -20.4948 & 7.8702 & -6.7329 \\
0 & 1 & -17.3963 & -16.6783 & 3.9217 & 2.2349 & -2 & 1 & -17.5209 & -13.9976 & 5.7151 & -7.2751 \\
0 & 0 & 2.6000 & 11.9671 & -9.1156 & -4.1538 & 0 & 0 & 16.4255 & -0.7512 & -2.5244 & 4.2492 \\
0 & 0 & 0 & -56.4737 & 25.5600 & 20.0819 & 0 & 0 & -61.2806 & 20.3181 & 10.3850 & -19.9690 \\
0 & 0 & 0 & 0 & -2.1626 & -1.6988 & 0 & 0 & 0 & -1.1900 & -2.4397 & -2.1935 \\
0 & 0 & 0 & 0 & 0 & -1.0008 & 0 & 0 & 0 & 0 & 1.7426 & 2.1659 \\
1 & 4 & -11.96 & -24.3261 & 8.9536 & 4.9653 & 1 & 7 & -25.1087 & -7.3735 & 0.3283 & -3.1143 \\
0 & 1 & -16.3091 & -16.1201 & 13.2282 & 12.3682 & 0 & 1 & -18.1647 & -9.9332 & -0.0956 & -0.5728 \\
0 & 0 & 9.4573 & 63.5397 & -28.9661 & -24.3836 & 0 & 0 & 68.4000 & -14.1468 & -16.8095 & 26.5193 \\
0 & 0 & 0 & 3.6585 & 1.2962 & -2.7405 & 0 & 0 & 0 & -2.3176 & 0.1515 & 2.0092 \\
0 & 0 & 0 & 0 & -0.8140 & -0.2898 & 0 & 0 & 0 & 0 & -1.0722 & -1.1692 \\
0 & 0 & 0 & 0 & 0 & 1.7267 & 0 & 0 & 0 & 0 & 0 & 0.0268
\end{pmatrix}.$$

$$S_{MJHESS} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1.0873 & -0.3930 & 0.1956 & 0.1412 & 0 & 0 & -0.2759 & -0.9642 & -0.7511 & 0.7638 \\
0 & 0 & -2.1745 & -0.0657 & -0.2349 & -0.1052 & 0 & 0 & 0.1380 & -2.0543 & 1.1686 & -0.8966 \\
0 & 0 & 0 & -0.3452 & 0.9642 & 0.8810 & 0 & 0 & -0.4599 & 0.3404 & -0.0599 & -0.5165 \\
0 & 0 & 0 & -0.9155 & 0.0072 & -0.0852 & 0 & 0 & -0.9197 & 0.4476 & 0.1553 & -0.2989 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.1126 & 0.4035 & -0.3223 & 0 & 0 & -0.4599 & -0.1176 & 0.0703 & -0.0050 \\
0 & 0 & 0 & 0.0563 & -0.2017 & 0.1612 & 0 & 0 & -0.2299 & 0.0588 & -0.0351 & 0.0025 \\
0 & 0 & 0 & -0.3640 & 0.8585 & 0.9745 & 0 & 0 & -0.4599 & 0.3472 & 0.3951 & 0.0684 \\
0 & 0 & 0 & -2.2213 & -0.0432 & -0.0732 & 0 & 0 & -2.0694 & 0.0521 & 0.0122 & -0.8054
\end{pmatrix}.$$

Via l'algorithme JHM^2SH, nous avons $H_{JHM^2SH} = S^J_{JHM^2SH} A S_{JHM^2SH}$ tel que :

$$H_{JHM^2SH} = \begin{pmatrix}
1 & -5 & -11.1803 & -4.7635 & 7.5960 & 7.4333 & 1 & -3 & 34.9452 & -25.3872 & 22.7873 & -32.5195 \\
0 & 1 & 7.1554 & 6.1503 & -3.0382 & -1.4835 & -2 & 1 & 0.9126 & -1.0209 & -12.4281 & 15.4257 \\
0 & 0 & 11.9209 & 66.8730 & -70.9397 & -45.4873 & 0 & 0 & 426.1113 & -520.7106 & -177.8519 & 248.2220 \\
0 & 0 & 0 & -53.3206 & 56.0449 & 31.8975 & 0 & 0 & -404.0210 & 472.6685 & 129.9843 & -170.5259 \\
0 & 0 & 0 & 0 & -1.3503 & -1.2078 & 0 & 0 & 0 & -4.1654 & -4.5274 & 1.6392 \\
0 & 0 & 0 & 0 & 0 & 1.2894 & 0 & 0 & 0 & 0 & 3.3886 & 1.1146 \\
1 & -4 & -4.9193 & -8.9704 & 6.9364 & 3.2959 & 1 & -7 & -32.3865 & 36.8582 & 11.9556 & -14.6133 \\
0 & 1 & 6.7082 & 5.9444 & -10.2480 & -8.2100 & 0 & 1 & 5.0831 & -10.7383 & -16.9141 & 25.5773 \\
0 & 0 & 1.6000 & 9.6375 & -9.2301 & -6.6575 & 0 & 0 & 59.0791 & -76.8608 & -24.2698 & 36.4735 \\
0 & 0 & 0 & 0.4975 & 0.3703 & -0.6708 & 0 & 0 & 0 & -5.4706 & 0.6878 & 3.1356 \\
0 & 0 & 0 & 0 & -0.4886 & -0.1490 & 0 & 0 & 0 & 0 & -1.8844 & -0.9160 \\
0 & 0 & 0 & 0 & 0 & 0.7608 & 0 & 0 & 0 & 0 & 0 & -2.2634
\end{pmatrix}.$$

$$S_{JHM^2SH} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.4472 & -0.1449 & 0.1516 & 0.0937 & 0 & 0 & 1.9345 & -1.6961 & -0.7175 & 0.8685 \\
0 & 0 & -0.8944 & -0.0242 & -0.1820 & -0.0698 & 0 & 0 & 5.5460 & -5.4174 & 1.2058 & -1.1405 \\
0 & 0 & 0 & -0.1273 & 0.7470 & 0.5848 & 0 & 0 & -1.1180 & 1.7299 & 1.1646 & -2.5385 \\
0 & 0 & 0 & -0.3376 & 0.0056 & -0.0565 & 0 & 0 & -2.2361 & 3.3535 & 0.2098 & -0.2801 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.0415 & 0.3126 & -0.2140 & 0 & 0 & -1.1180 & -0.0560 & 0.6104 & 0.6365 \\
0 & 0 & 0 & 0.0208 & -0.1563 & 0.1070 & 0 & 0 & -0.5590 & 0.0280 & -0.3052 & -0.3183 \\
0 & 0 & 0 & -0.1342 & 0.6651 & 0.6469 & 0 & 0 & -1.1180 & 1.7922 & 1.6157 & -1.8443 \\
0 & 0 & 0 & -0.8191 & -0.0334 & -0.0486 & 0 & 0 & -5.0312 & 5.3328 & -0.0398 & -1.0669
\end{pmatrix}.$$

Via l'algorithme JHM^2SH2, nous avons $H_{JHM^2SH2} = S^J_{JHM^2SH2} AS_{JHM^2SH2}$ tel que :

$$H_{JHM^2SH2} = \begin{pmatrix}
1 & -5 & -11.1803 & -4.7635 & 7.5960 & 7.4333 & 1 & -3 & 34.9452 & -25.3872 & 22.7873 & -32.5195 \\
0 & 1 & 7.1554 & 6.1503 & -3.0382 & -1.4835 & -2 & 1 & 0.9126 & -1.0209 & -12.4281 & 15.4257 \\
0 & 0 & 11.9209 & 66.8730 & -70.9397 & -45.4873 & 0 & 0 & 426.1113 & -520.7106 & -177.8519 & 248.2220 \\
0 & 0 & 0 & -53.3206 & 56.0449 & 31.8975 & 0 & 0 & -404.0210 & 472.6685 & 129.9843 & -170.5259 \\
0 & 0 & 0 & 0 & -1.3503 & -1.2078 & 0 & 0 & 0 & -4.1654 & -4.5274 & 1.6392 \\
0 & 0 & 0 & 0 & 0 & 1.2894 & 0 & 0 & 0 & 0 & 3.3886 & 1.1146 \\
1 & -4 & -4.9193 & -8.9704 & 6.9364 & 3.2959 & 1 & -7 & -32.3865 & 36.8582 & 11.9556 & -14.6133 \\
0 & 1 & 6.7082 & 5.9444 & -10.2480 & -8.2100 & 0 & 1 & 5.0831 & -10.7383 & -16.9141 & 25.5773 \\
0 & 0 & 1.6000 & 9.6375 & -9.2301 & -6.6575 & 0 & 0 & 59.0791 & -76.8608 & -24.2698 & 36.4735 \\
0 & 0 & 0 & 0.4975 & 0.3703 & -0.6708 & 0 & 0 & 0 & -5.4706 & 0.6878 & 3.1356 \\
0 & 0 & 0 & 0 & -0.4886 & -0.1490 & 0 & 0 & 0 & 0 & -1.8844 & -0.9160 \\
0 & 0 & 0 & 0 & 0 & 0.7608 & 0 & 0 & 0 & 0 & 0 & -2.2634
\end{pmatrix}.$$

$$S_{JHM^2SH2} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.4472 & -0.1449 & 0.1516 & 0.0937 & 0 & 0 & 1.9345 & -1.6961 & -0.7175 & 0.8685 \\
0 & 0 & -0.8944 & -0.0242 & -0.1820 & -0.0698 & 0 & 0 & 5.5460 & -5.4174 & 1.2058 & -1.1405 \\
0 & 0 & 0 & -0.1273 & 0.7470 & 0.5848 & 0 & 0 & -1.1180 & 1.7299 & 1.1646 & -2.5385 \\
0 & 0 & 0 & -0.3376 & 0.0056 & -0.0565 & 0 & 0 & -2.2361 & 3.3535 & 0.2098 & -0.2801 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -0.0415 & 0.3126 & -0.2140 & 0 & 0 & -1.1180 & -0.0560 & 0.6104 & 0.6365 \\
0 & 0 & 0 & 0.0208 & -0.1563 & 0.1070 & 0 & 0 & -0.5590 & 0.0280 & -0.3052 & -0.3183 \\
0 & 0 & 0 & -0.1342 & 0.6651 & 0.6469 & 0 & 0 & -1.1180 & 1.7922 & 1.6157 & -1.8443 \\
0 & 0 & 0 & -0.8191 & -0.0334 & -0.0486 & 0 & 0 & -5.0312 & 5.3328 & -0.0398 & -1.0669
\end{pmatrix}.$$

Ainsi, nous avons

$$D_1 = J^{\mathrm{T}} S^{\mathrm{T}}_{MJHESS} J S_{JHM^2SH} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.4113 & 0 & 0 & 0 & 0 & 0 & -2.3962 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.3688 & 0 & 0 & 0 & 0 & 0 & -2.3371 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.7747 & 0 & 0 & 0 & 0 & 0 & 1.2880 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.6638 & 0 & 0 & 0 & 0 & 0 & -1.9982 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.4312 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.7118 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.2908 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.5065
\end{pmatrix}.$$

$$\|D_1^{-1} H_{MJHESS} D_1 - H_{JHM^2SH}\|_2 = 5.3417e-13.$$

$$D_2 = J^{\mathrm{T}} S^{\mathrm{T}}_{MJHESS} J S_{JHM^2SH2} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.4113 & 0 & 0 & 0 & 0 & 0 & -2.3962 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.3688 & 0 & 0 & 0 & 0 & 0 & -2.3371 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.7747 & 0 & 0 & 0 & 0 & 0 & 1.2880 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.6638 & 0 & 0 & 0 & 0 & 0 & -1.9982 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.4312 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.7118 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.2908 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.5065
\end{pmatrix}.$$

$$\|D_2^{-1} H_{MJHESS} D_2 - H_{JHM^2SH2}\|_2 = 3.7881e-13.$$

$$D_3 = J^{\mathrm{T}} S^{\mathrm{T}}_{JHM^2SH} J S_{JHM^2SH2} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}.$$

$$\|D_3^{-1} H_{JHM^2SH} D_3 - H_3\|_2 = 9.4799e-13.$$

L'exemple 3.7.5 nous montre que la réduction sous la forme J-Hessenberg est essentiellement unique au sens symplectique. C'est-à-dire qu'il existe une matrice triviale D_i qui relie deux réductions sous forme J-Hessenberg de la même matrice A.

3.8 Conclusion

Dans cet chapitre, nous avons introduit ($JHSH$) une réduction d'une matrice sous la forme J-Hessenberg, en se basent exclusivement sur les transformations de Householder symplectiques, qui sont des transvections, et s'écrivent comme l'identité, modifiée par le rajout d'un terme de rang un. Cette réduction est une étape cruciale pour une implémentation efficace de l'algorithme SR. De point de vue algébrique, notre méthode dans le cas symplectique est l'analogue de la réduction d'une matrice sous la forme Hessenberg, par des transformations de Householder, dans le cas Euclidien. Ensuite, l'algorithme $JHOSH$ est dérivé de la version générale de l'algorithme $JHSH$. Ce dernier, a montré la pertinence du choix optimal des paramètres libres afin de garantir une meilleure préservation de la J-orthogonalité, aussi bien que la factorisation. En outre, l'algorithme $JHOSH$ est considérablement amélioré en montrant que la moitié de ces transformations de Householder symplectiques, non forcément orthogonales, peuvent être remplacées par d'autres transformations élémentaires, qui ont l'avantage d'être symplectiques et orthogonales. Ce qui nous a conduit à deux variantes $JHMSH$ et $JHMSH2$ qui sont significativement plus stables numériquement. Ces algorithmes se comportent d'une manière similaire à l'algorithme $JHESS$. De plus, tous ces algorithmes peuvent rencontrer, au même moment et position, un breakdown ou bien peuvent être proche d'une telle instabilité numérique. Nous avons implémenté une stratégie efficace pour remédier à ces cas. Les nouveaux algorithmes intégrant cette stratégie sont nommés par $MJHESS$, $MJHSH$,JHM^2SH et JHM^2SH2. Leur efficacité a été testée et prouvée par des testes numériques.

Chapitre 4

L'algorithme SR

« *If your actions inspire others to dream more, learn more, do more and become more, you are a leader* »

John Quincy Adams (1767-1848)

Sommaire

4.1 Introduction

Afin de développer des méthodes rapides et efficaces numériquement, pour résoudre des problèmes dans l'algèbre linéaire tel que le problème symplectique de calcul des valeurs et/ou des vecteurs propres, nous devons utiliser la structure mathématique riche du problème d'une façon similaire à ce qui a été fait pour les problèmes de calcul des valeurs et/ou des vecteurs propres dans le cas des matrices symétriques/hermitiennes et orthogonaux/unitaires.

Par exemple, dans le cas symétrique, l'une des approches classiques actuelles est d'applique tout d'abord la réduction de la matrice A sous la forme tridiagonale symétrique suivie d'une séquence d'étapes QR implicites, qui préservent cette forme tridiagonale symétrique [49, 60]. Ces méthodes de préservation de la structure sont souhaitables, comme les propriétés importantes du problème original sont préservées au cours des calculs actuels et ne sont pas détruites par les erreurs d'arrondi. De plus, en général, de telles méthodes permettent d'avoir des calculs plus rapides que les méthodes d'objectives générales.

Pour le cas du problème de calcul des valeurs et/ou des vecteurs propres d'une matrice symétrique, par exemple, l'application de l'algorithme QR implicite à la matrice symétrique nécessite $\mathcal{O}(n^3)$ d'opérations arithmétiques par étape, tandis que l'application une étape de l'algorithme QR implicite à la matrice tridiagonale symétrique similaire ne nécessite que $\mathcal{O}(n)$ d'opérations arithmétiques seulement, où n est l'ordre de cette matrice.

Si de plus, la matrice considérée est de grande taille et creuses, alors l'algorithme QR peut ne pas être un outil approprié pour résoudre le problème de calcul des valeurs et/ou des vecteurs propres. Dans ce cas, on utilisera habituellement la méthode de Lanczos [97, 113], qui est une technique spécialement réglée pour résoudre ce type problèmes.

Les valeurs propres et les sous-espaces invariants de la matrice symplectique S peuvent être calculés par l'algorithme QR. Mais ce dernier ne peut pas profiter de la structure symplectique de la matrice S, il traitera la matrice S comme n'importe quelle matrice arbitraire de taille $2n \times 2n$.

En général, les valeurs propres calculées ne seront pas en paires réciproques $(\lambda,\ \lambda^{-1})$, malgré que les valeurs propres exactes ont cette propriété. Pire encore, des petites perturbations peuvent provoquer la perte des valeurs propres proches du cercle unité.

Ainsi pour préserver la structure symplectique de la matrice S, nous devons employer des transformations de similarité avec matrices symplectiques plutôt que les transformations de similarité avec les matrices unitaires habituels dans l'algorithme QR. Sachant que les matrices symplectiques, muni de la multiplication, forment un groupe. Dans l'intention d'assurer la stabilité numérique, il serait préférable d'utiliser des transformations symplectiques et orthogonales. Sous certaines conditions, une matrice symplectique peut être réduite sous la forme de J-Hessenberg en utilisant des matrices de transformation symplectiques et orthogonaux.

Par conséquent, les méthodes générales de type QR doivent être examinées pour en déduire des méthodes qui préservent la structure afin de résoudre le problème de calcul des valeurs et/ou des vecteurs propres dans le cadre symplectique.

La décomposition SR peut servir comme une base pour une méthode de type QR,

l'algorithme SR, qui fonctionne pour les matrices arbitraires de dimensions paires. Elle préserve la structure et permet de développer des implémentations rapides et efficaces.

Les méthodes général de type QR, dans lesquelles la décomposition QR est remplacées par d'autres décompositions, ont été étudiés par plusieurs, voir [40, 47, 48, 81, 116].

4.2 L'algorithme SR

L'algorithme SR où bien la factorisation SR est une méthode symplectique itérative, analogue à l'algorithme QR dans le cas Euclidien, pour calculer les valeurs et vecteurs propres d'une matrice. Cet algorithme qui est une méthode de préservation de la structure, est basé sur la décomposition SR à chaque itération.

Soit une A matrice arbitraire tel que $A_0 = A$, alors :

$$\begin{aligned} A_0 &= S_0 R_0 \\ A_1 &= R_0 S_0 = S_1 R_1 \\ A_2 &= R_1 S_1 = S_2 R_2 \\ A_3 &= R_2 S_2 = S_3 R_3 \end{aligned}$$

et ainsi de suite ...

$$A_{k+1} = R_k S_k = S_{k+1} R_{k+1}.$$

Donc,

$$\begin{aligned} A_{k+1} &= S_{k+1} R_{k+1} = R_k S_k \\ &= S_k^J (S_k R_k) S_k \\ &= S_k^J (A_k) S_k \end{aligned}$$

et ainsi de suite ...

$$\begin{aligned} &= S_k^J S_{k-1}^J \dots S_1^J (A_1) S_1 \dots S_{k-1} S_k \\ &= S_k^J S_{k-1}^J \dots S_1^J (R_0 S_0) S_1 \dots S_{k-1} S_k \\ &= S_k^J S_{k-1}^J \dots S_1^J (S_0^J (S_0 R_0) S_0) S_1 \dots S_{k-1} S_k \\ &= S_k^J S_{k-1}^J \dots S_1^J S_0^J (S_0 R_0) S_0 S_1 \dots S_{k-1} S_k \\ &= S_k^J S_{k-1}^J \dots S_1^J S_0^J (A_0) S_0 S_1 \dots S_{k-1} S_k. \end{aligned}$$

Ainsi,

$$A_{k+1} = \underbrace{S_k^J S_{k-1}^J \dots S_1^J S_0^J}_{S^J} A_0 \underbrace{S_0 S_1 \dots S_{k-1} S_k}_{S}.$$

Autrement dit, nous pouvons écrire la matrice A_{k+1} sous la forme $A_{k+1} = S^J A_0 S$, tel que la matrice $S = S_0 S_1 S_2 S_3 \dots S_{k-1} S_k$ et la matrice $S^J = S_k^J S_{k-1}^J \dots S_3^J S_2^J S_1^J S_0^J$.

Sous certaines hypothèses, la suite de matrices A_k converge vers une matrice J-triangulaire supérieure. Cette dernière matrice admet les mêmes valeurs propres que la matrice de départ A comme ils sont deux matrices similaires.

L'algorithme SR pour une matrice A arbitraire de taille $2n \times 2n$ sans shift est le suivant :

Algorithme SR sans shift :

La forme basique de l'algorithme SR sans shift.

1: Soit $A_0 = A$
2: $k = 0$
3: **répéter**
4: Calculer la décomposition SR de la matrice $A_k = S_k R_k$
5: Calculer $A_{k+1} = R_k S_k$
6: $k = k + 1$
7: **jusqu'à** la convergence
8: **Fin**

Comme nous avons

$$\begin{aligned} A_{k+1} &= R_k S_k \\ &= (S_k^J S_k) R_k S_k \\ &= S_k^J (S_k R_k) S_k \\ &= S_k^J A_k S_k, \end{aligned}$$

alors les matrices A_{k+1} et A_k sont deux matrices similaires.

4.2.1 La stratégie du simple shift

Pour accélère la convergence de l'algorithme ci-dessus nous utilisons la stratégie de shift. Le paramètre σ est appelé décalage ou shift en anglais. En effet, plus le shift σ est proche de la valeur propre λ de la matrice A plus la convergence est rapide. Si nous choisissons le bon shift σ, alors l'entrée du coin inférieur droit de la matrice A converge vers la valeur propre λ la plus proche de la valeur σ. Donc nous appliquons la décomposition SR sur la matrice $(A - \sigma I)$ au lieu de l'appliquer sur la matrice A.

L'algorithme SR pour une matrice A arbitraire de taille $2n \times 2n$ avec shift est le suivant :

Algorithme *SR* avec shift :

La forme basique de l'algorithme *SR* avec shift.

1: Soit $A_0 = A$
2: $k = 0$
3: **répéter**
4: Choisir un shift σ_k proche d'une valeur propre de la matrice A
5: Calculer la décomposition SR de la matrice $A_k - \sigma_k I = S_k R_k$
6: Calculer $A_{k+1} = R_k S_k + \sigma_k I$
7: $k = k+1$
8: **jusqu'à** la convergence
9: **Fin**

Lemme 4.2.1. *Les matrices A_k et A_{k+1} sont deux matrices similaires.*

Preuve. En effet,

$$\begin{aligned} A_{k+1} &= R_k S_k + \sigma_k I \\ &= (S_k^J S_k) R_k S_k + \sigma_k (S_k^J S_k) \\ &= S_k^J (S_k R_k + \sigma_k I) S_k \\ &= S_k^J A_k S_k. \end{aligned}$$

□

De plus, si la matrice R_k est une matrice inversible, alors nous pouvons écrire :

$$\begin{aligned} A_{k+1} &= R_k S_k + \sigma_k I \\ &= R_k S_k (R_k R_k^{-1}) + \sigma_k (R_k R_k^{-1}) \\ &= R_k (S_k R_k + \sigma_k) R_k^{-1} \\ &= R_k A_k R_k^{-1}. \end{aligned}$$

En fait, dans le cas où la constante σ_k est une valeur propre exacte de la matrice $A_k \in \mathbb{R}^{2n \times 2n}$, alors nous constatons que l'itération SR converge en une seule étape. En réalité, puisque la constante σ_k est une valeur propre de la matrice A_k, la matrice $A_k - \sigma_k I$ est singulier, donc la matrice R_k est aussi singulier, et ainsi une entrée diagonale de la matrice R_k doit être nulle. Supposons que $R_k(2n, 2n)$ égale à zéro. Ceci implique que la dernière ligne de la matrice $R_k S_k$ est nulle, alors la dernière ligne de la matrice $A_{k+1} = R_k S_k + \sigma_k I$ égale $\sigma_k e_{2n}^{\mathrm{T}}$, où e_{2n} est le $2n^{\text{ième}}$ vecteur de la base canonique de $\mathbb{R}^{2n}$. En d'autres termes, la dernière ligne de la matrice A_{k+1} est nulle sauf pour la valeur propre σ_k apparaissant dans l'entrée de position $(2n, 2n)$. Le bon choix de la constante σ_k est le dernier élément sur la diagonale de la matrice A_k.

Dans le cas où la constante σ_k n'est pas une valeur propre exacte, mais elle est toujours proche d'une valeur propre, alors la valeur $A_k(2n, 2n)$ va converger vers une valeur propre lorsque le bloc inférieur $A_{k+1}(2n, 1:2n-1)$ est assez petit. Autrement dit, la matrice $A_k - \sigma_k I$ est presque singulier, ce qui signifie que ses colonnes sont presque linéairement

dépendantes. Il s'ensuit que l'entrée $R_k(2n,2n)$ est petite, et nous pouvons montrer que l'entrée $A_{k+1}(2n,n)$ est aussi petite et ainsi $A_{k+1}(2n,2n) \approx \sigma_k$. Par conséquent, la valeur σ_k est révélé par la structure de la matrice A_{k+1} comme une valeur propre approximative de la matrice A_k. Cela suggère d'utiliser la valeur $A_k(2n,2n)$ comme shift σ_k pendant chaque itération, car si l'entrée $A_k(2n,n)$ est assez petite en comparaison avec l'entrée $A_k(2n,2n)$, alors ce choix de shift entraînera la convergence de l'entrée $A_k(2n,n)$ vers zéro. En fait, nous pouvons montrer que cette stratégie entraîne généralement la convergence de l'entrée $A_k(2n,n)$ vers un zéro quadratiquement. Cette amélioration par rapport à la convergence linéaire est due au changement du shift à chaque étape.

Donc si σ_k est une très bonne approximation de la valeur propre λ_k, nous attendons une convergence plus rapide.

4.2.2 La stratégie du double shift

La stratégie du simple shift que nous avons décrit dans la section précédente donne une convergence quadratique locale, n'est pas globalement convergente. Par exemple, prenons un cas particulièrement embêtant : imaginons ce qui se passe si nous voulons calculer une paire conjuguée des valeurs propres complexes d'une matrice réelle. Avec la stratégie du simple shift, les itérations

$$\begin{aligned} A_k - \sigma_k I &= S_k R_k \\ A_{k+1} &= R_k S_k + \sigma_k I, \end{aligned}$$

ne produiront jamais une itération complexe, un shift complexe ou une valeur propre complexe. Le meilleur que nous pouvons espérer est que notre shift initial soit le plus proche de la paire conjuguée de deux valeurs propres complexes plutôt que de toute autre chose dans le spectre.

Une façon de contourner cette difficulté, si les valeurs propres sont complexes et pour garder les calculs en arithmétique réelle, consiste à appliquer les deux shifts conjugués dans les itérations consécutives. Autrement dit, cette étape consiste à exécuter deux étapes consécutives de l'algorithme SR en utilisant σ_k et σ_{k+1} comme décalages tel que $\sigma_k = \overline{\sigma}_{k+1}$, pour obtenir une convergence quadratique dans le cas complexe. C'est-à-dire que nous calculons :

$$\begin{aligned} A_k - \sigma_k I &= S_k R_k \\ A_{k+1} &= R_k S_k + \sigma_k I \\ A_{k+1} - \sigma_{k+1} I &= S_{k+1} R_{k+1} \\ A_{k+2} &= R_{k+1} S_{k+1} + \sigma_{k+1} I. \end{aligned}$$

Ainsi, ces équations peuvent être manipulées pour avoir :

$$(S_k S_{k+1})(R_{k+1} R_k) = (A_k - \sigma_k I)(A_k - \sigma_{k+1} I). \tag{4.1}$$

En fait, nous avons

$$\begin{aligned}(S_kS_{k+1})(R_{k+1}R_k) &= S_k(S_{k+1}R_{k+1})R_k\\ &= S_k(A_{k+1}-\sigma_{k+1}I)R_k\\ &= S_k(R_kS_k+(\sigma_k-\sigma_{k+1})I)R_k\\ &= S_kR_kS_kR_k+(\sigma_k-\sigma_{k+1})S_kR_k\\ &= S_kR_k(S_kR_k+(\sigma_k-\sigma_{k+1})I)\\ &= (A_k-\sigma_kI)(A_k-\sigma_{k+1}I)\\ &= (A_k-\sigma_kI)(A_k-\sigma_{k+1}I)\\ &= (A_k-\sigma_kI)(A_k-\overline{\sigma}_kI).\end{aligned}$$

□

Notons que dans l'équation (4.1) les matrices $(A_k-\sigma_kI)(A_k-\sigma_{k+1}I)$, (S_kS_{k+1}) et $(R_{k+1}R_k)$ sont des matrices réelles même si la matrice de départ admet des valeurs propres complexes.

En effet,

$$\begin{aligned}(S_kS_{k+1})(R_{k+1}R_k) &= (A_k-\sigma_kI)(A_k-\overline{\sigma}_kI)\\ &= A_k^2-(\sigma_k+\overline{\sigma}_k)A_k+\sigma_k\overline{\sigma}_kI.\end{aligned}$$

Étant donné que $\sigma_k = x+iy$ et $\overline{\sigma}_k = x-iy$ sont une paire de complexe conjuguée, il s'ensuit que $(\sigma_k+\overline{\sigma}_k) = 2\,\Re\mathrm{e}(\sigma) = 2x$ et $\sigma_k\overline{\sigma}_k = |\sigma|^2 = x^2+y^2$ sont des réels. Par conséquent, $S_kS_{k+1}R_{k+1}R_k = (S_kS_{k+1})(R_{k+1}R_k)$ représente la décomposition SR d'une matrice réelle. □

En outre, la matrice S_kS_{k+1} est la matrice J-orthogonale qui implémente la transformation de la similarité de la matrice A_k pour obtenir la matrice réelle A_{k+2}. Autrement dit, nous avons

$$A_{k+2} = (S_kS_{k+1})^JA_k(S_kS_{k+1}).$$

Comme,

$$\begin{aligned}A_{k+2} &= R_{k+1}S_{k+1}+\sigma_{k+1}I\\ &= S_{k+1}^J(S_{k+1}R_{k+1}+\sigma_{k+1}I)S_{k+1}\\ &= S_{k+1}^J(A_{k+1}-\sigma_{k+1}I+\sigma_{k+1}I)S_{k+1}\\ &= S_{k+1}^J(A_{k+1})S_{k+1}\\ &= S_{k+1}^J(R_kS_k+\sigma_kI)S_{k+1}\\ &= S_{k+1}^J(S_k^J(S_kR_k+\sigma_kI)S_k)S_{k+1}\\ &= S_{k+1}^J(S_k^J(A_k-\sigma_kI+\sigma_kI)S_k)S_{k+1}\\ &= S_{k+1}^JS_k^JA_kS_kS_{k+1}\\ &= (S_kS_{k+1})^JA_k(S_kS_{k+1}).\end{aligned}$$

□

D'une manière générale, l'algorithme SR pour une matrice A arbitraire de taille $2n \times 2n$ est sous cette forme :

Algorithme 4.1 Algorithme SR basique :

La forme basique de l'algorithme SR.

1: Soit $A_0 = A$
2: **Pour** $k = 1, 2, \ldots$ **faire**
3: Choisir le polynôme de shift p_k
4: Calculer la décomposition SR de $p_k(A_{k-1}) = S_k R_k$
5: Calculer $A_k = S_k^{-1} A_{k-1} S_k$
6: **Fin Pour**
7: **Fin**

Remarque 4.2.1. *Les polynômes p_k de shift sont généralement choisis :*

— *Pour être sous la forme : $p_k(x) = (x - \sigma_k)$, où le shift σ_k prend la valeur du dernier élément diagonal de la matrice A_k, c'est-à-dire $\sigma_k = A_k(2n, 2n)$.*

— *Ou bien pour être sous la forme : $p_k(x) = (x - \sigma_k)(x - \overline{\sigma}_k)$, sachant que les constantes σ_k et $\overline{\sigma}_k$ sont les deux valeurs propres de la matrice*

$$B = \begin{pmatrix} A_k(k,k) & A_k(k,n+k) \\ A_k(k+n) & A_k(n+k,n+k) \end{pmatrix},$$

où cette matrice B est forme par le dernier élément diagonal de chaque blocs parmi les quatre blocs $A_{k_{11}}$, $A_{k_{12}}$, $A_{k_{21}}$ et $A_{k_{22}}$ de la matrice A_k, telle que la matrice A_k est sous la forme $A_k = \begin{pmatrix} A_{k_{11}} & A_{k_{12}} \\ A_{k_{21}} & A_{k_{22}} \end{pmatrix}$, de taille $2n \times 2n$. En d'autres termes, la matrice B est constituée par les entrées $A_k(k,k)$, $A_k(k,n+k)$, $A_k(k+n)$ et $A_k(n+k,n+k)$ de la matrice A_k.

Nous rappelons que la décomposition SR d'une matrice A réelle de taille $2n \times 2n$ est donnée par $A = SR$ où la matrice $S \in \mathbb{R}^{2n \times 2n}$ est une matrice symplectique, et la matrice $R \in \mathbb{R}^{2n \times 2n}$ est une matrice J-triangulaire supérieure. Presque chaque matrice A peut être décomposée en un tel produit puisque l'ensemble des matrices $2n \times 2n$ décomposables en SR est dense dans $\mathbb{R}^{2n \times 2n}$. Dans le cas où la décomposition de SR existe, nous pouvons construire d'autres décompositions SR de la matrice A en passant par des facteurs triviaux. Autrement dit, si la matrice D est une matrice triviale, $\tilde{S} = SD^{-1}$ et $\tilde{R} = DR$ alors $A = \tilde{S}\tilde{R}$ est une autre décomposition SR de la matrice A. Donc si la matrice A est inversible, alors ceci est la seule façon de créer d'autres décompositions SR. En d'autres termes, la décomposition SR est unique dans ce sens.

Proposition 4.2.1. *Soit $A \in \mathbb{R}^{2n \times 2n}$ une matrice inversible. Soient $A = SR$ et $A = \tilde{S}\tilde{R}$ deux décompositions SR de la matrice A. Alors il existe une matrice D trivial, c'est-à-dire une matrice à la fois symplectique et J-triangulaire de la forme*

$$D = \begin{pmatrix} C & F \\ 0 & C^{-1} \end{pmatrix} \tag{4.2}$$

où $C = \mathrm{diag}(c_1, \ldots, c_n)$ *et* $F = \mathrm{diag}(f_1, \ldots, f_n)$ *de telle sorte que* $\begin{cases} \tilde{S} = & SD^{-1} \\ \tilde{R} = & DR \end{cases}$

Preuve. Puisque la matrice $A = SR = \tilde{S}\tilde{R}$, alors nous avons $\tilde{S}^{-1}S = \tilde{R}R^{-1} = D$ avec la matrice $D = \begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix}$ est une matrice symplectique et J-triangulaire supérieure. Comme de plus, la matrice D est une matrice symplectique, alors $D^{\mathrm{T}}JD = J$ et nous aurons

$$\begin{cases} D_{11}^{\mathrm{T}}D_{21} - D_{21}^{\mathrm{T}}D_{11} = & D_{12}^{\mathrm{T}}D_{22} - D_{22}^{\mathrm{T}}D_{12} = 0 \\ D_{12}^{\mathrm{T}}D_{21} - D_{22}^{\mathrm{T}}D_{11} = & -I \end{cases}.$$

De plus, la matrice D est aussi une matrice J-triangulaire supérieure ainsi les blocs D_{11}, D_{12}, D_{21} et D_{22} sont des matrices triangulaires supérieures et la matrice D_{21} a des zéros sur la diagonale.

En outre, la matrice D est inversible, alors les matrices D_{11} et D_{22} sont aussi deux matrices inversibles et nous pouvons écrire

$$D_{11}^{\mathrm{T}}D_{21}D_{11}^{-1} = D_{21}^{\mathrm{T}}$$

et

$$D_{22}^{\mathrm{T}}D_{12}D_{22}^{-1} = D_{12}^{\mathrm{T}}.$$

De la résolution de ces deux équations pour chaque composante, nous obtenions $D_{21} = 0$ et la matrice D_{12} est une matrice diagonale. Ainsi $D_{22}^{\mathrm{T}}D_{11} = I$ ce qui implique que les matrices D_{11} et D_{22} sont deux matrices diagonales et $D_{11} = D_{22}^{-1}$.

En fin, nous aurons la matrice D sous la forme $D = \begin{pmatrix} C & F \\ 0 & C^{-1} \end{pmatrix}$ avec la matrice $C = \mathrm{diag}(c_1, \ldots, c_n)$ et la matrice $F = \mathrm{diag}(f_1, \ldots, f_n)$. □

L'inconvénient de l'algorithme *SR* ci-dessus (l'algorithme 4.1) est que chaque étape nécessite une décomposition *SR* qui coût $\mathcal{O}((2n)^3)$. Par suite, si nous faisons qu'une seule itération pour chaque valeur propre alors le coût total de l'algorithme *SR* serait $\mathcal{O}((2n)^4)$. Heureusement, nous pouvons réduire ce coût à $\mathcal{O}((2n)^3)$ si nous réduisons tout d'abord la matrice A de départ sous la forme J-Hessenberg. En effet, le calcul de la décomposition *SR* d'une matrice sous la forme J-Hessenberg de taille $2n \times 2n$ ne nécessite que $\mathcal{O}((2n)^2)$ seulement, par rapport à $\mathcal{O}((2n)^3)$ si la matrice de départ est arbitraire. Par conséquent, après la réduction initiale sous la forme J-Hessenberg, le coût total devient $\mathcal{O}((2n)^3)$. En conséquence, pour une implémentation raisonnable de l'algorithme *SR* nous devons tout d'abord commencer par la réduction de la matrice A de départ sous la forme J-Hessenberg.

Algorithme 4.2 Algorithme *SR* avec réduction sous la forme J-Hessenberg

L'algorithme *SR* pour une matrice de taille $2n \times 2n$ arbitraire A avec réduction sous la forme J-Hessenberg.

1: Réduire la matrice A sous la forme d'une matrice J-Hessenberg H_0 telle que : $H_0 = S_0^{-1} A S_0 = S_0^J A S_0$
avec la matrice S_0 est une matrice symplectique.
2: Soit $S = S_0$
3: **Pour** $k = 1, 2, \ldots$ **faire**
4: Choisir le polynôme de shift p_k
5: Calculer la décomposition *SR* de $p_k(H_{k-1}) = S_k R_k$
6: Calculer $H_k = S_k^{-1} H_{k-1} S_k$
7: Calculer $S = S S_k$
8: **Fin Pour**
9: **Fin**

Remarque 4.2.2. *L'algorithme SR préserve la forme J-Hessenberg et dans toutes les itération les matrices H_k sont des matrices sous la forme J-Hessenberg et semblables.*

En effet, si la matrice $p_k(A_{k-1})$ est inversible et $p_k(A_{k-1}) = S_k R_k$, comme de plus la matrice S_k est une matrice symplectique, alors la matrice R_k est une matrice inversible. Par conséquent, vu que les matrices $p_k(A_{k-1})$ et A_{k-1} commutent, nous aurons :

$$\begin{aligned} A_k &= S_k^{-1} A_{k-1} S_k \\ &= R_k p_k(A_{k-1})^{-1} A_{k-1} p_k(A_{k-1}) R_k^{-1} \\ &= R_k A_{k-1} R_k^{-1}. \end{aligned}$$

Par suite, si la matrice A_{k-1} est une matrice sous la forme J-Hessenberg, alors la matrice A_k est aussi une matrice sous la forme J-Hessenberg. En fait, la matrice A_k est le produit d'une matrice A_{k-1} sous la forme J-Hessenberg et de deux matrices R_k et R_k^{-1} qui sont J-triangulaires. Autrement dit, les transformations de similarité symplectique préservent la structure Hamiltonienne :

$$\begin{aligned} [A_k J] &= (S_k^{-1} A_{k-1} S_k) J = S_k^{-1} A_{k-1} J S_k^{-\mathrm{T}} = S_k^{-1} J^{\mathrm{T}} A_{k-1}^{\mathrm{T}} S_k^{-\mathrm{T}} = J^{\mathrm{T}} S_k^{\mathrm{T}} A_{k-1}^{\mathrm{T}} S_k^{-\mathrm{T}} \\ &= [(S_k^{-1} A_{k-1} S_k) J]^{\mathrm{T}} = [A_k J]^{\mathrm{T}}. \end{aligned}$$

Dans le cas où la matrice $p_k(A_{k-1})$ est singulière, nous devons vérifier la forme spéciale de la matrice S_k pour savoir si la matrice A_k est sous la forme souhaitée dans le cas où la matrice A_{k-1} est sous la forme J-Hessenberg. Dans ce cas, nous pouvons diviser le problème en deux sous problèmes de dimensions inférieures : Si $\mathrm{rang}(A_{k-1}) = 2n - 2\nu = 2j$, alors le problème se divise en un problème de taille $2j \times 2j$ avec la forme J-Hessenberg et un problème de taille $2\nu \times 2\nu$ dont les valeurs propres sont exactement les shifts qui sont les valeurs propres de la matrice A_{k-1} ; autrement dit, ces sont les valeurs propres de la matrice A.

En raison de l'unicité essentielle de la réduction sous la forme J-Hessenberg, nous pouvons implémenter l'algorithme *SR* sans calculer explicitement les décompositions

SR de la matrice $p_k(A_{k-1}) = S_k R_k$. Il suffit de calculer la première colonne de la matrice $p_k(A_{k-1})$ par analogie à la version implicite de l'algorithme QR.

Nous pouvons effectuer implicitement une étape de l'algorithme SR de la façon suivante :

— Calculer une matrice symplectique $\tilde{S}_k$ tel que $\tilde{S}_k^{-1} p_k(A_{k-1}) e_1 = \alpha e_1$ avec le scalaire $\alpha \in \mathbb{R}$.

— Calculer $\hat{A}_k = \tilde{S}_k^{-1} A_{k-1} \tilde{S}_k$.

— Calculer une matrice symplectique $\hat{S}_k$ tel que la matrice $\hat{S}_k^{-1} \hat{A}_k \hat{S}_k$ soit sous la forme J-Hessenberg.

La matrice résultante de J-Hessenberg

$$\hat{S}_k^{-1} \hat{A}_k \hat{S}_k$$

est essentiellement la même que la matrice

$$S_k^{-1} A_{k-1} S_k$$

parce que $\hat{S}_k = D S_k$ pour une matrice triviale D (voir la proposition 4.2.1).

L'application de la première transformation $\tilde{S}_k$ à la matrice A_{k-1}, qui est sous la forme J-Hessenberg, donne une matrice avec une forme presque J-Hessenberg ayant une petite bosse ou "bulge". Ainsi, il y aura des entrées supplémentaires dans le coin supérieur gauche de chaque bloc $n \times n$ de la matrice $\tilde{S}_k^{-1} A_{k-1} \tilde{S}_k$.

Les transformations implicites restantes (c'est-à-dire les matrices $\hat{S}_k$) effectuent la chasse du bosse ou la "bulge chasing" pour restaurer la forme J-Hessenberg.

4.3 Le S-théorème implicite

Nous notons par $K(A, x, j)$ $(\in \mathbb{K}^{2n \times j})$ la matrice de Krylov de taille $(2n \times j)$, avec $A \in \mathbb{R}^{2n \times 2n}$, $x \in \mathbb{R}^{2n}$ et $j \in \mathbb{N}$, telle que

$$K(A, x, j) = [x, Ax, A^2 x, \ldots, A^{j-1} x] \in \mathbb{R}^{2n \times j}.$$

Théorème 4.3.1. *Soient la matrice $A \in \mathbb{R}^{2n \times 2n}$, la matrice $S \in \mathbb{S}_{2n}$ symplectique de taille $(2n \times 2n)$ et la permutation P définie par la matrice*

$$P = [e_1, e_3, \ldots, e_{2n-1}, e_2, e_4, \ldots, e_{2n}] \in \mathbb{R}^{2n \times 2n}.$$

Nous notons par le vecteur s_1 la première colonne de la matrice S ($s_1 = Se_1$). Alors nous avons :

1. *Soit $K(A, s_1, 2n)$ une matrice de Krylov inversible. Si la matrice $K(A, s_1, 2n)P$ admet une décomposition SR tel que $K(A, s_1, 2n)P = SR$, alors la matrice $H = S^{-1}AS$ est une matrice sous la forme J-Hessenberg irréductible.*
2. *Si la matrice $H = S^{-1}AS$ est une matrice sous la forme J-Hessenberg, alors la matrice $K(A, s_1, 2n)P$ admet une décomposition SR : $K(A, s_1, 2n)P = SR$. De plus, si la matrice H sous la forme J-Hessenberg est une matrice irréductible, alors la matrice R est une matrice inversible.*

3. *Soit $\tilde{S} \in \mathbb{S}_{2n}$ une matrice symplectique de taille $(2n \times 2n)$ dont la première colonne $\tilde{s}_1 = \tilde{S}e_1 = \lambda s_1$, avec $\lambda \in \mathbb{R} \setminus \{0\}$ et soient $H = S^{-1}AS$ et $\tilde{H} = \tilde{S}^{-1}A\tilde{S}$ deux matrices sous la forme J-Hessenberg où la matrice H est une matrice irréductible. Alors il existe une matrice*

$$D = \begin{pmatrix} C & F \\ 0 & C^{-1} \end{pmatrix},$$

avec $C = \mathrm{diag}(c_1, \ldots, c_n)$ et $F = \mathrm{diag}(f_1, \ldots, f_n)$ de telles sorte que $S = \tilde{S}D$ et

$$H = D^{-1}\tilde{H}D.$$

Preuve.

1. Soient la matrice $\tilde{S} = SP^{\mathrm{T}}$, les scalaires $\alpha_0, \ldots, \alpha_{2n-1} \in \mathbb{R}$ tels que

$$A^{2n}s_1 = \sum_{i=0}^{2n-1} \alpha_i A^i s_1,$$

et soit la matrice

$$C = \begin{pmatrix} 0 & 0 & & 0 & \alpha_0 \\ 1 & 0 & & 0 & \alpha_1 \\ 0 & 1 & \cdots & 0 & \alpha_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \alpha_{2n-1} \end{pmatrix},$$

alors nous avons

$$K(A, s_1, 2n)C = [As_1, A^2s_1, \ldots, A^{2n}s_1] = AK(A, s_1, 2n). \tag{4.3}$$

Si la matrice $K(A, s_1, 2n)P$ admet une décomposition SR tel que :

$$K(A, s_1, 2n)P = SR.$$

Alors,

$$K(A, s_1, 2n) = SRP^{\mathrm{T}} = SP^{\mathrm{T}}PRP^{\mathrm{T}} = \tilde{S}\tilde{R},$$

avec la matrice $\tilde{R}$ est une matrice triangulaire supérieure $\tilde{R} = PRP^{\mathrm{T}}$. Ainsi, l'équation (4.3) ci-dessus devient

$$\tilde{S}\tilde{R}C = A\tilde{S}\tilde{R},$$

donc

$$A\tilde{S} = \tilde{S}\tilde{R}C\tilde{R}^{-1}.$$

La matrice $\tilde{H} = \tilde{R}C\tilde{R}^{-1}$ est une matrice de Hessenberg supérieure irréductible et par conséquent la matrice $H = S^{-1}AS = P^{\mathrm{T}}\tilde{S}^{-1}A\tilde{S}P = P^{\mathrm{T}}\tilde{H}P$ est une matrice sous la forme J-Hessenberg supérieure irréductible.

2. Soient les matrices $H = S^{-1}AS$ et $\tilde{H} = PHP^{\mathrm{T}}$. Pour tous $i \in \mathbb{N}$ nous avons $A^i s_1 = SH^i S^{-1}s_1 = SH^i e_1$ et $\tilde{H}^i e_1 = PH^i P^{\mathrm{T}}e_1 = PH^i e_1$. Ainsi,

$$K(A, s_1, 2n) = SK(H, e_1, 2n),$$

$$K(\tilde{H}, e_1, 2n) = PK(H, e_1, 2n)$$

et par conséquent nous avons

$$K(A, s_1, 2n)P = SK(H, e_1, 2n)P = SP^{\mathrm{T}}K(\tilde{H}, e_1, 2n)P.$$

Puisque la matrice $\tilde{H}$ est une matrice de Hessenberg supérieure, alors la matrice $\tilde{R} = K(\tilde{H}, e_1, 2n)$ est une matrice triangulaire supérieure. Le $i^{\text{ième}}$ élément de la diagonale de la matrice $\tilde{R}$ est égal à $\prod_{j=1}^{i-1} \tilde{h}_{j+1,j}$ pour $i \geqslant 2$. En conséquence, si la matrice $\tilde{H}$ de Hessenberg est irréductible autrement dit si la matrice H de J-Hessenberg est irréductible, alors la matrice $\tilde{R}$ est une matrice inversible.

3. Si les matrices $H = S^{-1}AS$ et $\tilde{H} = \tilde{S}^{-1}A\tilde{S}$ sont deux matrices de J-Hessenberg, où $\tilde{s}_1 = \lambda s_1$ est la première colonne de la matrice $\tilde{S}$, alors d'après le deuxième point de ce théorème nous aurons deux décomposition SR tel que :

$$K(A, s_1, 2n)P = SR = \frac{1}{\lambda}\tilde{S}\tilde{R}.$$

En fin, d'après la proposition 4.2.1 nous aurons le résultat à démontre, c'est-à-dire qu'il existe une matrice D symplectique et J-triangulaire à la fois (triviale) tel que

$$D = \begin{pmatrix} C & F \\ 0 & C^{-1} \end{pmatrix},$$

avec $C = \operatorname{diag}(c_1, \ldots, c_n)$ et $F = \operatorname{diag}(f_1, \ldots, f_n)$ de telles sorte que $S = \tilde{S}D$ et $H = D^{-1}\tilde{H}D$.

□

L'implémentation de la version implicite de l'algorithme SR nécessite le S-théorème implicite qui découle de théorème 4.3.1 et la proposition 4.2.1. Le S-théorème implicite est l'analogue de le Q-théorème implicite dans le cas Euclidien.

Théorème 4.3.2. *Le S-théorème implicite.*
Soit la matrice $H = S^J AS$ tels que la matrice S est une matrice symplectique et $s_1 = Se_1$ sa première colonne. Nous supposons que la matrice H est une matrice de J-Hessenberg supérieure irréductible. Alors de la première colonne s_1 de la matrice S nous pouvons déterminer tous les autres vecteurs colonnes de cette matrice de la position 2 jusqu'à la position $2n$ d'une façon essentiellement unique.

Preuve. Nous supposons que les deux matrices $H = S^J AS$ et $\tilde{H} = \tilde{S}^J A\tilde{S}$ sont deux matrices de J-Hessenberg irréductibles, tels que les matrices S et $\tilde{S}$ sont deux matrices symplectiques et nous supposons que leurs premières colonnes sont égaux $s_1 = Se_1 = \tilde{S}e_1 = \tilde{s}_1$.
Soit la matrice $D = \tilde{S}^J S$, alors nous avons

$$\tilde{H}D = \tilde{H}\tilde{S}^J S = \tilde{S}^J A\tilde{S}\tilde{S}^J S = \tilde{S}^J AS = \tilde{S}^J SH = DH.$$

Donc nous avons $\tilde{H}D = DH$ et ainsi nous pouvons écrire $H = D^{-1}\tilde{H}D$. D'après le troisième point de théorème 4.3.1 et la proposition 4.2.1, nous obtenons : La

matrice D est une matrice triviale c'est à dire qu'elle est symplectique et J-triangulaire à la fois telle qu'elle a la forme suivante :

$$D = \begin{pmatrix} C & F \\ 0 & C^{-1} \end{pmatrix},$$

avec les matrices diagonales C et F sont sous la forme $C = \text{diag}(c_1, \ldots, c_n)$ et $F = \text{diag}(f_1, \ldots, f_n)$.
De plus comme les matrices S et $\tilde{S}$ ont la même première colonne et $S = \tilde{S}D$ alors $De_1 = e_1$ et $De_{n+1} = e_{n+1}$.

Ainsi, grâce à l'unicité essentielle de la réduction sous la forme J-Hessenberg, si nous avons la première colonne de la matrice symplectique S alors nous pouvons déterminer toutes les restes des colonnes de cette matrice d'une façon essentiellement unique. □

En outre, le S-théorème implicite peut servir comme une base pour la construction d'un algorithme *SR* implicite pour les matrices de J-Hessenberg, tout comme le Q-théorème implicite qui fournit une base pour l'algorithme *QR* implicite sur les matrices de Hessenberg supérieures. Dans les deux cas l'unicité dépend du caractère irréductible de la matrice J-Hessenberg ou Hessenberg. Bien que la décomposition *QR* d'une matrice toujours existe, mais la décomposition *SR* peut ne pas exister.

4.4 L'algorithme SR implicite

4.4.1 L'algorithme SR implicite avec shift

L'algorithme *SR* implicite avec shift admet le même principe que la version explicite de l'algorithme *SR* avec shift. Nous rappelons que :

$$\begin{aligned} S_k R_k &= A_{k-1} - \sigma_k I, \\ A_k &= R_k S_k + \sigma_k I, \end{aligned}$$

ainsi

$$A_k = S_k^J A_{k-1} S_k.$$

D'après le S-théorème implicite pour calculer la matrice A_k à partir de la matrice A_{k-1} dans l'algorithme *SR*, nous n'aurons besoin que de calculer la première colonne de la matrice S_k, qui est parallèle à la première colonne de la matrice $A_{k-1} - \sigma_k I$. Ensuite, nous choisissons les autres colonnes de la matrice S_k de telles sorte que la matrice S_k soit une matrice symplectique (J-orthogonale) et la matrice A_k soit une matrice de J-Hessenberg irréductible.

En effet, nous choisissons la première colonne $s_1 = S_k e_1$ de la matrice S_k pour être

proportionnelle à la première colonne de la matrice $p_k(A_{k-1}) = A_{k-1} - \sigma_k I$:

$$s_1 = \begin{pmatrix} a_{1,1} - \sigma_k \\ 0 \\ \vdots \\ 0 \\ a_{n+1,1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Cela signifie que la matrice S_k est la même que dans la décomposition SR de la matrice $A_{k-1} - \sigma_k I$.

4.4.2 L'algorithme SR implicite avec double shift

L'algorithme SR implicite avec double shift consiste à faire deux itérations de shift en même temps en utilisent σ et son conjugué $\overline{\sigma}$ comme shift. Par suite, nous pouvons avoir des valeurs propres complexes ou réelles sachant que les matrices intermédiaires A_k restent réelles durant toutes les itérations.

Nous faisons un shift avec la valeur σ :

$$\begin{aligned} A_0 - \sigma I &= S_1 R_1, \\ A_1 &= R_1 S_1 + \sigma I, \\ &= S_1^J (A_0 - \sigma I) S_1 + \sigma I, \end{aligned}$$

ainsi

$$A_1 = S_1^J A_0 S_1.$$

Puis, nous faisons un autre shift avec la valeur $\overline{\sigma}$:

$$\begin{aligned} A_1 - \overline{\sigma} I &= S_2 R_2, \\ A_2 &= R_2 S_2 + \overline{\sigma} I \\ &= S_2^J (A_1 - \overline{\sigma} I) S_2 + \overline{\sigma} I \\ &= S_2^J A_1 S_2. \end{aligned}$$

D'où, nous avons :

$$A_2 = S_2^J A_1 S_2 = S_2^J S_1^J A_0 S_1 S_2.$$

Donc nous pouvons choisir les matrices S_1 et S_2 de telles sorte que leur produit la matrice $S_1 S_2$ soit une matrice réelle et aussi la matrice A_2 soit une matrice réelle. En effet, comme $S_1 S_2 R_2 R_1$ est une décomposition SR de la matrice réel A_0 avec $S = S_1 S_2$ et $R = R_2 R_1$, alors nous pouvons choisir les matrices $S_1 S_2$ et $R_2 R_1$ de telles sorte qu'elles

soient des matrices réelles.

$$\begin{aligned} S_1S_2R_2R_1 &= S_1(A_1-\overline{\sigma}I)R_1 \\ &= S_1(R_1S_1+(\sigma-\overline{\sigma})I)R_1 \\ &= S_1R_1S_1R_1+(\sigma-\overline{\sigma})S_1R_1 \\ &= (A_0-\sigma I)(A_0-\sigma I)+(\sigma-\overline{\sigma})(A_0-\sigma I) \\ &= A_0^2-2\,\Re\mathrm{e}(\sigma)A_0+|\sigma|^2I. \end{aligned}$$

D'une manière générale, nous avons :

$$\begin{aligned} A_{k+2}=S_kS_{k+1}R_{k+1}R_k &= S_k(A_k-\overline{\sigma}_kI)R_k \\ &= S_k(R_kS_k+(\sigma_k-\overline{\sigma}_k)I)R_k \\ &= S_kR_kS_kR_k+(\sigma_k-\overline{\sigma}_k)S_kR_k \\ &= (A_{k-1}-\sigma_kI)(A_{k-1}-\sigma_kI)+(\sigma_k-\overline{\sigma}_k)(A_{k-1}-\sigma_kI) \\ &= (A_{k-1}-\sigma_kI)(A_{k-1}-\overline{\sigma}_kI) \\ &= A_{k-1}^2-2\,\Re\mathrm{e}(\sigma_k)A_{k-1}+|\sigma_k|^2I. \end{aligned}$$

Sachant que

$$\begin{aligned} A_{k-1}-\sigma_kI &= S_kR_k, \\ A_k &= R_kS_k+\sigma_kI \\ &= S_k^J(A_{k-1}-\sigma_kI)S_k+\sigma_kI \\ &= S_k^JA_{k-1}S_k, \end{aligned}$$

et

$$\begin{aligned} A_k-\overline{\sigma}_kI &= S_{k+1}R_{k+1}, \\ A_{k+1} &= R_{k+1}S_{k+1}+\overline{\sigma}_kI \\ &= S_{k+1}^J(A_k-\overline{\sigma}_kI)S_{k+1}+\overline{\sigma}_kI \\ &= S_{k+1}^JA_kS_{k+1} \\ &= S_{k+1}^JS_k^JA_{k-1}S_kS_{k+1}. \end{aligned}$$

En conclusion, la première colonne de la matrice S_1S_2 (respectivement la matrice S_kS_{k+1}) est proportionnelle à la première colonne de la matrice du polynôme de shift

$$p_1(A_0)=A_0^2-2\,\Re\mathrm{e}(\sigma)A_0+|\sigma|^2I$$

(respectivement du polynôme de shift $p_k(A_{k-1})=A_{k-1}^2-2\,\Re\mathrm{e}(\sigma_k)A_{k-1}+|\sigma_k|^2I$). Nous rappelons que la matrice A_0 (respectivement la matrice A_{k-1}) est une matrice de J-Hessenberg de taille $2n\times 2n$, ainsi la première colonne est sous la forme suivante :

$$\begin{pmatrix} a_{1,1}^2 + a_{1,n+1}a_{n+1,1} - 2\,\Re\mathrm{e}(\sigma)a_{1,1} + |\sigma|^2 \\ a_{2,n+1}a_{n+1,1} \\ 0 \\ \vdots \\ 0 \\ a_{n+1,1}(a_{1,1} + a_{n+1,n+1} - 2\,\Re\mathrm{e}(\sigma)) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Les restes des colonnes de la matrice S_1S_2 (respectivement la matrice S_kS_{k+1}) sont calculées implicitement par le S-théorème implicite.

Remarque 4.4.1. *Dans le calcul de la première colonne de la matrice S_kS_{k+1} qui est proportionnelle à la première colonne de la matrice du polynôme $p_k(A_{k-1}) = A_{k-1}^2 - 2\,\Re\mathrm{e}(\sigma_k)A_{k-1} + |\sigma_k|^2 I$, nous n'avons pas besoin de calculer le carré de toute la matrice A_{k-1}. Mais nous calculons seulement que les termes dont nous avons besoin d'utiliser, c'est-à-dire nous calculons la première, la deuxième, la $(n+1)^{ième}$ et la $(n+2)^{ième}$ entrées de la première colonne. En d'autres termes, pour avoir la première colonne de la matrice du polynôme de shift, nous calculons :*

— *Le produit d'une matrice 4×2 par un vecteur 2×1 pour avoir la première colonne de la matrice A_{k-1}^2 (une colonne similaire de taille 4 qui contient seulement les éléments non nuls de la première colonne) :*

$$\begin{pmatrix} a_{1,1} & a_{1,n+1} \\ 0 & a_{2,n+1} \\ a_{n+1,1} & a_{n+1,n+1} \\ 0 & 0 \end{pmatrix} \times \begin{pmatrix} a_{1,1} \\ a_{n+1,1} \end{pmatrix} = \begin{pmatrix} a_{1,1}^2 + a_{1,n+1}a_{n+1,1} \\ a_{2,n+1}a_{n+1,1} \\ a_{1,1}a_{n+1,1} + a_{n+1,1}a_{n+1,n+1} \\ 0 \end{pmatrix}. \tag{4.4}$$

— *Ensuite, pour la soustraction de la première colonne de la matrice $2\,\Re\mathrm{e}(\sigma_k)A_{k-1}$, plus précisément nous modifions seulement la première et troisième entrées du vecteur (4.4) parce que la matrice A_{k-1} est de J-Hessenberg et ainsi nous avons :*

$$\begin{pmatrix} a_{1,1}^2 + a_{1,n+1}a_{n+1,1} \\ a_{2,n+1}a_{n+1,1} \\ a_{1,1}a_{n+1,1} + a_{n+1,1}a_{n+1,n+1} \\ 0 \end{pmatrix} - \begin{pmatrix} 2\,\Re\mathrm{e}(\sigma_k)a_{1,1} \\ 0 \\ 2\,\Re\mathrm{e}(\sigma_k)a_{n+1,1} \\ 0 \end{pmatrix} = \begin{pmatrix} a_{1,1}^2 + a_{1,n+1}a_{n+1,1} - 2\,\Re\mathrm{e}(\sigma_k)a_{1,1} \\ a_{2,n+1}a_{n+1,1} \\ a_{n+1,1}(a_{1,1} + a_{n+1,n+1} - 2\,\Re\mathrm{e}(\sigma_k)) \\ 0 \end{pmatrix}. \tag{4.5}$$

— *Puis, pour l'addition de la matrice $|\sigma_k|^2 I$, nous ajoutons la valeur $|\sigma|^2$ à la première entrée du vecteur* (4.5) *tel que nous avons :*

$$\begin{pmatrix} a_{1,1}^2 + a_{1,n+1}a_{n+1,1} - 2\,\Re e(\sigma_k)a_{1,1} \\ a_{2,n+1}a_{n+1,1} \\ a_{n+1,1}(a_{1,1} + a_{n+1,n+1} - 2\,\Re e(\sigma_k)) \\ 0 \end{pmatrix} + \begin{pmatrix} |\sigma_k|^2 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a_{1,1}^2 + a_{1,n+1}a_{n+1,1} - 2\,\Re e(\sigma_k)a_{1,1} + |\sigma_k|^2 \\ a_{2,n+1}a_{n+1,1} \\ a_{n+1,1}(a_{1,1} + a_{n+1,n+1} - 2\,\Re e(\sigma_k)) \\ 0 \end{pmatrix}. \quad (4.6)$$

— *En fin, pour avoir la première colonne de la matrice du polynôme de shift, nous injectons ce vecteur* (4.6) *de dimension* 4 *dans un vecteur de dimension* $2n$ *tel que :*

$$\begin{pmatrix} a_{1,1}^2 + a_{1,n+1}a_{n+1,1} - 2\,\Re e(\sigma_k)a_{1,1} + |\sigma_k|^2 \\ a_{2,n+1}a_{n+1,1} \\ 0 \\ \vdots \\ 0 \\ a_{n+1,1}(a_{1,1} + a_{n+1,n+1} - 2\,\Re e(\sigma_k)) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Nous pouvons écrire

$$\begin{aligned} p_k(A_{k-1}) &= A_{k-1}^2 - 2\,\Re e(\sigma_k)A_{k-1} + |\sigma_k|^2 I \\ &= A_{k-1}^2 + b_k A_{k-1} + c_k I, \end{aligned}$$

tels que $b_k = -2\,\Re e(\sigma_k)$ et $c_k = |\sigma_k|^2$.

Au début, nous commençons par un algorithme qui calcule les coefficients b_k et c_k du polynôme $p_k(z) = z^2 + b_k z + c_k$. En effet, nous calculons ces coefficients à chaque itération, dans l'algorithme *SR*, afin de faire le shift.

Algorithme 4.3 Algorithme SRpoly :

L'algorithme calcule les coefficients b_k et c_k du polynôme de shift qui s'écrit sous la forme $p_k(z) = z^2 + b_k z + c_k$ de la matrice H de J-Hessenberg.

ENTRÉE(S) : La matrice $H = [a_1, a_2, \ldots, a_{2n}] \in \mathbb{R}^{2n \times 2n}$ de J-Hessenberg.

SORTIE(S) : Deux coefficients b et c du polynôme de shift $p(H) = H^2 + bH + c$.

1: Fonction $[b,c] = SRpoly(H)$
   % Nous calculons les coefficients b et c du polynôme $p(H)$ tel que $p(H) = H^2 + bH + c = (H-\sigma)(H-\overline{\sigma})$.
2: global $itr\ itc$;
3: $den =$ longueur (H);
4: $n = den/2$;
5: $B = H([n,den],[n,den])$;
   % la trace de la matrice B.
6: $trB = B(1,1) + B(2,2)$;
   % le déterminant de la matrice B.
7: $detB = B(1,1) \times B(2,2) - B(1,2) \times B(2,1)$;
   % Si $(trB)^2 > 4 \times detB$ alors les valeurs propres sont des valeurs réelles : Nous utilisons comme double shift la valeur propre la plus proche de l'entrée $H(den,den)$.
8: **Si** $(trB)^2 > 4 \times detB$ **alors**
9: $\lambda_1 = (trB + \sqrt{(trB)^2 - 4 \times detB})/2$;
10: $\lambda_2 = (trB - \sqrt{(trB)^2 - 4 \times detB})/2$;
11: **Si** $|\lambda_1 - H(end,end)| < |\lambda_2 - H(end,end)|$ **alors**
12: $\lambda_2 = \lambda_1$;
13: **Sinon**
14: $\lambda_1 = \lambda_2$;
15: **Fin Si**
   % $z^2 + bz + c = (z-\lambda_1)(z-\lambda_1)$ ou $(z-\lambda_2)(z-\lambda_2)$
16: $b = -\lambda_1 - \lambda_2$;
17: $c = \lambda_1 \times \lambda_2$;
18: $itr = itr + 1$;
19: **Sinon**
   % les valeurs propres sont des valeurs complexes.
20: $b = -trB$;
21: $c = detB$;
22: $itc = itc + 1$;
23: **Fin Si**
24: **Fin**

Après avoir déterminé les coefficients b_k et c_k de la fonction quadratique $p_k(A_{k-1})$ par l'algorithme 4.3 SRpoly, nous voulons maintenant calculer :

$$S_k R_k = p_k(A_{k-1}) = A_{k-1}^2 + b_k A_{k-1} + c_k I$$
$$A_k = S_k^J A_{k-1} S_k.$$

L'astuce est de se rendre compte que dans toutes les itérations, les matrices A_k sont

des matrices sous la forme J-Hessenberg à cause de l'unicité essentielle de la réduction sous la forme J-Hessenberg.

Dans la version explicite, le polynôme $p_k(A_{k-1})$ est calculé en $\mathcal{O}((2n)^3)$ opérations arithmétiques. Par conséquent, ce n'est pas une approche pratique. Nous pouvons contourner cette difficulté en utilisant le S-théorème implicite. En effet, au lieu de construire la matrice du polynôme $p_k(A_{k-1})$ qui est un polynôme du second degré d'une matrice de J-Hessenberg, nous ne calculons que sa première colonne. Cette dernière a seulement trois entrées non nulles.

4.4.3 Bulge chasing (chasser la bosse)

Par la suite, nous calculons la première transformation de Householder symplectique $T_0 = I - cvv^J$ de la décomposition SR appliquée implicitement à la première colonne de la matrice du polynôme de shift $p_k(A_{k-1})$. Par conséquent, T_0 transforme $p_k(A_{k-1})e_1$ en une colonne colinéaire au vecteur e_1, c'est-à-dire $T_0^J p_k(A_{k-1})e_1 = \alpha e_1$. Ensuite, nous calculons la matrice $T_0^J A_{k-1} T_0$, qui n'est plus de J-Hessenberg. La première colonne de la matrice S_k est la même que la première colonne de la matrice T_0. Les autres colonnes de la matrice S_k peuvent être déterminées par l'exigence selon laquelle la matrice A_k est sous la forme J-Hessenberg. Autrement dit, nous les calculons implicitement en appliquant l'algorithme de la réduction sous la forme J-Hessenberg à la matrice $T_0^J A_{k-1} T_0$, en profitant du fait que cette matrice a une structure spéciale. La complexité de cette réduction est $\mathcal{O}((2n)^2)$. Donc, nous appliquons une suite de transformations de Householder symplectiques $T_1, T_2, \ldots, T_{2n-2}$ à la matrice $T_0^J A_{k-1} T_0$, qui rétablissent la forme de J-Hessenberg. Nous notons par $\tilde{S}_k = T_0 T_1 \ldots T_{2n-2}$. Il s'ensuit que les matrices S_k et $\tilde{S}_k$ ont la même première colonne. D'après le S-théorème implicite les deux matrices S_k et $\tilde{S}_k$ sont essentiellement égaux.

De cette manière, nous réduisons la matrice A_{k-1} en une matrice de la forme J-Hessenberg $A_k = S_k^J A_{k-1} S_k$. Chaque étape de la réduction déplace un renflement vers le bas. Nous appelons cette procédure chasser la bosse ou bien la "bulge chasing".

Nous prenons un exemple d'une matrice A sous la forme J-Hessenberg de taille 10×10 ($n = 5$) pour montre le mouvement de la bosse durant la procédure et pour détailler les étapes des itérations de la bulge chasing.

Soit la matrice A de la forme :

$$A = \left(\begin{array}{ccccc|ccccc} x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & 0 & x & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

1. Afin de créer une bosse, nous multiplions cette matrice A à gauche par la matrice S_1^J

qui est une transformation de Householder symplectique de type un (osh_1), sachant que la première colonne de la matrice S_1 est proportionnel à la première colonne de la matrice du polynôme de shift $p_1(A) = A^2 + b_1 A + c_1 I$. Ainsi, nous pouvons écrire

$$S_1^J = I + c_1 \times v_1 \times v_1^{\mathrm{T}} \times J$$

et nous obtenons :

$$S_1^J A = \left(\begin{array}{ccccc|ccccc} \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ \oplus & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & x & x & x & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & x & x & x & x & 0 & x & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

Nous apercevons l'apparition d'une bosse à la position $(2,1)$ que nous la notons par le symbole "$\oplus$". Nous remarquons aussi que seulement la première, la deuxième et la $(n+1)^{\text{ième}}$ lignes qui sont modifiées. Nous notons par le symbole "$\overline{x}$" l'élément qui bouge.

Puis, nous multiplions la matrice $S_1^J A$ à droite par la transformation de Householder symplectique $S_1 = I - c_1 \times v_1 \times v_1^{\mathrm{T}} \times J$:

$$A_1 = S_1^J A S_1 = \left(\begin{array}{ccccc|ccccc} \overline{x} & x & x & x & x & \overline{x} & \overline{x} & x & x & x \\ \oplus & x & x & x & x & \overline{x} & \overline{x} & x & x & x \\ 0 & 0 & x & x & x & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline \overline{x} & x & x & x & x & \overline{x} & \overline{x} & x & x & x \\ \oplus & x & x & x & x & \oplus & \overline{x} & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

Par suit, deux autres bosses ont apparu aux positions $(n+1,1)$ et $(n+1,n+1)$. Cette bosse est de type 2×2. Dans cette étape seulement les colonnes 1, $(n+1)$ et $(n+2)$ qui bougent.

2. Après, nous commençons la chasse du bosse ou la "bulge chasing" pour remmener la matrice A_1 sous la forme J-Hessenberg. En outre, nous choisissons la transformation S_2^J afin d'éliminer les bosses de la première colonne aux positions $(2,1)$ et $(n+2,1)$ de la matrice A_1. Donc nous appliquons la transformation de Householder symplectique osh_2,

c'est-à-dire $S_2^J = I + c_2 \times v_2 \times v_2^{\mathrm{T}} \times J$, à gauche de la matrice A_1 :

$$S_2^J A_1 = \left(\begin{array}{ccccc|ccccc} \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & x & x & x & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & x & x & x & x & \oplus & x & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

De même ici seulement la première, la deuxième et la $(n+1)^{\text{ième}}$ lignes qui sont modifiées. La bosse à la position $(n+2, n+1)$ ne bouge pas.

Ensuite, nous multiplions la matrice $S_2^J A_1$ à droite par la matrice de la transformation de Householder symplectique du type deux (osh_2), c'est-à-dire la matrice $S_2 = I - c_2 \times v_2 \times v_2^{\mathrm{T}} \times J$:

$$A_2 = S_2^J A_1 S_2 = \left(\begin{array}{ccccc|ccccc} x & \overline{x} & x & x & x & \overline{x} & \overline{x} & x & x & x \\ 0 & \overline{x} & x & x & x & \overline{x} & \overline{x} & x & x & x \\ 0 & \oplus & x & x & x & \oplus & \overline{x} & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & \overline{x} & x & x & x & \overline{x} & \overline{x} & x & x & x \\ 0 & \overline{x} & x & x & x & \oplus & \overline{x} & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

Dans cette étape nous modifions la deuxième, la $(n+1)^{\text{ième}}$ et la $(n+2)^{\text{ième}}$ colonnes. La bosse aussi bouge et devient de type 2×2, localise aux positions $(3,2)$, $(3, n+1)$ et $(n+2, n+1)$.

3. Maintenant, nous voulons éliminer les bosses aux deux positions $(3, n+1)$ et $(n+2, n+1)$. Donc, nous multiplions la matrice A_2 à gauche par la transformation de Householder symplectique du type un (osh_1), $S_3^J = I + c_3 \times v_3 \times v_3^{\mathrm{T}} \times J$, nous obtenons :

$$S_3^J A_2 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & x & x & x & x & x & x \\ 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & \oplus & \overline{x} & \overline{x} & \overline{x} & 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & x & x & x & x & x & x & x & x & x \\ 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} & 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

Les lignes qui bougent sont la deuxième, la troisième et la $(n+2)^{\text{ième}}$ lignes. La bosse à la position $(3,2)$ ne disparut pas comme les deux autres bosses.

De même, nous multiplions la matrice $S_3^J A_2$ à droite par la transformation de Householder symplectique (osh_1), $S_3 = I - c_3 \times v_3 \times v_3^{\mathrm{T}} \times J$, ainsi nous aurons :

$$A_3 = S_3^J A_2 S_3 = \left(\begin{array}{ccccc|ccccc} x & \overline{x} & x & x & x & x & \overline{x} & \overline{x} & x & x \\ 0 & \overline{x} & x & x & x & x & \overline{x} & \overline{x} & x & x \\ 0 & \oplus & x & x & x & 0 & \overline{x} & \overline{x} & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & \overline{x} & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & \overline{x} & x & x & x & x & \overline{x} & \overline{x} & x & x \\ 0 & \overline{x} & x & x & x & 0 & \overline{x} & \overline{x} & x & x \\ 0 & \oplus & x & x & x & 0 & \oplus & \overline{x} & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

Les colonnes 2, $(n+1)$ et $(n+2)$ sont modifiées.

Remarque 4.4.2. *Nous remarquons que les bosses sont entrain de descendre vers le bas : les bosses sur la sous-diagonale se déplacer vers le bas d'un pas chaque deux étapes. L'autre bosse permute sa place entre les sous-diagonale de deux blocs* $(2,1)$ *et* $(1,2)$*, elle descendit d'un pas chaque deux étapes aussi.*

Maintenant, pour terminer la bulge chasing, nous allons répéter les étapes 2 et 3 jusqu'à la disparition totale des bosses.

4. Alors, nous multiplions à gauche et à droite par la transformation Householder symplectique du type deux (osh_2) et sa conjuguée, $S_4^J = I + c_4 \times v_4 \times v_4^{\mathrm{T}} \times J$ et $S_4 = I - c_4 \times v_4 \times v_4^{\mathrm{T}} \times J$ respectivement pour avoir la matrice A_4 :

$$S_4^J A_3 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & x & x & x & x & x & x \\ 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & \overline{x} & \overline{x} & \overline{x} & 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & 0 & x & x & x & x \\ 0 & 0 & \overline{x} & \overline{x} & \overline{x} & 0 & \oplus & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right),$$

$$A_4 = S_4^J A_3 S_4 = \left(\begin{array}{ccccc|ccccc} x & x & \overline{x} & x & x & x & \overline{x} & \overline{x} & x & x \\ 0 & x & \overline{x} & x & x & x & \overline{x} & \overline{x} & x & x \\ 0 & 0 & \overline{x} & x & x & 0 & \overline{x} & \overline{x} & x & x \\ 0 & 0 & \oplus & x & x & 0 & \oplus & \overline{x} & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & x & \overline{x} & x & x & x & \overline{x} & \overline{x} & x & x \\ 0 & x & \overline{x} & x & x & 0 & \overline{x} & \overline{x} & x & x \\ 0 & 0 & \overline{x} & x & x & 0 & \oplus & \overline{x} & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

D'abord, les bosses aux positions $(3,2)$, $(n+3,2)$ et $(n+3,n+2)$ se réduisent en une seule bosse à la position $(n+3,n+2)$. Après la multiplication à droite, la bosse se déplace vers les positions $(4,3)$, $(4,n+2)$ et $(n+3,n+2)$.

La multiplication à droite modifie les lignes 2, 3 et $(n+3)$, et la multiplication à gauche modifie les colonnes 3, $(n+2)$ et $(n+3)$.

5. Puis, nous multiplions à gauche et à droite par la transformation Householder symplectique du type un (osh_1) et sa conjuguée, $S_5^J = I + c_5 \times v_5 \times v_5^{\mathrm{T}} \times J$ et $S_5 = I - c_5 \times v_5 \times v_5^{\mathrm{T}} \times J$ respectivement pour avoir la matrice A_5 :

$$S_5^J A_4 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x & x & x & x \\ 0 & 0 & \overline{x} & \overline{x} & \overline{x} & 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & \oplus & \overline{x} & \overline{x} & 0 & 0 & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & 0 & x & x & x & x \\ 0 & 0 & \overline{x} & \overline{x} & \overline{x} & 0 & 0 & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right),$$

$$A_5 = S_5^J A_4 S_5 = \left(\begin{array}{ccccc|ccccc} x & x & \overline{x} & x & x & x & x & \overline{x} & \overline{x} & x \\ 0 & x & \overline{x} & x & x & x & x & \overline{x} & \overline{x} & x \\ 0 & 0 & \overline{x} & x & x & 0 & x & \overline{x} & \overline{x} & x \\ 0 & 0 & \oplus & x & x & 0 & 0 & \overline{x} & \overline{x} & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & x & \overline{x} & x & x & x & x & \overline{x} & \overline{x} & x \\ 0 & 0 & \overline{x} & x & x & 0 & x & \overline{x} & \overline{x} & x \\ 0 & 0 & \overline{x} & x & x & 0 & 0 & \overline{x} & \overline{x} & x \\ 0 & 0 & \oplus & x & x & 0 & 0 & \oplus & \overline{x} & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

De même, les bosses se réduisent en une seule bosse à la position $(4,3)$. Après la multiplication à droite, la bosse se déplace vers les positions $(4,3)$, $(n+4,3)$ et $(n+4,n+3)$.

La multiplication à gauche modifie les lignes 3, 4 et $(n+3)$, et la multiplication à droite modifie les colonnes 3, $(n+3)$ et $(n+4)$.

6. Après, nous multiplions à gauche et à droite par la transformation Householder symplectique du type deux (osh_2) et sa conjuguée, $S_6^J = I + c_6 \times v_6 \times v_6^{\mathrm{T}} \times J$ et $S_6 = I - c_6 \times v_6 \times v_6^{\mathrm{T}} \times J$ respectivement pour avoir la matrice A_6 :

$$S_6^J A_5 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x & x & x & x \\ 0 & 0 & \overline{x} & \overline{x} & \overline{x} & 0 & \overline{x} & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & \overline{x} & \overline{x} & 0 & 0 & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & x & x \\ \hline x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & 0 & x & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & \overline{x} & \overline{x} & 0 & 0 & \oplus & \overline{x} & \overline{x} \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right),$$

$$A_6 = S_6^J A_5 S_6 = \left(\begin{array}{ccccc|ccccc} x & x & x & \overline{x} & x & x & x & \overline{x} & \overline{x} & x \\ 0 & x & x & \overline{x} & x & x & x & \overline{x} & \overline{x} & x \\ 0 & 0 & x & \overline{x} & x & 0 & x & \overline{x} & \overline{x} & x \\ 0 & 0 & 0 & \overline{x} & x & 0 & 0 & \overline{x} & \overline{x} & x \\ 0 & 0 & 0 & \oplus & x & 0 & 0 & \oplus & \overline{x} & x \\ \hline x & x & x & \overline{x} & x & x & x & \overline{x} & \overline{x} & x \\ 0 & x & x & \overline{x} & x & 0 & x & \overline{x} & \overline{x} & x \\ 0 & 0 & x & \overline{x} & x & 0 & 0 & \overline{x} & \overline{x} & x \\ 0 & 0 & 0 & \overline{x} & x & 0 & 0 & \oplus & \overline{x} & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right).$$

Pareillement, les bosses se déplacent vers la positions $(n+4, n+3)$ puis vers les positions $(5,4)$, $(5, n+4)$ et $(n+4, n+3)$.

La multiplication à gauche modifie les lignes 3, 4 et $(n+4)$, et la multiplication à droite modifie les colonnes 4, $(n+3)$ et $(n+4)$.

7. Ensuite, nous multiplions à gauche et à droite par la transformation Householder symplectique du type un (osh_1) et sa conjuguée, $S_7^J = I + c_7 \times v_7 \times v_7^{\mathrm{T}} \times J$ et $S_7 = I - c_7 \times v_7 \times v_7^{\mathrm{T}} \times J$ respectivement pour avoir la matrice A_7 :

$$S_7^J A_6 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & 0 & x & x & x & x \\ 0 & 0 & 0 & \overline{x} & \overline{x} & 0 & 0 & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & \oplus & \overline{x} & 0 & 0 & 0 & \overline{x} & \overline{x} \\ \hline x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & 0 & x & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & \overline{x} & \overline{x} & 0 & 0 & 0 & \overline{x} & \overline{x} \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 & 0 & x \end{array}\right),$$

$$A_7 = S_7^J A_6 S_7 = \left(\begin{array}{ccccc|ccccc} x & x & x & \overline{x} & x & x & x & x & \overline{x} & \overline{x} \\ 0 & x & x & \overline{x} & x & x & x & x & \overline{x} & \overline{x} \\ 0 & 0 & x & \overline{x} & x & 0 & x & x & \overline{x} & \overline{x} \\ 0 & 0 & 0 & \overline{x} & x & 0 & 0 & x & \overline{x} & \overline{x} \\ 0 & 0 & 0 & \oplus & x & 0 & 0 & 0 & \overline{x} & \overline{x} \\ \hline x & x & x & \overline{x} & x & x & x & x & \overline{x} & \overline{x} \\ 0 & 0 & x & \overline{x} & x & 0 & x & x & \overline{x} & \overline{x} \\ 0 & 0 & x & \overline{x} & x & 0 & 0 & x & \overline{x} & \overline{x} \\ 0 & 0 & 0 & \overline{x} & x & 0 & 0 & 0 & \overline{x} & \overline{x} \\ 0 & 0 & 0 & \oplus & x & 0 & 0 & 0 & \oplus & \overline{x} \end{array}\right).$$

De même, les bosses se déplacent vers la position $(5,4)$ puis vers les positions $(5,4)$, $(n+5,4)$ et $(n+5,n+4)$.

La multiplication à gauche modifie les lignes 4, 5 et $(n+4)$, et la multiplication à droite modifie les colonnes 4, $(n+4)$ et $(n+5)$.

8. Puis, nous multiplions à gauche et à droite par la transformation Householder symplectique du type deux (osh_2) et sa conjuguée, $S_8^J = I + c_8 \times v_8 \times v_8^{\mathrm{T}} \times J$ et $S_8 = I - c_8 \times v_8 \times v_8^{\mathrm{T}} \times J$ respectivement pour avoir la matrice A_8 :

$$S_8^J A_7 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & 0 & x & x & x & x \\ 0 & 0 & 0 & \overline{x} & \overline{x} & 0 & 0 & \overline{x} & \overline{x} & \overline{x} \\ 0 & 0 & 0 & 0 & \overline{x} & 0 & 0 & 0 & \overline{x} & \overline{x} \\ \hline x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & 0 & x & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & \overline{x} & 0 & 0 & 0 & \oplus & \overline{x} \end{array}\right),$$

$$A_8 = S_8^J A_7 S_8 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & \overline{x} & x & x & x & \overline{x} & \overline{x} \\ 0 & x & x & x & \overline{x} & x & x & x & \overline{x} & \overline{x} \\ 0 & 0 & x & x & \overline{x} & 0 & x & x & \overline{x} & \overline{x} \\ 0 & 0 & 0 & x & \overline{x} & 0 & 0 & x & \overline{x} & \overline{x} \\ 0 & 0 & 0 & 0 & \overline{x} & 0 & 0 & 0 & \overline{x} & \overline{x} \\ \hline x & x & x & x & \overline{x} & x & x & x & \overline{x} & \overline{x} \\ 0 & x & x & x & \overline{x} & 0 & x & x & \overline{x} & \overline{x} \\ 0 & 0 & x & x & \overline{x} & 0 & 0 & x & \overline{x} & \overline{x} \\ 0 & 0 & 0 & x & \overline{x} & 0 & 0 & 0 & \overline{x} & \overline{x} \\ 0 & 0 & 0 & 0 & \overline{x} & 0 & 0 & 0 & \oplus & \overline{x} \end{array}\right).$$

Après la multiplication à droite et à gauche, la bosse reste à la même position $(n+5,n+4)$.

La multiplication à gauche modifie les lignes 4, 5 et $(n+5)$, et la multiplication à droite modifie les colonnes 5, $(n+4)$ et $(n+5)$.

9. Finalement, nous multiplions à gauche et à droite par la transformation Householder symplectique du type un (osh_1) et sa conjuguée, $S_9^J = I + c_9 \times v_9 \times v_9^{\mathrm{T}} \times J$ et $S_9 = I - c_9 \times v_9 \times v_9^{\mathrm{T}} \times J$ respectivement pour avoir la matrice A_9 :

$$S_9^J A_8 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & x & x & x & x & x \\ 0 & 0 & x & x & x & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & \overline{x} & 0 & 0 & 0 & \overline{x} & \overline{x} \\ \hline x & x & x & x & x & x & x & x & x & x \\ 0 & x & x & x & x & 0 & x & x & x & x \\ 0 & 0 & x & x & x & 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & \overline{x} & 0 & 0 & 0 & 0 & \overline{x} \end{array}\right),$$

$$A_9 = S_9^J A_8 S_9 = \left(\begin{array}{ccccc|ccccc} x & x & x & x & \overline{x} & x & x & x & x & \overline{x} \\ 0 & x & x & x & \overline{x} & x & x & x & x & \overline{x} \\ 0 & 0 & x & x & \overline{x} & 0 & x & x & x & \overline{x} \\ 0 & 0 & 0 & x & \overline{x} & 0 & 0 & x & x & \overline{x} \\ 0 & 0 & 0 & 0 & \overline{x} & 0 & 0 & 0 & x & \overline{x} \\ \hline x & x & x & x & \overline{x} & x & x & x & x & \overline{x} \\ 0 & x & x & x & \overline{x} & 0 & x & x & x & \overline{x} \\ 0 & 0 & x & x & \overline{x} & 0 & 0 & x & x & \overline{x} \\ 0 & 0 & 0 & x & \overline{x} & 0 & 0 & 0 & x & \overline{x} \\ 0 & 0 & 0 & 0 & \overline{x} & 0 & 0 & 0 & 0 & \overline{x} \end{array}\right).$$

Enfin, la bosse est disparue de la matrice $S^J AS$ et on récupère la forme de J-Hessenberg; sachant que $S = S_1 S_2 \ldots S_{2n-1}$ et $S^J = S_{2n-1}^J S_{2n-2}^J \ldots S_1^J$.

La multiplication à gauche modifie les lignes 5 et $(n+5)$, et la multiplication à droite modifie les colonnes 5 et $(n+5)$. □

L'algorithme qui fait cette procédure de la chasse du bosse ou bien la "bulge chasing" est le suivant :

Algorithme 4.4 Algorithme *SRstep*

L'algorithme SRstep est une étape de l'algorithme *SR* implicite avec double shift.

ENTRÉE(S) : La matrice $H \in \mathbb{R}^{2n \times 2n}$ sous la forme de J-Hessenberg.

SORTIE(S) : La matrice $H \in \mathbb{R}^{2n \times 2n}$ sous la forme de J-Hessenberg après la bulge chasing.

1: Fonction $[H] = SRstep(H)$
2: $den =$ longueur (H);
3: $n = den/2$;
4: $J_4 = [zeros(2), eye(2); -eye(2), zeros(2)]$;
% Nous calculons les coefficients b et c du polynôme de double shift $H^2 + bH + cI$ et sa première colonne $C1$.

La suite de l'algorithme de $SRstep$

5: $[b,c] = SRpoly(H)$;
6: $C1 = H([1:2, n+1:n+2],[1,n+1]) \times H([1,n+1],1)$;
7: $C1([1,3],1) = C1([1,3],1) + b \times H([1,n+1],1)$;
8: $C1(1) = C1(1) + c$;
% Nous calculons la transformation de Householder symplectique (osh_1) associée à la première étape de la décomposition SR sur la colonne $C1$.
9: $[c_1, v_1] = osh1(C1)$;
% Nous multiplions à gauche puis à droite la matrice H par la transformation S_k et sa conjuguée S_k^J.
10: $ro = [1:2, n+1:n+2]$;
11: $H(ro,:) = H(ro,:) + c_1 \times v_1 \times (v_1^{\mathrm{T}} \times J_4 \times H(ro,:))$;
12: $H(:,ro) = H(:,ro) - (H(:,ro) \times (c_1 \times v_1)) \times v_1^{\mathrm{T}} \times J_4$;
% Chasser la bosse ou bien la "bulge chasing".
13: **Pour** $j = 1,\ldots,n-1$ **faire**
14: $k = min(j+1,n)$;
15: $J = [zeros(k-j+1), eye(k-j+1); -eye(k-j+1), zeros(k-j+1)]$;
16: $ro = [j:k, n+j:n+k]$;
17: $[c, v_1] = osh2(H(ro,j))$;
18: $H(ro,:) = H(ro,:) + c \times v_1 \times (v_1^{\mathrm{T}} \times J \times H(ro,:))$;
19: $H(:,ro) = H(:,ro) - (H(:,ro) \times (c \times v_1)) \times v_1^{\mathrm{T}} \times J$;
20: $k = min(j+2,n)$;
21: $J = [zeros(k-j), eye(k-j); -eye(k-j), zeros(k-j)]$;
22: $ro = [j+1:k, n+j+1:n+k]$;
23: $[c, v_2] = osh1(H(ro, n+j))$;
24: $H(ro,:) = H(ro,:) + c \times v_2 \times (v_2^{\mathrm{T}} \times J \times H(ro,:))$;
25: $H(:,ro) = H(:,ro) - (H(:,ro) \times (c \times v_2)) \times v_2^{\mathrm{T}} \times J$;
26: **Fin Pour**
27: **Fin**

4.4.4 La déflation

Une séquence d'étapes SR implicite avec double shift, nous donnera généralement une convergence plus rapide. Donc, si les décalages choisis sont de bonnes approximations de valeurs propres, alors nous nous attendons à une déflation à la fin de l'étape de SR.

En effet, l'analyse précédente de la stratégie du décalage simple ou double montre que les coefficients de la sous-diagonale du blocs $(1,2)$, qui est une matrice de Hessenberg, convergent vers 0 très rapidement. Dans la $k^{\text{ième}}$ itération, quand le coefficient $H(k,2k-1)$ est suffisamment petit, nous réduisons la taille en supprimant la $k^{\text{ième}}$ et la $2k^{\text{ième}}$ lignes et la $k^{\text{ième}}$ et la $2k^{\text{ième}}$ colonnes de la matrice H_k courante. Nous continuons l'algorithme SR avec la matrice de J-Hessenberg réduite $H_k([1:k-1,k+1:2k-1],[1:k-1,k+1:2k-1])$ et les valeurs propres de la matrice $H_k([k-1,2k-1],[k-1,2k-1])$, de taille (2×2), comme des décalages pour l'itération suivante. Et ainsi de suite, jusqu'à obtenir à la fin une matrice

réduite de taille (2×2). Ce processus de réduction de la dimension du problème, nous l'appelons par déflation.

En d'autres termes, soit H une matrice sous la forme de J-Hessenberg :

$$H = \left(\begin{array}{ccccc|ccccc}
a_{1,1} & a_{1,2} & \cdots & \cdots & a_{1,n} & b_{1,1} & b_{1,2} & \cdots & \cdots & b_{1,n} \\
 & a_{2,2} & \ddots & \ddots & \vdots & b_{2,1} & b_{2,2} & \ddots & \ddots & \vdots \\
 & & \ddots & \ddots & \vdots & & \ddots & \ddots & \ddots & \vdots \\
 & & & a_{n-1,n-1} & a_{n-1,n} & & & \ddots & b_{n-1,n-1} & b_{n-1,n} \\
 & & & & a_{n,n} & & & & b_{n,n-1} & b_{n,n} \\
\hline
c_{1,1} & c_{1,2} & \cdots & \cdots & c_{1,n} & d_{1,1} & d_{1,2} & \cdots & \cdots & d_{1,n} \\
 & c_{2,2} & \ddots & \ddots & \vdots & & d_{2,2} & \ddots & \ddots & \vdots \\
 & & \ddots & \ddots & \vdots & & & \ddots & \ddots & \vdots \\
 & & & c_{n-1,n-1} & c_{n-1,n} & & & & d_{n-1,n-1} & d_{n-1,n} \\
 & & & & c_{n,n} & & & & & d_{n,n}
\end{array}\right).$$

Au début de l'étape $k^{\text{ième}}$ de l'itération, nous vérifions si certains élément $b_{k,k-1}$ est négligeable. C'est-à-dire si

$$|b_{k,k-1}| < tol \times (|a_{k,k}| + |d_{k-1,k-1}|), \tag{4.7}$$

avec $tol = \|H\|_F \times \sqrt{eps}$ où eps est la précision relative de la machine. Dans le cas où nous avons la condition (4.7), alors nous pouvons diviser notre problème en deux plus petits sous problèmes pour les matrices de J-Hessenberg :

$$\left(\begin{array}{ccccc|ccccc}
a_{1,1} & a_{1,2} & \cdots & \cdots & a_{1,k} & b_{1,1} & b_{1,2} & \cdots & \cdots & b_{1,k} \\
 & a_{2,2} & \ddots & \ddots & \vdots & b_{2,1} & b_{2,2} & \ddots & \ddots & \vdots \\
 & & \ddots & \ddots & \vdots & & \ddots & \ddots & \ddots & \vdots \\
 & & & a_{k-1,k-1} & a_{k-1,k} & & & \ddots & b_{k-1,k-1} & b_{k-1,k} \\
 & & & & a_{k,k} & & & & b_{k,k-1} & b_{k,k} \\
\hline
c_{1,1} & c_{1,2} & \cdots & \cdots & c_{1,k} & d_{1,1} & d_{1,2} & \cdots & \cdots & d_{1,k} \\
 & c_{2,2} & \ddots & \ddots & \vdots & & d_{2,2} & \ddots & \ddots & \vdots \\
 & & \ddots & \ddots & \vdots & & & \ddots & \ddots & \vdots \\
 & & & c_{k-1,k-1} & c_{k-1,k} & & & & d_{k-1,k-1} & d_{k-1,k} \\
 & & & & c_{k,k} & & & & & d_{k,k}
\end{array}\right)$$

et

$$\left(\begin{array}{ccccc|ccccc}
a_{k+1,k+1} & a_{k+1,k+2} & \cdots & \cdots & a_{k+1,n} & b_{k+1,k+1} & b_{k+1,k+2} & \cdots & \cdots & b_{k+1,n} \\
 & a_{k+2,k+2} & \ddots & \ddots & \vdots & b_{k+2,k+1} & b_{k+2,k+2} & \ddots & \ddots & \vdots \\
 & & \ddots & \ddots & \vdots & & \ddots & \ddots & \ddots & \vdots \\
 & & & a_{n-1,n-1} & a_{n-1,n} & & & \ddots & b_{n-1,n-1} & b_{n-1,n} \\
 & & & & a_{n,n} & & & & b_{n,n-1} & b_{n,n} \\
\hline
c_{k+1,k+1} & c_{k+1,k+2} & \cdots & \cdots & c_{k+1,n} & d_{k+1,k+1} & d_{k+1,k+2} & \cdots & \cdots & d_{k+1,n} \\
 & c_{k+2,k+2} & \ddots & \ddots & \vdots & & d_{k+2,k+2} & \ddots & \ddots & \vdots \\
 & & \ddots & \ddots & \vdots & & & \ddots & \ddots & \vdots \\
 & & & c_{n-1,n-1} & c_{n-1,n} & & & & d_{n-1,n-1} & d_{n-1,n} \\
 & & & & c_{n,n} & & & & & d_{n,n}
\end{array}\right)$$

et nous procédons à travailler sur chaque bloc séparément.

Nous faisons la déflation, quand le coefficient $b_{j,j-1}$ s'annule, vu que $PHP^{\mathrm{T}} =$

$$\left(\begin{array}{cc|cc|cc|cc|cc|cc}
a_{1,1} & b_{1,1} & a_{1,2} & b_{1,2} & a_{1,3} & b_{1,3} & \cdots & \cdots & \cdots & b_{1,n-1} & a_{1,n} & b_{1,n} \\
c_{1,1} & d_{1,1} & c_{1,2} & d_{1,2} & c_{1,3} & d_{1,3} & \ddots & \ddots & \ddots & \ddots & c_{1,n} & d_{1,n} \\
\hline
0 & b_{2,1} & a_{2,2} & b_{2,2} & a_{2,3} & b_{2,3} & a_{2,3} & \ddots & \ddots & \ddots & \ddots & b_{2,n} \\
0 & 0 & c_{2,2} & d_{2,2} & c_{2,3} & d_{2,3} & c_{2,4} & d_{2,4} & \ddots & \ddots & \ddots & \vdots \\
\hline
0 & 0 & 0 & b_{3,2} & a_{3,3} & b_{3,3} & a_{3,4} & b_{3,4} & a_{3,5} & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\hline
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
\hline
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & b_{n-1,n-2} & a_{n-1,n-1} & b_{n-1,n-1} & a_{n-1,n} & b_{n-1,n} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & c_{n-1,n-1} & d_{n-1,n-1} & c_{n-1,n} & d_{n-1,n} \\
\hline
0 & 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 & b_{n,n-1} & a_{n,n} & b_{n,n} \\
0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & c_{n,n} & d_{n,n}
\end{array}\right),$$

avec la matrice P est la permutation $P = (e_1, e_3, \ldots, e_{2n-1}, e_2, e_4, \ldots, e_{2n}) \in \mathbb{R}^{2n\times 2n}$. Cette permutation transforme la matrice H de J-Hessenberg en une matrice PHP^{T} de Hessenberg. Comme la dimension du problème principale doit être pair, alors quand nous déflatons la dimension des sous problèmes doit être aussi pair. Ainsi, contrairement au cas Euclidien pour calculer des valeurs et/ou des vecteurs propres d'une matrice via l'algorithme QR, nous ne pouvons pas faire la déflation lorsque le coefficient $c_{k,k}$ s'annule dans le cas symplectique et via l'algorithme SR. Donc, pour préserver la structure de la matrice, nous effectuons la déflation en supprimant un nombre pair des lignes et colonnes. De plus, à l'étape $k^{\text{ième}}$ de l'itération, nous choisissons les décalage parmi les valeurs propres du dernier bock 2×2 de la matrice H, c'est-à-dire du block $\begin{pmatrix} a_{k,k} & b_{k,k} \\ c_{k,k} & d_{k,k} \end{pmatrix}$. Par conséquent, les coefficients de la sous-diagonale du blocs $(1,2)$ de la matrice H commencent à s'annuler du bas vers le haut.

En fin, le programme principale de l'algorithme SR sous la forme implicite avec double shift est le suivant :

Algorithme 4.5 Algorithme $SRimplicite$

L'algorithme SRimplicite est l'algorithme SR sous la forme implicite avec double shift qui calcule les valeurs propres d'une matrice $A \in \mathbb{R}^{2n\times 2n}$.

ENTRÉE(S) : La matrice $A \in \mathbb{R}^{2n\times 2n}$.

SORTIE(S) : Le vecteur $T \in \mathbb{R}^{2n}$ des valeurs propres de la matrice A, le nombre d'itérations it avec shift simple, le nombre d'itérations itd avec shift double et le temps d'exécution t de l'algorithme.

Fonction $[T, it, itd, t] = SRimplicite(A)$
global $itr\ itc$;
$den =$ longueur (A);
$T = zeros(den, 1)$;
$tol = \|A\|_F \times \sqrt{eps}$;
$n = den/2$;
$k = n$;
$h = n$;
$l = n$;
$i = 1$;
$j = 1$;
% Nous réduisons la matrice A sous la forme J-Hessenberg tel que $H = S^J AS$.
$[S, H] = JHOSH(A)$;
$itr = 0$;
$itc = 0$;
tic;
% La déflation.
Tant que $(k \geqslant 2)$ et $(k > i)$ **faire**
  Si $(|H(k, n+h-1)| < tol \times (|H(k,k)| + |H(n+h-1, n+h-1)|))$ **alors**
    $T(2\times l-1 : 2\times l) = eig2x2([H(k,k)\,H(k,n+h); H(n+h,k)\,H(n+h,n+h)])$;
    $l = l-1$;
    $k = k-1$;
    $h = h-1$;
  Sinon Si $(|H(k-1, n+h-2)| < tol \times (|H(k-1,k-1)| + |H(n+h-2, n+h-2)|))$ **alors**
    $T(2\times l-3 : 2\times l) = eig4x4(H([k-1 : k, n+h-1 : n+h], [k-1 : k, n+h-1 : n+h]))$;
    $l = l-2$;
    $k = k-2$;
    $h = h-2$;
  Sinon Si $(|H(i+1, n+j)| < tol \times (|H(i+1,i+1)| + |H(n+j, n+j)|))$ **alors**
    $T(2\times l-1 : 2\times l) = eig2x2([H(i,i)\,H(i,n+j); H(n+j,i)\,H(n+j,n+j)])$;
    $l = l-1$;
    $i = i+1$;
    $j = j+1$;
  Sinon
    $H([i : k, n+j : n+h], [i : k, n+j : n+h]) =$
      $SRstep(H([i : k, n+j : n+h], [i : k, n+j : n+h]))$;
  Fin Si

La suite de l'algorithme $SRimplicite$

Fin Tant que
Si $((k-i+n+h-n+j+2)==2)$ **alors**
    $T(2\times l-1:2\times l)=eig2x2(H([i:k,n+j:n+h],[i:k,n+j:n+h]));$
Fin Si
$t=toc;$
$it=itr;$
$itd=itc;$
Fin

Nous notons que la fonction $eig2x2$ est un programme qui calcule les valeurs propres de matrice de taille 2×2 :

Algorithme 4.6 Algorithme $eig2x2$

L'algorithme $eig2x2$ est une fonction qui calcule les valeurs propres d'une matrice de taille 2×2.

ENTRÉE(S) : Une matrice $A\in\mathbb{R}^{2\times 2}$.
SORTIE(S) : Un vecteur $v\in\mathbb{R}^2$ qui contient les valeurs propres de la matrice A.

Fonction $[v]=eig2x2(A)$
    % La trace de la matrice A.
$trA=A(2,2)+A(1,1);$
    % Le déterminant de la matrice A.
$detA=A(2,2)\times A(1,1)-A(1,2)\times A(2,1);$
$sqrtd=\sqrt{trA^2-4\times detA};$
    % La première valeur propre.
$v(1)=(trA+sqrtd)/2;$
    % La deuxième valeur propre.
$v(2)=(trA-sqrtd)/2;$
$v=v(:);$
Fin

4.5 Cas d'une matrice Hamiltonienne

Dans cet paragraphe, nous développerons une implémentation de l'algorithme SR pour les matrices Hamiltoniennes. La structure Hamiltonienne est forcée à chaque étape et les erreurs d'arrondis ne peuvent pas détruire la structure Hamiltonienne. Notre but sera de dériver une description d'une étape SR implicite. Nous rappelons qu'une matrice réelle $H\in\mathbb{R}^{2n\times 2n}$ est dite matrice Hamiltonienne si $H^J=-H$, c'est-à-dire $HJ=(HJ)^{\mathrm{T}}$. En effet, une matrice Hamiltonienne a la forme suivante :

$$H=\begin{pmatrix}A & G\\ Q & -A^{\mathrm{T}}\end{pmatrix}=\begin{pmatrix}\diagdown & \diagdown\\ \diagdown & \diagdown\end{pmatrix}.$$

où les blocs $A,\ G,\ Q \in \mathbb{R}^{n\times n}$, sachant que les matrices G et Q sont deux matrices symétriques $G = G^{\mathrm{T}}$, $Q = Q^{\mathrm{T}}$.

Un certain nombre d'applications de la théorie du contrôle et des domaines connexes conduisent aux problèmes de calcul des valeurs propres d'une matrice Hamiltonienne. Nous citons par exemple : le rayon de stabilité et le calcul de la norme H_∞, les problèmes de contrôle linéaire quadratique optimal et la solution des équations algébriques continues de Riccati, le contrôle H_∞, réduction du modèle avec conservation de la structure, les problèmes de calcul des valeurs propres quadratiques, calcul des pseudo-spectres, etc.

Les transformations de similarité symplectique préservent la structure Hamiltonienne

$$(S^{-1}HS)^J = S^{-1}HJS^{-\mathrm{T}} = S^{-1}J^{\mathrm{T}}H^{\mathrm{T}}S^{-\mathrm{T}} = [(S^{-1}HS)J]^{\mathrm{T}}.$$

Parmi les propriétés les plus remarquables de la matrice Hamiltonienne est que ses valeurs propres se produisent toujours en paires $\{\lambda, -\lambda\}$ si λ est réel ou purement imaginaire ($\lambda \in \mathbb{R}$ ou $\lambda \in i\mathbb{R}$), ou en quadruples $\{\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}\}$ si $\lambda \in \mathbb{C} \setminus (\mathbb{R} \cup i\mathbb{R})$. Par conséquent, le spectre de toute matrice Hamiltonienne est symétrique par rapport à l'axe des réels et l'axe des imaginaires purs. Les méthodes numériques qui tiennent compte de cette structure sont capables de préserver cette symétrie des valeurs propres malgré la présence d'erreurs d'arrondis. En plus que la préservation de telles symétries des valeurs propres, il existe d'autres avantages de l'utilisation des algorithmes de préservation de la structure à la place des algorithmes généraux, comme la réduction du temps de calcul et une meilleure précision pour les valeurs et/ou les vecteurs propres.

Par conséquent, afin de développer des méthodes numériques rapides et efficaces pour le problème de calcul des valeurs propres d'une matrice Hamiltonienne, nous devons utiliser la structure mathématique du problème.

Les valeurs propres et les sous-espaces invariables des matrices Hamiltoniennes H peuvent être calculées par l'algorithme QR. Mais cette méthode ne peut pas profiter de la structure Hamiltonienne de la matrice H. En fait, elle traitera la matrice H comme n'importe quelle matrice arbitraire de taille $2n \times 2n$. Elle va imposer des erreurs non structurées inévitables sur les valeurs propres calculées, de sorte que chaque valeur propre d'une paire de valeurs propres ou d'une quadruple peut être modifiée d'une manière légèrement différente. Par suite, les valeurs propres calculées ne seront généralement pas en paire $\{\lambda, -\lambda\}$ ou en quadruple $\{\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}\}$, bien que les valeurs propres exactes aient cette propriété. Ainsi, les exactes informations sur le sous-espace invariant pourraient alors être perdues.

Donc, pour préserver la structure Hamiltonienne de la matrice H, nous devrions utiliser des matrices symplectiques pour les transformations de similarité au lieu des matrices unitaires usuelles de l'algorithme QR.

Pour une matrice Hamiltonienne, l'algorithme SR est presque comme celui décrit dans la section précédente.

Algorithme 4.7 Algorithme *SR* pour la matrice Hamiltonienne avec réduction sous la forme J-Hessenberg

L'algorithme *SR* pour une matrice de taille $2n \times 2n$ Hamiltonienne A avec réduction sous la forme J-Hessenberg.

ENTRÉE(S) : La matrice $A = [a_1, a_2, \ldots, a_{2n}] \in \mathbb{R}^{2n \times 2n}$.

SORTIE(S) : Deux matrices $S = [s_1, s_2, \ldots, s_{2n}] \in \mathbb{R}^{2n \times 2n}$ et $H \in \mathbb{R}^{2n \times 2n}$ respectivement une matrice symplectique et une matrice J-tridiagonale tel que $H = S^{-1}AS = S^JAS$.

1: Fonction $[S, H] = SR_Algorithme(A)$

2: Réduire la matrice A Hamiltonienne sous la forme d'une matrice de J-Hessenberg H_0 tel que :

$$H_0 = S_0^{-1}AS_0 = S_0^JAS_0 = \begin{pmatrix} \diagdown & \diagdown\!\diagdown\!\diagdown \\ \diagdown & \diagdown \end{pmatrix}.$$

Sachant que la matrice S_0 est une matrice symplectique et la matrice H_0 est matrice J-tridiagonale, c'est-à-dire une matrice Hamiltonienne et J-Hessenberg.

3: Soit $S = S_0$

4: **Pour** $k = 1, 2, \ldots$ **faire**

5: Choisir le polynôme de shift p_k

6: Calculer la matrice symplectique S_k de la décomposition *SR* du polynôme

$$p_k(H_{k-1}) = S_kR_k.$$

7: Calculer la matrice J-tridiagonale

$$H_k = S_k^JH_{k-1}S_k = \begin{pmatrix} \diagdown & \diagdown\!\diagdown\!\diagdown \\ \diagdown & \diagdown \end{pmatrix}.$$

8: Calculer $S = SS_k$

9: **Fin Pour**

10: **Fin**

Nous Rappelons que l'algorithme *SR* est un algorithme itératif, qui effectue une décomposition *SR* à chaque itération. Après la réduction de la matrice A sous la forme J-Hessenberg tel que :

$$H_0 = S_0^JAS_0,$$

si la matrice H_{k-1} est la matrice courante de l'itération $(k-1)^{\text{ième}}$, alors à l'étape $k^{\text{ième}}$ nous choisissons un polynôme de shift p_k comme dans le cas générale du choix du shift. Puis, nous calculons la matrice symplectique S_k de la décomposition *SR* tel que :

$$p_k(H_{k-1}) = S_kR_k.$$

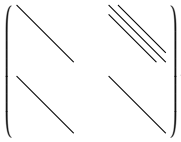
Ensuite, nous utilisons la matrice symplectique S_k pour effectuer la transformation de similarité sur la matrice H_{k-1} afin d'avoir la matrice H_k de l'itération suivante de telle

sorte que :

$$H_k = S_k^J H_{k-1} S_k.$$

Du coup, l'algorithme SR préserve la structure Hamiltonienne puisque nous avons utilisé seulement des transformations de similarité symplectiques.

Nous notons que toute matrice Hamiltonienne peut être réduite sous la forme Hamiltonienne J-Hessenberg, c'est-à-dire sous la forme J-tridiagonale



où chaque bloc est une matrice de taille $n \times n$ et à cause de la structure Hamiltonienne, les blocs H_{11} et H_{22} sont identiques ($H_{11} = -H_{22}$), tandis que le bloc H_{12} est une matrice symétrique. Par conséquent, toute matrice Hamiltonienne peut être représentée par $4n-1$ paramètres.

Par suite, à la $k^{\text{ième}}$ étape de l'algorithme SR la matrice H sera sous la forme suivante :

$$H = \left(\begin{array}{cccc|cccc} a_1 & & & & c_1 & b_1 & & \\ & a_2 & & & b_1 & \ddots & \ddots & \\ & & \ddots & & & \ddots & \ddots & b_{n-1} \\ & & & a_n & & & b_{n-1} & c_n \\ \hline q_1 & & & & -a_1 & & & \\ & q_2 & & & & -a_2 & & \\ & & \ddots & & & & \ddots & \\ & & & q_n & & & & -a_n \end{array}\right).$$

Dans le cas où λ est une valeur propre réelle de la matrice Hamiltonienne H, alors $-\lambda$ est aussi une valeur propre de la matrice A. Sinon, si λ n'est pas une valeur réelle, alors $-\lambda, \bar{\lambda}$, et $-\bar{\lambda}$ sont également des valeurs propres de la matrice A.

Ainsi, le polynôme caractéristique $p(\lambda)$ de la matrice Hamiltonienne J-tridiagonale de taille 4×4 admet une forme quadratique et ses quatre valeurs propres sont faciles à calculer.

Par conséquent, nous utilisons le polynôme

$$p_k(x) = (x-\lambda_1)(x-\lambda_2)(x-\lambda_3)(x-\lambda_4),$$

comme polynôme de shift où λ_1, λ_2, λ_3 et λ_4 sont les valeurs propres de la sous-matrice

$$B = \left(\begin{array}{cc|cc} a_{n-1} & 0 & c_{n-1} & b_{n-1} \\ 0 & a_n & b_{n-1} & c_n \\ \hline q_{n-1} & 0 & -a_{n-1} & 0 \\ 0 & q_n & 0 & -a_n \end{array}\right)$$

de la matrice H.

Nous nous attendons à une convergence lorsque seulement chaque deuxième sous-diagonale de la matrice tridiagonale H_{12} s'annule, c'est-à-dire lorsque b_{k-2} s'annule pour $k = n : -2 : 1$. Autrement dit, lorsque les sous-matrices (B) de taille 4×4 de la matrice H nous donnent des valeurs propres en quadruple $\{\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}\}$ avec $\lambda \in \mathbb{C} \setminus (\mathbb{R} \cup i\mathbb{R})$. De plus, nous nous attendons aussi que les autres b_k et q_k peuvent parfois s'annuler dans le cas où par exemple certaines valeurs propres de la matrice H sont réelles, à savoir les valeurs propres seront les paires réelles $\{a_k, -a_k\}$. Mais nous ne pouvons pas nous attendre à ce que les matrices H_k convergent vers la forme J-triangulaire supérieure, ce qui signifie ici que les éléments b_k et q_k tendent tous vers zéro.

4.5.1 L'algorithme SR implicite pour une matrice Hamiltonienne

Pareillement, comme dans le cas général, nous pouvons implémenter l'algorithme SR implicitement pour une matrice Hamiltonienne. En fait, nous commençons par la détermination de la première colonne du polynôme de shift $p_1(H_0)$. Ensuite, nous cherchons la première transformation symplectique S_1, qui rend la première colonne de $p_1(H_0)$ colinéaire au vecteur e_1, afin d'introduire des bulges $(\oplus)$ dans la matrice H_0 de J-Hessenberg de telle sorte que :

$$S_1^{-1} p_1(H_0) e_1 = S_1^J p_1(H_0) e_1 = \alpha e_1 \text{ avec } \alpha \in \mathbb{R}.$$

L'application de cette première transformation symplectique à la matrice de J-Hessenberg H_0 donne une matrice Hamiltonienne $S_1^J H_0 S_1$ avec une forme presque de J-Hessenberg ayant des bulges tel que :

$$S_1^J H_0 S_1 = \left(\begin{array}{cccccc|cccccc}
x & \oplus & \oplus & & & & x & x & \oplus & \oplus & & \\
\oplus & x & \oplus & & & & x & x & x & \oplus & & \\
\oplus & \oplus & x & & & & \oplus & x & x & x & & \\
 & & & x & & & \oplus & \oplus & x & x & \ddots & \\
 & & & & \ddots & & & & & \ddots & \ddots & \ddots \\
 & & & & & \ddots & & & & & \ddots & \ddots \\
 & & & & & & & & & & & x \\
\hline
x & \oplus & \oplus & & & & x & & & & & \\
\oplus & x & \oplus & & & & & x & & & & \\
\oplus & \oplus & x & & & & & \oplus & x & \oplus & \oplus & \\
 & & & x & & & & \oplus & \oplus & x & \oplus & \\
 & & & & \ddots & & & & & & \ddots & \\
 & & & & & \ddots & & & & & & \ddots \\
 & & & & & & & & & & & x
\end{array}\right).$$

Ainsi, cette matrice doit être réduite à nouveau sous la forme J-tridiagonale, sachant que le nombre d'opérations arithmétiques pour une telle étape est d'ordre $\mathcal{O}(n)$ seulement sans considérer la mise à jour de la matrice S.

Les transformations restantes exécutent un Bulge chasing (chasser la bosse) pour restaurer la forme de J-Hessenberg. C'est-à-dire qu'une deuxième matrice S_2 symplectique

est déterminée de telle sorte que la matrice $H_0 = S_2^J S_1^J H_0 S_1 S_2$ soit une matrice de la forme J-Hessenberg à nouveau.

4.5.1.1 Le polynôme du shift

En vertu de la structure spéciale Hamiltonienne, le polynôme du shift sera choisie soit comme suit :

4.5.1.1.1 Cas du simple shift

Dans une étape de l'algorithme *SR* implicite avec un simple shift pour une matrice réelle H Hamiltonienne, le polynôme du shift est de la forme suivante :

$$p_k(H) = H - \mu I,$$

avec $\mu \in \mathbb{R}$. La première colonne de $p_k(H)$ est sous la forme

$$p_k(H)e_1 = (H - \mu I)e_1 = \begin{pmatrix} a_1 - \mu \\ 0 \\ \vdots \\ 0 \\ q_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = (a_1 - \mu)e_1 + q_1 e_{n+1}.$$

Par conséquent, la première étape de l'algorithme *SR* implicite introduit des bulges (bosses) dans la matrice H_0 tel que :

$$S_1^J H_0 S_1 = \left(\begin{array}{cccc|cccc} a_1 & & & & c_1 & b_1 & & \\ \oplus & a_2 & & & b_1 & \ddots & \ddots & \\ & \ddots & & & & \ddots & \ddots & b_{n-1} \\ & & & a_n & & & b_{n-1} & c_n \\ \hline q_1 & & & & -a_1 & \oplus & & \\ & q_2 & & & & -a_2 & & \\ & & \ddots & & & & \ddots & \\ & & & q_n & & & & -a_n \end{array}\right).$$

4.5.1.1.2 Cas du double shift

Dans une étape de l'algorithme *SR* implicite avec un double shift pour une matrice réelle H Hamiltonienne, le polynôme du shift est de la forme suivante :

$$p_k(H) = (H - \sigma_k I)(H - \sigma_{k+1} I) = (H - \mu I)(H + \mu I) = H^2 - \mu^2 I,$$

avec $\mu \in \mathbb{R}$ ou $\mu \in i\mathbb{R}$. La première colonne de $p_k(H)$ est sous la forme

$$p_k(H)e_1 = (H^2 - \mu^2 I)e_1 = \begin{pmatrix} a_1^2 + q_1c_1 - \mu^2 \\ q_1b_1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = (a_1^2 + q_1c_1 - \mu^2)e_1 + q_1b_1e_2.$$

Dans ce cas, comme dans le cas général, nous calculons les valeurs propres $\{\lambda, -\lambda\}$ avec $\mu \in \mathbb{R}$ ou bien $\mu \in i\mathbb{R}$ en utilisant les deux valeurs propres la sous-matrice

$$B = \begin{pmatrix} a_n & c_n \\ q_n & -a_n \end{pmatrix}.$$

En réalité, nous n'avons pas besoin de calculer les valeurs propres de la matrice B. En effet, comme nous savons que $\mu^2 = -\det(B) = a_n^2 + q_nc_n$, alors

$$p_k(H)e_1 = (a_1^2 + q_1c_1 - a_n^2 - q_nc_n)e_1 + q_1b_1e_2.$$

Par conséquent, la première étape de l'algorithme *SR* implicite introduit des bulges (bosses) dans la matrice H_0 tel que la matrice $S_1^J H_0 S_1$ soit sous la forme :

$$\left(\begin{array}{cccccc|cccccc}
a_1 & \oplus & & & & & c_1 & b_1 & \oplus & & & \\
\oplus & a_2 & & & & & b_1 & c_2 & b_2 & & & \\
 & & a_3 & & & & \oplus & b_2 & c_3 & \ddots & & \\
 & & & \ddots & & & & & \ddots & \ddots & \ddots & \\
 & & & & \ddots & & & & & \ddots & \ddots & b_{n-1} \\
 & & & & & a_n & & & & & b_{n-1} & c_n \\
\hline
q_1 & \oplus & & & & & -a_1 & \oplus & & & & \\
\oplus & q_2 & & & & & \oplus & -a_2 & & & & \\
 & & q_3 & & & & & & \ddots & & & \\
 & & & \ddots & & & & & & \ddots & & \\
 & & & & \ddots & & & & & & \ddots & \\
 & & & & & q_n & & & & & & -a_n
\end{array}\right).$$

4.5.1.1.3 Cas du quadruple shift

Dans une étape de l'algorithme *SR* implicite avec un quadruple shift pour une matrice réelle H Hamiltonienne, le polynôme du shift est de la forme suivante :

$$\begin{aligned} p_k(H) &= (H - \mu I)(H + \mu I)(H - \bar{\mu} I)(H + \bar{\mu} I) \\ &= (H^2 - \mu^2 I)(H^2 - \bar{\mu}^2 I) \\ &= H^4 - (\mu^2 + \bar{\mu}^2)H^2 + (\mu^2\bar{\mu}^2)I, \end{aligned}$$

avec $\mu \in \mathbb{C}$ et $\Re e(\mu) \neq 0$.

Dans ce cas, nous calculons les valeurs propres $\{\mu, -\mu, \bar{\mu}, -\bar{\mu}\}$ en utilisant les quartes valeurs propres la sous-matrice

$$B = \left(\begin{array}{cc|cc} a_{n-1} & 0 & c_{n-1} & b_{n-1} \\ 0 & a_n & b_{n-1} & c_n \\ \hline q_{n-1} & 0 & -a_{n-1} & 0 \\ 0 & q_n & 0 & -a_n \end{array}\right).$$

En même, nous n'avons pas besoin de calculer les valeurs propres de la matrice B. En fait, nous savons que le polynôme caractéristique de la matrice B est sous la forme suivante :

$$z^4 - (a_n^2 + a_{n-1}^2 + q_n c_n + q_{n-1} c_{n-1}) z^2 + a_n^2 a_{n-1}^2 + a_n^2 q_{n-1} c_{n-1} + a_{n-1}^2 q_n c_n - q_{n-1} q_n b_{n-1}^2.$$

Nous remarquons aussi que itération en quadruple shift implicite peut également être utilisée pour effectuer deux itérations du double shift, dans ce cas nous notons par μ et γ ces deux shift, qui sont soit réels ou bien purement imaginaires. Ainsi, le polynôme du shift est de la forme

$$\begin{aligned} p_k(H) &= (H - \mu I)(H + \mu I)(H - \gamma I)(H + \gamma I) \\ &= (H^2 - \mu^2 I)(H^2 - \gamma^2 I) \\ &= H^4 - (\mu^2 + \gamma^2) H^2 + (\mu^2 \gamma^2) I. \end{aligned}$$

La première colonne de $p_k(H)$ est sous la forme

$$\begin{aligned} p_k(H) e_1 &= \begin{pmatrix} a_1^2 + q_1 c_1 + q_1 q_2 b_1^2 - (\mu^2 + \gamma^2)(a_1^2 + q_1 c_1) + \mu^2 \gamma^2 \\ q_1 b_1 [(a_1^2 + q_1 c_1 + a_2^2 + q_2 c_2) - (\mu^2 + \gamma^2)] \\ q_1 q_2 b_1 b_2 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\ &= [a_1^2 + q_1 c_1 + q_1 q_2 b_1^2 - (\mu^2 + \gamma^2)(a_1^2 + q_1 c_1) + \mu^2 \gamma^2] e_1 + \\ &\qquad q_1 b_1 [(a_1^2 + q_1 c_1 + a_2^2 + q_2 c_2) - (\mu^2 + \gamma^2)] e_2 + q_1 q_2 b_1 b_2 e_3. \end{aligned}$$

D'une manière générale, la première colonne de $p_k(H)$ est sous la forme

$$\begin{aligned} p_k(H) e_1 = [a_1^2 + q_1 c_1 + q_1 q_2 b_1^2 - (a_n^2 + q_n c_n + a_{n-1}^2 + q_{n-1} c_{n-1})(a_1^2 + q_1 c_1) + \\ (a_n^2 + q_n c_n)(a_{n-1}^2 + q_{n-1} c_{n-1}) q_{n-1} q_n b_{n-1}^2] e_1 + \\ q_1 b_1 [(a_1^2 + q_1 c_1 + a_2^2 + q_2 c_2) - a_n^2 - q_n c_n - a_{n-1}^2 - q_{n-1} c_{n-1}] e_2 + q_1 q_2 b_1 b_2 e_3. \end{aligned}$$

Par conséquent, la première étape de l'algorithme *SR* implicite introduit des bulges (bosses) dans la matrice H_0 tel que la matrice $S_1^J H_0 S_1$ soit sous la forme :

$$\left(\begin{array}{ccccc|ccccc}
a_1 & \oplus & \oplus & & & c_1 & b_1 & \oplus & \oplus & \\
\oplus & a_2 & \oplus & & & b_1 & c_2 & b_2 & \oplus & \\
\oplus & \oplus & a_3 & & & \oplus & b_2 & c_3 & b_2 & \\
 & & & \ddots & & \oplus & \oplus & b_3 & c_4 & \ddots \\
 & & & & \ddots & & & & \ddots & \ddots \\
 & & & & a_n & & & & & b_{n-1} \\
\hline
q_1 & \oplus & \oplus & & & -a_1 & \oplus & \oplus & & \\
\oplus & q_2 & \oplus & & & \oplus & -a_2 & \oplus & & \\
\oplus & \oplus & q_3 & & & \oplus & \oplus & -a_3 & & \\
 & & & \ddots & & & & & \ddots & \\
 & & & & \ddots & & & & & \ddots \\
 & & & & q_n & & & & & -a_n
\end{array}\right) \begin{array}{c|cc} & & \\ & & \\ & & \\ & & \\ & & \\ b_{n-1} & c_n & \\ \hline & & \\ & & \\ & & \\ & & \\ & & \\ & & \end{array}.$$

Remarque 4.5.1.

— *Soit la matrice B de taille* 2×2 *tel que :*

$$B = \begin{pmatrix} a_k & c_k \\ q_k & -a_k \end{pmatrix}.$$

Les valeurs propres de la matrice B sont données par

$$\lambda_i = \pm\sqrt{a_k^2 + q_k c_k},$$

pour $i \in \{1,2\}$.

— *Soit la matrice B de taille* 4×4 *tel que :*

$$B = \left(\begin{array}{cc|cc} a_{k-1} & 0 & c_{k-1} & b_{k-1} \\ 0 & a_k & b_{k-1} & c_k \\ \hline q_{k-1} & 0 & -a_{k-1} & 0 \\ 0 & q_k & 0 & -a_k \end{array}\right).$$

Les valeurs propres de la matrice B sont données par

$$\lambda_i = \pm\sqrt{-\frac{a_{k-1}^2 + q_{k-1}c_{k-1} + a_k^2 + q_k c_k}{2} \pm \sqrt{\left(\frac{a_{k-1}^2 + q_{k-1}c_{k-1} + a_k^2 + q_k c_k}{2}\right)^2 + q_{k-1}q_k b_{k-1}^2}},$$

pour $i \in \{1,2,3,4\}$.

4.5.1.2 La déflation

Pour faire la déflation, nous suivant la même démarche que le cas général. En fait, l'analyse précédente de la stratégie du simple, double ou quadruple shift montre que les coefficients de la sous-diagonale du blocs H_{12}, qui est une matrice de Hessenberg tridiagonale et symétrique, convergent vers 0 très rapidement. Dans la $k^{\text{ième}}$ itération, quand le coefficient $H(k,2k-1)$ (b_k) est suffisamment petit, nous réduisons la taille en supprimant la $k^{\text{ième}}$ et la $2k^{\text{ième}}$ lignes et la $k^{\text{ième}}$ et la $2k^{\text{ième}}$ colonnes de la matrice H_k courante. Nous continuons l'algorithme SR avec la matrice Hamiltonienne de J-Hessenberg réduite $H_k([1:k-1,k+1:2k-1],[1:k-1,k+1:2k-1])$ et nous choisirons les valeurs propres de la matrice B de taille (2×2) ou (4×4), comme des shift pour l'itération suivante, et ainsi de suite, jusqu'à la fin.

En d'autres termes, soit H une matrice Hamiltonienne sous la forme de J-Hessenberg :

$$H=\left(\begin{array}{cccc|cccc} a_1 & & & & c_1 & b_1 & & \\ & a_2 & & & b_1 & \ddots & \ddots & \\ & & \ddots & & & \ddots & \ddots & b_{n-1} \\ & & & a_n & & & b_{n-1} & c_n \\ \hline q_1 & & & & -a_1 & & & \\ & q_2 & & & & -a_2 & & \\ & & \ddots & & & & \ddots & \\ & & & q_n & & & & -a_n \end{array}\right).$$

Au début de l'étape $k^{\text{ième}}$ de l'itération, nous vérifions si certains élément b_k est négligeable, c'est-à-dire si nous avons

$$|b_k| < tol \times (|a_k| + |a_{k-1}|), \tag{4.8}$$

avec $tol = \|H\|_F \times \sqrt{eps}$ où eps est la précision relative de la machine. Dans le cas où nous avons la condition (4.8), alors nous pouvons diviser notre problème en deux plus petits sous problèmes pour les matrices Hamiltoniennes de J-Hessenberg :

$$\left(\begin{array}{cccc|cccc} a_1 & & & & c_1 & b_1 & & \\ & a_2 & & & b_1 & \ddots & \ddots & \\ & & \ddots & & & \ddots & \ddots & b_{k-1} \\ & & & a_k & & & b_{k-1} & c_k \\ \hline q_1 & & & & -a_1 & & & \\ & q_2 & & & & -a_2 & & \\ & & \ddots & & & & \ddots & \\ & & & q_k & & & & -a_k \end{array}\right)$$

et

$$\left(\begin{array}{cccc|cccc} a_{k+1} & & & & c_{k+1} & b_{k+1} & & \\ & a_{k+2} & & & b_{k+1} & \ddots & \ddots & \\ & & \ddots & & & \ddots & \ddots & b_{n-1} \\ & & & a_n & & & b_{n-1} & c_n \\ \hline q_{k+1} & & & & -a_{k+1} & & & \\ & q_{k+2} & & & & -a_{k+2} & & \\ & & \ddots & & & & \ddots & \\ & & & q_n & & & & -a_n \end{array}\right)$$

et nous procédons à travailler sur chaque bloc séparément.

Finalement, l'algorithme *SR* implicite avec shift pour une matrice Hamiltonienne est le même algorithme dans le cas général en changeant seulement l'algorithme $SRstep$, plus précisément nous changeons le calcule de la première colonne du polynôme de shift.

4.6 Exemples numériques

Nous avons testé nos algorithmes sur des matrices arbitraires. Nous citons ces exemples :
Cas d'une matrice arbitraire :

T_{SR}	T_{eig}	$v_{SR} - v_{eig}$
$4.3239e+00+0.0000e+00i$	$4.3239e+00+0.0000e+00i$	$4.3561e-11+0.0000e+00i$
$-8.2879e-01+0.0000e+00i$	$-8.2879e-01+0.0000e+00i$	$3.7448e-06+0.0000e+00i$
$6.2492e-01+0.0000e+00i$	$6.2492e-01+0.0000e+00i$	$-4.1744e-13+0.0000e+00i$
$2.0063e-01+4.2378e-01i$	$2.0063e-01+4.2378e-01i$	$-6.0982e-10-3.4907e-10i$
$2.0063e-01-4.2378e-01i$	$2.0063e-01-4.2378e-01i$	$-6.0982e-10+3.4907e-10i$
$4.1320e-01+4.8298e-02i$	$4.1320e-01+4.8298e-02i$	$-1.8902e-13-2.2458e-13i$
$4.1320e-01-4.8298e-02i$	$4.1320e-01-4.8298e-02i$	$-1.8902e-13+2.2458e-13i$
$-3.8101e-01+0.0000e+00i$	$-3.8101e-01+0.0000e+00i$	$-6.0209e-09+0.0000e+00i$

T_{SR}	T_{eig}	$v_{SR} - v_{eig}$
$5.7196e+00+0.0000e+00i$	$5.7196e+00+0.0000e+00i$	$1.7759e-07+0.0000e+00i$
$1.3407e+00+0.0000e+00i$	$1.3408e+00+0.0000e+00i$	$9.4579e-05+0.0000e+00i$
$-7.2988e-01+4.5358e-01i$	$-7.2988e-01+4.5358e-01i$	$-2.0442e-09-1.3193e-08i$
$-7.2988e-01-4.5358e-01i$	$-7.2988e-01-4.5358e-01i$	$-2.0442e-09+1.3193e-08i$
$-7.7519e-01+1.4282e-01i$	$-7.7519e-01+1.4282e-01i$	$-4.0762e-09+8.7880e-09i$
$-7.7519e-01-1.4282e-01i$	$-7.7519e-01-1.4282e-01i$	$-4.0762e-09-8.7880e-09i$
$4.3640e-01+3.5379e-01i$	$4.3639e-01+3.5379e-01i$	$-4.7949e-06-4.6784e-06i$
$4.3640e-01-3.5379e-01i$	$4.3639e-01-3.5379e-01i$	$-4.7949e-06-4.6784e-06i$
$-5.1280e-01+0.0000e+00i$	$-5.1280e-01+0.0000e+00i$	$7.2331e-09+0.0000e+00i$
$4.7201e-01+0.0000e+00i$	$4.7201e-01+0.0000e+00i$	$-1.6384e-06+0.0000e+00i$
$6.2609e-02+2.9070e-01i$	$6.2609e-02+2.9070e-01i$	$5.2299e-08-1.9355e-09i$
$6.2609e-02-2.9070e-01i$	$6.2609e-02-2.9070e-01i$	$5.2299e-08+1.9355e-09i$

Cas d'une matrice Hamiltonienne :

4.7 La préservation de la structure double pour les matrices Hamiltoniennes et symétriques/antisymétriques

T_{SR}	T_{eig}	$v_{SR} - v_{eig}$
$-4.9995e+00+0.0000e+00i$	$-4.9995e+00+0.0000e+00i$	$0.0000e+00+0.0000e+00i$
$4.9995e+00+0.0000e+00i$	$4.9995e+00+0.0000e+00i$	$-8.8818e-16+0.0000e+00i$
$5.7340e-01+1.7693e-01i$	$5.7340e-01+1.7693e-01i$	$-2.4425e-15-8.3267e-16i$
$5.7340e-01-1.7693e-01i$	$5.7340e-01-1.7693e-01i$	$-2.4425e-15+8.3267e-16i$
$-5.7340e-01+1.7693e-01i$	$-5.7340e-01+1.7693e-01i$	$6.6613e-16-1.4155e-15i$
$-5.7340e-01-1.7693e-01i$	$-5.7340e-01-1.7693e-01i$	$6.6613e-16+1.4155e-15i$
$-5.5956e-01+0.0000e+00i$	$5.5956e-01+0.0000e+00i$	$1.1191e+00+0.0000e+00i$
$5.5956e-01+0.0000e+00i$	$-5.5956e-01+0.0000e+00i$	$-1.1191e+00+0.0000e+00i$

Dans ces tableaux, nous désignons par T_{SR} le vecteur des valeurs propres v_{SR} qui ont été calculées par notre méthode et par T_{eig} le vecteur des valeurs propres v_{eig} qui ont été calculées par la fonction eig du Matlab.

4.7 La préservation de la structure double pour les matrices Hamiltoniennes et symétriques/antisymétriques

Dans [109], un algorithme a été proposé pour calculer les valeurs et les vecteurs propres d'une matrice symétrique et Hamiltonienne ou antisymétrique et Hamiltonienne. Ce dernier, exploite la structure double de la matrice d'origine (symétrique/antisymétrique et Hamiltonienne). D'abord, l'algorithme réduit la matrice sous la forme d'une matrice de type Hessenberg en se basant sur les transformations de similarités au sens de Van Loan [90, 114]. Sachant que de telles transformations préservent les doubles structures de la matrice. En conséquence, la structure du spectre est préservée et le coût du calcul est divisé par quatre.

Dans ce paragraphe, nous revisitons la méthode, en présentant une nouvelle façon de mettre en œuvre la réduction de la matrice sous la forme adéquate [103]. Cette réduction est basée sur les transformations de Givens et les transformations de Householder au sens de Van Loan [90, 114]. Ensuite, l'algorithme QR par bloc sous sa forme implicite, comme indiqué dans [109], a été utilisé. Les expériences numériques montrent une convergence cubique sous des stratégies appropriées pour le choix du shift.

La méthode proposée dans [109] concerne les matrices Hamiltoniennes qui possèdent également une structure symétrique ou antisymétrique. De plus, ces structures supplémentaires sont préservées par les transformations de similarités orthogonales. Nous notons que les transformations utilisées sont orthogonales et symplectiques. De cette façon, les deux structures sont préservées et la stabilité rétrograde (backward) est également garantie.

Soit la matrice $S = \begin{bmatrix} U & V \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$, avec les blocs U, $V \in \mathbb{R}^{2n\times n}$. Ainsi, la matrice S est orthogonale et symplectique si et seulement si $U^{\mathrm{T}}U = I$, $U^{\mathrm{T}}JU = 0$ et $V = J^{\mathrm{T}}U$. Dans un cas particulier, si la matrice $Q \in \mathbb{R}^{n\times 2n}$ est une matrice orthogonale, alors la matrice

$$S = \begin{bmatrix} Q & 0 \\ 0 & Q \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$$

est une matrice orthogonale et symplectique.

Les algorithmes donnés dans [109] ont utilisé deux types des transformations de similarité orthogonales et symplectiques.

Le premier type est

$$H(k,w) = \begin{pmatrix} \text{diag}(I_{k-1,}, P) & 0 \\ 0 & \text{diag}(I_{k-1}, P) \end{pmatrix}, \tag{4.9}$$

avec

$$P = I - 2ww^{\mathrm{T}}/w^{\mathrm{T}}w,\ w \in \mathbb{R}^{n-k+1}.$$

Nous rappelons que la transformation $H(k,w)$ est tout simplement la somme directe de deux matrices $n \times n$ de Householder ordinaires [116]. Nous nous référons à la transformation $H(k,w)$ par la transformations de Householder au sens de Van Loan.

Le deuxième type est

$$J(k,c,s) = \begin{pmatrix} C & S \\ -S & C \end{pmatrix}, \tag{4.10}$$

avec $c^2 + s^2 = 1$, et

$$C = \text{diag}(I_{k-1}, c, I_{n-k})$$
$$S = \text{diag}(0_{k-1}, s, 0_{n-k}).$$

Nous rappelons que la transformation $J(k,c,s)$ est une transformation de Givens, c'est une rotation $2n \times 2n$ ordinaire de Givens dans les plans k et $n+k$ [116]. Nous nous référons à la transformation $J(k,c,s)$ par la rotations de Givens au sens de Van Loan.

Les transformations de Householder et de Givens au sens de Van Loan sont à la fois orthogonales et symplectiques. Notons que pour $i \neq k$ et $i \neq n+k$, nous avons

$$J(k,c,s)e_i = e_i.$$

Nous avons aussi,

$$J(k,c,s)e_k = ce_k - se_{n+k}$$

et

$$J(k,c,s)e_{n+k} = se_k + ce_{n+k}.$$

C'est pourquoi la transformation $J(k,c,s)$ ne modifié pas toutes les lignes de la matrice, à la quelle elle a été multiplié, sauf la $k^{\text{ième}}$ et $(n+k)^{\text{ième}}$ lignes. Il est évident aussi que

$$H(k,w)e_i = e_i$$

pour $i = 1,\ldots,k-1$ et pour $i = n+1,\ldots,n+k-1$.

Nous notons que les algorithmes de ces deux transformations sont 3.9 et 3.8 respectivement.

Nous considérons la matrice H avec la structure suivante

$$H = \begin{pmatrix} A & G \\ G & -A \end{pmatrix}, \tag{4.11}$$

où les blocs A et G sont des matrices réelles de taille $n \times n$.

Nous avons le résultat suivant

Théorème 4.7.1. *La structure de la matrice H donnée dans* (4.11) *est conservée dans les produits SH et HS, où la matrice S est une transformations de la forme* (4.9) *ou bien de la forme* (4.10).

Preuve. Ce résultat est évident par simple calcul. □

L'intérêt de ce théorème est qu'il nous suffit de calculer que les n premières colonnes seulement du produit SH ou HS et les autres n dernières colonnes sont immédiatement déduites de la structure de la matrice H. Ainsi, le coût des calculs est divisé par deux. En outre, nous pouvons travailler avec $2n^2$ paramètres seulement au lieu de $4n^2$ paramètres.

4.7.1 La réduction de la forme

Nous considérons ici le cas symétrique. Soit H une matrice Hamiltonienne et symétrique tel que

$$H = \begin{pmatrix} A & G \\ G & -A \end{pmatrix},$$

avec les matrices A et G sont des matrices symétriques ($A^{\mathrm{T}} = A$ et $G^{\mathrm{T}} = G$).

Dans [109], il a été montré que la matrice H peut être transformé par des transformations de similarité orthogonales et symplectique sous la forme

$$\tilde{H} = \begin{pmatrix} T & D \\ D & -T \end{pmatrix},$$

où la matrice T est une matrice symétrique et tridiagonale et la matrice D est une matrice diagonale. Par exemple, pour $n = 4$ la matrice $\tilde{H}$ a la forme

$$\tilde{H} = \left(\begin{array}{cccc|cccc} a_1 & b_1 & & & c_1 & & & \\ b_1 & a_2 & b_2 & & & c_2 & & \\ & b_2 & a_3 & b_3 & & & c_3 & \\ & & b_3 & a_4 & & & & c_4 \\ \hline c_1 & & & & -a_1 & -b_1 & & \\ & c_2 & & & -b_1 & -a_2 & -b_2 & \\ & & c_3 & & & -b_2 & -a_3 & -b_3 \\ & & & c_4 & & & -b_3 & -a_4 \end{array}\right). \tag{4.12}$$

Un algorithme pour calculer cette réduction est dérivé dans [109]. Maintenant, nous présentons une nouvelle variante de celui dans [109]. Nous commençons par décrire en détail cette variante.

En effet, la première étape de la réduction est composée de deux sous-étapes suivantes :

La première sous-étape : nous utilisons $J(n,c,s)$ la transformation de Givens au sens de Van Loan. Lorsque nous appliquons la matrice $J(n,c,s)$ à gauche de la matrice H, elle crée un zéro dans la première colonne de la matrice G à la position $g_{n,1}$. Nous notons que

la matrice $J(k,c,s)$ ne modifie pas les lignes $1,\ldots,n-1$ et $n+1,\ldots,2n-1$ de la matrice H. Nous obtenons

$$J(n,c,s)H = \left(\begin{array}{cccc|cccc} * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & & * & * & * \\ \hline * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \end{array}\right).$$

D'après le théorème 4.7.1, nous calculons que les n premières colonnes du produit $J(n,c,s)H$. Ainsi, les n dernières colonnes restantes sont obtenues à partir de la préservation de la structure de la matrice H et non pas de la symétrie des blocs. Ensuite, nous actualisons la matrice $J(n,c,s)H$ par la multiplication à droite par la matrice $J(n,c,s)^{\mathrm{T}}$ pour avoir $J(n,c,s)HJ(n,c,s)^{\mathrm{T}}$. Cette multiplication ne modifie pas les colonnes $1,\ldots,n-1$ et $n+1,\ldots,2n-1$ de la matrice $J(n,c,s)H$. Nous obtenons

$$H_{(n,1)} = J(n,c,s)HJ(n,c,s)^{\mathrm{T}} = \left(\begin{array}{cccc|cccc} * & * & * & * & * & * & * & \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & & * & * & * \\ \hline * & * & * & & * & * & * & * \\ * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \end{array}\right).$$

Ensuite, nous utilisons la transformation $J(n-1,c,s)$ de telle sorte que la multiplication à gauche crée un zéro dans la première colonne de la matrice G à la position $g_{n-1,1}$. La multiplication de la matrice $H_{(n,1)}$ à gauche par la matrice $J(n-1,c,s)$ ne modifie que les lignes $n-1$ et $2n-1$. Nous obtenons

$$J(n-1,c,s)H_{(n,1)} = \left(\begin{array}{cccc|cccc} * & * & * & * & * & * & * & \\ * & * & * & * & * & * & * & * \\ * & * & * & * & & * & * & * \\ * & * & * & * & & * & * & * \\ \hline * & * & * & & * & * & * & * \\ * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \end{array}\right).$$

De même, la multiplication à droite par la matrice $J(n-1,c,s)^{\mathrm{T}}$ ne modifie que les colonnes $n-1$ et $2n-1$. D'après la théorème 4.7.1, des zéro sont créés dans les positions

$(n+1, n-1)$ et $(1, 2n-1)$ de la matrice actualisée $H_{(n-1,1)}$. Nous obtenons

$$H_{(n-1,1)} = J(n-1,c,s)H_{(n,1)}J(n-1,c,s)^{\mathrm{T}} = \left(\begin{array}{cccc|cccc} * & * & * & * & * & * & & \\ * & * & * & * & * & * & * & * \\ * & * & * & * & & * & * & * \\ * & * & * & * & & * & * & * \\ \hline * & * & & & * & * & * & * \\ * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \end{array}\right).$$

D'une manière générale pour créer les zéros désirés à la $j^{\text{ième}}$ étape, nous utilisons la transformation $J(j,c,s)$ telle que la matrice $J(j,c,s)$ appliquée à gauche de la dernière matrice actualisée $H_{(j+1,1)}$ crée un zéro dans la première colonne de la matrice G à la position $g_{j,1}$. De plus, la matrice $J(j,c,s)$ ne modifie pas les lignes de la matrice $H_{(j+1,1)}$ sauf les lignes j et $n+j$. La multiplication à droite par la matrice $J(j,c,s)^{\mathrm{T}}$ ne modifie pas les colonnes de la matrice $J(j,c,s)H_{(j+1,1)}$ sauf les colonnes j et $n+j$. Ainsi, la matrice actualisée a la forme

$$H_{(j,1)} = J(j,c,s)H_{(j+1,1)}J(j,c,s)^{\mathrm{T}}.$$

Deux zéros sont créés dans les positions $(n+j,1)$ et $(1,n+j)$ de la matrice $H_{(j,1)}$.

Dans la dernière étape, la matrice $H_{(2,1)}$ a la forme

$$H_{(2,1)} = J(2,c,s)H_{(3,1)}J(2,c,s)^{\mathrm{T}} = \left(\begin{array}{cccc|cccc} * & * & * & * & * & & & \\ * & * & * & * & & * & * & * \\ * & * & * & * & & * & * & * \\ * & * & * & * & & * & * & * \\ \hline * & & & & * & * & * & * \\ & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * \end{array}\right).$$

La structure double (Hamiltonien et symétrique) de la matrice actualisée est préservée. De ce fait, nous avons terminé la première sous-étape.

La deuxième sous-étape : nous utilisons $H(2,w)$ la transformation de Householder au sens de Van Loan qui agit sur les lignes de 2 à n des matrices A et G afin de créer des zéros dans la première colonne de la matrice A aux positions $a_{3,1},\ldots,a_{n,1}$. La matrice $H(2,w)$ ne modifie pas la première colonne de la matrice G ainsi que les lignes 1 et $n+1$ de la matrice actualisée. Nous obtenons

$$H(2,w)H_{(2,1)} = \left(\begin{array}{cccc|cccc} * & * & * & * & * & & & \\ * & * & * & * & & * & * & * \\ & * & * & * & & * & * & * \\ & * & * & * & & * & * & * \\ \hline * & & & & * & * & * & * \\ & * & * & * & * & * & * & * \\ & * & * & * & & * & * & * \\ & * & * & * & & * & * & * \end{array}\right).$$

La multiplications à droite par la matrice $H(2,w)^{\mathrm{T}}$ ne modifie la première et la $(n+1)^{\text{ième}}$ colonnes de la matrice $H(2,w)H_{(2,1)}H(2,w)^{\mathrm{T}}$. Elle crée des zéros dans les positions $(1,3:n)$ et $(n+1,n+3:2n)$. De coup, nous obtenons

$$H^{(1)} = H(2,w)H_{(2,1)}H(2,w)^{\mathrm{T}} = \left(\begin{array}{cccc|cccc} * & * & & & * & & & \\ * & * & * & * & & * & * & * \\ & * & * & * & & * & * & * \\ & * & * & * & & * & * & * \\ \hline * & & & & * & * & & \\ & * & * & * & * & * & * & * \\ & * & * & * & & * & * & * \\ & * & * & * & & * & * & * \end{array}\right).$$

La structure double (Hamiltonien et symétrique) de la matrice actualisée est préservée. Ainsi, nous avons complété la deuxième sous-étape de la première étape.

Nous comparons cette première étape à la première étape de l'algorithme donné dans [109]. Dans ce dernier, d'abord une transformation de Householder au sens de Van Loan est utilisée pour introduire des zéros dans la première colonne de la matrice G aux positions $g_{3,1},\ldots,g_{n,1}$. Ensuite, la transformation de Givens au sens de Van Loan est utilisée pour mettre un zéro à la position $g_{2,1}$. Enfin, une transformation de Householder au sens de Van Loan agit sur les lignes et les colonnes de deux matrices A et G de la position 2 jusqu'à la position n pour créer des zéros dans la première colonne de la matrice A aux positions $a_{3,1},\ldots,a_{n,1}$.

Cette première étape met en évidence la différence entre les deux versions. En fait, notre version utilise $n-1$ transformations de Givens au sens de Van Loan suivi d'une transformation de Householder au sens de Van Loan. Alors que la version de [109] utilise d'abord la transformation de Householder au sens de Van Loan suivie d'une transformation de Givens au sens de Van Loan et se termine par une transformation de Householder au sens de Van Loan.

La deuxième étape agit de la même manière sur la deuxième colonne de deux matrices A et G. En outre, les transformations de Givens au sens de Van Loan $J(n,c,s),\ldots,J(3,c,s)$ créent des zéros dans les positions $g_{n,2},\ldots,g_{3,2}$ respectivement. Ensuite, la transformation de Householder au sens de Van Loan $H(3,w)$ crée des zéros dans les positions $a_{4,2},\ldots,a_{n,2}$. De cette façon, nous aurons la matrice $H^{(2)}$ tel que

$$H^{(2)} = \left(\begin{array}{ccccc|ccccc} * & * & & & & * & & & & \\ * & * & * & & & & * & & & \\ & * & * & * & & & & * & * & \\ & & * & * & & & & * & * & \\ \hline * & & & & & * & * & & & \\ & * & & & & * & * & * & & \\ & & * & * & & & * & * & * & \\ & & * & * & & & & * & * & \end{array}\right).$$

D'une manière générale, après $n-1$ étapes la matrice est réduite sous la forme

$$\left(\begin{array}{cccc|cccc} * & * & & & * & & & \\ * & * & * & & & * & & \\ & * & * & * & & & * & * \\ & & * & * & & & * & * \\ \hline * & & & & * & * & & \\ & * & & & * & * & * & \\ & & * & * & & * & * & * \\ & & * & * & & & * & * \end{array}\right).$$

Du point de vue numérique, cette façon de calculer la réduction est significativement différente de celle proposée dans [109] même si les deux méthodes utilisent le même type de transformations élémentaires. La version de [109] commence par une transformation de Householder au sens de Van Loan, suivi par une transformation de Givens au sens de Van Loan et se termine par une transformation de Householder au sens de Van Loan. Cependant que notre variante commence par une transformation de Givens au sens de Van Loan et se termine avec une transformation de Householder au sens de Van Loan. En effet, le coût des deux versions est approximativement le même.

Le cas antisymétrique est obtenu d'une manière similaire au cas symétrique. Soit H une matrice Hamiltonienne et antisymétrique tel que

$$H = \begin{pmatrix} A & -G \\ G & A \end{pmatrix},$$

avec la matrice A est une matrice antisymétrique ($A^{\mathrm{T}} = -A$) et la matrice G est une matrice symétriques ($G^{\mathrm{T}} = G$).

Comme dans le cas symétrique la matrice H peut être transformé par des transformations de similarité orthogonales et symplectique sous la forme

$$\tilde{H} = \begin{pmatrix} T & -D \\ D & T \end{pmatrix},$$

avec la matrice T est une matrice antisymétrique et tridiagonale et la matrice D est une matrice diagonale. Par exemple, pour $n = 4$ la matrice $\tilde{H}$ a la forme

$$\tilde{H} = \left(\begin{array}{cccc|cccc} 0 & -b_1 & & & -c_1 & & & \\ b_1 & 0 & -b_2 & & & -c_2 & & \\ & b_2 & 0 & -b_3 & & & -c_3 & \\ & & b_3 & a_4 & & & & -c_4 \\ \hline c_1 & & & & 0 & -b_1 & & \\ & c_2 & & & b_1 & 0 & -b_2 & \\ & & c_3 & & & b_2 & 0 & -b_3 \\ & & & c_4 & & & b_3 & 0 \end{array}\right). \tag{4.13}$$

4.7.2 L'implicite algorithme QR avec shift

Maintenant, nous supposons que la matrice symétrique Hamiltonienne H a été réduite avec la nouvelle version sous la forme réduite illustrée par la matrice $\tilde{H}$ dans (4.12).

Donc,

$$H = \left(\begin{array}{cccc|cccc} a_1 & b_1 & & & c_1 & & & \\ b_1 & a_2 & b_2 & & & c_2 & & \\ & b_2 & a_3 & b_3 & & & c_3 & \\ & & b_3 & a_4 & & & & c_4 \\ \hline c_1 & & & & -a_1 & -b_1 & & \\ & c_2 & & & -b_1 & -a_2 & -b_2 & \\ & & c_3 & & & -b_2 & -a_3 & -b_3 \\ & & & c_4 & & & -b_3 & -a_4 \end{array}\right).$$

Si nous appliquons la permutation $P = [e_1, e_3, e_5, e_7, e_2, e_4, e_6, e_8]$ à la matrice H, alors le résultat sera

$$H_s = PHP^{\mathrm{T}} = \left(\begin{array}{cc|cc|cc|cc} a_1 & c_1 & b_1 & & & & & \\ c_1 & -a_1 & & -b_1 & & & & \\ \hline b_1 & & a_2 & c_2 & b_2 & & & \\ & -b_1 & c_2 & -a_2 & & -b_2 & & \\ \hline & & b_2 & & a_3 & c_3 & b_3 & \\ & & & -b_2 & c_3 & -a_3 & & -b_3 \\ \hline & & & & b_3 & & a_4 & c_4 \\ & & & & & -b_3 & c_4 & -a_4 \end{array}\right).$$

Les valeurs propres d'une matrice Hamiltonienne symétrique sont réelles et se produisent en paires $\pm\lambda$.

Ensuite, nous appliquons l'algorithme QR présenté dans [86] à cette structure. L'algorithme QR utilisé est un algorithme implicite avec double shift $\pm\rho$ qui chasse la bosse ("implicitly-shifted bulge-chasing algorithm").

En effet, chaque itération de l'algorithme QR prendra la forme

$$\tilde{H}_s = Q_s^{-1} H_s Q_s,$$

où la matrice Q_s est la matrice orthogonale de la décomposition QR

$$(H_s - \rho I)(H_s + \rho I) = Q_s R_s.$$

La matrice R_s est une matrice triangulaire par bloc où chaque bloc est de taille 2×2. Dans cette version implicite ne calculons pas ni la décomposition QR explicitement ni le produit $H_s - \rho I)(H_s + \rho I) = H_s^2 - \rho^2 I$. Comme le processus de l'algorithme SR implicite, nous avons juste besoin des deux premières colonnes de la matrice $H_s^2 - \rho^2 I$. C'est-à-dire nous avons besoin des colonnes 1 et $n+1$ de la matrice $H^2 - \rho^2 I$ qui sont p et $J^{\mathrm{T}} p$ respectivement tel que

$$p = \left[\begin{array}{c} a_1^2 + b_1^2 + c_1^2 - \rho^2 \\ b_1(a_1 + a_2) \\ b_1 b_2 \\ 0 \\ \hline 0 \\ (c_2 - c_1) b_1 \\ 0 \\ 0 \end{array}\right].$$

Nous calculons la transformation symplectique orthogonale Q_1 tel que $Q_1^{\mathrm{T}} p = \alpha e_1$. Puisque la matrice Q_1 est une matrice symplectique et orthogonale, alors nous avons $JQ_1 = Q_1 J$. Ainsi, $Q_1^{\mathrm{T}}(J^{\mathrm{T}} p) = J^{\mathrm{T}} Q_1^{\mathrm{T}} p = \alpha J^{\mathrm{T}} e_1 = \alpha e_{n+1}$. Donc, l'élimination dans $J^{\mathrm{T}} p$ se déroule automatiquement.

La transformation de similarité par la matrice Q_1 nous donne une matrice $H_1 = Q_1^{-1} H Q_1$ Hamiltonienne et symétrique qui n'a plus la forme (4.12) mais qui a des bulges dans les blocs T et D.

Par exemple, pour $n = 7$ la matrice H_1 est sous la forme

$$\left(\begin{array}{ccccccc|ccccccc}
* & * & * & * & & & & * & * & * & & & & \\
* & * & * & * & & & & * & * & * & & & & \\
* & * & * & * & & & & * & * & * & & & & \\
* & * & * & * & * & & & & & & * & & & \\
 & & & * & * & * & & & & & & * & & \\
 & & & & * & * & * & & & & & & * & \\
 & & & & & * & * & & & & & & & * \\
\hline
* & * & * & & & & & * & * & * & * & & & \\
* & * & * & & & & & * & * & * & * & & & \\
* & * & * & & & & & * & * & * & * & & & \\
 & & & * & & & & * & * & * & * & * & & \\
 & & & & * & & & & & & * & * & * & \\
 & & & & & * & & & & & & * & * & * \\
 & & & & & & * & & & & & & * & *
\end{array}\right).$$

Le reste de l'étape de l'algorithme QR consiste à retourner cette matrice sous la forme tridiagonale par bloc avec les mêmes techniques qui ont été décrites dans la section précédente.

Une fois que la matrice a été réduite sous la forme tridiagonale par bloc nous utilisons exactement les mêmes techniques que dans [109] pour stocker la forme réduite obtenue. La mémoire totale requise est exactement la même, c'est-à-dire $\mathcal{O}(n)$. En outre, le coût en flops (L'opération en virgule flottante par seconde) par étape reste identique, c'est à dire $\mathcal{O}(n)$ puisque le nombre de matrices des transformations est $\mathcal{O}(n)$ et que le coût d'application de chaque matrice de transformante est $\mathcal{O}(1)$.

Dans le cas où nous avons besoin de calculer la matrice des transformations de similarité qui est orthogonale et symplectique pour obtenir les vecteurs propres par exemple, nous suivons le même schéma que dans [109].

Pendant les itérations, les entrées b_i tendent en effet à converger vers zéro. Une fois que tous les b_i sont devenus suffisamment petits pour être considérés comme des zéros,

la matrice H a la forme suivante

$$H = \left(\begin{array}{cccc|cccc} a_1 & & & & c_1 & & & \\ & a_2 & & & & c_2 & & \\ & & a_3 & & & & c_3 & \\ & & & a_4 & & & & c_4 \\ \hline c_1 & & & & -a_1 & & & \\ & c_2 & & & & -a_2 & & \\ & & c_3 & & & & -a_3 & \\ & & & c_4 & & & & -a_4 \end{array}\right).$$

Chaque sous-matrice a la forme

$$\begin{pmatrix} a_i & c_i \\ c_i & -a_i \end{pmatrix},$$

admet une paire de valeurs propres $\pm\sqrt{a_i^2 + c_i^2}$.

En utilisant une seule transformation de Givens au sens de Van Loan nous pouvons diagonaliser cette matrice sous la forme

$$\begin{pmatrix} \hat{a}_i & 0 \\ 0 & -\hat{a}_i \end{pmatrix},$$

avec $\hat{a}_i = \sqrt{a_i^2 + c_i^2}$.

Nous avons testé cet algorithme avec les mêmes deux stratégies de shift que dans [109]. La plus simple consiste à prendre comme shift les valeurs propres de la sous-matrice

$$\begin{pmatrix} a_i & c_i \\ c_i & -a_i \end{pmatrix},$$

qui sont $\pm\sqrt{a_i^2 + c_i^2}$. Cette stratégie généralisée de shift via le quotient de Rayleigh a une convergence cubique locale. Parfois cette stratégie se bloque et nécessite beaucoup d'itérations pour trouver une paire de valeurs propres. Nous avons également utilisé la stratégie de shift de Wilkinson dans laquelle nous cherchons les valeurs propres de la sous-matrice

$$H_s = PHP^{\mathrm{T}} = \left(\begin{array}{cc|cc} a_{n-1} & c_{n-1} & b_{n-1} & \\ c_{n-1} & -a_{n-1} & & -b_{n-1} \\ \hline b_{n-1} & & a_n & c_n \\ & -b_{n-1} & c_n & -a_n \end{array}\right),$$

et prenons la plus proche paire de $\pm\sqrt{a_i^2 + c_i^2}$ comme shift. Nous avons remarqué aussi que la convergence est cubique et nous constatons comme dans [109] que cette stratégie est globalement convergente. Pour les problèmes de grandes tailles, nous adoptons la même déflation que celle de [109] avec sa stratégie de shift associée.

Puisque l'algorithme préserve les structures, alors il fournit des parfaites paires de valeurs propres réelles $\pm\lambda$ et les vecteurs propres correspondants sont fournis en paires v, Jv aussi.

Pour le cas antisymétrique, c'est-à-dire pour la matrice Hamiltonienne antisymétrique H qui a été réduite avec la nouvelle version sous la forme réduite illustrée par la matrice $\tilde{H}$ dans (4.13). Ainsi,

$$H = \left(\begin{array}{cccc|cccc} 0 & -b_1 & & & -c_1 & & & \\ b_1 & 0 & -b_2 & & & -c_2 & & \\ & b_2 & 0 & -b_3 & & & -c_3 & \\ & & b_3 & a_4 & & & & -c_4 \\ \hline c_1 & & & & 0 & -b_1 & & \\ & c_2 & & & b_1 & 0 & -b_2 & \\ & & c_3 & & & b_2 & 0 & -b_3 \\ & & & c_4 & & & b_3 & 0 \end{array}\right).$$

Si nous appliquons la permutation $P = [e_1, e_3, e_5, e_7, e_2, e_4, e_6, e_8]$ à la matrice H, alors le résultat sera

$$H_s = PHP^{\mathrm{T}} = \left(\begin{array}{cc|cc|cc|cc} & -c_1 & -b_1 & & & & & \\ c_1 & & & -b_1 & & & & \\ \hline b_1 & & & -c_2 & -b_2 & & & \\ & b_1 & c_2 & & & -b_2 & & \\ \hline & & b_2 & & & -c_3 & -b_3 & \\ & & & b_2 & c_3 & & & -b_3 \\ \hline & & & & b_3 & & & -c_4 \\ & & & & & b_3 & c_4 & \end{array}\right).$$

Les valeurs propres d'une matrice Hamiltonienne antisymétrique sont complexes et se produisent en paires conjugués purement imaginaires $\pm\mu$. Donc, nous utilisons des shift $\pm\rho$ avec ρ est un réel. De même dans l'algorithme QR, nous avons besoin des colonnes 1 et $n+1$ de la matrice $(H - i\rho I)(H^2 + i\rho I) = H^2 + \rho^2 I$ qui sont p et $J^{\mathrm{T}}p$ respectivement tel que

$$p = \left[\begin{array}{c} -b_1^2 - c_1^2 + \rho^2 \\ 0 \\ b_1 b_2 \\ 0 \\ \hline 0 \\ (c_2 - c_1) b_1 \\ 0 \\ 0 \end{array}\right].$$

La matrice Q_1 de la transformation de similarité peut être construite comme dans le cas symétrique tels que $Q_1^{\mathrm{T}} p = \alpha e_1$ et $Q_1^{\mathrm{T}} J^{\mathrm{T}} p = \alpha e_{n+1}$. La transformation de similarité par la matrice Q_1 nous donne la matrice $H_1 = Q_1^{-1} H Q_1$ qui n'a plus la forme (4.13) mais qui a des bulges. Les détails de l'algorithme QR sont légèrement plus simples que dans le cas symétrique en raison des zéros supplémentaires dans la forme antisymétrique. Cet algorithme nous donne des résultats similaires au cas symétrique.

4.7.3 Conclusion

Nous avons développé une nouvelle variante de l'algorithme dans [109]. La nouvelle version préserve les double structures Hamiltonienne et symétriques ou Hamiltoniennes et antisymétriques en donnant les valeurs propres en paires $\pm\lambda$. L'algorithme est robuste et stable numériquement. Les expériences numériques montrent une convergence cubique lorsque des shift adéquats sont choisis.

Chapitre 5

Blind image deconvolution via Hankel based method for computing the GCD of polynomials

« Be yourself. Above all, let who you are, what you are, what you believe shine through every sentence you write, every piece you finish »

John Jakes (1932)

Sommaire

Présentation

Dans ce chapitre, un nouveau point de vue est proposé pour déconvoluer une image à partir de ses versions floues. Dans la z-transformée, l'image désirée peut être considérée comme le plus grand commun diviseur polynomial de deux versions déformées. Ce problème devient un problème de détermination du plus grand commun diviseur (PGCD) de deux ou plus polynômes à des variables (2D). Le PGCD exact n'est pas souhaitable parce que, même une faible variation due à l'erreur de quantification ou du bruit additif peut détruire l'intégrité du système et conduire à une solution triviale. Notre approche pour ce problème de déconvolution d'images floues introduit une nouvelle et robuste (2-D) pour le calcul du PGCD basée sur une approche unidimensionnelle (1-D) via la matrice de Hankel en se basant sur l'inversion de la matrice de Toeplitz via l'algorithme de Bini révisée puis l'algorithme d'interpolation modifiée. Sachant que l'algorithme de PGCD unidimensionnel via Hankel à un coût compétitif $\mathcal{O}(n^2)$.

Dans de nombreux problèmes, y compris la communication, l'imagerie par satellite, radar, l'image émise par les avions sans pilot, etc., la sortie se compose d'une entrée souhaitée qui a été déformée par une fonction de flouage : comme mouvement de la caméra. L'image finale peut être représentée comme le résultat de convolution entre l'image voulue et une fonction de flouage (filtre). Dans toutes ces situations, la déconvolution d'image consiste à déterminer l'image nette et le filtre.

Dans les méthodes existantes qui traitant ces problèmes, Ghiglia (1993) a donné dans [56] une (2-D) approche systématique. Cet algorithme est très sensible au bruit, et a une complexité de calcul de $\mathcal{O}(n^8)$; pour une image de taille $n \times n$. Une autre méthode qui se base sur la matrice de Sylvester (2-D), la taille de la matrice de Sylvester est $2n^2 \times 2n^2$, ainsi on applique la décomposition en valeurs singulières (SVD) qui est proportionnel au cube de la taille du cette matrice alors la complexité de l'algorithme est $\mathcal{O}(n^6)$. Puis une autre (voir [92]) approche basée sur la matrice de Sylvester et la FFT coût $\mathcal{O}(n^4)$. En effet, Il est bien connu que la matrice de Bézout peut également être utilisée pour calculer le PGCD de deux polynômes. Par rapport à la matrice de Sylvester, la matrice de Bézout à une plus petite taille. Par exemple, la taille de la matrice Sylvester de deux polynômes avec le même degré n est $2n \times 2n$, tandis que la taille de la matrice Bézout est $n \times n$. En outre, la complexité de déconvolution d'image de taille $n \times n$ via la matrice de Bézout (en 2010) est $\mathcal{O}(n^2 \log(n))$ (voir [77]). De même notre méthode via la matrice de Hankel et FFT coûte aussi $\mathcal{O}(n^2 \log(n))$ mais elle est plus rapide en point du vu temps que la méthode basée sur la matrice de Bézout.

Ce chapitre est un article en collaboration avec Skander Belhaj, Marwa Dridi et Maher Moakher publié dans la revue "Mathematics and Computers in Simulation", (voir [13]).

Blind image deconvolution via Hankel based method for computing the GCD of polynomials [1]

Abstract : In this chapter we present an algorithm, that is based on computing approximate greatest common divisors (GCD) of polynomials, for solving the problem of blind image deconvolution. Specifically, we design a specialized algorithm for computing the GCD of bivariate polynomials corresponding to z-transforms of blurred images to recover the original image. The new algorithm is based on the fast GCD algorithm for univariate polynomials in which the successive transformation matrices are upper triangular Toeplitz matrices. The complexity of our algorithm is $\mathcal{O}(n^2\log(n))$ where the size of blurred images is $n \times n$. All algorithms have been implemented in Matlab and experimental results with synthetically blurred images are included to illustrate the effectiveness of our approach.

5.1 Introduction

Blind image deconvolution, which appears in a wide range of applications, such as astronomical imaging [69, 111], remote sensing [32], light microscopy [57], medical imaging [85], optics [88, 94], photography [115, 121], super resolution imaging [112], and motion tracking [45], among others, is a classical inverse problem in image processing. During acquisition as data pass through the sensing, transmission, and recording processes, images can be distorted. A degradation can be also observed as a result of noise and blurring, which is typically modelled by convolution with some blurring kernel called the Point Spread Function (PSF).

A grey level image can be represented by a matrix whose dimensions are equal to the size of the image and whose entries are the intensities, and a colour image can be represented by three matrices which represent the intensities of the Red, Green and Blue intensities. We have the following relation between the original image matrix P and the distorted image matrix F : $F = P * U + N$ where U is the blurring matrix and N is the additive noise matrix.

In [56], Ghiglia, Romero and Mastin have gave a (2D) systematic approach to the problem of blind image deconvolution. Their algorithm is very sensitive to noise, and has a computational complexity of $\mathcal{O}(n^8)$ operations for an image of size $n \times n$. Kaltofen, Yang and Zhi [67] have proposed an algorithm based on the Sylvester matrix of size $2n^2 \times 2n^2$ and the SVD technique with a reduced complexity of $\mathcal{O}(n^6)$ operations. This method was improved by Pillai and Ben Liang [92] with a substantial saving of complexity which is $\mathcal{O}(n^4)$ operations.

Recently, Li, Yang and Zhi [77] have introduced an approach using the Bézout matrix and Fast Fourier transform (FFT). Their algorithm requires about $\mathcal{O}(n^2\log(n))$ operations. Their idea is to solve the problem of blind image deconvolution by computing approximate greatest common divisors (GCD) of polynomials by using the two-dimensional (2D)

1. Ce chapitre est un article en collaboration avec Skander Belhaj, Marwa Dridi et Maher Moakher publié dans la revue "Mathematics and Computers in Simulation", (voir [13])

z-transform [92] which maps the elements of an $m \times n$ matrix P to the coefficients of the bivariate polynomial

$$p(x, y) = x^{\mathrm{T}} \cdot P \cdot y,$$

where $x = [1, x, x^2, \ldots, x^{m-1}]^{\mathrm{T}}$ and $y = [1, y, y^2, \ldots, y^{n-1}]^{\mathrm{T}}$. Hence,

$$F = P * U + N$$

is transformed to

$$f(x, y) = p(x, y)u(x, y) + v(x, y),$$

where $f(x, y)$, $p(x, y)$, $u(x, y)$ and $v(x, y)$ are the z-transforms of F, P, U and N, respectively. To recover the original image, we compute $p(x, y)$ the GCD of $p_1(x, y)$ and $p_2(x, y)$ corresponding to the z-transform of two distinct blurred images P_1 and P_2 of the same original image $P \in \mathbb{C}^{m \times n}$. Thus, to find the desired bivariate GCD we crucially sample the polynomials p_1 and p_2 in each variable at the DFT points on the unit circle $x_k = e^{\frac{-2k\pi i}{m}}$, $k = 0, 1, \ldots, m-1$ and $y_l = e^{\frac{-2l\pi i}{n}}$, $l = 0, 1, \ldots, n-1$. The GCD of the resulting univariate polynomials is found by using the Bézout-type GCD method [23].

The computation of the approximate GCD of univariate polynomials is beginning to be studied by Collins and Brown (see [34, 35, 43]) via efficient Euclidean-based methods. Naturally, the classical (exact) GCD algorithm is not desirable because even a small change, due to the quantization error or additive noise, can destroy the integrity of the system and leads to a trivial solution. One might think of designing stabilized numerical versions of the Euclidean algorithm to the approximate case (see [89, 110]).

Several papers have been devoted to describe the natural relation between the Euclidean algorithm and the block LU factorization of the Hankel and Bézout matrices (see [11, 12, 23, 24, 25, 26, 51, 55]). For more details on this research topic, we refer the reader to [44, 68, 76, 87, 91, 119, 120, 122, 123] and the references therein.

In this chapter, a new point of view is proposed to solve the problem of image deconvolution from its blurred versions. Our main contribution for this problem is the development of a robust two-dimensional GCD algorithm based on superfast techniques for computing inverses of Toeplitz matrices described in [14, 64, 78] and the techniques for computing the one-dimensional GCD via Hankel matrices, with a competitive cost of $\mathcal{O}(3n^2)$ operations, introduced recently in [11, 12]. This algorithm for computing the bivariate GCD of polynomials via Hankel based method requires only $\mathcal{O}(n^2 \log(n))$ operations.

Although the new algorithm requires a similar complexity with respect to the latest method based on the Bézout matrix [77], the computational time (in sec) is very low. In addition, our method is successful both from the point of view of the numerical criterion PSNR[2] and the visual point of view. Besides, the SSIM[3] of the method is in general very close to 1 with respect to existing methods.

2. The PSNR (Peak Signal-to-Noise Ratio) is most commonly used as a measure of quality of reconstruction of lossy compression codecs (e.g., for image compression). Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, where higher is better.

3. The SSIM (Structural SIMilarity) is a method for measuring the similarity between two images. The resultant SSIM index is a decimal value between −1 and 1, and value 1 is only reachable in the case of two identical sets of data.

The outline of the remainder of this chapter is as follows. In Section 2, we introduce some notations and present classical results. The new bivariate polynomials GCD via a Hankel matrix is proposed in Section 3. In Section 4, we show the effectiveness of our method with respect to experimental results on synthetically blurred images collected from the literature in terms of computation times and numerical criterion (PSNR, SSIM). Finally, a summary and perspectives of future work are given in Section 5.

5.2 Notations and preliminaries

We start by introducing some notations and giving some necessary tools about Toeplitz and Hankel computations which will be used throughout this chapter. Some basic results are extended.

— $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$.

— $H(s) \in \mathbb{K}^{n\times n}$ denotes the Hankel matrix associated to a list s of length $(2n-1)$. This means that the first row is given by the first n terms of s and the last column is given by the last n terms of s.

— $lH(s) \in \mathbb{K}^{n\times n}$ denotes the lower Hankel triangular matrix (with respect to the anti-diagonal) associated to a list s such that the last column is defined by s.

— $uH(s) \in \mathbb{K}^{n\times n}$ denotes the upper Hankel triangular matrix (with respect to the anti-diagonal) associated to a list s such that the first column is defined by s.

— $T(s) \in \mathbb{K}^{n\times n}$ denotes the Toeplitz matrix associated to a list s of length $(2n-1)$. The first row is given by the first n terms of s and the last column is given by the last n terms of s.

— $lT(s) \in \mathbb{K}^{n\times n}$ denotes the lower Toeplitz triangular matrix associated to a list s such that the last row is defined by s.

— $uT(s) \in \mathbb{K}^{n\times n}$ denotes the upper Toeplitz triangular matrix associated to a list s such that the first row is defined by s.

— For $p \in \mathbb{N}$, let $\Sigma_p \in \mathbb{K}^{p\times p}$; $\Sigma_p = [\varepsilon_{jk}]_{j,k=1}^{p}$; where all entries of Σ_p are zero except that $\varepsilon_{j+k,j} = \varepsilon_k$ for $j,k = 1,2,...,p$.

— Given $P \in \mathbb{K}^{n\times m}$; $\tilde{P} = J_m P^t J_n$ where $J_p = lH(1,0,\ldots,0)$; $p \in \mathbb{N}$.

— For $a \in \mathbb{K}$ and $\mu > 0$; $V(a,\mu) = (a-\mu, a+\mu)$ denotes a neighbourhood of a.

5.2.1 1-D z-transform

Definition 5.2.1. *Let $q(z) = q_0 + \cdots + q_{n-1}z^{n-1} \in \mathbb{K}[z]$, and let $Q = (q_0, q_1, \ldots, q_{n-1}) \in \mathbb{K}^n$. Then, $q(z)$ is called the z-transform of Q and denoted by $ZT(Q)$, and Q is called the coefficient vector of q.*

Definition 5.2.2. *Let $F \in \mathbb{K}^n$ and $G \in \mathbb{K}^m$. The convolution of F and G is the element of $\mathbb{K}^{n+m-1}$ given element-wise by*

$$(F * G)(i) = \sum_{j=max(1,i+1-m)}^{min(i,n)} F(j)G(i+1-j),\ 1 \le i \le n+m-1.$$

Lemma 5.2.1. *Let $F \in \mathbb{K}^n$ and $G \in \mathbb{K}^m$. Then the z-transform of the convolution of F and G is the product of their z-transforms :*

$$ZT(F * G) = ZT(F)ZT(G).$$

5.2.2 2-D z-transform

Definition 5.2.3. *The two-dimensional z-transform maps the elements of an $m \times n$ matrix Q to the coefficients of the bivariate polynomial*

$$q(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^{\mathrm{T}} \cdot Q \cdot \boldsymbol{y} = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} q_{i,j} x^i y^j \in \mathbb{K}[\boldsymbol{x},\boldsymbol{y}],$$

where $\boldsymbol{x} = [1, x, x^2, \ldots, x^{m-1}]^{\mathrm{T}}$, $\boldsymbol{y} = [1, y, y^2, \ldots, y^{n-1}]^{\mathrm{T}}$. Then the matrix

$$Q = \begin{pmatrix} q_{0,0} & q_{0,1} & \cdots & q_{0,n-1} \\ q_{1,0} & q_{1,1} & \cdots & q_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m-1,0} & q_{m-1,1} & \cdots & q_{m-1,n-1} \end{pmatrix} \in \mathbb{K}^{m \times n},$$

can be considered as the coefficient matrix of $q(\boldsymbol{x},\boldsymbol{y})$, and $q(\boldsymbol{x},\boldsymbol{y})$ is the two-dimensional z-transform of matrix Q.

5.2.3 Hankel and Toeplitz matrices

We begin this subsection by introducing the following classical results about Toeplitz and Hankel matrices :

Definition 5.2.4. *$H = (h_{i,j})$ is a Hankel matrix if $h_{i,j} = h_{i-k,j+k}$ for all positive k, that is, if the entries of H are invariant under a shift along the anti-diagonal direction. A Hankel matrix is completely defined by its first row and last column.*

Definition 5.2.5. *$T = (t_{i,j})$ is a Toeplitz matrix if $t_{i,j} = t_{i+k,j+k}$ for all positive k, that is, if the entries of T are invariant under a shift along the diagonal direction. A Toeplitz matrix is therefore completely defined by its first row and first column.*

Remark 5.2.1. *Toeplitz and Hankel matrices of size n are completely specified by $2n-1$ parameters, thus requiring less storage space than ordinary dense matrices. Moreover, many computations with Toeplitz or Hankel matrices can be performed faster.*

Proposition 5.2.1. *[26] The multiplication of a Hankel or Toeplitz matrix of size n by a vector can be reduced to multiplication of two polynomials of degree at most $2n$ and performed with a computational cost of $\mathcal{O}(n \log n)$.*

Proposition 5.2.2. *[26] A nonsingular linear system of n equations with Hankel or Toeplitz matrix can be solved with a computational cost of $\mathcal{O}(n \log^2 n)$.*

Theorem 5.2.1. *[51] Let* $u(x) = \sum_{i=0}^{n} u_i x^i$ *and* $v(x) = \sum_{i=0}^{m} v_i x^i$ *be two polynomials in* $\mathbb{K}[x]$ ($\mathbb{K} = \mathbb{R}$ *or* $\mathbb{C}$) *of degree* n *and* m*, respectively, where* $m \leq n$*. The power series expansion of the function* $R(x) = v(x)/u(x)$ *at the infinity :* $R(x) = \sum_{k=0}^{\infty} h_k x^{-k}$ *defines the* $n \times n$ *Hankel matrix,* $H = H(u,v)$ *; associated to* $u(x)$ *and* $v(x)$ *as*

$$H(u,v) = \begin{pmatrix} h_1 & h_2 & h_3 & \cdots & \cdots & h_n \\ h_2 & h_3 & & \iddots & \iddots & h_{n+1} \\ h_3 & & \iddots & \iddots & \iddots & \vdots \\ \vdots & \iddots & \iddots & \iddots & & \vdots \\ \vdots & \iddots & \iddots & & & \vdots \\ h_n & h_{n+1} & \cdots & \cdots & \cdots & h_{2n-1} \end{pmatrix}.$$

In addition, every nonsingular Hankel matrix can be viewed as a Hankel matrix associated to two polynomials.

5.2.4 Approximate block diagonalization of a Hankel matrix

We introduce the approximate block diagonalization of a Hankel matrix presented in [8, 10].

Lemma 5.2.2. *Let* $h = H(h_1, ..., h_{2n-1})$ *be a square Hankel matrix of order* $n \in N^*$ *and let* p *be a given positive constant. Suppose that* $h_j = \varepsilon_j$ *with* $\varepsilon_j \in V(0,\mu)$ *for* $j = 1, 2, ..., p-1$ *and* $h_p \notin V(0,\mu)$*. Then* h *has the form*

$$h = H\left(\varepsilon_1, ..., \varepsilon_{p-1}, h_p, ..., h_{2n-1}\right) = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix}, \tag{5.1}$$

where

$$\begin{aligned} h_{11} &= lH\left(h_p, ..., h_{2p-1}\right) + J_p \Sigma_p, & h_{22} &= H\left(h_{2p+1}, ..., h_{2n-1}\right), \\ h_{12} &= H\left(h_{p+1}, ..., h_{n+p-1}; p; n-p\right), & h_{21} &= h_{12}^t. \end{aligned}$$

We can successively construct from h the following two matrices :

— A square lower Hankel triangular matrix $\mathscr{H}$ of order $(2n-p)$,

$$\mathscr{H} = lH\left(h_p, ..., h_{2n-1}\right) = \begin{pmatrix} 0 & 0 & H_{13} \\ 0 & h_{11} & h_{12} \\ H_{31} & h_{12}^t & h_{22} \end{pmatrix}, \tag{5.2}$$

where $H_{31} = H_{13} = lH(h_p, ..., h_{n-1})$.

— A square upper triangular Toeplitz matrix $\mathscr{T}$,

$$\mathscr{T} = J_{2n-p}\mathscr{H} = uT\left(h_p, ..., h_{2n-1}\right) = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ 0 & t_{22} & t_{23} \\ 0 & 0 & t_{33} \end{pmatrix}, \tag{5.3}$$

where $t_{11} = t_{33} = uT\left(h_p, ..., h_{n-1}\right)$, $t_{22} = J_p h_{11}$, $t_{13} = J_{n-p} h_{22}$, $t_{12} = J_{n-p} h_{12}^t$, and $t_{23} = J_p h_{12}$.

Lemma 5.2.3. *Let T be a nonsingular upper triangular Toeplitz matrix with non-zero diagonal elements. Then $T^{-1} = uT(\mu_1,...,\mu_{2n-p})$ and has the following block decomposition*

$$T^{-1} = \begin{pmatrix} (T^{-1})_{11} & (T^{-1})_{12} & (T^{-1})_{13} \\ 0 & (T^{-1})_{22} & (T^{-1})_{23} \\ 0 & 0 & (T^{-1})_{33} \end{pmatrix} = \begin{pmatrix} t_{11}^{-1} & \widetilde{P} & Q \\ 0 & t_{22}^{-1} & P \\ 0 & 0 & t_{11}^{-1} \end{pmatrix}, \tag{5.4}$$

where

$$P = T(\mu_2,...,\mu_n; p; n-p), \quad \widetilde{P} = J_{n-p}P^tJ_p,$$
$$t_{22}P + t_{23}t_{11}^{-1} = 0_{(p,n-p)}, \quad h_{11}P + Mt_{11}^{-1} = 0_{(p,n-p)},$$
$$t_{11}\widetilde{P} + t_{12}t_{22}^{-1} = 0_{(n-p,p)}, \quad t_{11}Q + t_{12}P + t_{13}t_{11}^{-1} = 0_{(n-p,n-p)}.$$

Theorem 5.2.2. *(Approximate block diagonalization of a Hankel matrix)*
Let $h = H(\varepsilon_1,...,\varepsilon_{p-1},h_p,...,h_{2n-1})$ be an approximate Hankel matrix (i.e., an approximation of an "exact" Hankel matrix $H(0,...,0,h_p,...,h_{2n-1})$) where $\varepsilon_j \in V(0,\mu)$ for $j = 1,2,...,p-1$ with $h_p \notin V(0,\mu)$ and

$$t = uT(h_p,...,h_{n+p-1}), \quad t^{-1} = uT(\mu_1,...,\mu_n).$$

Then

$$h' = (t^{-1})^t h t^{-1} = \begin{pmatrix} h'_{11} & \epsilon' \\ (\epsilon')^t & h'_{22} \end{pmatrix}, \tag{5.5}$$

where

$$h'_{11} = lH(\mu_1,...,\mu_p) + (t_{22}^{-1})^t J_p\Sigma_p t_{22}^{-1},$$
$$h'_{22} = -H(\mu_{p+2},...,\mu_{2n-p}) + P^tJ_p\Sigma_pP, \quad \epsilon' = (t_{22}^{-1})^t J_p\Sigma_pP.$$

Remark 5.2.2. *Theorem 5.2.2 gives an approximate reduction for real Hankel matrix. Thus, if $\varepsilon_j = 0$, Theorem 5.2.2 provides the exact case. To iterate this result, h'_{22} must be a Hankel matrix, but although $-H(\mu_{p+2},...,\mu_{2n-p})$ is a Hankel matrix, $P^tJ_p\Sigma_pP$ is only a symmetric matrix. If we choose ε_j very close to zero, we can conclude that $\Sigma_p \approx 0_p$, and the process converges to the exact case. To solve this problem, we can choose*

$$h'_{22} = -H(\mu_{p+2},...,\mu_{2n-p}) + \Theta,$$

where Θ is a Hankel matrix built from the first column and the last row of $P^tJ_p\Sigma_pP$.

Remark 5.2.3. *The key of our approach is based on the superfast techniques described in [14, 64, 78] for the inversion of a triangular Toeplitz matrix.*

5.2.5 Approximate block diagonalization for $H(u,v)$

Let

$$u(x) = \sum_{k=0}^{n} u_k x^k \text{ and } v(x) = \sum_{k=0}^{m} v_k x^k \tag{5.6}$$

be two polynomials with $\deg(u(x)) = n$, $\deg(v(x)) = m$ and $m < n$. The classical Euclidean algorithm applied to $u(x)$ and $v(x)$ returns a sequence of quotients $q_k(x)$ and remainders $r_k(x)$, such that

$$\begin{aligned} &r_{-1}(x) = u(x), \quad r_0(x) = v(x) \\ &r_{k-2}(x) = r_{k-1}(x)\, q_k(x) - r_k(x), \quad k = 1, \ldots, K \end{aligned} \tag{5.7}$$

where $-r_k(x)$ is the polynomial remainder of the division of $r_{k-2}(x)$ and $r_{k-1}(x)$ and $r_K(x)$ is the greatest common divisor (GCD) of $u(x)$ and $v(x)$.

We recall the correlation between the approximate Euclidean algorithm applied to two polynomials $u(x)$ and $v(x)$ and the approximate block diagonalization of a Hankel matrix [9, 11].

Theorem 5.2.3. *Let $\widetilde{H}(u,v) = H(\varepsilon_1, \ldots, \varepsilon_{n-m-1}, h_{n-m}, \ldots, h_{2n-1})$ be an approximate Hankel matrix as defined in (5.1) associated with two coprime polynomials in $\mathbb{R}[x]$, $u(x) = \sum_{k=0}^{n} u_k x^k$ and $v(x) = \sum_{k=0}^{m} v_k x^k$, $\deg(u(x)) = n$, $\deg(v(x)) = m$ and $m < n$. Let*

$$t = uT(h_{n-m}, \ldots, h_{2n-m-1}), \quad t^{-1} = uT(\mu_1, \ldots, \mu_n).$$

Then

$$\left(t^{-1}\right)^t \widetilde{H}(u,v)\, t^{-1} = \begin{pmatrix} J_{n-m}\widetilde{B}(q,1)\, J_{n-m} & \epsilon' \\ (\epsilon')^t & \widetilde{H}(v,r) \end{pmatrix}, \tag{5.8}$$

where

$$\begin{aligned} J\widetilde{B}(q,1)\, J &= J_{n-m} B(q,1)\, J_{n-m} + J_{n-m}\left(t_{22}^{-1}\right)^t J_{n-m}\Sigma_{n-m} t_{22}^{-1} J_{n-m}, \\ \widetilde{H}(v,r) &= H(v,r) + P^t J_{n-m}\Sigma_{n-m} P, \quad \epsilon' = \left(t_{22}^{-1}\right)^t J_{n-m}\Sigma_{n-m} P, \end{aligned}$$

$q(x)$ and $r(x)$ are the polynomials quotient and remainder of the division $u(x)/v(x)$.

5.2.6 Univariate polynomials GCD of a Hankel matrix

Let us devise the algorithm for the computation of the approximate polynomials GCD applied to two polynomials $u(x)$ and $v(x)$ of degree n and m, respectively, with $m \le n$.

Algorithm 5.1 the algorithm for the computation of the approximate polynomials GCD

Given $u(x) = \sum_{i=0}^{n} u_i x^i$ and $v(x) = \sum_{i=0}^{m} v_i x^i$ two polynomials of degree n and m, respectively, where $m \le n$; this algorithm computes the approximate GCD polynomial.

1: **Step 1 :** Construct $H(u,v) = H(\varepsilon_1, \ldots, \varepsilon_{n-m-1}, h_{n-m}, \ldots, h_{2n-1})$.
2: **Step 2 :** Define an upper triangular Toeplitz matrix $t = uT(h_{n-m}, \ldots, h_{2n-m-1})$.
3: **Step 3 :** Compute t^{-1} via modified interpolation and compute $t^{-t} H(u,v)\, t^{-1}$.
4: **Step 4 :** Set $h'_{11} = h'(1 : n-m, 1 : n-m)$ and $h'_{22} = h'(n-m+1 : n, n-m+1 : n)$.
5: **Step 5 :** Recover the coefficients of the quotient and the remainder polynomials.
6: **Step 6 :** Recursively apply Algorithm 5.1 to $h = H(h'_{22}(1 : m, 1) h'_{22}(m, 1 : m))$ and extract the GCD.

Theorem 5.2.4. *The GCD evaluation of two polynomials via Algorithm 5.1 requires $\mathcal{O}(3n^2)$ flops.*

Remark 5.2.4. *Algorithm 5.1 was tested for several non-coprime polynomials and is numerically stable. For more details, see [12].*

5.3 Image deconvolution via bivariate polynomials GCD for a Hankel matrix

5.3.1 Blind image deconvolution

Blind image deconvolution is the process of identifying both the true image and the blurring function from the degraded image. Let F be an observed image of an original image P. The observed image is related to the original image by

$$F = P * U + N,$$

where U is the blur and N is an additive noise. Using the ideas of S. Pillai and B. Liang in [92], when multiple blurred versions of the same scene are available, the problem of blind image deconvolution can be transformed to computing approximate GCDs of polynomials by the use of z-transforms.

By the two-dimensional z-transform, we map the elements of matrices to coefficients of bivariate polynomials. Hence, $F = P * U + N$, is transformed into

$$f(x,y) = p(x,y)u(x,y) + v(x,y),$$

where $f(x,y)$, $p(x,y)$, $u(x,y)$ and $v(x,y)$ are the z-transforms of F, P, U and N, respectively. Such a model is applicable in all scenarios where the distortion can be modeled as a linear filter acting on the original image. For example, camera motion and intermediate medium in satellite photography. Upon degradation, there may be unwanted information which represents the noise. Thus, the usefulness of an additive noise N in a corrupted image is natural.

Definition 5.3.1. *(Convolution)*
Let P an image and D a matrix (filter). By linear filtering, i.e., convolving P and D, we get a convoluted image $P_1 = P * D$.

Definition 5.3.2. *(Deconvolution)*
Let P_1 and P_2 be two blurred images of the same original image $P \in \mathbb{C}^{m\times n}$, and let D_1 and D_2 be two distinct filters. Then $P_1 = P * D_1$ and $P_2 = P * D_2$.
When applying z-transforms and Lemma 5.2.1, we will have $p_1(x,y) = p(x,y)d_1(x,y)$ and $p_2(x,y) = p(x,y)d_2(x,y)$, where $p(x,y) = ZT(P)$ and $p_k(x,y) = ZT(P_k)$, $d_k(x,y) = ZT(D_k)$ for $k = 1,2$.

5.3.2 Bivariate polynomials GCD for a Hankel Matrix

To find the original image we compute the GCD of two blurred images P_1 and P_2 :

$$p(x,y) = GCD(p_1(x,y), p_2(x,y)).$$

We assume that $\deg_x(p_1) = \deg_x(p_2) = m$ and $\deg_y(p_1) = \deg_y(p_2) = n$.
Thus, to find the bivariate GCD we use the univariate GCD via Hankel matrix as follows :

Let $x_k = e^{\frac{-2k\pi i}{m}}$ and $y_l = e^{\frac{-2l\pi i}{n}}$ be primitive m-th and n-th complex roots of unity, respectively. These knots allow the evaluation using the FFT; we substitute x by $x_k = e^{\frac{-2k\pi i}{m}}$, $0 \le k \le m-1$ into p_1 and p_2. Then we compute

$$c(x_k)p(x_k, y) = GCD(p_1(x_k, y), p_2(x_k, y)).$$

After that, we evaluate y by using FFT; we get $c(x_k)p(x_k, y_l)$ which we put in a matrix A. Similarly, we do this with y then we get

$$d(y_l)p(x, y_l) = GCD(p_1(x, y_l), p_2(x, y_l)),$$

which we put in a matrix B. If we assume that $a_{k+1} = c(x_k)^{-1}$ and $b_{l+1} = d(y_l)^{-1}$ then

$$p(x_k, y_l) = a_{k+1}A_{k+1,l+1} \text{ and } p(x_k, y_l) = b_{l+1}B_{k+1,l+1}.$$

Therefore

$$A_{k+1,l+1}a_{k+1} - B_{k+1,l+1}b_{l+1} = 0, \tag{5.9}$$

which we write in matrix form as :

$$\Gamma\mathbf{y} \triangleq \begin{pmatrix}
A_{1,1} & 0 & \dots & 0 & -B_{1,1} & 0 & \dots & 0 \\
A_{1,2} & 0 & \dots & 0 & 0 & -B_{1,2} & \dots & 0 \\
A_{1,3} & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
A_{1,n} & 0 & \dots & 0 & 0 & 0 & 0 & -B_{1,n} \\
0 & A_{2,1} & \dots & 0 & -B_{2,1} & 0 & \dots & 0 \\
0 & A_{2,2} & \dots & 0 & 0 & -B_{2,2} & \dots & 0 \\
0 & A_{2,3} & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & A_{2,n} & 0 & 0 & 0 & 0 & \dots & -B_{2,n} \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & \dots & A_{m,1} & -B_{m,1} & 0 & \dots & 0 \\
0 & 0 & \dots & A_{m,2} & 0 & -B_{m,2} & \dots & 0 \\
0 & 0 & \dots & A_{m,3} & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \dots & A_{m,n} & 0 & 0 & \dots & -B_{m,n}
\end{pmatrix}
\begin{pmatrix}
a_1 \\ \vdots \\ a_m \\ b_1 \\ \vdots \\ b_n
\end{pmatrix} = 0. \tag{5.10}$$

Equation (5.10) is extremely overdetermined homogeneous system and it can be solved through SVD, such that a solution is given by the eigenvector of the matrix $\Gamma^{\mathrm{T}}\Gamma$ corresponding to the smallest singular value of Γ. Then, the estimated Fourier transform of the original image is given by

$$p(x_k, y_l) = \frac{1}{2}(A_{k+1,l+1}a_{k+1} + B_{k+1,l+1}b_{l+1}), \text{ for } 0 \le k \le m-1 \text{ and } 0 \le l \le n-1. \tag{5.11}$$

By taking the inverse Fourier transform of (5.11), we obtain an estimate of the original image. Thus, we give the algorithm for computing the approximate bivariate polynomial GCD algorithm via Hankel matrices as follows :

Algorithm 5.2 The approximate bivariate polynomial GCD algorithm via Hankel based method

Input : $p_1(x,y), p_2(x,y) \in \mathbb{C}[x,y]$, with $\deg_x(p_1) = \deg_x(p_2) = m$ and $\deg_y(p_1) = \deg_y(p_2) = n$.

Output : $p(x,y) \in \mathbb{C}[x,y]$: an approximate GCD computed via a Hankel matrix of p_1 and p_2.

1: **Step 1 :** Apply the Approximate Univariate Polynomial GCD Algorithm via a Hankel matrix to compute $[p(x_k, y_l)]$, where

$$x_k = e^{\frac{-2k\pi i}{m}},\ 0 \le k \le m-1 \text{ and } y_k = e^{\frac{-2l\pi i}{n}},\ 0 \le l \le n-1.$$

2: **Step 2 :** Apply inverse FFT to $[p(x_k, y_k)]$ to compute $p(x,y)$.

Theorem 5.3.1. *The algorithm for identifying both the original image and the blurring functions from blurred images of size $n \times n$ when the blurring functions have very low degree requires $\mathcal{O}(n^2 \log(n))$ flops. Explicitly, Algorithm 5.2 is bounded by $\mathcal{O}(5n^2 \log(n))$ flops.*

Proof. **Idea on the cost of Algorithm 5.2.** The GCD evaluation requires $\mathcal{O}(3n^2)$ flops. Thus, Step 1 is bounded by $c \times \mathcal{O}(n^2)$ flops. The very expansive operation in this algorithm is the FFT of the matrix (5.11) which costs $\mathcal{O}(n^2 \log(n))$ flops (Step 2). So, as a result the algorithm requires $\mathcal{O}(5n^2 \log(n))$ flops. □

Remark 5.3.1. *Algorithm 5.2 is bounded by $\mathcal{O}(5n^2 \log(n))$ flops which is about half of the complexity of the method based on Bézout (See [77]).*

Remark 5.3.2. *In the case where we have only one image, it is easy to consider two distinct portions from the blurred image. So, we can see that the GCD of the two distinct portions will be the filter. Since we know the blurring function, we can find the original image. In the case where we have only one blurred RGB image, we assume that the three channels have the same blurring function. Hence, we can get $F_1 = P_1 * U + N_1$, $F_2 = P_2 * U + N_2$ and $F_3 = P_3 * U + N_3$. By z-transforms, we have $f_1(x,y) = p_1(x,y) * u(x,y) + v_1(x,y)$, $f_2(x,y) = p_2(x,y) * u(x,y) + v_2(x,y)$ and $f_3(x,y) = p_3(x,y) * u(x,y) + v_3(x,y)$. Then, the blurring function $u(x,y)$ is now the approximate GCD of f_1, f_2, f_3. For more details, see [92].*

5.4 Experimental results

The following examples come from the literature on image deconvolution [56, 67, 77, 92]. In the simulations, images will be corrupted by "Salt & Pepper" noise. Also a wide range of noise levels starting from 5% will be tested. Here, the noise level is fixed to 5%. Restoration performances are quantitatively measured by the peak signal-to-noise ratio (PSNR). For simplicity, a 3×3 distortion filter constructed randomly via Matlab is used.

Our algorithm can successfully reconstruct original images from blurred images in few seconds. We have implemented both Hankel-type and Bézout-type univariate GCD algorithms in Matlab (R2012A). All experiments are ran on an Intel(R) Core(TM)2 CPU T5600 laptop with a 1.83GHz processor and 2046Mb of RAM under Windows.

Example 1. (Reconstructing Gray level Image from two Blurred Images)
In figure 5.1, Figure 5.1a is the original image of Cameraman of size 512×512 [77, page 6]. Figures 5.1b and 5.1c are two images built by convolving Figure 5.1a with a 3×3 distortion filter. Figure 5.1e is the image reconstructed in about 0.51 seconds by running Algorithm 5.2. Whereas the time obtained by running the algorithm in [77] (Figure 5.1d) is 1.03 seconds.

(a) Original image (b) Distorted image 1 (c) Distorted image 2

(d) Image via Bézout [77] (e) Image via Hankel

FIGURE 5.1 – Blind debluring from two distorted images.

Example 2. (Reconstructing Gray level Image from two Blurred Images)
In figure 5.2, Figure 5.2a is the original image of size 200×200 [62]. Figures 5.2b and 5.2c are two images built by convolving Figure 5.2a with a 3×3 distortion filter. Figure 5.2e is the image reconstructed by running Algorithm 5.2 with restoration performances $PSNR = +25.50dB$ and $SSIM = 0.9768$. Whereas restoration performances obtained by running the algorithm [77] (Figure 5.2d) are $PSNR = +13.93dB$ and $SSIM = 0.9491$.

Example 3. (Reconstructing RGB Image from two Blurred Images)
In figure 5.3, Figure 5.3a is the original image of size 256×256 recovered from Matlab database. Figures 5.3b and 5.3c are two images built by convolving Figure 5.3a with a 3×3 distortion filter. Figure 5.3e is the image reconstructed in about 0.34 seconds by running Algorithm 5.2. Whereas the time obtained by running the algorithm in [77] (Figure 5.3d) is 0.57 seconds.

(a) Original image (b) Distorted image 1 (c) Distorted image 2

(d) Image via Bézout [77] (e) Image via Hankel

FIGURE 5.2 – Blind debluring from two distorted images.

(a) Original image (b) Distorted image 1 (c) Distorted image 2

(d) Image via Bézout [77] (e) Image via Hankel

FIGURE 5.3 – Blind debluring from two distorted RGB images.

Example 4. (Reconstructing RGB Image from two Blurred Images)
In figure 5.4, Figure 5.4a is the original image of size 128×128 [77]. Figures 5.4b and 5.4c

are two images built by convolving Figure 5.4a with a 3×3 distortion filter. Figure 5.4e is the image reconstructed in about 0.048 seconds by running Algorithm 5.2. Whereas the time obtained by running the algorithm in [77] (Figure 5.4d) is 0.072 seconds.

(a) Original image (b) Distorted image 1 (c) Distorted image 2

(d) Image via Bézout [77] (e) Image via Hankel

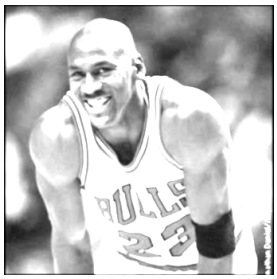
FIGURE 5.4 – Blind debluring from two distorted RGB images.

Example 5. (Reconstructing Grey level Image from one Blurred Image)
In figure 5.5, Figure 5.5a is the original image of size 337×337 recovered from [62]. Figures 5.5b is an image built by convolving Figure 5.5a with a 3×3 distortion filter. Figure 5.5d is the image reconstructed by running Algorithm 5.2 with restoration performances $PSNR = +20.05dB$ and $SSIM = 0.6348$. Whereas restoration performances obtained by running the algorithm [77] (Figure 5.5c) are $PSNR = +18.20dB$ and $SSIM = 0.7494$.

Example 6. (Reconstructing Grey level Image from two Blurred and Noisy Images)
In figure 5.6, Figure 5.6a is the original image of size 200×200 [62]. Figures 5.6b and 5.6c are two distorted and noisy images built by convolving Figure 5.6a with a 3×3 distortion filter and by using the additive "salt & pepper" noise. Figure 5.6e is the image reconstructed by running Algorithm 5.2 with restoration performances $PSNR = +19.43dB$ and $SSIM = 0.9936$. Whereas restoration performances obtained by running the algorithm [77] (Figure 5.6d) are $PSNR = +19.30dB$ and $SSIM = 0.9854$.

Example 7. (Reconstructing RGB Image from two Blurred and Noisy Images)
In figure 5.7, Figure 5.7a is the original image of size 200×200 recovered from Matlab database. Figures 5.7b and 5.7c are two distorted and noisy images built by convolving Figure 5.7a with a 3×3 distortion filter and by using the additive "salt & pepper" noise. Figure 5.7e is the image reconstructed in about 0.31 seconds by running Algorithm 5.2. Whereas the time obtained by running the algorithm in [77] (Figure 5.7d) is 0.41 seconds.

(a) Original image



(b) Distorted image

(c) Image via Bézout [77]

(d) Image via Hankel

FIGURE 5.5 – Blind deblurring from one distorted image.

(a) Original image

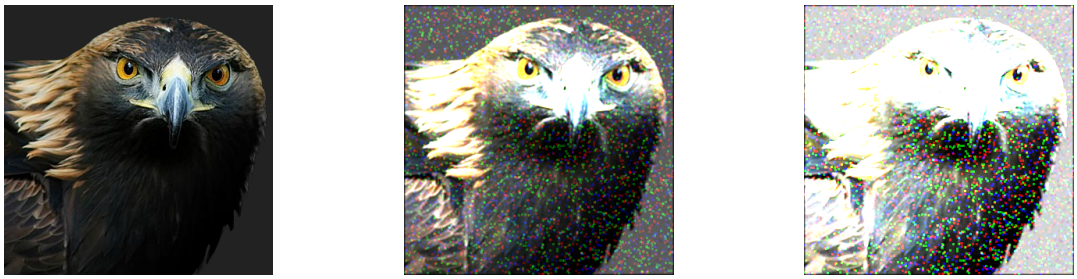
(b) Distorted image 1

(c) Distorted image 2

(d) Image via Bézout [77]

(e) Image via Hankel

FIGURE 5.6 – Blind deblurring from two distorted and noisy images.



(a) Original image (b) Distorted image 1 (c) Distorted image 2

(d) Image via Bézout [77] (e) Image via Hankel

FIGURE 5.7 – Blind deblurring from two distorted and noisy RGB images.

5.4.1 CPU time comparison

It is clear that the Hankel based method and the Bézout based method both require $\mathcal{O}(n^2 log(n))$ operations [77]. Therefore, it is interesting to compare the execution time necessary for each method.

Name of the image	Size of the image	Bézout	Hankel
Laure_gray_256.jpg	256 × 256	0.278725	**0.199620**
Lena_gray_256.tif	256 × 256	0.246597	**0.179761**
Catherine_gray_256.jpg	256 × 256	0.250656	**0.181455**
Cameraman_gray_512.tif	512 × 512	1.032338	**0.511549**

TABLE 5.1 – Comparison of the CPU time for gray-level images

In tables 5.1 and 5.2, we give the computational times (in sec) for the two methods for four different gray level images and four different color images. We observe that the Bézout method requires about twice time than our approach.

5.4.2 Similarity and PSNR

In Table 5.3, we note that the PSNR of the restored image by our method is successful both from the point of view of the numerical criterion (PSNR) and the visual point of view.

Name of the image	Size of the image	Bézout	Hankel
Lena_color_512.tif	512 × 512	3.061329	**1.569268**
Lena_color_256.tif	256 × 256	0.592194	**0.377097**
Laure_color_256.jpg	256 × 256	0.575275	**0.395922**
Catherine_color_256.jpg	256 × 256	0.578055	**0.349498**

TABLE 5.2 – Comparison of the CPU time for color images

	Example 1		Example 2	
	SSIM	PSNR	SSIM	PSNR
Hankel	**0.9395**	**+25.50 dB**	**0.9768**	**+20.14 dB**
Bézout	0.8151	+20.41 dB	0.9491	+13.93 dB

TABLE 5.3 – Reconstruction with two blurred images

In order to measure the robustness of the proposed method, Table 5.4 and Table 5.5 show similarity and PSNR of the image of Example 6 with respect to 3×3 and 11×11 fixed distortion filter (which gives us best results), respectively and a "salt & pepper" additive noise. The level noise is varied from 10% to 30%.

Noise level		SSIM	PSNR
10%	Hankel	**0.9650**	**+13.66 dB**
	Bézout	0.8203	+12.34 dB
20%	Hankel	**0.9674**	**+10.63 dB**
	Bézout	0.8564	+9.49 dB
30%	Hankel	**0.9143**	**+9.48 dB**
	Bézout	0.8997	+9.46 dB

TABLE 5.4 – Measuring performance with a 3×3 fixed distortion filter and a "salt & pepper" additive noise.

To provide a complete analysis of the results, we show the behaviour of the similarity and PSNR of the image $Lena\_gray\_512.tif$ recovered from Matlab with respect to the additive noise and the size filter, respectively.
In Figure 5.8, we convoluted the original image by a 3×3 fixed distortion filter (here, the choice of the fixed filter is based on a filter which gives us the worst results) and a "salt & pepper" additive noise with level noise varied from 1% to 21%. In Figure 5.9, we fixed the "salt & pepper" additive noise to 5% and we convoluted the original image by a random

Noise level		SSIM	PSNR
10%	Hankel	**0.9086**	**+12.93 dB**
	Bézout	0.8951	+11.50 dB
20%	Hankel	**0.8981**	**+11.24 dB**
	Bézout	0.7476	+9.56 dB
30%	Hankel	**0.8270**	**+9.79 dB**
	Bézout	0.7221	+8.72 dB

TABLE 5.5 – Measuring performance with a 11×11 fixed distortion filter and a "salt & pepper" additive noise.

distortion filter varied from 3×3 to 21×21.

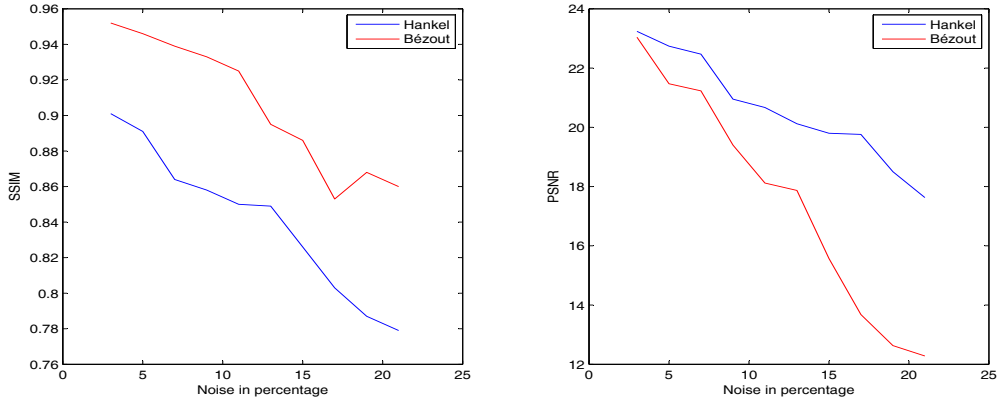


FIGURE 5.8 – Comparison of the behaviour of the similarity and PSNR with respect to the noise level

Based on a negligible difference (about 0.05 for SSIM and 1 for PSNR) for the obtained results, experiments show that our algorithm is the best choice with respect to PSNR and Algorithm [77] is more better than our algorithm with respect to SSIM. In addition, it is natural to say that the image quality for any blind image deconvolution method deteriorates when the size filter and the additive noise increase.

5.4.3 Comparative studies with a standard deconvolution method

In this subsection we propose numerical comparisons to evaluate the performance of the proposed method with respect to the standard deconvolution algorithm. To deblur an image, our choice is the use of the Matlab function "deconvblind.m". The blind standard deconvolution algorithm proposed can be used effectively when no information about

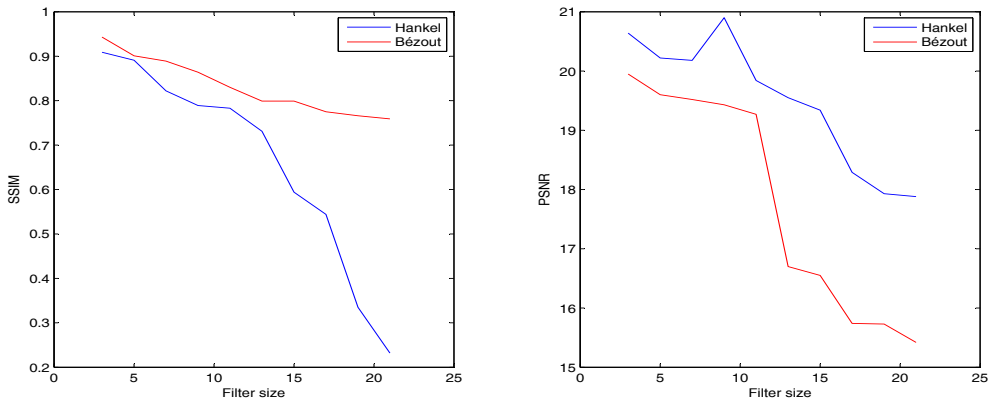


FIGURE 5.9 – Comparison of the behaviour of the similarity and PSNR with respect to the filter size

the distortion (blurring and noise) is known. Note that the function "deconvblind.m" is an iterative and computationally-intensive method. We initialized "deconvblind.m" with "PSF" blur and "Gaussian" noise.

In Table 5.6, we compare the computational time of our method and the standard method. Table 5.7 and Table 5.8 concern the comparison of the two methods for the PSNR and SSIM, respectively.
In general, our algorithm restores an image from two blurred image. To allow the comparison with the function "deconvblind.m" which needs only one blurred image, we can take as inputs for our algorithm two same blurred images.

Name of the image	size of image	Hankel	"deconvblind.m"
Laure_gray_512.jpg	512 × 512	**0.058**	0.716
Lena_gray_256.tif	256 × 256	**0.071**	0.674
Catherine_gray_256.jpg	256 × 256	**0.062**	0.638
Cameraman_gray_512.tif	256 × 256	**0.052**	0.766

TABLE 5.6 – Comparison of the CPU time for gray level images

Based on experimental results, we have shown the performance of the proposed method with respect to the standard deconvolution method. This enables us to implement competitive solution for a blind image deconvolution problem.

Name of the image	size of image	Hankel	"deconvblind.m"
Laure_gray_512.jpg	512 × 512	**+17.32 dB**	+16.85 dB
Lena_gray_256.tif	256 × 256	**+18.42 dB**	+18.11 dB
Catherine_gray_256.jpg	256 × 256	**+17.11 dB**	+16.54 dB
Cameraman_gray_512.tif	512 × 512	**+17.82 dB**	+17.23 dB

TABLE 5.7 – Results for PSNR

Name of the image	size of image	Hankel	"deconvblind.m"
Laure_gray_512.jpg	512 × 512	**0.963**	0.942
Lena_gray_256.tif	256 × 256	**0.935**	0.902
Catherine_gray_256.jpg	256 × 256	**0.944**	0.930
Cameraman_gray_512.tif	512 × 512	**0.926**	0.913

TABLE 5.8 – Results for SSIM

5.5 Conclusion

In this work, we have studied the problem of blind image deconvolution by proposing a robust algorithm using FFT and a Hankel matrix based on the fast inversion of a triangular Toeplitz matrix via a modified interpolation algorithm. It is true that the proposed method via Hankel has the same order of complexity $\mathcal{O}(n^2 \log(n))$ compared to its direct competitor based on the Bézout method [77] but our approach gives a promising result in point of view of computation time. It would be critical to study the numerical stability and the errors appearing in the bivariate GCD algorithm via a Hankel matrix. Our algorithm, based essentially on FFT, has a total cost of $\mathcal{O}(n^2 \log(n))$ arithmetic operations. It would be interesting to improve the FFT, something which is not at all easy, to design a robust algorithm with a low complexity. Finally, we can design a specialized algorithm via generalized Hankel based method for computing the GCD of bivariate polynomials of several blurred images in a compact way by using the idea introduced in [50].

Conclusion et perspectives

« *If you do not change, you can become extinct* »

Spencer Johnson (1938-2017)

Dans cette thèse, nous avons introduit des méthodes d'algèbre linéaire numérique pour le calcul des valeurs et vecteurs propres d'une classe importante de matrices structurées.

D'une part, nous avons proposé une version révisée de l'algorithme *SR*, qui est un algorithme de type *QR* en préservant la structure des matrices Hamiltoniennes, anti-Hamiltoniennes, symplectiques, etc. Cet algorithme est basé sur la combinaison de la décomposition *SR* et la réduction de la matrice en question sous la forme J-Hessenberg.

Ensuite, nous avons présenté des versions améliorées de la décomposition *SR* en se basant sur l'algorithme de Gram–Schmidt symplectique et aussi sur une ré-J-orthogonalisation afin d'améliorer la perte de la J-orthogonalité.

Également, nous avons proposé des versions révisées de la décomposition *SR* via les transformations de Householder symplectiques et aussi sur les transformations au sens de Van Loan (Givens et Householder).

Une étude d'erreur des algorithmes proposés a été aussi introduite.

D'autre part, nous avons montré qu'il est possible de réduire une matrice sous la forme J-Hessenberg en se basant sur les transformations de Householder symplectiques. À cet effet, nous avons proposé une version généralisée en montrant la pertinence du choix des paramètres libres pour garantir une meilleure préservation de la J-orthogonalité, aussi bien que la factorisation. De plus, deux variantes de la réduction sous la forme J-Hessenberg qui se comportent d'une manière similaire à l'algorithme *JHESS* ont été introduites.

Une particulière attention sur les questions de breakdown et near-breakdwon a fait aussi l'objet d'une investigation sérieuse dans cette thèse. Dans ce cadre, nous avons réussi à élaborer une stratégie très efficace pour remédier ce genre de problèmes.

Par ailleurs, nous avons introduit une nouvelle variante de l'algorithme *QR* appliquée aux matrices Hamiltoniennes (symétriques ou antisymétriques) qui préserve leur structure double pendant la procédure.

Finalement, et dans un autre cadre de matrices structurées, nous avons proposé un algorithme rapide pour résoudre le problème de la déconvolution d'images. Nous avons introduit un algorithme robuste via FFT (la transformation de Fourier rapide) et la matrice de Hankel, qui est basé sur le calcul approché de plus grand diviseur commun (PGCD) de deux polynômes, pour résoudre le problème de la déconvolution d'images. Plus précisément, nous concevons un algorithme spécialisé pour le calcul du PGCD de deux polynômes à deux variables. La nouvelle approche est basée sur l'algorithme rapide, de complexité quadratique $\mathcal{O}(n^2)$, pour le calcul du PGCD des polynômes unidimensionnels. La complexité de notre algorithme est $\mathcal{O}(n^2 \log(n))$ où la taille des images floues est $n \times n$. Les résultats expérimentaux avec des images synthétiquement floues ont été inclus pour illustrer l'efficacité de notre approche.

Les perspectives de cette thèse sont nombreuses. À titre d'exemple, il s'avère intéressant de réaliser une étude de la propagation des erreurs et la perte de la J-orthogonalité des algorithmes introduits.

La connexion des nouvelles méthodes symplectiques avec des applications tels que le problème du contrôle optimal, la résolution de l'équation algébrique de Riccati, la réduction du modèle, etc. fera l'objet d'un travail futur.

Il est aussi primordiale de concevoir une version généralisée pour la décomposition SR par blocs.

Il semble intéressant d'aborder la question de recherche du calcul de PGCD de plusieurs polynômes en se basant sur la matrice de Hankel généralisée (associée à plusieurs polynômes). Dans ce cadre, la connexion avec le problème de la déconvolution d'images aveugles sera cruciale.

Annexe A

An upper J-Hessenberg reduction of a matrix through symplectic Householder transformations

An upper J- Hessenberg reduction of a matrix through symplectic Householder transformations

Ahmed Salam[a,*], Haithem Ben Kahla[a,b]

[a] *University Lille Nord de France, ULCO, LMPA. CS 80699, F62228, Calais, Cedex, France.*
[b] *University of Tunis El Manar, ENIT-LAMSIN, BP 37, 1002, Tunis, Tunisia.*

Abstract

In this paper, we introduce a reduction of a matrix to a condensed form, the upper J- Hessenberg form, via elementary symplectic Householder transformations, which are rank-one modification of the identity. Features of the reduction are highlighted and a general algorithm is derived. Then, we study different possibilities to specify the general algorithm in order to built better versions. We are led to two variants numerically more stables that we compare to JHESS algorithm. JHESS as well as the new algorithms may meet a fatal breakdown, under the same condition. We show that such breakdown is not necessarily insurmountable. Thus, an efficient strategy is sketched out to cure it. The numerical tests attest the efficiency of the approach. Also, some numerical experiments for comparing the different algorithms are given.

Keywords: Indefinite inner product, structure-preserving eigenproblems, symplectic Householder transformations, SR decomposition, upper J-Hessenberg form.
2000 MSC: 65F15, 65F50

1. Introduction

Let A be a $2n \times 2n$ real matrix. The SR factorization consists in writing A as a product SR, where S is symplectic and $R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$ is such

*corresponding author
Email addresses: ahmed.salam@univ-littoral.fr (Ahmed Salam), benkahla@univ-littoral.fr (Haithem Ben Kahla)

that $R_{11},\ R_{12},\ R_{22}$ are upper triangular and R_{21} is strictly upper triangular [3, 4]. This decomposition plays an important role in structure-preserving methods for solving the eigenproblem of a class of structured matrices.

More precisely, the SR decomposition can be interpreted as the analogue of the QR decomposition [5], when instead of an Euclidean space, one considers a symplectic space : a linear space, equipped with a skew-symmetric inner product (see for example [7] and the references therein). The orthogonal group with respect to this indefinite inner product, is called the symplectic group and is unbounded (contrasting with the Euclidean case).

There are two classes of methods for computing the SR decomposition. The first lies in the Gram-Schmidt like algorithms and leads to the symplectic Gram-Schmidt (SGS) algorithms. The second class is constructed from a variety of elementary symplectic transformations. Each choice of such transformations leads to the corresponding SR decomposition. Since these elementary transformations are quite heterogeneous, the SR decomposition is considerably affected by their choice.

Results on numerical aspects of SGS-algorithms can be found for example in [7]. These algorithms and their modified versions are usually involved in structure-preserving Krylov subspace-type methods, for sparse and large structured matrices.

In the literature, the symplectic elementary transformations involved in the SR decomposition can be partitioned in two subsets. The first subset is constituted of two kind of both symplectic and orthogonal transformations introduced in [6, 12] and a third symplectic but non-orthogonal transformations, proposed in [2]. In fact, in [3], it has been shown that SR decomposition of a general matrix could not be carried out by using only the above orthogonal and symplectic transformations. An algorithm, named SRDECO, based on these three transformations was derived in [2].

From linear algebra point of view, the SR decomposition via SRDECO algorithm does not correspond to the analogue of Householder QR decomposition, since SRDECO involves transformations which are not elementary rank-one modification of the identity (transvections), see [1, 5].

In [8] a study, based on linear algebra concepts and focusing on the construction of the analogue of Householder transformations in a symplectic linear space, has been accomplished. This has led to the second subset of transformations. Such analogue transformations, which are rank-one modification of the identity are called symplectic Householder transformations. Their main features have been established, especially the mapping problem

has been solved. Then, the analogue of Householder QR decomposition in a symplectic linear space has been derived. The algorithm SRSH for computing the SR decomposition, using these symplectic Householder transformations has been then presented in details. Unlike Householder QR decomposition, the new algorithm SRSH involves free parameters and advantages may be taken from this fact. It has been demonstrated how these parameters can be determined in an optimal way providing an optimal version[9] of the algorithm (SROSH). The error analysis and computational aspects of this algorithm have been studied [10]. Also, recently, a mathematical and numerical equivalence between modified symplectic Gram-Schmidt and Householder SR algorithms (typically SRSH or SROSH) have been established in [11]. Computational aspects and numerical comparisons between SGS and SROSH have clearly showed the superiority of SROSH over SGS.

In order to build a SR-algorithm (which is a QR-like algorithm) for computing the eigenvalues and eigenvectors of a matrix [13], a reduction of the matrix to an upper J-Hessenberg form is crucial. This is due to the fact that the final algorithm we are looking for should have $O(n^3)$ as complexity.

In [2], a reduction of a general matrix to an upper J-Hessenberg form is presented, using to this aim, the three symplectic transformations of the above first subset. The algorithm, called JHESS, is based on an adaptation of SRDECO.

In this paper, we focus on the reduction of a general matrix, to an upper J-Hessenberg form, using only the symplectic Householder transformations (the second subset above). We show how this reduction can be constructed. The new algorithm, which will be called JHSH algorithm, is based on an adaptation of SRSH algorithm. A variant of JHSH, named JHOSH is then obtained by taking some optimal choice of the free parameters. The JHOSH is numerically better than JHSH. However, to enforce the accuracy in the computations, we are led to derive another variant, based in replacing when possible, each symplectic non-orthogonal transformation by another one, which is symplectic and orthogonal. This gives rise to JHMSH algorithm and its variant JHMSH2.

In this work, we restrict ourselves to the construction of such algorithms and the study of their features. Numerical aspects of the new algorithms and new insights on JHESS algorithm (breakdowns/near-breakdowns and their prediction, different strategies of curing breakdowns/near-breakdowns, ...) are very important questions and deserve a detailed study in a devoted work. Nevertheless, we give some illustrating numerical examples, showing

in particular that the algorithms JHESS, JHMSH and its variant JHMSH2 behave quite similarly. Furthermore, all of these algorithms are subject to a fatal breakdown under the same condition. We sketch out an efficient way to remedy to such breakdown. The numerical experiments attest of the efficiency of the approach.

The remainder of this paper is organized as follows. Section 2, is devoted to the necessary preliminaries. In the section 3, we introduce the method of reducing a general matrix to an upper J-Hessenberg, based only on the use of symplectic Householder transformations, which are rank-one modification of the identity. Also, we present the different variants, motivated by numerical considerations. In the section 4, we discuss the numerical aspects of the algorithms. Thus, an efficient strategy is sketched out for curing fatal breakdowns. Numerical experiments and comparisons between JHESS and JHMSH and its variant JHMSH2 are given. We conclude in the section 5.

2. Preliminaries

Let J_{2n} (or simply J) be the $2n$-by-$2n$ real matrix

$$J_{2n} = \begin{bmatrix} 0_n & I_n \\ -I_n & 0_n \end{bmatrix}, \tag{1}$$

where 0_n and I_n stand respectively for n-by-n null and identity matrices. The linear space $\mathbb{R}^{2n}$ with the indefinite skew-symmetric inner product

$$(x, y)_J = x^T J y \tag{2}$$

is called symplectic. For $x,\ y \in \mathbb{R}^{2n}$, the orthogonality $x \perp' y$ stands for $(x, y)_J = 0$. The symplectic adjoint x^J of a vector x, is defined by

$$x^J = x^T J. \tag{3}$$

The symplectic adjoint of $M \in \mathbb{R}^{2n\times 2k}$ is defined by

$$M^J = J_{2k}^T M^T J_{2n}. \tag{4}$$

A matrix $S \in \mathbb{R}^{2n\times 2k}$ is called symplectic if

$$S^J S = I_{2k}. \tag{5}$$

The symplectic group (multiplicative group of square symplectic matrices) is denoted $\mathbb{S}$. A transformation T given by

$$T = I + cvv^J \text{ where } c \in \mathbb{R}, \quad v \in \mathbb{R}^\nu \quad (\text{with } \nu \text{ even}), \tag{6}$$

is called symplectic Householder transformation [8]. It satisfies

$$T^{-1} = T^J = I - cvv^J. \tag{7}$$

The vector v is called the direction of T.

For $x,\ y \in \mathbb{R}^{2n}$, there exists a symplectic Householder transformation T such that $Tx = y$ if $x = y$ or $y^J x \neq 0$. When $y^J x \neq 0$, T is given by

$$T = I + \frac{1}{y^J x}(y - x)(y - x)^J.$$

Moreover, each non null vector x can be mapped onto any non null vector y by a product of at most two symplectic Householder transformations [8]. Symplectic Householder transformations are rotations, i.e. $det(T) = 1$ and the symplectic group $\mathbb{S}$ is generated by symplectic Householder transformations.

We recall that a matrix $H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$, is upper J-Hessenberg when H_{11}, H_{21}, H_{22} are upper triangular and H_{12} is upper Hessenberg. H is called unreduced when H_{21} is nonsingular and the Hessenberg H_{12} is unreduced, i.e. the entries of the subdiagonal are all nonzero.

3. Upper *J*-Hessenberg reduction via symplectic Householder transformations

3.1. Toward the algorithm

Let $\{e_1, \ldots, e_{2n}\}$ be the canonical basis of $\mathbb{R}^{2n}$ and $a \in \mathbb{R}^{2n}$ be a given vector. We seek for symplectic Householder transformations T_1 and T_2 such that

$$T_1 a = \rho e_1, \tag{8}$$

for certain $\rho \in \mathbb{R}$ and

$$T_2 e_1 = e_1,\ T_2 a = \mu e_1 + \nu e_{n+1}, \tag{9}$$

for certain $\mu, \nu \in \mathbb{R}$. The fact that T_2 is a symplectic isometry yields the necessary condition

$$(T_2 a)^J (T_2 e_1) = a^J e_1, \tag{10}$$

which implies $\nu = a_{n+1}$ (the $n+1$th component of a) and μ is arbitrary. We get

Theorem 1. *Let $\rho \neq 0$, μ be arbitrary scalars and $\nu = a_{n+1}$. Setting*

$$c_1 = -\frac{1}{\rho a^J e_1},\ v_1 = \rho e_1 - a,\ c_2 = -\frac{1}{a^J(\mu e_1 + \nu e_{n+1})},\ v_2 = \mu e_1 + \nu e_{n+1} - a,$$

then

$$T_1 = I + c_1 v_1 v_1^J (\text{ respectively } T_2 = I + c_2 v_2 v_2^J) \text{ satisfy (8) (respectively(9)).} \tag{11}$$

Remark 1. Since the $n+1$th component of v_2 is zero, T_2 keeps the $n+1$th component of $T_2 x$ unchanged, for any $x \in \mathbb{R}^{2n}$. More on the properties of such transformations T_1 or T_2 can be found in [9, 10].

We also need the following

Theorem 2. *Let $v \in \mathbb{R}^{2n}$, with the partition $v = [0^T, u^T, 0^T, w^T]^T$, where $[u, w] \in \mathbb{R}^{(n-i)\times 2}$, for a given integer $1 \leq i \leq n-1$ and set $\tilde{v} = [u^T, w^T]^T$. Consider the symplectic transformations $T = I + cvv^J$ and $\tilde{T} = I + c\tilde{v}\tilde{v}^J$. We have*
$\forall \alpha \in \mathbb{R}^i,\ \forall \beta \in \mathbb{R}^i,\ \forall x \in \mathbb{R}^{n-i},\ \forall y \in \mathbb{R}^{n-i}$,

$$T[\alpha^T, x^T, \beta^T, y^T]^T = [\alpha^T, x'^T, \beta^T, y'^T]^T,\ \text{ with } [x'^T, y'^T]^T = \tilde{T}[x^T, y^T]^T.$$

Proof. We have $v^J[\alpha^T, x^T, \beta^T, y^T]^T = u^T y - w^T x = [u^T w^T] J [x^T y^T]^T$. Then $T[\alpha^T, x^T, \beta^T, y^T]^T = [\alpha^T, x^T, \beta^T, y^T]^T + c[0^T, u^T, 0^T, w^T]^T [u^T w^T] J [x^T y^T]^T$. We check easily $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + c \begin{bmatrix} u \\ w \end{bmatrix} [u^T w^T] J \begin{bmatrix} x \\ y \end{bmatrix} = \tilde{T} \begin{bmatrix} x \\ y \end{bmatrix}$, and
$T[\alpha^T, 0^T, \beta^T, 0^T]^T = [\alpha^T, 0^T, \beta^T, 0^T]^T$.

Note that the Theorem 2 remains valid if one takes T^J instead of T. This result, with Theorem 1, constitute the main tool on which the SR factorization (based on symplectic Householder transformations) is constructed. We will adapt this tool for reducing a general matrix to an upper J-Hessenberg form, based on these symplectic Householder transformations.

3.2. The J-Hessenberg reduction : the JHSH algorithm

We explain here the steps of the algorithm by illustrating the general pattern. Let $A \in \mathbb{R}^{2n\times 2n}$ be a given matrix and set $A^{(0)} = A$. We will use the notation $A_{(i_1:i_2,j_1:j_2)}$, $A_{(i_1:i_2,:)}$, $A_{(:,j_1:j_2)}$ to denote respectively the submatrix obtained from the matrix A by deleting all rows and columns except rows i_1 until i_2 and columns j_1 until j_2, by deleting all rows except rows i_1 until i_2, by deleting all columns except columns j_1 until j_2.

1. The first step of the algorithm relies in determining a symplectic Householder transformation H_1 (i.e. $c_1 \in \mathbb{R}$ and $v_1 \in \mathbb{R}^{2n}$), with $H_1e_1 = e_1$, to zero out entries 2 through n and entries $n+2$ through $2n$ of the first column of $A^{(0)}$. The vector e_1 stands for the first canonical vector of $\mathbb{R}^{2n}$. The transformation H_1 corresponds to the transformation T_2, given in Theorem 1. Set v_1 the direction vector of H_1. Since $H_1e_1 = e_1$, we obtain $v_1^Je_1 = v_1^TJe_1 = 0$. Thus the $n+1$th component of v_1 is zero. It follows that for any vector x, the $n+1$th component of H_1x remains unchanged. The direction v_1 of H_1 is given by $v_1 = A^{(1)}_{(1,1)}e_1 + A^{(0)}_{(n+1,1)}e_{n+1} - A^{(0)}_{(:,1)}e_{n+1}$, where $A^{(1)}_{(1,1)}$ is an arbitrary given scalar. Notice that we have also $H_1^Je_1 = e_1$, and hence the first column of H_1 and H_1^J is e_1. Thus, multiplying $A^{(0)}$ on the left by H_1 leaves unchanged the $n+1$th row and creates the desired zeros in the first column. We get

$$A'^{(1)} = H_1A^{(0)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A'^{(1)}_{(1,2:n)} & A'^{(1)}_{(1,n+1:2n)} \\ 0 & A'^{(1)}_{(2:n,2:n)} & A'^{(1)}_{(2:n,n+1:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(0)}_{(n+1,2:n)} & A^{(0)}_{(n+1,n+1:2n)} \\ 0 & A'^{(1)}_{(n+2:2n,2:n)} & A'^{(1)}_{(n+2:2n,n+1:2n)} \end{bmatrix}.$$

The step involves the free parameter $A^{(1)}_{(1,1)}$.

Multiplying $H_1A^{(0)}$ on the right by H_1^J leaves the first column of $H_1A^{(0)}H_1^J$ unchanged, and we obtain

$$A^{(1)} = H_1A^{(0)}H_1^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1:2n)} \\ 0 & A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1:2n)} \\ 0 & A^{(1)}_{(n+2:2n,2:n)} & A^{(1)}_{(n+2:2n,n+1:2n)} \end{bmatrix}.$$

The next step consists in choosing a symplectic Householder H_2 to zero out the entries 3 through n, the entries $n+2$ through $2n$ of the $n+1$th

column of $A^{(1)}$. To do this, let $\tilde{A}^{(1)} = \begin{bmatrix} A^{(1)}_{(2:n,2:n)} & A^{(1)}_{(2:n,n+1:2n)} \\ A^{(1)}_{n+2:2n,2:n} & A^{(1)}_{n+2:2n,n+1:2n} \end{bmatrix}$ be the the matrix obtained from $A^{(1)}$ by deleting the first column and the first and the $n+1$th rows. And let $A^{(2)}_{(2,n+1)} \neq 0$ be an arbitrary given scalar. We apply $\tilde{H}_2 = I_{2n-2} + c_2 \tilde{v}_2 \tilde{v}_2^J$ given by Theorem 1, with $\tilde{v}_2 = \begin{bmatrix} u_2 \\ \hline w_2 \end{bmatrix} = A^{(2)}_{(2,n+1)} e_1 - \tilde{A}^{(1)}(:,n) \in \mathbb{R}^{2n-2}$, $u_2 \in \mathbb{R}^{n-1}$, $w_2 \in \mathbb{R}^{n-1}$, where e_1 stands for the first canonical vector of $\mathbb{R}^{2n-2}$. We obtain

$$\tilde{A}'^{(2)} = \tilde{H}_2 \tilde{A}^{(1)} = \begin{bmatrix} A'^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A'^{(2)}_{(2,n+2:2n)} \\ A'^{(2)}_{(3:n,2:n)} & 0 & A'^{(2)}_{(3:n,n+2:2n)} \\ A'^{(2)}_{(n+2:2n,2:n)} & 0 & A'^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

The transformation $\tilde{H}_2$ corresponds to the choice T_1 in Theorem 1. Setting $H_2 = I_{2n} + c_2 v_2 {v_2}^J$, with $v_2 = \begin{bmatrix} 0 \\ \hline u_2 \\ \hline 0 \\ \hline w_2 \end{bmatrix} \in \mathbb{R}^{2n}$ then H_2 is a symplectic Householder transformation. Using Theorem 2, we get

$$A'^{(2)} = H_2 A^{(1)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(1)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(1)}_{(1,n+2:2n)} \\ 0 & A'^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A'^{(2)}_{(2,n+2:2n)} \\ 0 & A'^{(2)}_{(3:n,2:n)} & 0 & A'^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(1)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(1)}_{(n+1,n+2:2n)} \\ 0 & A'^{(2)}_{(n+2:2n,2:n)} & 0 & A'^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

H_2 leaves the first and the $n+1$ th rows of $H_2 A^{(1)}$ unchanged. It leaves the first column of $H_2 A^{(1)}$ unchanged, and creates the desired zeros in the column $n+1$.

The multiplication of $H_2 A^{(1)}$ on the right by H_2^J leaves the first and the $n+1$th columns of $H_2 A^{(1)} H_2^J$ unchanged. We obtain

$$A^{(2)} = H_2 A^{(1)} H_2^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2:n)} & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2:2n)} \\ 0 & A^{(2)}_{(2,2:n)} & A^{(2)}_{(2,n+1)} & A^{(2)}_{(2,n+2:2n)} \\ 0 & A^{(2)}_{(3:n,2:n)} & 0 & A^{(2)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2:n)} & A^{(1)}_{(n+1,n+1)} & A^{(2)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2:2n,2:n)} & 0 & A^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}.$$

It is worth noting that $H_2 e_1 = e_1$ and $H_2 e_{n+1} = e_{n+1}$. Thus the first column (respectively the $n+1$th column) of H_2 and H_2^J is e_1 (respectively e_{n+1}).

In the next step, we want to zero out the entries 3 through n and $n+3$ through $2n$ of the second column of $A^{(2)}$ and the entries 4 through n and $n+3$ through $2n$ of the column $n+2$ of $A^{(2)}$. Let $\tilde{A}^{(2)}$ be the matrix obtained from $A^{(2)}$ by deleting the first, the $n+1$th rows, and the corresponding columns, ie. $\tilde{A}^{(2)} = \begin{bmatrix} A^{(2)}_{(2:n,2:n)} & A^{(2)}_{(2:n,n+2:2n)} \\ A^{(2)}_{(n+2:2n,2:n)} & A^{(2)}_{(n+2:2n,n+2:2n)} \end{bmatrix}$.

2. We apply now exactly the same two steps of **1.**, to the new size reduced matrix $\tilde{A}^{(2)}$. In other words, we choose a symplectic Householder transformation $\tilde{H}_3$, which means to compute a vector $\tilde{v}_3 = [u_3^T, w_3^T]^T$ with $u_3 \in \mathbb{R}^{n-1}$, $w_3 \in \mathbb{R}^{n-1}$ and a real c_3 such that $\tilde{H}_3 = I + c_3 \tilde{v}_3 \tilde{v}_3^J$ zero out the entries 2 through $n-1$ and the entries $n+1$ through $2n-2$ of the first column of $\tilde{A}^{(2)}$ with $\tilde{H}_3 e_1 = e_1 \in \mathbb{R}^{2n-2}$. Here e_1 stands for the first canonical vector of $\mathbb{R}^{2n-2}$. The transformation $\tilde{H}_3$ corresponds to the transformation T_2, in Theorem 1. Let e_n denote the nth canonical vector of $\mathbb{R}^{2n-2}$. The direction vector $\tilde{v}_3$ of $\tilde{H}_3$ is given by $\tilde{v}_3 = A^{(3)}_{(2,2)} e_1 + \tilde{A}^{(2)}(n,1) e_n - \tilde{A}^{(2)}(:,1)$, where $A^{(3)}_{(2,2)}$ is an arbitrary non zero scalar. $\tilde{H}_3$ leaves unchanged the nth row of $\tilde{H}_3 \tilde{A}^{(2)}$. We get

$$\tilde{A}'^{(3)} = \tilde{H}_3 \tilde{A}^{(2)} = \begin{bmatrix} A^{(3)}_{(2,2)} & A'^{(3)}_{(2,3:n)} & A'^{(3)}_{(2,n+2:2n)} \\ 0 & A'^{(3)}_{(3:n,3:n)} & A'^{(3)}_{(3:n,n+2:2n)} \\ A^{(2)}_{(n+2,2)} & A^{(2)}_{(n+2,3:n)} & A^{(2)}_{(n+2,n+2:2n)} \\ 0 & A'^{(3)}_{(n+3:2n,3:n)} & A'^{(3)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

Remark that the nth component of $\tilde{v}_3$ is zero. Take now $v_3 = [0\ u_3^T | 0\ w_3^T]^T$ and set $H_3 = I + c_3 v_3 v_3^J$. Then H_3 is obviously a symplectic Householder transformation of order $2n$. The components 1, $n+1$ and $n+2$ of v_3 are equal

to zero. Thus H_3 leaves the rows 1, $n+1$ and $n+2$ of $H_3A^{(2)}$ unchanged and satisfy $H_3(e_1) = e_1$, $H_3e_2 = e_2$ and $H_3e_{n+1} = e_{n+1}$. Thus

H_3 leaves the first and the $n+1$th columns of $H_3A^{(2)}$ unchanged and zero out the entries 3 through n and the entries $n+3$ through $2n$ of the second column.

We have

$$A'^{(3)} = H_3A^{(2)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(2)}_{(1,3:n)} & A^{(1)}_{(1,n+1)} & A^{(2)}_{(1,n+2:2n)} \\ 0 & A^{(3)}_{(2,2)} & A'^{(3)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A'^{(3)}_{(2,n+2:2n)} \\ 0 & 0 & A'^{(3)}_{(3:n,3:n)} & 0 & A'^{(3)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2)} & A^{(2)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(2)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2,2)} & A^{(2)}_{(n+2,3:n)} & 0 & A^{(2)}_{(n+2,n+2:2n)} \\ 0 & 0 & A'^{(3)}_{(n+3:2n,3:n)} & 0 & A'^{(3)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

The transformation H_3^J leaves the column 1, 2 and $n+1$ of $H_3A^{(2)}H_3^J$ unchanged since $H_3^J(e_1) = e_1$, $H_3^Je_2 = e_2$ and $H_3^Je_{n+1} = e_{n+1}$. We get

$$A^{(3)} = H_3A^{(2)}H_3^J = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(3)}_{(1,3:n)} & A^{(1)}_{(1,n+1)} & A^{(3)}_{(1,n+2:2n)} \\ 0 & A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2:2n)} \\ 0 & 0 & A^{(3)}_{(3:n,3:n)} & 0 & A^{(3)}_{(3:n,n+2:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2)} & A^{(3)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(3)}_{(n+1,n+2:2n)} \\ 0 & A^{(2)}_{(n+2,2)} & A^{(3)}_{(n+2,3:n)} & 0 & A^{(3)}_{(n+2,n+2:2n)} \\ 0 & 0 & A^{(3)}_{(n+3:2n,3:n)} & 0 & A^{(3)}_{(n+3:2n,n+2:2n)} \end{bmatrix}.$$

Now, deleting the rows 1, 2, $n+1$, $n+2$ and the columns 1, 2, $n+1$ of $A^{(3)}$ and setting $\tilde{A}^{(3)} = \begin{bmatrix} A^{(3)}_{(3:n,3:n)} & A^{(3)}_{(3:n,n+2:2n)} \\ A^{(3)}_{(n+3:2n,3:n)} & A^{(3)}_{(n+3:2n,n+2:2n)} \end{bmatrix}$, we find $c_4 \in \mathbb{R}$ and $\tilde{v}_4 = \begin{bmatrix} u_4 \\ \hline w_4 \end{bmatrix}$, with $u_4 \in \mathbb{R}^{n-2}$ and $w_4 \in \mathbb{R}^{n-2}$ such that the action of $\tilde{H}_4 = I + c_4\tilde{v}_4\tilde{v}_4{}^J$ gives

$$\tilde{A}'^{(4)} = \tilde{H}_4\tilde{A}^{(3)} = \begin{bmatrix} A'^{(4)}_{(3,3:n)} & A^{(4)}_{(3,n+2)} & A'^{(4)}_{(3,n+3:2n)} \\ A'^{(4)}_{(4:n,3:n)} & 0 & A'^{(4)}_{(4:n,n+3:2n)} \\ A'^{(4)}_{(n+3:2n,3:n)} & 0 & A'^{(4)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

The coefficient $A^{(4)}_{(3,n+2)}$ is an arbitrary chosen scalar. Taking $v_4 = [0\ 0\ u_4^T|0\ 0\ w_4^T]^T$ then the transformation $H_4 = I + c_4 v_4 v_4^J$ leaves unchanged the rows 1, 2, $n+1$, $n+2$ and columns 1, 2, and $n+1$ of $A'^{(4)} = H_4 A^{(3)}$ and creates the desired zeros in the column $n+2$. We obtain

$$A'^{(4)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(3)}_{(1,3:n)} & A^{(1)}_{(1,n+1)} & A^{(3)}_{(1,n+2)} & A^{(3)}_{(1,n+3:2n)} \\ 0 & A^{(3)}_{(2,2)} & A^{(3)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(3)}_{(2,n+3:2n)} \\ 0 & 0 & A'^{(4)}_{(3,3:n)} & 0 & A^{(4)}_{(3,n+2)} & A'^{(4)}_{(3,n+3:2n)} \\ 0 & 0 & A'^{(4)}_{(4:n,3:n)} & 0 & 0 & A'^{(4)}_{(4:n,n+3:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2)} & A^{(3)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(3)}_{(n+1,n+2)} & A^{(3)}_{(n+1,n+3:2n)} \\ 0 & A^{(2)}_{(n+2,2)} & A^{(3)}_{(n+2,3:n)} & 0 & A^{(3)}_{(n+2,n+2)} & A^{(3)}_{(n+2,n+3:2n)} \\ 0 & 0 & A'^{(4)}_{(n+3:2n,3:n)} & 0 & 0 & A'^{(4)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

H_4^J leaves unchanged the first, the second, the $n+1$, $n+2$ columns of $A^{(4)} = H_4 A^{(3)} H_4^J$ since $H_4^J(e_i) = e_i$ for $i = 1,\ 2\ , n+1,\ n+2$. Hence, we get

$$A^{(4)} = \begin{bmatrix} A^{(1)}_{(1,1)} & A^{(2)}_{(1,2)} & A^{(4)}_{(1,3:n)} & A^{(1)}_{(1,n+1)} & A^{(3)}_{(1,n+2)} & A^{(4)}_{(1,n+3:2n)} \\ 0 & A^{(3)}_{(2,2)} & A^{(4)}_{(2,3:n)} & A^{(2)}_{(2,n+1)} & A^{(3)}_{(2,n+2)} & A^{(4)}_{(2,n+3:2n)} \\ 0 & 0 & A^{(4)}_{(3,3:n)} & 0 & A^{(4)}_{(3,n+2)} & A^{(4)}_{(3,n+3:2n)} \\ 0 & 0 & A^{(4)}_{(4:n,3:n)} & 0 & 0 & A^{(4)}_{(4:n,n+3:2n)} \\ A^{(0)}_{(n+1,1)} & A^{(2)}_{(n+1,2)} & A^{(4)}_{(n+1,3:n)} & A^{(1)}_{(n+1,n+1)} & A^{(3)}_{(n+1,n+2)} & A^{(4)}_{(n+1,n+3:2n)} \\ 0 & A^{(2)}_{(n+2,2)} & A^{(4)}_{(n+2,3:n)} & 0 & A^{(3)}_{(n+2,n+2)} & A^{(4)}_{(n+2,n+3:2n)} \\ 0 & 0 & A^{(4)}_{(n+3:2n,3:n)} & 0 & 0 & A^{(4)}_{(n+3:2n,n+3:2n)} \end{bmatrix}.$$

3. The jth step is now clear. It involves two sub-steps. The first consists in finding H_{2j-1} , i.e. the scalar c_{2j-1} and the vector v_{2j-1} such that $H_{2j-1} = I + c_{2j-1} v_{2j-1} v_{2j-1}^J$ leaves the rows $1, \ldots, j-1$, the rows $n+1, \ldots, n+j$, the columns $1, \ldots, j-1$, and the columns $n+1, \ldots, n+j-1$ of $H_{2j-1} A^{(2j-2)}$ unchanged and zero out the entries $j+1$ through n and the entries $n+j+1$ through $2n$ of the jth column. The vector $v_{2j-1} \in \mathbb{R}^{2n}$ has the structure $v_{2j-1} = [0^T, u_{2j-1}^T, 0^T, w_{2j-1}^T]^T$, with $u_{2j-1} \in \mathbb{R}^{n-j+1}$, $w_{2j-1} \in \mathbb{R}^{n-j+1}$. The first component of w_{2j-1} is zero.Thus $H_{2j-1} e_i = e_i$ for $i = 1, \ldots, j$ and for $i = n+1, \ldots, n+j-1$. The jth column $H_{2j-1} A^{(2j-2)}(:, j)$ is transformed as

follows

$$H_{2j-1}A^{(2j-2)}(:,j) = \begin{bmatrix} A^{(2j-2)}(1:j-1,j) \\ A^{(2j-1)}(j,j) \\ 0 \\ A^{(2j-2)}(n+1:n+j,j) \\ 0 \end{bmatrix} \begin{matrix} \{j-1\} \\ \{1\} \\ \{n-j\} \\ \{j\} \\ \{n-j\} \end{matrix} .$$

The entry $A^{(2j-1)}(j,j)$ is a free parameter.

The multiplication of $H_{2j-1}A^{(2j-2)}$ on the right by H^J_{2j-1} leaves the columns $1,\ldots,j$, and the columns $n+1,\ldots,n+j-1$, of $H_{2j-1}A^{(2j-2)}H^J_{2j-1}$ unchanged. The coefficient c_{2j-1}, the vector v_{2j-1} and hence the symplectic transformation H_{2j-1} are simply and explicitly given by Theorem 1. The matrix $A^{(2j-1)} = H_{2j-1}A^{(2j-2)}H^J_{2j-1}$ has the desired form. Let us set $\tilde{H}_{2j-1} = I + c_{2j-1}\tilde{v}_{2j-1}\tilde{v}^J_{2j-1}$, $\tilde{v}_{2j-1} = [u^T_{2j-1}, w^T_{2j-1}]^T$, where $[u_{2j-1}, w_{2j-1}] \in \mathbb{R}^{\alpha_j \times 2}$, with $\alpha_j = n-j+1$ and $\tilde{A}^{(2j-2)}(:,j)$ the jth column of $\tilde{A}^{(2j-2)}$ obtained from $A^{(2j-2)}(:,j)$ by deleting the rows $1,\ldots,j-1$ and rows $n+1,\ldots,n+j-1$. We obviously obtain $\tilde{H}_{2j-1}\tilde{A}^{(2j-2)}(:,j) = A^{(2j-1)}(j,j)e_1 + A^{(2j-2)}(n+j,j)e_{\alpha_j+1}$. Here e_1 and e_{α_j+1} denote the first and the α_j+1th canonical vectors of $\mathbb{R}^{2\alpha_j}$.

In a similar way, the second sub-step consists in finding H_{2j} , i.e. the scalar c_{2j} and the vector v_{2j} such that $H_{2j} = I + c_{2j}v_{2j}v^J_{2j}$ leaves the rows $1,\ldots,j$, the rows $n+1,\ldots,n+j$, the columns $1,\ldots,j$, and the columns $n+1,\ldots,n+j-1$ of $H_{2j}A^{(2j-1)}$ unchanged and zero out the entries $j+2$ through n and the entries $n+j+1$ through $2n$ of the $n+j$th column. The vector $v_{2j} \in \mathbb{R}^{2n}$ has the structure $v_{2j} = [0^T, u^T_{2j}, 0^T, w^T_{2j}]^T$, with $u_{2j} \in \mathbb{R}^{n-j}$, $w_{2j} \in \mathbb{R}^{n-j}$. Thus $H_{2j}e_i = e_i$ for $i = 1,\ldots,j$ and for $i = n+1,\ldots,n+j$. The $n+j$th column of $H_{2j}A^{(2j-1)}(:,n+j)$ is transformed as follows

$$H_{2j}A^{(2j-1)}(:,n+j) = \begin{bmatrix} A^{(2j-1)}(1:j,n+j) \\ A^{(2j)}(j+1,n+j) \\ 0 \\ A^{(2j-1)}(n+1:n+j,n+j) \\ 0 \end{bmatrix} \begin{matrix} \{j\} \\ \{1\} \\ \{n-j-1\} \\ \{j\} \\ \{n-j\} \end{matrix} .$$

The entry $A^{(2j)}(j+1,n+j)$ is a free parameter.

The multiplication of $H_{2j}A^{(2j-1))}$ on the right by H^J_{2j} leaves the columns $1,\ldots,j$, and the columns $n+1,\ldots,n+j$, of $H_{2j}A^{(2j-1)}H^J_{2j}$ unchanged. The coefficient c_{2j}, the vector v_{2j} and hence the symplectic transformation H_{2j}

are explicitly given by Theorem 1. The matrix $A^{(2j)} = H_{2j}A^{(2j-1)}H_{2j}^J$ has the desired form.

Let us set $\tilde{H}_{2j} = I + c_{2j}\tilde{v}_{2j}\tilde{v}_{2j}^J$, with $\tilde{v}_{2j} = [u_{2j}^T, w_{2j}^T]^T$, where $[u_{2j}, w_{2j}] \in \mathbb{R}^{\beta_j \times 2}$, $\beta_j = n-j$ and $\tilde{A}^{(2j-1)}(:, n+j)$ the $n+j$th column of $\tilde{A}^{(2j-1)}$ obtained from $A^{(2j-1)}(:, n+j)$ by deleting the rows $1, \ldots, j$ and rows $n+1, \ldots, n+j$. We obviously obtain $\tilde{H}_{2j}\tilde{A}^{(2j-1)}(:, n+j) = A^{(2j)}(j+1, n+j)e_1$. Here e_1 denotes the first canonical vector of $\mathbb{R}^{2\beta_j}$.

Thus, it is worth noting that each step j involves two free parameters $A^{(2j-1)}(j, j)$ and $A^{(2j)}(j+1, n+j)$, and that these parameters are located as highlighted above, in the corresponding symplectic Householder transformations H_{2j-1} and H_{2j} (or equivalently $\tilde{H}_{2j-1}$ and $\tilde{H}_{2j}$).

At the last step (the $n-1$th step), we obtain $H_{2n-2}\ldots H_2H_1A(H_{2n-2}\ldots H_2H_1)^J = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} = H \in \mathbb{R}^{2n\times 2n}$, with H_{11}, H_{21}, H_{22} upper triangular and H_{12} upper Hessenberg. We get $A = S^JHS$ with $S = H_{2n-2}\ldots H_1$. The entries of the diagonal of H_{11} are the free parameters $A^{(2j-1)}(j, j)$, ie. $H_{11}(j, j) = A^{(2j-1)}(j, j)$ for $j = 1, \ldots, n$. Also, The entries of the sub-diagonal of H_{12} are the free parameters $A^{(2j)}(j+1, n+j)$, ie. $H_{12}(j+1, j) = A^{(2j)}(j+1, n+j)$ for $j = 1, \ldots, n-1$. We propose here the algorithm in its general version, written in pseudo Matlab code, for computing the reduction of a matrix to the upper J-Hessenberg form, via symplectic Householder transformations (JHSH algorithm).

Algorithm 3. *function [S,H]=JHSH(A)*
twon $= size(A(:, 1))$; $n = twon/2$; $S = eye(twon)$;
for $j = 1 : n-1$
$J = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)]$;
$ro = [j : n, n+j : 2n]$; $co = [j : n, n+j : 2n]$;
$[c, v] = sh2(A(ro, j))$;
% Updating A :
$A(ro, co) = A(ro, co) + c * v * (v' * J * A(ro, co))$;
$A(:, co) = A(:, co) - (A(:, co) * (c * v)) * v' * J$;
% Updating S (if needed):
$S(ro, 2 : end) = S(ro, 2 : end) + c * (v * v') * J * S(ro, 2 : end)$;
$J = [zeros(n-j), eye(n-j); -eye(n-j), zeros(n-j)]$;
$ro = [j+1 : n, n+j+1 : 2n]$;
$[c, v] = sh1(A(ro, n+j))$;
%Updating A:

$A(ro, co) = A(ro, co) + c * v * (v' * J * A(ro, co))$;
$A(:, co) = A(:, co) - (A(:, co) * (c * v)) * v' * J$;
%Updating S (if needed):
$S(ro, 2 : end) = S(ro, 2 : end) + c * (v * v') * J * S(ro, 2 : end)$;
end
end

Algorithm 4. *function [c, v] = sh1(a)*
%compute c and v such that $T_1 a = \rho e_1$,
% $a = [a_1, \ldots, a_{2n}]$.
%ρ *is a free parameter, and* $T_1 = (eye(twon) + c * v * v' * J)$;
$twon = length(a)$; $n = twon/2$;
$J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
choose ρ; $aux = a_1 - \rho$;
if $aux == 0$
$c = 0$; $v = zeros(twon, 1)$; $\%T = eye(twon)$;
elseif $a_{n+1} == 0$
display('division by zero');
return
else
$v = \dfrac{a}{aux}$; $c = \dfrac{aux^2}{\rho \times a_{n+1}}$; $v(1) = 1$;
end
end

Algorithm 5. *function [c, v] = sh2(a)*
%compute c and v such that $T_2 e_1 = e_1$, *and* $T_2 a = \mu e_1 + \nu e_{n+1}$,
%μ *is a free parameter, and* $T_2 = (eye(twon) + c * v * v' * J)$;
% $a = [a_1, \ldots, a_{2n}]$.
$twon = length(a)$; $n = twon/2$;
$J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
if $n == 1$
$v = zeros(twon, 1)$; $c = 0$; $\%T = eye(twon)$;
else
choose μ;
$\nu = a_{n+1}$;
if $\nu == 0$

display('division by zero')
return
else
$v = \mu e_1 + \nu e_{n+1} - a,\ c = \dfrac{1}{a_{n+1}(a_1 - \mu)}$;
end
end

3.3. JHOSH, JHMSH algorithms

From an linear algebra point of view, JHSH is the analogue in the symplectic case, of the algorithm performing the Hessenberg reduction of a matrix via Householder transformations in the Euclidean case. Recall that JHSH involves two free parameters at each step. The question is then how these free parameters can be chosen? In the sequel, we show how one can take benefit from these free parameters in some optimal way. In order to get an algorithm numerically stable as possible, the free parameters are chosen so that the symplectic Householder transformations used in the reduction have minimal norm-2 condition number. The choice of such parameters is as follows [9] :

Theorem 6. *Let $a = [a_1, \ldots, a_{2n}] \in \mathbb{R}^{2n}$ be a given vector and $\{e_1, \ldots, e_{2n}\}$ be the canonical basis of $\mathbb{R}^{2n}$. Take $\rho = sign(a_1)\left\|a\right\|_2$, $\mu = a_1 \pm \xi$ and $\nu = a_{n+1}$, with $\xi = \sqrt{\sum_{i=2, i\neq n+1}^{2n} a_i^2}$. Setting*

$$c_1 = -\frac{1}{\rho a^J e_1},\ v_1 = \rho e_1 - a,\ c_2 = -\frac{1}{a^J(\mu e_1 + \nu e_{n+1})},\ v_2 = \mu e_1 + \nu e_{n+1} - a,$$

then

$$T_1 = I + c_1 v_1 v_1^J (\text{ respectively } T_2 = I + c_2 v_2 v_2^J) \text{ satisfy (8) (respectively(9))}, \tag{12}$$

with T_1 (respectively T_2) has the minimal norm-2 condition number.

Proof. See [9].

For these choices of the free parameters, we refer to T_1 (respectively T_2) as the first optimal symplectic Householder (osh1) transformation (respectively the second optimal symplectic Householder osh2) transformation. This optimal version of JHSH is referred to as JHOSH algorithm and is given as follows :

Algorithm 7. *function [S,H]=JHOSH(A)*
replace in the body of JHSH the sh1 by osh1 and sh2 by osh2.
end.

The pseudo code Matlab of $osh1$ and $osh2$ is a follows

Algorithm 8. *function [c, v] = osh1(a)*
$twon = length(a);\ n = twon/2;$
$J = [zeros(n), eye(n); -eye(n), zeros(n)];$
$\rho = sign(a(1)) * \|a\|_2;\ aux = a(1) - \rho;$
if $aux == 0$
$c = 0;\ v = zeros(twon, 1);\ \%T = eye(twon);$
elseif $a_{n+1} == 0$
display('division by zero');
return
else
$v = \dfrac{a}{aux};\ c = \dfrac{aux^2}{\rho * a_{n+1}};\ v(1) = 1;$
$\%T = (eye(twon) + c * v * v' * J);$
end
end

Algorithm 9. *function [c, v] = osh2(a)*
$twon = length(u);\ n = twon/2;$
$J = [zeros(n), eye(n); -eye(n), zeros(n)];$
if $n == 1$
$v = zeros(twon, 1);\ c = 0;\ \%T = eye(twon);$
else
$I = [2 : n, n + 2 : twon];\ \xi = norm(a(I));$
if $\xi == 0$
$v = zeros(twon, 1);\ c = 0;\ \ \%T = eye(twon);$
else
$\nu = a_{n+1};$
if $\nu == 0$
display('division by zero')
return
else
$v = -a/\xi;\ v(1) = 1;\ v(n + 1) = 0;\ c = \xi/\nu;$

$\%T = (eye(twon) + c * v * v' * J);$
end
end
end
end

We have seen that the symplectic Householder transformations used in JHOSH algorithm have minimal norm-2 condition number, and thus numerically, JHOSH presents a significant advantage over JHSH. However, all these symplectic Householder transformations are not orthogonal. It is well known that it is not possible to handle a SR decomposition using only transformations which are both symplectic and orthogonal (see [3]). Nevertheless, we will show that half of them (all the transformations H_{2j} above) may be replaced by specified transformations which are both orthogonal and symplectic. Furthermore, we will show that the two type of orthogonal and symplectic transformations, introduced by Paige et al. [6, 12] can be used to replace the symplectic transformations H_{2j}, to zero desired components of a vector. The first type is

$$H(k,w) = \begin{pmatrix} \mathrm{diag}(I_{k-1}, P) & 0 \\ 0 & \mathrm{diag}(I_{k-1}, P) \end{pmatrix}, \tag{13}$$

where

$$P = I - 2ww^T/w^Tw, \quad w \in \mathbb{R}^{n-k+1}.$$

The transformation $H(k,w)$ is just a direct sum of two "ordinary" $n-$by$-n$ Householder matrices [14]. We refer to $H(k,w)$ as Van Loan's Householder transformations. The second type is

$$J(k,c,s) = \begin{pmatrix} C & S \\ -S & C \end{pmatrix}, \tag{14}$$

where $c^2 + s^2 = 1$, and

$$C = diag(I_{k-1}, c, I_{n-k}),$$

$$S = diag(0_{k-1}, s, 0_{n-k}).$$

$J(k,c,s)$ is a Givens transformation, which is an "ordinary" $2n$-by-$2n$ Givens rotation that rotates in planes $(k, k+n)$ [14]. We refer to $J(k,c,s)$ as Van Loan's Givens rotation. Van Loan's Householder and Givens transformations

are both orthogonal and symplectic. It is worth noting that for $i \neq k$ and $i \neq n+k$, we have $J(k,c,s)e_i = e_i$. Also, we have $J(k,c,s)e_k = ce_k - se_{n+k}$ and $J(k,c,s)e_{n+k} = se_k + ce_{n+k}$. Thus, $J(k,c,s)$ leaves unchanged all the rows of $J(k,c,s)a$ except rows k and $n+k$. It is obvious also that $H(k,w)e_i = e_i$ for $i = 1,\ldots,k-1$ and $i = n+1,\ldots,n+k-1$. The modification of the even sub-steps of JHOSH (or JHSH) algorithm is as follows. Let $A = [a_1,\ldots,a_n,a_{n+1},\ldots,a_{2n}] \in \mathbb{R}^{2n\times 2n}$ be a given matrix and set $A^{(0)} = A$. The first sub-step is obtained by creating the desired zeros in the first column, via the H_1 as above. The updated matrix is $A^{(1)}$. Now, for creating the desired zeros in the column $n+1$ and keeping the first column unchanged, we shall use the Van Loan's transformations, instead of H_2. For $k = n,\ldots,2$, we compute $J(k,c,s)$ such that a zero is created in position $n+k$ in the $n+1$th column of $J(k,c,s)A^{(1)}$. The first column as well as the already created zeros in the current $n+1$ column of $A^{(1)}$ remain unchanged. The first and the $n+1$th columns of $J(k,c,s)A^{(1)}$ leave unchanged when the latter is multiplied on the right by $J(k,c,s)^T$. The matrix $A^{(1)}$ is then updated with $A^{(2)} = J(k,c,s)A^{(1)}J(k,c,s)^T$. So the entries at positions $n+2,\ldots,2n$ in the $n+1$ column of $A^{(2)}$ are zeros. Now, we compute w so that the action of Van Loan's Householder in the product $H(2,w)A^{(2)}$ creates zeros in the positions $3,\ldots,n$ in the $n+1$ column. The first column of $H(2,w)A^{(2)}$ as well as the already created zeros remain unchanged. The transformation $H(2,w)$ leaves unchanged the first and the $n+1$ columns of the updated matrix $A^{(2)} = H(2,w)A^{(2)}H(2,w)^T$.

At the jth step, the first sub-step is obtained by creating the desired zeros in the jth column , via the H_{2j-1} as in JHOSH. The updated matrix is $A^{(2j-1)}$. Now, the desired zeros in the column $n+j$ are created by using the Van Loan's givens rotations, instead of H_{2j}. For $k = n,\ldots,j+1$, we compute $J(k,c,s)$ such that a zero is created in position $n+k$ in the $n+j$th column of $J(k,c,s)A^{(2j-1)}$. The columns $1,\ldots,j$ and $n+1,\ldots,n+j-1$ as well as the already created zeros in the current $n+j$ column of $A^{(2j-1)}$ remain unchanged. The columns $1,\ldots,j$ and $n+1,\ldots,n+j$ of $J(k,c,s)A^{(2j-1)}$ leave unchanged when the latter is multiplied on the right by $J(k,c,s)^T$. The matrix $A^{(2j-1)}$ is then updated with $A^{(2j)} = J(k,c,s)A^{(2j-1)}J(k,c,s)^T$. So the entries at positions $n+j+1,\ldots,2n$ in the $n+j$ column of $A^{(2j)}$ are zeros. Now, we compute w so that the action of Van Loan's Householder in the product $H(j,w)A^{(2j)}$ creates zeros in the positions $j+2,\ldots,n$ in the $n+j$th column. The columns $1,\ldots,j$ and $n+1,\ldots,n+j-1$ as well as the already created zeros in the current $n+j$ column of $A^{(2j)}$ remain unchanged. $H(j,w)$

leaves unchanged the columns $1,\ldots,j$ and $n+1,\ldots,n+j$ of the updated matrix $A^{(2j)} = H(j,w)A^{(2j)}H(j,w)^T$. We obtain the following algorithm

Algorithm 10. *function [S,H]=JHMSH(A)*

$twon = size(A(:,1));\ n = twon/2;\ S = eye(twon);$

$for\ j = 1 : n-1$

$J = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)];$

$ro = [j : n, n+j : 2n];\ co = [j : n, n+j : 2n];$

$[c, v] = osh2(A(ro, j));$

% Updating A :

$A(ro, co) = A(ro, co) + c * v * (v' * J * A(ro, co));$

$A(:, co) = A(:, co) - (A(:, co) * (c * v)) * v' * J;$

% Updating S (if needed):

$S(:, co) = S(:, co) - c * (v * v') * J * S(:, co);$

$for\ k = 2n : n+j+1,$

$[c, s] = vlg(k, A(:, n+j)),$

%Updating A:

$$\begin{bmatrix} A(k,co) \\ A(n+k,co) \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} A(k,co) \\ A(n+k,co) \end{bmatrix};$$

$$\begin{bmatrix} A(:,k) & A(:,n+k) \end{bmatrix} = \begin{bmatrix} A(:,k) & A(:,n+k) \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix};$$

%Updating S (if needed):

$$\begin{bmatrix} S(:,k) & S(:,n+k) \end{bmatrix} = \begin{bmatrix} S(:,k) & S(:,n+k) \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix};$$

end

if $j \le n-2$

[β,w]=vlh(j+1,A(:,n+j));

%Updating A:

$A(j+1 : n, co) = A(j+1 : n, co) - \beta * w * w' * A(j+1 : n, co)$

$A(j+1+n : 2n, co) = A(j+1+n : 2n, co) - \beta * w * w' * A(j+1+n : 2n, co);$

$A(:, j+1 : n) = A(:, j+1 : n) - \beta * A(:, j+1 : n)w * w';$

$A(:, n+j+1 : 2n) = A(:, n+j+1 : 2n) - \beta * A(:, n+j+1 : n)w * w';$

%Updating S (if needed):

$S(:, j+1 : n) = S(:, j+1 : n) - \beta * S(:, j+1 : n)w * w';$

$S(:, n+j+1 : 2n) = S(:, n+j+1 : 2n) - \beta * S(:, n+j+1 : n)w * w';$

end

end

end

Algorithm 11. *function[c,s]=vlg(k,a)*

$\%\ a = [a_1, \ldots, a_{2n}]$.
$twon = length(a);\ n = twon/2;$
$r = \sqrt{a_k^2 + a_{n+k}^2};$
if $r = 0$ *then* $c = 1;\ s = 0;$
else $\quad c = \dfrac{a_k}{r};\ \ s = \dfrac{a_{n+k}}{r};$
end

Algorithm 12. *function[β,w]=vlh(k,a)*

$\%\ a = [a_1, \ldots, a_{2n}]$.
$twon = length(a);\ n = twon/2;$
$\%\ w = (w_1, \ldots, w_{n-k+1})^T;$
$r_1 = \displaystyle\sum_{i=2}^{n-k+1} a_{i+k-1}^2;$
$r = \sqrt{a_k^2 + r_1};$
$w_1 = a_k + sign(a_k)r;$
$w_i = a_{i+k-1}$ *for* $i = 2, \ldots, n-k+1;$
$r = w_1^2 + r_1;\ \ \beta = \dfrac{2}{r};$
$\%P = I - \beta w w^T;\ \ (H(k,w)a)_i = 0$ *for* $i = k+1, \ldots, n.$
end

4. Discussion, numerical experiments

4.1. Curing breakdown

In this work, our goal was to introduce an new algorithm for computing a J-Hessenberg reduction of a matrix, via symplectic Householder transformations, which are rank-one modification of the identity. We showed how this reduction may be handled. The reduction process involves free parameters. We outlined how some optimal choice can be done, which gave rise to JHOSH algorithm. In order to enforce accuracy, we succeed to modify the JHOSH algorithm, by replacing half of the involved symplectic transformations with other transformations, which are both orthogonal and symplectic. This gave rise to JHMSH algorithm, which behaves with satisfactory properties and is better than all the previous ones. The algorithms JHESS

as well as JHSH and its different variants JHOSH, JHMSH, JHMSH2 may meet breakdowns/near-breakdowns. Such breakdowns/near-breakdowns occur exactly in the same condition for all these algorithms. This gives rise to very important questions concerning for example the different strategies for curing, when it is possible, such breakdowns/near-breakdowns. An extended and detailed study is needed. This will be the focus of a forthcoming paper. Nevertheless, we sketch out here an very efficient way to cure such breakdown (near-breakdown), as attested by the following numerical experiments. In the literature, to our knowledge, only the JHESS algorithm is used to perform a J-Hessenberg reduction of a matrix, with symplectic transformations. When one applies JHESS, as presented in [2], to a matrix A for which, a fatal breakdown is encountered, the algorithm breaks. As consequence, no J-Hessenberg reduction is obtained. Furthermore, the SR-algorithm can not be executed. Such fatal breakdown is not necessarily incurable. To overcome this serious difficulty, we introduce the following idea : instead of applying JHESS (respectively JHMSH, JHMSH2) directly to A, one builds a similar matrix $S^{-1}AS$ for which the J-Hessenberg reduction exists. For an efficient design of the algorithm, the matrix S for which we are looking for must be symplectic (for preserving structures) and orthogonal (for accuracy). Also, S must be such that the similarity transformation $S^{-1}AS$ does not destroy the already created zeros in A and the cost for computing the matrix $S^{-1}AS$ should be very negligible, say $O(n)$. We are led to an algorithmic choice of S, that we implemented in the modified versions of all algorithms. Thus, JHESS incorporating this modification will be refereed to as MJHESS, and as JHM^{2}SH and JHM^{2}SH2 for JHMSH and JHMSH2 respectively. Details on this strategy for curing breakdowns and related topics will be the focus of a forthcoming paper.

4.2. Numerical experiments

We propose here some numerical examples in different situation, allowing us the comparison between the different algorithms.

Example 1: Let us take $A = randn(2n)$ and run JHESS, JHMSH, JHMSH2, JHOSH. One can observe that JHESS, JHMSH, JHMSH2 provide sensibly similar results, with a significant disadvantage for JHOSH. Notice also that injecting the Van Loan's Givens transformations before applying $osh2$ in the algorithm JHMSH2 enforce only moderately the accuracy

Loss of J-orthogonality $\left\|I - S^J S\right\|_2$				
$2n$	$JHESS$	$JHMSH$	$JHMSH2$	$JHOSH$
4	$2.2377e-16$	$5.0453e-16$	$1.5701e-16$	$1.3878e-16$
6	$1.2362e-15$	$1.0314e-14$	$7.2445e-15$	$7.8665e-15$
8	$1.1262e-15$	$4.4185e-15$	$2.0318e-15$	$8.2489e-15$
10	$5.5159e-15$	$6.6951e-14$	$6.1371e-14$	$2.3061e-13$
12	$8.3091e-15$	$8.3005e-13$	$1.2185e-13$	$7.7104e-13$
14	$5.5932e-14$	$4.5568e-13$	$3.8058e-13$	$8.9718e-11$
16	$1.4082e-14$	$2.0836e-13$	$9.2822e-14$	$5.9120e-12$
18	$2.8530e-14$	$1.5159e-12$	$4.1867e-13$	$1.8129e-10$
20	$1.5660e-13$	$8.1831e-11$	$2.4944e-11$	$1.9407e-09$
22	$1.6207e-14$	$3.0169e-13$	$3.8383e-13$	$8.4138e-11$
24	$6.5797e-14$	$9.3572e-12$	$8.3027e-12$	$8.0934e-09$
26	$1.2295e-13$	$3.9518e-11$	$5.5587e-12$	$1.0048e-05$
28	$4.5993e-14$	$1.6715e-12$	$1.1331e-12$	$1.6731e-09$
30	$6.1491e-13$	$5.9323e-11$	$1.1910e-11$	$2.8166e-05$

Table 1: Loss of J-orthogonality, $A = randn(2n)$.

compared to JHMSH. The loss of J-orthogonality and the error in the J-Hessenberg reduction, for the different algorithms, are displayed in Table 1 and Table 2 respectively.

Example 2: Let us now consider the following matrix $A = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$, with $M_{11} = \begin{pmatrix} 1 & & & \\ 2 & 1 & & \\ & \ddots & \ddots & \\ & & 2 & 1 \end{pmatrix}$, $M_{12} = \begin{pmatrix} 1 & 2 & & \\ 2 & 1 & \ddots & \\ & \ddots & \ddots & 2 \\ & & 2 & 1 \end{pmatrix}$, $M_{21} = \begin{pmatrix} 0 & 2 & & \\ 0 & 1 & \ddots & \\ & \ddots & \ddots & 2 \\ & & 0 & 1 \end{pmatrix}$ and $M_{22} = \begin{pmatrix} 1 & & & \\ 3 & 1 & & \\ & \ddots & \ddots & \\ & & 3 & 1 \end{pmatrix}$. Each block M_{ij} is of size $n \times n$.
The algorithms JHESS, JHMSH and JHMSH2, when applied to the matrix A encounter a fatal breakdown since the first step, for all $n \geq 2$. In Table 3 are displayed the loss of J-orthogonality and the error in the J-Hessenberg reduction, for the algorithms MJHESS, JHM^{2}SH and JHM^{2}SH2. The results

Error of J-Hessenberg reduction $\left\|H - S^J AS\right\|_2$				
$2n$	$JHESS$	$JHMSH$	$JHMSH2$	$JHOSH$
4	$7.6284e-16$	$1.7280e-15$	$2.1593e-15$	$1.4299e-15$
6	$1.1399e-14$	$1.3724e-13$	$1.4030e-13$	$4.4936e-13$
8	$5.4087e-15$	$2.7576e-14$	$2.6325e-14$	$8.9172e-14$
10	$4.1767e-14$	$2.6532e-12$	$4.1763e-13$	$1.2819e-11$
12	$4.9776e-14$	$7.1119e-12$	$9.9413e-13$	$2.0655e-11$
14	$1.7671e-13$	$1.3752e-11$	$3.8978e-12$	$6.4185e-09$
16	$1.2971e-13$	$1.4069e-12$	$1.8731e-12$	$2.1820e-10$
18	$1.7410e-13$	$6.4203e-12$	$8.1253e-12$	$3.1054e-08$
20	$1.6234e-12$	$3.3818e-09$	$1.5008e-10$	$7.7719e-07$
22	$1.2996e-13$	$7.0366e-12$	$9.0303e-12$	$2.4491e-09$
24	$7.4530e-13$	$3.1000e-10$	$1.5243e-10$	$7.4156e-07$
26	$1.2377e-12$	$1.2405e-09$	$5.7954e-11$	$6.3300e-02$
28	$7.0871e-13$	$1.8094e-11$	$9.4246e-12$	$4.9546e-07$
30	$3.9641e-12$	$1.4573e-09$	$1.4503e-10$	$1.1500e-02$

Table 2: Error of J-Hessenberg reduction, $A = randn(2n)$

show efficiency of our strategy for curing the breakdown.

Example 3: consider now the Hamiltonian case :

$$A = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}, \text{where } M_{11} = \begin{pmatrix} 1 & & & \\ 2 & 1 & & \\ & \ddots & \ddots & \\ & & 2 & 1 \end{pmatrix}, M_{12} = \begin{pmatrix} 1 & 2 & & \\ 2 & 1 & \ddots & \\ & \ddots & \ddots & 2 \\ & & 2 & 1 \end{pmatrix},$$

$$M_{21} = \begin{pmatrix} 0 & 0 & & \\ 0 & 1 & 3 & \\ & 3 & \ddots & 3 \\ & & 3 & 1 \end{pmatrix} \text{ and } M_{22} = -M_{11}^T.$$ As for example 2, the algorithm JHESS as described in [2] meets a fatal breakdown since the first step for all $n \geq 2$ and the algorithm stops. No Hamiltonian J-Hessenberg form is obtained and hence the SR-algorithm may not executed. The algorithms JHMSH, JHMSH2 are subject to the same fatal breakdown, at the same moment and position. However, such fatal breakdown is not insurmountable. In Table 4 are displayed the loss of J-orthogonality and the error in the J-Hessenberg reduction for MJHESS, JHM^{2}SH and JHM^{2}SH2. The results

	Loss of J-orthogonality $\left\|I - S^J S\right\|_2$			Error $\left\|H - S^J AS\right\|_2$		
n	$MJHESS$	JHM^2SH	JHM^2SH2	$MJHESS$	JHM^2SH	JHM^2SH2
2	$1.0717e-15$	$2.5168e-16$	$3.1402e-16$	$8.9915e-15$	$1.0361e-15$	$1.0262e-15$
3	$1.6767e-15$	$1.0412e-15$	$9.7146e-16$	$3.9086e-15$	$1.0623e-14$	$5.6678e-15$
4	$1.0717e-15$	$3.1015e-15$	$3.6572e-15$	$8.9915e-15$	$6.3153e-14$	$2.9172e-14$
5	$5.5610e-15$	$2.8250e-14$	$3.3284e-14$	$2.5502e-14$	$1.4279e-13$	$6.8545e-14$
6	$5.2871e-15$	$4.1918e-14$	$4.3812e-14$	$5.1641e-14$	$2.5845e-13$	$1.6997e-13$
7	$1.3446e-14$	$2.0709e-13$	$1.1965e-13$	$4.9408e-14$	$2.7021e-12$	$5.7755e-13$
8	$1.8294e-14$	$1.7497e-12$	$7.4477e-13$	$1.2355e-13$	$1.0972e-11$	$3.5435e-12$
9	$3.8698e-13$	$1.2988e-10$	$5.8035e-11$	$1.2932e-12$	$1.0461e-09$	$3.8219e-10$
10	$2.4877e-13$	$4.8062e-10$	$1.1476e-10$	$2.4578e-12$	$3.4164e-09$	$7.1532e-10$
11	$3.2961e-13$	$6.6942e-10$	$1.7784e-10$	$2.7462e-12$	$4.7274e-09$	$5.7041e-10$
12	$4.2368e-13$	$4.5165e-10$	$1.7250e-10$	$6.5230e-12$	$1.1306e-08$	$8.0399e-10$
13	$9.8990e-13$	$7.9908e-10$	$2.9785e-10$	$7.3703e-12$	$7.4063e-09$	$1.7637e-09$
14	$1.4096e-12$	$7.6406e-10$	$1.7497e-10$	$1.1269e-11$	$8.3607e-09$	$1.0158e-09$
15	$8.8834e-13$	$1.7248e-09$	$1.9073e-10$	$7.3380e-12$	$1.1932e-08$	$9.8201e-10$
16	$8.3168e-13$	$6.9530e-10$	$1.9133e-10$	$6.7075e-12$	$5.6770e-09$	$1.1922e-09$
17	$9.0923e-13$	$1.9515e-09$	$2.1889e-10$	$8.2522e-12$	$1.4054e-08$	$1.2598e-09$
18	$1.4143e-12$	$1.1824e-09$	$6.2781e-10$	$3.4928e-11$	$1.4967e-07$	$5.7161e-09$
19	$3.0133e-12$	$3.6906e-09$	$2.2293e-10$	$2.1809e-11$	$2.5400e-08$	$1.4194e-09$
20	$3.0854e-12$	$2.8172e-09$	$2.6019e-10$	$6.7873e-11$	$1.2725e-07$	$2.0413e-09$
21	$2.4744e-12$	$1.5606e-08$	$8.6765e-10$	$3.9748e-11$	$2.6936e-07$	$5.1208e-09$
22	$1.1614e-12$	$1.0522e-09$	$2.4081e-10$	$6.5786e-11$	$1.1047e-08$	$1.9222e-09$
23	$1.2597e-12$	$3.8242e-09$	$2.6805e-10$	$6.3465e-11$	$2.1954e-08$	$1.6025e-09$
24	$2.0548e-12$	$1.1119e-09$	$4.8392e-10$	$6.2066e-11$	$5.6800e-08$	$3.2751e-09$
25	$3.0479e-12$	$3.9755e-09$	$4.2710e-10$	$2.3255e-11$	$2.2816e-08$	$2.6839e-09$
26	$3.8862e-12$	$1.8132e-09$	$1.4496e-09$	$2.9622e-11$	$3.2416e-08$	$1.0678e-08$
27	$4.3655e-12$	$1.2417e-08$	$1.1257e-09$	$1.0102e-10$	$1.0768e-07$	$1.0010e-08$
28	$4.3449e-12$	$2.2564e-09$	$1.1255e-09$	$1.4271e-10$	$1.4462e-07$	$8.2262e-09$
29	$9.6426e-12$	$3.9904e-08$	$2.3791e-09$	$5.3487e-11$	$6.3257e-07$	$4.1958e-08$
30	$5.3754e-12$	$1.6554e-09$	$5.4776e-10$	$4.6021e-11$	$5.9380e-08$	$4.0406e-09$

Table 3: Curing the breakdown

	Loss of J-orthogonality $\|I - S^J S\|_2$			Error $\|H - S^J AS\|_2$		
n	$MJHESS$	JHM^2SH	JHM^2SH2	$MJHESS$	JHM^2SH	JHM^2SH2
2	$2.6809e-16$	$1.3843e-16$	$2.7756e-17$	$1.2230e-15$	$3.4732e-16$	$7.5047e-16$
3	$9.1518e-16$	$2.1967e-15$	$4.1153e-15$	$4.6309e-15$	$1.4123e-14$	$9.5826e-15$
4	$3.6585e-15$	$3.1724e-14$	$1.1623e-14$	$1.1179e-14$	$1.0235e-13$	$1.1283e-13$
5	$1.3451e-14$	$5.5639e-13$	$4.5393e-13$	$1.0634e-13$	$2.2678e-12$	$1.4082e-12$
6	$3.2002e-15$	$1.3229e-14$	$3.1824e-14$	$2.0835e-14$	$1.6308e-13$	$1.8500e-13$
7	$1.7497e-14$	$1.9456e-13$	$2.9018e-13$	$3.6429e-13$	$4.2300e-12$	$5.7276e-12$
8	$1.1440e-14$	$2.4182e-13$	$9.1255e-14$	$5.3612e-14$	$2.6360e-12$	$1.2184e-12$
9	$4.7591e-14$	$7.0030e-12$	$4.6008e-12$	$4.2431e-13$	$2.8308e-11$	$6.0019e-11$
10	$9.8212e-14$	$6.7908e-11$	$1.8421e-11$	$5.5556e-13$	$1.8128e-10$	$4.2484e-11$
11	$2.5071e-13$	$1.2746e-10$	$3.6111e-11$	$6.2363e-12$	$1.2132e-09$	$1.3393e-10$
12	$3.0863e-13$	$1.6379e-09$	$1.1448e-10$	$3.2918e-12$	$5.6804e-09$	$1.0683e-09$
13	$2.3432e-12$	$5.7401e-09$	$1.8386e-09$	$1.7487e-11$	$4.3477e-07$	$5.7596e-09$
14	$1.5649e-12$	$5.9220e-09$	$2.7826e-09$	$1.2069e-11$	$1.1117e-07$	$1.1405e-08$
15	$1.2911e-11$	$1.1198e-07$	$1.5282e-08$	$1.2035e-10$	$8.4815e-07$	$2.1596e-07$

Table 4: Curing breakdown, Hamiltonian case

show the efficiency of our strategy for remedying this breakdown.

5. Conclusion

In this paper, we introduce a reduction of a matrix to the upper J-Hessenberg form, based on the symplectic Householder transformations, which are rank-one modification of the Identity. This reduction is the crucial step for constructing an efficient SR-algorithm. The method is the analogue of the reduction of a matrix to Hessenberg form, via Householder transformations, when instead of an Euclidean linear space, one takes a sympletctic one. Then the algorithm JHOSH is derived, corresponding to an optimal choice of the free parameters. Furthermore, JHOSH is significantly improved by showing that half of these symplectic Householder transformations may be replaced by Van Loan's symplectic and orthogonal transformations leading to two variants JHMSH and JHMSH2 which are significantly more stable numerically. These algorithms behave quite similarly to JHESS algorithm. Moreover, all of them may meet fatal breakdown at the same moment and position. We sketched out an efficient strategy to cure such fatal breakdowns. The new algorithms incorporating this strategy are refereed to MJHESS, JHM^{2}SH

and JHM^{2}SH2. Their efficiency is attested by the numerical experiments. However, the treatment of breakdowns/near-breakdowns and related topics deserve more investigations and will be the focus of a forthcoming work.

[1] E. Artin, *Geometric Algebra*, Interscience Publishers, New York, 1957.

[2] A. Bunse-Gerstner and V. Mehrmann, A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation, IEEE Trans. Automat. Control **AC-31** (1986), 1104–1113.

[3] A. Bunse-Gerstner, Matrix factorizations for symplectic QR-like methods, Linear Algebra Appl. **83** (1986), 49–77.

[4] J. Della-Dora, Numerical linear algorithms and group theory, Linear Algebra Appl. **10** (1975), 267–283.

[5] G. Golub and C. Van Loan, *Matrix Computations*, third ed., The Johns Hopkins U.P., Baltimore, 1996.

[6] C. Paige and C. Van Loan, A Schur decomposition for Hamiltonian matrices, Linear Algebra Appl. **41** (1981), 11–32.

[7] A. Salam, On theoretical and numerical aspects of symplectic Gram-Schmidt-like algorithms, Numer. Algo., **39** (2005), 237-242.

[8] A. Salam, A. El Farouk, E. Al-Aidarous, Symplectic Householder Transformations for a QR-like decomposition, a Geometric and Algebraic Approaches, J. of Comput. and Appl. Math., Vol. 214, Issue 2, 1 May 2008, Pages 533-548.

[9] A. Salam and E. Al-Aidarous and A. Elfarouk, Optimal symplectic Householder transformations for SR-decomposition, Linear Algebra and Its Appl., 429 (2008), no. 5-6, 1334-1353.

[10] A. Salam, E. Al-Aidarous, Error analysis and computational aspects of SR factorization, via optimal symplectic Householder Transformations, Electronic Trans. on Numer. Anal., Vol. 33, pp. 189-206, 2009.

[11] A. Salam and E. Al-Aidarous, Equivalence between modified symplectic Gram-Schmidt and Householder SR algorithms, BIT Numer. Math., Vol. 54, pp. 283-302, 2014.

[12] C. Van Loan, A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix, Linear Algebra Appl. **61** (1984), 233–251.

[13] D.S. Watkins, *The Matrix Eigenvalue Problem : GR and Krylov subspace methods*, SIAM, 2007.

[14] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England.

Annexe B

A treatment of breakdowns and near breakdowns in a reduction of a matrix to upper J-Hessenberg form and related topics

A treatment of breakdowns and near breakdowns in a reduction of a matrix to upper J-Hessenberg form and related topics

Ahmed Salam[a,*], Haithem Ben Kahla[a,b]

[a] *University Lille Nord de France, ULCO, LMPA. BP 699, F62228, Calais, Cedx, France.*
[b] *University of Tunis El Manar, ENIT-LAMSIN, BP 37, 1002, Tunis, Tunisia.*

Abstract

The reduction of a matrix to an upper J-Hessenberg form is a crucial step in the SR-algorithm (which is a QR-like algorithm), structure-preserving, for computing eigenvalues and vectors, for a class of structured matrices. This reduction may be handled via the algorithm JHESS or via the recent algorithm JHMSH and its variants.

The main drawback of JHESS (or JHMSH) is that it may suffer from a fatal breakdown, causing a brutal stop of the computations and hence, the SR-algorithm does not run. JHESS may also encounter near-breakdowns, source of serious numerical instability.

In this paper, we focus on these aspects. We first bring light on the necessary and sufficient condition for the existence of the SR-decomposition, which is intimately linked to J-Hessenberg reduction. Then we will derive a strategy for curing fatal breakdowns and also for treating near breakdowns. Hence, the J-Hessenberg form may be obtained. Numerical experiments are given, demonstrating the efficiency of our strategies to cure and treat breakdowns or near breakdowns.

Keywords: SR decomposition, symplectic Householder transformations, upper J- Hessenberg form, breakdowns and near-breakdowns, SR-algorithm.
2000 MSC: 65F15, 65F50

∗. corresponding author
Email addresses: ahmed.salam@univ-littoral.fr (Ahmed Salam), benkahla@univ-littoral.fr (Haithem Ben Kahla)

1. Introduction

Let A be a $2n \times 2n$ real matrix. The SR factorization consists in writing A as a product SR, where S is symplectic and $R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$ is such that $R_{11},\ R_{12},\ R_{22}$ are upper triangular and R_{21} is strictly upper triangular [3, 4]. The factor R is called J-triangular. This decomposition plays an important role in structure-preserving methods for solving the eigenproblem of a class of structured matrices.

More precisely, the SR decomposition can be interpreted as the analog of the QR decomposition [6], when instead of an Euclidean space, one considers a symplectic space : a linear space, equipped with a skew-symmetric inner product (see for example [8] and the references therein). The orthogonal group with respect to this indefinite inner product, is called the symplectic group and is unbounded (contrasting with the Euclidean case).

In the literature, the SR decomposition is carried out, via the algorithm SRDECO, derived in [2]. SRDECO is based in the use of two kind of both symplectic and orthogonal transformations introduced in [7, 13] and a third symplectic but non-orthogonal transformations, proposed in [2]. In fact, in [3], it has been shown that SR decomposition of a general matrix can not be performed by employing only the above orthogonal and symplectic transformations.

We mention that the above transformations involved in SRDECO algorithm are not elementary rank-one modification of the identity (transvections), see [1, 6].

Recently in [9], an algorithm, SRSH, based on symplectic transformations which are rank-one modification of the identity is derived, for computing the SR decomposition. These transformations are called symplectic Householder transformations. The new algorithm SRSH involves free parameters and advantages may be taken from this fact. An optimal version of SRSH, called SROSH is given in [10]. Error analysis and computational aspects of this algorithm have been studied [11].

In order to build a SR-algorithm (which is a QR-like algorithm) for computing the eigenvalues and eigenvectors of a matrix [14], a reduction of the matrix to an upper J-Hessenberg form is needed and is crucial.

In [2], the algorithm JHESS, for reducing a general matrix to an upper J-Hessenberg form is presented, using to this aim, an adaptation of SRDECO.

In [12], the algorithm JHSH, based on an adaptation of SRSH, is introduced, for reducing a general matrix, to an upper J-Hessenberg form. Variants of JHSH, named JHOSH, JHMSH and JHMSH2 are then derived, motivated by the numerical stability. The algorithms JHESS as well as JHSH and its variants have $O(n^3)$ as complexity.

The algorithm JHESS (and also SRDECO), as described in [2] may be subject of a fatal breakdown, causing a brutal stop of the computations. As consequence, the J-Hessenberg reduction can not be computed and the SR-algorithm does not run. Moreover, we demonstrate that the algorithm JHESS may breaks down while a condensed J Hessenberg form exists. These algorithms also may suffer from severe form of near-breakdowns, source of serious numerical instability.

In this paper, we restrict ourselves to the study of such aspects, bringing significant insights on SRDECO and JHESS algorithms.

We will show derive a strategy for curing fatal breakdowns and treating near breakdowns. To this aim, we first bring light on the SR-decomposition and SRDECO algorithm, in connection with the theory developed by Elsner in [5]. Then, a strategy for remedying to such breakdowns is proposed. The same strategy is used for treating the near-breakdowns. Numerical experiments are given, demonstrating the efficiency of our strategies to cure breakdowns or to treat near breakdowns.

The remainder of this paper is organized as follows. Section 2, is devoted to the necessary preliminaries. In the section 3, the algorithms SRDECO, SRSH or SRMSH are presented. Then, we establish a connection between some coefficients of the current matrix produced by the SRDECO algorithm, when applied to a matrix A and the necessary and sufficient condition for the existence of the SR decomposition of A, as given in [5]. In section 4, we recall the algorithms JHESS and JHMSH. We present then an example, for which a fatal breakdown is meet, in JHESS algorithm (also for JHMSH), for reducing the matrix to an upper J-Hessenberg. And hence, following the description of JHESS in [2], the algorithm stops. No hope to build then a SR-algorithm. We will show on the same example, that such breakdown is curable. We then develop our strategies for curing the fatal breakdowns. The same strategies are applied for near-breakdowns. The Section 5 is devoted to numerical experiments and comparisons. We conclude in the section 6.

2. Preliminaries

Let J_{2n} (or simply J) be the $2n$-by-$2n$ real matrix

$$J_{2n} = \begin{bmatrix} 0_n & I_n \\ -I_n & 0_n \end{bmatrix}, \tag{1}$$

where 0_n and I_n stand respectively for n-by-n null and identity matrices. The linear space $\mathbb{R}^{2n}$ with the indefinite skew-symmetric inner product

$$(x, y)_J = x^T J y \tag{2}$$

is called symplectic. For $x,\ y \in \mathbb{R}^{2n}$, the orthogonality $x \perp' y$ stands for $(x, y)_J = 0$. The symplectic adjoint x^J of a vector x, is defined by

$$x^J = x^T J. \tag{3}$$

The symplectic adjoint of $M \in \mathbb{R}^{2n\times 2k}$ is defined by

$$M^J = J_{2k}^T M^T J_{2n}. \tag{4}$$

A matrix $S \in \mathbb{R}^{2n\times 2k}$ is called symplectic if

$$S^J S = I_{2k}. \tag{5}$$

The symplectic group (multiplicative group of square symplectic matrices) is denoted $\mathbb{S}$. A transformation T given by

$$T = I + cvv^J \text{ where } c \in \mathbb{R},\ \ v \in \mathbb{R}^\nu \quad (\text{with } \nu \text{ even}), \tag{6}$$

is called symplectic Householder transformation [9]. It satisfies

$$T^J = I - cvv^J. \tag{7}$$

The vector v is called the direction of T.

For $x,\ y \in \mathbb{R}^{2n}$, there exists a symplectic Householder transformation T such that $Tx = y$ if $x = y$ or $x^J y \neq 0$. When $x^J y \neq 0$, T is given by

$$T = I - \frac{1}{x^J y}(y - x)(y - x)^J.$$

Moreover, each non null vector x can be mapped onto any non null vector y by a product of at most two symplectic Householder transformations

[9]. Symplectic Householder transformations are rotations, i.e. $det(T) = 1$ and the symplectic group $\mathbb{S}$ is generated by symplectic Householder transformations. In [7, 13] two orthogonal and symplectic transformations have been introduced. The first, for which we refer as Van Loan's Householder transformation, has the form

$$H(k,w) = \begin{pmatrix} \text{diag}(I_{k-1}, P) & 0 \\ 0 & \text{diag}(I_{k-1}, P) \end{pmatrix}, \tag{8}$$

where

$$P = I - 2ww^T/w^Tw, \;\; w \in \mathbb{R}^{n-k+1}.$$

The second, for which we refer as Van Loan's Givens transformation, is

$$J(k,\theta) = \begin{pmatrix} C & S \\ -S & C \end{pmatrix}, \tag{9}$$

where

$$C = diag(I_{k-1}, cos\theta, I_{n-k}) \text{ and } S = diag(0_{k-1}, sin\theta, 0_{n-k}).$$

$J(k,\theta)$ is a Givens symplectic matrix, that is an "ordinary" $2n$-by-$2n$ Givens rotation that rotates in planes k and $k+n$ [15]. The SR factorization can not be performed for a general matrix by using the sole $H(k,w)$ and $J(k,\theta)$ transformations [2]. A third type, introduced in [2], is given by

$$G(k,\nu) = \begin{pmatrix} D & F \\ 0 & D^{-1} \end{pmatrix}, \tag{10}$$

where $k \in \{2, \ldots, n\}$, $\nu \in \mathbb{R}$ and $D,\ F$ are the $n \times n$ matrices

$$D = I_n + (\frac{1}{(1+\nu^2)^{1/4}} - 1)(e_{k-1}e_{k-1}^T + e_k e_k^T),$$

$$F = \frac{\nu}{(1+\nu^2)^{1/4}}(e_{k-1}e_k^T + e_k e_{k-1}^T).$$

The matrix $G(k,\nu)$ is symplectic and non-orthogonal. The SRDECO algorithm is then derived for computing SR factorization for a general matrix, based on H, J and G transformations. A reduction of a general matrix to an upper J-Hessenberg, is obtained by using the same transformations involved in SRDECO, giving rise to JHESS algorithm. The breakdown in SR-decompoisition via SRDECO or in the reduction to an upper J-Hessenberg form via JHESS, when it occurs, is caused by the latest transformations G.

3. SRDECO, SRSH algorithms

The aim of this work, is to bring significant contributions on the understanding and the behaviour of the algorithms SRDECO, SRSH, JHESS and JHMSH. Also, we propose strategies for curing breakdowns. Similar strategies are applied also for remedying to near breakdowns.

3.1. SR decomposition : SRDECO, SRSH algorithms

We consider the SRDECO algorithm, as introduced in [2]. Given $A \in \mathbb{R}^{2n\times 2n}$, the algorithm determines an SR decomposition of A, using functions vlg, vlh and gal below. The function vlg uses Van Loan's Givens transformation $J(k,c,s)$ as follows : for a given integer $1 \leq k \leq n$ and a vector $a \in \mathbb{R}^{2n}$, it determines coefficients c and s such that the $n+k$th component of $J(k,c,s)a$ is zero. All components of $J(k,c,s)a$ remain unchanged, except eventually the kth and the $n+k$.

Algorithm 1. *function[c,s]=vlg(k,a)*
$twon = length(a);\ n = twon/2;$
$r = \sqrt{a(k)^2 + a(n+k)^2};$
if $r = 0$ *then* $c = 1;\ s = 0;$
else $c = \dfrac{a(k)}{r};\ \ s = \dfrac{a(n+k)}{r};$
end

The function *vlh* uses Van Loan's Householder transformation $H(k,w)$ as follows : for a given integer $k \leq n$ and a vector $a \in \mathbb{R}^{2n}$, a vector $w = (w_1,\ldots,w_{n-k+1})^T$ is determined such that the components $k+1,\ldots,n$ of $H(k,w)a$ are zeros. All components $1,\ldots,k-1$ and $n+1,\ldots,n+k-1$ remain unchanged.

Algorithm 2. *function[β,w]=vlh(k,a)*
$twon = length(a);\ n = twon/2;$
$\%\ w = (w_1,\ldots,w_{n-k+1})^T;$
$r1 = \sum_{i=2}^{n-k+1} a(i+k-1)^2;$
$r = \sqrt{a(k)^2 + r1};$
$w_1 = a(k) + sign(a(k))r;$
$w_i = a(i+k-1)$ *for* $i = 2,\ldots,n-k+1;$
$r = w_1^2 + r1;\ \ \beta = \dfrac{2}{r};$
$\%P = I - \beta w w^T;\ \ (H(k,w)a)_i = 0$ *for* $i = k+1,\ldots,n.$
end

The function *gal* uses the transformation $G(k,\nu)$ as follows : for a given integer $k \leq n$ and a vector $a \in \mathbb{R}^{2n}$, satisfying the condition $a_{n+k} = 0$ only if $a_{k+1} = 0$, it determines ν such that the $k+1$th of $G(k,\nu)a$ is zero.

Algorithm 3. *function*$[\nu] = gal(k, a)$
$twon = length(a);\ n = twon/2;$
if $a_k = 0$
$\nu = 0;$
else
$\nu = -\dfrac{a_{k+1}}{a_{n+k}}$;
end
end

The algorithm SRDECO is as follows : the matrix A is overwritten by the J-upper triangular matrix. If A has no SR decomposition, the algorithm stops.

Algorithm 4. *function [S,A]=SRDECO(A)*
1. *For* $j = 1, \ldots, n$
2. *For* $k = n, \ldots, j$
3. *Zero the entry* $(n+k, j)$ *of* A *by running the function* $[c, s] = vlg(k, A(:, j))$ *and computing* $J_{k,j} = J(k, c, s)$.
4. *Update* $A = J_{k,j}A$ *and* $S = SJ_{k,j}^T$
5. *End for.*
6. *Zero the entries* $(j+1, j), \ldots, (n, j)$ *of* A *by running the function* $[\beta, w] = vlh(j, A(:, j))$ *and computing* $H_j = H(j, w)$.
7. *Update* $A = H_jA$ *and* $S = SH_j^T$.
8. *If* $j \leq n-1$
9. *For* $k = n, \ldots, j+1$
10. *Zero the entry* $(n+k, n+j)$ *of* A *by running the function* $[c, s] = vlg(k, A(:, n+j))$ *and computing* $J_{k,n+j} = J(k, c, s)$.
11. *Update* $A = J_{k,n+j}A$ *and* $S = SJ_{k,n+j}^T$.
12. *End for.*
13. *Zero the entries* $(j+2, n+j), \ldots, (n, n+j)$ *of* A *by running the function* $[\beta, w] = vlh(j+1, A(:, j))$ *and computing* $H_{n+j} = H(j+1, w)$.
14. *Update* $A = H_{n+j}A$ *and* $S = SH_{n+j}^T$.
15. *If the entry* $(j+1, n+j)$ *of* A *is nonzero and the entry* $(n+j, n+j)$ *is zero then stop the algorithm,*
16. *else*

17. *Zero the entry* $(j+1, n+j)$ *of* A *by running the function* $[\nu] = gal(j+1, A(:, n+j))$ *and computing* $G_{j+1} = G(j+1, \nu)$.
18. *Update* $A = G_{j+1}A$ *and* $S = SG_{j+1}^{-1}$. % $G_{j+1}^{-1} = G_{j+1}^{J}$.
19. *End if*
20. *End if*
21. *End for.*

The SR decomposition can be also performed using only the symplectic Householder transformations T of (6), which are rank-one modifications of the identity, giving rise to the algorithm SRSH. More on this can be found in [9, 10, 11]. A modified version of SRSH, numerically more stable, is SRMSH. It turn out that SRMSH shares the same steps (1-16) of SRDECO, but not the remaining ones. In fact, the function *gal* and the symplectic matrices G_{j+1} in SRDECO are replaced by the function $sh2$ and T_j having the form of (6). The function $sh2$ is as follows

Algorithm 5. *function [c, v] = sh2(a)*
%compute c and v such that $T_2 e_1 = e_1$, *and* $T_2 a = \mu e_1 + \nu e_{n+1}$,
%μ *is a free parameter, and* $T_2 = (eye(twon) + c * v * v' * J)$;
$twon = length(a)$; $n = twon/2$;
$J = [zeros(n), eye(n); -eye(n), zeros(n)]$;
If $n == 1$
$v = zeros(twon, 1)$; $c = 0$; $\%T = eye(twon)$;
else
Choose μ;
$\nu = a(n+1)$;
If $\nu == 0$
Display('division by zero')
Return
else
$v = \mu e_1 + \nu e_{n+1} - a$, $c = \dfrac{1}{a(n+1)(a(1)-\mu)}$;
End
End

We obtain the algorithm

Algorithm 6. *function [S,A]=SRMSH(A)*
1. *Run steps* ***1.-16.*** *of SRDECO*
17. *Set* $c0 = [j : n, n+j : 2n]$, $c1 = [j+1 : n, n+j : 2n]$,

18. *Zero the entry* $(j+1, n+j)$ *of* A *by running the function* $[c, v] = sh2(A(co, n+j))$
19. *compute* $T_j = I + cvv^T J$, *update* $A(:, c1) = T_j A(:, c1)$ *and* $S(:, c0) = ST_j^J$.
20. *End if*
21. *End if*
22. *End for.*

Remark 1. The function $sh2$ in the body of the algorithm $SRMSH$ may by replaced by the function $osh2$ (see [10, 11]) which presents the best conditioning among all possible choices.

3.2. Discussion : existence of SR decomposition, link with SRDECO and SRMSH

In this subsection, we bring light on the connection between the existence of SR decomposition and the algorithm SRDECO or equivalently SRMSH. We recall first the following result, given in [5] :

Theorem 7. *Let* $A \in \mathbb{R}^{2n\times 2n}$ *be nonsingular and* P *the permutation matrix* $P = [e_1, e_{n+1}, e_2, e_{n+2}, \ldots, e_n, e_{2n}]$, *where* e_i *denotes the* i*th canonical vector of* $\mathbb{R}^{2n}$. *There exists* $S \in \mathbb{R}^{2n\times 2n}$ *symplectic and* $R \in \mathbb{R}^{2n\times 2n}$ *upper* J*-triangular, such that* $A = SR$ *if and only if all even leading minors of* $P^T A^T JAP$ *are nonzero.*

In [2], a comment on SRDECO states : "if at any stage $j \in \{1, \ldots, n-1\}$ the algorithm ends because of the stopping condition, then the $2j$th leading principal minor of $P^T A^T JAP$ is zero, and A has no SR decomposition (see Theorem 7)." However, a proof of how is connected the stopping condition of the algorithm SRDECO with the condition of Theorem 7 is not given. Remark also that for SRDECO algorithm, the condition A nonsingular is not required, while it is for Theorem 7. Here we give a proof on how this connection is made. Notice first that if $A = SR$, where S is any symplectic matrix and R is any matrix, then $A^T JA = R^T S^T JSR = R^T JR$. Hence a minor of $A^T JA$ is equal to its corresponding one of $R^T JR$. The same equality between minors is valid also for $P^T A^T JAP$ and $P^T R^T JRP$. The following Theorem establishes an explicit relation between the leading $2j$-by-$2j$ minors of $P^T A^T JAP$ and the computed coefficients which determine the stopping condition of SRDECO. For a given matrix M, let us denote by $M_{[j,j]}$ the submatrix obtained from M by deleting all rows and columns except rows and columns $1, \ldots, j$. We have

Theorem 8. *Let $A \in \mathbb{R}^{2n\times 2n}$ be a matrix (not necessarily nonsingular), and let R be the matrix that one obtains at stage $1 \le j \le n-1$ of the algorithm SRDECO, by executing instructions **1.** to **14.** (corresponding to the current updated matrix A in the process, at stage j and until instruction **14.**). Then the leading $2j$-by-$2j$ minor of P^TA^TJAP satisfies*

$$det((P^TA^TJAP)_{[2j,2j]}) = [r_{1,1}r_{n+1,n+1}\dots r_{i,i}r_{n+i,n+i}\dots r_{j,j}r_{n+j,n+j}]^2 . \quad (11)$$

PROOF. Partitioning $R = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix}$, then $R_{11},\ R_{12},\ R_{21},\ R_{22}$ have the form

$$R_{11} = \begin{bmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,j} & r_{1,j+1} & \dots & r_{1,n} \\ 0 & r_{2,2} & \dots & r_{2,j} & r_{2,j+1} & \dots & r_{2,n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \dots & \vdots \\ \vdots & & \ddots & r_{j,j} & r_{j,j+1} & \dots & r_{j,n} \\ 0 & \dots & \dots & 0 & r_{j+1,j+1} & \dots & r_{j+1,n} \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & & \dots & 0 & r_{n,j+1} & \dots & r_{n,n} \end{bmatrix},$$

$$R_{12} = \begin{bmatrix} r_{1,n+1} & r_{1,n+2} & \dots & r_{1,n+j-1} & r_{1,n+j} & r_{1,n+j+1} & \dots & r_{1,2n} \\ 0 & r_{2,n+2} & \dots & r_{2,n+j-1} & r_{2,n+j} & r_{2,n+j+1} & \dots & r_{2,2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ \vdots & & \ddots & r_{j-1,n+j-1} & r_{j-1,n+j} & r_{j-1,n+j+1} & \dots & r_{j-1,2n} \\ 0 & \dots & \dots & 0 & r_{j,n+j} & r_{j,n+j+1} & \dots & r_{j,2n} \\ 0 & \dots & \dots & 0 & r_{j+1,n+j} & r_{j+1,n+j+1} & \dots & r_{j+1,2n} \\ 0 & \dots & \dots & 0 & 0 & r_{j+2,n+j+1} & \dots & r_{j+2,2n} \\ \vdots & & & \vdots & \vdots & \vdots & & \vdots \\ 0 & & \dots & 0 & 0 & r_{n,n+j+1} & \dots & r_{n,2n} \end{bmatrix},$$

$$R_{21} = \begin{bmatrix} 0 & r_{n+1,2} & \dots & r_{n+1,j} & r_{n+1,j+1} & \dots & r_{n+1,n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & r_{n+j-1,j} & r_{n+j-1,j+1} & \dots & r_{n+j-1,n} \\ 0 & \dots & 0 & 0 & r_{n+j,j+1} & \dots & r_{n+j,n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & r_{2n,j+1} & \dots & r_{2n,n} \end{bmatrix},$$

and

$$R_{22} = \begin{bmatrix} r_{n+1,n+1} & r_{n+1,n+2} & \dots & r_{n+1,n+j} & r_{n+1,n+j+1} & \dots & r_{n+1,2n} \\ 0 & r_{n+2,n+2} & \dots & r_{n+2,n+j} & r_{n+2,n+j+1} & \dots & r_{n+2,2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \dots & \ddots & r_{n+j,n+j} & r_{n+j,n+j+1} & \dots & r_{n+j,2n} \\ 0 & \dots & \dots & 0 & r_{n+j+1,n+j+1} & \dots & r_{n+j+1,2n} \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & \dots & \dots & 0 & r_{2n,n+j+1} & \dots & r_{2n,2n} \end{bmatrix}.$$

The stopping condition at this stage j is "$r_{n+j,n+j} = 0$ and $r_{j+1,n+j} \neq 0$". We will establish connection between the coefficients $r_{i,i},\ r_{n+i,n+i}$ of the current matrix R, with $1 \leq i \leq j$ and the leading $2j$-by-$2j$ minor of $P^T A^T JAP$. Setting $\hat{J} = P^T JP$ and $\hat{R} = P^T RP$, we get $P^T A^T JAP = P^T R^T JRP = (P^T R^T P)(P^T JP)(P^T RP) = \hat{R}^T \hat{J} \hat{R}$. Recall that $\hat{J} = diag(J_2, \dots, J_2)$. Partitioning $\hat{R} = \begin{pmatrix} \hat{R}_{11} & \hat{R}_{12} \\ \hat{R}_{21} & \hat{R}_{22} \end{pmatrix}$, with the block $\hat{R}_{11}$ is $2j$-by-$2j$. Then we obtain for $\hat{R}_{11},\ \hat{R}_{12},\ \hat{R}_{21},\ \hat{R}_{22}$:

$$\hat{R}_{11} = \begin{bmatrix} r_{1,1} & r_{1,n+1} & \dots & r_{1,i} & r_{1,n+i} & \dots & r_{1,j} & r_{1,n+j} \\ 0 & r_{n+1,n+1} & \dots & r_{n+1,i} & r_{n+1,n+i} & \dots & r_{n+1,j} & r_{n+1,n+j} \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & r_{i,i} & r_{i,n+i} & \dots & r_{i,j} & r_{i,n+j} \\ \vdots & \vdots & & 0 & r_{n+i,n+i} & \dots & r_{n+i,j} & r_{n+i,n+j} \\ \vdots & \vdots & & 0 & 0 & \ddots & \vdots & \vdots \\ \vdots & \vdots & & 0 & 0 & \ddots & r_{j,j} & r_{j,n+j} \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 & r_{n+j,n+j} \end{bmatrix}, \quad (12)$$

which is a upper $2j$-by-$2j$ triangular matrix. The $2(n-j)$-by-$2j$ block $\hat{R}_{21}$ turn out to have all entries zeros except the entry in position $(1, j)$. More precisely, $\hat{R}_{21} = \begin{pmatrix} 0 & r_{j+1,n+j} \\ 0 & 0 \end{pmatrix}$. Setting $\hat{J}_{2k} = diag(J_2, \dots, J_2) \in \mathbb{R}^{2k \times 2k}$ for an integer k, and due to the special structures of $\hat{J}$, we get

$$\begin{aligned} \hat{R}^T \hat{J} \hat{R} &= \begin{pmatrix} \hat{R}_{11} & \hat{R}_{12} \\ \hat{R}_{21} & \hat{R}_{22} \end{pmatrix}^T \hat{J} \begin{pmatrix} \hat{R}_{11} & \hat{R}_{12} \\ \hat{R}_{21} & \hat{R}_{22} \end{pmatrix} \\ &= \begin{pmatrix} \hat{R}_{11}^T & \hat{R}_{21}^T \\ \hat{R}_{12}^T & \hat{R}_{22}^T \end{pmatrix} \begin{pmatrix} \hat{J}_{2j} \hat{R}_{11} & \hat{J}_{2j} \hat{R}_{12} \\ \hat{J}_{2(n-j)} \hat{R}_{21} & \hat{J}_{2(n-j)} \hat{R}_{22} \end{pmatrix}. \end{aligned}$$

Let $(\hat{R}^T\hat{J}\hat{R})_{[2j,2j]}$ denote the leading $2j$-by-$2j$ block of $\hat{R}^T\hat{J}\hat{R}$, we obtain

$$(\hat{R}^T\hat{J}\hat{R})_{[2j,2j]} = \hat{R}_{11}^T\hat{J}_{2j}\hat{R}_{11} + \hat{R}_{21}^T\hat{J}_{2(n-j)}\hat{R}_{21}.$$

Denoting $\tilde{e}_j = (0,\ldots,0,1)^T$ the jth canonical vector of $\mathbb{R}^j$ and $e_1 = (1,0,\ldots,0)^T$, $e_2 = (0,1,0,\ldots,0)^T$ respectively the first and the second canonical vectors of $\mathbb{R}^{2(n-j)}$, then $\hat{R}_{21}$ may be expressed as $\hat{R}_{21} = r_{j+1,n+j}e_1\tilde{e}_{2j}^T$. Hence, $\hat{R}_{21}^T\hat{J}_{2(n-j)}\hat{R}_{21} = r_{j+1,n+j}^2\tilde{e}_{2j}e_1^T\hat{J}_{2(n-j)}e_1\tilde{e}_{2j}^T$. As $\hat{J}_{2(n-j)}e_1 = -e_2$, we obtain directly $\hat{R}_{21}^T\hat{J}_{2(n-j)}\hat{R}_{21} = -r_{j+1,n+j}^2\tilde{e}_{2j}e_1^Te_2\tilde{e}_{2j}^T = 0$. Thus

$$(\hat{R}^T\hat{J}\hat{R})_{[2j,2j]} = \hat{R}_{11}^T\hat{J}_{2j}\hat{R}_{11}.$$

It follows

$$det((\hat{R}^T\hat{J}\hat{R})_{[2j,2j]}) = det(\hat{R}_{11}^T)det(\hat{J}_{2j})det(\hat{R}_{11}).$$

As $det(\hat{J}_{2j}) = 1$, and $det(\hat{R}_{11}^T) = det(\hat{R}_{11}))$, we get

$$det((\hat{R}^T\hat{J}\hat{R})_{[2j,2j]}) = (det(\hat{R}_{11}))^2.$$

Since $P^TA^TJAP = (\hat{R}^T\hat{J}\hat{R})$, it follows that

$$(P^TA^TJAP)_{[2j,2j]} = (\hat{R}^T\hat{J}\hat{R})_{[2j,2j]},$$

which implies for the $2j$-by-$2j$ leading minor of P^TA^TJAP

$$det((P^TA^TJAP)_{[2j,2j]}) = det((\hat{R}^T\hat{J}\hat{R})_{[2j,2j]}) = (det(\hat{R}_{11}))^2.$$

The matrix $\hat{R}_{11}$ is $2j$-by-$2j$ upper triangular matrix, and from relation (12), we have

$$det((P^TA^TJAP)_{[2j,2j]}) = \left[r_{1,1}r_{n+1,n+1}\ldots r_{i,i}r_{n+i,n+i}\ldots r_{j,j}r_{n+j,n+j}\right]^2.$$

Corollary 9. *Let $A \in \mathbb{R}^{2n\times 2n}$ be a nonsingular matrix, and let R be the matrix that one obtains at stage $1 \le j \le n-1$ of the algorithm SRDECO, by executing instructions **1.** to **14.** (corresponding to the current updated matrix A in the process, at stage j and until instruction **14.**). Then A admits an SR decomposition if and only if $r_{n+j,n+j} \neq 0$, $\forall j \in \{1,\ldots,n\}$.*

Proof. Since A is nonsingular and using Threorem 7 and Theorem 8, we have : A admits a SR decomposition if and only if $r_{1,1}r_{n+1,n+1}\ldots r_{j,j}r_{n+j,n+j} \neq 0, \forall j \in \{1,\ldots,n\}$. At the stage j, we have $A = SR$ for some symplectic matrix S. Due to the structure of R, we deduce that the coefficients $r_{1,1}, r_{2,2}, \ldots, r_{j,j}$ are automatically all nonzero (otherwise R would be singular and so would be A). The result is then straightforward.

If the condition A nonsingular is not required, one may ask in this case whether the SR-decomposition exits even when a $2j$-by-$2j$ leading minor $det((P^TA^TJAP)_{[2j,2j]})$ is equal zero for some j. We precise this in the following result

Theorem 10. *Let $A \in \mathbb{R}^{2n\times 2n}$ be any matrix, and let R be the matrix that one obtains at stage $1 \leq j \leq n-1$ of the algorithm SRDECO (or SRMSH), by executing instructions **1.** to **14.** (corresponding to the current updated matrix A in the process, at stage j and until instruction **14.**) Then A admits an SR decomposition if and only if ($r_{n+j,n+j} \neq 0$ or $r_{j+1,n+j} = 0$), $\forall j \in \{1, \ldots, n\}$.*

Proof. The condition is sufficient, since if it is satisfied, the stopping condition in SRDECO (or SRMSH) is never meet and a SR decomposition is furnished at the end of the process. We show now that the condition is necessary, i.e. we show that if there exists an index j such that $r_{n+j,n+j} = 0$ and $r_{j+1,n+j} \neq 0$, then SR decomposition does not exist. In the fact, suppose that there exists an integer $1 \leq j \leq n-1$ such that $r_{n+j,n+j} = 0$ and $r_{j+1,n+j} \neq 0$ and let us seek for a symplectic matrix S_j such that the product $S_ja = a$ for any vector a possessing the same structure of any column $1, \ldots, j$ and $n+1, \ldots, n+j-1$ of R and transforms the $n+j$ th column $R(:, n+j) = \sum_{i=1}^{j+1} r_{i,n+j}e_i + \sum_{i=1}^{j-1} r_{n+i,n+j}e_{n+i}$ into the desired form

$$S_jR(:, n+j) = \sum_{i=1}^{j} r'_{i,n+j}e_i + \sum_{i=1}^{j} r'_{n+i,n+j}e_{n+i}. \tag{13}$$

The matrix S_j has necessarily the form

$$S_j = [e_1, \ldots, e_j, s_{j+1}, \ldots, s_n, e_{n+1}, \ldots, e_{n+j-1}, s_{n+j}, \ldots, s_{2n}],$$

where e_k stands for the kth canonical vector of $\mathbb{R}^{2n}$. Hence we get

$$S_jR(:, n+j) = \sum_{i=1}^{j} r_{i,n+j}e_i + r_{j+1,n+j}s_{j+1} + \sum_{i=1}^{j-1} r_{n+i,n+j}e_{n+i}. \tag{14}$$

In one hand, from relation (13), we get $e_j^TS_jR(:, n+j) = r'_{n+j,n+j}$. In the other hand, from relation (14), and the fact that S_j is symplectic, we get $e_j^TS_jR(:, n+j) = 0$. Thus, we deduce $r'_{n+j,n+j} = 0$. Therefore, the relations (13 - 14), imply $r_{j+1,n+j}s_{j+1}$ belongs to the space spanned by $\{e_1, \ldots, e_j, e_{n+1}, \ldots, e_{n+j-1}\}$. Since the vectors of $e_1, \ldots, e_j, s_{j+1}, e_{n+1}, \ldots, e_{n+j-1}$ are linearly independents, we deduce $r_{j+1,n+j} = 0$, which is absurd. The matrix S_j does not exist and hence SR decomposition does not exist.

Remark 2. Remark that S_j corresponds to the symplectic matrix G_{j+1} for SRDECO and to the symplectic matrix T_j for SRMSH.

4. Curing breakdowns or treating near-breakdowns in JHESS, JHMSH algorithms

4.1. Breakdowns or near-breakdowns in JHESS, JHMSH algorithms

The algorithm SRDECO may be adapted for reducing a matrix to the condensed upper J-Hessenberg form, see [2]. This leads to the algorithm JHESS. In a similar way, the algorithm SRSH or its variant SRMSH may be adapted for handling the reduction of a matrix to J-Hessenberg form, see [12]. We recall that a matrix $H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$, is upper J-Hessenberg when H_{11}, H_{21}, H_{22} are upper triangular and H_{12} is upper Hessenberg. H is called unreduced when H_{21} is nonsingular and the Hessenberg H_{12} is unreduced, i.e. the entries of the subdiagonal are all nonzero.

The algorithm JHESS is formulated in [2] as follows : "given a matrix $A \in \mathbb{R}^{2n\times 2n}$ and $S = I_{2n}$, the following algorithm reduces, if it is possible, A to upper J-Hessenberg form $H = \Pi^{-1}A\Pi$, with a symplectic matrix Π whose first column is a multiple of e_1. A is overwritten by the J-Hessenberg matrix H and S is overwritten by the transforming matrix Π. If this reduction of A does not exist, the algorithm stops".

Algorithm 11. *function [S,A]=JHESS(A)*
1. *For* $j = 1, \ldots, n-1$
2. *For* $k = n, \ldots, j+1$
3. *Zero the entry* $(n+k, j)$ *of* A *by running the function* $[c, s] = vlg(k, A(:, j))$ *and computing* $J_{k,j} = J(k, c, s)$.
4. *Update* $A = J_{k,j}AJ_{k,j}^T$ *and* $S = SJ_{k,j}^T$
5. *End for.*
6. *Zero the entries* $(j+2, j), \ldots, (n, j)$ *of* A *by running the function* $[\beta, w] = vlh(j+1, A(:, j))$ *and computing* $H_j = H(j+1, w)$.
7. *Update* $A = H_jAH_j^T$ *and* $S = SH_j^T$.
8. *If the entry* $(j+1, j)$ *of* A *is nonzero and the entry* $(n+j, j)$ *is zero then stop the algorithm*
9. *else*
10. *Zero the entry* $(j+1, j)$ *of* A *by running the function* $[\nu] = gal(A(:, j))$
11. *Compute* $G_{j+1} = G(j+1, \nu)$.

12. *Update* $A = G_{j+1} A G_{j+1}^{-1}$ *and* $S = S G_{j+1}^{-1}$.
13. *End if*
14. *For* $k = n, \ldots, j+1$
15. *Zero the entry* $(n+k, n+j)$ *of* A *by running the function* $[c, s] = vlg(k, A(:, n+j))$ *and compute* $J_{k,n+j} = J(k, c, s)$.
16. *Update* $A = J_{k,n+j} A J_{k,n+j}^T$ *and* $S = S J_{k,n+j}^T$.
17. *End for.*
18. *If* $j \leq n-2$
19. *Zero the entries* $(j+2, n+j), \ldots, (n, n+j)$ *of* A *by running the function* $[\beta, w] = vlh(j+1, A(:, n+j))$ *and compute* $H_{n+j} = H(j+1, w)$.
20. *Update* $A = H_{n+j} A H_{n+j}^T$ *and* $S = S H_{n+j}^T$.
21. *End if*
22. *End for.*

One of the main drawback of JHESS is that a fatal breakdown can be encountered. To illustrate our purpose, we consider the following example. Let A_6 be the 6-by-6 matrix

$$A_6 = \begin{pmatrix} 1 & 0 & 0 & 1 & 2 & 0 \\ 2 & 1 & 0 & 2 & 1 & 0 \\ 0 & 2 & 1 & 0 & 2 & 1 \\ 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 & 1 & 0 \\ 0 & 0 & 1 & 0 & 3 & 1 \end{pmatrix}. \tag{15}$$

The algorithm JHESS, applied to A_6, meets a fatal breakdown at the first step : the entry $A_6(2,1) \neq 0$ and the entry $A_6(4,1) = 0$, the algorithm stops. In fact, it is impossible to find a symplectic matrix S_1, with the first column proportional to e_1 such that $S A_6 e_1 = \alpha e_1 + \beta e_4$, as showed in the above subsection. Thus, A_6 can not be reduced to an upper J-Hessenberg form, via symplectic similarity transformations, for which the first column is proportional to e_1. The SR-algorithm as described in [2] , uses first JHESS algorithm for reducing a matrix to the J-Hessenberg form. As consequence, if applied to A_6, the SR-algorithm stops also at the first step. Let us remark also that the basic SR-algorithm (which can be roughly described as consisting in repeating the factorisation $A = SR$, and the product $A = RS$) works and converges, when applied to the example A_6. The algorithm JHESS may also suffers from another serious problem : the near-breakdown. The latter occurs when the condition number of the symplectic and non-orthogonal

matrix G_{j+1} at the step **11.** of the algorithm JHESS becomes very large. This causes a dramatic growth of the rounding errors.

The following strategy is proposed in [2] for remedying to such problems : if in the jth iteration, the condition number of the matrix G_{j+1} is larger than a certain tolerance, the iteration is stopped. In the implicit form (which is the useful one) of the algorithm SR, an exceptional similarity transformation is computed, with the symplectic (but non-orthogonal) matrix $S_j = I - ww^T J$, where w is a random vector with $\|w\|_2 = 1$. The algorithm JHESS is then applied to the new similar matrix $S_j^{-1} A S_j$. If the number of encountered near-breakdowns/breakdowns exceeds a given bound, the whole process is definitively stopped. This strategy presents certain serious drawbacks : 1) The condition number of $S_j^{-1} A S_j$ will be worse than the condition number of A. This du to the fact that S_j can never be orthogonal. Hence, numerical instability is expected. 2) The cost of forming the product $S_j^{-1} A S_j$ is $O(n^2)$ where $2n$ is the dimension of A. 3) The product $S_j^{-1} A S_j$ fills-up the matrix and destroys the previous partially created J-Hessenberg form of A. Hence an additional cost of $O(n^3)$ is needed to restore the J-Hessenberg form. To summarize, each application of this strategy creates a current matrix with worse condition number than the previous, and needs an expensive cost of $O(n^3)$ flops.

In the sequel, we propose two alternatives, for which either all or some of the above drawbacks are avoided. The first consists in a careful choice of the random vector w so that the product $S_j^{-1} A S_j$ does not fill-up the matrix and preserves all the created zeros during the previous steps $1, \ldots, j-1$. This diminish considerably the cost. However, the condition number of $S_j^{-1} A S_j$ may become worse than this of A. The second alternative is more attractive since it allows us to avoid all of the above drawbacks. It consists in computing a similarity transformation $S_j^{-1} A S_j$ for which : 1) the proposed matrix S_j is not only symplectic but also orthogonal. Thus, the condition number of $S_j^{-1} A S_j$ remains the same, and the process is numerically as accurate as possible. 2) The cost for computing the product $S_j^{-1} A S_j$ is only $O(n)$. Thus, a gain of an order-of- magnitude is guaranteed. 3) The product $S_j^{-1} A S_j$ does not fills-up the matrix and preserves all the created zeros in previous steps. Also, to restore the J-Hessenberg form of A, only a cheaper additional cost of $O(n^2)$ is needed.

In the sequel, we explain first how one may remedy to the fatal breakdown, encountered by JHESS, when applied to the example A_6 and highlights

the main lines of the method. Then we present a method to cure the fatal breakdown in the general case. The idea is the following : one seeks for a symplectic transforming matrix S so that the similar matrix SA_6S^{-1}, may be reduced by JHESS. The choice of S should be done carefully. A judicious choice of S consists in taken S equal to Van Loan's Householder matrix

$$S = \begin{pmatrix} H_2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & H_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{16}$$

or Van Loan's Givens matrix

$$S = \begin{pmatrix} G_2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & G_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{17}$$

where H_2 (respectively G_2) is a 2-by-2 Householder matrix (respectively a 2-by-2 Givens matrix) such that $H_2(1,2)^T = \sqrt{5}(1,0)^T$ (respectively $G_2(1,2)^T = \sqrt{5}(1,0)^T$). If we proceed with choices (16) or (17), we get the first column of SA_6 proportional to e_1 and only rows 1,2 and 4,5 of SA_6 may change. With the choice (17), we obtain $G_2 = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$, with $c = 1/\sqrt{5}$, $s = 2c$, and

$$SA_6 = \begin{pmatrix} \sqrt{5} & 2/\sqrt{5} & 0 & \sqrt{5} & 4/\sqrt{5} & 0 \\ 0 & 1/\sqrt{5} & 0 & 0 & -3/\sqrt{5} & 0 \\ 0 & 2 & 1 & 0 & 2 & 1 \\ 0 & 4/\sqrt{5} & 4/\sqrt{5} & 7/\sqrt{5} & 2/\sqrt{5} & 0 \\ 0 & -3/\sqrt{5} & 2/\sqrt{5} & 1/\sqrt{5} & 1/\sqrt{5} & 0 \\ 0 & 0 & 1 & 0 & 3 & 1 \end{pmatrix}.$$

The multiplication of SA_6 on the left by S^{-1} acts only on the columns 1, 2 and 4, 5 of SA_6. The other columns remain unchanged. We obtain

$$SA_6S^{-1} = \begin{pmatrix} 9/5 & -8/5 & 0 & 13/5 & -6/5 & 0 \\ 2/5 & 1/5 & 0 & -6/5 & 3/5 & 0 \\ 4/\sqrt{5} & 2/\sqrt{5} & 1 & 4/\sqrt{5} & 2/\sqrt{5} & 1 \\ 8/5 & 4/5 & 4/\sqrt{5} & 11/5 & {}_12/5 & 0 \\ -6/5 & -3/5 & 2/\sqrt{5} & 3/5 & -1/5 & 0 \\ 0 & 0 & 1 & 6/\sqrt{5} & 3/\sqrt{5} & 1 \end{pmatrix}. \tag{18}$$

We applied JHESS (also JHMSH) to the matrix SA_6S^{-1} of (18). The algorithm run well and the reduction to the J-Hessenberg form is obtained. The SR-algorithm is then applied with explicit and implicit versions, and both converge. Thus the fatal breakdown of JHESS (or similarly JHMSH) is cured. Recall that the algorithm JHMSH as described in [12] is as follows

Algorithm 12. *function [S,H]=JHMSH(A)*

$twon = size(A(:,1));\ n = twon/2;\ S = eye(twon);$

for $j = 1 : n-1$

$J = [zeros(n-j+1), eye(n-j+1); -eye(n-j+1), zeros(n-j+1)];$

$ro = [j : n, n+j : 2n];\ co = [j : n, n+j : 2n];$

$[c, v] = osh2(A(ro, j));$

% Updating A :

$A(ro, co) = A(ro, co) + c * v * (v' * J * A(ro, co));$

$A(:, co) = A(:, co) - (A(:, co) * (c * v)) * v' * J;$

% Updating S (if needed) :

$S(:, co) = S(:, co) - c * (v * v') * J * S(:, co);$

for $k = 2n : n+j+1,$

$[c, s] = vlg(k, A(:, n+j)),$

%Updating A :

$$\begin{bmatrix} A(k, co) \\ A(n+k, co) \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} A(k, co) \\ A(n+k, co) \end{bmatrix};$$

$$\begin{bmatrix} A(:, k) & A(:, n+k) \end{bmatrix} = \begin{bmatrix} A(:, k) & A(:, n+k) \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix};$$

%Updating S (if needed) :

$$\begin{bmatrix} S(:, k) & S(:, n+k) \end{bmatrix} = \begin{bmatrix} S(:, k) & S(:, n+k) \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix};$$

end

if $j \le n-2$

$[\beta, w] = vlh(j+1, A(:, n+j));$

%Updating A :

$A(j+1 : n, co) = A(j+1 : n, co) - \beta * w * w' * A(j+1 : n, co)$

$A(j+1+n : 2n, co) = A(j+1+n : 2n, co) - \beta * w * w' * A(j+1+n : 2n, co);$

$A(:, j+1 : n) = A(:, j+1 : n) - \beta * A(:, j+1 : n) w * w';$

$A(:, n+j+1 : 2n) = A(:, n+j+1 : 2n) - \beta * A(:, n+j+1 : n) w * w';$

%Updating S (if needed) :

$S(:, j+1 : n) = S(:, j+1 : n) - \beta * S(:, j+1 : n) w * w';$

$S(:, n+j+1 : 2n) = S(:, n+j+1 : 2n) - \beta * S(:, n+j+1 : n) w * w';$

end
end
end

The breakdown in JHMSH occurs exactly in the same conditions as in JHESS, and is located in the call of the function $osh2$. A slight different version of JHMSH is JHMSH2 (see [12]).

4.2. Curing breakdowns in JHESS, JHMSH algorithms

We present here, in a general manner, the strategy of curing breakdowns or near breakdowns which may occur in JHESS or JHMSH algorithms. Let us remark that breakdowns (or near-breakdowns) in JHESS (respectively in JHMSH) may occur only when the function gal (respectively $osh2$) is called, and hence it concerns only columns from the first half of the current matrix.

Let $A \in \mathbb{R}^{2n\times 2n}$ be a matrix and let H be the matrix that one obtains at stage $1 \leq j \leq n-1$ of the algorithm JHESS, by executing instructions **1.** to **7.** (corresponding to the current updated matrix A in the process, at stage j and until instruction **7.**). Partitioning $H = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}$, then $H_{11},\ H_{12},\ H_{21},\ H_{22}$ have the form

$$H_{11} = \begin{bmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,j} & h_{1,j+1} & \dots & h_{1,n} \\ 0 & h_{2,2} & \dots & h_{2,j} & h_{2,j+1} & \dots & h_{2,n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \dots & \vdots \\ 0 & & 0 & h_{j,j} & h_{j,j+1} & \dots & h_{jn} \\ 0 & \dots & 0 & h_{j+1,j} & h_{j+1,j+1} & \dots & h_{j+1,n} \\ 0 & \dots & 0 & 0 & h_{j+2,j+1} & \dots & h_{j+2,n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & h_{n,j+1} & \dots & h_{n,n} \end{bmatrix},$$

$$H_{12} = \begin{bmatrix} h_{1,n+1} & h_{1,n+2} & \dots & h_{1,n+j-1} & h_{1,n+j} & \dots & h_{1,2n} \\ h_{2,n+1} & h_{2,n+2} & \dots & h_{2,n+j-1} & h_{2,n+j} & \dots & h_{2,2n} \\ \vdots & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & & \ddots & h_{j-1,n+j-1} & h_{j-1,n+j} & \dots & h_{j-1,2n} \\ 0 & \dots & \dots & h_{j,n+j-1} & h_{j,n+j} & \dots & h_{j,2n} \\ 0 & \dots & \dots & 0 & h_{j+1,n+j} & \dots & h_{j+1,2n} \\ 0 & \dots & \dots & 0 & h_{j+2,n+j} & \dots & h_{j+2,2n} \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & & \dots & 0 & h_{n,n+j} & \dots & h_{n,2n} \end{bmatrix},$$

$$H_{21} = \begin{bmatrix} h_{n+1,1} & h_{n+1,2} & \dots & h_{n+1,j} & h_{n+1,j+1} & \dots & h_{n+1,n} \\ 0 & \ddots & \ddots & \vdots & \vdots & \dots & \vdots \\ \vdots & \ddots & \ddots & h_{n+j-1,j} & h_{n+j-1,j+1} & \dots & h_{n+j-1,n} \\ \vdots & & \ddots & h_{n+j,j} & h_{n+j,j+1} & \dots & h_{n+j,n} \\ 0 & \dots & \dots & 0 & h_{n+j+1,j+1} & \dots & h_{n+j+1,n} \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & \dots & \dots & 0 & h_{2n,j+1} & \dots & h_{2n,n} \end{bmatrix},$$

$$H_{22} = \begin{bmatrix} h_{n+1,n+1} & h_{n+1,n+2} & \dots & h_{n+1,n+j-1} & h_{n+1,n+j} & \dots & h_{n+1,2n} \\ 0 & h_{n+2,n+2} & \dots & h_{n+2,n+j-1} & h_{n+2,n+j} & \dots & h_{n+2,2n} \\ 0 & 0 & \ddots & & \vdots & & \vdots \\ \vdots & \vdots & \ddots & h_{n+j-1,n+j-1} & h_{n+j-1,n+j} & \dots & h_{n+j-1,2n} \\ 0 & \dots & \dots & 0 & h_{n+j,n+j} & \dots & h_{n+j,2n} \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & \dots & \dots & 0 & h_{2n,n+j} & \dots & h_{2n,2n} \end{bmatrix}.$$

The breakdown occurs in JHESS when the coefficient $h_{n+j,j} = 0$ and $h_{j+1,j} \neq 0$. In this case, JHESS stops computations. To overcome this fatal breakdown, we construct the orthogonal matrix $P^{(j)} = diag(I_{j-1}, H_2^{(j)}, I_{n-j-1})$ where $H_2^{(j)}$ is a 2-by-2 Householder matrix. We set $S^{(j)} = diag(P^{(j)}, P^{(j)})$. The matrix $S^{(j)}$ is symplectic and orthogonal. The choice of the 2-by-2 Householder matrix $H_2^{(j)}$ is so that $H_2^{(j)} \begin{pmatrix} h_{j,j} \\ h_{j+1,j} \end{pmatrix} = \begin{pmatrix} h'_{j,j} \\ 0 \end{pmatrix}$. Thus, the action $S^{(j)}H$ annihilates the position $(j+1, j)$ of the updated matrix H and keep unchanged all zeros created previously except potentially the position $(j+1, n+j-1)$ (in the block H_{12})). Keep in mind that the action of $S^{(j)}H$ on H affects only

rows $j, j+1, n+j, n+j+1$. The action $S^{(j)}H[S^{(j)}]^{-1}$ on $S^{(j)}H$ affects only the columns $j, j+1, n+j, n+j+1$. Thus, the columns $1, \ldots, j-1$ and $n+1, \ldots, n+j-1$ remain unchanged (hence all zeros created previously are not affected). The process is pursued as follows : one must annihilate the potentially nonzero entry in position $(j+1, n+j-1)$ and keep unchanged all zeros created previously in columns $1, \ldots, j-1$ and $n+1, \ldots, n+j-1$. This may be addressed by applying the similarity $T^{(j)}H[T^{(j)}]^{-1}$ to the obtained matrix H, where $T^{(j)} = diag(Q^{(j)}, Q^{(j)})$ states for Van Loan's Householder matrix, given by $Q^{(j)} = diag(I_{j-1}, K_2^{(j)}, I_{n-j-1})$ where $K_2^{(j)}$ is a 2-by-2 Householder matrix. The choice of the 2-by-2 Householder matrix $K_2^{(j)}$ is so that $K_2^{(j)} \begin{pmatrix} h_{j,n+j-1} \\ h_{j+1,n+j-1} \end{pmatrix} = \begin{pmatrix} h'_{j,n+j-1} \\ 0 \end{pmatrix}$. The cost of this curing strategy step is $O(n)$. Next, the algorithm is pursued normally by executing again step j.

Remark 3. The 2-by-2 Householder matrix $H_2^{(j)}$ (respectively $K_2^{(j)}$) may be replaced by a 2-by-2 Givens matrix $G_2^{(j)} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$ (respectively by a 2-by-2 Givens $L_2^{(j)} = \begin{pmatrix} c' & s' \\ -s' & c' \end{pmatrix}$) where the coefficient c, s (respectively c', s') are chosen so that $G_2^{(j)} \begin{pmatrix} h_{j,j} \\ h_{j+1,j} \end{pmatrix} = \begin{pmatrix} h'_{j,j} \\ 0 \end{pmatrix}$ (respectively $L_2^{(j)} \begin{pmatrix} h_{j,n+j-1} \\ h_{j+1,n+j-1} \end{pmatrix} = \begin{pmatrix} h'_{j,n+j-1} \\ 0 \end{pmatrix}$).

It is worth noting that one may take an arbitrary k-by-k Householder matrix $H_k^{(j)}$ instead of $H_2^{(j)}$, with $P^{(j)} = diag(I_{j-1}, H_k^{(j)}, I_{n-j-k+1})$. The action $P^{(j)}H$ on H affects only rows $j, \ldots, j+k-1$, rows $n+j, \ldots, n+j+k-1$. The action $P^{(j)}H[P^{(j)}]^{-1}$ on $P^{(j)}H$ affects only columns $j, \ldots, j+k-1$, and columns $n+j, \ldots, n+j+k-1$. Hence, all zeros created previously in columns $1, \ldots, j-1$ and columns $n+1, \ldots, n+j-1$ remain unchanged, except potentially in positions $(j+1, n+j-1), \ldots, (j+k-1, n+j-1)$. To pursue the process, one must annihilate the potentially nonzero entries in position $(j+1, n+j-1), \ldots, (j+k-1, n+j-1)$ and keep unchanged all zeros created previously in columns $1, \ldots, j-1$ and $n+1, \ldots, n+j-1$. This may be addressed by applying the similarity $T^{(j)}H[T^{(j)}]^{-1}$ to the obtained matrix H, where $T^{(j)} = diag(Q^{(j)}, Q^{(j)})$ states for Van Loan's Householder matrix, given by $Q^{(j)} = diag(I_{j-1}, K_k^{(j)}, I_{n-j-1})$ where $K_k^{(j)}$ is a k-by-k Householder matrix. The choice of the k-by-k Householder matrix $K_k^{(j)}$ is so that

$K_k^{(j)} \begin{pmatrix} h_{j,n+j-1} \\ \vdots \\ h_{j+k-1,n+j-1} \end{pmatrix} = \begin{pmatrix} h'_{j,n+j-1} \\ 0 \end{pmatrix}$. The cost of this curing strategy step is $O(kn)$.

4.3. Curing near-breakdowns in JHESS, JHMSH algorithms

The near-breakdown occurs in JHESS (or in JHMSH) when the coefficients $h_{n+j,j}$ and $h_{j+1,j}$ are both different from zero but are near to the situation of breakdown. This can be measured by the fact that the ratio $\dfrac{h_{j+1,j}}{h_{n+j,j}}$ is very large. In this case, the non-orthogonal and symplectic transformations involved in JHESS (respectively JHMSH) become ill-conditioned and numerical instability is encountered reducing the accuracy of the reduction. In order to remedy to a such near breakdown in JHESS (or JHMSH) algorithm, one may proceed exactly as for curing a breakdown, the only difference is that the test $h_{n+j,j} = 0$ and $h_{j+1,j} \neq 0$ (corresponding to a breakdown) is replaced by the $\dfrac{h_{j+1,j}}{h_{n+j,j}} \geq \tau$ (corresponding to a near-breakdown), where τ is a certain tolerance.

4.4. SR algorithm

The SR algorithm, is a QR like algorithm which can roughly be described as follows. For a given matrix $M \in \mathbb{R}^{2n\times 2n}$, it computes :
1. An upper J-Hessenberg reduction $M_1 = S_0^J M S_0$ where S_0 is symplectic. Set $S = S_0$.
2. Iteration : For $k = 1, \ldots,$ compute $M_{k+1} = S_k^J M_k S_k$ where S_k stands for the symplectic factor of the SR decomposition $p_k(M_k) = S_k R_k$ of a polynomial p_k of M_k and update $S = SS_k$.
The iterate M_{k+1} remain J-Hessenberg if the matrix M_k is J-Hessenberg. Like the QR algorithm, SR algorithm admits an implicit version : the decompositions $p_k(M_k) = S_k R_k$ are not performed explicitly. Since SR algorithm is based on J-Hessenberg reductions and SR decompositions, breakdowns or near-breakdowns may be encountered both in the explicit or implicit versions of the algorithm. Of course, the implicit form is preferred to the explicit one. In [2] there is no strategy proposed when a breakdown is meet. The algorithm is topped. However, a technique has been proposed in the situation of a near breakdown, occurring at the iteration j.

The proposed technique consists in computing the similarity $M_{j+1} = S_j M_j [S_j]^{-1}$, where $S_j = I - ww^T J$ and $w \in \mathbb{R}^{2n \times 2n}$ is a random vector, with $\|w\|_2 = 1$. The algorithm continue with M_{j+1}. This technique presents several serious drawbacks :
1. The symplectic transformation $(I - ww^T J)$ is never orthogonal (except for $w = 0$) and hence its condition number may be large. The condition number of M_{j+1} could be worse than this of M_j.
2. Computing the similarity $M_{j+1} = S_j M_j [S_j]^{-1}$ costs $O(n^2)$.
3. The similarity $M_{j+1} = S_j M_j [S_j]^{-1}$ destroys the structure J-Hessenberg of M_j. Thus M_j is no longer J-Hessenberg. Moreover, the matrix M_{j+1} fills up. Hence, a reduction to a J-Hessenberg form is needed to restore the previous structure. The cost of this restoration is $O(n^3)$ which is very expensive. Instead of this similarity, when breakdown or near breakdown occurs with respect to the column say i of the matrix M_j, we propose the similarity $M_{j+1} = P^{(j)} M_j [P^{(j)}]^{-1}$, where $P^{(j)} = diag(I_{i-1}, H_l^{(i)}, I_{n-i-l+1})$ and $H_l^{(i)}$ an arbitrary l-by-l Householder matrix. The action $P^{(j)} M_j$ on M_j affects only rows $i, \ldots, i+l-1$, rows $n+i, \ldots, n+i+l-1$. The action $P^{(j)} M_j [P^{(j)}]^{-1}$ on $P^{(j)} M_j$ affects only columns $i, \ldots, i+l-1$, and columns $n+i, \ldots, n+i+l-1$. Hence, all zeros in columns $1, \ldots, i-1$ and columns $n+1, \ldots, n+i-1$ (because of the form J-Hessenberg of M_j) remain unchanged, except potentially in positions $(i+1, n+i-1), \ldots, (i+l-1, n+i-1)$. To pursue the process, one restores the J-Hessenberg form. The advantage of this similarity is that first $P^{(j)}$ is symplectic and orthogonal (hence it is stable), preserves most of the created zeros because of the J-Hessenberg structure of M_j and the cost of restoring the J-Hessenberg form of M_j do not exceed $O(in)$. Unlike QR algorithm, SR algorithm still needs profound investigations. This will be the aim of a forthcoming work.

5. Numerical experiments

To illustrate our purpose, we consider the following numerical example. Let A be the 12-by-12 matrix

$$A = \begin{pmatrix} 1 & 5 & 7 & 9 & 5 & 1 & 1 & 3 & 1 & 3 & 7 & 2 \\ 0 & 1 & 4 & 6 & 1 & 2 & 2 & 1 & 5 & 4 & 3 & 5 \\ 0 & 0 & 1 & 2 & 3 & 2 & 0 & 0 & 1 & 2 & 5 & 3 \\ 0 & 0 & 2 & 1 & 9 & 8 & 0 & 0 & 2 & 1 & 2 & 4 \\ 0 & 0 & 0 & 2 & 1 & 3 & 0 & 0 & 5 & 2 & 1 & 2 \\ 0 & 0 & 0 & 4 & 2 & 1 & 0 & 0 & 4 & 3 & 2 & 1 \\ 1 & 4 & 7 & 2 & 1 & 3 & 1 & 7 & 6 & 1 & 6 & 7 \\ 0 & 1 & 9 & 3 & 5 & 1 & 0 & 1 & 4 & 5 & 8 & 3 \\ 0 & 0 & 0 & 2 & 7 & 9 & 0 & 0 & 1 & 3 & 4 & 5 \\ 0 & 0 & 0 & 1 & 2 & 8 & 0 & 0 & 3 & 1 & 7 & 3 \\ 0 & 0 & 0 & 2 & 1 & 2 & 0 & 0 & 4 & 3 & 1 & 2 \\ 0 & 0 & 0 & 9 & 3 & 1 & 0 & 0 & 1 & 2 & 3 & 1 \end{pmatrix}.$$

Following the steps of the algorithms JHESS, JHMSH and JHMSH2, one remarks that the condition of a breakdown is fulfilled at the beginning of the step $j = 3$ for all of them. Let us call MJHESS (respectively JHM^{2}SH and JHM^{2}SH2) the modified algorithm JHESS (respectively JHMSH and JHMSH2) obtained by applying our strategy for curing breakdowns. We obtain the following numerical results, showing the efficiency of the method. Thus the J-orthogonality is numerically preserved up to the machine precision for MJHESS (respectively JHM^{2}SH and JHM^{2}SH2). It is worth noting that preserving the J-orthogonality is crucial for SR-algorithm in order to get accurate eigenvalues and vectors of a matrix.

$2n$	Loss of J-Orthogonality $\left\|I - S^J S\right\|_2$			
	$JHESS$	$MJHESS$	JHM^2SH	JHM^2SH2
12	fails	$1.8553e-15$	$5.0842e-15$	$6.6428e-15$

One observes also that the error in the reduction to J-Hessenberg form is very satisfactory for MJHESS (respectively JHM^{2}SH and JHM^{2}SH2). Notice that the algorithm JHESS as given in [2], applied to the matrix A, without our strategy for curing breakdown, simply fails to perform a reduction to a J-Hessenberg form.

$2n$	Reduction error $\|A - SHS^{-1}\|_2$			
	$JHESS$	$MJHESS$	JHM^2SH	JHM^2SH2
12	fails	$3.2709e-14$	$3.8777e-13$	$2.7653e-13$

Let $A = S_i H_i S_i^{-1},\ i = 1, 2, 3$ be the J-Hessenberg reduction obtained respectively by the algorithms MJHESS, JHM^{2}SH and JHM^{2}SH2. It is known (see [2]) that there exist $D_1,\ D_2,\ D_3$ such that $S_1 D_1 = S_2,\ D_1^{-1} H_1 D_1 = H_2,\ S_1 D_2 = S_3,\ D_2^{-1} H_1 D_2 = H_3,\ S_2 D_3 = S_3$, and $D_3^{-1} H_2 D_3 = H_3$, with each matrix $D_i = \begin{pmatrix} C_i & F_i \\ 0 & C_i^{-1} \end{pmatrix}$, where C_i and F_i are diagonals.

We obtain numerically $C_1 = diag(1, 1, 0.4113, 0.3688, 0.7747, 0.6638)$ and $F_1 = diag(0, 0, -2.3962, -2.3371, 1.2880, -1.9982)$, and

$$\|D_1^{-1} H_1 D_1 - H_2\|_2 == 5.3417e-13.$$

Also, we have
$C_2 = (1, 1, 0.4113, 0.3688, 0.7747, 0.6638),\ F_2 = diag(0, 0, -2.3962, -2.3371, 1.2880, -1.9982)$, and

$$\|D_2^{-1} H_1 D_2 - H_3\|_2 = 3.7881e-13,$$

and finally

$$D_3 = I_{12},$$

where I_{12} stands for identity matrix of size 12, with

$$\|D_3^{-1} H_2 D_3 - H_3\|_2 = 9.4799e-13.$$

Thus the matrices D_i have numerically the desired forms and as expected, the algorithms JHM^{2}SH and JHM^{2}SH2 perform quite the same results.

6. Conclusions

In this work, we linked the necessary and sufficient condition of the existence of a SR-decomposition with the computations during the process, of some coefficients of the current matrix. The SR-decomposition is intimately related to the J-Hessenberg reduction via the algorithm JHESS. The later (also JHMSH and its different variants) may encounter fatal breakdowns or suffer from near-breakdowns. We derive efficient strategies for treating them. The numerical experiments show the efficiency of these strategies.

[1] E. Artin, *Geometric Algebra*, Interscience Publishers, New York, 1957.

[2] A. Bunse-Gerstner and V. Mehrmann, A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation, IEEE Trans. Automat. Control **AC-31** (1986), 1104–1113.

[3] A. Bunse-Gerstner, Matrix factorizations for symplectic QR-like methods, Linear Algebra Appl. **83** (1986), 49–77.

[4] J. Della-Dora, Numerical linear algorithms and group theory, Linear Algebra Appl. **10** (1975), 267–283.

[5] L. Elsner, On some algebraic problems in connection with general eigenvalue algorithms, Linear Alg. Appl., 26 :123-38 (1979).

[6] G. Golub and C. Van Loan, *Matrix Computations*, third ed., The Johns Hopkins U.P., Baltimore, 1996.

[7] C. Paige and C. Van Loan, A Schur decomposition for Hamiltonian matrices, Linear Algebra Appl. **41** (1981), 11–32.

[8] A. Salam, On theoretical and numerical aspects of symplectic Gram-Schmidt-like algorithms, Numer. Algo., **39** (2005), 237-242.

[9] A. Salam, A. El Farouk, E. Al-Aidarous, Symplectic Householder Transformations for a QR-like decomposition, a Geometric and Algebraic Approaches, J. of Comput. and Appl. Math., Vol. 214, Issue 2, 1 May 2008, Pages 533-548.

[10] A. Salam and E. Al-Aidarous and A. Elfarouk, Optimal symplectic Householder transformations for SR-decomposition, Linear Algebra and Its Appl., 429 (2008), no. 5-6, 1334-1353.

[11] A. Salam, E. Al-Aidarous, Error analysis and computational aspects of SR factorization, via optimal symplectic Householder Transformations, Electronic Trans. on Numer. Anal., Vol. 33, pp. 189-206, 2009.

[12] A. Salam and H. Ben Kahla, An upper J- Hessenberg reduction of a matrix through symplectic Householder transformations, submitted.

[13] C. Van Loan, A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix, Linear Algebra Appl. **61** (1984), 233–251.

[14] D.S. Watkins, *The Matrix Eigenvalue Problem : GR and Krylov subspace methods*, SIAM, 2007.

[15] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England.

Bibliographie

[1] Nabih N. Abdelmalek. Round off error analysis for Gram–Schmidt method and solution of linear least squares problems. *BIT Numerical Mathematics*, 11(4) :345–367, December 1971. 45

[2] Gregory S. Ammar, Peter Benner, and Volker Mehrmann. A multishift algorithm for the numerical solution of algebraic Riccati equations. *Electronic Transactions on Numerical Analysis*, 1 :33–48, September 1993. 3

[3] Gregory S. Ammar, Christian Mehl, and Volker Mehrmann. Schur-like forms for matrix Lie groups, Lie algebras and Jordan algebras. *Linear algebra and its applications*, 287(1-3) :11–39, 15 January 1999. 2

[4] Gregory S. Ammar and Volker Mehrmann. On Hamiltonian and symplectic Hessenberg forms. *Linear algebra and its applications*, 149 :55–72, 15 April 1991. 3

[5] Emil Artin. *Geometric algebra*, volume 3 of *Interscience tracts in pure and applied mathematics*. Interscience Publishers, Inc., New York-London, New York, NY, USA, 1957. 2, 11, 35

[6] Michael Athans and Peter L. Falb. *Optimal control : an introduction to the theory and its applications*. McGraw-Hill Book Co., New York-Toronto, Ont.-London, 1966. 1

[7] Jesse Louis Barlow and Alicja Smoktunowicz. Reorthogonalized block classical Gram–Schmidt. *Numerische Mathematik*, 123(3) :395–423, March 2013. 35, 45

[8] Skander Belhaj. A fast method to block-diagonalize a Hankel matrix. *Numerical Algorithms*, 47(1) :15–34, 01 January 2008. 233

[9] Skander Belhaj. Block factorization of Hankel matrices and Euclidean algorithm. *Mathematical Modelling of Natural Phenomena*, 5(7) :48–54, 26 August 2010. 235

[10] Skander Belhaj. Computing the block factorization of complex Hankel matrices. *Computing*, 87(3-4) :169–186, 01 May 2010. 233

[11] Skander Belhaj. Computing the polynomial remainder sequence via Bézout matrices. *Journal of Computational and Applied Mathematics*, 250(Supplement C) :244–255, 01 October 2013. 230, 235

[12] Skander Belhaj and Haithem Ben Kahla. On the complexity of computing the GCD of two polynomials via Hankel matrices. *ACM Communications in Computer Algebra*, 46(3/4) :74–75, 15 January 2013. 230, 235

[13] Skander Belhaj, Haithem Ben Kahla, Marwa Dridi, and Maher Moakher. Blind image deconvolution via Hankel based method for computing the GCD of polynomials. *Mathematics and Computers in Simulation*, 144(Supplement C) :138 – 152, February 2018. 4, 228, 229

[14] Skander Belhaj and Marwa Dridi. A note on computing the inverse of a triangular Toeplitz matrix. *Applied Mathematics and Computation*, 236(Supplement C) :512–523, 01 June 2014. 230, 234

[15] Peter Benner, Matthias Bollhöfer, Daniel Kressner, Christian Mehl, and Tatjana Stykel. *Numerical Algebra, Matrix Theory, Differential-Algebraic Equations and Control Theory : Festschrift in honor of Volker Mehrmann*. Springer International Publishing Switzerland, first edition, 2015. 1, 34

[16] Peter Benner, Ralph Byers, Heike Faßbender, Volker Mehrmann, and David S. Watkins. Cholesky-like factorizations of skew-symmetric matrices. *Electronic Transactions on Numerical Analysis*, 11(1) :85–93, 2000. 2

[17] Peter Benner and Cedric Effenberger. A rational SHIRA method for the Hamiltonian eigenvalue problem. *Taiwanese Journal of Mathematics*, 14(3A) :805–823, June 2010. 3

[18] Peter Benner and Heike Faßbender. An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem. *Linear Algebra and its Applications*, 263 :75–111, 15 September 1997. 1, 3, 34, 35

[19] Peter Benner and Heike Faßbender. An implicitly restarted symplectic Lanczos method for the symplectic eigenvalue problem. *SIAM Journal on Matrix Analysis and Applications*, 22(3) :682–713, 2001. 34, 35

[20] Peter Benner, Daniel Kressner, and Volker Mehrmann. Structure preservation : A challenge in computational control. *Future Generation Computer Systems*, 19(7) :1243–1252, October 2003. 1

[21] Peter Benner, Volker Mehrmann, and Hongguo Xu. A new method for computing the stable invariant subspace of a real Hamiltonian matrix. *Journal of computational and applied mathematics*, 86(1) :17–43, 28 November 1997. 3

[22] Peter Benner, Volker Mehrmann, and Hongguo Xu. A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils. *Numerische Mathematik*, 78(3) :329–358, January 1998. 3

[23] Dario Andrea Bini and Paola Boito. Structured matrix based methods for polynomial ϵ-gcd : analysis and comparisons. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, ISSAC '07, pages 9–16, New York, NY, USA, 2007. ACM. 230

[24] Dario Andrea Bini and Luca Gemignani. Fast parallel computation of the polynomial remainder sequence via Bézout and Hankel matrices. *SIAM Journal on Computing*, 24(1) :63–77, 01 February 1995. 230

[25] Dario Andrea Bini and Luca Gemignani. Fast fraction-free triangularization of Bezoutians with applications to sub-resultant chain computation. *Linear Algebra and its Applications*, 284(1–3) :19–39, 15 November 1998. 230

[26] Dario Andrea Bini and Victor Y. Pan. *Polynomial and Matrix Computations (Vol. 1) : Fundamental Algorithms*. Birkhäuser Verlag, Basel, Switzerland, Switzerland, 1994. 230, 232

[27] Åke Björck. Solving linear least squares problems by Gram–Schmidt orthogonalization. *BIT Numerical Mathematics*, 7(1) :1–21, March 1967. 35, 45

[28] Åke Björck. Numerics of Gram–Schmidt orthogonalization. *Linear Algebra and Its Applications*, 197 :297–316, January/February 1994. 35

[29] Åke Björck. *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 1996. 35, 45

[30] Åke Björck and Christopher C. Paige. Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm. *SIAM Journal on Matrix Analysis and Applications*, 13(1) :176–190, January 1992. 35, 45, 86

[31] Thierry Braconnier, Philippe Langlois, and Jean-Christophe Rioual. The influence of orthogonality on the Arnoldi method. *Linear Algebra and its Applications*, 309(1-3) :307–323, 15 April 2000. 45

[32] Timo Bretschneider, Philip J. Bones, Stephen J. McNeill, and David Pairman. Image-based quality assessment of SPOT data. *Proceedings of the American Society for Photogrammetry & Remote Sensing, St. Louis ASPRS*, 2001. Unpaginated CD-ROM. 229

[33] Claude Brezinski. *Computational aspects of linear control*, volume 1 of *Numerical Methods and Algorithms*. Springer US, Kluwer Academic Publishers, P.O. Box 17, 3300 AA Dordrecht, The Netherlands, USA, 2013. 1

[34] William Steven Brown. On Euclid's algorithm and the computation of polynomial greatest common divisors. *Journal of the Association for Computing Machinery (JACM)*, 18(4) :478–504, October 1971. 230

[35] William Steven Brown and Joseph Frederick Traub. On Euclid's algorithm and the theory of subresultants. *Journal of the Association for Computing Machinery (JACM)*, 18(4) :505–514, October 1971. 230

[36] James R. Bunch. The weak and strong stability of algorithms in numerical linear algebra. *Linear Algebra and Its Applications*, 88 :49–66, April 1987. 45

[37] Angelika Bunse-Gerstner. An analysis of the HR algorithm for computing the eigenvalues of a matrix. *Linear Algebra and its Applications*, 35 :155–173, February 1981. 2

[38] Angelika Bunse-Gerstner. Matrix factorizations for symplectic QR-like methods. *Linear Algebra and its Applications*, 83 :49–77, November 1986. 1, 34, 35, 36

[39] Angelika Bunse-Gerstner, Ralph Byers, and Volker Mehrmann. Numerical methods for algebraic Riccati equations. In Sergio Bittanti, editor, *Proceedings of Workshop on the Riccati Equation in Control, Systems, and Signals*, pages 107–116, Como, Italy, 16-29 June 1989. 3

[40] Angelika Bunse-Gerstner and Volker Mehrmann. A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation. *Institute of Electrical and Electronics Engineers (IEEE) transactions on automatic control*, 31(12) :1104–1113, December 1986. 1, 3, 26, 35, 77, 82, 106, 107, 130, 147, 153, 175

[41] Angelika Bunse-Gerstner and Volker Mehrmann. The HHDR algorithm and its application to optimal control problems. *RAIRO Automatique-productique informatique industrielle*, 23(4) :305–329, January 1989. 1

[42] Ralph Byers. A Hamiltonian QR algorithm. *SIAM Journal on Scientific and Statistical Computing*, 7(1) :212–229, January 1986. 3

[43] George E. Collins. Subresultants and reduced polynomial remainder sequences. *Journal of the Association for Computing Machinery (JACM)*, 14(1) :128–142, January 1967. 230

[44] Robert M. Corless, Stephen M. Watt, and Lihong Zhi. QR factoring to compute the GCD of univariate approximate polynomials. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Signal Processing*, 52(12) :3394–3402, December 2004. 230

[45] Shengyang Dai, Ming Yang, Ying Wu, and Aggelos K. Katsaggelos. Tracking motion-blurred targets in video. In *2006 IEEE International Conference on Image Processing*, pages 2389–2392, Atlanta, GA, October 2006. 229

[46] James W. Daniel, Walter Bill Gragg, Linda Kaufman, and Gilbert W. Stewart. Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. *Mathematics of Computation*, 30(136) :772–795, October 1976. 35, 45

[47] Jean Della-Dora. *Sur quelques algorithmes de recherche de valeurs propres*. Theses, Université Joseph-Fourier - Grenoble I (L'université scientifique et Médicale de Grenoble), July 1973. 19, 34, 175

[48] Jean Della-Dora. Numerical linear algorithms and group theory. *Linear Algebra and its Applications*, 10(3) :267–283, June 1975. 19, 34, 38, 175

[49] James Weldon Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, first edition, 1997. 174

[50] Gema María Diaz-Toca and Skander Belhaj. Blind image deconvolution through Bezoutians. *Journal of Computational and Applied Mathematics*, 315(Supplement C) :98–106, 01 May 2017. 247

[51] Gema María Diaz-Toca and Nadia Ben Atti. Block LU factorization of Hankel and Bézout matrices and Euclidean algorithm. *International Journal of Computer Mathematics*, 86(1) :135–149, January 2009. 230, 233

[52] Heike Faßbender. *Symplectic Methods for the Symplectic Eigenproblem*. Kluwer Academic/Plenum Publishers, Springer Science & Business Media, New York, NY, USA, first edition, 2000. 34

[53] Roland W. Freund and Volker Mehrmann. A symplectic look-ahead Lanczos algorithm for the Hamiltonian eigenvalue problem. *AT & T Numerical Analysis Manuscript, Bell Laboratories, Murray Hill, NJ*, 1994. 3

[54] Roland W. Freund and Noël M. Nachtigal. Software for simplified Lanczos and QMR algorithms. *Applied Numerical Mathematics*, 19(3) :319–341, December 1995. 2

[55] Luca Gemignani. GCD of polynomials and Bézout matrices. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (Kihei, HI)*, ISSAC '97, pages 271–277, New York, NY, USA, 1997. ACM. 230

[56] Dennis C. Ghiglia, Louis A. Romero, and Gary Arthur Mastin. Systematic approach to two-dimensional blind deconvolution by zero-sheet separation. *Journal of the Optical Society of America A*, 10(5) :1024–1036, May 1993. 228, 229, 238

[57] Sarah Frisken Gibson and Frederick Lanni. Experimental test of an analytical model of aberration in an oil-immersion objective lens used in three-dimensional light microscopy. *Journal of the Optical Society of America A*, 8(10) :1601–1613, October 1991. 229

[58] Luc Giraud, Julien Langou, and Miroslav Rozložník. The loss of orthogonality in the Gram–Schmidt orthogonalization process. *Computers & Mathematics with Applications*, 50(7) :1069–1075, October 2005. 45, 51

[59] Sergei Konstantinovich Godunov and Miloud Sadkane. Numerical determination of a canonical form of a symplectic matrix. *Siberian Mathematical Journal*, 42(4) :629–647, July 2001. 34

[60] Gene Howard Golub and Charles Francis Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996. 35, 36, 45, 174

[61] Eric James Grimme, Danny C. Sorensen, and Paul Van Dooren. Model reduction of state space systems via an implicitly restarted Lanczos method. *Numerical algorithms*, 12(1) :1–31, March 1996. 2

[62] A. Raymond Heindl. Fourier transform, polynomial GCD, and image restoration. Master's thesis, Department of Mathematical Sciences, Clemson University, 2005. 239, 241

[63] Nicholas John Higham. *Accuracy and stability of numerical algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, second edition, 2002. 45, 92, 99

[64] Jie Huang, Ting-Zhu Huang, and Skander Belhaj. Scaling Bini's algorithm for fast inversion of triangular Toeplitz matrices. *Journal of Computational Analysis and Applications*, 15(5) :858–867, 2013. 230, 234

[65] Imad M. Jaimoukha and Ebrahim M. Kasenally. Krylov subspace methods for solving large Lyapunov equations. *SIAM Journal on Numerical Analysis*, 31(1) :227–251, February 1994. 2

[66] William Jalby and Bernard Philippe. Stability analysis and improvement of the block Gram–Schmidt algorithm. *SIAM Journal on Scientific and Statistical Computing*, 12(5) :1058–1073, September 1991. 45

[67] Erich Kaltofen, Zhengfeng Yang, and Lihong Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, ISSAC '06, pages 169–176, New York, NY, USA, 2006. ACM. 229, 238

[68] Narendra K. Karmarkar and Yagati N. Lakshman. Approximate polynomial greatest common divisors and nearest singular polynomials. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation*, ISSAC '96, pages 35–39, New York, NY, USA, 1996. ACM. 230

[69] John Krist. Simulation of HST PSFs using Tiny Tim. In R. A. Shaw, H. E. Payne, and J. J. E. Hayes, editors, *Astronomical Data Analysis Software and Systems IV*, volume 77 of *Astronomical Society of the Pacific Conference Series*, page 349, 1995. Provided by the SAO/NASA Astrophysics Data System. 229

[70] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*, volume 1. Wiley-interscience a Diuision of John Wiley & Sons, Inc., New York, NY, USA, 1972. 1

[71] Peter Lancaster and Leiba Rodman. *Algebraic Riccati equations*. Oxford Science Publications. The Clarendon Press, Oxford University Press, New York, 1995. 3

[72] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4) :255–282, October 1950. 2

[73] Alan J. Laub. A Schur method for solving algebraic Riccati equations. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Automatic Control*, 24(6) :913–921, December 1979. 3, 4

[74] Alan J. Laub. Invariant subspace methods for the numerical solution of Riccati equations. In Sergio Bittanti, Alan J. Laub, and Jan C. Willems, editors, *The Riccati Equation*, Communications and Control Engineering Series, pages 163–196. Springer Berlin Heidelberg GmbH, Berlin, Heidelberg, first edition, 1991. 3

[75] Alan J. Laub and Kenneth Meyer. Canonical forms for symplectic and Hamiltonian matrices. *Celestial Mechanics*, 9(2) :213–238, April 1974. 3

[76] Bingyu Li, Zhuojun Liu, and Lihong Zhi. A structured rank-revealing method for Sylvester matrix. *Journal of Computational and Applied Mathematics*, 213(1) :212 – 223, 15 March 2008. 230

[77] Zijia Li, Zhengfeng Yang, and Lihong Zhi. Blind image deconvolution via fast approximate GCD. In *Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation*, ISSAC '10, pages 155–162, New York, NY, USA, 2010. ACM. 228, 229, 230, 238, 239, 240, 241, 242, 243, 245, 247

[78] Fu-Rong Lin, Wai-Ki Ching, and Michael K. Ng. Fast inversion of triangular Toeplitz matrices. *Theoretical Computer Science*, 315(2-3) :511–523, 06 May 2004. 230, 234

[79] Luciano Lopez and Valeria Simoncini. Preserving geometric properties of the exponential matrix by block Krylov subspace methods. *BIT Numerical Mathematics*, 46(4) :813–830, December 2006. 2

[80] D. Steven Mackey, Niloufer Mackey, and Françoise Tisseur. $\mathbb{G}$-reflectors : Analogues of Householder transformations in scalar product spaces. *Linear algebra and its applications*, 385 :187–213, July 2004. 2

[81] Volker Mehrmann. *Der SR-algorithmus zur bestimmung der eigenwerte einer matrix*. PhD thesis, Diplomarbeit, Universität Bielefeld, 1979. 34, 35, 175

[82] Volker Mehrmann. *The autonomous linear quadratic control problem : theory and numerical solution*, volume 163 of *Lecture notes in control and information sciences : M. Thoma and A. Wyner editors*. Springer-Verlag, Berlin Heidelberg, Germany, first edition, 1991. 1

[83] Volker Mehrmann and David S. Watkins. Structure-preserving methods for computing eigenpairs of large sparse skew-Hamiltonian/Hamiltonian pencils. *SIAM Journal on Scientific Computing*, 22(6) :1905–1925, 2001. 1, 3

[84] Gao Mei. *A new method for solving the algebraic Riccati equation.* Master's thesis, second edition, Nanjing Aeronautical Institute, Campus PO Box 245, Nanjing, P.R. China, 1986. 3

[85] Oleg V. Michailovich and Dan Adam. A novel approach to the 2-D blind deconvolution problem in medical ultrasound. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Medical Imaging*, 24(1) :86–104, 03 January 2005. 229

[86] George S. Miminis and Christopher C. Paige. Implicit shifting in the QR and related algorithms. *SIAM journal on matrix analysis and applications*, 12(2) :385–400, 1991. 222

[87] Marilena Mitrouli, Nicos Karcanias, and Christos Koukouvinos. Numerical performance of the matrix pencil algorithm computing the greatest common divisor of polynomials and comparison with other matrix-based methodologies. *Journal of Computational and Applied Mathematics*, 76(1–2) :89–112, 17 December 1996. 230

[88] Peter Nisenson and Richard Barakat. Partial atmospheric correction with adaptive optics. *Journal of the Optical Society of America A*, 4(12) :2249–2253, December 1987. 229

[89] Matu-Tarow Noda and Tateaki Sasaki. Approximate GCD and its application to ill-conditioned equations. *Journal of Computational and Applied Mathematics*, 38(1–3) :335 – 351, 23 December 1991. 230

[90] Christopher C. Paige and Charles Francis Van Loan. A Schur decomposition for Hamiltonian matrices. *Linear Algebra and its Applications*, 41 :11–32, December 1981. 3, 34, 35, 215

[91] Victor Y Pan. Computation of approximate polynomial GCDs and an extension. *Information and Computation*, 167(2) :71–85, 15 June 2001. 230

[92] S. Unnikrishna Pillai and Ben Liang. Blind image deconvolution using a robust GCD approach. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Image Processing*, 8(2) :295–301, February 1999. 228, 229, 230, 236, 238

[93] John Rischard Rice. Experiments on Gram–Schmidt orthogonalization. *Mathematics of Computation*, 20(94) :325–328, April 1966. 45

[94] Michael C. Roggemann. Limited degree-of-freedom adaptive optics and image reconstruction. *Applied Optics*, 30(29) :4227–4233, October 1991. 229

[95] Miroslav Rozložník, Felicja Okulicka-Dłużewska, and Alicja Smoktunowicz. Indefinite orthogonalization with rounding errors. (submitted), 28 November 2013. 45

[96] Axel Ruhe. Numerical aspects of Gram–Schmidt orthogonalization of vectors. *Linear algebra and its applications*, 52 :591–601, July 1983. 45

[97] Yousef Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics (Siam), Philadelphia, PA, USA, second edition, 2003. 35, 36, 174

[98] Miloud Sadkane and Ahmed Salam. A note on symplectic block reflectors. *Electronic Transactions on Numerical Analysis*, 33 :45–52, 31 March 2009. 2

[99] Ahmed Salam. On theoretical and numerical aspects of symplectic Gram–Schmidt-like algorithms. *Numerical Algorithms*, 39(4) :437–462, August 2005. 35, 36, 38, 45

[100] Ahmed Salam and Eman Al-Aidarous. Error analysis and computational aspects of SR factorization via optimal symplectic Householder transformations. *Electronic Transactions on Numerical Analysis*, 33 :189–206, 11 December 2009. 99

[101] Ahmed Salam and Eman Al-Aidarous. Equivalence between modified symplectic Gram–Schmidt and Householder SR algorithms. *BIT Numerical Mathematics*, 54(1) :283–302, March 2014. 45, 87

[102] Ahmed Salam, Eman Al-Aidarous, and Anas El Farouk. Optimal symplectic Householder transformations for SR decomposition. *Linear Algebra and its Applications*, 429(5-6) :1334–1353, 1 September 2008. 22, 35, 106, 107, 153

[103] Ahmed Salam and Haithem Ben kahla. A new variant of a double structure-preserving QR algorithm for symmetric and Hamiltonian matrices. *World Academy of Science, Engineering and Technology, International Science Index, Mathematical and Computational Sciences*, 3(12) :966, September 2016. 215

[104] Ahmed Salam and Haithem Ben Kahla. An upper J-Hessenberg reduction of a matrix through symplectic Householder transformations. *ArXiv e-prints*, December 2016. 106

[105] Ahmed Salam and Haithem Ben Kahla. A treatment of breakdowns and near breakdowns in a reduction of a matrix to upper J-Hessenberg form and related topics. *ArXiv e-prints*, October 2017. 154

[106] Ahmed Salam and Abderrahman Bouhamidi. A symplectic Arnoldi method for Hamiltonian matrix eigenvalue problem. Technical Report 166, LMPA, January 2002. 3

[107] Ahmed Salam and Anas El Farouk. Round off error analysis of symplectic Gram–Schmidt-like algorithms. Technical Report 280, LMPA, February 2006. 35, 45, 87

[108] Ahmed Salam, Anas El Farouk, and Eman Al-Aidarous. Symplectic Householder transformations for a QR-like decomposition, a geometric and algebraic approaches. *Journal of Computational and Applied Mathematics*, 214(2) :533–548, 1 May 2008. 2

[109] Ahmed Salam and David S. Watkins. Structured QR algorithms for Hamiltonian symmetric matrices. *Electronic Journal of Linear Algebra*, 22(1) :573–585, May 2011. 215, 216, 217, 220, 221, 223, 224, 226

[110] Arnold Schönhage. Quasi-GCD computations. *Journal of Complexity*, 1(1) :118–137, October 1985. 230

[111] Timothy J. Schulz. Multiframe blind deconvolution of astronomical images. *Journal of the Optical Society of America A*, 10(5) :1064–1073, May 1993. 229

[112] C. Andrew Segall, Rafael Molina, and Aggelos K. Katsaggelos. High-resolution images from low-resolution compressed video. *Institute of Electrical and Electronics Engineers (IEEE) Signal Processing Magazine*, 20(3) :37–48, May 2003. 229

[113] Gilbert W. Stewart. *Introduction to matrix computations*. Computer Science and Applied Mathematics. Academic Press New York, 1973. 174

[114] Charles Francis Van Loan. A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix. *Linear Algebra and its Applications*, 61 :233–251, September 1984. 3, 34, 35, 64, 77, 106, 130, 215

[115] Filip Šroubek and Jan Flusser. Multichannel blind deconvolution of spatially misaligned images. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Image Processing*, 14(7) :874–883, July 2005. 229

[116] David S. Watkins. *The matrix eigenvalue problem : GR and Krylov subspace methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 2007. 25, 26, 34, 64, 131, 175, 216

[117] James Hardy Wilkinson. *Rounding Errors in Algebraic Processes*. Number 32 in Notes on applied science. London : Her Majesty's Stationery Office (H.M.S.O), National Physical Laboratory, Teddmgton, Middlesex, England, 1963. 35

[118] James Hardy Wilkinson. *The algebraic eigenvalue problem*. Monographs on numerical analysis. The Clarendon Press, Oxford University Press, New York, NY, USA, 1965. Oxford Science Publications. 35

[119] Joab R. Winkler and John D. Allan. Structured total least norm and approximate GCDs of inexact polynomials. *Journal of Computational and Applied Mathematics*, 215(1) :1–13, May 2008. 230

[120] Joab R. Winkler and Xin Lao. The calculation of the degree of an approximate greatest common divisor of two polynomials. *Journal of Computational and Applied Mathematics*, 235(6) :1587–1603, January 2011. 230

[121] Yu-Li You and Mostafa Kaveh. A regularization approach to joint blur identification and image restoration. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Image Processing*, 5(3) :416–428, March 1996. 229

[122] Christopher J. Zarowski, Xiaoyan Ma, and Frederick W. Fairman. QR-factorization method for computing the greatest common divisor of polynomials with inexact coefficients. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Signal Processing*, 48(11) :3042–3051, November 2000. 230

[123] Zhonggang Zeng and Barry H. Dayton. The approximate GCD of inexact polynomials. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, ISSAC '04, pages 320–327, New York, NY, USA, 2004. ACM. 230

Sur des méthodes préservant les structures d'une classe de matrices structurées

Résumé Les méthodes d'algèbre linéaire classiques, pour le calcul de valeurs et vecteurs propres d'une matrice, ou des approximations de rang inférieurs (low-rank approximations) d'une solution, etc., ne tiennent pas compte des structures de matrices. Ces dernières sont généralement détruites durant le procédé du calcul. Des méthodes alternatives préservant ces structures font l'objet d'un intérêt important par la communauté. Cette thèse constitue une contribution dans ce domaine.

Ainsi l'algorithme SR, qui est un algorithme de type QR, et qui préserve les structures d'une classe importante de matrices, est basé sur la décomposition SR (qui est de type QR) et sur une réduction à une forme condensée : J-Hessenberg forme (de type Hessenberg).

La décomposition SR peut être calculé via l'algorithme de Gram-Schmidt symplectique. Comme dans le cas classique, une perte d'orthogonalité peut se produire. Pour y remédier, nous avons proposé deux algorithmes $RSGSi$ et $RMSGSi$, qui consistent à ré-orthogonaliser deux fois les vecteurs à calculer. La perte de la J-orthogonalité s'est améliorée de manière très significative.

L'étude directe de la propagation des erreurs d'arrondis dans les algorithmes de Gram-Shmidt symplectique est très difficile à effectuer. Nous avons réussi à contourner cette difficulté et donner des majorations pour la perte de la J-orthogonalité et de l'erreur de factorisation.

Une autre façon de calculer la décomposition SR est basée sur les transformations de Householder symplectiques. Celles ci présentent des paramètres libres. Un choix optimal a abouti à l'algorithme $SROSH$. Cependant, ce dernier peut être sujet à une instabilité numérique. Nous avons proposé une version modifiée nouvelle $SRMSH$, qui a l'avantage d'être aussi stable que possible. Une étude approfondie a été faite, présentant les différentes versions : $SRMSH$ et $SRMSH2$.

Dans le but de construire un algorithme SR, d'une complexité d'ordre $\mathcal{O}(n^3)$ où $2n$ est la taille de la matrice, une réduction (appropriée) de la matrice à une forme condensée (J-Hessenberg forme) via des similarités adéquates, est cruciale. Cette réduction peut être effectué via l'algorithme $JHESS$.

Nous avons montré qu'il possible de réduire une matrice sous la forme J-Hessenberg, en se basant exclusivement sur les transformations de Householder symplectiques. Le nouvel algorithme, appelé $JHSH$, est basé sur une adaptation de l'algorithme $SRSH$. D'un point de vue algébrique, cette méthode est l'analogue de la réduction d'une matrice sous la forme Hessenberg, par des transformations de Householder, dans le cas Euclidien. Ce nouveau algorithme peut être aussi sujet à une instabilité numérique. Nous avons réussi à proposer deux nouvelles variantes, aussi stables que possible : $JHMSH$ et $JHMSH2$. Une étude approfondie a été faite. Nous avons constaté que ces algorithmes se comportent d'une manière similaire à l'algorithme $JHESS$.

Une caractéristique importante de tous ces algorithmes ($JHESS$, $JHMSH$, $JHMSH2$) est qu' ils peuvent rencontrer un breakdown fatal, ou un "near breakdown" rendant impossible la suite des calculs, ou débouchant sur une instabilité numérique, privant le résultat final de toute signification. Ce phénomène n'a pas d'équivalent dans le cas Euclidien.

Nous avons réussi à élaborer une stratégie très efficace pour "guérir" le breakdown fatal et traîter le near breakdwon. Les nouveaux algorithmes intégrant cette stratégie sont désignés par $MJHESS$, $MJHSH$, JHM^2SH et JHM^2SH2.

Ces stratégies ont été ensuite été intégrées dans la version implicite de l'algorithme SR lui permettant de surmonter les difficultés rencontrées du fatal breakdown ou du near-breakdown. Rappelons que, sans ces stratégies, l'algorithme SR s'arrête.

Ensuite, nous présenté une variante de l'algorithme QR appliquée aux matrices Hamiltonienne (symétriques ou antisymétriques) qui préserve leur structure pendant la procédure.

Finalement, et dans un autre cadre de matrices structurées, nous avons présenté un algorithme robuste via FFT et la matrice de Hankel, basé sur le calcul approché de plus grand diviseur commun (PGCD) de deux polynômes, pour résoudre le problème de la déconvolution d'images. Plus précisément, nous avons conçu un algorithme pour le calcul du PGCD de deux polynômes bivariés. La nouvelle approche est basée sur un algorithme rapide, de complexité quadratique $\mathcal{O}(n^2)$, pour le calcul du PGCD des polynômes unidimensionnels. La complexité de notre algorithme est $\mathcal{O}(n^2\log(n))$ où la taille des images floues est $n \times n$. Les résultats expérimentaux avec des images synthétiquement floues, illustrent l'efficacité de notre approche.

Mots-clés : Produit scalaire antisymétrique, la préservation de la structure, matrice structurée, les transformations de Householder symplectiques, Gram-Schmidt symplectique, la décomposition SR, la forme de J-Hessenberg, réduction de matrice, l'algorithme SR, le PGCD approché, la matrice de Hankel, la matrice Hamiltonienne, la matrice symplectique, déconvolution d'image floue, restauration d'images.

On structure-preserving methods of a class of structured matrices

Abstract The classical linear algebra methods, for calculating eigenvalues and eigenvectors of a matrix, or lower-rank approximations of a solution, etc., do not consider the structures of matrices. Such structures are usually destroyed in the numerical process. Alternative structure-preserving methods are the subject of an important interest mattering to the community. This thesis establishes a contribution in this field.

Thus the SR algorithm, which is a QR-like algorithm, structure-preserving for a large class of structured matrices, is based on the SR decomposition (which is a QR-like decomposition) and on a reduction to the condensed matrix form : upper J-Hessenberg form (Hessenberg-like).

The SR decomposition is usually implemented via the symplectic Gram-Schmidt algorithm. As in the classical case, a loss of orthogonality can occur. To remedy this, we have proposed two algorithms $RSGSi$ and $RMSGSi$, where the reorthogonalization of a current set of vectors against the previously computed set is performed twice. The loss of J-orthogonality has significantly improved.

A direct rounding error analysis of symplectic Gram-Schmidt algorithm is very hard to accomplish. We managed to get around this difficulty and give the error bounds on the loss of the J-orthogonality and on the factorization.

Another way to implement the SR decomposition is based on symplectic Householder transformations. This algorithm involves some free parameters. An optimal choice of free parameters provided an optimal version of the algorithm $SROSH$. However, the latter may be subject to numerical instability. We have proposed a new modified version $SRMSH$, which has the advantage of being numerically more stable. By a detailed study, we are led to two new variants numerically more stables : $SRMSH$ and $SRMSH2$.

In order to build a SR algorithm of complexity $\mathcal{O}(n^3)$, where $2n$ is the size of the matrix, a reduction to the condensed matrix form (upper J-Hessenberg form) via adequate similarities is crucial. This reduction may be handled via the algorithm $JHESS$.

We have shown that it is possible to perform a reduction of a general matrix, to an upper J-Hessenberg form, based only on the use of symplectic Householder transformations. The new algorithm, which will be called $JHSH$ algorithm, is based on an adaptation of $SRSH$ algorithm. From a linear algebra point of view, $JHSH$ is the analogue in the symplectic case, of the algorithm performing the Hessenberg reduction of a matrix via Householder transformations in the Euclidean case. This new algorithm may also be subject to numerical instability. We are led to two news variants algorithms $JHMSH$ and $JHMSH2$ which are significantly more stable numerically. A detailed study has been done. We found that these algorithms behave quite similarly to $JHESS$ algorithm.

The main drawback of all these algorithms ($JHESS$, $JHMSH$, $JHMSH2$) is that they may encounter fatal breakdowns or may suffer from a severe form of near-breakdowns, causing a brutal stop of the computations, the algorithm breaks down, or leading to a serious numerical instability. This phenomenon has no equivalent in the Euclidean case.

We sketch out a very efficient strategy for curing fatal breakdowns and treating near breakdowns. Thus, the new algorithms incorporating this modification will be referred to as $MJHESS$, $MJHSH$, JHM^2SH and JHM^2SH2.

These strategies were then incorporated into the implicit version of the SR algorithm to overcome the difficulties encountered by the fatal breakdown or near-breakdown. We recall that without these strategies, the SR algorithm breaks.

Next, we presented a new variant of a double structure-preserving QR Algorithm for symmetric\skew-symmetric and Hamiltonian Matrices.

Finally and in another framework of structured matrices, we presented a robust algorithm via FFT and a Hankel matrix, based on computing approximate greatest common divisors (GCD) of polynomials, for solving the problem of blind image deconvolution. Specifically, we designed a specialized algorithm for computing the GCD of bivariate polynomials. The new algorithm is based on the fast GCD algorithm for univariate polynomials, of quadratic complexity $\mathcal{O}(n^2)$ flops. The complexity of our algorithm is $\mathcal{O}(n^2\log(n))$ where the size of blurred images is $n \times n$. The experimental results with synthetically blurred images are included to illustrate the effectiveness of our approach.

Key-words : Indefinite inner product, structure-preserving eigenproblems, structured matrix, symplectic Householder transformations, symplectic Gram-Schmidt, SR decomposition, upper J-Hessenberg form, breakdowns and near-breakdowns, matrix reduction, SR-algorithm, approximate GCD, Hankel matrix, Hamiltonian matrix, symplectic matrix, blind image deconvolution, image restauration.