M2AN - MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE

EVARISTE KAZAMARANDE
PIERRE COMON

Stabilité numérique de l'algorithme de Levinson

M2AN - Modélisation mathématique et analyse numérique, tome 29, n° 2 (1995), p. 123-170

<http://www.numdam.org/item?id=M2AN_1995__29_2_123_0>

© AFCET, 1995, tous droits réservés.

L'accès aux archives de la revue « M2AN - Modélisation mathématique et analyse numérique » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM
Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

MATHEMATICAL MODELLING AND NUMERICAL ANALYSIS
MODÉLISATION MATHÉMATIQUE ET ANALYSE NUMÉRIQUE

(Vol. 29, n° 2, 1995, p. 123 à 170)

STABILITÉ NUMÉRIQUE DE L'ALGORITHME DE LEVINSON (*)

by Evariste KAZAMARANDE (1) et Pierre COMON (2) (3).

Communiqué par F. CHATELIN

Résumé. — *Les propriétés de stabilité numérique de l'algorithme de Levinson-Durbin (LD) ont été le sujet de beaucoup de controverses. Pour réconcilier les points de vue divergents, il faut en réalité prendre en compte le conditionnement du système linéaire Toeplitz considéré [5] [9]. Le but de notre discussion est d'analyser en détail les performances numériques de cet algorithme, et de conclure sur sa stabilité (résistance aux erreurs d'arrondi).*

Pour les matrices Toeplitz symétriques définies positives, Cybenko a établi en 1980 des bornes inférieure et supérieure au conditionnement, et a montré que l'algorithme est faiblement stable. Dans ce travail nous proposons d'abord une majoration du conditionnement qui, en plus d'être plus fine que celle proposée par Cybenko, se généralise au cas indéfini. A l'aide d'une analyse directe de la propagation des erreurs, nous établissons ensuite une borne supérieure de l'erreur commise sur la solution calculée et en déduisons que l'algorithme LD est faiblement stable pour toute matrice fortement bien conditionnée (c'est-à-dire les sous-matrices principales sont aussi bien conditionnées). A l'aide d'une analyse inverse d'erreur, nous analysons enfin la stabilité inverse et la stabilité forte de l'algorithme LD.

Abstract. — *Numerical stability properties of the Levinson-Durbin (LD) algorithm have been much debated. Actually, controversies vanish if the conditioning of the linear system considered is taken into account [5] [9]. The goal of this paper is to analyse in detail numerical performances of this algorithm, and to come to a conclusion regarding its stability (robustness with respect to rounding errors).*

For positive definite symmetric Toeplitz matrices, Cybenko has established in 1980 lower and upper bounds to the condition number, and has proved that the LD algorithm is weakly stable. Here, the upper bound proposed is not only more accurate than the one proposed by Cybenko, but also generalises to the indefinite case. Based on a forward analysis of errors, an upper bound on the error in the computed solution is derived, and it is shown that the LD algorithm is weakly stable for any strongly well conditionned matrix (i.e. all its principal submatrices are well conditioned). Next, based on an backward analysis of errors, a strong inverse stability of the LD algorithm is proved.

Enregistré en tant que rapport technique IMAG LMC, n° RT94, janvier 1993.
(*) Manuscrit reçu le 5 février 1993, accepté le 30 mai 1994.
(1) LMC, INPG, 46 av. Félix Viallet, 38031 Grenoble Cedex.
(2) Thomson-Sintra, BP 157, 06903 Sophia-Antipolis Cedex.
(3) I3S - CNRS, 250 av. Einstein, 06560 Sophia-Antipolis.

M² AN Modélisation mathématique et Analyse numérique 0764-583X/95/02/$ 4.00
Mathematical Modelling and Numerical Analysis © AFCET Gauthier-Villars

1. INTRODUCTION

Quand on résoud un problème donné sur ordinateur, généralement de façon seulement approchée, outre la complexité de calcul de l'algorithme utilisé et l'espace mémoire nécessaire, on est souvent attentif à la précision des résultats obtenus. Il est bien connu que cette précision dépend à la fois de la qualité (stabilité) numérique de l'algorithme et de la nature (conditionnement) même du problème à résoudre. Ces deux notions, pour l'instant qualitatives, seront définies en détail dans la Section 2.

Outre l'imprécision due aux incertitudes sur les données, une erreur apparaît au cours de l'exécution de l'algorithme de résolution. Cette erreur provient du fait que l'algorithme est nécessairement fini et que chaque opération arithmétique s'accompagne en principe d'une « erreur d'arrondi ». L'accumulation de telles erreurs peut très bien fausser grossièrement les résultats. Avec une analyse explicite de la propagation des erreurs, on arrive à déceler ces deux sources d'imprécision (imprécision due aux erreurs de données et celle due erreurs d'arrondi) et à les borner séparément. Ainsi, à partir de l'erreur sur la solution due aux erreurs de données propagées, on étudie le *conditionnement du problème* (comment varie la solution quand les données sont modifiées) et, à partir de l'erreur due aux arrondis propagés, on peut juger la *qualité (stabilité) numérique des algorithmes* et les comparer entre eux. Dans ce travail, nous nous intéressons aux propriétés de stabilité numérique de l'algorithme de Levinson-Durbin pour résoudre les systèmes linéaires Toeplitz.

On appelle matrice « Töplitz » (du nom du mathématicien allemand Otto Töplitz), une matrice dont les éléments T_{ij} ne dépendent que de la différence indicielle $i-j$. Notons que cette désignation est désormais orthographiée « Toeplitz » pour des raisons de facilité typographique. La résolution de tels systèmes intervient comme étape essentielle dans le traitement numérique de nombreux problèmes scientifiques ou techniques, en particulier dans des problèmes variés relatifs au traitement des signaux numériques tels que la prédiction linéaire, l'identification des systèmes, le filtrage numérique, etc... [18] [12] [17].

De nombreuses méthodes numériques ont été imaginées en vue de construire efficacement une bonne solution. A l'heure actuelle, il existe beaucoup d'algorithmes rapides de résolution en $O(n^2)$ opérations arithmétiques (en $O\left(n \log^2(n)\right)$ opérations pour les algorithmes ultra-rapides) contre $O(n^3)$ opérations pour une matrice arbitraire [17] [12] [23]. Parmi eux, les plus connus sont l'algorithme de Levinson qui fournit aussi les facteurs de Choleski de l'inverse T_n^{-1} et l'algorithme de Schur qui, lui, calcule les facteurs de Choleski de la matrice T_n elle-même. Ainsi, en exploitant la structure « Toeplitz » de la matrice, on a pu réduire la complexité de calcul de $O(n^3)$ à $O(n^2)$ opérations arithmétiques pour résoudre un système linéaie Toeplitz d'ordre n.

Il reste d'importance fondamentale d'étudier les propriétés de stabilité numérique de ces algorithmes rapides, et d'évaluer leur robustesse au conditionnement du système linéaire. La détermination de cette précision étant une tâche difficile, un bon nombre d'ouvrages ou articles ne disent rien sur la précision des résultats ou réduisent la démonstration à l'avantage de l'une d'elles par des arguments empiriques et des considérations intuitives. Notre discussion porte sur l'algorithme de Levinson-Durbin (LD).

Les propriétés de stabilité numérique de cet algorithme ont été le sujet de beaucoup de controverses. Quelques auteurs affirment que l'algorithme LD a une performance médiocre en précision, tandis que d'autres le trouvent acceptable [9]. Tous ces points de vue sont accompagnés de résultats expérimentaux. Ces controverses ne peuvent être réconciliées que si le conditionnement du système associé est pris en considération. Il est en effet possible qu'un algorithme très stable produise des résultats entâchés de grandes erreurs relatives si le problème est mal conditionné. Il importe de séparer le conditionnement d'un problème et la stabilité numérique de l'algorithme utilisé. Nous nous proposons ici d'analyser en détail les performances numériques (stabilité, résistance aux erreurs d'arrondi) de l'algorithme de Levinson vis-à-vis du conditionnement du système linéaire Toeplitz qu'il résoud.

A cette fin, quelques résultats importants ont été déjà obtenus dans [9] et [5] :

— Bunch (1985) a montré que les algorithmes basés sur le partitionnement de la matrice peuvent être instables sauf peut-être si la matrice est symétrique définie positive [5]. L'algorithme de Levinson est de cette base.

— Cybenko (1980) a établi les bornes inférieure et supérieure *a posteriori* du conditionnement des matrices Toeplitz symétriques définies positives et a prouvé la faible stabilité de l'algorithme LD [9].

Dans ce travail nous étendons d'abord ces résultats de Cybenko au cas indéfini, et analysons ensuite les propriétés de stabilités inverse et forte de l'algorithme LD.

Avant d'entrer dans le vif du sujet, nous fixons d'abord dans § 2 le sens des mots (erreur d'arrondi, conditionnement d'un problème, stabilité de l'algorithme), non seulement pour clarifier la présentation, mais aussi parce que les solutions se logent déjà dans les termes. L'algorithme de Levinson-Durbin (LD) est rappelé ensuite dans le § 3.

Dans le § 4, nous établissons une majorante (meilleure que celle proposée dans [9]) du conditionnement de **toute** matrice Toeplitz symétrique basée sur la notion de « générateurs ». Nous présentons dans § 5 une analyse détaillée de la propagation des erreurs aussi bien directe qu'inverse dans l'algorithme LD. De l'analyse directe d'erreurs, nous établissons que l'algorithme est faiblement stable pour toute matrice fortement bien conditionnée. Et de l'analyse inverse d'erreurs, nous déterminons une perturbation (de norme minimale) sur la matrice initiale pour laquelle la solution *exacte* du système

linéaire perturbé associé est la solution *calculée* par l'algorithme LD. Une borne de cette perturbation nous permettra de déterminer la classe des matrices Toeplitz symétriques pour lesquelles l'algorithme LD est stable. Nous terminerons par une discussion sur la stabilité forte de cet algorithme. Nous montrons que la perturbation $\delta r^{(n)}$ convenable sur les données $r^{(n)}$ est solution d'un système linéaire structuré spécial : la matrice associée est une matrice « Toeplitz-plus-Hankel ».

2. DÉFINITIONS ET CONCEPTS DE BASE

2.1. Notations

Nous adopterons dans cet article des conventions afin de simplifier les écritures ; lorsque $x = (x_1, ..., x_n)$ (souvent noté $(x^{(n)})$ est un vecteur de dimension n, nous noterons par $\underline{x}$ le vecteur (x_0, x) de dimension $n+1$, où x_0 est fixé dans le contexte. On désignera par x^* une valeur approchée de x et par $\tilde{x}$ la valeur calculée de x, c'est-à-dire le produit du calcul de x en arithmétique finie. Et par conséquent, $\tilde{\underline{x}}$ désignera le vecteur $(\tilde{x}_0, \tilde{x})$.

Nous noterons également par $\Delta x = \tilde{\underline{x}} - x$ (respectivement $\delta x = x^* - x$) la différence entre la valeur exacte x et sa valeur calculée $\tilde{x}$ (respectivement sa valeur approchée x^*).

Si $\mathbf{u}$ est un vecteur de dimension m, on convient de noter $\|\mathbf{u}\|_p$ sa norme L^p définie par :

$$\|u\|_p = \left[\sum_{i=1}^{m} |u_i|^p \right]^{1/p}. \tag{2.1.1}$$

Si A est une matrice de dimension $m \times n$, on convient de noter la norme L^p de A comme suit, pour tout $p \in \mathbb{N}^*$:

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}. \tag{2.1.2}$$

On peut vérifier que pour toute matrice A de dimension $m \times n$, on a :

$$\|A\|_1 = \max_j \sum_{i=1}^{m} |A_{ij}|, \quad \|A\|_\infty = \max_j \sum_{j=1}^{n} |A_{ij}|. \tag{2.1.3}$$

Cette norme est une norme d'opérateur linéaire, et vérifie par conséquent l'inégalité de composition :

$$\|AB\|_p \leq \|A\|_p \|B\|_p, \quad \forall p > 0. \tag{2.1.4}$$

Dans la suite de notre article, nous utiliserons essentiellement la norme 1 et la norme infinie, qui sont d'ailleurs liées par la relation suivante [11] :

$$\|A\|_1 = \|A^t\|_\infty. \tag{2.1.5}$$

Lorsque dans une somme ou un produit, le premier indice est strictement supérieur au dernier, alors la somme vaut 0 et le produit vaut 1 :

$$\sum_{i=m}^{n} f(i) = 0 \quad \text{et} \quad \prod_{i=m}^{n} g(i) = 1, \quad \text{si} \quad n < m. \tag{2.1.6}$$

2.2. Représentation des nombres en machine et erreurs d'arrondi

Dans la plupart des systèmes informatiques, les calculs numériques sont effectués avec une arithmétique à virgule flottante. Tout nombre réel x est représenté sous la forme

$$x = m \cdot \beta^e, \quad m = \pm \sum_{i=1}^{t} d_i \beta^{-i}, \quad 0 \leq d_i < \beta \tag{2.2.1}$$

où le réel m et l'entier e sont appelés respectivement la mantisse et l'exposant du nombre considéré x, et β est la base de numération. Dans la pratique, la base β est égale à deux, à une puissance de deux, ou à dix (les calculettes). Ce système de numération présente quelques inconvénients :

- d'une part, l'exposant e est limité à un intervalle de la forme $[-M, N]$. Cette limitation sur l'exposant a pour conséquence que l'on ne peut représenter que des nombres réels compris en valeur absolue entre β^{-M-1} et β^N, sinon on dit qu'il y a dépassement de capacité (*overflow*), ou remplacement par zéro des nombres trop petits en valeur absolue (*underflow*) ;
- d'autre part, la mantisse m comporte un nombre fini t de chiffres, appelé *longueur de la mantisse.* Ainsi apparaissent **les erreurs d'arrondi.** Elles proviennent simplement du fait qu'une machine n'est capable de conserver, pour chaque nombre manipulé, qu'un nombre fini t de chiffres. Par exemple, si l'on travaille avec 10 chiffres, le nombre π sera remplacé par 3,141592653 ; dans cette approximation, une erreur de l'ordre de 6.10^{-10} est commise et va se propager dans la suite des calculs.

Pour majorer ces erreurs, nous avons à notre disposition le théorème classique suivant :

THÉORÈME 2.2.2 : *Si on fait abstraction des phénomènes d'overflow et d'underflow, dans l'arithmétique* $FP(\beta, t, c)$ *à virgule flottante en base* β, t *longueur de la mantisse (précision machine) et* c *la technique d'arrondi (chopping ou rounding), tout nombre réel* x *(non nul) est représenté en machine avec une erreur relative bornée par l'unité d'arrondissage* $\varepsilon_M = p\beta^{1-t}$, *avec* $p = 1$ *ou* $p = 1/2$, *selon la technique d'arrondi mise en œuvre (chopping ou rounding).*

En effet, pour tout réel x, il existe un et un seul entier e tel que $\beta^{e-1} \leqslant |x| < \beta^e$. Soit $fl(x)$ le nombre machine le plus proche du réel x. Il est clair que $|x - fl(x)| \leqslant p\beta^{e-t}$ avec $p = 1$ ou $1/2$, selon qu'il s'agit du chopping ou du rounding. Puisque $\beta^{e-1} \leqslant |x| < \beta^e$, il en résulte que l'erreur relative sur x est

$$\rho_x = \frac{|x - fl(x)|}{|x|} \leqslant p\frac{\beta^{e-t}}{\beta^{e-1}} = p\beta^{1-t} = \varepsilon_M . \tag{2.2.3}$$

En d'autres termes, s'il n'existe pas d'overflow ni d'underflow, et si la machine arrondit correctement, alors on aura

$$fl(x) = x(1 + \varepsilon) \quad avec \quad |\varepsilon| \leqslant \varepsilon_M . \tag{2.2.4}$$

On constate que l'erreur relative commise en représentant x par $fl(x)$ est inférieure à ε_M. La quantité ε_M est (selon Wilkinson) le plus petit nombre machine tel que $1 + \varepsilon_M \neq 1$, et est souvent appelé l'erreur machine ou « *l'unité d'arrondissage* ».

Le fait de travailler en virgule flottante rend l'arithmétique inexacte en principe. Lors d'une opération arithmétique sur deux nombres stockés en machine, le résultat ne sera en général qu'une approximation puisqu'à nouveau, on ne retiendra pas (sauf dans certains cas) toutes les décimales que donne l'opération exacte. Chaque opération arithmétique $(+, \times, -, /)$ s'effectue en principe avec une certaine erreur, et on montre que cette erreur est bornée relativement par l'unité d'arrondissage ε_M.

En règle générale, pour réaliser les fonctions les plus courantes, le système simule des *fonctions machines* réalisées au moyen de matériel ou de logiciel. Les fonctions machines les plus élémentaires sont les opérations arithmétiques de base $(+, -, \times, /)$ et les fonctions élémentaires courantes telles que $\sqrt{}$, log, e, fonctions trigonométriques de base et leurs inverses. Cette simulation est rarement parfaite. Même si x est un nombre machine, $f(x)$ n'est pas en général un nombre machine, de sorte que le mieux que l'on puisse espérer est que $\tilde{f}(x) = fl(f(x))$. Cependant, on peut postuler [14] que :

Postulat : Les fonctions machines sont évaluées avec une erreur relative bornée, c'est-à-dire qu'il existe un nombre positif ε_f, appelé constante d'erreur de f, tel que

$$\tilde{f}(x) = f(x)(1 + \varepsilon_{f(x)}) \quad \text{avec} \quad |\varepsilon_{f(x)}| \leq \varepsilon_f . \tag{2.2.5}$$

Autrement dit, l'erreur relative est majorée :

$$|\rho_{f(x)}| \leq \varepsilon_f , \quad \forall x \in \text{domaine de définition de } f . \tag{2.2.6}$$

Une fonction machine f sera dite « *idéalement réalisée* » si $\varepsilon_f \leq \varepsilon_M$. Sur plusieurs ordinateurs modernes, les opérations arithmétiques de base sont idéalement réalisées et les fonctions élémentaires sont souvent presque idéalement réalisées aussi longtemps que $f(x) \neq 0$; le cas $f(x) = 0$ étant toutefois aussi admis pour les nombres machines.

2.3. Modèles de propagation des erreurs

Soit x^* une valeur approchée pour une quantité dont la valeur exacte est x. Seule x^* est en général accessible. Un problème fondamental en analyse numérique est de savoir dans quelle mesure une erreur (par exemple d'arrondi) introduite à un endroit donné dans le calcul affecte les résultats ultérieurs.

Supposons que l'algorithme commence avec un ensemble fini de données $t_0 = (x_1, x_2, ..., x_n)$ et produit alors, selon quelques prescriptions mathématiques, une suite finie de quantités $t_1, t_2, ..., t_{T(A)}$ avec

$$(A) \qquad t_0 = (x_1, x_2, ..., x_n) = \text{les données d'entrées du problème ,}$$

$$t_k = f_k(t_1, t_2, ..., t_{k-1} ; x_1, ..., x_n), \quad k = 1, ..., T(A) ,$$

où f_k est une fonction machine, à $v(i)$ variables et définie de $\mathbb{R}^{v(k)}$ dans $\mathbb{R}$, et $T(A) \in \mathbb{N}$. *Un algorithme mathématique peut donc être représenté par le système dynamique discret* (A). Naturellement, dans plusieurs cas, un ou deux arguments apparaissent explicitement dans f_i. Le résultat désiré est soit le $t_{T(A)}$ final, soit plusieurs t_k, soit tous les t_k.

Si cet algorithme est exécuté sur une machine, outre la difficulté que les entrées t_0 doivent être remplacées par des nombres machines $\tilde{t}_0$, l'ordinateur n'exécutera pas correctement l'algorithme (A). En effet, il générera les nombres machines $\tilde{t}_1, \tilde{t}_2, ..., \tilde{t}_{T(A)}$ définis par le système $(\tilde{A})$:

$$(\tilde{A}) \quad \tilde{t}_0 = (\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n) \text{ les données machines d'entrées du problème}$$

$$\tilde{t}_k = \tilde{f}_k(\tilde{t}_1, \tilde{t}_2, ..., \tilde{t}_{k-1} ; \tilde{x}_1, ..., \tilde{x}_n) , \quad k = 1, ..., T(A)$$

où $\tilde{f}_k$ est une fonction simulant f_k. Le système $(\tilde{A})$ représente l'image machine de l'algorithme (A). Le problème du contrôle de l'erreur se présente alors mathématiquement de la manière suivante :

$$\text{contrôler } (\tilde{t}_k - t_k) \text{, en particulier pour } k = T(A) .$$

Les méthodes d'évaluation d'erreurs d'arrondi, bien que peu unifiées et la plupart du temps empiriques, peuvent se répartir dans deux classes : les *méthodes probabilistes* et les *méthodes déterministes* [10]. Seule l'approche déterministe, en particulier par majorations récursives d'erreurs, retiendra notre attention. Nous nous intéresserons donc à majorer l'erreur affectant le résultat final, à partir de majorations d'erreurs apparaissant dans les différentes étapes du calcul. Le but est d'estimer (et de proposer une majoration de) $\| \tilde{t}_k - t_k \|$, où $\| \cdot \|$ est une norme quelconque. Il faut donc étudier comment les erreurs se propagent d'une étape $k-1$ à l'étape k par la fonction élémentaire associée f_k.

Pour les opérations arithmétiques, on a

$$\Delta(x+y) = \Delta x + \Delta y + (x+y)\,\varepsilon^{+} \quad \text{et } \rho_{x+y} = \frac{x}{x+y}\rho_x + \frac{y}{x+y}\rho_y + \varepsilon^{+} \; ; (2.3.1)$$

$$\Delta(x-y) = \Delta x - \Delta y + (x-y)\,\varepsilon^{-} \quad \text{et } \rho_{x-y} = \frac{x}{x-y}\rho_x - \frac{y}{x+y}\rho_y + \varepsilon^{-} \; ; (2.3.2)$$

$$\Delta(xy) = y\,\Delta x + x\,\Delta y + (xy)\,\varepsilon^{*} \quad \text{et } \rho_{xy} = \rho_x + \rho_y + \varepsilon^{*} \; ; \qquad (2.3.3)$$

$$\Delta(x/y) = \frac{1}{y}\Delta x - \frac{x}{y^2}\Delta y + (x/y)\,\varepsilon' \quad \text{et } \rho_{x/y} = \rho_x - \rho_y + \varepsilon' . \qquad (2.3.4)$$

Ces opérations étant idéalement réalisées, on aura $|\varepsilon| < \varepsilon_M$ pour l'une quelconque d'entre elles. L'accumulation de telles erreurs d'arrondi peut très bien fausser grossièrement les résultats.

Parmi les méthodes d'analyse explicite des erreurs, les plus couramment utilisées sont :

- L'analyse classique usuelle proposée par Wilkinson [24] [25]. Elle consiste à remplacer explicitement dans l'algorithme la valeur exacte d'une quantité x par sa valeur calculée $\tilde{x} = x(1+\rho_x)$, où ρ_x est l'erreur relative sur x, puis d'effectuer l'expression de sorte que résultat final S s'écrive sous la forme : $\tilde{S} = S(1+E)$; E sera alors l'erreur relative sur le résultat S.
- Une autre méthode est celle de *l'analyse par graphe* (process graph) proposée par Bauer [2]. Elle consiste à représenter un algorithme par un graphe orienté dont les nœuds correspondent soit aux données, soit aux résultats intermédiaires ou dinals. Le nœud t_j est lié au nœud t_i par une branche si le résultat correspondant au nœud t_j est un opérande de fonction élémentaire qui produit le résultat correspondant au nœud t_i. Chaque branche et chaque

nœud est pondéré par le coefficient de propagation d'erreur, et la constante d'erreur de la fonction correspondante, respectivement. Pour obtenir la contribution totale des erreurs, on multiplie tous les coefficients de propagation d'erreur le long des différents chemins et on fait la somme.

2.4. Conditionnement d'un problème

Soit f une fonction de E_d dans E_s, où E_d et E_s sont des espaces (resp. des données et des solutions) normés sur $\mathbb{R}$ et considérons le problème du calcul de $y = f(x)$ à partir des données x. En pratique, les données sont très souvent tirées de mesures physiques ou sont le résultat d'autres calculs. Elles sont donc généralement d'une précision limitée. La question qui se pose alors est de savoir dans quelle mesure les résultats du calcul seront sensibles à de légères modifications des données. Si une petite perturbation des données induit un changement important des résultats, il est douteux que les résultats obtenus puissent être fiables. On parle alors de problème *mal conditionné.* Avant de mettre en œuvre un algorithme, il importe de prévoir des tests permettant de déterminer si le problème abordé est mal conditionné.

Dans la suite, nous ne nous intéresserons qu'aux **problèmes bien posés,** c'est-à-dire que la solution $y = f(x)$ existe, est unique et dépend continûment des données. Soulignons que les problèmes mal posés nécessitent des techniques de résolution avancées (compliquées). Penser par exemple aux systèmes linéaires singuliers.

Notons : x^* les données perturbées du problème, $y = f(x)$ la solution exacte (théorique) du problème, et $y^* = f(x^*)$ la solution exacte du problème dont les données sont perturbées.

On aimerait que la quantité $y^* = f(x^*)$ soit aussi proche que possible de $f(x)$. Mais certains problèmes souffrent du passage de x à x^*. Pour caractériser la sensibilité de la fonction f, **on suppose qu'elle soit différentiable.** Dans ce cas, si x est changé en $x + dx$, alors $y = f(x)$ est perturbée de $dy = f'(x)\,dx$. La grandeur $\|f'(x)\|$ décrit au premier ordre le conditionnement d'un problème au point x, et est appelé « condition number » en anglais, mais simplement « *conditionnement* » en français. Le conditionnement absolu au point x, noté ici par $\kappa_{\text{abs}}(f, x)$, est donné par la norme de la matrice jacobienne de f (de $\mathbb{R}^n$ dans $(\mathbb{R}^m)$:

$$\kappa_{\text{abs}}(f, x) = \left\| \left(\frac{\partial f_i(x)}{\partial x_j} \right)_{i,j} \right\|, \quad 1 \leq i \leq m, 1 \leq j \leq n, \qquad (2.4.2a)$$

et

$$\|\Delta y\| \leq \kappa_{\text{abs}}(f, x) \|\Delta x\| + O(\|\Delta x\|^2). \qquad (2.4.2b)$$

Le problème est *bien conditionné au point x* si le conditionnement est petit, et *mal conditionné* si ce nombre est grand.

Dans certains cas, il est plus convenable de comparer les erreurs relatives des données et des résultats. Alors, il est approprié de considérer le « *conditionnement* », noté $\kappa_{\text{rel}}(f, x)$:

$$\kappa_{\text{rel}}(f, x) = \frac{\|f'(x)\|}{\|f(x)\|} \|x\|, \tag{2.4.3a}$$

et

$$\rho_y \leqslant \kappa_{\text{rel}}(f, x)\, \rho_x + O(\|\Delta x\|^2) \quad \text{avec} \quad \rho_x = \frac{\|\Delta x\|}{\|x\|}, \rho_y = \frac{\|\Delta y\|}{\|y\|} \tag{2.4.3b}$$

Cas de la résolution de systèmes linéaires

Lorsque le problème à résoudre est un système linéaire $Ax = b$, le résultat (2.4.3) prend la forme du théorème suivant [11].

THÉORÈME : *Si* $Ax = b$ *et* $(A + \Delta A)(x + \Delta x) = b + \Delta b$, *où* A *est régulière et* $\|\Delta A\|_p \|A^{-1}\|_p < 1$, *alors* $A + \Delta A$ *est aussi régulière et*

$$\frac{\|\Delta x\|_p}{\|x\|_p} \leqslant \frac{\|A\|_p \|A^{-1}\|_p}{1 - \|\Delta A\|_p \|A^{-1}\|_p} \left(\frac{\|\Delta A\|_p}{\|A\|_p} + \frac{\|\Delta b\|_p}{\|b\|_p} \right). \tag{2.4.4}$$

Le facteur $\kappa_p(A) = \|A\|_p \|A^{-1}\|_p$ est appelé le « *conditionnement* » de la matrice A, et mesure la variation relative de la solution x vis-à-vis des variations relatives des données A et b. On dira qu'un système linéaire est *bien* ou *mal conditionné* selon que le conditionnement de la matrice associé est « petit » ou « grand ». Les propriétés de $\kappa_p(A)$ peuvent être trouvées dans la littérature ; notons simplement ici que $\kappa_p(A) \geqslant 1$.

Remarque : Le conditionnement d'un problème, ne faisant aucune mention sur l'algorithme de résolution utilisé, est donc une propriété purement mathématique du problème, qui n'a rien à voir avec l'arithmétique finie de l'ordinateur, ni avec l'algorithme de résolution utilisé. Il est indépendant de tout algorithme choisi pour le résoudre, et s'évalue *en arithmétique exacte.* Elle existe dans le problème avant même que celui-ci soit résolu numériquement. C'est pourquoi d'ailleurs on l'appelle parfois la stabilité mathématique du problème [10].

2.5. Stabilité numérique d'un algorithme

Si un algorithme est exécuté sur une machine, même si les données d'entrées étaient des nombres machines, l'ordinateur n'exécuterait pas exactement l'algorithme mathématique. A chaque pas de l'algorithme, la fonction élémentaire associée engendre une erreur d'arrondi qui se propage dans la suite du calcul. Il est clair que l'accumulation de telles erreurs, appelées ici *erreurs d'arrondi-propagées,* dépendra de la décomposition du problème en fonctions élémentaires, c'est-à-dire de l'algorithme. Deux algorithmes mathématiquement équivalents (c'est-à-dire associés au même problème) peuvent fournir des résultats numériquement différents. La notion de stabilité permet de juger la qualité numérique des algorithmes et de les comparer entre eux. Un certain nombre d'approches existent, et sont bien exposées par exemple dans [8b]. Pour notre propos, les deux définitions générales suivantes seront suffisantes, mais il faudra les adapter ensuite au cas des systèmes linéaires structurés.

Stabilité directe

Plus généralement, on dira qu'un algorithme est *stable au sens direct* (forward stable) pour les données x si la solution calculée $\tilde{y} = \tilde{f}(x)$ est proche de la solution exacte $y = f(x)$,

$$\tilde{f}(x) = f(x) + \Delta_1, \ \|\Delta_1\| \leqslant D_1 \varepsilon_M, \tag{2.5.1}$$

où D_1 est une constante « pas trop grande » et ε_M l'unité d'arrondissage. Par opposition, l'instabilité numérique est présente si ces erreurs intermédiaires ont une très grande influence sur le résultat. Un algorithme est *numériquement plus stable* qu'un autre pour calculer $f(x)$ si l'erreur finale due aux erreurs d'arrondi-propagées est plus faible.

Stabilité inverse

Par opposition à l'analyse directe d'erreurs (*forward analysis*) qui consiste à borner la différence $f(x) - \tilde{f}(x)$, une analyse inverse (*backward analysis*) d'erreurs, introduite par Wilkinson, consiste à écrire la solution calculée $\tilde{f}(x)$ comme la solution exacte $f(x^*)$ du problème (exact) pour les données perturbées x^*.

DÉFINITION 2.5.2 : *On dira qu'un algorithme est* stable au sens inverse *(selon Wilkinson) pour une classe X de données si* $\forall x \in X$, $\exists x^*$ *proche de x, tel que* $f(x^*) = \tilde{f}(x)$. *C'est-à-dire que la solution calculée* $\tilde{f}(x)$ *est la solution exacte* $f(\tilde{x})$ *du problème pour des données « légèrement » modifiées* $x^* = x + \Delta_2$:

$$\tilde{f}(x) = f(x + \Delta_2), \ \|\Delta_2\| \leqslant D_2 \varepsilon_M,$$

où D_2 est une constante « pas trop grande » (relativement petite par rapport à $\|x\|$).

Dans cette dernière analyse, on se contente souvent d'un algorithme lorsque l'effet des erreurs d'arrondi qu'il propage sur la solution est du même ordre de grandeur que l'effet de certaines modifications (considérées comme petites) dans les données.

La stabilité inverse ne garantit donc pas à elle seule une solution précise, mais on peut constater que

$$\tilde{f}(x) = f(x) + \kappa_{\text{abs}}(f, x)\, \Delta_2' \,, \quad \|\Delta_2'\| \leqslant D_2\, \varepsilon_M + o(\varepsilon_M^2) \,; \qquad (2.5.3)$$

la solution calculée est aussi bonne que la solution exacte du problème dont les données sont modifiées de $\|x^* - x\|$. De l'équation (2.5.3), on déduit que *la précision $\|\tilde{y} - y\|$ des résultats dépend à la fois du conditionnement du problème et de la stabilité numérique de l'algorithme de résolution.*

Résolution des systèmes linéaires

Pour la résolution d'un système linéaire $Ax = b$, où $A \in \mathbb{R}^{n \times n}$ et $b \in \mathbb{R}^n$, le concept de stabilité peut être défini de diverses façons. Nous avons relevé les quatre définitions ci-dessous.

DÉFINITION 2.5.4 (Wilkinson) : *Un algorithme conçu pour résoudre $Ax = b$ est* inversement stable *(au sens classique) pour une classe de matrices* **A** *si $\forall A \in \mathbf{A}$ et $\forall b$, la solution calculée $\tilde{x}$ satisfait $A^* \tilde{x} = b^*$, pour A^* et b^* proches de A et b, respectivement.*

Dans cette définition, on s'intéresse à borner les différences

$$\frac{\|A - A^*\|}{\|A\|} \quad \text{et} \quad \frac{\|b - b^*\|}{\|b\|}$$

et on se contente de vérifier que ces bornes sont « assez petites ». Mais que veut dire « assez petit » ? Cette notion est relative à la précision des données et celle que l'on veut avoir sur la solution puisque

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leqslant \frac{\kappa(A)}{1 - \kappa(A)\, \rho_A} (\rho_A + \rho_b) \quad \text{où} \quad \rho_A = \frac{\|A - A^*\|}{\|A\|} \quad \text{et} \quad \rho_b = \frac{\|b - b^*\|}{\|b\|} \,.$$

Noter qu'il n'est pas nécessaire que A^* soit dans la classe **A** et que x soit proche de $\tilde{x}$. Cependant dans certains problèmes physiques, il est pourtant nécessaire que A^* soit issue d'une classe **A** bien définie pour que la définition de stabilité ait un sens physique [6]. C'est ce que précise la définition de Bunch proposée dans [6]. Afin de pouvoir prendre en compte le cas de seconds membres structurés, nous avons aménagé la définition de [6] comme suit.

DÉFINITION 2.5.5 (Bunch [6]) : *Un algorithme résolvant un système* $Ax = b$ *est* fortement (inversement) stable *pour une classe de matrices* $\mathbf{A}$ *et de seconds membres* $\mathbf{B}$ *si* $\forall A \in \mathbf{A}$ *et* $\forall b \in \mathbf{B}$, *la solution calculée* $\tilde{x}$ *de* $Ax = b$ *satisfait* $A^* \tilde{x} = b^*$, *où* $A^* \in \mathbf{A}$ *et* $b^* \in \mathbf{B}$, *avec* A^* *proche de* A *et* b^* *proche* b *(la proximité étant à entendre au sens donné en* 2.5.9).

Beaucoup d'utilisateurs sont satisfaits d'un algorithme s'il produit des solutions précises. Bien sûr, si le problème est mal conditionné, on doit s'attendre à avoir des difficultés. C'est pourquoi on ne s'intéresse qu'à la précision relative des résultats pour les problèmes bien conditionnés. D'où la définition suivante.

DÉFINITION 2.5.6 (Bunch [6]) : *Un algorithme résolvant un système* $Ax = b$ *est* faiblement (directement) stable *pour une classe* $\mathbf{A}$ *de matrices si, pour toute matrice* A *bien conditionnée dans* $\mathbf{A}$ *et pour tout vecteur* b, *la solution* $\tilde{x}$ *calculée de* $Ax = b$ *est telle que l'erreur relative* $\|\Delta x\| / \|x\|$ *soit petite.*

Quelquefois, on peut se contenter d'un algorithme $\tilde{f}$ si la solution $\tilde{f}(x)$ qu'il fournit est « assez proche » de la solution exacte $f(x^*)$ du problème pour des données « légèrement » perturbées. Ceci rejoint la notion de proximité dans la *stabilité mixée* selon Stewart [21].

DÉFINITION 2.5.7 (Stewart [21, p. 76] : *Un algorithme* $\tilde{f}$ *calculant une solution* $\tilde{f}(s)$ *en arithmétique finie est dit* fortement stable sur $\mathbf{S}$ *si pour tout* $s \in \mathbf{S}$, *il existe un* $s^* \in \mathbf{S}$ *proche de* s *tel que* $\tilde{f}(s)$ *soit proche de* $f(s^*)$.

Cette dernière définition est plus faible que la définition de Bunch donnée plus haut, puisque la solution calculée n'est pas forcément égale à la solution exacte d'un système perturbé, mais est simplement tenue à se trouver dans son voisinage. Ceci nous conduit à, pour la résolution de systèmes linéaires structurés avec second membre structuré.

DÉFINITION 2.5.8 : *Un algorithme résolvant un système* $Ax = b$ *est* fortement stable *(selon Stewart) sur une classe de problèmes* $\{A, b\} \in \mathbf{A} \times \mathbf{B}$ *si la solution calculée* $\tilde{x}$ *de* $Ax = b$ *est proche d'une solution* x^* *satisfaisant exactement* $A^* x^* = b^*$, *où* $A^* \in \mathbf{A}$ *est proche de* A, *et* $b^* \in \mathbf{B}$ *est proche de* b.

DÉFINITION 2.5.9 : *Par « a proche de b », on entendra* $\|a - b\| \leq p(n)\, \varepsilon_M$, *où* $p(n)$ *est un polynôme connu en* n *et* ε_M *l'unité d'arrondissage. Cette définition est standard, celle adoptée par exemple dans* [25], [11].

Pour simplifier les terminologies, nous adopterons dans la suite les notations suivantes :

« W-stable » pour inversement stable au sens classique de Wilkinson, conformément à (2.5.4).

« B-stable » pour fortement stable selon Bunch, conformément à (2.5.5).

« F-stable » pour faiblement stable selon Bunch, conformément à (2.5.6).

« S-stable » pour fortement stable selon Stewart, conformément à (2.5.8).

Observations

Remarquons les propriétés suivantes :

La B-stabilité entraîne la W-stabilité, qui implique à son tour la S-stabilité et la F-stabilité. (2.5.10)

Pour montrer qu'un algorithme est instable, il suffit d'exhiber une matrice bien conditionnée pour laquelle l'algorithme fournit une mauvaise solution (imprécise). (2.5.11)

Les algorithmes W-stables pour les matrices symétriques sont aussi B-stables pour cette même classe de matrices (Bunch *et al.* [8]). (2.5.12)

3. ALGORITHME DE LEVINSON-DURBIN

Comme le nom l'indique, l'algorithme a été introduit la première fois par Levinson, en 1946, pour résoudre les systèmes linéaires à second membre quelconque, et puis a été reformulé par Durbin pour résoudre rapidement un système linéaire à second membre particulier (*équations normales de Yule-Walker*) :

$$T_{n-1}\, a^{(n)} = -\, r^{(n)} \quad \text{où} \quad a^{(n)} = \begin{pmatrix} a_1^{(n)} \\ a_2^{(n)} \\ \vdots \\ a_n^{(n)} \end{pmatrix}, \quad r^{(n)} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix}, \tag{3.1}$$

et où T_{n-1} est la matrice Toeplitz de taille n :

$$T_{n-1} = \begin{pmatrix} 1 & r_1 & \cdots & r_{n-1} \\ r_1 & 1 & & \vdots \\ \vdots & & \ddots & r_1 \\ r_{n-1} & \cdots & r_1 & 1 \end{pmatrix}, \quad r_0 = 1\,. \tag{3.2}$$

D'où le nom d'algorithme de Levonson-Durbin (LD). Nous pouvons de façon équivalente considérer le système linéaire $(n+1) \times (n+1)$ suivant :

$$T_n\, \bar{a}^{(n)} = \sigma_n\, e_1\,, \tag{3.3}$$

où nous avons posé

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \sigma_n = 1 + a^{(n)t} r^{(n)} \text{ et } \quad \bar{a}^{(n)} = \begin{pmatrix} 1 \\ a^{(n)} \end{pmatrix}. \tag{3.4}$$

L'algorithme est récursif en ordre sur les coefficients de prédiction linéaire $a_i^{(n)}$ et basé sur les propriétés de déplacement et de centro-symétrie des matrices Toeplitz. Il est donné par

Algorithme de Levinson-Durbin (cas symétrique) (3.5) :

$$\sigma_0 = r_0 = 1 , \quad a_0^{(0)} = 1 .$$

Pour $i = 1$ à n faire

$$\begin{cases} w_i & = r_i + \sum_{j=1}^{i-1} a_j^{(i-1)} r_{i-j} \\ k_i & = -\dfrac{w_i}{\sigma_{i-1}} \\ \sigma_i & = \sigma_{i-1}(1 - k_i^2) \\ a_j^{(i)} & = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)} \quad (\text{pour } j = 1 \text{ à } i-1) \\ a_i^{(i)} & = k_i \end{cases}$$

Les quantités intermédiaires k_i, $1 \leqslant i \leqslant n$, reçoivent des interprétations variées dans la littérature. Elles sont souvent appelées paramètres de Schur, coefficients de corrélation partielle, ou coefficients de réflexion, respectivement dans le monde mathématique, statistique, et des sciences de l'ingénieur. A cause de leur interprétation physique usuelle, les coefficients k_i seront appelés coefficients de réflexion dans la suite du travail. Nous appellerons la quantité σ_i « résidu de prédiction d'ordre i ».

L'algorithme que nous avons l'habitude d'utiliser lorsque le second membre est arbitraire, est en réalité une adaptation de l'algorithme de Levinson-Durbin. La solution est donnée par l'exécution d'une instruction supplémentaire après le calcul de $a^{(i)}$ dans (3.5) :

$$x^{(i)} = \begin{bmatrix} x^{(i-1)} \\ 0 \end{bmatrix} + \mu_i \begin{bmatrix} Ja^{(i-1)} \\ 1 \end{bmatrix} \quad \text{avec} \quad \mu_i = \frac{b_i^{(n)} - \sum_{j=1}^{i-1} x_j^{(i-1)} r_{i-j}}{\sigma_{i-1}}, \quad (3.6)$$

où J désigne la matrice « anti-identité » inversant l'ordre des lignes et $b^{(n)}$ le second membre. Cet algorithme résoud en réalité simultanément les deux systèmes linéaires $T_{n-1}\, a^{(n)} = -\, r^{(n)}$ et $T_{n-1}\, x^{(n)} = b^{(n)}$.

L'algorithme de Levinson-Durbin fonctionne tant que les résidus σ_i sont non nuls, ce qui reste vrai tant que $|k_i| \neq 1$. Nous ferons donc cette hypothèse :

$$\sigma_i \neq 0, \quad |k_i| \neq 1 . \quad (3.7)$$

On peut montrer que l'algorithme de Levinson réalise une factorisation de la forme suivante :

$$T_n^{-1} = A_n\, S_n^{-1}\, A_n^t \quad \text{avec} \quad S_n = \text{Diag}\{\sigma_0, \sigma_1, ..., \sigma_n\} \quad \text{et} \quad \sigma_0 = r_0 , \quad (3.8)$$

et où A_n est triangulaire supérieure. Ce type de factorisation est souvent appelé « factorisation de Crout ». La matrice A_n est alors composée des coefficients de prédiction successifs :

$$A_n = \begin{pmatrix} 1 & a_1^{(1)} & \dots & a_n^{(n)} \\ & 1 & \vdots & \vdots \\ & & 1 & a_1^{(n)} \\ & & & 1 \end{pmatrix} . \quad (3.9)$$

Autrement dit, la i-ième colonne de A_n est $\overline{Ja}^{(i)}$ complété par des zéros, avec les notations définies plus haut.

Lorsque T_n est définie positive, la matrice S_n ne contient que des éléments positifs, et (3.8) n'est autre qu'une factorisation de Choleski T_n^{-1}, normalisée par S_n. Si T_n est définie positive, nous avons donc en outre :

$$\sigma_i > 0 \quad \text{et} \quad |k_i| < 1 . \quad (3.10)$$

La condition (3.10) constitue le test de Schur-Cohn de stabilité du filtre construit à partir des coefficients $a_i^{(n)}$ de prédiction linéaire [18]. Attirons l'attention du lecteur sur le fait qu'il existe deux factorisations de Choleski possibles, et que Cybenko n'utilise pas la même que (3.9) dans [9]. En particulier, il utilise $T_n^{-1} = C_n D_n^{-1} C_n^t$, avec $C_n = JA_n J$ et $D_n^{-1} = JS_n^{-1} J$.

4. CONDITIONNEMENT DES MATRICES TOEPLITZ SYMÉTRIQUES

Nous avons vu en (2.4.4) que le conditionnement d'un système linéaire $Tx = b$ fait intervenir les normes des matrices T et T^{-1}, alors que seule la matrice T est donnée. Dans cette section, nous allons proposer une borne à la norme de T^{-1} ne faisant intervenir que des quantités accessibles avant ou pendant l'exécution de l'algorithme de résolution. De ce fait, le conditionnement sera disponible dès lors que la solution finale aura été calculée. Notre borne est basée sur la notion de « générateurs ».

4.1. Générateurs d'une matrice Toeplitz symétrique et de son inverse

Définissons la matrice de déplacement suivante

$$Z = \begin{pmatrix} 0 & & & 0 \\ 1 & 0 & & \\ & \ddots & \ddots & \\ 0 & & 1 & 0 \end{pmatrix}. \tag{4.1.1}$$

Une prémultiplication par Z a pour effet de déplacer les lignes vers le bas, tandis qu'une postmultiplication par Z^t décale les colonnes vers la droite. Il est alors facile de voir que dans la matrice

$$\nabla T_n = \text{def } T_n - ZT_n Z^t, \tag{4.1.2}$$

seules les premières ligne et colonne sont non nulles. Cette matrice est donc de rang au plus deux, et peut donc être construite à l'aide de deux vecteurs, communément appelés « générateurs ». Si nous notons $\bar{r}$ la première colonne de T, alors les vecteurs $\bar{r}$ et $Z\bar{r}$ sont des générateurs. La matrice ∇T s'écrit en effet :

$$\nabla T_n = \bar{r}\bar{r}^t - Z\bar{r}\bar{r}^t Z^t.$$

A l'aide de (3.8) et (3.9), on peut montrer que les vecteurs

$$g_1 = \begin{pmatrix} 1 \\ a_1^{(n)} \\ \vdots \\ a_n^{(n)} \end{pmatrix} = \overline{a}^{(n)} \quad \text{et} \quad g_2 = \begin{pmatrix} 0 \\ a_n^{(n)} \\ \vdots \\ a_1^{(n)} \end{pmatrix} = \begin{pmatrix} 0 \\ J\mathbf{a}^{(n)} \end{pmatrix} \tag{4.1.3}$$

sont des générateurs de T_n^{-1}, et que la matrice ∇T_n^{-1} s'écrit alors

$$\nabla T_n^{-1} = \frac{1}{\sigma_n} \; g_1 \; g_1^t - \frac{1}{\sigma_n} \; g_2 \; g_2^t . \tag{4.1.4}$$

Après examen de l'expression $T_n^{-1} - ZT_n^{-1} Z^t$, on constate que T_n^{-1} peut être reconstruite facilement à partir de ∇T_n^{-1} ; plus précisément, puisque Z est nilpotente nous avons :

$$T_n^{-1} = \sum_{i=1}^{n} Z^i \nabla T_n^{-1} Z^{ti} . \tag{4.1.5}$$

En combinant (4.1.4) et (4.1.5), nous obtenons la décomposition parfois attribuée à *Gohberg et Semencul :*

$$T_n^{-1} = \frac{1}{\sigma_n} \{ L_1 L_1^t - L_2 L_2^t \} , \tag{4.1.6}$$

où L_1 et L_2 sont les matrices triangulaires Toeplitz ayant pour première colonne g_1 et g_2, respectivement. Notons pour terminer que l'expression (4.1.4) à (4.1.6) se généralisent à une classe plus large de matrices structurées que celle des matrices Toeplitz, mais ceci sort du cadre de notre propos.

4.2. Conditionnement

Considérons une matrice Toeplitz symétrique dont la première ligne est $(1 \; r_1, \ldots, r_n)$, et admettons que $|r_i| \leqslant 1$. Alors la majoration de sa norme L_1 est évidente, d'après (2.1.3) :

$$\| T_n \|_1 \leqslant (n+1) \max_i |r_i| = n + 1 . \tag{4.2.1}$$

D'autre part, la décomposition de Gohberg-Semencul (4.1.6) va nous permettre de majorer la norme de la matrice inverse grâce à (2.1.5) :

$$\| T_n^{-1} \|_1 \leqslant \frac{1}{|\sigma_n|} \{ \| L_1 \|_1 \, \| L_1^t \|_1 + \| L_2 \|_1 \, \| L_2^t \|_1 \} . \tag{4.2.2}$$

Or d'après (4.1.3),

$$\| L_1 \|_1 = \| L_1^t \|_1 = \| g_1 \|_1 = \| \bar{a}^{(n)} \|_1 \quad \text{et} \quad \| L_2 \|_1 = \| g_2 \|_1 = \| a^{(n)} \|_1 .$$

Il vient donc la majoration :

$$\| T_n^{-1} \|_1 \leqslant \frac{1}{|\sigma_n|} \{ \| \bar{a}^{(n)} \|_1^2 + \| a^{(n)} \|_1^2 \} . \tag{4.2.3}$$

Le conditionnement peut donc être borné par :

$$\kappa_1(T_n) \leqslant \frac{n+1}{|\sigma_n|} (1 + 2 \| a^{(n)} \|_1^2 + 2 \| a^{(n)} \|_1) . \tag{4.2.4}$$

D'après (3.5), le vecteur $\bar{a}^{(n)}$ s'exprime aussi récurcivement en fonction des coefficients de réflexion :

$$\bar{a}^{(n)} = \begin{pmatrix} \bar{a}^{(n-1)} \\ 0 \end{pmatrix} + k_n \begin{pmatrix} 0 \\ J\bar{a}^{(n-1)} \end{pmatrix} , \tag{4.2.5}$$

de sorte que, puisque $\bar{a}^{(0)} = 1$:

$$\| \bar{a}^{(n)} \|_p \leqslant \prod_{m=1}^{n} (1 + |k_m|) , \quad \forall p > 0 . \tag{4.2.6}$$

La borne (4.2.4) peut donc s'écrire aussi uniquement en fonction des coefficients k_m. Ce qui constitue la généralisation du résultat de Cybenko [9] :

$$\frac{\| \bar{a}^{(n)} \|_1}{|\sigma_n|} \leqslant \| T_n^{-1} \|_1 \leqslant \frac{1}{|\sigma_n|} \prod_{m=1}^{n} (1 + |k_m|)^2 \tag{4.2.7}$$

aux matrices Toeplitz symétriques indéfinies.

Dans le cas Toeplitz symétrique indéfini, le majorant (4.2.3) de $\| T_n^{-1} \|_1$ peut être sensiblement plus fin que (4.2.7). Le lecteur pourra vérifier à titre d'exemple que si $(r_0, r_1, r_2, r_3) = (1, 0{,}999, 0{,}9, 0{,}998)$ et $n = 3$, alors nous avons :

$$(k_1, k_2, k_3) = (-0{,}999, 49{,}0, 1{,}04),$$

$$(\sigma_1, \sigma_2, \sigma_3) = (0{,}002, -4{,}80, 0{,}396),$$

$$a^{(3)} = (1{,}04, -2{,}97, 1{,}03),$$

d'où nous tirons d'après (4.2.3) $\| T_3^{-1} \|_1 \leqslant 532{,}65$, alors qu'avec (4.2.7) nous obtenons $\| T_3^{-1} \|_1 \leqslant 1{,}05 \times 10^5$. Dans cet exemple, la valeur exacte de $\| T_3^{-1} \|_1$ est 15,26.

Ceci s'explique par le fait que, pour les matrices indéfinies, les coefficients $|k_i|$ peuvent être très grands alors que la norme $\| \overline{a^{(n)}} \|_p$ reste suffisamment petite. Et la majoration (4.2.6) devient irréaliste (i.e. trop lâche).

Contrairement à la borne (4.2.7) (proposée par Cybendo) de $\| T_n^{-1} \|_1$ basée sur la factorisation de Cholesky, la borne (4.2.3) est basée sur la notion de générateurs et exploite mieux la structure Toeplitz de la matrice. Ce qui justifie la meilleure qualité du majorant (4.2.3) que nous proposons. *Outre le fait que le majorant* (4.2.3) *est plus fin que* (4.2.7), *il se généralise au cas indéfini.*

OBSERVATIONS

1) Il importe de noter que la borne inférieure $\| a^{(n)} \|_1 / |\sigma_n|$ de $\| T_n^{-1} \|_1$ est atteinte pour les matrices Toeplitz circulantes (si $(a^{(n)} = Ja^{(n)})$ et leur inverse est aussi Toeplitz circulante dont la première ligne est $\overline{a}^{(n)}$.

2) Une matrice bien conditionnée peut avoir une sous-matrice diagonale T_i mal conditionnée. En effet, puisque

$$|\sigma_n| = |\sigma_i| \cdot \left| \prod_{m=i+1}^{n} (1 - k_m^2) \right|,$$

$|\sigma_n|$ peut être grand alors que $|\sigma_i|$ est très petit. Il suffit que l'autre produit $\left| \prod_{m=i+1}^{n} (1 - k_m^2) \right|$ soit très grand, c'est-à-dire quand les $|k_m|$ $(m > i)$ sont très grands.

Cependant, si la matrice Toeplitz est définie positive $(|k_m| < 1, m = 1, ..., n)$, le produit $\left| \prod_{m=i+1}^{n} (1 - k_m^2) \right|$ est toujours inférieur à 1 et $|\sigma_n| \leqslant |\sigma_i|$; et toutes les sous-matrices principales d'une matrice Toeplitz symétrique définie positive bien conditionnée seront aussi bien conditionnées. Dans le cas Toeplitz indéfini, une matrice bien conditionnée peut

avoir une sous-matrice principale mal conditionnée (presque singulière). Ceci confirme le résultat de Bunch [5].

3) L'équation (4.2.7) montre que $|\sigma_n|$ mesure la qualité du schéma de prédiction et, à chaque étape i, le conditionnement s'amplifie par le facteur $\left|1 / 1 - |k_i|\right|$.

Pour la plus petite valeur de $|\sigma_n|$, on doit s'attendre à une imprécision (essentiellement inévitable) dans la solution calculée due à la nature même du problème, quel que soit l'algorithme (même stable) de résolution utilisé.

4) Pour les matrices définies positives, le mauvais conditionnement augmente avec la taille de la matrice puisque $\sigma_i \leqslant \sigma_{i-1}$. Ainsi, on doit s'attendre à une grande imprécision sur les coefficients de prédiction linéaire dont l'ordre est très élevé, quelque soit l'algorithme utilisé. C'est pourquoi on pourrait condamner à tort l'algorithme de Levinson (qui est pourtant stable, *cf.* § 5).

5. STABILITÉ DE L'ALGORITHME DE LEVINSON

5.1. Position du problème

Il est bien connu que la précision des résultats en machine dépend à la fois du conditionnement du problème et de la stabilité de l'algorithme utilisé. Dans le paragraphe précédent, nous venons de discuter du conditionnement des systèmes linéaires Toeplitz ; et maintenant nous allons analyser les performances numériques de l'algorithme LD vis-à-vis du conditionnement du système linéaire qu'il résoud. Rappelons d'abord quelques résultats importants déjà connus :

- Cybenko a montré (par une analyse de type inverse des erreurs) que le vecteur résidu associé aux solutions de $Tx = -r$ calculées par l'algorithme de LD peut être borné [9] ; l'expression de cette borne est comparable à celle que l'on obtient dans la méthode de Choleski, connue pour sa stabilité. En fait, il a prouvé la *stabilité faible* de l'algorithme LD puisque

$$\frac{\|x - \tilde{x}\|_p}{\|x\|_p} \leqslant \kappa_p(T) \frac{\|\Delta b\|_p}{\|b\|_p},$$

et il a montré que $\|\Delta b\|_p$ est borné (avec $b = r$ et $p = 1$) par une expression « relativement acceptable ». Ceci veut dire que si la matrice est bien conditionnée, la solution calculée par l'algorithme de Levinson a une bonne précision relative.

- Bunch a montré [5] que les algorithmes basés sur le partitionnement de la matrice peuvent être instables, sauf peut-être si la matrice est symétrique définie positive. L'algorithme de Levinson est de cette classe.

Dans cette section, nous étendons d'abord le résultat de Cybenko au cas indéfini et discutons ensuite de la stabilité inverse et la stabilité forte de l'algorithme LD. Deux approches d'analyse (directe et inverse) d'erreurs sont respectivement présentées.

5.2. Précision numérique des résultats et stabilité faible

A cause de l'arithmétique finie des ordinateurs, les résultats obtenus ne sont qu'approchés et seront considérés comme complets que s'ils sont accompagnés d'une estimation de leur précision. Une analyse directe de la propagation des erreurs d'arrondi dans un algorithme permet d'obtenir un majorant de l'erreur commise sur la solution finale. Bien que cette borne d'erreur soit souvent irréaliste (i.e. trop lâche) en pratique, elle permet de détecter les instabilités (numériques de l'algorithme ou mathématiques du problème) éventuelles, et d'en déduire une classe des données pour lesquelles l'algorithme utilisé est faiblement stable (la solution précise quand le problème est bien conditionné). Nous présentons ici cette approche d'analyse pour l'algorithme LD.

A l'aide d'une analyse directe de la propagation des erreurs, nous établissons d'abord un majorant de l'erreur commise sur la solution calculée et en déduirons que l'algorithme LD est faiblement stable pour les matrices fortement bien conditionnées (c'est-à-dire sous-matrices principales sont aussi bien conditionnées). Ceci étend le résultat de Cybenko au cas indéfini.

Pour ce faire, notons Δx la différence (appelée souvent erreur absolue) entre la solution calculée $\tilde{x}$ et sa valeur exacte x, et étudions d'abord comment les erreurs d'arrondi se propagent d'une étape donnée $i-1$ à l'étape suivante i. Pour $i = 1$ jusqu'à n, on a d'après (3.5) :

$$\Delta w_i = \sum_{j=1}^{i-1} r_{i-j}\, \Delta a_j^{(i-1)} + \omega_i = (Jr^{(i-1)})^t\, \Delta a^{(i-1)} + \omega_i , \qquad (5.2.1)$$

$$\Delta k_i = -\frac{1}{\sigma_{i-1}} (\Delta w_i + k_i\, \Delta \sigma_{i-1}) + \eta_i ,$$

$$= -\frac{1}{\sigma_{i-1}} \{ (Jr^{(i-1)})^t\, \Delta a^{(i-1)} + k_i\, \Delta \sigma_{i-1} + \omega_i - \sigma_{i-1}\, \eta_i \} \qquad ((5.2.2))$$

$$\Delta \sigma_i = (1 - k_i^2)\, \Delta \sigma_{i-1} - 2\, k_i\, \sigma_{i-1}\, \Delta k_i + \gamma_i , \qquad (5.2.3)$$

et pour $j = 1$ à $i-1$, on a

$$\Delta a_j^{(i)} = \Delta a_j^{(i-1)} + k_i\, \Delta a_{i-j}^{(i-1)} + a_{i-j}^{(i-1)}\, \Delta k_i + \alpha_j^{(i)} , \qquad (5.2.4a)$$

$$\Delta a_i^{(i)} = \Delta k_i , \qquad (5.2.4b)$$

ou encore sous forme vectorielle

$$\Delta a^{(\iota)} = \begin{bmatrix} \Delta a^{(\iota-1)} + k_\iota J \Delta a^{(\iota-1)} \\ 0 \end{bmatrix} + \Delta k_\iota \begin{bmatrix} J a^{(\iota-1)} \\ 1 \end{bmatrix} + \begin{bmatrix} \alpha^{(\iota)} \\ 0 \end{bmatrix}. \quad (5.2.5)$$

Les erreurs absolues ω_ι, η_ι, γ_ι et $\alpha_j^{(\iota)}$ s'écrivent, au premier ordre (c'est-à-dire $(O(\varepsilon_\iota \varepsilon_j) = 0)$:

$$\omega_\iota = \sum_{j=1}^{\iota-1} r_{\iota-j} a_j^{(\iota-1)} \varepsilon_j^* + \sum_{j=2}^{\iota-1} s_j \varepsilon_j^+ + w_\iota \varepsilon_\iota^+ \quad \text{avec} \quad s_j = \sum_{m=1}^{j} a_{\iota-m}^{(\iota-1)} r_m, \quad (5.2.6)$$

$$\eta_i = k_\iota \varepsilon_\iota^*, \quad (5.2.7)$$

$$\gamma_\iota = \sigma_\iota(\varepsilon_\iota^- + \varepsilon_\iota^*) - k_\iota^2 \sigma_{\iota-1} \varepsilon_\iota^*, \quad (5.2.8)$$

$$\alpha_j^{(\iota-1)} = k_\iota a_{\iota-j}^{(\iota-1)} \varepsilon_j^* + a_j^{(\iota)} \varepsilon_j^+. \quad (5.2.9)$$

Dans les expressions ci-dessus, ε_j^+, ε_j^-, ε_j^* symbolisent des erreurs arithmétiques relatives, majorées par l'unité d'arrondissage $\varepsilon_M = \beta^{1-t}/2$, et engendrées respectivement par les opérations +, – et *.

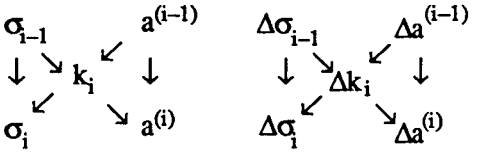

Figure 1. — Graphe de dépendance des variables calculées dans l'algorithme de LD.

En observant les équations (5.2.1) à (5.2.5), on peut remarquer que les erreurs Δk_ι, $\Delta a^{(\iota)}$, $\Delta \sigma_\iota$ sont localement couplées. Ceci traduit directement les effets de la dépendance des variables dans l'algorithme (*cf. fig.* 1) et entraîne de sérieuses complications pour déterminer un majorant de l'erreur sur la solution finale $a^{(n)}$. Pour découpler ces erreurs et obtenir un majorant de l'erreur $\Delta a^{(n)}$ sur la solution finale, nous utilisons constamment les lemmes suivant :

LEMME 5.2.10 : *Avec les notations ci-dessus et* $\phi_t = \omega_t - \sigma_{t-1}\eta_t$, *on a*

$$\Delta\sigma_t = (r^{(t)})^t \Delta a^{(t)} + P_t$$

avec

$$P_t = P_{t-1} - (r^{(t-1)})^t \alpha^{(t)} + k_t \phi_t + \gamma_t \; ; \; P_0 = 0 \; .$$

Démonstration : Ce résultat est dû à Cybenko ([9], formule (5.11)) dans le cas Toeplitz symétrique défini positif. On peut montrer qu'il reste valide même dans le cas indéfini. Une démonstration simple et claire est donnée en annexe. Notons également que P_t pourrait représenter l'erreur d'arrondi sur σ_t engendrée lors de son calcul par la formule $\sigma_t = 1 + (r^{(t)})^t a^{(t)}$, si cette formule était utilisée au lieu de $\sigma_{t-1}(1 - k_t^2)\,\sigma_{t-1}$ pour calculer σ_t.

LEMME 5.2.11 : *Soient* $\{x_t\}_{t \in \mathbb{N}}$, $\{\alpha_t\}_{t \in \mathbb{N}}$, $\{\beta_t\}_{t \in \mathbb{N}}$ *des suites réelles positives vérifiant*

$$x_t \leqslant \alpha_t x_{t-1} + \beta_t \, .$$

Alors

$$x_t \leqslant \left[\prod_{m=1}^{t} \alpha_m \right] x_0 + \sum_{j=1}^{t-1} \left[\prod_{m=j+1}^{t} \alpha_m \right] \beta_j + \beta_t \, .$$

La démonstration est immédiate par récurrence.

LEMME 5.2.12 : *Soit* $\{\alpha_t\}_{t \in \mathbb{N}}$ *une suite réelle positive. On a*

$$\sum_{j=1}^{t-1} \left[\prod_{m=j+1}^{t} (1 + \alpha_m) \right] \alpha_j + \alpha_t = \prod_{m=1}^{t} (1 + \alpha_m) - 1 \qquad (5.2.12a)$$

et de même

$$\sum_{j=1}^{t} \left[\prod_{m=1}^{j-1} (1 + \alpha_m) \right] \alpha_j = \prod_{m=1}^{t} (1 + \alpha_m) - 1 \, . \qquad (5.2.12b)$$

La démonstration est immédiate par récurrence.

Le lemme (5.2.10) permet d'exprimer l'erreur $\Delta a^{(\iota)}$ sur $a^{(\iota)}$ uniquement en fonction de son erreur $\Delta a^{(\iota-1)}$ à l'étape précédente. En effet, à partir de (5.2.2) et le lemme (5.2.10), il vient que

$$\begin{aligned}\Delta k_\iota \begin{bmatrix} Ja^{(\iota-1)} \\ 1 \end{bmatrix} &= -\frac{1}{\sigma_{\iota-1}} \begin{bmatrix} Ja^{(\iota-1)} \\ 1 \end{bmatrix} \{ (Jr^{(\iota-1)})^t \, \Delta a^{(\iota-1)} \\ &\quad + k_\iota \{ (Jr^{(\iota-1)})^t \, \Delta Ja^{(\iota-1)} + P_{\iota-1} \} + \phi_\iota \} \\ &= -\frac{1}{\sigma_{\iota-1}} \left\{ \begin{bmatrix} Ja^{(\iota-1)} \\ 1 \end{bmatrix} [(Jr^{(\iota-1)})^t \quad 0] \right\} \\ &\quad \times \begin{bmatrix} \Delta a^{(\iota-1)} + k_\iota \, \Delta Ja^{(\iota-1)} \\ 0 \end{bmatrix} \\ &\quad - \frac{1}{\sigma_{\iota-1}} \begin{bmatrix} Ja \\ 1 \end{bmatrix} (k_\iota P_{\iota-1} + \phi_\iota) ,\end{aligned}$$

et la relation (5.2.5) s'écrit sous la forme

$$\Delta a^{(\iota)} = B_\iota \begin{bmatrix} \Delta a^{(\iota-1)} + k_\iota \, \Delta Ja^{(\iota-1)} \\ 0 \end{bmatrix} + Q^{(\iota)} \qquad (5.2.13a)$$

où

$$B_\iota = I_\iota - \frac{1}{\sigma_{\iota-1}} \left\{ \begin{bmatrix} Ja^{(\iota-1)} \\ 1 \end{bmatrix} [(Jr^{(\iota-1)})^t \quad 0] \right\} , \qquad (5.2.13b)$$

et

$$Q^{(\iota)} = -\frac{1}{\sigma_{\iota-1}} \begin{bmatrix} Ja^{(\iota-1)} \\ 1 \end{bmatrix} (k_\iota P_{\iota-1} + \phi_\iota) + \begin{bmatrix} \alpha^{(\iota)} \\ 0 \end{bmatrix} . \qquad (5.2.13c)$$

La relation (5.2.13) exprime l'erreur $\Delta a^{(\iota)}$ sur la solution $a^{(\iota)}$ uniquement en fonction de son erreur $\Delta a^{(\iota-1)}$ à l'étape précédente et des erreurs $Q^{(\iota)}$ d'arrondi-propagées sur $a^{(\iota)}$. On en résulte que

$$\| \Delta a^{(\iota)} \|_1 \leq \| B_\iota \|_1 (1 + |k_\iota|) \| \Delta a^{(\iota-1)} \|_1 + \| Q^{(\iota)} \|_1 , \qquad (5.2.14a)$$

où

$$\|B_t\|_1 \leq 1 + \frac{1}{|\sigma_{t-1}|} \|\bar{a}^{(t-1)}\|_1 \|r^{(t-1)}\|_\infty , \qquad (5.2.14b)$$

et

$$\|Q^{(t)}\|_1 \leq \frac{1}{|\sigma_{t-1}|} \|\bar{a}(i-1)\|_1 (|k_t||P_{t-1}| + |\phi_t|) + \|\alpha^{(t)}\|_1 . \qquad (5.2.15)$$

Procédons à présent à la majoration de $\|Q^{(t)}\|_1$. A partir de (5.2.6) il vient que

$$|\omega_t| \leq (\|r^{(t-1)}\|_\infty \|a^{(t-1)}\|_1 + (i-2) \max\{|s_j|\} + |w_t|)\, \varepsilon_M .$$

Mais puisque

$$\max\{|s_j|, j = 2 \text{ à } i-1\} \leq \sum_{j=1}^{t-1} |r_{t-j} a_j^{(t-1)}| \leq \|a^{(t-1)}\|_1 \|r^{(t-1)}\|_\infty ,$$

nous pouvons en déduire une première majoration

$$\begin{aligned} |\omega_t| &\leq \{(i-1)\|a^{(t-1)}\|_1 \|r^{(t-1)}\|_\infty + |w_t|\}\, \varepsilon_M \\ &\leq i \|\bar{a}^{(t-1)}\|_1 \|r^{(t)}\|_\infty \varepsilon_M , \end{aligned} \qquad (5.2.16)$$

en notant que $|w_t| \leq \|\bar{a}^{(t-1)}\|_1 \|r^{(t)}\|_\infty$.

La majoration de l'erreur γ_t s'obtient à partir de (5.2.8) :

$$|\gamma_t| \leq \{2|\sigma_t| + k_t^2 |\sigma_{t-1}|\}\, \varepsilon_M . \qquad (5.2.17)$$

La majoration sur $\alpha^{(t)}$ se calcule à partir de (5.2.9) et du fait que $\|a^{(t)}\|_1 \leq (1 + |k_t|)\|a^{(t-1)}\|_1 + |k_t|$:

$$\begin{aligned} \|\alpha^{(t)}\|_1 &\leq |k_t|\, \|a^{(t-1)}\|_1 \varepsilon_M + (\|a^{(t)}\|_1 - |k_t|)\, \varepsilon_M \\ &\leq \|a^{(t-1)}\|_1 (1 + 2|k_t|)\, \varepsilon_M . \end{aligned} \qquad (5.2.18)$$

En utilisant les inégalités

$$|k_t|\, |\sigma_{t-1}| = |w_t| \leq \|\bar{a}^{(t-1)}\|_1 \|r^{(t)}\|_\infty , \qquad (5.2.19a)$$

$$k_n^2 |\sigma_{t-1}| = |k_t|\, |w_t| \leq (1 + |k_t|)\, \|\bar{a}^{(t-1)}\|_1 \|r^{(t)}\|_\infty \qquad (5.2.19b)$$

et

$$|\sigma_t| = \left| 1 + \sum_{j=1}^{t} r_j \, a_j^{(t)} \right| \leqslant \|\bar{a}^{(t)}\|_1 \|\bar{r}^{(t)}\|_\infty \leqslant (1 + |k_t|) \, \|\bar{a}^{(t-1)}\|_1 \|\bar{r}^{(t)}\|_\infty , \tag{5.2.19c}$$

on déduit que

$$|\phi_t| \leqslant \varepsilon_M (i+1) \, \|\bar{a}^{(t-1)}\|_1 \|r^{(t)}\|_\infty . \tag{5.2.20}$$

Et du lemme (5.2.10), il vient que

$$\begin{aligned} |P_t| &\leqslant |P_{t-1}| + \|\alpha^{(t)}\|_1 \, \|r^{(t-1)}\|_\infty + |k_t| \, |\phi_t| + |\gamma_t| \\ &\leqslant |P_{t-1}| + \varepsilon_M \{ 2(1 + |k_t|) \, \|a^{(t-1)}\|_1 \, \|r^{(t-1)}\|_\infty \\ &\quad + (i+2) \, |k_t| \, \|\bar{a}^{(t-1)}\|_1 \, \|r^{(t)}\|_\infty + 2 \|\bar{a}^{(t)}\|_1 \, \|\bar{r}^{(t)}\|_\infty \\ &\leqslant |P_{t-1}| + \varepsilon_M \{ 4(1 + |k_t|) \, \|\bar{a}^{(t-1)}\|_1 \, \|\bar{r}^{(t)}\|_\infty \\ &\quad + (i+2) \, |k_t| \, \|\bar{a}^{(t-1)}\|_1 \, \|r^{(t)}\|_\infty . \end{aligned}$$

Par le lemme (5.2.11), la relation (4.2.6) et puis le lemme (5.2.12*b*), on déduit que

$$\begin{aligned} |P_t| &\leqslant \left\{ 4 \sum_{j=1}^{t} (1 + |k_j|) \, \|\bar{a}^{(j-1)}\|_1 \, \|\bar{r}^{(j)}\|_\infty \right. \\ &\qquad \left. + \sum_{j=1}^{t} (j+2) \, |k_j| \, \|\bar{a}^{(j-1)}\|_1 \, \|r^{(j)}\|_\infty \right\} \varepsilon_M \\ &\leqslant \left\{ 4 \, i \prod_{m=1}^{t} (1 + |k_m|) \, \|\bar{r}^{(t)}\|_\infty + \right. \\ &\qquad \left. + (i+2) \left[\prod_{m=1}^{t} (1 + |k_m|) - 1 \right] \|r^{(t)}\|_\infty \right\} \varepsilon_M . \end{aligned}$$

Il en résulte que

$$|P_i| \leqslant \varepsilon_M(5\,i+2) \prod_{m=1}^{i} (1 + |k_m|) \, \|\bar{r}^{(i)}\|_\infty , \tag{5.2.21}$$

et

$$\begin{aligned} |k_i| \, |P_{i-1}| + |\phi_i| &\leqslant \left\{ (5\,i-3) \, |k_i| \prod_{m=1}^{i-1} (1 + |k_m|) \, \|\bar{r}^{(i-1)}\|_\infty \right. \\ &\qquad \left. + (i+1) \, \|\bar{a}^{(i-1)}\|_1 \, \|r^{(i)}\|_\infty \right\} \varepsilon_M , \\ &\leqslant \left\{ (1 + |k_i|)(5\,i-3) \prod_{m=1}^{i-1} (1 + |k_m|) \, \|\bar{r}^{(i)}\|_\infty \right\} \varepsilon_M \\ &= \left\{ (5\,i-3) \prod_{m=1}^{i} (1 + |k_m|) \, \|\bar{r}^{(i)}\|_\infty \right\} \varepsilon_M . \end{aligned} \tag{5.2.22}$$

Si nous posons

$$c_{i-1} = \frac{1}{|\bar{\sigma}_{i-1}|} \, \|\bar{a}^{(i-1)}\|_1 \, \|\bar{r}^{(i-1)}\|_\infty , \tag{5.2.23}$$

et remarquons encore que

$$\|\alpha^{(i)}\|_1 = 2(1 + |k_i|) \, \|\bar{a}^{(i-1)}\|_1 \, \varepsilon_M \leqslant 2 \prod_{m=1}^{i} (1 + |k_m|) \, \varepsilon_M , \tag{5.2.24}$$

nous déduisons, de (5.2.15) à (5.2.24). que

$$\begin{aligned} \|Q^{(i)}\|_1 &\leqslant \left\{ \prod_{m=1}^{i} (1 + |k_m|)((5\,i-3)\,c_{i-1} + 2) \right\} \varepsilon_M \\ &\leqslant \left\{ \prod_{m=1}^{i} (1 + |k_m|)(5\,i-1)\,c_{i-1} \right\} \varepsilon_M \text{ (en notant que } c_{i-1} \geqslant 1) \end{aligned} \tag{5.2.25}$$

et (5.2.14) devient

$$\| \Delta a^{(i)} \|_1 \leq (1 + c_{i-1})(1 + |k_i|) \| \Delta a^{(i-1)} \|_1 + \left\{ (5\,i - 1)\, c_{i-1} \prod_{m=1}^{i} (1 + |k_m|) \right\} \varepsilon_M .$$

Par le lemme (5.2.11), on déduit que

$$\| \Delta a^{(i)} \|_1 \leq \prod_{m=1}^{i} (1 + |k_m|) \left\{ \sum_{j=1}^{i-1} \left[\prod_{m=j+1}^{i} (1 + |c_{m-1}|) \right] \times (5\,j - 1)\, c_{j-1} + (5\,i - 1)\, c_{i-1} \right\} \varepsilon_M ,$$

$$\leq (5\,i - 1) \prod_{m=1}^{i} (1 + |k_m|) \left\{ \sum_{j=1}^{i-1} \left[\prod_{m=j+1}^{i} (1 + c_{m-1}) \right] c_{j-1} + c_{i-1} \right\} \varepsilon_M$$

et la relation (5.2.12*a*) permet de conclure que

$$\| \Delta a^{(i)} \|_1 \leq (5\,i - 1)\, \varepsilon_M \prod_{m=1}^{i} (1 + |k_m|) \left[\prod_{m=1}^{i} (1 + c_{m-1}) - 1 \right]. \qquad (5.2.26a)$$

Il vient alors que l'erreur relative sur la solution calculée $\tilde{a}^{(i)}$ est bornée par

$$\frac{\| \Delta a^{(i)} \|_1}{\| \tilde{a}^{(i)} \|_1} \leq (5\,i - 1)\, \varepsilon_M \frac{\prod_{m=1}^{i} (1 + |k_m|)}{\| \tilde{a}^{(i)} \|_1} \left[\prod_{m=1}^{i} (1 + c_{m-1}) - 1 \right]. \qquad (5.2.26b)$$

Majorant de l'erreur Δk_i *commise sur* k_i :

De la relation (5.2.2) et le lemme (5.2.10), on a

$$\Delta k_i = -\frac{1}{\sigma_{i-1}} \{ (J r^{(i-1)} + k_i\, r^{(i-1)})\, \Delta a^{(i-1)} + \phi_i + k_i\, P_{i-1} \} \qquad (5.2.27)$$

et, par majoration des normes et les relations (5.2.22) et (5.2.26*a*), on déduit que

$$\Delta k_i = -\frac{1}{\sigma_{i-1}} \{ (1 + |k_i|) \, \| \Delta a^{(i-1)} \|_1 \, \| r^{(i-1)} \|_\infty + |\phi_i| + |k_i| \, |P_{i-1}| \}$$

et

$$|\Delta k_i| \leqslant \frac{1}{\sigma_{i-1}} (5\,i - 3)\, \varepsilon_M \prod_{m=1}^{i} (1 + |k_m|) \prod_{m=1}^{i-1} (1 + c_{m-1}) \, \| \bar{r}^{(i)} \|_\infty . \quad (5.2.28)$$

ou encore

$$\frac{|\Delta k_i|}{1 + |k_i|} \leqslant (5\,i - 3)\, \varepsilon_M \frac{\prod_{m=1}^{i-1} (1 + |k_m|)}{|\sigma_{i-1}|} \| \bar{r}^{(i)} \|_\infty \prod_{m=1}^{i-1} (1 + c_{m-1}) . \quad (5.2.29)$$

Majorant de l'erreur $|\Delta\sigma_i|$ *commise sur* σ_i :

Par majoration des normes dans le lemme (5.2.10), on a

$$|\Delta\sigma_i| \leqslant \| \Delta a^{(i)} \|_1 \, \| r^{(i)} \|_\infty + |P_i| \quad (5.2.30)$$

et, de (5.2.26*a*) et (5.2.21), il résulte que

$$|\Delta\sigma_i| \leqslant (5\,i + 2)\, \varepsilon_M \prod_{m=1}^{i} (1 + |k_m|) \prod_{m=1}^{i} (1 + c_{m-1}) \, \| \bar{r}^{(i)} \|_\infty . \quad (5.2.31)$$

Il vient que l'erreur relative commise sur $\tilde{\sigma}_i$ est bornée par

$$\frac{|\Delta\sigma_i|}{|\sigma_i|} \leqslant (5\,i + 2)\, \varepsilon_M D_i \prod_{m=1}^{i} (1 + c_{m-1}) , \quad (5.2.32)$$

où

$$D_i = \frac{\prod_{m=1}^{i} (1 + |k_m|)}{|\sigma_i|} \| \bar{r}^{(i)} \|_\infty . \quad (5.2.33)$$

Noter que $c_i \leqslant D_i$ pour tout $i \geqslant 0$.

OBSERVATIONS

1) Les majorants (5.2.26*b*), (5.2.29) et (5.2.32) des erreurs commises sur $a^{(i)}$, Δk_i, σ_i resteront petits (acceptables) quand tous les c_{m-1} ($m = 1$ à n) sont

assez petits. Puisque $c_m \leq \kappa_1(T_m)$ pour tout $m \geq 0$, on déduit que l'erreur relative sur la solution sera petite en particulier pour les matrices dont les sous-matrices diagonales sont bien conditionnées. De telles matrices seront appelées ici les *matrices fortement bien conditionnées.* On conclut que *l'algorithme LD est faiblement stable pour toute matrice Toeplitz symétrique fortement bien conditionnée (même indéfinie).*

2) Dans le cas symétrique défini positif, il est bien connu que toute sous-matrice diagonale principale d'une matrice T_m symétrique définie positive est mieux conditionnée que la matrice T_m elle-même [5]. Il en résulte que « fortement bien conditionnée » est équivalent à « bien conditionnée » dans le cas défini positif, et nous en déduisons que notre résultat généralise celui de Cybenko [9] au cas indéfini fortement bien conditionné.

Une analyse directe des erreurs permet d'obtenir un majorant de l'erreur sur la solution, mais elle présente deux inconvénients : d'une part, ce majorant est souvent irréaliste (jamais atteint en pratique) puisque l'on travaille avec des majorants des erreurs d'arrondi engendrés à chaque étape de l'algorithme alors que ces erreurs peuvent se compenser, et d'autre part, on ne sait pas distinguer l'imprécision des résultats due à l'algorithme et celle provenant du conditionnement du problème à résoudre. L'analyse inverse d'erreurs nous apporte quelques éléments de réponse.

5.3. W-stabilité (stabilité inverse) de l'algorithme LD

A l'aide d'une analyse des erreurs d'arrondi, nous venons de démontrer que l'algorithme LD est faiblement stable pour les matrices Toeplitz symétriques, non seulement définies positives, mais également indéfinies. Pour prouver sa stabilité inverse (au sens de Wilkinson), il faut montrer que la solution calculée $\{\tilde{a}^{(n)}, \tilde{\sigma}_n\}$ de $T_n \begin{bmatrix} 1 \\ a^{(n)} \end{bmatrix} = \sigma_n e_1$ est la solution exacte d'un système perturbé

$$(T_n + \delta T_n) \begin{bmatrix} 1 \\ \tilde{a}^{(n)} \end{bmatrix} = \tilde{\sigma}_n e_1 = (\sigma_n + \Delta\sigma_n) e_1 , \tag{5.3.1}$$

où la norme de la matrice δT_n est relativement petite. Et il sera fortement stable si la matrice δT_n de perturbation associée est aussi Toeplitz symétrique. Le problème consiste alors à construire la matrice δT_n (et souvent un majorant de sa norme) et en déduire une classe de matrices T_n pour lesquelles la norme de δT_n est relativement petite.

De (5.3.1), on en déduit que

$$\delta T_n \begin{bmatrix} 1 \\ \tilde{a}^{(n)} \end{bmatrix} = \bar{d}^{(n)} \tag{5.3.2}$$

avec

$$\bar{d}^{(n)} = \tilde{\sigma}_e e_1 - T_n \begin{bmatrix} 1 \\ \tilde{a}^{(n)} \end{bmatrix} = (\sigma_n + \Delta\sigma_n) e_1 - T_n \left(\begin{bmatrix} 1 \\ a^{(n)} \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta a^{(n)} \end{bmatrix} \right) \quad (5.3.3)$$

$$= \Delta\sigma_n e_1 - T_n \begin{bmatrix} 0 \\ \Delta a^{(n)} \end{bmatrix}.$$

Ainsi, toute solution δT_n de l'équation matricielle (5.3.2) vérifie la relation (5.3.1) et conviendrait donc comme perturbation sur T_n.

THÉORÈME 5.3.4 : *Soient u et v deux vecteurs de $\mathbb{R}^n$. Alors la solution*

$$X = \frac{1}{u^t u} v u^t \quad (5.3.4)$$

de l'équation matricielle $Xu = v$ a la plus petite norme parmi toutes les solutions de cette équation matricielle.

En effet, il est facile de vérifier que (5.3.4) est une solution. Elle est de norme minimale puisque la borne inférieure

$$\|v\| / \|u\| \leqslant \|X\|$$

des normes des solutions est atteinte.

Par le théorème (5.3.4) et la relation (5.3.2), on déduit le résultat important suivant :

THÉORÈME 5.3.5 : *La matrice δT_n*

$$\delta T_n = \frac{1}{(\tilde{\tilde{a}}^{(n)})^t \tilde{\tilde{a}}^{(n)}} \bar{d}^{(n)} (\tilde{\tilde{a}}^{(n)})^t \quad \text{où} \quad (\tilde{\tilde{a}}^{(n)})^t = [1, (\tilde{a}^{(n)})^t] \quad (5.3.5a)$$

est une perturbation (sur T_n) de 2-norme minimale parmi toutes les matrices δT_n qui satisfont le système perturbé $(T_n + \delta T_n) \begin{bmatrix} 1 \\ \tilde{a}^{(n)} \end{bmatrix} = \tilde{\sigma}_n e_1$. De plus,

$$\|\delta T_n\|_2 = \|\bar{d}^{(n)}\|_2 / \|\tilde{\tilde{a}}^{(n)}\|_2, \quad (5.3.5b)$$

$$\|\delta T_n\|_\infty = \frac{1}{\|\tilde{\tilde{a}}^{(n)}\|_2^2} \|\bar{d}^{(n)}\|_\infty \|\tilde{\tilde{a}}^{(n)}\|_1, \quad (5.3.5c)$$

$$\|\delta T_n\|_1 = \frac{1}{\|\tilde{\tilde{a}}^{(n)}\|_2^2} \|\bar{d}^{(n)}\|_1 \|\tilde{\tilde{a}}^{(n)}\|_\infty \leqslant \|d^{(n)}\|_1 / \|\tilde{\tilde{a}}^{(n)}\|_2. \quad (5.3.5d)$$

L'algorithme LD est stable si la norme $\|\delta T_n\|$ (bornée par $(\|\bar{d}^{(n)}\|_1 / \|\tilde{\tilde{a}}^{(n)}\|_2)$ est relativement petite par rapport à $\|T_n\|$. Noter que la matrice δT_n peut être explicitement construite puisque les vecteurs $\bar{d}^{(n)}$ et $\tilde{\tilde{a}}^{(n)}$ sont calculables. Mais, dans la théorie des erreurs, on se contente du majorant. Ainsi, pour prouver la stabilité de l'algorithme LD, il nous reste à calculer un majorant de la norme de $\bar{d}^{(n)}$ et en déduire une classe de matrices T_n pour lesquelles l'algorithme est stable. De (5.3.3) et le lemme (5.2.10), nous avons que

$$\bar{d}^{(n)} = \Delta\sigma_n e_1 - T_n \begin{bmatrix} 0 \\ \Delta a^{(n)} \end{bmatrix} = \begin{bmatrix} \Delta\sigma_n (r^{(n)})^t \Delta a^{(n)} \\ - T_{n-1} \Delta a^{(n)} \end{bmatrix} = \begin{bmatrix} P_n \\ - T_{n-1} \Delta a^{(n)} \end{bmatrix}$$

et, en utilisant les relations (5.2.5), (5.2.2) et le lemme (5.2.10),

$$T_{n-1} \Delta a^{(n)} = \begin{bmatrix} T_{n-2} \{ \Delta a^{(n-1)} + k_n \Delta J a^{(n-1)} \} \\ - k_n P_{n-1} \end{bmatrix} + \begin{bmatrix} 0 \\ - \phi_n \end{bmatrix} + T_{n-1} \begin{bmatrix} \alpha^{(n)} \\ 0 \end{bmatrix}.$$

Il résulte que

$$\bar{d}^{(n)} = \begin{bmatrix} P_n \\ - T_{n-1} \Delta a^{(n)} \end{bmatrix} = \begin{bmatrix} P_{n-1} \\ - T_{n-2} \{ \Delta a^{(n-1)} + k_n J \Delta a^{(n-1)} \} \\ - k_n P_{n-1} \end{bmatrix} +$$

$$+ \begin{bmatrix} k_n \phi_n + \gamma_n \\ 0 \\ \phi_n \end{bmatrix} + \begin{bmatrix} - (r^{(n-1)})^t \alpha^{(n)} \\ - T_{n-1} \begin{bmatrix} \alpha^{(n)} \\ 0 \end{bmatrix} \end{bmatrix},$$

ou encore

$$\bar{d}^{(n)} = \begin{bmatrix} \bar{d}^{(n-1)} \\ 0 \end{bmatrix} + k_n \begin{bmatrix} 0 \\ J\bar{d}^{(n-1)} \end{bmatrix} + \theta^{(n)}, \tag{5.3.6}$$

avec

$$\theta^{(n)} = \begin{bmatrix} k_n \phi_n + \gamma_n \\ 0 \\ \phi_n \end{bmatrix} - T_n \begin{bmatrix} 0 \\ \alpha^{(n)} \\ 0 \end{bmatrix}. \tag{5.3.7}$$

Et on en déduit que

$$\|\bar{d}^{(n)}\| \leq (1 + |k_n|)\|\bar{d}^{(n-1)}\| + \|\theta^{(n)}\| \tag{5.3.8}$$

et, en utilisant le lemme (5.2.11),

$$\|\bar{d}^{(n)}\| \leq \sum_{i=1}^{n-1} \left[\prod_{m=i+1}^{n} (1 + |k_m|) \right] \|\theta^{(i)}\| + \|\theta^{(n)}\| . \tag{5.3.9}$$

En particulier,

$$\|\bar{d}^{(n)}\|_1 \leq \sum_{i=1}^{n-1} \left[\prod_{m=i+1}^{n} (1 + |k_m|) \right] \|\theta^{(i)}\|_1 + \|\theta^{(n)}\|_1 . \tag{5.3.10}$$

Procédons maintenant à la majoration de $\|\theta^{(n)}\|_1$. Par majoration des normes et les relations (5.2.20) et (5.2.17), on a que

$$\|\theta^{(n)}\|_1 \leq (1 + |k_n|)|\phi_n| + \{2|\sigma_n| + k_n^2|\sigma_{n-1}|\} \varepsilon_M + \|T_n\|_1 \|\alpha^{(n)}\|_1 .$$

En utilisant (5.2.20), (5.2.19) et (5.2.18) dans l'inégalité précédente, on arrive à

$$\|\theta^{(n)}\|_1 \leq \{(n+1)(1 + |k_n|)\|\bar{a}^{(n-1)}\|_1 \|\bar{r}^{(n)}\|_\infty$$

$$+ 3(1 + |k_n|)\|\bar{a}^{(n-1)}\|_1 \|\bar{r}^{(n)}\|_\infty + \|T_n\|_1 (1 + 2|k_n|)\|a^{(n)}\|_1\} \varepsilon_M .$$

Il vient, en notant que $\|T_n\|_1 \leq (n+1)\|\bar{r}^{(n)}\|_\infty$ et l'inégalité (4.2.6), que

$$\|\theta^{(n)}\|_1 \leq (3n+6)(1 + |k_n|)\|\bar{a}^{(n-1)}\|_1 \|\bar{r}^{(n)}\|_\infty \varepsilon_M$$

$$\leq (3n+6) \prod_{m=1}^{n} (1 + |k_m|)\|\bar{r}^{(n)}\|_\infty \varepsilon_M . \tag{5.3.11}$$

De (5.3.10) et la majoration (5.3.11), on déduit alors que

$$\|\bar{d}^{(n)}\|_1 \leq \varepsilon_M f(n) \left[\prod_{m=1}^{n} (1 + |k_m|) \right] \|\bar{r}^{(n)}\|_\infty \tag{5.3.12}$$

où $f(n) = \sum_{i=1}^{n} (3i+6) = \frac{3}{2} n(n+5)$,

et de (5.3.5*d*), il vient que

$$\|\delta T_n\|_1 \leqslant \frac{3}{2} n(n+5)\,\varepsilon_M \frac{\prod_{m=1}^{n} (1+|k_m|)}{\|\tilde{\bar{a}}^{(n)}\|_2} \|\bar{r}^{(n)}\|_\infty + O(\varepsilon_M^2)\,. \quad (5.3.13)$$

D'où le résultat suivant :

PROPOSITION 5.3.14 : *La solution calculée* $\{\tilde{a}^{(n)}, \tilde{\sigma}_n\}$ *du système* $T_n \begin{bmatrix} 1 \\ a^{(n)} \end{bmatrix} = \sigma_n e_1$ *par l'algorithme LD satisfait le système linéaire* $(T_n + \delta T_n) \begin{bmatrix} 1 \\ \tilde{a}^{(n)} \end{bmatrix} = \tilde{\sigma}_n e_1$, *où* δT_n *est donnée par* (5.3.5*a*) *et bornée par* (5.3.13). *De plus,* δT_n *est de 2-norme minimale parmi toutes les solutions de* (5.3.1).

De cette proposition, nous pouvons déduire la *W*-stabilité de l'algorithme LD (conformément à la définition (2.5.4). Pour ce faire, partitionnons la matrice δT_n définie par (5.3.5*a*) sous la forme :

$$\delta T_n = \frac{1}{q_n} \begin{bmatrix} d_0 & d_0[\tilde{a}^{(n)}]^t \\ d^{(n)} & d^{(n)}[\tilde{a}^{(n)}]^t \end{bmatrix} = \begin{bmatrix} \dfrac{d_0}{q_n} & \dfrac{d_0}{q_n}[\tilde{a}^{(n)}]^t \\ \dfrac{d^{(n)}}{q_n} & \delta T_{n-1} \end{bmatrix} \quad \text{avec} \quad q_n = |\tilde{\bar{a}}^{(n)}|_2^2 .$$

Alors

$$T_n + \delta T_n = \begin{bmatrix} r_0 & [r^{(n)}]^t \\ r^{(n)} & T_{n-1} \end{bmatrix} + \frac{1}{q_n} \begin{bmatrix} d_0 & d_0[\tilde{a}^{(n)}]^t \\ d^{(n)} & d^{(n)}[\tilde{a}^{(n)}]^t \end{bmatrix}$$

$$= \begin{bmatrix} r_0 + \dfrac{d_0}{q_n} & \left[r^{(n)} + \dfrac{d_0}{q_n}\tilde{a}^{(n)}\right]^t \\ r^{(n)} + \dfrac{d^{(n)}}{q_n} & T_{n-1} + \delta T_{n-1} \end{bmatrix},$$

et on peut transformer le système (5.3.1) en un système linéaire

$$(T_{n-1} + \delta T_{n-1})\,\tilde{a}^{(n)} = -(r^{(n)} + \delta r^{(n)})\,, \quad (5.3.16)$$

où

$$\delta T_{n-1} = \frac{1}{\|\tilde{\bar{a}}^{(n)}\|_2^2} d^{(n)}[\tilde{a}^{(n)}]^t\,, \quad \delta r^{(n)} = \frac{1}{\|\tilde{\bar{a}}^{(n)}\|_2^2} d^{(n)}\,, \quad (5.3.17)$$

avec

$$d^{(n)} = -(r^{(n)} + T_{n-1}\,\tilde{a}^{(n)}) \quad \text{et} \quad d_0 = \tilde{\sigma}_n - \sum_{i=0}^{n} r_i\,\tilde{\tilde{a}}_i^{(n)}. \qquad (5.3.18)$$

Noter que la première ligne du système linéaire (5.3.1) :

$$\tilde{\sigma}_n = \sum_{i=0}^{n} \left(r_i + \frac{d_0}{q_n}\,\tilde{\tilde{a}}_i^{(n)} \right) \tilde{\tilde{a}}_i^{(n)}$$

équivaut à la définition (5.3.18) de d_0. De cette discussion, on déduit que :

COROLLAIRE 5.3.19 : *La solution calculée* $\{\tilde{a}^{(n)}\}$ *du système linéaire* $T_{n-1}\,a^{(n)} = -r^{(n)}$ *par l'algorithme LD est solution exacte du système linéaire perturbé* $(T_{n-1} + \delta T_{n-1})\,\tilde{a}^{(n)} = -(r^{(n)} + \delta r^{(n)})$, *où* δT_{n-1} *et* $\delta r^{(n)}$ *sont donnés par* (5.3.17) *et leur 1-norme bornée par* (5.3.13).

Remarque : Si on suppose que le second membre ne change pas ($\delta r^{(n)} = 0$), en partant du système perturbé

$$(T_{n-1} + \delta T_{n-1})\,\tilde{a}^{(n)} = -r^{(n)}, \qquad (5.3.20)$$

il est facile de voir que $\delta T_{n-1}\,\tilde{a}^{(n)} = d^{(n)}$ et d'en déduire que la perturbation δT_{n-1}

$$\delta T_{n-1} = \frac{1}{\|\tilde{a}^{(n)}\|_2^2}\, d^{(n)}[\tilde{a}^{(n)}]^t \qquad (5.3.21)$$

convient, puisque sa 1-norme $\|\delta T_n\|_1$ peut être bornée à l'aide de (5.3.12).

OBSERVATIONS

1) On peut remplacer, dans (5.3.13), k_m par $\tilde{k}_m$ en n'introduisant que des erreurs de l'ordre ε_M^2 (ε_M étant l'unité d'arrondissage). On a donc une majoration du type

$$\|\delta T_n\|_1 \leqslant \varepsilon_M\, p(n) + O(\varepsilon_M^2),$$

où $p(n)$ est un polynôme connu en n de degré 2. Ce qui prouve que $\|\delta T_n\|_1$ est petit, d'après la définition (2.5.9). Et on en déduit que *l'algorithme LD est W-stable pour toute matrice Toeplitz symétrique, même indéfinie.*

2) La relation (5.3.12) fournit un majorant du vecteur résidu $\overline{d}^{(n)}$ et généralise ainsi le résultat de Cybenko [9] pour les matrices Toeplitz symétriques définies positives au cas indéfini. Notre majorant est meilleur que celui

proposée par Chan et Hansen [28] qui comporte, en plus, le facteur $\max(|k_i|, i = 1 \text{ à } n)$.

5.4. B-stabilité de l'algorithme LD

Pour prouver la B-stabilité (stabilité forte selon Bunch) de l'algorithme LD pour résoudre les équations de Yule-Walker, il faut montrer qu'il existe une perturbation Toeplitz symétrique, δT_n

$$\delta T_n = \begin{pmatrix} \delta r_0 & \delta r_1 & \cdots & \delta r_n \\ \delta r_1 & \delta r_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \delta r_1 \\ \delta r_n & \cdots & \delta r_1 & \delta r_0 \end{pmatrix}, \tag{5.4.1}$$

de norme relativement petite, qui vérifie le système linéaire perturbé (5.3.1). Il en résulte que δT_n vérifiera l'équation matricielle (5.3.2). Pour déterminer δT_n, on pourrait avoir recours au produit de Kronecker (*cf.* Higham *et al.* [29]) pour transformer l'équation matricielle en un système linéaire. Mais cette approche ne permet pas de faire apparaître facilement la structure éventuelle de l'inconnue matricielle. Nous proposons ici une façon simple de déterminer cette matrice, et surtout de montrer qu'elle est aussi structurée (Toeplitz-plus-Hankel).

Décomposons δT_n en L et U comme suit : $\delta T_n = L + U$ avec

$$L = \begin{pmatrix} \frac{\delta r_0}{2} & 0 & & \\ \delta r_1 & \frac{\delta r_0}{2} & & 0 \\ \vdots & \ddots & \ddots & 0 \\ \delta r_n & \cdots & \delta r_1 & \frac{\delta r_0}{2} \end{pmatrix} \quad \text{et} \quad U = \begin{pmatrix} \frac{\delta r_0}{2} & \delta r_1 & \cdots & \delta r_n \\ 0 & \frac{\delta r_0}{2} & \ddots & \vdots \\ & & \ddots & \delta r_1 \\ 0 & & 0 & \frac{\delta r_0}{2} \end{pmatrix}. \tag{5.4.2}$$

Remarquons d'abord que L_i et U_i, respectivement les ième colonnes de L et de U, s'écrivent

$$L_i = \begin{bmatrix} 0 \\ \vdots \\ \delta r_0/2 \\ \delta r_1 \\ \vdots \\ \delta r_{n-i} \end{bmatrix} = Z^i \begin{bmatrix} \delta r_0/2 \\ \delta r_1 \\ \vdots \\ \delta r_{n-i} \\ \vdots \\ \delta r_n \end{bmatrix}, \quad U_i = \begin{bmatrix} \delta r_i \\ \vdots \\ \delta r_1 \\ \delta r_0/2 \\ \\ 0 \end{bmatrix} = (Z^t)^{n-i} \begin{bmatrix} \delta r_n \\ \vdots \\ \delta r_i \\ \vdots \\ \delta r_1 \\ \delta r_0/2 \end{bmatrix}, i = 0 \text{ à } n, \tag{5.4.3}$$

où Z est la matrice de décalage dont l'effet sur un vecteur consiste à la décaler d'un cran vers le bas. L'équation matricielle (5.3.2) s'écrit

$$\delta T_n \tilde{\bar{a}}^{(n)} = \sum_{i=0}^{n} \tilde{a}_i^{(n)} (L_i + U_i) = \sum_{i=0}^{n} \tilde{a}_i^{(n)} \{ Z^i + (Z^t)^{n-i} J \} \, \delta \bar{r}^{(n)} = \bar{d}^{(n)} , \qquad (5.4.4)$$

où $(\delta \bar{r}^{(n)})^t = (\delta r_0 /2, \delta r_1, ..., \delta r_n)^t$. En observant que la matrice

$$(Z^t)^{n-i} J = \begin{pmatrix} 0 & & 1 & 0 \\ & \iddots & & \\ 1 & & & \\ 0 & & & 0 \end{pmatrix}$$

contient des « 1 » sur la i-ième sur-antidiagonale et des « 0 » partout ailleurs, il vient, de (5.3.17), que

$$\left\{ \begin{pmatrix} 1 & 0 & & 0 \\ \tilde{a}_1^{(n)} & 1 & & \\ \vdots & \ddots & \ddots & \\ \tilde{a}_n^{(n)} & \cdots & \tilde{a}_1^{(n)} & 1 \end{pmatrix} + \begin{pmatrix} 1 & \tilde{a}_1^{(n)} & \cdots & \tilde{a}_1^{(n)} \\ \tilde{a}_1^{(n)} & \iddots & \iddots & \\ \vdots & \tilde{a}_n^{(n)} & & 0 \\ \tilde{a}_n^{(n)} & 0 & & 0 \end{pmatrix} \right\} \begin{bmatrix} \delta r_0 /2 \\ \delta r_1 \\ \vdots \\ \delta r_{n-i} \\ \vdots \\ \delta r_n \end{bmatrix} = \bar{d}^{(n)} \qquad (5.4.5)$$

$$\{ \quad A_1 \quad + \quad A_2 \quad \} \, \delta \hat{\bar{r}}^{(n)} = \bar{d}^{(n)} ,$$

où A_1 est la matrice Toeplitz triangulaire inférieure et A_2 la matrice Hankel triangulaire supérieure dont la première colonne le vecteur $(\tilde{\bar{a}}^{(n)}) = (1, \tilde{a}_1^{(n)}, ..., \tilde{a}_n^{(n)})^t$.

Il existe des algorithmes rapides (performants) pour résoudre les systèmes linéaires ayant la structure (Toeplitz-plus-Hankel). Pour les références, on peut consulter Zarowski [30].

De la discussion ci-dessus, on conclut que

PROPOSITION 5.4.6 : *La solution calculée* $\{\tilde{a}^{(n)}, \tilde{\sigma}_n\}$ du système linéaire $T_n \begin{bmatrix} 1 \\ a^{(n)} \end{bmatrix} = \sigma_n e_1$ *par l'algorithme LD satisfait le système linéaire* $(T_n + \delta T_n) \begin{bmatrix} 1 \\ \tilde{a}^{(n)} \end{bmatrix} = \tilde{\sigma}_n e_1$, *où la matrice* δT_n

$$\delta T_n = \begin{pmatrix} \delta r_0 & \delta r_1 & \cdots & \delta r_n \\ \delta r_1 & \delta r_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \delta r_1 \\ \delta r_n & \cdots & \delta r_1 & \delta r_0 \end{pmatrix}$$

est aussi Toeplitz symétrique avec $\delta \hat{r}^{(n)} = (\delta r_0/2, \delta r_1, \ldots, \delta r_n)^t$ *solution du système linéaire particulier* (5.4.5).

En partitionnant la matrice $T_n + \delta T_n$ comme suit

$$T_n + \delta T_n = \begin{bmatrix} r_0 + \delta r_0 & [r^{(n)} + \delta r^{(n)}]^t \\ r^{(n)} + \delta r^{(n)} & T_{n-1} + \delta T_{n-1} \end{bmatrix}, \tag{5.4.7}$$

on peut transformer le système $(T_n + \delta T_n) \begin{bmatrix} 1 \\ \tilde{a}^{(n)} \end{bmatrix} = \tilde{\sigma}_n e_1$ en un système linéaire

$$(T_{n-1} + \delta T_{n-1})\, \tilde{a}^{(n)} = -(r^{(n)} + \delta r^{(n)}) \tag{5.4.8}$$

et l'équation

$$\tilde{\sigma}_n = \sum_{i=0}^{n} (r_i + \delta r_i)\, \tilde{\tilde{a}}_i^{(n)} = (r_0 + \delta r_0) + (r^{(n)} + \delta r^{(n)})^t\, \tilde{a}^{(n)}, \tag{5.4.9}$$

où $(\delta r_0/2, \delta r_1, \ldots, \delta r_n)$ est solution du système linéaire (5.4.5) et δT_{n-1} définie par (5.4.1). D'où le résultat suivant :

COROLLAIRE 5.4.10 : *La solution calculée* $\tilde{a}^{(n)}$ *du système linéaire* $T_{n-1}\, a^{(n)} = -r^{(n)}$ *par l'algorithme LD est solution exacte du système linéaire perturbé* $(T_{n-1} + \delta T_{n-1})\, \tilde{a}^{(n)} = -(r^{(n)} + \delta r^{(n)})$, *où la matrice* δT_{n-1}

$$\delta T_{n-1} = \begin{pmatrix} \delta r_0 & \delta r_1 & \cdots & \delta r_{n-1} \\ \delta r_1 & \delta r_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \delta r_1 \\ \delta r_{n-1} & \cdots & \delta r_1 & \delta r_0 \end{pmatrix}$$

est aussi Toeplitz symétrique avec $(\delta r_0/2, \delta r_1, ..., \delta r_n)$ *solution du système linéaire* (5.4.5). *La solution* $(\delta r_0/2, \delta r_1, ..., \delta r_n)$ *vérifie en plus la contrainte* (5.4.9).

OBSERVATIONS :

1) Conformément à la définition (2.5.5), nous déduisons, du corollaire ci-dessus, que *l'algorithme LD sera B-stable pour la matrice* T_n *Toeplitz symétrique si la perturbation* $\delta\hat{\bar{r}}^{(n)} = (\delta r_0/2, \delta r_1, ..., \delta r_n)$ *(solution du système linéaire spécial* (5.4.5)) *est relativement petite en norme.* Noter en plus que

$$\frac{\|\bar{d}^{(n)}\|}{2\,\|\tilde{\bar{a}}^{(n)}\|} \leqslant \|\delta\hat{\bar{r}}^{(n)}\| \leqslant \|(A_1 + A_2)^{-1}\|\,\|\bar{d}^{(n)}\| . \qquad (5.4.11)$$

2) De la définition de d_0, la première ligne du système linéaire (5.4.5) équivaut à (5.4.9)

$$\delta r_0 + (\tilde{a}^{(n)})^t \delta r^{(n)} = d_0^{(n)} = \tilde{\sigma}_n - (1, \tilde{a}^{(n)})^t \begin{bmatrix} r_0 \\ r^{(n)} \end{bmatrix}$$

$$\Leftrightarrow \tilde{\sigma}_n = r_0 + \delta r_0 + (r^{(n)} + \delta r^{(n)})^t \tilde{a}^{(n)} =: \sigma_n^* . \qquad (5.4.12)$$

Ce qui signifie que l'erreur calculée $\tilde{\sigma}_n$ de prédiction linéaire serait l'erreur exacte σ_n^* de prédiction linéaire faite à partir des données perturbées $(\bar{r}^{(n)} + \delta\bar{r}^{(n)})$.

3) A partir du système linéaire perturbé

$$(T_{n-1} + \delta T_{n-1})\,\tilde{a}^{(n)} = -(r^{(n)} + \delta r^{(n)}) , \qquad (5.4.13)$$

où δT_{n-1} est définie par (5.4.1), on peut retrouver (voir en annexe B) le résultat du corollaire (5.4.10) en imposant à la solution $\delta\hat{\bar{r}}^{(n)} = (\delta r_1/2, \delta r_1, ..., \delta r_n)$ la contrainte (5.4.9). Si, au lieu d'imposer la contrainte (5.4.9) à la solution, on pose par contre $\delta r_0 = 0$, la solution $\delta r^{(n)} = (\delta r_1, ..., \delta r_n)$ convenable vérifiera le système linéaire (aussi structuré) :

$$\left\{ \begin{pmatrix} 1 & 0 & & 0 \\ a_1^{(n)} & 1 & & \\ \vdots & \ddots & \ddots & 0 \\ a_{n-1}^{(n)} & \cdots & a_1^{(n)} & 1 \end{pmatrix} + \begin{pmatrix} a_2^{(n)} & a_3^{(n)} & a_n^{(n)} & 0 \\ a_3^{(n)} & \iddots & & \\ a_n^{(n)} & & & \\ 0 & & & 0 \end{pmatrix} \right\} \begin{bmatrix} \delta r_1 \\ \vdots \\ \vdots \\ \delta r_n \end{bmatrix} = d^{(n)} \qquad (5.4.14)$$

$$\{ \quad B_1 \quad + \quad B_2 \quad \}\ \delta r^{(n)} = d^{(n)} ,$$

où B_1 (respectivement B_2) est la matrice Toeplitz triangulaire inférieure (respect. Hankel supérieure) dont la première colonne est $(1, \tilde{a}_1^{(n)}, ..., \tilde{a}_{n-1}^{(n)})^t$ (respect. $(a_2^{(n)}, ..., \tilde{a}_n^{(n)}, 0)^t$). Dans ce cas, le prédicteur calculé $\tilde{a}^{(n)}$ est le prédicteur exact fait à partir des données perturbées $(1, r_1 + \delta r_1, ..., r_n + \delta r_n)$, mais l'erreur calculée $\tilde{\sigma}_n$ de prédiction ne sera pas égale à l'erreur exacte σ_n^* de prédiction faite à partir de ces données perturbées. D'où le résultat suivant (voir la démonstration en annexe B).

COROLLAIRE 5.4.15 : *La solution calculée $\tilde{a}^{(n)}$ du système linéaire $T_{n-1}\, a^{(n)} = - r^{(n)}$ par l'algorithme LD est solution exacte du système linéaire perturbé $(T_{n-1} + \delta T_{n-1})\, \tilde{a}^{(n)} = - (r^{(n)} + \delta r^{(n)})$, où la matrice δT_{n-1}*

$$\delta T_{n-1} = \begin{pmatrix} 0 & \delta r_1 & \cdots & \delta r_{n-1} \\ \delta r_1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \delta r_1 \\ \delta r_{n-1} & \cdots & \delta r_1 & 0 \end{pmatrix}$$

est aussi Toeplitz symétrique avec $(\delta r_1, ..., \delta r_n)$ solution du système linéaire (5.4.14). *Mais la solution $(\delta r_1, ..., \delta r_n)$ ne vérifie pas nécessairement la contrainte* (5.4.9) :

$$\tilde{\sigma}_n = r_0 + (r^{(n)} + \delta r^{(n)})^t\, \tilde{a}^{(n)} := \sigma_n^* .$$

4) Si on suppose que le second membre ne varie pas (c'est-à-dire $\delta r^{(n)} = 0$ dans le système (5.4.13) et on part du système linéaire perturbé

$$(T_{n-1} + \delta T_{n-1})\, \tilde{a}^{(n)} = - r^{(n)} , \tag{5.4.16}$$

on peut montrer (*cf.* annexe B) que la perturbation δT_{n-1} convenable est Toeplitz symétrique définie par $(\delta r_0/2, \delta r_1, ..., \delta r_{n-1})$, solution du système linéaire

$$\left\{ \begin{pmatrix} a_1^{(n)} & 0 & & 0 \\ a_2^{(n)} & a_1^{(n)} & & \\ \vdots & \ddots & \ddots & 0 \\ a_{n-1}^{(n)} & \cdots & a_2^{(n)} & a_1^{(n)} \end{pmatrix} + \begin{pmatrix} a_1^{(n)} & a_2^{(n)} & \cdots & 0 \\ a_2^{(n)} & \iddots & a_n^{(n)} & \\ \vdots & \iddots & & \\ a_n^{(n)} & & & 0 \end{pmatrix} \right\} \begin{bmatrix} \delta r_0/2 \\ \delta r_1 \\ \vdots \\ \delta r_{n-1} \end{bmatrix} = d^{(n)} \tag{5.4.17}$$

$$\{ \quad C1 \quad + \quad C2 \quad \}\ \delta \hat{\tilde{r}}^{(n-1)} = d^{(n)} ,$$

qui est aussi structurée « Toeplitz-plus-Hankel ».

6. CONCLUSION

Dans un premier temps, nous avons proposé un majorant (basé sur la notion de « générateurs ») du conditionnement d'une matrice Toeplitz qui, outre le fait qu'il est plus fin que celui proposé par Cybenko, se généralise au cas indéfini.

A l'aide d'une analyse directe de la propagation des erreurs d'arrondi, nous avons ensuite établi un majorant de l'erreur $\| \Delta a^{(n)} \|_1$ (la précision) commise sur la solution calculée $\tilde{a}^{(n)}$ par l'algorithme de Levinson-Durbin. Cette borne d'erreur est relativement petite (par rapport à la norme de $\tilde{a}^{(n)}$) quand la matrice (même indéfinie) est *fortement bien conditionnée* (c'est-à-dire toutes les sous-matrices principales sont bien conditionnées). Nous en avons déduit que l'algorithme LD est faiblement stable pour toute matrice Toeplitz symétrique fortement bien conditionnée. En se rappelant que dans le cas défini positif, « fortement bien conditionnée » est équivalent à « bien conditionnée », nous avons conclu que le résultat de Cybenko [9] s'étend aux matrices indéfinies fortement bien conditionnées.

A l'aide d'une analyse inverse d'erreur, nous avons enfin déterminé une perturbation δT_n de 2-norme minimale qui vérifie

$$(T_n + \delta T_n) \begin{bmatrix} 1 \\ \tilde{a}^{(n)} \end{bmatrix} = \tilde{\sigma}_n e_1$$

et proposé un majorant de sa norme $\| \delta T_n \|_1$. Ce majorant reste toujours relativement petit par rapport à $\| T_n \|_1$, ce qui entraîne la stabilité inverse de l'algorithme LD pour toute matrice Toeplitz symétrique. Pour la stabilité forte (au sens de Bunch [6]) de l'algorithme LD, nous avons montré que la perturbation $\delta \hat{\tilde{r}}^{(n)} = (\delta r_0 /2, \delta r_1, ..., \delta r_n)$ convenable sur les données $r^{(n)}$ est la solution du système linéaire structuré dont la matrice est la somme d'une matrice Toeplitz et d'une matrice Hankel triangulaires.

RÉFÉRENCES

[1] S. T. ALEXANDER & ZONG M. RHEE, 1987, Analytical Finite Precision Results for Burg's Algorithm and the Autocorrelation Method for Linear Prediction, *IEEE Trans. on ASSP,* **35,** n° 5, pp. 626-634.

[2] R. R. BITMEAD & B. D. O. ANDERSON, 1980, Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations, *Linear Algebra & Its Applications,* **34,** pp. 103-116.

[3] A. BJÖRCK, 1991, « Errors Analysis of Least Squares Algorithms », *NATO ASI Series,* vol. F 70, Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, Edited by G. H. Golub and P. Van Dooren, Springer-Verlag Berlin Heidelberg, pp. 41-73.

[4] F. L. BAUER, 1974, « Computational Graphs and Rounding Error », *SIAM J. Numer. Anal.*, vol. 11, p. 87-96.

[5] J. R. BUNCH, 1985, « Stability of Methods for Solving Toeplitz Systems of Equations », *SIAM J. Sci. Stat. Comput.*, vol. 6, n° 2, pp. 349-364.

[6] J. R. BUNCH, 1987, « The Weak and Strong Stability of Algorithms in Numerical Linear Algebra », *Linear Algebra & Its Applications,* 88/89, pp. 49-66.

[7] J. R. BUNCH, 1991, « The weak Stability of Algorithms of Matrix Computations », *NATO ASI Series,* vol. F 70, Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, Edited by G. H. Golub and P. Van Dooren, Springer-Verlag Berlin Heidelberg, pp. 429-433.

[8a] J. R. BUNCH, W. DEMMEL & C. F. Van Loan, 1989, « The Strong Stability of Algorithms for solving Symmetric Linear Systems », *SIAM J. Matr. Anal. Appl.*, vol. 10, n° 4, pp. 494-499.

[8b] F. CHATELIN, V. FRAYSSÉ & T. BRACONNIER. 1993, « Qualitative Computing : elements of a theory for finite precision computation », Tech. report CERFACS TR/PA/93/12. *Lecture Notes for the Workshop on Reliability of Computations,* March 30-April 1, Toulouse.

[9] G. CYBENKO, 1980, « The Numerical Stability of the Levinson-Durbin Algorithm for Toeplitz Systems of Equations », *SIAM J. Sci. Stat. Comput.*, vol. 1, n° 3, pp. 303-319.

[10] P. FRANÇOIS, 1989, *Contribution à l'Etude de Méthodes de Contrôle Automatique de l'Erreur d'arrondi,* la Méthodologie SCALP ; Thèse de Doctorat de l'INPG, Grenoble.

[11] G. H. GOLUB & C. F. VAN LOAN, 1983, *Matrix Computations,* John Hopkins University Press.

[12] C. GUEGUEN, 1987, « An Introduction to Displacement Ranks and Related Fast Algorithms », *Signal Processing,* vol. XLV, Lacoume Durrani Stora Editors, Elsevier, pp. 705-780.

[13] F. G. GUSTAVSON & D. Y. YUN, 1989, « Fast Algorithm of Rational Hermite Approximation and Solution of Toeplitz Systems », *IEEE Trans. on Circuits and Systems,* vol. CAS-26, n° 9, pp. 750-755.

[14] P. HENRICI, 1982, *Essentials of Numerical Analysis,* Wiley.

[15] F. de HOOG, « A New Algorithm for Solving Toeplitz Systems of Equations », *Linear Algebra & Its Applications,* 88/89, pp. 123-138.

[16] K. JAINANDUNSING & E. F. DEPRETTERE, « A New Class of Parallel Algorithms for Solving Systems of Linear Equations », *SIAM J. Sci. Stat. Comput.*, vol. 10, n° 5, pp. 880-912.

[17] T. KAILATH, A. VIEIRA & M. MORF, 1978, « Inverse of Toeplitz Operators Innovations and Orthogonal Polynomials », *SIAM Review,* vol. 20, n° 1, pp. 106-119.

[18] J. MAKHOUL, 1975, « Linear Prediction : A Tutorial Review », *Proceeding of IEEE,* vol. 63, n° 4, pp. 561-580.

[19] R. E. MOORE, 1966, *Interval Analysis,* Prentice-Hall, Englewood cliffs, NJ.

[20] P. H. STERBENZ, 1974, *Floating Point Computation,* Prentice-Hall, Englewood cliffs, NJ.

[21] G. W. STEWART, 1973, *Introduction to Matrix Computations,* Academic Press.

[22] F. STUMMEL, « Perturbation Theory for Evaluation Algorithms of Arithmetic Expressions », *Math. Comput.,* vol. 37, n° 156, pp. 435-473.

[23] W. F. TRENCH, 1964, « An Algorithm For the Inverson Finite Toeplitz Matrices », *SIAM J. Applied Math.,* vol. 12, pp. 512-522.

[24] J. H. WILKINSON, 1963, *Rounding Errors in Algebraic Processes,* Prentice-Hall, Englewood Cliffs, NJ.

[25] J. H. WILKINSON, 1965, *The Algebraic Eigenvalue Problem,* Oxford University Press, London.

[26] S. ZOHAR, 1969, « Toeplitz Matrix Inversion : The Algorithm of W. F. Trench », *Journal of the ACM,* vol. 16, n° 4, pp. 592-601.

[27] S. ZOHAR, 1974, « The Solution of a Toeplitz set of Linear Equations », *Journal of ACM,* vol. 21, n° 1, pp. 272-276.

[28] T. F. CHAN & P. C. HANSEN, 1992, « A Look-ahead Levinson Algorithm for Indefinite Toeplitz Systems », *SIAM J. Matrix Anal. Appl.,* vol. 13, n° 2, pp. 490-506.

[29] D. J. HIGHAM & N. J. HIGHAM, 1992, « Backward Error and Condition of Structured Linear Systems », *SIAM J. Matrix Anal. Appl.,* vol. 13, n° 1, pp. 162-175.

[30] C. J. ZAROWSKI, 1992, « A Schur Algorithm and Linearly Connected Processor Array for Toeplitz-plus-Hankel Matrices », *IEEE Trans. on Signal Processing,* vol. 40, n° 8, pp. 2065-2078.

Annexe A

Démonstration du lemme (5.2.10).

Notons d'abord que, de (5.2.2), on déduit

$$\Delta k_t \, \sigma_{t-1} = -\left\{ (Jr^{(t-1)})^t \, \Delta a^{(t-1)} + k_t \, \Delta \sigma_{t-1} + \phi_t \right\} \tag{A1}$$

avec $\phi_t = \omega_t - \sigma_{t-1} \, \eta_t$ et (5.2.3) peut s'écrire

$$\Delta \sigma_t = (1 - k_t^2) \, \Delta \sigma_{t-1} - k_t \, \sigma_{t-1} \, \Delta k_t - k_t \, \sigma_{t-1} \, \Delta k_t + \gamma_t \, . \tag{A2}$$

En substituant $-\sigma_{t-1} \, \Delta k_t$ par (A1) dans un des termes $-k_t \, \sigma_{t-1} \, \Delta k_t$ de (A2), et $-k_t \, \sigma_{t-1}$ par sà valeur

$$-k_t \, \sigma_{t-1} = r_t + (r^{(t-1)})^t \, J a^{(t-1)} \tag{A3}$$

dans l'autre terme, on arrive à

$$\begin{aligned} \Delta \sigma_t = (1 - k_t^2) \, \Delta \sigma_{t-1} + k_t \{ & (Jr^{(t-1)})^t \, \Delta a^{(t-1)} \\ & + k_t \, \Delta \sigma_{t-1} + \phi_t \} - k_t \, \sigma_{t-1} \, \Delta k_t + \gamma_t \end{aligned} \tag{A4}$$

$$= \Delta \sigma_{t-1} + k_t \{ (Jr^{(t-1)})^t \, \Delta a^{(t-1)} \} - k_t \, \sigma_{t-1} \, \Delta k_t + k_t \, \phi_t + \gamma_t$$

$$= \Delta \sigma_{t-1} + (r^{t-1})^t \{ k_t \, J \, \Delta a^{(t-1)} - \Delta k_t \, J \, \Delta a^{(t-1)} \} + \Delta k_t \, r_t + k_t \, \phi_t + \gamma_t \, .$$

Puisque

$$\{ k_t \, J \, \Delta a^{(t-1)} - \Delta k_t \, J \, \Delta a^{(t-1)} \} = \Delta \underline{a}^{(t)} - \Delta a^{(t-1)} - \alpha^{(t)} \tag{A5}$$

avec $\underline{a}^{(t)} = (a_1^{(t)}, \ldots, a_{t-1}^{(t)})$, nous avons

$$\Delta \sigma_t = \Delta \sigma_{t-1} + (r^{(t-1)})^t \{ \Delta \underline{a}^{(t)} - \Delta a^{(t-1)} - \alpha^{(t)} \} + \Delta k_t \, r_t + k_t \, \phi_t + \gamma_t \tag{A6}$$

et

$$\begin{aligned} \Delta \sigma_t - (r^{(t-1)})^t \, \Delta \underline{a}^{(t)} - \Delta k_t \, r_t = \Delta \sigma_{t-1} - (r^{(t-1)})^t \, \Delta a^{(t-1)} \\ - (r^{(t-1)})^t \, \alpha^{(t)} + k_t \, \phi_t + \gamma_t \, . \end{aligned} \tag{A7}$$

En notant que $\Delta k_t = \Delta a_t^{(t)}$ et utilisant le lemme (5.2.11) avec

$$x_t = \Delta \sigma_t - (r^{(t)})^t \, \Delta a^{(t)} \, , \quad \alpha_t = 1 \quad \text{et} \quad \beta_t = -(r^{(t-1)})^t \, \alpha^{(t)} + k_t \, \phi_t + \gamma_t \, , \tag{A8}$$

on déduit que

$$\Delta\sigma_i - (r^{(i)})^t \Delta a^{(i)} = \sum_{j=1}^{i} \beta_j = P_i \quad \text{avec} \quad P_i = P_{i-1} + \beta_i . \tag{A9}$$

D'où le lemme (5.2.10).

Annexe B

Démonstration de (5.4.10) à (5.4.17).
A partir du système linéaire perturbé (5.4.13).

$$(T_{n-1} + \delta T_{n-1})\, \tilde{a}^{(n)} = -(r^{(n)} + \delta r^{(n)}) , \tag{B1}$$

avec

$$\delta T_{n-1} = \begin{pmatrix} \delta r_0 & \delta r_1 & \cdots & \delta r_{n-1} \\ \delta r_1 & \delta r_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \delta r_1 \\ \delta r_{n-1} & \cdots & \delta r_1 & \delta r_0 \end{pmatrix}$$

et $\delta r^{(n)} = (\delta r_1, ..., \delta r_n)^t$, on déduit que

$$\delta T_{n-1}\, \tilde{a}^{(n)} + \delta r^{(n)} = -(r^{(n)} + T_{n-1}\, \tilde{a}^{(n)}) =: d^{(n)} . \tag{B2}$$

Avec la décomposition (5.4.2) de δT_{n-1} en L et U, on peut écrire $\delta T_{n-1}\, \tilde{a}^{(n)}$ comme

$$\delta T_{n-1}\, \tilde{a}^{(n)} = \sum_{i=0}^{n-1} a_{i+1}^{(n)} \{L_i + U_i\} = \sum_{i=0}^{n-1} a_{i+1}^{(n)} \{Z^i + (Z^t)^{n-1-i} J\} \begin{bmatrix} \delta r_0/2 \\ \delta r_1 \\ \vdots \\ \delta r_{n-1} \end{bmatrix} , \tag{B3}$$

et (B2) devient

$$\sum_{i=0}^{n-1} a_{i+1}^{(n)} \{Z^i + (Z^t)^{n-1-i} J\} \begin{bmatrix} \delta r_0/2 \\ \delta r_1 \\ \vdots \\ \delta r_{n-1} \end{bmatrix} + \begin{bmatrix} \delta r_1 \\ \vdots \\ \delta r_{n-1} \\ \delta r_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \\ d_n \end{bmatrix} . \tag{B4}$$

Le système linéaire (B4) en $(\delta r_0/2, \delta r_1, ..., \delta r_n)$ comporte $(n+1)$ inconnues et n équations, et est donc indéterminé.

• Si on impose à la solution $(\delta r_0/2, \delta r_1, ..., \delta r_n)$ la condition supplémentaire (5.4.9) :

$$\tilde{\sigma}_n = \sum_{t=0}^{n} (r_t + \delta r_t)\, \tilde{a}_t^{(n)} =: \sigma_n^* \tag{B5}$$

l'erreur calculée $\tilde{\sigma}_n$ de prédiction linéaire est égale à l'erreur exacte σ_n^* de prédiction linéaire faite à partir des données perturbées $(\overline{r}^{(n)} + \delta\overline{r}^{(n)})$, ce qui équivaut à

$$d_0 = \tilde{\sigma}_n - \sum_{t=0}^{n} r_t\, \tilde{a}_t^{(n)} = \sum_{t=0}^{n} \delta r_t\, \tilde{a}_t^{(n)}, \tag{B6}$$

il est facile de remarquer que (B4) et (B6) donnent le système linéaire (5.4.5).

• Par contre, si on suppose que $\delta r_0 = 0$, le système (B4) se transforme en

$$\left[\sum_{t=0}^{n-1} a_{t+1}^{(n)} \{Z^{t+1} + (Z^t)^{n-t} J\} + I_n\right] \begin{bmatrix} \delta r_1 \\ \vdots \\ \delta r_{n-1} \\ \delta r_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \\ d_n \end{bmatrix}, \tag{B7}$$

en notant que $Z(\delta r_1, ..., \delta r_{n-1}, *)^t = (0, \delta r_1, ..., \delta r_{n-1})^t$, où * est un élément arbitraire, et en particulier δr_n. Et plus explicitement, on a

$$\left\{\begin{pmatrix} 1 & 0 & & 0 \\ a_1^{(n)} & 1 & & \\ \vdots & \ddots & \ddots & 0 \\ a_{n-1}^{(n)} & \cdots & a_1^{(n)} & 1 \end{pmatrix} + \begin{pmatrix} a_2^{(n)} & a_3^{(n)} & a_n^{(n)} & 0 \\ a_3^{(n)} & \ddots & & \\ a_n^{(n)} & & & \\ 0 & & & 0 \end{pmatrix}\right\} \begin{bmatrix} \delta r_1 \\ \vdots \\ \vdots \\ \delta r_n \end{bmatrix} = d^{(n)}. \tag{B8}$$

Notons cependant que l'erreur calculée $\tilde{\sigma}_n$ de prédiction ne sera pas en général l'erreur exacte σ_n^* de prédiction faite à partir des données perturbées. D'où le corollaire 5.4.15.

• Si on suppose que le second membre r^n du système $T_{n-1}\, a^{(n)} = -\, r^{(n)}$ ne change pas et que, seule, la matrice T_{n-1} varie, on part alors du système linéaire perturbé (5.4.16)

$$(T_{n-1} + \delta T_{n-1})\, \tilde{a}^{(n)} = -\, r^{(n)}.$$

Ceci équivaut à poser $\delta r^{(n)} = 0$ dans (B1), et le système (B4) devient

$$\sum_{i=0}^{n-1} a_{i+1}^{(n)} \{Z^i + (Z^t)^{n-1-i} J\} \begin{bmatrix} \delta r_0/2 \\ \delta r_1 \\ \vdots \\ \delta r_{n-1} \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{B10}$$

qui, en l'explicitant, donne le système (5.4.17).