STABILITE AU SENS LARGE DES SYSTEMES LINEAIRES A TEMPS DISCRET

M. Benidir et B. Picinbono

Laboratoire des Signaux et Systèmes-ESE
Plateau du Moulon 91190 Gif-sur-Yvette

RESUME. Définissant la stabilité au sens large d'un système linéaire à temps discret par le fait que sa fonction de transfert peut éventuellement admettre des pôles sur le cercle unité, on établit directement à partir de l'algorithme de Routh et sans passer par les développements en fraction continue un algorithme pour tester une telle stabilité.

De plus les algorithmes existants et qui permettent de tester la stabilité au sens strict (pôles à l'intérir du cercle unité) sont rappelés brièvement et mis sous une même forme polynômiale.

Tous ces algorithmes sont alors comparés et on montre qu'il n'y a en fait que trois algorithmes différents fondés sur une même procédure, qui consiste à associer à un polynôme de degré n une suite de n coefficients.

SUMMARY. Defining the wide sense stability for a discrete time linear system by the fact that its transfert function can have some poles on the unit circle, an algorithm for testing this stability directly can be established from Routh's algorithm and without using the continued fraction expansion.

The existing algorithms for testing the stability in the strict sense (poles inside the unit circle) are recalled briefly and presented in the same polynomial form.

All these algorithms are then compared and it is shown that there are only three different algorithms based on a same procedure, which consists of associating a sequence of n coefficients to a polynomial of degree n.

1. Introduction

La stabilité au sens strict des systèmes linéaires à temps discret (**SLTD**), définie par le fait que les fonctions de transfert de ces derniers ne doivent admettre que des pôles à l'intérieur du cercle unité (CU), a fait l'objet de nombreux travaux durant les dernières décennies. D'autre part une notion de stabilité au sens large, définie par le fait que la fonction de transfert peut éventuellement admettre des pôles sur le CU, a été introduite récemment [1][2]. Deux critères permettant de tester une telle stabilité ont alors été établis : un premier [1] pour les systèmes à coefficients réels et un deuxième [2] pour ceux à coefficients complexes. Cependant, le critère proposé dans [1] est fondé sur des concepts liés aux développements en fraction continue (**DFC**) et l'algorithme donné dans [3], qui permet de mettre en pratique ce critère, fait intervenir des étapes de calcul non nécéssaires. Par ailleurs il repose lui aussi sur des propriétés liées au **DFC** établies dans [4][5]. De plus, d'autres algorithmes sont proposés dans [6] et [7] pour tester la stabilité au sens strict des **SLTD** réels. En raison de cette vaste littérature, un peu confuse et parfois incomplète comme indiqué dans [8], on se propose de montrer qu'il n'y a en fait que trois algorithmes différents et de faire ensuite une étude comparée de ces derniers. Dans la deuxième section on rappelle quelques résultats établis dans [9] et concernant la stabilité des systèmes linéaires à temps continu (**SLTC**). On en déduit dans la troisième section que la plupart des résultats établis dans [1][3][5] peuvent être obtenus sans les **DFC** et conduisent à un algorithme récursif pour tester la stabilité et dénommé ci-dessous **A1**. Dans la quatrième section on rappelle l'algorithme, dit de Levinson, mais adapté dans [2] à la stabilité au sens large et dénommé **A2**. On montre que dans le cas réel **A2** peut prendre une autre forme dénommée **A2R** qui est équivalente à l'algorithme **A3** donné dans [7]. Il reste en fait trois types d'algorithmes **A1**, **A2** et **A2R** qui sont comparés entre eux.

2. Stabilité des SLTC

On rappelle dans cette section l'algorithme de Routh qui permet de tester si un polynôme donné admet ou non des racines à partie réelle positive. L'algorithme est mis sous forme polynômiale et on distingue le cas des polynômes de Hurvitz (**H**), polynômes n'ayant que des racines à partie réelle négative, et celui des polynômes de Hurwitz généralisés (**HG**), polynômes n'ayant aucune racine à partie réelle positive. Tout polynôme g, de degré n, peut s'écrire d'une façon unique sous la forme

$$g = f_n \oplus f_m \ , \qquad (2\text{-}1)$$

où f_n et f_m représentent ses parties paire et impaire selon la parité de n. Pour tester la propriété de Hurwitz on construit à partir de g une suite de polynômes de degrés décroissants

$$S(g) = \{f_n, f_m, .., f_o\}, \qquad (2\text{-}2)$$

à l'aide de l'algorithme suivant
1) Conditions initiales : f_n et f_m données par (2-1) ;
2) pour j=n,n-1,..,

$$r_{j-2} = -\alpha_j s f_{j-1}(s) + f_j(s), \qquad (2\text{-}3)$$
$$f_{j-2} = r_{j-2} \text{ si } d^o(r_{j-2}) = j-2, \qquad (2\text{-}4)$$
$$f_{j-2} = f_{j-1}' \text{ si } d^o r_{j-2} = 0. \qquad (2\text{-}5)$$

où $d^o(f)$ et f' représentent le degré et la dérivée de f. Lorsque f_{j-2} est donné par (2-4) il vérifie la relation

$$f_{j-2}(s) = -\alpha_j s f_{j-1}(s) + f_j(s). \qquad (2\text{-}6)$$

De plus chaque polynôme f_j de la suite S(g) vérifie

$$f_j(-s) = (-1)^j f_j(s), \qquad (2\text{-}7)$$

et est appelé polynôme pair.impair.

Propriété 1 Le polynôme g de degré n, donné par (2-1), est un polynôme **HG ssi** m=n-1 et l'algorithme (2-3,4,5) permet de déterminer le vecteur $\underline{\alpha}_n$ ayant n composantes α_j positives.

3. Stabilité des SLTD

Soit $\mathbf{B}_n$ la transformation qui associe à tout polynôme g, de degré n, le polynôme G donné par :

$$G(z) = (z-1)^n g\{(z+1)/(z-1)\}. \qquad (3\text{-}1)$$

On peut vérifier facilement que la relation (3-1) définit une application $\mathbf{B}_n$ inversible et que le polynôme $g = \mathbf{B}_n^{-1}\{G\}$ est de degré n **ssi** $G(1) \neq 0$. On considère alors dans toute la suite des polynômes G tels que $G(1) \neq 0$ et on appelle g le polynôme associé à G. Soit $F_j = \mathbf{B}_j(f_j)$ l'image du polynôme f_j, élément de S(g) et $F_{D,j-1} = \mathbf{B}_{j-1}(f_j')$. Tenant compte de la définition (3-1), le polynôme $F_{D,j}$ peut s'exprimer directement en fonction du polynôme F_{j+1} et de sa dérivée F_{j+1}' à l'aide de la relation suivante

$$2F_{D,j} = (j+1)F_{j+1} + (1-z)F_{j+1}'. \qquad (3\text{-}2)$$

D'autre part, exprimant la relation de récurrence (2-6) en fonction des F_j, on obtient

$$F_{j-2} = \{-\alpha_j(z+1)F_{j-1} + F_j\}/(z-1)^2. \qquad (3\text{-}3)$$

Comme les polynômes f_j satisfont (2-6), les polynômes F_j vérifient la relation

$$F_j = F_j^* \ , \qquad (3\text{-}4)$$

où

$$F_j^*(z) = z^j F_j(z^{-1}). \qquad (3\text{-}5)$$

Tout polynôme qui vérifie (3-5) est appelé dans la suite polynôme autoréciproque.

Ainsi, partant d'un polynôme donné G, de degré n, on peut lui associer la suite de polynômes de degrés décroissants $\{F_n, F_m, .., F_o\}$, calculés à l'aide des relations (3-2,3) et à partir des polynômes F_n et F_m. Il reste donc à déterminer ces deux derniers polynômes à partir de G. Or tenant compte de la définition (3-1), on obtient à partir de (2-1)

$$G = F_n + (z-1)^{n-m} F_m. \qquad (3\text{-}6)$$

Mais comme F_n et F_m vérifient (3-4), on a aussi

$$G^* = F_n - (z-1)^{n-m} F_m. \qquad (3\text{-}7)$$

La résolution du système (3-6,7) donne alors

$$F_n = (G + G^*)/2, \qquad (3\text{-}8)$$
$$F_m = (G - G^*)/\{2(z-1)^{n-m}\}. \qquad (3\text{-}9)$$

Finalement on a établi l'algorithme suivant, dénommé **A1** :
1) Conditions initiales : F_n et F_m donnés par (3-8,9) ;
2) pour j=n,n-1,...

$$\alpha_j = F_j(1)/2F_{j-1}(1), \qquad (3\text{-}10)$$
$$R_{j-2} = \{-\alpha_j(z+1)F_{j-1} + F_j\}/(z-1)^2, \qquad (3\text{-}11)$$
$$F_{j-2} = R_{j-2}, \text{ si } d^o(R_{j-2}) = j-2 \ , \qquad (3\text{-}12)$$
$$F_{j-2} = F_{D,j-2}, \text{ si } R_{j-2} = 0. \qquad (3\text{-}13)$$

L'étape (j-2) peut être soit (3-12) soit (3-13) et, par convention, elle est dite régulière dans le premier cas et singulière dans le second.

Propriété 2 (critère de stabilité au sens large) Soit G un polynôme de degré n vérifiant $G(1)\neq 0$. Alors G n'admet aucune racine à l'extérieur du cercle unité **ssi** m=n-1 et l'algorithme **A1** permet de déterminer un vecteur $\underline{\alpha}_n$ ayant n composantes α_j positives.

Démonstration : Conséquence directe de la propriété 1 et des propriétés de la transformation homographique introduite par B_n.

Remarques : Soit G un polynôme stable au sens large et $\underline{\alpha}_n$ le vecteur correspondant déterminé par **A1**. Alors dans la détermination de $\underline{\alpha}_n$ trois cas sont possibles.

i) On ne passe aucune fois par l'étape singulière (3-13) : dans ce cas G n'admet aucune racine sur le CU et le **SLTD** correspondant est asymptotiquement stable (stabilité stricte).

ii) On passe une seule fois par l'étape singulière (3-13) : dans ce cas, si le polynôme qui a été dérivé est de degré m, G admet n-m racines à l'intérieur de CU et m racines simples sur le CU ; le **SLTD** correspondant est alors stable mais asymptotiquement instable.

iii) On passe plus d'une fois par l'étape singulière (3-13) : dans ce cas, si le premier polynôme qui a été dérivé est de degré m, G admet n-m racines à l'intérieur de CU et m racines sur le CU non toutes distinctes ; le **SLTD** correspondant est dit stable au sens large bien qu'il soit instable pour certains auteurs.

Un polynôme pour lequel on ne peut pas déterminer le vecteur $\underline{\alpha}_n$ ou pour lequel on rencontre un $\alpha_j<0$ admet forcément des racines à l'extérieur du CU ; la stabilité au sens large n'est donc pas vérifiée et le **SLTD** correspondant est dit complètement instable.

4. Comparaison entre les différents algorithmes

Dans cette section on va comparer l'algorithme **A1**, donné dans la section précédente, aux algorithmes **A2** [2] et **A3** [7] qui permettent de tester la stabilité des **SLTD** à coefficients respectivement réels ou complexes. Pour cela on va tout d'abord rappeler brièvement, et sous forme polynômiale, les algorithmes **A2** et **A3**. On montre ensuite que **A2** peut prendre dans le cas réel une forme, dénommée **A2R**, équivalente à **A3** et on compare enfin les algorithmes **A1, A2, A2R**. Dans la suite $n_i(G)$ désigne le nombre de racines de G à l'intérieur du CU.

Algorithme A2 [2] : Cet algorithme permet de calculer un vecteur $\underline{k}_n$, associé au polynôme G, à l'aide des relations, h = n, n-1,..1,

$$k_h = - P_h(0)/P_h^*(0), \qquad (4\text{-}1)$$

$$zP_{h-1} = P_h(z) + k_h P_h^* \qquad (4\text{-}2)$$

et à partir des conditions initiales $P_n=G_n$ et $P_n^*=G_n^*$ où le polynôme Q^* assosié à un plynôme Q est défini par (3-5). On a alors

$$n_i(G)=n \quad \textbf{ssi} \quad k_h^2<1,\ 1\leq h\leq n. \qquad (4\text{-}3)$$

Algorithme A3 [7] : L'algorithme **A3** permet de calculer un vecteur $\underline{\delta}_n$, associe au polynôme G, à l'aide des relations, h = n, n-1,..1,

$$\delta_h = T_h(0)/T_{h-1}(0), \qquad (4\text{-}4)$$

$$zT_{h-2} = \delta_h(z+1)T_{h-1} - T_h \qquad (4\text{-}5)$$

et à partir des conditions initiales $T_n=F_n$ et $T_{n-1}=F_{n-1}$. On a alors

$$n_i(G)=n \quad \textbf{ssi} \quad T_h(1)/T_{h-1}(1)>0,\ 1\leq h\leq n. \qquad (4\text{-}6)$$

Algorithme A2R (A2 : cas réel) : Dans le cas des polynômes à coefficients réels la relation (4-2) conduit, après calculs, à la récurrence suivante :

$$\mu_{h+1}Q_{h+1} = (1+z)Q_h - zQ_{h-1} \qquad (4\text{-}7)$$

où

$$Q_h = P_h^* + P_h \qquad (4\text{-}8)$$

et

$$\mu_{h+1} = (1+k_{h+1})(1-k_h). \qquad (4\text{-}9)$$

On obtient, après calculs, la version suivante de **A2**, adapté au cas réel, dénotée **A2R** : h = n, n-1,.., 1,

$$\mu_h = Q_{h-1}(0)/Q_h(0), \qquad (4\text{-}10)$$

$$zQ_{h-2} = (1+z)Q_{h-1} - \mu_h Q_h. \qquad (4\text{-}11)$$

Cet algorithme permet de calculer les coefficients μ_h à partir des conditions initiales $Q_n=2T_n=2F_n$ et Q_{n-1} donné par

$$zQ_{n-1}= zP_n^*+P_n + k_n(zP_n+P_n^*). \quad (4\text{-}12)$$

D'où

$$Q_{n-1} = G^*+k_nG + (G+k_nG^*)/z. \quad (4\text{-}13)$$

La deuxième condition initiale Q_{n-1} n'est donc plus donnée par (3-9) comme pour les algorithmes **A1** et **A3**. Par ailleurs, d'après la définition (4-8), les polynômes Q_h vérifient (3-5) et sont donc eux aussi autoréciproques, comme les polynômes F_h et T_h intervenant respectivement dans **A1** et **A3**.

Equivalence entre A2R et A3 : On va montrer que les algorithmes **A2R** et **A3** sont définis par des relations récursives identiques et que la seule différence entre **A2R** et **A3** réside dans les conditions initiales. Pour cela, associons à la suite de polynômes Q_h, $1\le h\le n$, la suite de polynômes

$$R_h = d_h^{-1}Q_h, \quad (4\text{-}14)$$

où les d_h sont définis par $d_n=d_{n-1}=1$ et

$$d_{h-2} = \mu_h d_h, \ 1 \le h \le n, \quad (4\text{-}15)$$

et posons

$$\sigma_h = d_{h-1}/d_{h-2}, \ 1 \le h \le n. \quad (4\text{-}16)$$

On obtient alors de (4-15,16)

$$\mu_h = 1/\sigma_h\sigma_{h+1}, \ 1 \le h \le n, \quad (4\text{-}17)$$

et de (4-1)

$$zR_{h-2} = \sigma_h(1+z)R_{h-1} - R_h \quad (4\text{-}18)$$

$$\sigma_h = R_h(0)/R_{h-1}(0). \quad (4\text{-}19)$$

Comme $\sigma_{n+1}=1$, on a d'aprés (4-16), $\mu_h>0$, $1\le h\le n$, **ssi** $\sigma_h>0$, $1\le h\le n$.

Ainsi l'algorithme **A2R** peut être remplacé par (4-18,19) et ces relations sont identiques aux relations (4-5,4) définissant l'algorithme **A3**. La seule différence entre **A2R** et **A3** réside donc dans les conditions initiales : dans **A2R** on calcule le vecteur $\underline{\mu}_n$ à partir des conditions initiales (3-8), (4-12) et dans **A3** on calcule le vecteur $\underline{\delta}_n$ à partir des conditions initiales (3-8) et (3-9). Le test de stabilité dans **A2R** ne porte pas sur le signe des μ_h mais il reste donné par (4-3) où les k_h se déduisent simplement des $Q_h(1)$.

Sous forme de récurrences polynômiales les trois algorithmes : **A1** défini par (3-10,11), **A2** par (4-1,2) et **A2R** (**A3**) par (4-4,5), peuvent être comparés facilement. Les algorithmes **A1** et **A2R** sont des récurrences entre trois polynômes réels et autoréciproques, c'est-à-dire vérifiant chacun la relation (3-4). Ces deux algorithmes ne permettent de tester que des polynômes à coefficients réels et du point de vue nombre d'opérations **A2** est plus intéressant que **A1**. Par contre l'algorithme **A2** est une récurence entre deux polynômes seulement et il permet de tester les polynômes à coefficients réels ou complexes.

Références

[1] Y.S. Lai, "A Generalized Test for Discrete System Stability", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, pp. 1242-1243, Dec. 1984.

[2] M. Benidir, B. Picinbono, "Extensions for the stability criterion for ARMA Filters", à paraître dans IEEE, Trans. ASSP. Avril 1987.

[3] P. Steffen, "An algorithm for testing stability of discrete systems," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-25, pp. 454-456, oct. 1977.

[4] H.W. Schussler, "A stability theorem for discrete systems", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-24, pp. 87-89, Feb. 1976.

[5] J. Szczupac, S. K. Mitra, and E. I. Jury, "Some new results on discrete system stability", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-25, pp. 101-102, Feb. 1977.

[6] M. Ismail, " A simplified algorithm for testing stability of discrete systems via bilinear continued fractions ", IEEE Trans. Circuits Syst., Vol. CAS-33, pp. 544-547, May 1986.

[7] Y. Bistritz, " Zero Location with Respect to the Unit Circle of Discrete-Time Linear System Polynomials," Proceedings of the IEEE, vol. 72, no.9, Sep. 1984.

[8] R. Gnanasekaran, " A note on the niew 1-D and 2-D stability theorems for discrete systems," IEEE Trans., Acoust., Speech, Signal Processing, vol. ASSP-29, pp. 1211-1212, Dec. 1976.

[9] M. Benidir, B. Picinbono, " Extension du critère de stabilité de Routh ", Rapport interne au L2S.