HUITIEME COLLOQUE SUR LE TRAITEMENT DU
SIGNAL ET SES APPLICATIONS

NICE du 1er au 5 JUIN 1981

OPERATEUR DE RICCATI RAPIDE
IMPLEMENTATION D'ALGORITMES DE CALCUL
DU GAIN DE KALMAN OPTIMAL

- Alain GIULIERI -

Laboratoire L.A.I.A.T.
Université de TOULON - 83130 - LA GARDE

RESUME

On complète sur le plan algorithmique la synthèse effectuée dans l'article publié au précédent colloque du GRETSI sur la comparaison de différentes méthodes de résolution numérique de l'équation de RICCATI discrète en y ajoutant quelques critiques et en présentant les nouveaux résultats obtenus en 1979 et 1980 (ces algorithmes ont été simulés sur ordinateur CII-HB 66/10).

De façon à préparer la synthèse de l'opérateur de RICCATI Rapide, certains de ces algorithmes ont été implémentés sur le micro-processeur monolithique 9900 de Texas Instrument en vue d'une intégration dans un dispositif de filtrage en temps réel. On présente les résultats obtenus par cette première réalisation sur micro-processeur d'un opérateur de filtrage matriciel généralisé.

SUMMARY

In order to achieve the synthesis of a Fast operator dedicated to the numerical resolution of the Riccati matrix-equation, the author has completed the former description he had presented at the 79' GRETSI meeting, dealing with the comparison (Theoretical and experimental) of different methods.

The theoretical works have now been achieved, leading, during 1979 and 1980, to simulation experiments first on a general purpose CII-HB 66/10 computer, then on a 16 bits one chip microprocessor (TI 9900). This paper presents the main results of the experimentation.

OPERATEUR DE RICCATI RAPIDE
IMPLEMENTATION D'ALGORITHMES DE CALCUL
DU GAIN DE KALMAN OPTIMAL

1. - INTRODUCTION

L'obtention du gain optimal de KALMAN se fait de manière classique par la résolution de l'équation de RICCATI.

$$(1)\quad P = F\,P\,F^T - F\,P\,H^T\,[HP\,H^T + R]^{-1} HPF^T + Q$$

où F est la matrice de transition
Q est la matrice de covariance de bruit de commande
H est la matrice d'observation
R est la matrice de covariance de bruit de mesure.

avec $\dim F = n \times n$ $\qquad \dim Q = n \times n$
$\dim H = m \times n$ $\qquad \dim R = m \times m$

Q est symétrique semi définie positive
R est symétrique définie positive.

La matrice P solution de l'équation (1) permet de calculer le gain optimal.

$$K = PH^T\,[HPH^T + R]^{-1}$$

Différentes méthodes de résolution de l'équation de RICCATI ont été étudiées dans [1] et [2] ainsi que la transformation de cette équation en un type différent d'équation.

Après avoir rappelé brièvement les principaux algorithmes précédemment étudiés, il leur sera composé un nouvel algorithme à convergence quadratique puis on présentera les résultats de l'implementation des algorithmes choisis sur micro-processeur 16 eb. TI.9900.

2. - METHODES D'OBTENTION DU GAIN DE KALMAN OPTIMAL

L'obtention du gain de KALMAN se fait selon trois méthodes distinctes. La première consiste à calculer les valeurs propres du Hamiltonien du système [3], ce qui se ramène à résoudre une équation de degré 2n ou une équation de degré n et n équations de degré 2 compte-tenu des propriétés des racines de la première équation. Le gain obtenu par cette méthode sera appelé K sur les courbes 2.1 et 2.2

La deuxième méthode consiste à résoudre l'équation de RICCATI directement de façon itérative et donnera un gain KR, ou à calculer cette équation avec un algorithme de type NEWTON, c'est la méthode de quasi linéarisation où on calcule:

$$B_{k+1} = F\,P_k\,H^T\,[H\,P_k\,H^T + R]^{-1} \text{ avec } P_o = Q$$

puis on résoud l'équation de LIAPUNOV

$$P_k - (F - B_k\,H)\,P_k\,(F - B_k\,H)^T = Q + B_k\,R\,B_k^T$$

le gain obtenu ainsi sera noté KQ.

L'équation de RICCATI peut être transformée en un autre type d'équation. C'est le cas de l'algorithme de KAILATH qui la transforme en équation de type CHANDRASEKHAR et qui itère sur une projection de P : $L = P\,H^T$.

Enfin on peut appliquer une méthode d'accélération d'algorithme pour obtenir une convergence quadratique [6 à 8], c'est le cas de l'algorithme de BIERMAN qui s'écrit :

$$A_{k+1} = A_k\,(1 + B_k\,C_k)^{-1}\,A_k$$
$$B_{k+1} = B_k + A_k\,(1 + B_k\,C_k)^{-1}\,A_k^T$$
$$C_{k+1} = C_k + A_k^T\,(1 + B_k\,C_k)^{-1}\,A_k$$

avec $A_o = F$
$B_o = H\,R^{-1}\,H^T$
$C_o = Q$

et $P \triangleq \lim_{k \to \infty} C_k = P_{2k}$

Toutes les matrices sont de dimension n x n

Cet algorithme représente l'inconvénient d'inverser à chaque itération une matrice de dimension n x n. Le gain obtenu par cet algorithme sera noté KB.

Les algorithmes de KAILATH, de RICCATI itératif de BIERMAN et de quasi linéarisation ont été simulés en FORTRAN sur l'ordinateur CII HB 66/10 du CAPCA à la DCAN de TOULON.

Les courbes 2.1 et 2.2 montrent l'évolution des composantes du gain de KALMAN au cours des itérations.

Pour les courbes 2.1 les matrices du système sont : (modèle 1)

$$F = \begin{bmatrix} 1. & 0.05 \\ -0.05 & .97 \end{bmatrix} \qquad Q = \begin{bmatrix} 0.1 & 0.01 \\ 0.01 & 0.1 \end{bmatrix}$$

$$H = [1.\ ,\ 0] \qquad R = 1.$$

Dans ce cas les valeurs propres de F sont très proches de l'unité. Les courbes 2.2 correspondent au modèle 2.

$$F = \begin{bmatrix} 0. & 0.5 \\ 1. & 0.3 \end{bmatrix} \qquad Q = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}$$

$$H = [1.\ ,\ 0] \qquad R = 1.$$

Dans les deux cas l'algorithme qui converge le plus vite est l'algorithme de quasi linéarisation. Cependant sur les courbes 2.1 l'algorithme de KAILATH nécessite plus d'itérations car les valeurs propres sont voisines de l'unité mais compte tenu de l'écriture très simple de cet algorithme le temps de calcul n'en est pas pour autant plus important.

L'équation de LIAPUNOV discrète intervient pour l'initialisation de l'algorithme de KAILATH et à chaque itération dans l'algorithme de quasi linéarisation.

Cette équation est résolue de façon itérative par l'équation réccurente

$$P_{k+1} = F\,P_k\,F^T + Q$$
avec $P_o = Q$.

Cet algorithme a une convergence linéaire et peut être transformé pour obtenir une convergence quadratique en écrivant l'équation récurrente [11]

$$P_{2k+1} = F^{2k}\,P_{2k}\,(F^{2k})^T + P_{2k}$$
avec $P_o = Q$.

Les itérations dans tous les algorithmes sont arrêtées lorsque est vérifié le test de convergence :

$$\frac{\sum_{I=1}^{N}\sum_{J=1}^{N} \left| P_{k+1}(I,J) - P_k(I,J) \right|}{\sum_{I=1}^{N}\sum_{J=1}^{N} \left| P_{k+1}(I,J) \right|} < \varepsilon$$

où ε est la precision souhaitée.

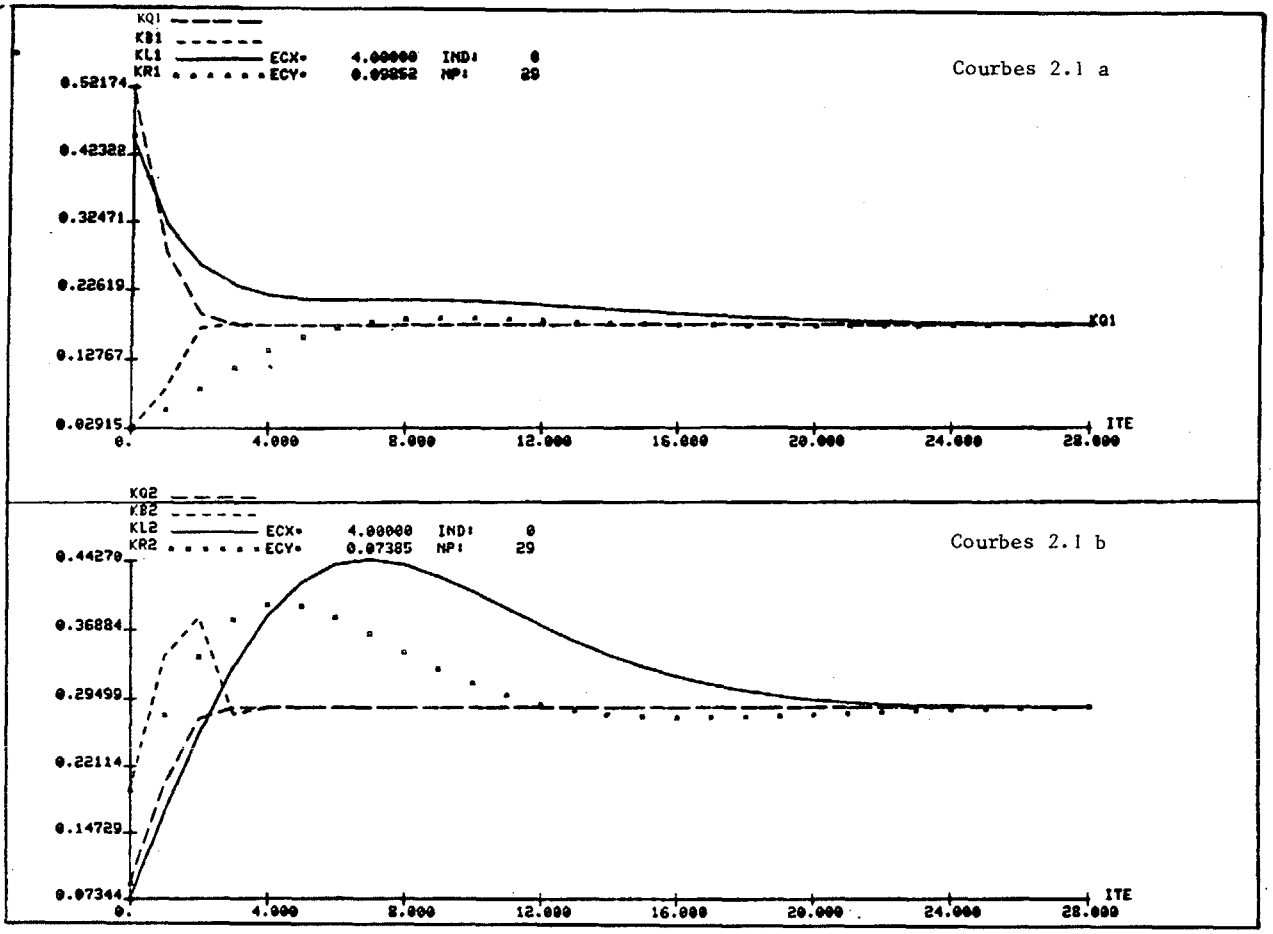
Les tableaux 2.1 à 2.6 donnent les temps de calculs pour 100 itérations avec chaque algorithmes.

Les programmes ne font pas appel à des sous programmes de multiplication ou d'addition matricielle et, pour minimiser le nombre d'opérations tout en conservant la propriété de symétrie de la matrice P, il n'est calculé à chaque itération que la partie triangulaire supérieure, l'autre partie en étant déduite par recopie.

En conclusion ce sont les algorithmes de quasi linéarisation et KAILATH qui seront implémentés sur micro-processeur avec la résolution de l'équation de LIAPUNOV par l'algorithme à convergence quadratique. Ces algorithmes seront écrits en PASCAL pour une sortie scalaire mais avec une dimension n variable.

OPERATEUR DE RICCATI RAPIDE
IMPLEMENTATION D'ALGORITHMES DE CALCUL
DU GAIN DE KALMAN OPTIMAL

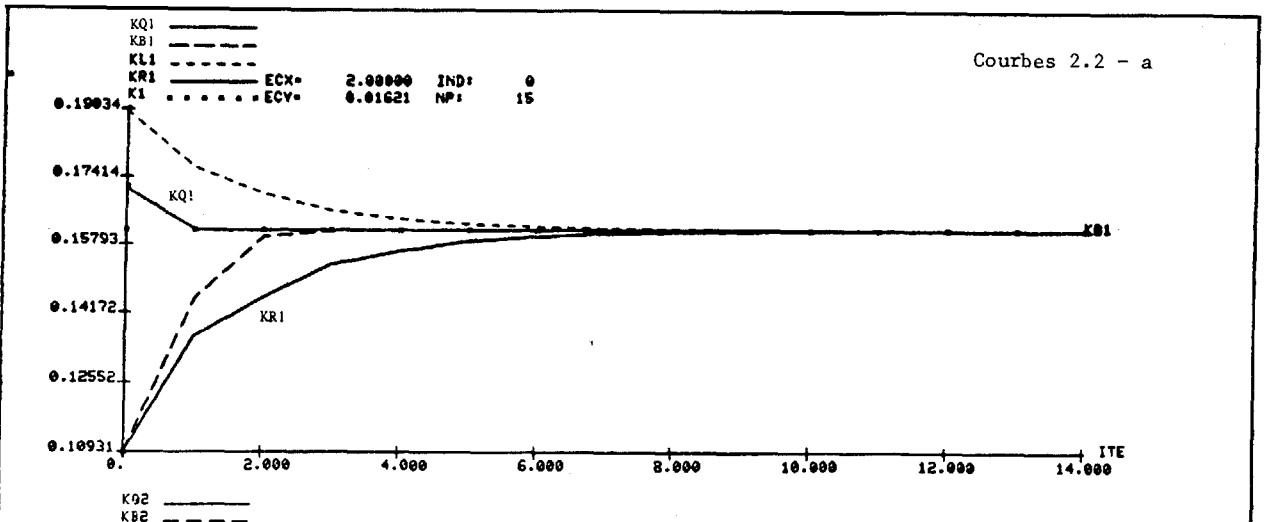
- Courbes 2.1
a: Evolution de K (1)
b: Evolution de K (2)



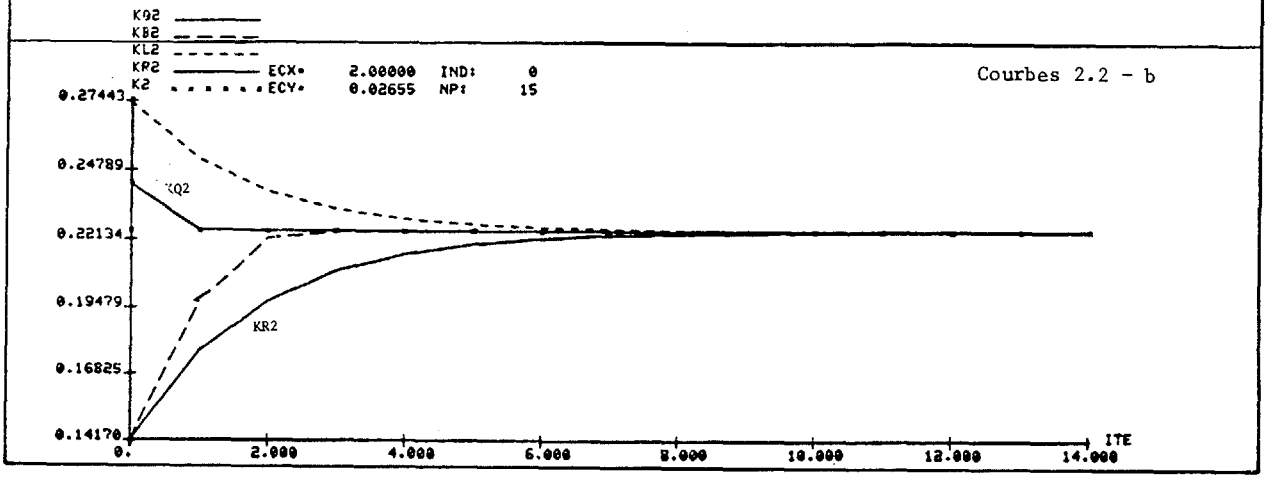
Courbes 2.1 a

Courbes 2.1 b

- Courbes 2.2
a: Evolution de K (1)
b: Evolution de K (2)



Courbes 2.2 - a



Courbes 2.2 - b

OPERATEUR DE RICCATI RAPIDE
IMPLEMENTATION D'ALGORITHMES DE CALCUL
DU GAIN DE KALMAN OPTIMAL

ALGORITHME DE RICCATI ITERATIF

TEMPS DE CALCUL SUR CII-HB 66/10 EN SECONDES
pour 100 iterations

N	M=1	M=2	M=3	M=4	M=5	M=6	M=7	M=8	M=9	M=10
2	.43	.75								
3	1.10	1.66	2.48							
4	1.90	2.60	3.60	4.95						
5	3.71	4.82	6.26	8.14	10.5					
6	7.60	9.40	11.7	14.3	17.7	21.2				
7	9.4	11.3	13.6	16.5	19.9	24.0	28.7			
8	11.7	13.8	16.0	19.0	22.4	26.2	31.0	36.5		
9	18.6	21.5	24.8	28.7	33.3	38.6	44.7	51.8	60.1	
10	30.5	34.7	36.4	44.2	50.5	57.4	65.5	74.4	84.0	95.
11	39.5	44.2	49.7	56.0	63.0	70.8	79.6	89.7	100.	112.
12	50.8	56.5	63.0	69.1	76.3	86.0	96.0	107.	118.	131.
13	63.8	70.5	77.4	84.6	94.1	104.	115.	126.	140.	154.
14	78.5	85.5	93.8	102.	112.	123.	135.	148.	163.	179.
15	75.8	82.3	89.8	98.2	107.	118.	128.	140.	153.	167.
16	77.4	84.2	91.5	99.4	108.	118.	128.	138.	149.	162.
17	108.	116.	125.	136.	147.	159.	172.	186.	201.	218.
18	161.	172.	184.	198.	212.	228.	245.	263.	283.	304.
19	188.	200.	214.	228.	244.	261.	280.	300.	387.	348.
20	217.	231.	246.	262.	279.	298.	318.	339.	362.	386.

Tableau 2.1

ALGORITHME DE KAILATH

TEMPS DE CALCUL SUR CII-HB EN SECONDES
pour 100 iterations et sans resolution du LYAPUNOV d'initialisation

N	M=1	M=2	M=3	M=4	M=5	M=6	M=7	M=8	M=9	M=10
2	.32	1.08								
3	.45	1.46	3.41							
4	.56	1.69	3.84	7.00						
5	.76	2.22	4.77	8.86	14.3					
6	1.14	3.27	6.87	12.2	19.7	29.2				
7	1.16	3.17	6.55	11.4	18.0	27.0	38.2			
8	1.27	3.34	6.77	11.5	18.2	27.0	38.5	52.6		
9	1.62	4.23	8.30	14.1	21.9	32.0	44.6	60.7	79.2	
10	2.27	5.88	11.3	18.8	28.8	41.7	57.7	77.3	101.	131.
11	2.62	6.67	12.6	20.8	31.5	45.1	62.1	82.6	107.	139.
12	3.02	7.54	14.3	23.0	34.5	49.2	67.4	89.0	115.	145.
13	3.43	8.50	15.6	25.2	37.5	52.9	71.8	94.8	121.	153.
14	3.81	9.38	17.1	27.3	40.3	56.6	76.5	100.	158.	161.
15	3.27	7.90	14.3	22.7	33.8	47.5	64.1	84.1	107.	135.
16	3.30	7.90	14.3	22.6	33.1	46.1	62.3	81.3	104.	129.
17	3.99	9.48	16.9	26.5	38.8	53.9	72.3	94.0	119.	148.
18	5.75	13.5	24.0	37.3	53.9	74.0	98.2	127.	160.	199.
19	6.29	14.7	25.9	40.0	57.5	78.7	104.	134.	169.	208.
20	6.89	16.0	27.9	42.9	61.4	83.6	110.	141.	177.	219.

Tableau 2.2

COMPARAISON CAS SACLAIRE H = [1, 0,...,0]

N	TEMPS DE CALCUL SUR CII-HB EN SECONDES POUR 100 ITERATIONS	
	KAILATH	RICCATI
2	.09	.35
3	.15	1.03
4	.22	1.92
5	.31	3.98
6	.44	7.90
7	.52	10.14
8	.60	13.1
9	.78	20.6
10	.99	33.3
11	1.17	43.7
12	1.37	56.0
13	1.56	70.7
14	1.80	87.7
15	1.91	107.2
16	1.94	129.4
17	2.36	154.4
18	2.80	183.4
19	3.08	214.1
20	3.39	249.4

Tableau 2.3

ALGORITHME DE QUASI LINEARISATION

TEMPS DE CALCUL SUR CII-HB EN SECONDES
pour 100 iterations et sans resolution du LYAPUNOV

N	M=1	M=2	M=3	M=4	M=5	M=6	M=7	M=8	M=9	M=10
2	.36	.79								
3	.68	1.38	2.42							
4	.99	1.92	3.24	5.00						
5	1.56	2.96	4.79	7.15	10.1					
6	2.75	5.07	8.01	11.6	16.1	19.3				
7	2.86	5.22	8.19	11.8	16.0	21.2	27.0			
8	3.26	5.88	9.17	13.2	17.6	23.0	29.3	36.5		
9	4.47	8.0	12.3	17.3	23.1	29.8	37.4	46.1	55.9	
10	6.8	12.2	18.3	25.6	34.0	43.3	53.7	65.5	78.9	93.
11	8.1	14.4	21.5	29.9	39.2	49.7	61.5	74.6	89.0	104.
12	9.6	16.8	25.1	34.5	45.2	57.2	70.3	84.6	100.	118.
13	11.1	19.4	28.8	39.5	51.2	64.8	79.0	95.1	112.	131.
14	12.7	22.2	32.7	44.6	57.8	72.5	88.6	106.	125.	146.
15	10.9	19.0	28.2	38.4	49.7	62.4	76.1	91.2	108.	126.
16	11.0	19.1	28.2	38.3	49.5	61.9	75.5	90.5	106.	125.
17	13.8	24.0	35.3	47.7	61.4	76.8	93.	111.	130.	151.
18	20.5	35.2	51.4	69.2	88.8	110.	132.	157.	184.	212.
19	22.8	39.0	56.8	76.3	98.2	121.	146.	173.	201.	232.
20	25.1	42.9	62.4	83.6	106.	131.	158.	186.	217.	250.

Tableau 2.4

ALGORITHME DE BIERMAN

VALEUR DE N	TEMPS DE CALCUL SUR CII-HB 66/10 pour 100 iterations EN SECONDES
2	.85
3	2.74
4	5.48
5	11.4
6	23.9
7	30.6
8	40.3
9	63.6
10	104.
11	138.
12	178.
13	225.
14	279.
15	272.
16	294.
17	393.
18	587.
19	688.
20	806.

Tableau 2.5

COMPARAISON DES ALGORITHMES DE RESOLUTION DE L' EQUATION DE LYAPUNOV DISCRETE

VALEUR DE N	TEMPS DE CALCUL SUR CII-HB 66/10 EN SECONDES POUR 100 ITERATIONS	
	ITERATIF	ACCELERE
2	.18	.3
3	.58	.9
4	1.04	1.8
5	2.27	3.7
6	4.68	8.0
7	5.94	9.7
8	7.38	13.3
9	12.28	20.2
10	20.23	34.8
11	26.64	45.7
12	34.2	58.8
13	43.2	74.2
14	53.5	92.1
15	51.6	84.6
16	52.9	93.6
17	47.5	122.4
18	111.6	192.6
19	130.7	227.9
20	152.1	263.5

k itérations de l'algorithme accéléré correspond à 2^k itérations de l'algorithme de LIAPUNOV itératif normal.

Tableau 2.6

OPERATEUR DE RICCATI RAPIDE
IMPLEMENTATION D'ALGORITHMES DE CALCUL
DU GAIN DE KALMAN OPTIMAL

3. - IMPLEMENTATION SUR MICRO-PROCESSEUR

3.1 - Implementation sur le micro-processeur 9900 de Texas INSTRUMENTS [9]

Les algorithmes de KAILATH et de quasi linéarisation ont été implementés sur le micro processeur 9900 qui possède une horloge de 4 MHz.

Les nombres réels sont codés en héxadécimal flottant sur deux octets selon la représentation :

0	1....7	8.....15
S	exposant	8 e.bles + significatifs
16 e.b les moins significatifs		

Les programmes ont été écrit en PASCAL puis compilés. Les mesures ont été effectuées sur une carte - cible TM 990/101 M et équipée d'une carte d'extension mémoire de 32 K octets.

Les calculs en virgule flottante sont fait par programme. Il est à noter que le nouveau micro-processeur 9995 de TEXAS a une fréquence d'horloge plus rapide (6 MH_z) et réalise les calculs en virgule flottante par micro-code interne, ce qui devrait augmenter la rapidité des calculs d'un facteur 20 à 30.

Les itérations de tous les algorithmes implementés sont arrêtées lorsque la différence relative entre deux itérations successives est inférieure à 10^{-3}.

3.2 - Comparaison des algorithmes de quasi linéarisation et de KAILATH.

Ces deux algorithmes ont été implémentés dans le cas d'une sortie scalaire avec une matrice d'observation qui s'écrit.

$$H = [1,0.....,0]$$

et pour une dimension n quelconque.

Les temps de calculs ont été mesurés pour 20 itérations et sans résolution de l'équation de LIAPUNOV. Les résultats sont présentés sur le tableau 3.1 pour les temps de calcul et sur le tableau 3.3 pour leur occupation mémoire.

DIMENSION	KAILATH	Quasi Linéarisation
N=2	0,089	1,08
3	0,131	1,88
4	0,185	2,97
5	0,248	4,36
6	0,320	6,00
7	0,401	7,95
8	0,491	10,16
9	0,589	12,84
10	0,696	15,68
11	0,812	18,78
12	0,937	22,17
13	1,07	25,88
14	1,21	29,8
15	1,36	34,1
16	1,52	38,6

Temps en secondes

tableau 3-1. Comparaison des temps de calcul des algorithmes de KAILATH et de quasi linéarisation pour 20 itérations sur micro-processeur 9900.

3.3 - Comparaison des algorithmes de résolution de l'équation de LIAPUNOV discrète.

Les deux algorithmes de l'équation de LIAPUNOV ont été implémentés sur le micro-processeur 9900.

Les tableaux 3.2 et 3.3 donnent la comparaison de temps de calcul de 20 itérations ainsi que leur occupation mémoire vive, mémoire morte.

DIMENSION	LIAPUNOV	LIAPUNOV ACCELEREE
N=2	0,932	1,573
3	2,458	3,68
4	5,21	7,52
5	9,55	13,62
6	15,84	22,5
7	24,45	34,6
8	35,73	50,6
9	50,4	71,0
10	67,7	96,2
11	89,2	126,8
12	114,7	163,3
13	144,8	206,2
14	179,7	256,0
15	219,7	313,4
16	265,4	378,8

Temps en secondes

tableau 3.2 - Comparaison des temps de calcul (en secondes des algorithmes de résolution de l'équation de LIAPUNOV sur micro-processeur 9900 pour 20 itérations ,indépendemment de la convergence

Algorithme	Mémoire morte (ROM) en octets	Mémoire vive (RAM) en mot (=2 octets)
LIAPUNOV	766	$5n^2 + 4$
LIAPUNOV Accéléré	1224	$4n^2 + 4$
Quasi Linéarisation	736	$2n^2 + 2n + 5$
KAILATH	258	$2n^2 + 3n + 7$

tableau 3.3 - Comparaison de l'occupation mémoire nécessaire aux différents algorithmes implémentés sur le micro-processeur TEXAS 9900. (PASCAL compilé).

Tableau de comparaison des algorithmes en fonction de la convergence
Valeur du test : $\varepsilon = 0,001$

	Modèle 1		Modèle 2	
ALGORITHME DE KAILATH				
LIAPUNOV accéléré Nombre d'itérations	9	temps : 0,707 s	6	temps : 0,471 s
Nombre d'itérations KAILATH	30	temps : 0,133 s	5	temps : 0,022 s
Total		0,840 s	Total	0,491 s
ALGORITHME DE QUASILINEARISATION				
Nombre d'itérations quasilinéarisation	3	temps : 0,162 s	4	temps : 0,216 s
Nombre d'itérations LIAPUNOV accéléré	5+5+5	temps : 1,179 s	5+4+4	temps : 1,022 s
Total		1,341 s	Total	1,236 s

Tableau de comparaison de la vitesse de convergence des algorithmes de résolution de l'équation de LIAPUNOV équivalence des nombres d'itérations pour obtenir la convergence.

LIAPUNOV itératif	1	2	3	4 à 7	8 à 15	16 à 31	32 à 64	2^n à $2^{n+1}-1$
LIAPUNOV accéléré	1	2	3	4	5	6	7	n+2

4. CONCLUSION

La résolution de l'équation de LIAPUNOV par un algorithme à convergence quadratique améliore très nettement les temps de calcul des deux algorithmes qui avaient été retenus pour réaliser l'opérateur de RICCATI rapide. L'algorithme de KAILATH est beaucoup plus rapide car son écriture dans le cas d'une sortie scalaire est très simple. Si la dimension de la sortie devient importante par rapport à celle du système, c'est l'algorithme de quasilinéarisation qui est beaucoup plus rapide.

Remerciements.

Je tiens à remercier Monsieur l'Ingénieur Principal de l'Armement C. BOZZO. (GESTA/CAPCA de Toulon) pour toutes les discussions constructives que nous avons eu ainsi que pour les moyens qu'il a mis à ma disposition.
Je tiens aussi à remercier Mme ACKERMAN (TEXAS Instrument) pour les conseils qu'elle m'a prodigués à l'occasion de l'utilisation du micro-processeur 9900.

[1] A. GIULIERI. C. BOZZO."Analyse des méthodes de résolution numérique de l'équation discrète de RICCATI. Septième colloque sur le traitement du signal et ses applications. NICE Juin 1979.

[2] A. GIULIERI : Rapport final de la convention 77-34-229 : Etude théorique des algorithmes de résolution de l'équation de RICCATI. Juin 1979.

[3] D. VAUGHAN " A Non Recursive Algebraic Solution for the Discrete de RICCATI équation IEEE Aut Cont. p 597-599 Octobre 1970.

[4] D. ZILMER Rapport TP5180. Naval Weapons Center Mai 1971.

[5] B. FRIEDLANDER, T. KAILATH, L. TJUNG. " Scattering Théory and Linear Least Squares Estimation - II Discrete time Problem" Journal of the FRANKLIN Institute Vol 301 n° 1 et 2. Janv-Feb 1976.

[6] G. BIERMAN, G. SIDHU : " Integration Free Interval Doubling for RICCATI Equation Solutions" - IEEE Automatic Control Vol.22 n° 5 p. 831-835, Juin 1977.

[7] D. LAINOTIS : "A Unifing Framework for linear Estimation Generalized Partitioned Algorithms" - Information Sciences 10, p. 243-278, (1976).

[8] B. ANDERSON : "Second Order Convergent Algorithms for the Steady State RICCATI Equation" - Int. J. Control Vol. 78 n° 2 p. 295-306, (1978).

[9] - Documentation TEXAS Instruments.

[10] GENSHIRO - KITAGAWA "An Algorithm for solving the Matrix Equation $X = FXF^T + S$" Int. Journal Control. Vol. 25 n° 5 p.745-753 1977