Articles soumis par le département ETP à des revues de traitement du signal

- Centre de Recherches En Physique de L'Environnement Terrestre Et Planétaire

CENTRE NATIONAL D'ETUDES DES TELECOMMUNICATIONS

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

CENTRE DE RECHERCHES EN PHYSIQUE DE L'ENVIRONNEMENT TERRESTRE ET PLANETAIRE

CRPE

NOTE TECHNIQUE

CRPE / 188

ARTICLES SOUMIS PAR LE DEPARTEMENT ETP A DES REVUES DE TRAITEMENT DU SIGNAL

Par

Collectif groupe " TRAITEMENT DU SIGNAL"

RPE/ETP
38-40, rue du Général Leclerc
92131 ISSY-LES-MOULINEAUX,FRANCE

CENTRE NATIONAL D'ETUDES
DES TELECOMMUNICATIONS
Centre Paris B

CENTRE NATIONAL DE LA
RECHERCHE SCIENTIFIQUE
Département TOAE

**CENTRE DE RECHERCHES EN PHYSIQUE DE
L'ENVIRONNEMENT TERRESTRE ET PLANETAIRE**

NOTE TECHNIQUE CRPE/188

ARTICLES SOUMIS PAR LE DEPARTEMENT ETP A DES REVUES DE TRAITEMENT DU SIGNAL

par

Collectif groupe "TRAITEMENT DU SIGNAL"
RPE/ETP
38-40 rue du Général Leclerc
92131 ISSY-LES-MOULINEAUX

Le Directeur

G. SOMMERIA

Le Directeur Adjoint

J.J. BERTHELIER

Juin 1990

LISTE DE DIFFUSION SYSTEMATIQUE

CNET

MM.	POITEVIN	Directeur du CNET
	THABARD	Directeur Adjoint du CNET
	COLONNA	Adjoint Militaire au Directeur du CNET
	MERLIN	Directeur des Programmes
	BLOCH	DICET
	THUE	DICET
MME	HENAFF	DICET
MM.	PIGNAL	PAB
	RAMAT	PAB
	NOBLANC	PAB-BAG
	ABOUDARHAM	PAB-SHM
	HOCQUET	PAB-STC
	THEBAULT	PAB-STS
MME	PARIS	PAB-RPE
MM.	BAUDIN	PAB-RPE
	BERTHELIER	PAB-RPE
	BIC	PAB-RPE
	CERISIER	PAB-RPE
	GENDRIN	PAB-RPE
	LAVERGNAT	PAB-RPE
	ROBERT	PAB-RPE
	ROUX	PAB-RPE
	SOMMERIA	PAB-RPE
	TESTUD	PAB-RPE
	VIDAL-MADJAR	PAB-RPE

CNRS

MM.	BERROIR	TOAE
	CHARPENTIER	SPI
MME	SAHAL	TOAE
MM.	COUTURIER	INSU
MME	LEFEUVRE	AD3
M.	DUVAL	AD5

CNES

MMES	AMMAR
	DEBOUZY
MM.	BAUDOIN
	FELLOUS
	HERNANDEZ (Toulouse)

Bibliothèques

CNET-SDI (3)
CNET-EDB
CNET-RPE (Issy) (5)
CNET-RPE (St Maur) (2)
Observatoire de Meudon
CNRS-SA
CNRS-INIST
CNRS-LPCE

LISTE COMPLEMENTAIRE

LAA/TSS/CMC	DI FRANCESCO
LAA/TSS/CMC	GILLOIRE
LAA/TSS/CMC	PETIT
LAB/MER/STA	RENAN
LAB/IFE/CIP	TREGUIER
LAB/IFE/COD	
PAB/BAG/CMM	CAQUOT
PAB/STC/PSN	MAITRET
PAB/SHM/SMC	TORTELIER
PAB/SHM/PHZ	
PAB/RPE/EMI	CERISIER
PAB/RPE/ETP	BENESTY
PAB/RPE/ETP	GLOAGUEN
PAB/RPE/ETP	YVON
PAB/RPE/ITS	MARZOUG

CNS - GRENOBLE

DIR/SVP	ARNDT
CCI/MCC	CAND
CCI/AMS	PRIVAT

CCETT - RENNES

SRL/MNC	CASTELAIN
SRL/MTT	LANOISELEE
SRL/MTT	PALICOT
PFI/IPT	ROCHE
DOCUMENTATION	

SEPT - CAEN

PEM	VALLEE

EXTERIEUR

ENST	BARRAL
ENST	GRENIER
ENST	MOU
ENST	RENARD
ENST	BIBLIOTHEQUE

Le délai de parution des articles dans les revues à comité de lecture comme IEEE Trans. on Audio, Speech, and Signal Processing est très long : les articles sont acceptés en moyenne un an après leur date de soumission (statistique officielle) et paraissent en ce moment un an après leur acceptation (à cause de la très longue file d'attente), soit un délai total moyen de 2 ans après soumission.

Ce délai nous semblant peu raisonnable, il nous a semblé nécessaire d'une part de diffuser l'information en interne au CNET, et d'autre part de disposer d'une référence interne permettant de dater le travail. C'est pourquoi nous publierons désormais régulièrement en note technique les articles soumis à des revues à comité de lecture.

Un certain nombre d'articles vont paraître dans un délai raisonnable. Il ne nous a pas paru nécessaire de les joindre. Nous en fournissons ci-après la liste.

Les personnes intéressées peuvent en obtenir une copie avant parution dans la revue en contactant les auteurs.

ZHANG H.M. DUHAMEL P., TRESSENS.S.
An improved Burg type recursive lattice method for autoregressive spectral analysis
IEEE Trans. ASSP [à paraître, août 1990]

DUHAMEL P.
Algorithms meeting the lower bounds on the multiplicative complexity of length-2^n DFT's and their connection with practical algorithms
IEEE Trans. ASSP [à paraître, septembre 1990]

DUHAMEL P.
A connection between bit-reversal and matrix transposition. Hardware and software consequences.
IEEE Trans. ASSP [à paraître, novembre 1990]

ARTICLE 1

Short-length FIR filters and their use in fast non-recursive filtering,
par Z.J. MOU et P. DUHAMEL

soumis en Mai 1989 à IEEE Trans. on ASSP

Short-length FIR filters and their use in fast non-recursive filtering

Z.J. MOU, P. DUHAMEL

Centre National d'Etudes des Télécommunications
CNET/PAB/RPE 38-40 rue du Général Leclerc
94131 ISSY-LES-MOULINEAUX
France

Abstract

This paper provides the basic tools required for an efficient use of the recently proposed fast FIR algorithms. These algorithms allow not only to reduce the arithmetic complexity but also maintain partially the multiply-accumulate structure, thus resulting in efficient implementations.

A set of basic algorithms is dèrived, together with some rules for combining them. Their efficiency is compared with that of classical schemes in the case of three different criteria, corresponding to various types of implementation. It is shown that this class of algorithms (which includes classical ones as special cases) allows to find the best tradeoff corresponding to any criterion.

Manuscript received_____

The authors are with Centre National d'Etudes des Télécommunications, CNET/PAB/RPE, 92131 Issy-Les-Moulineaux, France.

1. Introduction

A lot of algorithms are known to reduce the arithmetic complexity of FIR filtering. The widely used ones are indirect algorithms, based either on the cyclic convolution or on the aperiodic convolution using fast transforms as an intermediate step. Direct methods without transforms were also proposed by Winograd [1].

Both direct and indirect methods require large block processing: they make use of the redundancy between at least L successive output computations (L is the length of the filter) to reduce the number of operations to be performed per output point.

Furthermore, the structure of the resulting algorithm has completely changed : the initial computation is mainly based on a multiply-accumulate (MAC) structure, while the fast algorithms always involve global exchange of data inside a large vector of size at least 2L.

These are the main reasons why the above fast algorithms are not of wide interest for real-time filtering : Hardware implementations require pipelining the whole system with many intermediate memories, which results in a large amount of hardware. On another side, software implementations on Digital Signal Processors (DSP's) are not very efficient, except for very large L, since those fast algorithms have lost the multiply-accumulate structure for which all DSP's are optimized.

In other words, the usefulness of such algorithms was diminished because the reduction in arithmetic requirements per output point was obtained at the expense of a loss of structural regularity.

But structural regularity is difficult to quantify : Hardware implementations do not require the same kind of regularity as VLSI implementations do and "structural regularity" is still another matter when thinking of DSP implementations.

Anyway, one fact remains: MAC structure is very efficient on any type of implementation, including those on general purpose computer.

Recently, a new class of fast FIR filtering algorithms taking these considerations into account was proposed [2,3,4]: These algorithms retain partially the FIR filter structure, while reducing the arithmetic complexity. They allow various tradeoffs between

structural regularity and arithmetic efficiency, including all classical schemes as special cases [10]. This flexibility in the derivation of the algorithms allows to find the best possible solution in any type of implementation.

The purpose of this paper is to provide the basic tools required for the derivation of algorithms meeting various tradeoffs in different implementations.

A brief description of these new algorithms is provided in Section 2. This description allows to understand the structure of the new algorithms: Short-length FIR filters with reduced arithmetic complexity where all multiplications are replaced by decimated subfilters. Since the process can be reiterated on the subfilters, the short-length filters are recognized to be the basic building tools of these fast algorithms.

Hence, Section 3 is concerned with the derivation of a set of algorithms. This section is mostly based on Winograd's work. We bring some improvements in the number of additions by recognizing that the FIR filtering, seen as a running process, involves a pseudocirculant matrix [5] instead of a general Toeplitz one. Another advantage of this presentation is the easy understanding of the transposition principle in the context of multi-input multi-output systems, overlapping between blocks being naturally taken into account. Using this pseudocirculant presentation, we can derive the transposed version of all fast FIR filtering algorithms in a very easy manner.

Section 4 addresses the case of multifactor algorithms. Iterating the basic process raises the question of the best ordering of the short modules and of the length where the decomposition has to be stopped. We provide the rules for obtaining the ordering of factors which results in the lowest arithmetic complexity. A comparison with classical algorithms (FFT-based ones) is also provided in the case of real valued signals.

Section 5 concludes and explains some open problems.

2. General description of the algorithm

Let us consider the filtering of a sequence $\{x_i\}$ by a length-L FIR filter with fixed coefficients $\{h_i\}$:

(1)

$$y_n = \sum_{i=0}^{L-1} x_{n-i}\, h_i \qquad n = 0, 1, 2, \ldots, \infty$$

In z-domain formulation, this convolution becomes a polynomial product :

(2)

$$Y(z) = H(z)\,X(z)$$

Where X and Y are of infinite degree while H(z) has degree L-1: In z domain, the filtering equation, seen as a running process, is described by the product of an infinite degree polynomial and a finite degree one.

Let us now decimate each of the three terms in eq.(2) into N interleaved sequences:

(3)

$$H_j(z) = \sum_{m=0}^{L/N-1} h_{mN+j} z^{-m} \quad ; j = 0,1,..,N-1$$

$$X_k(z) = \sum_{m=0}^{\infty} x_{mN+k} z^{-m} \quad ; k = 0,1,...,N-1$$

$$Y_i(z) = \sum_{m=0}^{\infty} y_{mN+i} z^{-m} \quad ; i = 0,1,...,N-1$$

Eq.(2) then becomes :

(4)

$$\sum_{i=0}^{N-1} Y_i(z^N) z^{-i} = \sum_{j=0}^{N-1} H_j(z^N) z^{-j} \sum_{k=0}^{N-1} X_k(z^N) z^{-k}$$

Eq.(4) is in the form of a polynomial product or an aperiodic convolution. The two polynomials to be multiplied have finite degree N-1, and their coefficients are themselves polynomials, either of finite degree, such as $\{H_i\}$, or of infinite degree, such as $\{X_i\}$ or $\{Y_i\}$.

Let us now forget for a while that the coefficients of the $(N-1)^{th}$ degree polynomials are also polynomials, and apply a fast polynomial product algorithm to compute the polynomial product in eq.(4). It is well known, since the work of Winograd [1] that the product of two polynomials with N coefficients can be obtained with a minimum of 2N-1 general multiplications. This minimum can be reached for small N, while for larger N the optimal algorithm involves too many additions to be of practical interest. In that case, suboptimal ones are often prefered. Hence, application of these polynomial product algorithms to eq.(4) will result in a scheme requiring 2N-1 "products", each one being in fact the product of a finite degree polynomial by an infinite

degree sequence, that is an FIR filtering of length-L/N.

Compared to the initial situation, the arithmetic complexity is now as follows :

Eq.(4) requires N^2 filterings of length L/N, which is about L multiply-accumulates (Macs) per output (it is only a rearrangement of the initial equation), while the fast polynomial product based scheme requires (2N-1) filterings of length L/N, which is about $L(2N-1)/N^2$ Macs per output. Thus the improvement in arithmetic complexity is proportional to the length of the filter, and this is obtained at a fixed cost, depending on N. This means that, for large L, this approach will always be of interest. Precise comparisons are provided in Section 3 for each algorithm.

Slight additional improvements can be obtained by further considering eq.(4): In fact, eq.(4) contains not only the polynomial product, which allows the arithmetic complexity to be reduced, but also the so-called "overlap" in classical FFT-based schemes. By equating both sides of eq.(4), we have :

(5)

$$Y_{N-1} = \sum_{i=0}^{N-1} X_{N-1-i} H_i$$

$$Y_k = z^{-N} \sum_{i=k+1}^{N-1} X_{N+k-i} H_i + \sum_{i=0}^{k} X_{k-i} H_i \qquad 0 \le k \le N-2$$

or in matrix form:

(6)

$$\begin{bmatrix} Y_{N-1} \\ Y_{N-2} \\ . \\ . \\ . \\ Y_0 \end{bmatrix} = \begin{bmatrix} H_0 & H_1 & . & . & . & H_{N-1} \\ z^{-N}H_{N-1} & H_0 & . & & & . \\ . & . & . & . & & . \\ . & & . & . & . & \\ . & & & . & . & H_1 \\ z^{-N}H_1 & . & . & . & z^{-N}H_{N-1} & H_0 \end{bmatrix} \begin{bmatrix} X_{N-1} \\ X_{N-2} \\ . \\ . \\ . \\ X_0 \end{bmatrix}$$

The right side of eq.(6) is the product of a pseudocirculant matrix [5] and a vector. Note that $\{X_i\}$ and $\{H_i\}$ play a symmetric role, hence can be exchanged in eq.(6). The equation is clearly in the form of a length-N FIR filter whose coefficients are $\{H_i\}$, of which N outputs $\{Y_i\}$ are computed. Following the notations of Winograd [1], we shall denote in the following an algorithm computing M outputs of a length-N FIR filter by an F(M, N)

algorithm. Considering the FIR filter as a whole, and the fast FIR algorithm as the "diagonalization" of the pseudocirculant matrix of eq.(6) results in some circumstances in a reduction of the number of additions involved, compared with the usual approach which separates the polynomial product and the overlap. This will be seen in Section 3.

With the explanation above, the proposed algorithms can be understood as a multidimensional formulation of the FIR filtering, where computations along one dimension are performed through an efficient F(N,N) algorithm, resulting in a reduced arithmetic complexity, while the other dimension uses a direct computation, thus allowing the process to be a running one.

Let us also point out that, if the FIR filter of eq.(5) or eq.(6) is computed through an FFT-based scheme, and with the appropriate choice of N versus L, the usual FFT-based implementation of FIR filters can be seen to be a member of that class of algorithms. This is explained in [10], where it is shown that all fast FIR schemes including FFT-based ones can be expressed as :

- decimation of the involved sequences (N on input and output, M on the filter)
- evaluation of the obtained polynomials at N+M-1 "interpolation points" $\{\alpha_i\}$;
- "dot product" (or filtering);
- reconstruction of the resulting polynomials and overlap.

All the algorithms differ only by the choice of N, M, and $\{\alpha_i\}$.

Section 3 provides various short-length FIR filtering algorithms in the case of real valued sequences.

3. Short-length FIR filtering algorithms

Let us first explain in some detail the simplest case : an F(2,2) algorithm, as given in [2,3,4]: considering N = 2, $\{\alpha_i\} = \{0,1,\infty\}$, we obtain :

(a1)
$$a_0 = x_0 \qquad b_0 = h_0$$
$$a_1 = x_0 + x_1 \qquad b_1 = h_0 + h_1$$
$$a_2 = x_1 \qquad b_2 = h_1$$

$$m_i = a_i\, b_i\,; \qquad i = 0,1,2$$

$$y_0 = m_0 + z^{-2} m_2$$
$$y_1 = m_1 - m_0 - m_2$$

When used in eq.(5) above, this algorithm results in the filtering scheme of Fig.1 where the problem of computing 2 outputs of a length N filter is turned into that of computing one output of three length-N/2 filters, at the cost of 4 adds per block of 2 outputs.

Comparison of arithmetic complexities are now as follows : the initial scheme has a cost of L mults and (L-1) adds per output point, to be compared with

3/4 L mults per output
$2 + 3/2\ (L/2 - 1) = 3/4\ L + 1/2$ adds per output.

It is seen that, excepted for very small L, both numbers of multiplications and additions have been reduced.

Successive decompositions are feasible, up to the point where the desired tradeoff has been obtained. Of course, this tradeoff depends on the implementation. On general purpose computers where a multiplication and an addition require about the same amount of time, the tradeoff allowing the fastest implementation will certainly correspond to a decomposition near the one minimizing the total number of arithmetic operations. On Digital Signal Processors, a multiply-accumulate operation will in general cost one clock cycle, and an appropriate criterion is certainly to count a Mac as a single operation. A third criterion of interest is the minimum number of multiplications. In the following, tables giving the minimum numbers for these three criteria will be provided.

Nevertheless, in nearly all type of implementations, the situation is much alike: the decomposition provided in Fig.1 reduces the arithmetic load in a manner proportional to L, the length of the filter, at a fixed cost (initialization of one Mac loop, or one Mac loop plus two adds). Hence the balance between the performances of algorithms depends on the timing spent in the initializations. The precise N by which the splitting becomes of interest therefore depends on the specific machine or circuit. Anyway, this type of splitting will always be of interest for large length filters, or even medium-size ones.

The remaining part of this section provides the simplest F(M,N) algorithms that can be used to reduce the arithmetic complexity of a length-L filtering. Different versions are provided, resulting in various operation counts, and various sensitivities to roundoff noise, a point which will not be dealt with in this paper.

The following algorithm is an F(2,2) algorithm with interpolation points $\{\alpha_i\} = \{0, -1, \infty\}$. It has exactly the same complexity as the previous one :

(a2)

$$a_0 = x_0 \qquad b_0 = h_0$$
$$a_1 = x_0 - x_1 \qquad b_1 = h_0 - h_1$$
$$a_2 = x_1 \qquad b_2 = h_1$$

$$m_i = a_i\, b_i\,; \qquad i = 0,1,2$$

$$y_0 = m_0 + z^{-2}\, m_2$$
$$y_1 = m_0 + m_2 - m_1$$

For the sake of completeness, two algorithms computing F(2,2) with $\{\alpha_i\} = \{0, 1, -1\}$ and $\{\alpha_i\} = \{1, -1, \infty\}$ are provided in Appendix B. They may be of interest as long as roundoff noise is concerned, but they require 3 multiplications and 6 additions, that is one more addition per output. Note that on some implementations (and essentially on DSP's), this is not a real drawback, since these additions are now of the type a±b, which can be efficiently implemented in many cases.

It is well known in the case of the usual implementation of a digital filter that the transposition of a graph provides a digital filter with the same transfer function. Winograd has proposed an approach allowing to obtain short-length FIR filtering algorithms by transposing polynomial product algorithms. We have shown that simple overlapping of polynomial products in eq.(5) is sufficient to construct FIR filtering algorithms. The transposition of polynomial products is not necessary. It only provides alternative versions of the algorithms. In the context of pseudocirculant matrix, we can transpose the algorithms in a much easier way, and we can prove that the total arithmetic complexity of all these algorithms is not changed by the transposition operation (both number of multiplications and additions). More details are given in Appendix A.

The following algorithms are the transposed versions of algorithms (a1) and (a2).

(a3)

$$a_0 = x_0 - x_1 \qquad b_0 = h_0$$
$$a_1 = x_0 \qquad b_1 = h_0 + h_1$$
$$a_2 = z^{-2}\, x_1 - x_0 \qquad b_2 = h_1$$

$$m_i = a_i b_i\,; \qquad i = 0,1,2$$

$$y_0 = m_1 + m_2$$
$$y_1 = m_1 - m_0$$

(a4)

$$a_0 = x_0 + x_1 \qquad b_0 = h_0$$
$$a_1 = x_0 \qquad b_1 = h_0 - h_1$$
$$a_2 = z^{-2} x_1 + x_0 \qquad b_2 = h_1$$

$$m_i = a_i b_i ; \qquad i = 0,1,2$$

$$y_0 = m_1 + m_2$$
$$y_1 = m_0 - m_1$$

The use of (a3) in a large FIR filter is provided in Fig.2.

Iterating the above algorithms results in radix-2 FIR algorithms, with a tree-like structure, as proposed in [4].

Higher radix algorithms can also be derived, and should be more efficient, as seen at the end of Section 2, since the ratio $(2N-1)/N^2$ decreases. However, an optimal F(3,3) algorithm would require 5 different interpolation points, that is one more than the simplest ones: $\{\alpha_i\} = \{0,1,-1, ?,\infty\}$, and the next simplest choices of the last interpolation point $\{\pm 2,\pm 1/2\}$ result in an increased number of additions, and an increased sensitivity to roundoff noise. This is the reason why it is advisable to use a suboptimal F(3,3) algorithm which provides a better tradeoff between the number of multiplications and the number of additions.

Such an algorithm can be obtained by applying twice the algorithm (a1) as follows. Let us remark that (a1) is based on the following equation :

(8)

$$(x_0 + x_1 z^{-1}) (h_0 + h_1 z^{-1})$$
$$= x_0 h_0 + x_1 h_1 z^{-2} + [(x_0 + x_1) (h_0 + h_1) - x_0 h_0 - x_1 h_1] z^{-1}$$

Inspired by eq.(8) we rewrite the radix-3 aperiodic convolution equation as follows :

(9)

$$[x_0 + (x_1 + x_2 z^{-1}) z^{-1}] [h_0 + (h_1 + h_2 z^{-1}) z^{-1}]$$
$$= (x_0 + Fz^{-1}) (h_0 + Gz^{-1})$$
$$= x_0 h_0 + [(x_0 + F) (h_0 + G) - x_0 h_0 - FG]z^{-1} + FGz^{-2}$$

Then, the algorithm in eq.(8) is once more applied to the computation of $(x_0+F)(h_0+G)$ and FG which are still radix-2 aperiodic convolutions. This results in an aperiodic convolution algorithm requiring 6 mults and 9 adds (an optimal one would require 5 mults and 20 adds [11, pp.86]). Overlap has then to be performed by merging the terms z^0 and z^{-3}, and the term z^{-1} and z^{-4}, with the appropriate delay. Hence, the F(3,3) algorithm seems to require 2 more adds than the corresponding radix-3 aperiodic convolution algorithm. Nevertheless, considering the redundancy during the overlap in F(3,3) algorithm allows a further reduction of the number of additions to 10 adds, the lowest number of operations to our knowledge.

(a5)
$$a_0 = x_0 \qquad b_0 = h_0$$
$$a_1 = x_1 \qquad b_1 = h_1$$
$$a_2 = x_2 \qquad b_2 = h_2$$
$$a_3 = x_0 + x_1 \qquad b_3 = h_0 + h_1$$
$$a_4 = x_1 + x_2 \qquad b_4 = h_1 + h_2$$
$$a_5 = x_0 + a_4 \qquad b_5 = h_0 + h_1 + h_2$$

$$m_i = a_i b_i ; \qquad i = 0,1,2,3,4,5$$

$$t_0 = m_0 - m_2 z^{-3}$$
$$t_1 = m_3 - m_1$$
$$t_2 = m_4 - m_1$$

$$y_0 = t_0 + t_2 z^{-3}$$
$$y_1 = t_1 - t_0$$
$$y_2 = m_5 - t_1 - t_2$$

The use of this F(3,3) algorithm to reduce the arithmetic complexity of a larger FIR filter is provided in Fig.3, showing that the overall structure is that of a multirate filter bank, where $\{H_1 + H_2, H_1, H_0 + H_1, H_2, H_0, H_0 + H_1 + H_2\}$ are decimated FIR filters.

Transposition of algorithm (a5) results in the following one, which is depicted in Fig.4.

(a6)
$$a_0 = x_2 - x_1 \qquad b_0 = h_0$$
$$a_1 = (x_0 - x_2 z^{-3}) - (x_1 - x_0) \qquad b_1 = h_1$$
$$a_2 = - a_0 z^{-3} \qquad b_2 = h_2$$
$$a_3 = (x_1 - x_0) \qquad b_3 = h_0 + h_1$$

$a_4 = (x_0 - x_2 z^{-3})$ $\qquad$ $b_4 = h_1 + h_2$

$a_5 = x_0$ $\qquad$ $b_5 = h_0 + h_1 + h_2$

$m_i = a_i b_i$; $\qquad$ i =0,1,2,3,4,5

$y_0 = m_2 + (m_4 + m_5)$

$y_1 = m_1 + m_3 + (m_4 + m_5)$

$y_2 = m_0 + m_3 + m_5$

When used in a length-L FIR filter, both schemes (a5) and (a6) require 2L/3 multiplications and (2L+4)/3 additions per output point, to be compared with L and L-1 operations respectively in the direct computation. This means that the computational load has been reduced by nearly 1/3.

Careful examination of Fig.1 - 2 and 3 - 4 shows that the distribution of the additions between the input samples and the subfilters' outputs is not the same in the initial algorithms and their transposed versions: In all cases, transposed algorithms have more input additions and less output additions. This fact should give them more robustness towards quantization noise.

Another case, which looks interesting at first glance is as follows: why not decimating the filter by a factor of 2, and X and Y by a factor of 3? This would be solved by an F(3,2) algorithm, which requires 3 + 2 - 1 = 4 interpolating points, that is the very number of the simplest interpolating points {0,1,-1,∞}. This means that F(3,2) or F(2,3) are the largest filtering modules that can be computed efficiently with an optimum number of multiplications :

(a7) $a_0 = x_3 - x_1$ $\qquad$ $b_0 = h_0$

$a_1 = x_1 + x_2$ $\qquad$ $b_1 = (h_0 + h_1)/2$

$a_2 = x_1 - x_2$ $\qquad$ $b_2 = (h_0 - h_1)/2$

$a_3 = x_2 - x_0$ $\qquad$ $b_3 = h_1$

$m_i = a_i b_i$; $\qquad$ i = 0,1,2,3

$y_0 = (m_1 + m_2) - m_3$

$y_1 = m_1 - m_2$

$y_2 = m_0 + (m_1 + m_2)$

This algorithm looks promising, since the same performance as an F(3,3) algorithm is obtained with a simpler one: F(3,2) requires 4 multiplications and 8 additions, which means that it reduces of the number of Macs by 1/3, at the cost of 8 additions. Nevertheless, problems arise when using this algorithm for speeding up the computation of a large length-L filter. The overall structure is depicted in Fig.5, where the main computing modules are a kind of 1/3 FIR decimators. Obtaining 3 successive outputs of the filter requires the computation of 4 length-L/2 inner products plus 13 adds, to be compared with algorithm (a5) which requires 6 length-L/3 inner products, plus 10 adds. Therefore, (a7) and (a5) have nearly the same arithmetic complexity. But (a7) requires 3L memory registers, instead of 2L registers in (a5), and a more complex control system, since the inner products are not true FIR filters any more.

All aperiodic convolution algorithms can be turned into FIR filtering algorithms by appropriate overlap. Suboptimal higher-radix($\geq$5) aperiodic convolution algorithms can be derived using the approach in [12]. An F(5,5) algorithm based on this approach is given in Appendix B. This algorithm, requiring 12 mults and 40 adds is not optimum as far as the number of multiplications is concerned, but reaches the best tradeoff we could obtain.

All these F(N,N) algorithms are clearly similar to the ones proposed by Winograd [1]. They differ essentially on several points :

First, the recognition that, by using decimated sequences, the arithmetic complexity of an FIR filter can be reduced as soon as two ouputs of the filter are computed regardless of the filter's length. Winograd's algorithms always require the computation of at least as many outputs as the filter's order.

Second, a straightforward derivation through eq.(4) of the F(N,N) algorithms by polynomial product algorithms which were extensively studied in the litterature.

Third, a reduction in the number of additions, which is made feasible by naturally taking into account the "overlap" between two consecutive blocks of outputs (Hence the use of the pseudocirculant matrix).

Fourth, a new interpretation, in the context of pseudocirculant matrix, of algorithm transposition and a systematic method of obtaining transposed versions.(See appendix A)

4. Composite length algorithms

We have explained in Section 2 that the application of an F(N,N) algorithm could break the computation of a length-L FIR filter into that of several length-L/N filters, in such a manner that the arithmetic complexity is decreased. Nevertheless, the same process can iteratively be applied to the subfilters of length L/N as well, leading to composite length algorithms.

This section is concerned with the problem of finding the best way of combining the small-length filters, depending on various criteria.

We first evaluate the arithmetic complexity of a length $L = N_1 N_2 L_2$ filter, when using two successive decompositions by $F(N_1,N_1)$ first , and then by $F(N_2,N_2)$. Let us assume that, with the notations of Section 3, the length-N_i filter requires M_i multiplications and A_i additions.

The first decomposition by $F(N_1,N_1)$ turns the initial problem into that of computing M_1 filters of length $N_2 L_2$ at the cost of A_1 additions. Further decompositions of the length-$N_2 L_2$ subfilters using $F(N_2,N_2)$ require the consideration of a block of N_2 outputs of these subfilters (hence a block of $N_1 N_2$ outputs of the whole filter). Each of these subfilters is then transformed into M_2 filters of length L_2 at the cost of A_2 additions.

The global decomposition thus has the following arithmetic complexity:

$N_2 A_1$ additions $+ M_1$ [M_2 length-L_2 subfilters + A_2 additions].

That is :

(10) $M = M_1 M_2 L_2$

(11) $A = N_2 A_1 + M_1 A_2 + M_1 M_2 (L_2 - 1)$

Let $m_i = M_i/N_i$ and $a_i = A_i/N_i$ where i=1,2. m_i and a_i are the numbers of operations (mults and adds respectively) per output required by $F(N_i,N_i)$. Since eq.(10) and (11) are the arithmetic load for computing $(N_1 N_2)$ outputs, the numbers of operations per output for the whole algorithm are :

(12) $m = m_1 m_2 L_2$

(13) $a = a_1 + m_1 a_2 + m_1 m_2 (L_2 - 1)$

An application of $F(N_2,N_2)$ first, followed by $F(N_1,N_1)$ would result in the same

number of multiplications , and a number of additions which will be higher than eq.(13) as long as :

(14) $a_1 + m_1 a_2 < a_2 + m_2 a_1$ or $(m_1-1)/a_1 < (m_2 -1)/a_2$

or, equivalently:

(15) $(M_1 -N_1)/A_1 < (M_2 -N_2)/A_2$

Let us define

(16) $Q[F(N_i,N_i)] = (m_i - 1)/a_i = (M_i -N_i)/A_i$

Q is a parameter specific of an algorithm. Its use has already been proposed in [8] for the cyclic convolution. Eq.(15) means that the lowest number of additions is obtained by first applying the short length FIR filter with the smallest Q, and then the one with the second smallest Q and so on. Table.1 provides Q[F(N,N)] for the most useful short-length filters. Two useful properties of Q[F(N,N)] are as follows :

- A straightforward FIR filtering "algorithm" has Q[F(1,N)] =1, whatever N is. This means that, in order to minimize the number of additions, they must be located at the "center" of the overall algorithm, as was implicitely assumed up to that point.

- Iteratively applying an algorithm F(p,p) to obtain $F(p^k,p^k)$ results in an algorithm with the same coefficient Q:

(17) $Q[F(p^k,p^k)] = Q[F(p, p)]$

The demonstration is easily obtained by simply recognizing that applying first $F(p^k,p^k)$ and second F(p,p) or applying them in the reverse order results in the same $F(p^{k+1},p^{k+1})$.

Hence, as an example, an optimal ordering for $L = 120\ L_i = 2^3 x 3 x 5\ L_i$ would be :

F (5,5), F (2,2), F (2,2), F (2,2), F (3,3), F $(1,L_i)$

Of course, when it is desired to implement a length-L filter, it is very unlikely that F(L,L) is the most suitable algorithm for a specific type of implementation, even in the

case where L is composite. The best thing to do, is to search for the tradeoff minimizing some criteria depending on the implementation.

It is not our propose here to perform such an optimization in a special case, but we shall try to show, in the following, that improvements are feasible whatever the criterion is.

Assuming that $L = N_1 \ldots N_i \ldots N_r L_r$, and that a fast algorithm $F(N_i, N_i)$ is used for all N_i, a straightforward one being used for L_r, the general formulae for evaluating the arithmetic complexity per output of a length-L filter are as follows :

(18)

$$m = L_r \prod_{i=1}^{r} m_i$$

(19)

$$a = \sum_{i=1}^{r} a_i \prod_{j=1}^{i-1} m_j + (L_r - 1) \prod_{i=1}^{r} m_i$$

A first criterion of interest would be the minimization of the number of multiplications. Examination of eq.(18) shows that such a minimization is performed by fully decomposing L into $N_1 \ldots N_i \ldots N_r L_r$, and this results in the number of operations given in Table.2.

All the operation counts are provided assuming that the signal and the filter are real-valued, and the comparison is made with the real-valued FFT-based schemes [6,9] of twice the filter's length. Table.2 shows that the proposed approach is more efficient than FFT-based schemes up to L = 36, and very competitive up to L = 64 which covers most useful lengths.

However, with today's technology, multiplication timings are not the dominant part of the computations any more when a parallel multiplier is built in the computer, and a very useful criterion is the sum of the number of additions and multiplications, i.e., M+A.

Table 3 provides a description of the algorithms requiring the minimum value for such a criterion. It is seen that, although the basic F(N,N) algorithms do not improve the criterion, all the composite ones can be improved, and are even more efficient than FFT-based schemes up to N = 64. Consider N = 16, for example: FFT-based schemes hardly improve the direct computation (29.5 operations per point versus 31), while

F(2,2)-based algorithm requires only 19.6 operations per point.

The previous criterion is well suited for general purpose computer implementation, where all operations are performed sequentially, but Digital Signal Processors still state another problem. In fact, DSP's perform generally a whole multiply-accumulate(Mac) operation in a single clock cycle, so that a Mac should be considered as a single operation, which is not more costly than an addition alone.

Table.4 provides a description of the algorithm minimizing the corresponding criterion: the sum of the number of Macs and I/O additions. Of course, this criterion is a rough measure of efficiency, since initialization time of Mac loops is not taken into account. Nevertheless, what Table.4 shows in that even this kind of criterion, taking partially into account the structure of DSP's, can be improved using our approach : a length - 64 filter can be implemented using this type of algorithms with nearly half the total number of operations (Macs+ I/O adds) per output point, compared to the trivial algorithm. And this is obtained with a block size of only 8 points. This shows that moderate up to large length filters can be efficiently implemented on DSP's using these techniques.

Two points should be emphasized here :

A lot of systems requiring digital filtering have constraints on the input/output delay, which prevents the use of FFT-based implementation of digital filter. Our approach allows to obtain a reduction of the arithmetic complexity whatever the block size is. This means that our approach allows to reduce the arithmetic complexity by taking into account such requirements as a constraint on the I/O delay.

Another remark, which is not apparent in the tables, is that for a given filter length L, there are often several algorithms providing comparable performance, for a given criterion. It allows us to choose among them the one which is the most suited for the specific implementation.

5. Conclusion

We have presented a new class of algorithms for FIR filtering, showing that the basic building tools are short-length FIR modules in which the "multiplications" are replaced by decimated subfilters.

We have provided also the basic tools required for implementing these algorithms:

First, we propose short-length FIR filter modules of the Winograd-type with a small number of multiplications and the smallest number of additions. Their transposed versions are also provided.

Second, we give some rules concerning the best way of cascading these short-length FIR modules to obtain composite-length algorithms.

Finally, we show that, for three different criteria, this class of algorithms allows to obtain better tradeoffs than the previously known algorithms, which should make them useful in any kind of implementation.

It is concluded that the presented algorithms allow not only to compute from moderate to long length FIR filtering on DSP's in a more efficient manner, but also to compute short-length (<64) FIR filtering more efficiently than FFT-based algorithms when considering the total number of operations.

These algorithms also suggest efficient multiprocessor implementations due to their inherent parallelism, and efficient realization in VLSI, since their implementations require only local communication, instead of a global exchange of data, as is the case for FFT-based algorithms.

Appendix A

A.1 Transposition of an F(N,N) algorithm

Let us consider an F(N,N) algorithm as an (N-input, N-output) system shown in Fig.6(a). Its transmission matrix is pseudocirculant:

(20)

$$\mathbf{P}(z) = \begin{bmatrix} H_0 & H_1 & \cdots & & H_{N-1} \\ z^{-N}H_{N-1} & H_0 & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & & \cdot & \cdot & \cdot \\ \cdot & & & \cdot & \cdot \; H_1 \\ z^{-N}H_1 & \cdot & \cdot & \cdot & z^{-N}H_{N-1} \; H_0 \end{bmatrix}$$

The transposed system (shown in Fig.6(b)) will have $\mathbf{P}^t(z)$, which is the transpose

of $\mathbf{P}(z)$, as its transmission matrix according to Tellegen's Theorem for digital networks [13].

Unlike (1-input, 1-output) systems, an (N-input, N-output) system's transpose will not perform the same function as the initial one unless $\mathbf{P}(z)$ is symmetric, i.e., $\mathbf{P}(z)= \mathbf{P}^t(z)$. Owing to $\mathbf{P}$'s Toeplitz structure, we can manage to get the tranposed system performing the same function as the initial one.

The transposed system is as follows:

(21)

$$\begin{bmatrix} Y'_{N-1} \\ Y'_{N-2} \\ . \\ . \\ . \\ Y'_0 \end{bmatrix} = \begin{bmatrix} H_0 & z^{-N}H_{N-1} & . & . & & z^{-N}H_1 \\ H_1 & H_0 & . & & & . \\ . & . & . & . & . & . \\ . & & & . & . & \\ . & & & & . & z^{-N}H_{N-1} \\ H_{N-1} & . & . & . & H_1 & H_0 \end{bmatrix} \begin{bmatrix} X'_{N-1} \\ X'_{N-2} \\ . \\ . \\ . \\ X'_0 \end{bmatrix}$$

If we permute the input elements $\{X'_i\}$ and the output elements $\{Y'_i\}$ in such a way that their order is reversed, we will get the transposed system to perform the appropriate function:

(22)

$$\begin{bmatrix} Y'_0 \\ Y'_1 \\ . \\ . \\ . \\ Y'_{N-1} \end{bmatrix} = \begin{bmatrix} H_0 & H_1 & . & . & . & H_{N-1} \\ z^{-N}H_{N-1} & H_0 & . & & & . \\ . & . & . & . & . & . \\ . & & & . & . & \\ . & & & & . & H_1 \\ z^{-N}H_1 & . & . & . & z^{-N}H_{N-1} & H_0 \end{bmatrix} \begin{bmatrix} X'_0 \\ X'_1 \\ . \\ . \\ . \\ X'_{N-1} \end{bmatrix}$$

The above demonstration is general, so that all (N-input,N-output) systems whose transmission matrix is Toeplitz can be transposed to perform the same function as the initial system after permuting the inputs and outputs in a reversed order. Thus, it can be applied to all kinds of convolution systems. Winograd has already proposed to transpose circular convolution algorithms in the same manner [7]. We summarize this principle as the following theorem.

Theorem: Having Toeplitz transmission matrix is sufficient for an (N-input,N-output) system to be transposed and to perform the same function after permuting the inputs and

outputs in a reversed order.

Hence, we can transpose an F(N,N) algorithm in a rather easy way. In fact, a fast F(N,N) algorithm diagonalizes a pseudocirculant matrix:

(23) $\mathbf{P}(z) = \mathbf{A}_{NxM}\,\mathbf{H}_{MxM}\,\mathbf{B}_{MxN}$

where $\mathbf{H}_{MxM}$ is a diagonal matrix. Then the transposition of $\mathbf{P}(z)$ is

(24) $\mathbf{P}^t(z) = \mathbf{B}^t\,\mathbf{H}\,\mathbf{A}^t$

Permuting the inputs and outputs in a reversed order is equivalent to permute the columns of $\mathbf{A}^t$ and the rows of $\mathbf{B}^t$ in a reversed order. If we denote a matrix $\mathbf{V}$ after such permutation of rows by $\mathbf{V}^{\sim}$, the following equation holds:

(25) $\mathbf{P}(z) = (\mathbf{B}^t)^{\sim}\,\mathbf{H}\,(\mathbf{A}^{\sim})^t$

Therefore, we get the transposed version of (23).

Let us consider as an example an F(2,2) algorithm , as given in (a1). It is expressed in matrix form as:

$$\begin{bmatrix} y_1 \\ y_0 \end{bmatrix} = \begin{bmatrix} h_0 & h_1 \\ z^{-2}h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_0 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 1 & -1 \\ 1 & 0 & z^{-2} \end{bmatrix} \begin{bmatrix} h_0 & & \\ & h_0 + h_1 & \\ & & h_1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_0 \end{bmatrix}$$

therefore

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 0 & z^{-2} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

Following (25), we obtain the transposed version of (a1):

$$\begin{bmatrix} y_1 \\ y_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} h_0 & & \\ & h_0 + h_1 & \\ & & h_1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \\ z^{-2} & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_0 \end{bmatrix}$$

which is the matrix expression of (a3).

A.2 Identical arithmetic complexity in initial and transposed F(N,N) algorithms

It is clear, following the above explanations, that the multiplicative complexity is not changed by transposition. Let us consider the additive complexity in an (N-input,N-output) system.

A digital network is composed of only branches and nodes. There are two kinds of nodes: a (M,1) summing node that adds M inputs into 1 output, and a (1,M) branching node that branches 1 input into M outputs. A (M,1) summing node can be splited into M-1 (2,1) summing nodes. A (1,M) branching node can be also splited into M-1 (1,2) branching nodes. Then it is easy to transform the network to an equivalent one having only (2,1) summing nodes and (1,2) branching nodes. After transposition, the summing nodes become branching nodes and vice versa.

The number of additions in the initial network is equal to that of (2,1) summing nodes, denoted by Ns. The number of additions in the transposed network is equal to that of (2,1) branching nodes in the initial network, denoted by Nb. The additive complexity in initial and transposed algorithms is identical if and only if Ns=Nb, and we show in the following that this property holds for (N-input, N-output) systems.

Proof: For an (N-input,N-output) network, if we connect the N inputs to the N outputs graphically, we get a closed network where every branch coming out from a node should enter into another node. Then the number of outputs of all nodes is equal to that of inputs of all nodes. A (2,1) summing node has two inputs and one output while a (1,2) branching node has one input and two outputs. We get:

Nb+2Ns=2Nb+Ns
Ns=Nb

end of proof.

Appendix B Several short length FIR algorithms.

F(2,2) algorithm with (3 multiplications, 6 additions), $\{\alpha_i\} = \{0,1,-1\}$.

(a8)
$$a_0 = x_0 \qquad b_0 = h_0$$
$$a_1 = x_0 + x_1 \qquad b_1 = (h_0 + h_1)/2$$
$$a_2 = x_0 - x_1 \qquad b_2 = (h_0 - h_1)/2$$

$$m_i = a_i b_i ; \qquad i = 0,1,2$$

$$y_0 = m_0 + z^{-2}(m_1 + m_2 - m_0)$$
$$y_1 = m_1 - m_2$$

F(2,2) algorithm with (3 multiplications, 6 additions), $\{\alpha_i\} = \{1,-1,\infty\}$.

(a9)
$$a_0 = x_0 + x_1 \qquad b_0 = (h_0 + h_1)/2$$
$$a_1 = x_0 - x_1 \qquad b_1 = (h_0 - h_1)/2$$
$$a_2 = x_1 \qquad b_2 = h_1$$

$$m_i = a_i b_i ; \qquad i = 0,1,2$$

$$y_0 = m_0 + m_1 - m_2 + z^{-2}m_2$$
$$y_1 = m_0 - m_1$$

F(5,5) algorithm with (12 multiplications, 40 additions). This algorithm is based on the approach in [12].

(a10)
$$c_0 = x_0 + x_3 \qquad g_0 = (h_0 + h_3)/2$$
$$c_1 = x_1 + x_4 \qquad g_1 = (h_1 + h_4)/2$$
$$c_2 = x_2 \qquad g_2 = h_2/2$$
$$c_3 = x_0 - x_3 \qquad g_3 = (h_3 - h_0)/2$$
$$c_4 = x_1 - x_4 \qquad g_4 = (h_4 - h_1)/2$$
$$c_5 = x_2 \qquad g_5 = -h_2/2$$

$$a_0 = c_0 + c_1 + c_2 \qquad b_0 = (g_0 + g_1 + g_2)/3$$
$$a_1 = c_0 - c_2 \qquad b_1 = g_0 - g_2$$
$$a_2 = c_1 - c_2 \qquad b_2 = g_1 - g_2$$
$$a_3 = a_1 + a_2 \qquad b_3 = (b_1 + b_2)/3$$

$a_4 = c_3 - c_4 + c_5$ $\quad b_4 = (g_3 - g_4 + g_5)/3$

$a_5 = c_3 - c_5$ $\quad b_5 = (-2g_3 - g_4 + g_5)/3$

$a_6 = c_3 + c_4$ $\quad b_6 = (g_3 + 2g_4 + g_5)/3$

$a_7 = c_4 + c_5$ $\quad b_7 = (g_3 - g_4 - 2g_5)/3$

$a_8 = x_0$ $\quad b_8 = h_0$

$a_9 = x_0$ $\quad b_9 = h_1$

$a_{10} = x_1$ $\quad b_{10} = h_0$

$a_{11} = x_4$ $\quad b_{11} = h_4$

$m_i = a_i\, b_i\,;$ $\quad i = 0,1,\ldots,11$

$u_0 = m_1 - m_3$

$u_1 = m_2 - m_3$

$d_0 = m_0 + u_0$

$d_1 = m_0 - u_0 - u_1$

$d_2 = m_0 + u_1$

$d_3 = m_4 - m_5 + m_7$

$d_4 = -m_4 + m_6 + m_7$

$d_5 = m_4 + m_5 + m_6$

$d_6 = m_9 + m_{10}$

$f_0 = d_2 - d_5$

$f_1 = d_1 - d_4$

$f_2 = d_0 - d_3$

$f_3 = d_2 + d_5$

$f_4 = d_1 + d_4$

$f_5 = d_0 + d_3$

$y_0 = m_8 + z^{-5} f_5$

$y_1 = d_6 + z^{-5} (f_0 - m_8)$

$y_2 = f_2 - m_{11} + z^{-5} (f_1 - d_6)$

$y_3 = f_3 + z^{-5} m_{11}$

$y_4 = f_4$

Table.1 Quality factors for several short-length algorithms.

Algorithms	Q
F(1,N)	1
F(2,2)	0.25
F(3,3)	0.3
F(5,5)	0.175

Table.2 Arithmetic complexity of an F(N,N) algorithm by short-length FIR and by FFT-based cyclic convolution.

	Short-length FIR				FFT-based FIR				Direct FIR	
N	M	m	A	a	M_{FFT}	m	A_{FFT}	a	m	a
2	3	1.5	4	2					2	1
3	6	2	10	3.3					3	2
4	9	2.25	20	5	15	3.75	43	10.75	4	3
5	12	2.4	40	8					5	4
6	18	3	42	7	33	5.5	83	13.83	6	5
8	27	3.38	76	9.5	43	5.38	131	16.38	8	7
9	36	4	90	10					9	8
10	36	3.6	128	12.8					10	9
12	54	4.5	150	12.5					12	11
15	72	4.8	240	16	112	7.47	353	23.53	15	14
16	81	5.06	260	16.25	115	7.19	355	22.19	16	15
18	108	6	306	17					18	17
20	108	5.4	400	20					20	19
24	162	6.75	498	20.75					24	23
25	144	5.76	680	27.2					25	24
27	216	8	630	23.33					27	26
30	216	7.2	744	24.8	225	7.50	829	27.63	30	29
32	243	7.59	844	26.38	291	9.09	899	28.09	32	31
36	324	9	990	27.5	271	7.53	1084	30.11	36	35
60	648	10.8	2280	38	455	7.58	1955	32.58	60	59
64	729	11.39	2660	41.56	701	10.95	2179	34.05	64	63
128	2187	17.09	8236	64.34	1667	13.02	5123	40.02	128	127
256	6561	25.63	25220	98.52	3483	13.61	11779	46.01	256	255
512	19683	38.44	76684	149.77	8707	17.01	26627	52.01	512	511
1024	59049	57.67	232100	226.66	19459	19.00	59395	58.00	1024	1023

Table.3 Minimum sum of operations (Mults + Adds). (mxn) means the decomposition by a fast F(m,m) algorithm using optimal ordering followed by direct length-n FIR filters. (FFT) means using FFT-based schemes.

N	Decomposition	M+A	/N	Block size
2	direct	6	3	1
3	direct	15	5	1
4	(2x2)	26	6.5	2
5	direct	45	9	1
6	(3x2)	56	9.33	3
8	(2x2x2)	94	11.75	4
9	(3x3)	120	13.33	3
10	(5x2)	152	15.2	5
12	(2x3x2)	192	16	6
15	(5x3)	300	20	5
16	(2x2x2x2)	314	19.63	8
18	(2x3x3)	396	22	6
20	(5x2x2)	472	23.6	10
24	(2x2x3x2)	624	26	12
25	(5x5)	740	29.6	5
27	(3x3x3)	810	30	9
30	(5x3x2)	912	30.4	10
32	(2x2x2x2x2)	1006	31.44	16
36	(2x2x3x3)	1260	35	12
60	(5x2x3x2)	2784	46.4	30
64	(FFT)	2880	45.00	64
128	(FFT)	6790	53.05	128
256	(FFT)	15262	59.62	256
512	(FFT)	35334	69.01	512
1024	(FFT)	78854	77.01	1024

Table.4 Minimum number of Macs (Length of scalar products + I/O adds). (mxn) means the decomposition by a fast F(m,m) algorithm using optimal ordering followed by direct length-n FIR filters. (FFT) means using FFT-based schemes.

N	Decomposition	MAC	/ N	Block size
2	direct	4	2	1
3	direct	9	3	1
4	direct	16	4	1
5	direct	25	5	1
6	direct	36	6	1
8	direct	64	8	1
9	direct	81	9	1
10	(2x5)	95	9.5	2
12	(2x6)	132	11	2
15	(3x5)	200	13.33	3
16	(2x8)	224	14	2
18	(2x9)	279	15.5	2
20	(2x2x5)	325	16.25	4
24	(2x2x6)	444	18.5	4
25	(5x5)	500	20	5
27	(3x9)	576	21.33	3
32	(2x2x8)	736	23	4
36	(2x2x9)	909	25.25	4
60	(5x2x6)	2064	34.4	10
64	(2x2x2x8)	2336	36.5	8
128	(2^4x8)	7264	56.75	16
256	(2^5x8)	22304	87.13	32
512	(2^6x8)	67936	132.69	64
1024	(2^7x8)	205856	201.03	128

References

[1] S. Winograd, "Arithmetic complexity of computations", CBMS-NSF Reginal Conf. Series in Applied Mathematics, SIAM Publications No. 33, 1980.

[2] Z.J. Mou, P. Duhamel, "Fast FIR filtering: algorithms and implementations", Signal Processing, Dec. 1987, pp.377-384.

[3] M. Vetterli, "Running FIR and IIR filtering using multirate filter banks", IEEE Trans. Acoust. Speech, Signal Processing, Vol. 36, N°5, May 1988, pp.730-738.

[4] H.K. Kwan, M.T. Tsim, "High speed 1-D FIR digital filtering architecture using polynomial convolution", in Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing, Dallas, USA, April, 1987, pp.1863-1866.

[5] P.P. Vaidyanathan, S.K. Mitra, "Polyphase networks, block digital filtering, LPTV systems and alias-free QMF banks: A unified approach based on pseudocirculants," IEEE Trans. Acoust. Speech, Signal Processing, Vol. 36, N°3, March 1988, pp.381-391.

[6] P. Duhamel, M. Vetterli, "Improved Fourier and Hartley transform algorithms: application to cyclic convolution of real data", IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-35, N°6, June 1988, pp.818-824.

[7] S. Winograd, "Some bilinear forms whose multiplicative complexity depends on the field of constants", Mathematical Systems Theory 10 (Sept. 1977), pp.169-180. Also in Number Theory in Digital Signal Processing (ed. by J.H. McClellan and C.M. Rader), Prentice-Hall, Englewood Cliffs, N.J., 1979.

[8] R.C. Agarwal and J.W. Cooley, "New algorithms for digital convolution", IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-25, Oct.1977, pp. 392-410.

[9] H.V. Sorensen, D.L. Jones, M.T. Heideman, C.S. Burrus, "Real-valued fast Fourier transform algorithms", IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-35, N°6, June 1988, pp.849-863.

[10] Z.J. Mou, P. Duhamel, "A unified approach to the fast FIR filering algorithms", in Proc. IEEE Int. Conf. Acoust. Speech, Signal Processing, New-York, USA, April 1988, pp.1914-1917.

[11] R.E. Blahut, Fast Algorithms for Digital Signal Processing, Addison-Wesley, Reading, MA,1985.

[12] P.C. Balla, A. Antoniou, S.D. Morgera, "Higher radix aperiodic convolution algorithms", IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-34, N°1, February 1986, pp.60-68.

[13] R.E.Crochiere, L.R.Rabiner, Multirate Digital Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1983.

Table captions:

Table.1 Quality factors for several short-length algorithms.

Table.2 Arithmetic complexity of an F(N,N) algorithm by short-length FIR and by FFT-based cyclic convolution.

Table.3 Minimum sum of operations (Mults + Adds). (mxn) means the decomposition by a fast F(m,m) algorithm using optimal ordering followed by direct length-n FIR filters. (FFT) means using FFT-based schemes.

Table.4 Minimum number of Macs (Length of scalar products + I/O adds). (mxn) means the decomposition by a fast F(m,m) algorithm using optimal ordering followed by direct length-n FIR filters. (FFT) means using FFT-based schemes.

Figure captions

Fig.1 FIR filtering based on algorithm (a1).
Fig.2 FIR filtering based on algorithm (a2).
Fig.3 FIR filtering based on algorithm (a3).
Fig.4 FIR filtering based on algorithm (a4).
Fig.5 FIR filtering based on algorithm (a5).
Fig.6 (a) initial system; (b) transposed system.

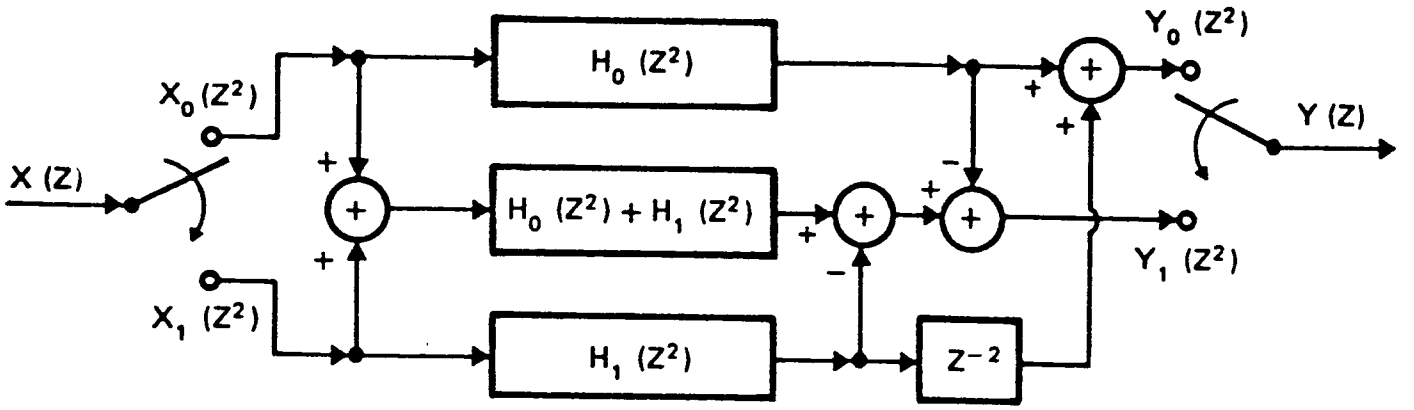


Figure 1

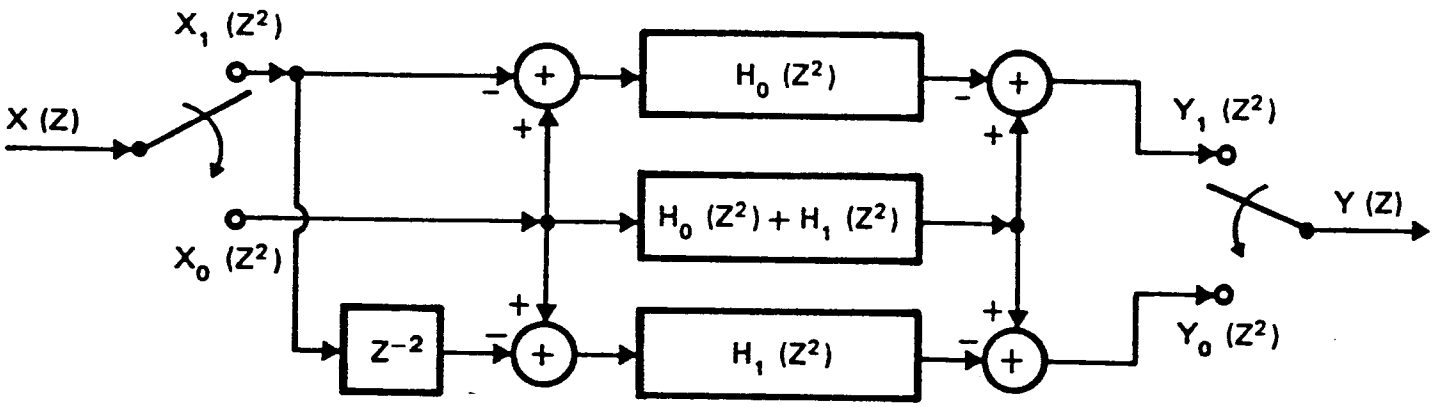


Figure 2

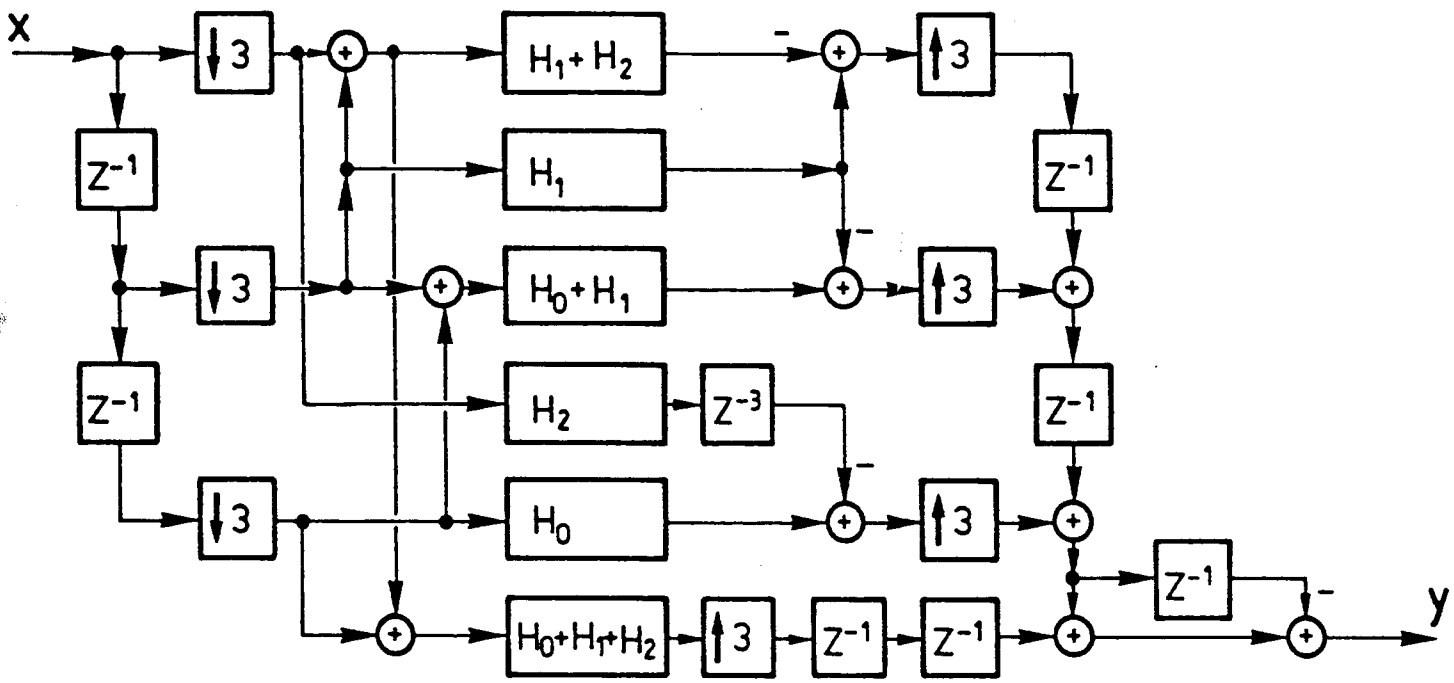


Fig. 3

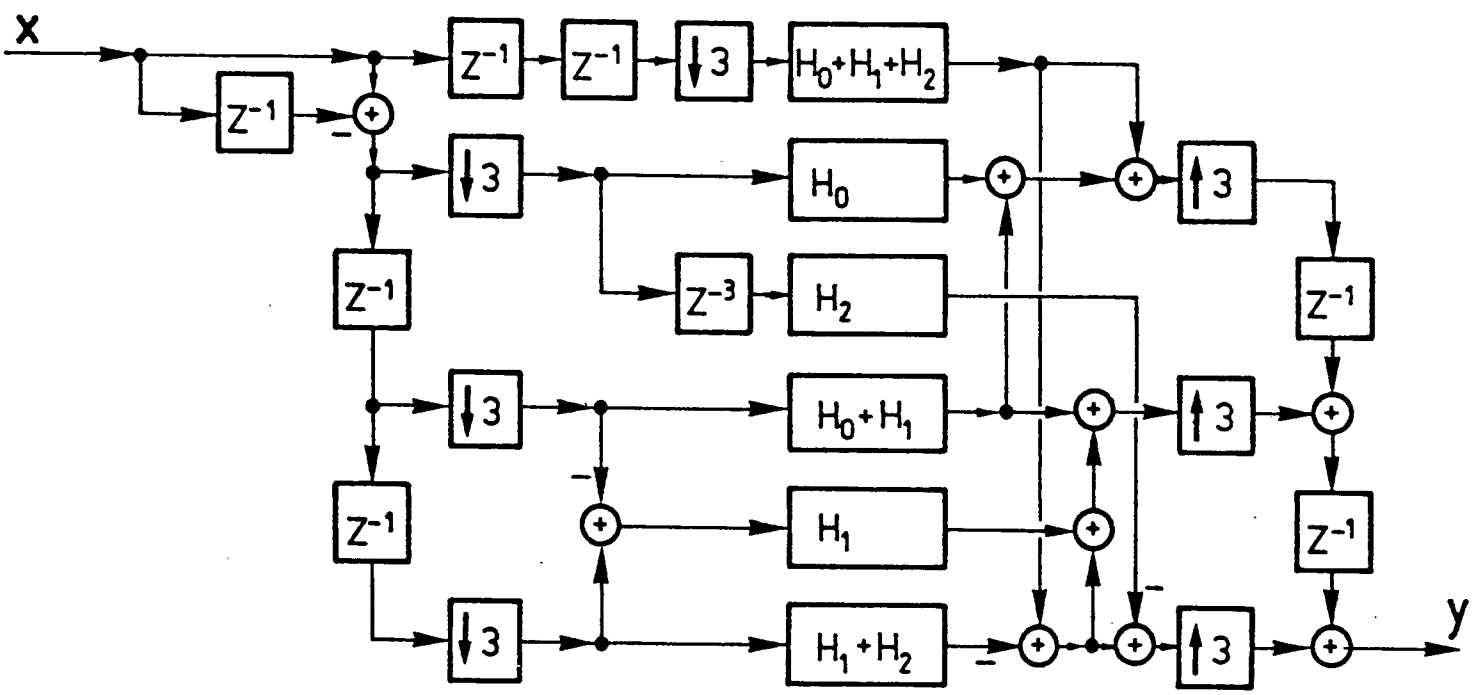

x
Z-1
Z-1
Z-1
Z-1
Z-1
Z-1
Z-3
3
3
3
3
3
3
H0+H1+H2
H0
H2
H0+H1
H1
H1+H2
3
3
3
Z-1
Z-1
Z-1
y

Fig 4

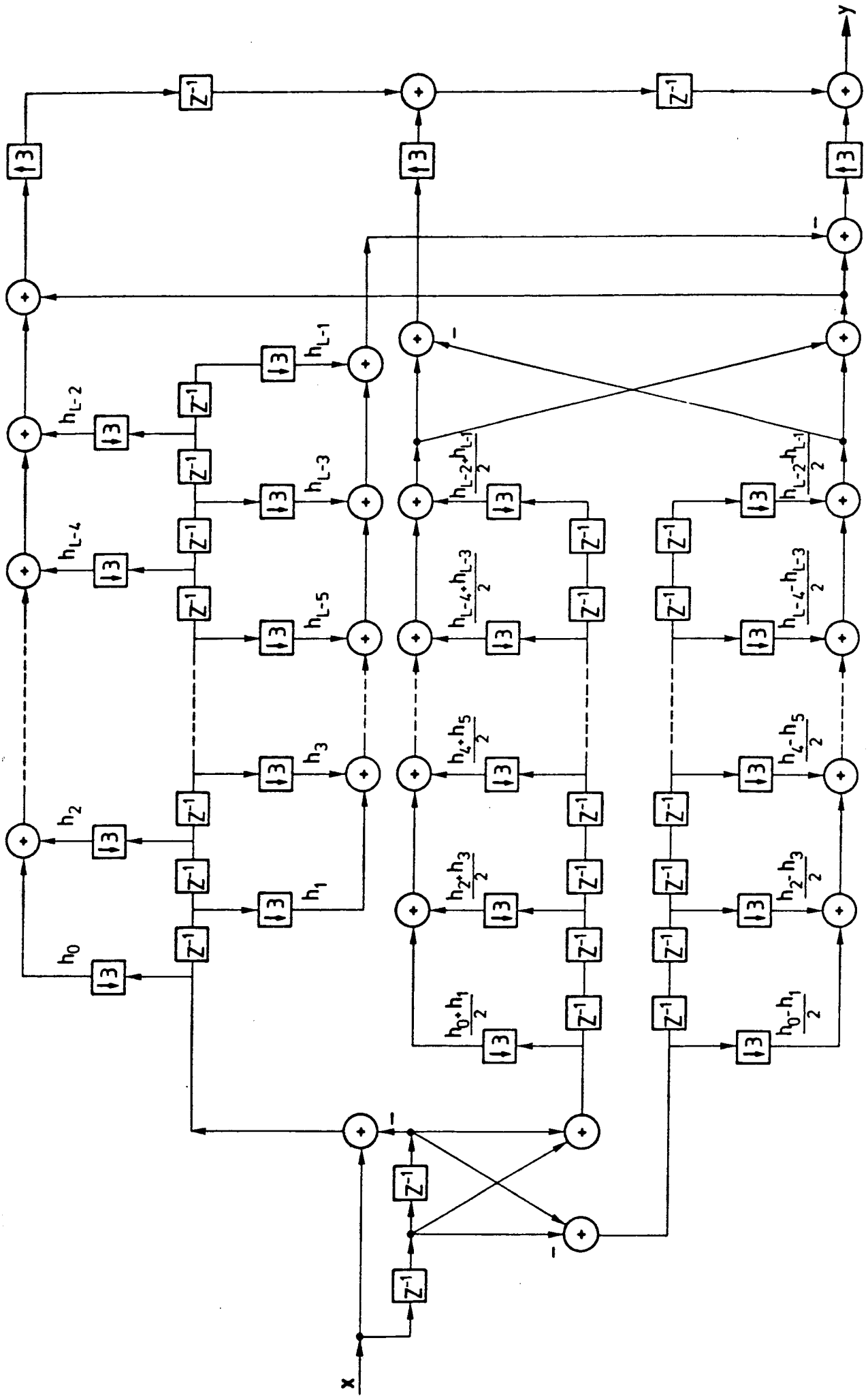

x
y
Z-1
↓3
h0
h1
h2
h3
hL-1
hL-2
hL-3
hL-4
hL-5
h0+h1/2
h0-h1/2
h2+h3/2
h2-h3/2
h4+h5/2
h4-h5/2
hL-4+hL-3/2
hL-4-hL-3/2
hL-2+hL-1/2
hL-2-hL-1/2
-

Fig 5

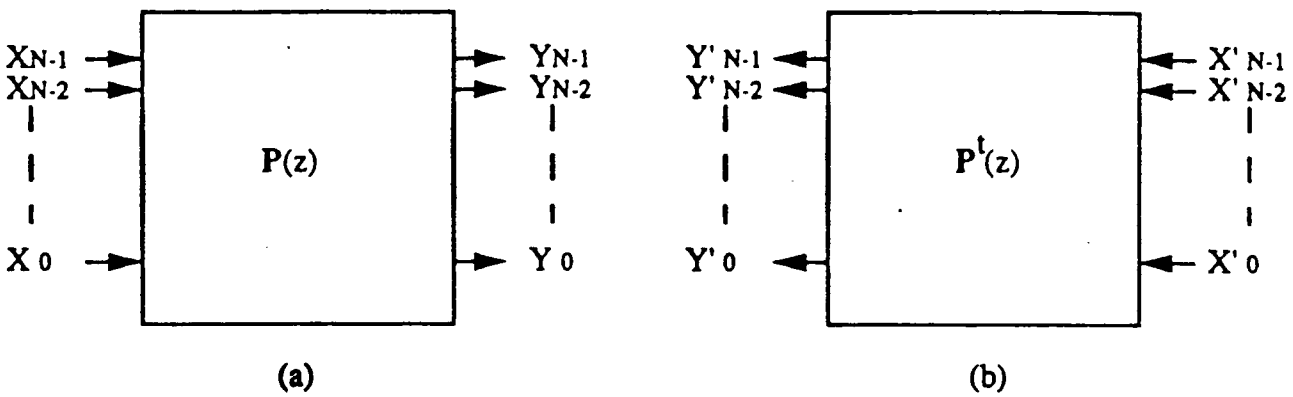


Fig.6 (a) initial system; (b) transposed system.

ARTICLE 2

Approximate Maximum Likelihood extension of MUSIC for correlated sources,
par H. CLERGEOT et S. TRESSENS

soumis en Août 1989 à IEEE Trans. ASSP

APPROXIMATE MAXIMUM LIKELIHOOD EXTENSION OF MUSIC FOR CORRELATED SOURCES

H. Clergeot * S. Tressens **

ABSTRACT

A new algorithm is introduced as an Approximate Maximum Likelihood Method (AMLM) . It also takes advantage of the EigenValue Decomposition (EVD) of the covariance matrix and can been seen as an improvement of MUSIC in presence of correlated sources. Using previous results on the statistical perturbation of the covariance matrix the theoretical statistical performances of both estimators are derived and compared to the Cramer Rao Bounds (CRB) . In presence of correlated sources MUSIC is no longer efficient while our algorithm achieve the CRB, down to a lower threshold that is carefully quantified.

We then derive a fast computation efficient Newton algorithm to find the solution of the AMLM and we perform an extensive analysis of the convergence behaviour and of the practical performance by Monte Carlo simulation.

The theoretical analysis is confirmed by the simulation results. At low SNR our algorithm proves to achieve far better resolution properties than MUSIC . We introduce a significant SNR "cross over level", for which the theoretical standard deviation is equal to the source separation . We demonstrate that the AMLM provides unbiased estimates and fits the theoretical variance down the cross over, while MUSIC loosses resolution at a significantly greater SNR (by up to 20 dB or more) .

Another attractive feature of the algorithm is its ability to operate in presence of rank defficient sample covariance matrix for the sources. In particular it is effective with a small number of snapshots, even a single one.

* LESIR/ENSC ,61 Avenue Pt Wilson, 94230 Cachan, FRANCE
** CRPE/CNET ,38 rue du Général Leclerc , 92131 Issy-Les-Moulineaux,FRANCE

APPROXIMATE MAXIMUM LIKELIHOOD EXTENSION OF MUSIC FOR CORRELATED SOURCES

I INTRODUCTION

High resolution spectral methods are widely used in array processing for source localization. One of the most popular is the MUSIC method for its ability to locate sources with an arbitrary array configuration. The MUSIC method is based on the EigenValue Decomposition (EVD) of the covariance matrix [1],[2],[3] . But it is known that its performances are severely degraded in the presence of correlated sources, and that in the particular situation of unitary source correlation the localization fails [4] [5] [6]. In the case of localization with a linear array of equispaced elements, the problem can be solved by the use of spatial smoothing [7] [8], but this method cannot be applied to an arbitrary array geometry .

Another interesting approach can be obtained by the Maximum Likelihood Criterion . Under very general conditions, above some Signal-to-Noise Ratio (SNR) threshold, this method achieves the Cramer Rao Bounds (CRB) for the estimation variance. Several maximization algorithms have been proposed, but they involve complex non linear iteration, and are very sensitive to initialization [9] [10] .

The aims of this article are first to present an algorithm [5] [11] [12]derived as an Approximate Maximum Likelihood Method (AMLM), and then to give its comparison with the MUSIC method. For this purppose the analytical expressions of the variance are calculated for both methods and related to the CRB. Note that since our first publication [11 , 12] a similar form has been suggested in [6], appendix .

The proposed method achieves the CRB, at high SNR, even in the case of unitary correlation between sources. It turns out that this AMLM method reduces to MUSIC in the case of decorrelated sources . This is coherent with the fact that MUSIC is optimal in this particular situation [3] [4][6][11][12][13] .

To derive the AMLM method , a reduction of the Log-likelihood function is introduced . The same approach was used by the authors to derive a simple form of the CRB

[13].

In the case of M sources, the solution for the AMLM requires the minimization of an M variables function . But compared to the exact Maximum Likelihood method, our algorithm takes advantage of a very simple form of the gradient and Hessian,which enables fast convergence with a Newton-like method . In its initial formulation , the method relies on an estimate of the covariance matrix of the source powers. A variant taking account of a recursive update of this matrix is proposed, and proves to have superior convergence properties and better statistical behaviour at low SNR . Assuming proper initialization, it remains effective if the signal matrix is rank deficient or for a small number of snapshots (even a single one).

In section II the signal model, the basis of EVD methods and MUSIC are recalled. In section III, the reduced log-likelihood and the AMLM method are presented . Section IV is a recall of the derivation of the CRB and of the statistical perturbation on the covariance matrix . These results are used in section V for the theoretical variance calculation, for MUSIC and AMLM . The minimization algorithm is presented in section VI,together with a demonstration of its fast convergence in simulation . Statistical performances are checked on Monte Carlo simulation (section VII) .

II SIGNAL MODEL

II A- SIGNAL MODEL

For an arbitrary propagation model and array geometry in the presence of additive noise, the signal on an N element array resulting from M sources may be represented by an N vector $\underline{X}(q)$. We assume that we have Q observations (snapshots). For the snapshot q, the observation vector can be written as:

$$\underline{X}(q) \triangleq [X_1(q).....X_N(q)]^t \quad , M < N \tag{1}$$

$$\underline{X}(q) = \sum_{m=1}^{M} g_m(q)\, \underline{S}_m + \underline{B}\,(q)$$

$$\|\underline{S}_m\|^2 = 1$$

where g_m is an amplitude parameter associated to source m, and $\underline{S}_m$ is the corresponding "source vector" which may be interpreted as the transfer function between source m and each array element. For a given propagation model and array geometry, $\underline{S}_m = \underline{S}\,(\theta)$ is a known vector function of a position parameter θ . $\underline{B}\,(q)$ is an additive complex gaussian noise . We assume that the covariance matrix of $\underline{B}$ is :

$$R_B = E[\,\underline{B}\ \underline{B}^{\dagger}\,] = \sigma^2 \mathbf{I} \tag{2}$$
$$E[\,\underline{B}\ \underline{B}^{t}\,] = 0$$

The noise vectors are assumed independent from one snapshot to the other. The amplitudes g_m are considered as unknown deterministic parameters. Calculation will be made for an arbitrary source model $\underline{S}\,(\theta)$. The special case of a linear array will be considered in the last section, for simulations .For generality, it will be assumed that the source vectors and the vector function $\underline{S}\,(\theta)$ are normalized to one : $\|\underline{S}\,(\theta)\|^2 = 1$.

From (1) the vector $\underline{X}(q)$ may be written as:

$$\underline{X}(q)=\underline{Y}(q)+\underline{B}(q) \tag{3}$$

and the vector noiseless observation $\underline{Y}(q)$ as :

$$\underline{Y}(q)=\sum_{m=1}^{M} g_m(q)\,\underline{S}_m = \mathbf{S}\,\underline{G}(q) \tag{4}$$

where $\mathbf{S}$ and $\underline{G}(q)$ are defined as :

$$\mathbf{S}=[\underline{S}_1\ \underline{S}_2 \ldots \underline{S}_M] \quad ; \quad M<N$$
$$\underline{G}(q)= [\, g_1(q) \ldots\ldots g_M(q)]^t \tag{5}$$

A classical identifiability condition is that matrix $\mathbf{S}$ is rank M . This is assumed in what follows. From (4) , $\underline{Y}(q)$ lies in the M-dimensional subspace ξ_M, spanned by the source vectors $\underline{S}_1$, $\underline{S}_2 \ldots, \underline{S}_M$, called source subspace . Conversely, if there is no deterministic relation between the amplitudes $g_m(q)$ and if Q>M, the vector $\underline{Y}(q)$ will span all the manifold ξ_M . The complementary subspace will be named "noise subspace" ,ξ_B.

The projectors on ξ_S and ξ_B will be noted Π_S and Π_B with $\Pi_S + \Pi_B=\mathbf{I}$, the identity matrix. The projector Π_S may be written in terms of the source matrix according to :

$$\Pi_S = \mathbf{I} - \Pi_B = \mathbf{S}\,(\,\mathbf{S}^{\dagger}\,\mathbf{S}\,)^{-1}\mathbf{S}^{\dagger} \tag{6}$$

which is a known function of the position parameters $\{\theta_m\}$. Invertibility of $(S^\dagger S)$ is guaranteed by the fact that S is rank M.

The sample covariance matrices of $\underline{X}(q)$ and $\underline{Y}(q)$ are respectively :

$$\hat{R}_x = \frac{1}{Q}\sum_{q=1}^{Q} \underline{X}(q)\underline{X}^\dagger(q) \qquad (7)$$

$$R_y = \frac{1}{Q}\sum_{q=1}^{Q}\underline{Y}(q)\underline{Y}(q)^\dagger$$

The additive noise only is considered as random . The expectation of matrix $\hat{R}_X$ is, according to (2):

$$R_x = E\left[\hat{R}_x\right] = R_y + R_B = R_y + \sigma^2 I \qquad (8)$$

Using (3) and (4) , the sample covariance matrix R_y of the signal may be written:

$$R_y = S\ P\ S^\dagger \qquad (9)$$

$$P \triangleq \frac{1}{Q}\sum_{q=1}^{Q}\underline{G}(q)\underline{G}(q)^\dagger$$

The structure of matrix **P** plays an important part in the discussion of optimality . The diagonal elements are the average power p_m of the sources,while the off-diagonal elements are the sample cross correlations . They may be normalized by the powers p_m to introduce the coherence coefficients between sources ρ_{mn} . Let :

$$P_{mm} = p_m = \frac{1}{Q}\sum_{q=1}^{Q}|g_m(q)|^2$$

$$\rho_{mn} \triangleq \frac{1}{Q}\sum_{i=1}^{Q}(g_m g_n^*)(P_m P_n)^{-1/2} \qquad (10)$$

Matrix **P** can be factorized in terms of the source powers, and the coherence matrix ρ which reflects the intercorrelation between sources only:

$$\mathbf{P}=\mathbf{P_d}^{1/2}\,\rho\,\mathbf{P_d}^{1/2}$$

$$\mathbf{P_d}= \text{diag}\,(p_1 \ldots p_M) \qquad (11)$$

It is important to note that we have used only ensemble average on q for all definitions concerning the second order properties of the amplitudes. The results is that our further derivations will be valid in the small sample case,even for Q=1. This is not so in the most commonly used approach, in which the amplitudes are considered as random with given covariance. The results concerning the CRB on the variance would then be valid only in the asymptotic case $Q \dashrightarrow \infty$.

Another very common hypothesis is that **P** is diagonal. According to the previous remarks,in the random case ,this will be true not only if the amplitudes are decorrelated, but also if Q is large enough (so that the sample average is close to the expectation) . In such an asymptotic case matrix ρ becomes equal to the identity matrix.

On the other hand, in (11) matrix ρ would be singular whenever there exists some deterministic relation between the source amplitudes .

II B- BASIS OF EVD METHODS

These methods consits in first estimating the signal subspace ξ_S by EVD and then estimating the positions from ξ_S or ξ_B, according to the source vector model $\underline{S}(\theta)$.

The estimation of ξ_S is related to the fact that in general , for Q>M, the manifold $\{ \underline{Y}(q) \}$ spans the whole subspace ξ_S [7,8]. Then matrix $\mathbf{R}_y$ is rank M, i.e., it has M non zero eigenvalues, and the corresponding eigenvectors constitute an orthonormal basis for ξ_S . The N-M other eigenvectors, corresponding to the degenerate zero eigenvalue, constitute an orthonormal basis for ξ_B.

In the presence of white noise, according to (8), $\mathbf{R}_x$ and $\mathbf{R}_y$ have the same eigenvectors. The eigenvalues of $\mathbf{R}_x$ are raised by σ^2 . The signal eigenvectors are identified as those corresponding to the M greatest eigenvalues.

The eigenvalues will be denoted by l_i for $\mathbf{R}_x$ and λ_i for $\mathbf{R}_y$, and are assumed to be in order of non increasing values.The eigenvectors are denoted by $\underline{U}_i$ and the projectors on ξ_S and ξ_B by Π_S and Π_B . We obtain the following relations:

$$\mathbf{R}_x = \sum_{i=1}^{N} l_i \underline{U}_i \underline{U}_i^{\dagger} \quad ; \mathbf{R}_y = \sum_{i=1}^{N} \lambda_i \underline{U}_i \underline{U}_i^{\dagger} \quad ; \quad l_i = \lambda_i + \sigma^2$$

$$\Pi_S = \sum_{i=1}^{M} \underline{U}_i \underline{U}_i^{\dagger} \quad ; \Pi_B = \sum_{i=M+1}^{N} \underline{U}_i \underline{U}_i^{\dagger} \qquad (12)$$

The position estimation is based on the orthogonality of any vector $\underline{A}$ of ξ_B to the signal subspace , and in particular to all source vectors:

$$\forall\, m \in [1, M], \qquad \forall \ \underline{A} \in \xi_B, \qquad \underline{S}_m^{\dagger} . \underline{A} = \underline{S}^{\dagger}(\theta_m) . \underline{A} = 0 \qquad (13)$$

We may now introduce the following discriminating function:

$$f_A(\theta)=|\underline{S}(\theta)^{\dagger}\cdot\underline{A}|^2 \qquad (14)$$

Relation (13) states that the source locations θ_m are zeros of the discriminating function $f_A(\theta)$.

II C- MUSIC ALGORITHM

Any eigenvector $\underline{U}_i$ of $\mathbf{R}_X$ for i >M belongs to the noise subspace and may be used as the vector $\underline{A}$ in(14) . Robustness with respect to the noise and spurious zeros in $f_A(\theta)$ is obtained by the use of an average on all these vectors[1,2].The corresponding function is:

$$f(\theta) = \sum_{i=1+M}^{N} \left|\underline{S}^{\dagger}(\theta)\underline{U}_i\right|^2 \qquad (15)$$

According to (12) this function may be written in terms of the projector Π_B on the noise eigenspace:

$$f(\theta)= \underline{S}(\theta)^{\dagger}\ \Pi_B\ \underline{S}(\theta) \qquad (16)$$

For actual data Π_B is replaced by its estimate $\hat{\Pi}_B$ given by EVD of $\hat{\mathbf{R}}_X$ according to relations (12).

The "MUSIC spectrum" is the plot of $f(\theta)^{-1}$where the positions of the sources, due to inversion, appear as sharp peaks corresponding to the minima of $f(\theta)$.

II D - SOURCE AMPLITUDE ESTIMATION

The sample covariance **P** of the source amplitude, is easily calculated by the inversion of relation (9):

$$\mathbf{P} = (\mathbf{S}^{\dagger}\mathbf{S})^{-1}\mathbf{S}^{\dagger}\mathbf{R}_y\mathbf{S}(\mathbf{S}^{\dagger}\mathbf{S})^{-1}$$

or:

$$\mathbf{P}^{-1} = \mathbf{S}^{\dagger}\mathbf{R}^{\#}_y\mathbf{S} \qquad (17)$$

The second shorter expression is valid only if $\mathbf{R}_y$ is rank M . Matrix $\mathbf{R}^{\#}_y$, pseudo inverse of $\mathbf{R}_y$, is obtained from the EVD of $\mathbf{R}_x$ or $\mathbf{R}_y$ using :

$$\mathbf{R}^{\#}_y = \sum_{i=1}^{M} \frac{1}{\lambda_i} \underline{U}_i \underline{U}^{\dagger}_i = \sum_{i=1}^{M} \frac{1}{l_i - \sigma^2} \underline{U}_i \underline{U}^{\dagger}_i \qquad (18)$$

For the evaluation of **P** from actual data, S, λ_i , U_i, σ^2 are replaced by proper estimates . The estimates of $\underline{U}_i$ and λ_i are obtained by EVD of $\hat{\mathbf{R}}_x$. The estimate of S is deduced, according to the sources model $\underline{S}(\theta)$, from the position estimation given by MUSIC (or any other position estimation method) . The noise variance estimate is obtained by an average over the noise subspace eigenvalues :

$$\hat{\sigma}^2 = \frac{1}{N - M} \sum_{i=M+1}^{N} \hat{l}_i \qquad (19)$$

III THE MAXIMUM LIKELIHOOD APPROACH

III A - THE LOG-LIKELIHOOD AND ITS REDUCTION

For given unknown values of $\{ g_m(q), \sigma^2, \theta_m \}$, assuming a gaussian additive complex white noise, the probability density of the observation is easily found to be given by:

$$\text{Log}[p(X(q)] = -N\text{Log}(2\pi\sigma^2) - \frac{1}{\sigma^2}\sum_{q=1}^{Q} \|X(q) - S\underline{G}(q)\|^2 \qquad (20)$$

where S and $\underline{G}(q)$ are defined in (5) as a function of $\{ g_m(q), \theta_m \}$

Given an observation X(q),candidates $\underline{\hat{G}}(q), \underline{\hat{\theta}}, \hat{\sigma}$, are tried for the exact unknown values of the parameters $\underline{G}(q)$, $\underline{\theta}$, σ by substitution in the expression (20). We obtain the likelihood function:

$$V_{X(q)}(\underline{\hat{G}}(q), \underline{\hat{\theta}}, \hat{\sigma}) = -NQ\text{Log}(2\pi\hat{\sigma}^2) - \frac{1}{\hat{\sigma}^2}\sum_{q=1}^{Q} \|X(q) - \hat{S}\,\underline{\hat{G}}(q)\|^2 \qquad (21)$$

$$\underline{\hat{\theta}} \triangleq [\hat{\theta}_1, \ldots, \hat{\theta}_M]^t$$

where the candidate values of parameters appear in $\hat{S}$ and $\underline{\hat{G}}(q)$. The Maximum Likelihood estimates $\underline{G}_{ML}(q), \underline{\theta}_{ML}, \sigma_{ML}$, are selected by maximization of this function.

A first reduction of the likelihood is provided by a separate maximization with

respect to the amplitudes [12]. For a given value of matrix $\overset{\Delta}{S}$ it is easily found that the corresponding signal estimate $\underline{Y}_{ML}(q)=\overset{\Delta}{S}\ G_{ML}(q)$ is the projection of $\underline{X}(q)$ on ξ_S:

$$\underline{Y}_{ML}(q) = \overset{\Delta}{S}\,\underline{G}_{ML}(q) = \overset{\Delta}{\Pi}_S\,\underline{X}(q) \qquad (22)$$

$\overset{\Delta}{\Pi}_S$ is the projector on the subspace ξ_S, function of the source position parameter $\overset{\Delta}{\underline{\theta}}$. Then :

$$\left\|\underline{X}(q) - \overset{\Delta}{S}\,\underline{G}_{ML}(q)\right\| = \left\|(I - \overset{\Delta}{\Pi}_S)\underline{X}(q)\right\| = \left\|\overset{\Delta}{\Pi}_B\,\underline{X}(q)\right\| \qquad (23)$$

Substitution of these values in (21) gives the reduced expression of the Log-likelihood, function of $\overset{\Delta}{\underline{\theta}}$ and $\overset{\Delta}{\sigma}$ only :

$$V_2(\overset{\Delta}{\underline{\theta}}, \overset{\Delta}{\sigma}) = -NQLog(2\pi\,\overset{\Delta}{\sigma}^2) - \frac{Q}{\overset{\Delta}{\sigma}^2}Tr(\overset{\Delta}{\Pi}_B\,\hat{R}_x)$$

$$\hat{R}_x = \frac{1}{Q}\sum_{q=1}^{Q}\underline{X}(q)\underline{X}^{\dagger}(q) \qquad (24)$$

where $\overset{\Delta}{\Pi}_B$ is parametrized in function of $\overset{\Delta}{\underline{\theta}}$.

The exact ML solution $\underline{\theta}_{ML}$ for the source position would then be obtained by the maximization of (24).

Note that (24) has a direct interpretation in terms of an alternative model,which is of some interest for later discussion. Obviously no separation can be made between the signal and the projection of the noise on the signal subspace, so that we may consider the sum $\underline{Y}+\Pi_S\underline{B}$ as the unknown signal to be estimated, with an additive noise $\Pi_B\underline{B}$. The corresponding Log-likelihood is similar to (24) (N being replaced by N-M in the first term). Strictly speaking the CRB, as derived in section IV A, is valid for this model .

III B- THE AML

Let us consider the expectation of the Log-likelihood function $V_2(\underline{\theta},\sigma)$:

$$E[V_2(\underline{\hat{\theta}},\hat{\sigma})] = -NQLog(\pi\hat{\sigma}^2) - \frac{Q}{\hat{\sigma}^2}Tr\left(\hat{\Pi}_B R_x\right) =$$

$$-NQLog(\pi\hat{\sigma}^2) - \frac{Q}{\hat{\sigma}^2}Tr\left(\hat{\Pi}_B (R_y + \sigma^2 I)\right) \qquad (25)$$

According to (9)

$$E[V_2(\underline{\hat{\theta}},\hat{\sigma})] = -NQLog(\pi\hat{\sigma}^2) - \left(\frac{\sigma}{\hat{\sigma}}\right)^2 Q(N-M) - \frac{Q}{\hat{\sigma}^2}(Tr\,\hat{\Pi}_B S P S^{\dagger}) \qquad (26)$$

It is easily verified that for $\hat{\underline{\theta}}$ close to $\underline{\theta}$ the difference between $S^{\dagger}\hat{\Pi}_B S$ and $\hat{S}^{\dagger}\hat{\Pi}_B\hat{S}$ is the order of $\|\hat{\underline{\theta}}-\underline{\theta}\|^3$ [13] . We then obtain the approximation:

$$E[V_2(\underline{\hat{\theta}},\hat{\sigma})] = -NQLog(\pi\hat{\sigma}^2) - \left(\frac{\sigma}{\hat{\sigma}}\right)^2 Q(N-M) - \frac{Q}{\hat{\sigma}^2}Tr(\hat{S}^{\dagger}\hat{\Pi}_B\hat{S}P) \qquad (27)$$

$$+ O\left(\|\theta - \hat{\theta}\|^3\right)$$

This suggests the expression of an approximate MLM, by making the same transformation in the likelihood (24) as was made in the expectation in order to derive (27).

According to (24) the exact ML solution for $\underline{\theta}$ is obtained by the minimization of

$\mathrm{Tr}[\hat{\Pi}_B \hat{R}_X]$. Then, by induction from (27) the proposed criterion [10,11] is to minimize the expression :

$$f(\underline{\theta}) = \mathrm{Tr}\left(\underline{S}^{\dagger} \hat{\Pi}_B \underline{S}\, P\right)$$
$$\cong \mathrm{Tr}\left(\underline{\Pi}_B \hat{R}_x\right) - (N - M)\,\hat{\sigma}^2 \qquad (28)$$

where $\hat{\Pi}_B$ is the projector estimated from the EVD of $\hat{R}_X$.

The statistical properties of this estimator will be derived in section V , in which we demonstrate that it achieves the CR bounds at high SNR, even in the presence of correlated sources.Compared to the exact MLM, the advantage is that expression (28) is quadratic with respect to the source vectors:

$$f(\underline{\theta}) = \sum_{m,n=1}^{M} P_{nm} \underline{S}^{\dagger}(\underline{\theta}_m)\, \underline{\Pi}_B\, \underline{S}(\underline{\theta}_n) \qquad (29)$$

This allows the easy implementation of the Newton Gauss algorithm in order to find the minimum, assuming a correct starting value $\underline{\theta}_0$. In practice, matrix **P** will be replaced by an estimate of the amplitude covariance matrix updated recursively according to (17) (see sectionVI for a discussion of convergence).

Relation (29) appears similar to a generalization of MUSIC . In fact in the special case when **P** is diagonal it can be easily seen that the minimization gives the solution of MUSIC .

IV CRAMER RAO BOUNDS AND STATISTICAL PERTURBATION ON $\hat{R}_X$

With reference to a previous study [13], we recall some theoretical results concerning the CRB and the statistical perturbation of $\mathbf{R}_X$ which will be used in the next paragraph to discuss the optimality of our new method, and its connection with MUSIC.

IV A - CRAMER RAO BOUNDS

The same reduced likelihood expectation (27) was used in [5,13] to derive an expression for the CRB.

The matrix bound [CRB] is given by the inverse of the Fisher information matrix [9]

$$[CRB]^{-1}_{m,n} = - \frac{\partial^2}{\partial \overset{\Delta}{\theta}_m \partial \overset{\Delta}{\theta}_n} E\left[V_2(\underline{\overset{\Delta}{\theta}}, \underline{\overset{\Delta}{\sigma}}) \right]_{\underline{\overset{\Delta}{\theta}}=\underline{\theta}, \overset{\Delta}{\sigma}=\sigma} \qquad (30)$$

It was demonstrated in [13] and it may be easily verified from (27) that :

$$[CRB]^{-1}_{m,n} = \frac{Q}{\sigma^2} \left\{ 2 \, \mathrm{Re\,al} \left[\underline{\dot{S}}^{\dagger}_m \, \Pi_B \, \underline{\dot{S}}_n \, P_{nm} \right] \right\} \qquad (31)$$

where $\underline{\dot{S}}_m$ denotes the derivative of the vector function $\underline{S}(\theta)$ for $\theta=\theta_m$:

$$\dot{S}_m = \left[\frac{\partial \underline{S}(\theta)}{\partial(\theta)} \right]_{\theta = \theta_m} \qquad (32)$$

The CRB can be written in terms of a "modified coherence matrix". For this purpose the diagonal elements in (31) will be normalized to one, by the use of two diagonal matrices, the matrix $\mathbf{P}_d$ of the source power (11) and the following sensitivity matrix:

$$W = \mathrm{diag}\left[\left(\dot{\underline{S}}_1^{\dagger}\Pi_B\dot{\underline{S}}_1\right),\ldots,\left(\dot{\underline{S}}_M^{\dagger}\Pi_B\dot{\underline{S}}_M\right)\right] \tag{33}$$

Using this definition in (31) the matrix $[CRB^{-1}]$ may then be written :

$$[CRB]^{-1} = \frac{2Q}{\sigma^2}(P_d W)^{1/2}\rho_{CR}\ (P_d W)^{1/2} \tag{34}$$

the normalized matrix ρ_{CR} being defined by :

$$[\rho_{CR}]_{mn} = |\rho_{mn}|\cos(\propto_{mn})\cos(\beta_{mn}) \tag{35}$$

where :

$$\cos(\beta_{mn}) = \frac{\left|\dot{\underline{S}}_m^{\dagger}\Pi_B\dot{\underline{S}}_n\right|}{[\dot{\underline{S}}_m^{\dagger}\Pi_B\dot{\underline{S}}_m]^{1/2}[\dot{\underline{S}}_n^{\dagger}\Pi_B\dot{\underline{S}}_n]^{1/2}} \tag{36}$$

which may be interpreted as the angle cosine between vectors $\Pi_B\dot{\underline{S}}m$ and $\Pi_B\dot{\underline{S}}_n$. The angle α_{mn} in (35) is defined by :

$$\propto_{mn} = \arg(\dot{\underline{S}}_m^{\dagger}\Pi_B\dot{\underline{S}}_n P_{nm}) = \arg(\dot{\underline{S}}_m^{\dagger}\Pi_B\dot{\underline{S}}_n) + \arg(P_{nm}) \tag{37}$$

It is then directly connected to the average phase difference between sources m and n, given by the argument of P_{mn}.

By inversion of (34) , on the diagonal, we obtain the expression of the bound on the frequency variance:

$$E\left[\left(\hat{\theta}_m - \theta_m\right)^2\right]_{CR} \leq \frac{\sigma^2}{2Qp_m\left(\dot{\underline{S}}_m^{\dagger}\Pi_B\dot{\underline{S}}_m\right)}\left[\rho_{CR}^{-1}\right]_{mm} \qquad (38)$$

Note that the diagonal elements of ρ and ρ_{CR} are equal to one. For decorrelated sources, ρ and ρ_{CR} are the identity matrices. Matrix ρ_{CR} is always "closer" than ρ to the identity, due to the weighting in (35) by factors less than one, which provide a "pseudo-decorrelation" . In particular, ρ_{CR} may be invertible even if ρ is singular .

IV B - STATISTICAL PERTURBATION ON THE SOURCE SPACE ESTIMATE

Expressions for the eigenvalue and eigenvectors variances in the presence of additive noise are classical [14] . To characterize the perturbation of the signal subspace,our method is to consider only the component $\delta\underline{U}_{i\perp}$ of the perturbation of the signal eigenvector orthogonal to ξs [5, 13].

Using a first order perturbation we found that if $\underline{U}_i$ is eigenvector of $\mathbf{R}_X$, $\underline{U}_i + \delta\underline{U}_{i\perp}$ is eigenvector of $\hat{\mathbf{R}}_X$ with :

$$\begin{aligned} \delta\underline{U}_{i\perp} &= \mathcal{P}\underline{U}_i \\ \mathcal{P} &= \Pi_B\left\langle \underline{B}\,\underline{X}^{\dagger}\right\rangle R_y^{\#} \end{aligned} \qquad (39)$$

$$\left\langle \underline{B}\,\underline{X}^{\dagger}\right\rangle = \frac{1}{Q}\sum_{q=1}^{Q}\underline{B}(q)\underline{X}^{\dagger}(q)$$

in which $\mathbf{R}_y^{\#}$ denotes the pseudo inverse of $\mathbf{R}_y$.

Relation (39)being valid for any vector of the basis $\underline{U}_1 \ldots \underline{U}_M$ of ξ_S to any vector $\underline{S} \in \xi_S$ we may associate the vector $\underline{S} + \delta\underline{S}_{\perp}$ of ξ_S with :

$$\delta\underline{S}_{\perp} = \mathcal{P}S \qquad (40)$$

This expression will arise in the derivation of the variance of the usual estimators. The covariance for the random vectors $\delta \underline{S}_{m\perp}$ has been calculated in [13] in a more general context . The derivation in the simple case when vectors $\underline{B}(q)$ are decorrelated is recalled in the appendix A . From (A.9) :

$$E\left(\delta \underline{S}_{m\perp} \delta \underline{S}_{n\perp}^{\dagger}\right) \equiv \frac{\sigma^2}{Q}\left[\underline{S}_n^{\dagger}\left\{\mathbf{R}_y^{\#}(\mathbf{S}\mathbf{P}\mathbf{S}^{\dagger} + \sigma^2 \mathbf{I})\,\mathbf{R}_y^{\#}\right\}\underline{S}_m\right]\Pi_B \qquad (41)$$

or according to (9) :

$$E\left[\delta \underline{S}_{m\perp} \delta \underline{S}_{n\perp}^{\dagger}\right] \equiv \frac{\sigma^2}{Q}\left[\underline{S}_n^{\dagger}\,(\mathbf{R}_y^{\#} + \sigma^2\,\mathbf{R}_y^{\#\,2})\,\underline{S}_m\right]\Pi_B \qquad (42)$$

Assuming that matrix $\mathbf{P}$ is regular , according to (17):

$$\mathbf{S}^{\dagger}\mathbf{R}_y^{\#}\mathbf{S} = \mathbf{P}^{-1}$$

$$\mathbf{S}^{\dagger}\mathbf{R}_y^{\#\,2}\mathbf{S} = (\mathbf{P}\mathbf{S}^{\dagger}\mathbf{S}\mathbf{P})^{-1} \qquad (43)$$

Then (42) can be written as :

$$E\left[\delta \underline{S}_{m\perp} \delta \underline{S}_{n\perp}^{\dagger}\right] \equiv \frac{\sigma^2}{Q}\left\{\left[\mathbf{P}^{-1}\right]_{mn} + \sigma^2\left[\mathbf{P}^{-1}(\mathbf{S}^{\dagger}\mathbf{S})^{-1}\mathbf{P}^{-1}\right]_{mn}\right\}\Pi_B \qquad (44)$$

In this relation $\mathbf{P}$ may be replaced by (11) in terms of the coherence matrix ρ.The second term of (44) can be normalized by the diagonal matrix $\mathbf{P_d}$ and written as :

$$\mathbf{P}^{-1}(\mathbf{S}^{\dagger}\mathbf{S})^{-1}\mathbf{P}^{-1} = \mathbf{P}_d^{-1}\left\{\mathbf{P}_d^{1/2}\rho^{-1}\mathbf{P}_d^{-1/2}\left[\mathbf{S}^{\dagger}\mathbf{S}\right]^{-1}\mathbf{P}_d^{-1/2}\rho^{-1}\mathbf{P}_d^{1/2}\right\}\mathbf{P}_d^{-1}$$

$$= \mathbf{P}_d^{-1}\,\mathbf{D}_{MU}^{-1}\,\mathbf{P}_d^{-1} \qquad (45)$$

with :

$$D_{MU}^{-1} = C_{MU}^{-1} S_{MU}^{-1} C_{MU}^{-1\ \dagger}$$

$$C_{MU} \triangleq P_d^{1/2} \rho P_d^{-1/2} \qquad (46)$$

$$S_{MU} \triangleq S^{\dagger} S$$

Matrices C_{MU} and S_{MU} are dimensionless normalized matrices, with diagonal elements equal to one . C_{MU} reduces to the identity matrix for decorrelated sources; it is influenced only by the correlation and power ratios between sources . The source vectors being assumed normalized to one $S_{MU} = S^{\dagger}S$ is the matrix of the cosine between sources vectors and reduces to the identity matrix for well separated sources; it reflects the geometry of the antenna and source pattern.

Using (44) and (46) , we may write:

$$E\left[\delta S_{m\perp} \delta S_{n\perp}^{\dagger}\right] \equiv \frac{\sigma^2}{Q\sqrt{P_m P_n}}\left[(\rho^{-1})_{mn} + \frac{\sigma^2}{\sqrt{P_m P_n}} \left(D_{MU}^{-1}\right)_{mn}\right] \Pi_B \qquad (47)$$

This expression highlights the effect of a bad conditioning of ρ or $S^{\dagger}S$, corresponding respectively to strong correlation or to sources with small angular separations : the determinants become very small and the variance increases drastically (due to the inversions ρ^{-1} and D_{MU}^{-1}) .

IV C- PERTURBATION OF THE SIGNAL PROJECTOR

From the previous results, in the first order approximation, vectors $\hat{\underline{U}}_i = \underline{U}_i + \delta\underline{U}_{i\perp}$ form an othonormal basis for $\hat{\xi}_S$. Under this approximation and according to relation (39) the estimated projector is given by :

$$\hat{\Pi}_S = \Pi_S + \delta\Pi_S \equiv \sum_{i=1}^{M} \hat{\underline{U}}_i \, \hat{\underline{U}}_i^{\dagger} \equiv \Pi_S + \mathcal{P}\Pi_S + \Pi_S\mathcal{P}^{\dagger}$$

$$\delta\Pi_S = -\delta\Pi_B \equiv \mathcal{P}\Pi_S + \Pi_S\mathcal{P}^{\dagger} \qquad (48)$$

The corresponding perturbed noise projector will be noted:

$$\hat{\Pi}_B = \Pi_B + \delta\Pi_B \qquad (49)$$

The previous results will now be applied to the derivation of MUSIC and AMLM theoretical variances .

V VARIANCE OF MUSIC AND AMLM COMPARED TO THE CRB

VA- MUSIC ALGORITHM

When the exact projector Π_B is replaced by the estimate $\hat{\Pi}_B$ in (16), the function $f(\theta)$ is non longer equal to zero for θ_m, but has a minimum close to θ_m. This minimun corresponds to a zero of the derivative $f'(\theta) = df/d\theta$. Using a first order Taylor expansion of $f'(\theta)$, near θ_m we get :

$$f'(\theta_m) \cong f'(\theta_m) + (\theta - \theta_m) f''(\theta_m) \tag{50}$$

using notation (32) and (49) we obtain for the first and second derivatives of (16) :

$$\begin{aligned} f'(\theta_m) &= \dot{\underline{S}}_m^{\dagger} (\Pi_B + \delta\Pi_B)\underline{S}_m + cc \\ f''(\theta_m) &= \ddot{\underline{S}}_m^{\dagger} (\Pi_B + \delta\Pi_B)\underline{S}_m + cc + 2\,\dot{\underline{S}}_m^{\dagger} (\Pi_B + \delta\Pi_B)\,\dot{\underline{S}}_m \end{aligned} \tag{51}$$

where cc denotes complex conjugate .

From the fact that $\Pi_B \underline{S}_m = 0$, we deduce from (51) that :

$$\begin{aligned} f'(\theta_m) &= 2\,\mathrm{Real}(\dot{\underline{S}}_m^{\dagger}\, \delta\Pi_B \underline{S}_m) \\ f''(\theta_m) &\cong E[f''(\theta_m)] = 2\dot{\underline{S}}_m^{\dagger} \Pi_B \dot{\underline{S}}_m \end{aligned} \tag{52}$$

By substitution of (48) and (40) in the expression of f'(θ), from the fact that $\mathcal{P}^{\dagger}\underline{S}_m=0$ we obtain :

$$\delta\Pi_B \underline{S}_m \equiv -\mathcal{P}\underline{S}_m = -\delta\underline{S}_{m\perp}$$

$$f'(\theta_m) \equiv -2\,\mathrm{Real}\left(\dot{\underline{S}}_m^{\dagger}\,\delta\underline{S}_{m\perp}\right) \tag{53}$$

From (50) and the derivatives expressions(52),(53) it results that a zero of f'(θ) is obtained for the value $\hat{\theta}_m$ of the variable θ given by :

$$\hat{\theta}_m - \theta_m \equiv \frac{\mathrm{Real}\left(\dot{\underline{S}}_m^{\dagger}\,\delta\underline{S}_{m\perp}\right)}{\dot{\underline{S}}_m^{\dagger}\,\Pi_B\,\dot{\underline{S}}_m} \tag{54}$$

According to (39),(40), $\delta\underline{S}_m$ is a linear combination of random circular variables, and so is $\underline{S}^{\dagger}_m\,\delta\underline{S}_m$. We may then apply the results of appendix B to calculate the covariance of (54), and we obtain :

$$E\left[\left(\hat{\theta}_m-\theta_m\right).\left(\hat{\theta}_n-\theta_n\right)\right] \equiv \frac{1}{2}\,\mathrm{Real}\left\{\frac{\dot{\underline{S}}_m^{\dagger}\,E\left[\delta\underline{S}_{m\perp}\,\delta\underline{S}_{n\perp}^{\dagger}\right]\dot{\underline{S}}_n}{\left|\dot{\underline{S}}_m^{\dagger}\,\Pi_B\,\dot{\underline{S}}_m\right|\left|\dot{\underline{S}}_n^{\dagger}\,\Pi_B\,\dot{\underline{S}}_n\right|}\right\} \tag{55}$$

By substitution in this relation of the perturbation $\delta \underline{S}_{m\perp}$ covariance (47), we obtain, for m=n :

$$E\left[\left(\hat{\theta}_m - \theta_m\right)^2\right]_{MU} \equiv \frac{\sigma^2}{2Qp_m \dot{\underline{S}}_m^{\dagger} \Pi_B \dot{\underline{S}}_m} \left[\left(\rho^{-1}\right)_{mm} + \frac{\sigma^2}{p_m}\left(D_{MU}^{-1}\right)_{mm}\right] \qquad (56)$$

where ρ is the coherence matrix and D_{MU} is defined in (46).This form is very similar to the CRB (38) .A detailed comparison will be made in section V.C.

V B VARIANCE OF THE AMLM

Using the vector variable $\underline{\theta}=(\theta_1,\ldots,\theta_M)^t$ a derivation similar to the case of MUSIC is used to obtain the variance of the AMLM.

The function (28),(29) to minimize is:

$$f(\underline{\hat{\theta}}) = \mathrm{Tr}\left[\underline{S}^{\dagger}(\underline{\hat{\theta}})\, \hat{\Pi}_B\, \underline{S}(\underline{\hat{\theta}})\, P\right] \qquad (57)$$

$$f(\underline{\hat{\theta}}) = \sum_{m,n}\left[P_{nm}\, \underline{S}_m^{\dagger}(\underline{\hat{\theta}})\, \hat{\Pi}_B\, \underline{S}_n(\underline{\hat{\theta}})\right]$$

For the local minima of this function the gradient $\underline{\nabla} f$ is zero. Such a minimum must occur for a value of $\underline{\hat{\theta}}$ close to the exact value of the parameter vector. We may then use the first order Taylor expansion of the gradient $\underline{\nabla} f(\underline{\hat{\theta}})$ near $\underline{\theta}$,given by :

$$\underline{\nabla} f\left(\overset{\Delta}{\underline{\theta}}\right) \cong \underline{\nabla} f(\underline{\theta}) + \Delta f(\underline{\theta})\delta\underline{\theta}$$

$$\delta\underline{\theta} = \overset{\Delta}{\underline{\theta}} - \underline{\theta} \qquad (58)$$

In this expression $\Delta f(\theta)$ is the Hessian . The gradient and Hessian components are defined by :

$$\underline{\nabla} f(\underline{\theta})_m = \frac{\partial f\left(\overset{\Delta}{\underline{\theta}}\right)}{\partial \theta_m}$$

$$\Delta f(\underline{\theta})_{mn} = \frac{\partial^2 f\left(\overset{\Delta}{\underline{\theta}}\right)}{\partial \theta_m \partial \theta_n} \qquad (59)$$

For the minima , $\underline{\nabla}$ f must be equal to zero . The corresponding solution of (58) for $\delta\underline{\theta}$ is the solution of :

$$\Delta f(\underline{\theta})\delta\underline{\theta} \cong - \underline{\nabla} f\left(\overset{\Delta}{\underline{\theta}}\right) \qquad (60)$$

In this relation , from (57) the components of $\underline{\nabla}$ f and Δf are :

$$\underline{\nabla} f(\underline{\theta})_m = 2\,\mathrm{Re\,al}\left(\sum_i P_{im}\, \dot{\underline{S}}_m^{\dagger}\, \hat{\Pi}_B\, \underline{S}_i\right)$$

$$\Delta f(\underline{\theta})_{m,n} = 2\,\mathrm{Re\,al}\left[\left(\dot{\underline{S}}_m^{\dagger}\, \hat{\Pi}_B\, \dot{\underline{S}}_n + \ddot{\underline{S}}_m^{\dagger}\, \hat{\Pi}_B\, \underline{S}_n\, \delta_{mn}\right) P_{nm}\right] \qquad (61)$$

where δ_{mn} is the Kronecker symbol.

From (48) (40), using the fact that $\Pi_B \underline{S}_i = 0$ and $\mathcal{P}^{\dagger} \underline{S}_i = \underline{0}$ we obtain in the expression of the gradient :

$$\hat{\Pi}_B S_i = \delta \Pi_B S_i \equiv \mathcal{P} \Pi_S S_i = \delta S_{i\perp}$$

$$\underline{\nabla} f(\underline{\theta})_m = 2\,\mathrm{Re\,al}\left(\sum_i P_{im}\, \dot{\underline{S}}_m^{\dagger}\, \delta \underline{S}_{i\perp}\right) \qquad (62)$$

In the first order approximation the Hessian may be approximated by its expectation, i.e $\hat{\Pi}_B$ is replaced by its exact value Π_B. Then we obtain:

$$\Delta f(\underline{\theta})_{mn} \equiv E\left[\Delta\, f(\underline{\theta})_{mn}\right] = 2\,\mathrm{Re\,al}\left(\dot{\underline{S}}_m^{\dagger}\, \Pi_B\, \dot{\underline{S}}_n\, P_{nm}\right) \qquad (63)$$

By comparison with (31) we can see that the Hessian is proportional to the Fisher matrix .It will then be assumed regular.From (60) using the reduced notation $\underline{\nabla}$ f and Δf for the gradient and Hessian:

$$\Delta f\; E\left[\delta \underline{\theta}\, \delta \underline{\theta}^{\dagger}\right] \Delta f \equiv E\left[\underline{\nabla} f \underline{\nabla} f^{\dagger}\right]$$
or: (64)
$$E\left[\delta \underline{\theta}\, \delta \underline{\theta}^{\dagger}\right] \equiv \Delta f^{-1}\, E\left[\underline{\nabla} f \underline{\nabla} f^{\dagger}\right] \Delta f^{-1}$$

This covariance calculation is made in appendix C and gives the following results (C.14):

$$E\left[\left(\hat{\theta}_m - \theta_m\right)^2\right]_{AML} \equiv \frac{\sigma^2}{2Qp_m\, \dot{\underline{S}}_m^{\dagger}\, \Pi_B\, \dot{\underline{S}}_n} \left\{\left[\rho_{CR}^{-1}\right]_{mm} + \frac{\sigma^2}{p_m}\left[D_{ML}^{-1}\right]_{mm}\right\} \qquad (65)$$

Where ρ_{CR} is the modified coherence matrix introduced in (35) and D_{ML} is defined by a relation

similar to (46) :

$$\mathbf{D}_{ML}^{-1} = \mathbf{C}_{ML}^{-1}\,\mathbf{S}_{ML}^{-1}\,\mathbf{C}_{ML}^{-1} \tag{66}$$

Matrices $\mathbf{C}_{ML}$ and $\mathbf{S}_{ML}$ are related to the corresponding matrices introduced in (46) by:

$$[\mathbf{C}_{ML}]_{mn} = [\mathbf{C}_{MU}]_{mn}\cos\beta_{mn}$$

$$[\mathbf{S}_{ML}^{-1}]_{mn} = [\mathbf{S}_{MU}^{-1}]_{mn}\cos\beta_{mn}\cos\gamma_{mn} \tag{67}$$

$$\gamma_{mn} \triangleq \arg\left(\dot{\underline{S}}_m^{\dagger}\,\Pi_B\,\dot{\underline{S}}_n\,[\mathbf{S}^{\dagger}\mathbf{S}]_{mn}^{-1}\right)$$

where $\cos\beta_{mn}$ is the cosine of the angle between $\Pi_B\dot{\underline{S}}_m$ and $\Pi_B\dot{\underline{S}}_n$, defined in (36).

VC COMPARISON OF THE THEORETIC STATISTICAL PERFORMANCE

VC 1 - Preliminary discussion

The analytical expression of the variance for the two methods and the CRB are summarized in Table I .

Remember that statistical perturbations are only first order approximations, in the order of $\varepsilon_m=\sigma^2/(Qp_m)$ according to (47) . All higher order terms in this variable should be meaningless . In particular the correcting terms proportional to σ^2/p_m are valid only asymptotically for $Q >> 1$ so that the corresponding terms in ε_m can be ignored.

To clarify the discussion we will first consider two limiting cases: the case of decorrelated sources, and the high SNR limit .

VC 2 -Decorrelated sources

In this particular situation (**P** diagonal), both MUSIC and AMLM give the same estimate $\hat{\theta}_m$. It is easily verified that the expressions of the variance in table I are identical . Matrices ρ and ρ_{CR} reduce to the identity matrix, so that the variances reduce to:

$$E\left[\delta\theta_m^2\right]_{MU} = E\left[\delta\theta_m^2\right]_{AML} \equiv CRB\left\{1 + \frac{\sigma^2}{p_m}\left[S^\dagger S\right]^{-1}_{mm}\right\} \tag{68}$$

$$CRB = \frac{\sigma^2}{2Qp_m}\ \frac{1}{\dot{\underline{S}}_m^\dagger\ \Pi_B\ \dot{\underline{S}}_m}$$

The two estimators then achieve the CRB at high SNR ($\sigma^2/p_m \to 0$). The threshold under which they lose efficiency is equal to the value of $[S^\dagger S]_{mm}^{-1}$, connected to the separation between sources . For well separated sources the source vectors are orthogonal , $S^\dagger S$ reduces to

the identity matrix and the threshold is equal to one (SNR = 0 db) .

The effect of smaller separation is best illustrated in the case of two sources; then the SNR threshold is :

$$\frac{p_m}{\sigma^2} = \frac{1}{1 - \left| \underline{S}_1^{\dagger} \underline{S}_2 \right|^2} \qquad (69)$$

For sources with small separation, $|\underline{S}_1^{\dagger}\underline{S}_2|$ becomes close to one and the threshold may occur for SNR values well over 0 db) . The variation of $\underline{S}_1\ \underline{S}_2$ as a function of source separation is represented figure 1 (dashed line) in the case of a linear array. This curve represents as well the ambiguity function for the conventional beamforming.

VC 3 -High SNR limit

In this case ($\sigma^2/p_m \rightarrow 0$) the correcting terms in the expression of the variance vanish . It is readily seen that the AMLM achieves the CRB; for MUSIC, the only difference is that ρ_{CR}^{-1} is replaced by ρ^{-1} . Generally speaking ρ_{CR} is better conditioned for inversion, due to the weighting of the off diagonal elements by factors less than one, according to (35) . In particular ρ_{CR} will be invertible even in the case of full correlation, when ρ is singular. This may be seen in the simple case of two sources:

$$\left[\rho^{-1}\right]_{11} = \left[\rho^{-1}\right]_{22} = \frac{1}{1 - |\rho_{12}|^2} \qquad (70)$$

while :

$$\left[\rho_{CR}^{-1}\right]_{11} = \left[\rho_{CR}^{-1}\right]_{22} = \frac{1}{1 - |\rho_{12}|^2 \cos^2\alpha_{12}\cos^2\beta_{12}} \qquad (70b)$$

In the presence of strong correlation ($|\rho_{12}| \approx 1$), the first expression may be very large. The second will remain close to one either for well separated sources ($\cos \beta_{12} \approx 0$), or according to the average phase relationship between sources (see (37)) if $\alpha_{12} \approx \Pi/2$ .The effect of source separation on $\cos \beta_{12}$ is represented figure 1 (solid line), compared to the ambiguity function in conventional beamforming (dashed line) for a linear array. The variable is the reduced frequency,normalized to have the first zero of the ambiguity function at f=1 . The curve for $\cos \beta_{12}=1$ is wider, with a first zero at f≈2 . The maximum at f=0 is very flat .

VC 4 -Comparison at lower SNR

Under an SNR threshold given by the ratio of $[D_{MU}^{-1}]_{mm}$ to $[\rho^{-1}]_{mm}$ the variance is dominated by the second term between brackets, and becomes proportional to $(1/Q)(p_m/\sigma^2)^2$.

In the factorization (46) of D_{MU} it can be noted that S_{MU} depends only on the sources/ antenna pattern, while C_{MU} is a function of the power ratios and correlations .

For the AMLM, due to the fact that again ρ^{-1} is replaced by ρ_{CR}^{-1}, the variance will in general be smaller than for MUSIC .

All the previous discussion will be now ilustrated in the case of a linear array.

VC 5 -Theoretical variance for two sources and a linear array

The theoretical variances have been calculated for a linear array of N=10 equispaced elements. We have used the position parameter

$$\theta=2\Pi(d/\lambda)\cos\varphi$$

where λ is the wavelenght, d the spacing between elements and φ the bearing of the source.

The separation between sources is best characterized in terms of the corresponding reduced frequency:

$$f=\theta/(2\Pi N)$$

the Fourier resolution in conventional beamforming corresponding to $f_2-f_1=1$ (see figure 1) .

Two cases have been considered,the first one corresponding to the Fourier limit $f_2-f_1=1$, and the other one to sources with small separation $f_2-f_1=0.064$.

According to (1) , the source vectors $\underline{S}_1$, $\underline{S}_2$ are normalized to one.

We suppose that $p_1=p_2$. The SNR is defined as $10\log(p_1/\sigma^2)$; due to the normalization of the source vectors this corresponds to the SNR for one channel (and not for one element).The number of snapshots is Q=100 .

The theoretical variances as a function of SNR are represented in figures 2,3,4, together with the results of Monte Carlo simulations concerning the total estimation error and the bias. The Monte Carlo simulations will be discussed in section VII . On the variance plots, the horizontal dashed line corresponds to the level where the standard deviation (std) becomes equal to the source separation,which will play an important part in this discussion .

The case of decorrelated sources is represented figure 2. As expected both estimators give the same results and achieve the CRB at high SNR . In the first case ($f_2-f_1=1$) the SNR threshold is just a little higher than 0 dB. For small source separation the SNR threshold is much higher, in the order of 30 dB (fig 2c).

For correlated sources (ρ=0.9, figure 3 and 4) the AMLM achieves the CRB at

high SNR . It can be seen that for $\alpha=90°$ the variance is the same as when $\rho=0$. For other values of the phase difference, the threshold is a complex function of α, but the variance is always greater.

For comparison with MUSIC two factors play a major part: the values of $\cos\alpha$, $\cos\beta$ and their "decorrelating effect", which appears in (70b) for instance .

For $|\cos\alpha| = 1$ (sources in phase, or in opposition),only remains the value of $\cos\beta$. If we refer to figure 4, for (f_2-f_1)=0.064 ,$\cos^2\beta_{12}$ is much closer to one than $\rho^2=0.81$(in fact $\cos^2\beta_{12}=0.995$) : the decorrelating effect is negligible. Indeed in figure 4a and 4e the results of MUSIC and AMLM are identical.

When the separation between sources increases, $\cos^2\beta$ becomes smaller. For $f_2-f_1=1$ it is in the same order as ρ^2, and this explains the relative degradation of MUSIC in figure 3a and 3e . The degradation could be much greater, either if the correlation ρ is closer to one or if the separation between sources increases a little more (see figure1 the variation of $\cos\beta$).

For $|\cos\alpha| \neq 1$ an additional decorrelation appears which culminates for $\alpha=\Pi/2$ (fig. 3c and 4c) .

Note that this decorrelation connected to α can be introduced in MUSIC by the use of Forward Backward averaging [13] . But this is possible only for a symmetric array and it has not been introduced here,since the paper deals in general with an arbitrary array geometry .

VI THE MINIMIZATION ALGORITHM

Two formulations are given for the algorithm. In the basic form, matrix **P** in expression (28) to be minimized is considered as constant . Usually the algorithm is not very sensitive to small deviations in **P**, so that any reasonable estimate may be substituted for it .

This is no longer true when **P** is singular, i.e. when sources are fully correlated . Matrix **P** may then be updated at each iteration according to (17), but the basic algorithm convergence rate is slow . In the improved algorithm the dependence of the estimate of **P** with respect to the parameters is taken into account. The modification is quite straightforward and a fast convergence is restored. This modified version will be retained as the "AMLM" algorithm in the following .

VI A- BASIC ALGORITHM

VI A.1 Gradient and Hessian

The gradient and Hessian have been already calculated and they are given in (61).

An alternate compact form is obtained if we define matrices $\dot{S}$ and $\ddot{S}$ by :

$$\dot{S} = \left[\dot{\underline{S}}_1, \ldots, \dot{\underline{S}}_M\right] \quad , \ddot{S} = \left[\ddot{\underline{S}}_1, \ldots, \ddot{\underline{S}}_M\right] \tag{71}$$

and if we denote by Θ the element by element product of matrices. Then (61) may be written:

$$\underline{\nabla} f = 2\,\mathrm{Re\,al}\left[\mathrm{diag}\left(\dot{\underline{S}}^{\dagger}(\underline{\theta})\,\hat{\Pi}_B\,\underline{S}(\underline{\theta})P\right)\right]$$

$$\Delta f = 2\,\mathrm{Re\,al}\left[\left(\dot{\underline{S}}^{\dagger}(\underline{\theta})\,\hat{\Pi}_B\,\dot{\underline{S}}(\underline{\theta})\Theta P^{*} + \ddot{\underline{S}}^{\dagger}(\underline{\theta})\,\hat{\Pi}_B\,\underline{S}(\underline{\theta})P\;\Theta I\right)\right] \tag{72}$$

where, for any matrix **M** , diag(**M**) denotes the column vector formed with the diagonal elements of **M** and **M*** denotes the complex conjugate of **M** .

Note that for the exact values of the projector and the source matrix, $\Pi_B S=0$, so that in the expression of the Hessian the second term will be deleted. This guarantees that Δf is non negative. Furthermore, by comparison with (31) we can see that it is proportional to the Fisher information matrix and is invertible,unless the parameters are not identifiable.

VI A.2 Algorithm

The basic Newton Gauss recursion is then given by :

$$\underline{\theta}_{k+1} = \underline{\theta}_k - [\Delta f]^{-1} \underline{\nabla} f \qquad (73)$$

The recursion is in two steps : update of the power estimate from θ_k using relation (17), and the calculation of θ_{k+1} by (73)

$$\text{step 1 : } \mathbf{P}_k = \left[S^{\dagger}(\underline{\theta}_k) S(\underline{\theta}_k)\right]^{-1} S^{\dagger}(\underline{\theta}_k) \hat{\mathbf{R}}_y S(\underline{\theta}_k) \left[S^{\dagger}(\underline{\theta}_k) S(\underline{\theta}_k)\right]^{-1} \qquad (74)$$

$$\text{step 2 : } \quad \underline{\theta}_{k+1} = \underline{\theta}_k - [\Delta f_k]^{-1} \underline{\nabla} f_k$$

$$\underline{\nabla} f_k = 2\,\mathrm{Re\,al}\left[\mathrm{diag}\left(\dot{S}(\theta_k) \hat{\Pi}_B S(\theta_k) \mathbf{P}_k\right)\right] \qquad (75)$$

$$\Delta f_k = 2\,\mathrm{Re\,al}\left[\dot{S}(\theta_k) \hat{\Pi}_B S(\theta_k) \Theta \mathbf{P}_k\right]$$

VI B -Improved practical algorithm

According to the update equation (74), $\mathbf{P}_k$ is now considered a function of $\underline{\theta}$. Then by the substitution of (74) in (57) :

$$f(\underline{\theta}) = \mathrm{Tr}\left[\hat{\Pi}_B \, \Pi_S(\underline{\theta}) \hat{\mathbf{R}}_y \, \Pi_S(\underline{\theta})\right] \qquad (76)$$

$$\Pi_S(\underline{\theta}) = S(\underline{\theta}) \left[S^{\dagger}(\underline{\theta}) S(\underline{\theta})\right]^{-1} S^{\dagger}(\underline{\theta})$$

It is proved in annex D that near the convergence point the gradient and Hessian can be approximated by the following expressions (D.5),(D.10) , very similar to (72) :

$$\nabla f_k = 2\,\mathrm{Re\,al}\left[\mathrm{diag}\left(\dot{S}(\theta_k)\,\Pi_B(\theta_k)\hat{\Pi}_B S(\theta_k)\right)P_k\right]$$

$$\Delta f_k = 2\,\mathrm{Re\,al}\left[\dot{S}(\theta_k)\,\Pi_B(\theta_k)\hat{\Pi}_B\,\Pi_B(\theta_k)\,\dot{S}(\theta_k)\Theta P_k^*\right] \qquad (77)$$

$$\Pi_B(\theta) \triangleq I - S(\theta)\left[S^\dagger S\right]^{-1} S^\dagger(\theta)$$

The recursion is the same as in (74),(75), the expressions of the gradient and Hessian being replaced by (77) .

VI C -Convergence study

Three main points are of interest: the speed of convergence, the existence of secondary extrema and the resolution of close sources in presence of noise.

On these points the comparison has been made in simulation between three algorithms. The first one is the practical algorithm defined by (73) an (77) which will be refered as "AMLM"; the second, introduced only as a theoretical reference, is obtained by substitution in the basic equation (75) of the exact power **P** instead of $\mathbf{P_k}$. In practice it could not be implemented since **P** is unknown. But it is interesting for comparison by simulation because,strictly speaking, it is in this case that the theoretical variance analysis apply. It will be named "RAML" . The third algorithm is again a straighforward variant of the basic algorithm, used to find the solution of MUSIC: as already noted we just have to substitute for $\mathbf{P_k}$ in (75) any diagonal matrix. We have used diag(p_1 , p_2.. p_m). The algorithm is refered as "MUSIC" .

The simulation have been made as previously in the case of a linear array of 10 elements, with two sources of equal powers $p_1 = p_2 = 1$. We will analyse only the case of two

close sources θ_1 =-0.02 , θ_2 =+0.02 (Δf=0.063) for which the convergence problems are most critical.

VI C1 -Contour plots without noise

For the RAMLM and MUSIC algorithms the function minimized are given by (57), **P** being replaced by the identity matrix for MUSIC (p_1=p_2=1). For the AMLM the function is (76). The contour plots of these functions gives a good insight of the occasional problems of convergence. They are displayed figure 5 for ρ=0.9, α_{12}=0, Π/2, Π .For α_{12} =Π/2 the contours are independent of ρ so that figures 5c and 5d are valid for ρ =0 . For MUSIC the contour plot is independent of α and ρ and it is identical to figure 5d.

Obviously if ($-\theta_0$,θ_0) is solution (θ_0,$-\theta_0$) is also solution, which explains the symmetry of the plots with respect to the axis θ_1=θ_2 .

A first important remark is that the AMLM,but for this symmetry, has a unique minimum, while the other ones have a spurious minimum on the axis θ_1=θ_2 . The consequence is that whatever the initialization, the AMLM will always give the right solution (-0.02, 0.02) while the others ones may converge to the spurious minima (-0.02, -0.02) or (0.02 ,0.02) .

Three initialization points, far from the "well behaved" central part of the contours, have been tried to test the speed of convergence. The convergence path are represented figure 5a and 5b . The corresponding variation of θ_1 and θ_2 as a function of the iteration index are plotted figure 6a and 6b . It can be observed that convergence is obtained in at most 5 iterations. But for RAMLM and MUSIC it goes to the wrong mininum if the initialization is outside the quadrant $\theta_1.\theta_2$ <0.

A surprising feature is that, in the noiseless case, the convergence path are independent ρ of and α : the contours are very much distorted, but this is exactly compensated by corresponding modifications of the Hessian .

VI C2 -Typical contour plots with noise

Typical contour plots corresponding to an SNR of 50 dB and an average on 100 snapshots are represented figure 7. For the current small source separation and ρ=0.9 , this correspond more or less to the resolution limit of MUSIC.

For the AMLM the general shape remains the same as in figure 5, with only a small random deviation of the extrema.

For MUSIC, compared to figure 5d, figure 7c, 7f, 7i are typical occurrences of the contours that can be obtained at the limit of resolution, with either one single minimum (7c) or four ones, but usually closer (7f) than without noise.

Concerning RAMLM, for α=0 the "good" minima have been enhanced (7b compared to 5f) while only the "bad" ones remains for $\alpha = \Pi$ (7h compared to 5f), in which case the sources are non longer resolved. For $\alpha=\Pi/2$, the symmetry is lost: this is because the algorithm "knows" **P** and expect the source at θ_1 to be in quadrature before θ_2 which is true in the half plane $\theta_1 < \theta_2$ and wrong on the other side.

The main conclusion is that the AMLM convergence properties are far less affected by the noise than that of the two other methods. This is confirmed by a more extensive Monte Carlo analysis given in figure 8, which represents the convergence points obtained for 100 independents trials, at the same SNR, the algorithms being initialized at the exact position (-0.02,0.02) .

The AMLM exhibits good performance with repartitions of the dots centered on the exact position, with shapes reproducing the corresponding contours in figure 7 a, 7d, 7g .

The RAMLM has a good results for α=0 and $\alpha=\Pi/2$ but is very bad for $\alpha=\Pi$,with a numbers of points along the axis $\theta_1 = \theta_2$ corresponding to unresolved sources.

For MUSIC, in accord with figure 7c, 7f the results are bad for α =0 or $=\Pi/2$ with a lot of estimates biased or unresolved, while for $\alpha = \Pi$ the estimation remains correct.

As a conclusion the fact that the AMLM algorithm rely on a weighting matrix $\mathbf{P}_E(\underline{\theta})$ reestimated at each point (θ_1, θ_2) seams to improve significativly the resolution . The secondary minima are "rubbed out" in the same way as spurious solutions may be eliminated afterwards in other methods on the basis of the corresponding power estimates amplitude .

VII STATISTICAL PERFORMANCE ANALYSIS BY MONTE CARLO SIMULATION

The three algorithms, AMLM, RAMLM, and MUSIC have been compared in the two situations considered for the calculation of the theoretical variances in subsection VC3 : linear array of N=10 elements, Q=100 snapshots, reduced frequency separations $\Delta f=1$ and $\Delta f=0.063$,for correlation $\rho=0$ or $\rho=0.9$ with $\alpha=0, \Pi/2, \Pi$.

For each set of parameters we have computed, by an average on 100 independent trials, the *total mean square deviation* of the estimates of θ_1 and θ_2 and the average values of the estimates.

These estimates are the values obtained by the 3 algorithms, initialized at the exact value, after 10 iterations . No sorting has been done to eliminate the large estimates or the cases of non-resolution .

We may expect substantial deviation from the first order theoretical analysis at least when the theoretical standard deviation becomes equal to the separation between sources . The corresponding level is represented by horizontal dashed line in figures 2, 3 ,4 of the theoretical variance, where the results of the Monte Carlo simulations have been represented for comparison .

VII A COMPARISION FOR $\rho=0$

In the case of $\Delta f=1$ (fig.2a,2b) the cross over point (between theoretical variance and source separation) is for an SNR of -10 dB , at the lower limit of the figure.

For the AMLM the deviation from the theoretical error is small . For MUSIC and RAMLM a deviation appears under 5 dB . The bias remains small for the three methods.

In the case of $\Delta f=0.063$ (figure 2c,2d) the cross over point is for an SNR of 30 dB. At this point the AMLM sharply departs from the theoretical curve and an important positive bias appears . For MUSIC and RAMLM (identical for $\rho=0$) a deviation from the theory appears 20 dB higher, for an SNR of about 50 dB .This corresponds to an important negative bias, resulting from a loss of resolution : very soon the sources are non longer resolved in about 100% of the cases under 35 dB of SNR .

VII A -COMPARISON FOR $\rho=0.9$,$\Delta f=1$

For $\alpha=\Pi/2$ (figure3) the variance for the MLM and RAMLM is the same than for $\rho=0$ with the cross over point at SNR=-10 dB . The theoretical results remains in good agreement with a small bias . For MUSIC the cross over point is 0 dB and this is the point where appears a bias and a deviation of the varaiance .

For $\alpha=0$ and $\alpha=\Pi$ the AMLM is good up to the cross over of -5 dB , where a positive bias appears.

There is a loss of resolution for MUSIC, $\alpha=0$ and RAMLM,$\alpha=\Pi$ about 10 dB over the corresponding cross over points .

VII C-COMPARISION FOR $\rho=0,9$ $\Delta f=0.063$

Again, for the RAMLM a strong positive bias and a deviation from the theoretical curve appear at the cross over point .

For MUSIC and AMLM a loss of resolution appears very early, for values of SNR 10 to more than 20 dB over the cross over point, the source being very soon unresolved (average values egal to zero) .

For $\alpha=0$ the degradation affects MUSIC before the RAMLM while for $\alpha=\Pi$ the RAMLM gives the worst results .

It may be noted that the RAMLM makes significant incursions *under* the CRB . Do not forget that this is a quite unrealistic algorithm, since it relies on the exact value of the power matrix **P** . In fact the CRB corresponding to the case of a known matrix P is well under the value represented figure 4 . When the first order approximation used to establish the theoretical curve is non longer valid, do no be surprised if the deviation is *under* instead of *over* the corresponding value! .

VII D-PERFORMANCE OF AMLM FOR ONE SINGLE SNAPSHOT

A significant advantage of AMLM is its ability to resolve fully correlated sources, and to operate on a single snapshot.

Monte Carlo simulation have been made in this last case, for $\Delta f=1$, using two exponentials of unit amplitude and a phase difference of $\alpha=0, \Pi, \Pi/2$ in additive noise. The results for $\alpha=0$ are represented figure 9, compared to the CRB.

The performance are similar to the previous cases: the variance fits the theory down to the cross over point. The same observations have been verified for $\alpha=\Pi/2$ and $\alpha=\Pi$.

VII E -CONCLUSION ON STATISTICAL PERFORMANCES

The analysis of the previous cases confirms that the cross over SNR corresponding to the point when the theoretical standard deviation equals the separation between sources is an important element in the interpretation .

For the AMLM no loss of resolution appears when the noise increases. On the contrary there is a tendency to over-estimate the source separation for SNR under the cross over . The theoretical results for the variance hold approximatly down to the cross over .

For MUSIC and RAMLM a loss of resolution appears for the values of the SNR that may be more than 20 dB over the cross over point, with a corresponding increase in the mean square estimation error . Over this value of SNR, the variance is also in perfect agreement with the theory .

For a linear array it is well known that root MUSIC achieves a better resolution than conventional MUSIC at low SNR. It has not been considered here since the paper deals in general with an arbitrary array geometry in which case root MUSIC cannot be applied.

VIII GENERAL CONCLUSION

Starting from a modified form of the log-likelihood we have suggested an approximate maximum likelihood estimator of the source bearings [12], which may be considered as a generalization of MUSIC to the case of correlated sources.

Making use of previous statistical analysis [13] we could derive the theorical performance compared to MUSIC and to the CRB, and quantify the improvement that could be expected from our method .

A Newton-Gauss algorithm has been derived for a computation-efficient implementation of the basic algorithm an a practical variant of it, relying on an estimate of the power matrix instead of the exact theoretical one.

An extensive simulation analysis has, on the one hand, confirmed the theoretical improvement of the variance at high SNR with respect to MUSIC, and has on the other hand demonstrated superior convergence and resolution performances, down to some significant "cross over SNR" . In the case of two sources this cross over corresponds to the SNR for which the theoretical variance becomes equal to the separation between sources. Down to this value of the SNR, for AMLM, the bias is negligible and the total mean squares error fits the theoretical prevision, while for MUSIC a significant loss of resolution and a variance increase appear for values of SNR that may be more than 20 dB larger .

Another advantage of the AMLM algorithm is its ability to provide efficient estimation even for one single snapshot. This opens a way for efficient recursive estimation schemes .

APPENDIX A : PERTURBATION $\delta \underline{S}_\perp$ COVARIANCE

To any vector $\underline{S}$ of ξ_S may be associated the vector $\underline{S} + \delta \underline{S}_\perp$ of the perturbed subspace $\hat{\xi}_S$ with, according to (39) and (40) :

$$\delta \underline{S}_\perp \equiv \Pi_B \langle \underline{B}\, \underline{X}^\dagger \rangle R_y^{\#} \underline{S} \qquad (A.1)$$

For two vectors $\underline{S}_m$ and $\underline{S}_n$ using (3) and (39), we obtain with simplified notation ($\underline{Y}$ and $\underline{B}$ are implicitly indexed by q , and $\underline{Y}'$ and $\underline{B}'$ by q') :

$$E\left[\delta S_{m\perp} \delta S'^{\dagger}_{n\perp}\right] \equiv \frac{1}{Q^2} \Pi_B \left\{ \sum_{q,q'} E\left[\underline{B}\,(\underline{Y}+\underline{B})^\dagger \underline{V}_m \underline{V}'^{\dagger}_n (\underline{Y}'+\underline{B}')\underline{B}'^{\dagger}\right]\right\} \Pi_B$$

$$\underline{V}_m \triangleq R_y^{\#} S_m \qquad , \underline{V}_n \triangleq R_y^{\#} S_n \qquad (A.2)$$

In the expectation only the noise is considered as random , with zero mean gaussian distribution . The components of the noise vector $\underline{B}$ are assumed complex circular, i.e., the real and the imaginary parts are decorrelated with the same variance .Then :

$$E\left[\underline{B}\,\underline{B}'^{\dagger}\right] = \sigma^2 I\, \delta_{qq'}$$

$$E\left[\underline{B}\,\underline{B}'^{t}\right] = 0 \qquad (A.3)$$

As a consequence the odd order moments are zero and there remain only two terms in the development of (A2) :

$$E\left[\underline{B}\,(\underline{Y}+\underline{B})^\dagger \underline{V}\,\underline{V}'^{\dagger}\,(\underline{Y}'+\underline{B}')\underline{B}'^{\dagger}\right] = E\left[\underline{B}\,\underline{Y}^\dagger\,\underline{V}\,\underline{V}'^{\dagger}\,\underline{Y}'\,\underline{B}'^{\dagger}\right]$$

$$+ E\left[\underline{B}\,\underline{B}^\dagger \underline{V}\,\underline{V}'^{\dagger}\,\underline{B}'\,\underline{B}'^{\dagger}\right] = 2^{nd}\text{order} + 4^{th}\text{order} \qquad (A.4)$$

These two terms will be referred as "2nd order " and "4th order". For the second order we easily obtain :

$$2^{nd}\text{order} = (\underline{V}'^{\dagger}\underline{V}')(\underline{Y}^{\dagger}\underline{V}) E[\underline{B}^{\dagger}\underline{B}'] \qquad (A.5)$$

$$= \underline{S}_m^{\dagger} \mathbf{R}_y^{\#} \underline{Y}\, \underline{Y}^{\dagger} \mathbf{R}_y^{\#} \underline{S}_n \sigma^2 \mathbf{I} \delta_{qq'}.$$

For the fourth order let us introduce the zero mean random circular gaussian variables:

$$v \triangleq \underline{V}_m^{\dagger}\underline{B} \quad , \quad v' \triangleq \underline{V}_m^{\dagger}\underline{B}' \qquad (A.6)$$

Using the usual properties of Gaussian complex variables, we then obtain :

$$4^{th}\text{order} = E[\underline{B}v^{*}v'\,\underline{B}'^{\dagger}] = E[\underline{B}v^{*}]E[v'\,\underline{B}'^{\dagger}] + E[\underline{B}\,\underline{B}'^{\dagger}]E[v^{*}v']$$

$$= (\sigma^2\mathbf{I})\underline{V}_m\underline{V}_n^{\dagger}(\sigma^2\mathbf{I}) + \delta_{qq'}(\sigma^2\mathbf{I})\underline{V}_n^{\dagger}(\sigma^2\mathbf{I})\underline{V}_m \qquad (A.7)$$

$$= \sigma^4 \mathbf{R}_y^{\#}\underline{S}_m\underline{S}_n^{\dagger}\mathbf{R}_y^{\#} + \sigma^4 \underline{S}_n^{\dagger}\mathbf{R}_y^{\#2}\underline{S}_m\delta_{qq'}.$$

Note that the first term has no contribution when substituted in (A.2), due to the fact that $\Pi_B \mathbf{R}^{\#}_y = 0$. The remaining terms in (A.4), resulting from (A.5) and (A.7) are then zero for $q \neq q'$,

so that the double sum on q and q' reduces to a single sum on q, with q=q'.With explicit indexes, in the summation on the second order term (A.5), from (7) and (9) :

$$\frac{1}{Q}\sum_q \underline{Y}(q)\underline{Y}^{\dagger}(q) = \mathbf{S}\mathbf{P}\mathbf{S}^{\dagger} = \mathbf{R}_y \qquad (A.8)$$

By substitution of (A.5),(A.7) in (A.2) according to (A.4) and (A.5), we then obtain :

$$E\left[\delta\underline{S}_{m\perp}\delta\underline{S}_{n\perp}^{\dagger}\right] = \frac{\sigma^2}{Q}\Pi_B\underline{S}_n^{\dagger}\left\{\mathbf{R}_y^{\#}[\mathbf{S}\mathbf{P}\mathbf{S}^{\dagger} + \sigma^2\mathbf{I}]\mathbf{R}_y^{\#}\right\}\underline{S}_m\Pi_B \qquad (A.9)$$

APPENDIX B: REAL COMPONENT COVARIANCE FOR COMPLEX RANDOM VECTORS

We want to demonstrate two propositions:

proposition 1: if $\underline{Z}$ is a complex zero mean random vector such that:

$$E[\underline{Z}\,\underline{Z}^t] = 0 \qquad \text{(B. 1)}$$

then the following property holds:

$$E[\,Real(\underline{Z})Real(\underline{Z}^t)] = \frac{1}{2}Real[E\,[\underline{Z}\,\underline{Z}^\dagger]\,] \qquad \text{(B.2)}$$

proposition 2: if the components of a vector Z are linear deterministic combinations of the components of a vector $\underline{\mathfrak{B}}$ which are random circular decorrelated variables, then the property (B.1) holds .

In the paper, vector $\underline{\mathfrak{B}}$ will be the concatenation of the noise vectors for the different snapshots:

$$\underline{\mathfrak{B}}^t = [B^t(1), \ldots, B^t(q)] \qquad \text{(B. 3)}$$

The demonstration of these propositions is quite straigthforward :

$$Real[\underline{Z}] = \frac{1}{2}[\underline{Z} + \underline{Z}^*] \qquad \text{(B.4)}$$

so that :

$$Real[Z]Real[Z]^t = \frac{1}{4}\{Z\,Z^\dagger + (Z\,Z^\dagger)^* + Z\,Z^t + (Z\,Z^t)^*\}$$

$$= \frac{1}{2}\{Real[Z\,Z^\dagger] + Real[Z\,Z^t]\} \qquad \text{(B.5)}$$

Taking expectation on both sides, relation (B.2) results if the condition (B.1) is satisfied .
For proposition 2, if there exists a linear relationship it may be written in matrix form:

$$\mathbf{Z} = \mathbf{Q}\mathbf{B} \tag{B.6}$$

with $\mathbf{Q}$ a deterministic matrix .Then :

$$E[\mathbf{Z}\,\mathbf{Z}^{t}] = \mathbf{Q}E[\mathbf{B}\,\mathbf{B}^{t}]\,\mathbf{Q} \tag{B.7}$$

and the property (B.1) holds for $\underline{Z}$ if it is verified for $\underline{B}$: indeed it is true for random decorrelated circular complex variables,which demonstrates proposition 2.

APPENDIX C

In (64), let

$$\mathbf{H} \triangleq E[\underline{\nabla} f \underline{\nabla} f^{\dagger}] \qquad (C.1)$$

We verify that $\underline{\nabla} f$ is the real part of a vector which satisfies the conditions of appendix B : in (62), according to (39) (40) $\delta \underline{S}_{\perp}$ is a linear combination of the components of $\underline{B}$. By application of relation (B.2) the elements of matrix **H** are then given by :

$$H_{mn} = 2\,\mathrm{Re\,al}\left\{ \sum_{ij} P_{mi}\, P^{*}_{nj}\, \dot{\underline{S}}^{\dagger}_{m}\, E\left[\delta \underline{S}_{i\perp}\, \delta \underline{S}^{\dagger}_{j\perp} \right] \dot{\underline{S}}_{n} \right\} \qquad (C.2)$$

By substitution of (44) in (C.2) we obtain :

$$H_{mn} = 2\frac{\sigma^2}{Q}\,\mathrm{Re\,al}\left(\dot{\underline{S}}^{\dagger}_{m}\, \Pi_B\, \dot{\underline{S}}_{n} \right) \sum_{ij} P_{mi} \left[P^{-1} + \sigma^2 P^{-1} \left(S^{\dagger} S \right)^{-1} P^{-1} \right]_{ij} P_i\, P^{*}_{nj} \qquad (C.3)$$

The summation on i and j may be interpreted in terms of matrix product, so that :

$$H_{mn} = 2\frac{\sigma^2}{Q} \left\{ \mathrm{Re\,al}\left(P_{mn}\, \dot{\underline{S}}^{\dagger}_{m}\, \Pi_B\, \dot{\underline{S}}_{n} \right) + \sigma^2\, \mathrm{Re\,al}\left(\left[\left(S^{\dagger} S \right)^{-1} \right]_{mn} \dot{\underline{S}}^{\dagger}_{m}\, \Pi_B\, \dot{\underline{S}}_{n} \right) \right\} \qquad (C.4)$$

From (63) the first term in the brackets is equal to Δf_{mn} so that (C.4) may be written :

$$H = \frac{\sigma^2}{Q}[\Delta f + 2\sigma^2 H'] \qquad (C.5)$$

$$H'_{mn} = Real\left\{\left[(S^{\dagger}S)^{-1}\right]_{mn} \dot{\underline{S}}_m^{\dagger} \Pi_B \dot{\underline{S}}_n \right\}$$

The variance expression using (64),(C.5), is then:

$$E[\delta \underline{\theta}\, \delta \underline{\theta}^{\dagger}] = \frac{\sigma^2}{Q}[2\Delta f^{-1} + 4\sigma^2 \Delta f^{-1} H' \Delta f^{-1}] \qquad (C.6)$$

In function of the coherence matrix, using the definitons (36),(11) of diagonal matrices **W** and $\mathbf{P_d}$, the Hessian (63) may be written :

$$\Delta f = 2(\mathbf{W}\mathbf{P}_d)^{1/2} \rho_{CR} (\mathbf{W}\mathbf{P}_d)^{1/2} \qquad (C.7)$$

where ρ_{CR} is the modified coherence matrix (35).

By substitution of (C.7) in (C.6) the second terms may be written:

$$\Delta f^{-1} H' \Delta f^{-1} = \frac{1}{4}\left\{ (WP_d)^{-1/2}\left[\rho_{CR}^{-1}(WP_d)^{-1/2} H'(WP_d)^{-1/2}\rho_{CR}^{-1}\right](WP_d)^{-1/2}\right\}$$

$$= \frac{1}{4} W^{-1/2} P_d^{-1}\left\{\left(P_d^{1/2}\rho_{CR}^{-1}P_d^{-1/2}\right)\left(W^{-1/2}H'W^{-1/2}\right)\left(P_d^{-1/2}\rho_{CR}^{-1}P_d^{1/2}\right)\right\} W^{-1/2}P_d^{-1}$$

(C.8)

Let us define the two matrices:

$$C_{ML}^{-1} = P_d^{-1}\,\rho_{CR}^{-1}\,P_d^{-1}$$

(C.9)

$$S_{ML}^{-1} = (W)^{-1/2}\,H'\,(W)^{-1/2}$$

Then from definition in (C.8) of H' and (34) of $\cos\beta_{mn}$ the elements of S_{ML}^{-1} are :

$$S_{ML}^{-1}{}_{mn} = \mathrm{Re\,al}\frac{\dot{\underline{S}}_m^{\dagger}\,\Pi_B\,\dot{\underline{S}}_n}{\sqrt{\dot{\underline{S}}_m^{\dagger}\,\Pi_B\,\dot{\underline{S}}_m}\sqrt{\dot{\underline{S}}_n^{\dagger}\,\Pi_B\,\dot{\underline{S}}_n}}\left[S^{\dagger}S\right]_{mn}^{-1} = \left[S^{\dagger}S\right]_{mn}^{-1}\cos\beta_{mn}\cos\gamma_{mn}$$

(C.10)

$$\gamma_{mn} = \arg\left\{\dot{\underline{S}}_m^{\dagger}\,\Pi_B\,\dot{\underline{S}}_n\left[S^{\dagger}S\right]_{mn}^{-1}\right\}$$

Using these definitions in (C.8) we find :

$$\Delta f^{-1} H' \Delta f^{-1} = \frac{1}{4} W^{-1/2} P_d^{-1} D_{ML}^{-1} P_d^{-1} W^{-1/2} \qquad \text{(C.11)}$$

with :

$$\mathbf{D}_{ML}^{-1} = \text{Real}\{\mathbf{C}_{ML}^{-1}\mathbf{S}_{ML}\mathbf{C}^{\dagger}{}_{ML}^{-1}\} \tag{C.12}$$

By substitution of (C.7) and (C.11) in (C.6) we then obtain:

$$E[\delta\underline{\theta}\delta\underline{\theta}^{\dagger}] = \frac{\sigma^2}{2Q}(\mathbf{P}_d\mathbf{W})^{-1/2}\left[\rho_{CR}^{-1} + \sigma^2\mathbf{P}_d^{-1}\mathbf{D}_{ML}^{-1}\mathbf{P}_d^{-1}\right](\mathbf{P}_d\mathbf{W})^{-1/2} \tag{C.13}$$

or , on the diagonal :

$$E\left[\left(\hat{\theta}_m - \theta_m\right)^2\right]_{AML} = \frac{\sigma^2}{2Qp_m\left(\dot{\underline{S}}_m^{\dagger}\,\Pi_B\,\dot{\underline{S}}_m\right)}\left[\left[\rho_{CR}^{-1}\right]_{mm} + \frac{\sigma^2}{p_m}\left[\mathbf{D}_{ML}^{-1}\right]_{mm}\right] \tag{C.14}$$

APPENDIX D

Using a derivation very similar to that of the basic algorithm, we find:

$$\underline{\nabla} f_m = 2\,\mathrm{Re\,al}\left\{ \mathrm{Tr}\left[\hat{\Pi}_B \left(\frac{\partial \Pi_S}{\partial \theta_m} \right) \hat{R}_y \, \Pi_S (\underline{\theta}) \right] \right\} \qquad (D.1)$$

$$\Delta f_{mn} = 2\,\mathrm{Re\,al}\left\{ \mathrm{Tr}\left[\hat{\Pi}_B \left\{ \frac{\partial \Pi_S}{\partial \theta_m} \hat{R}_y \frac{\partial \Pi_S}{\partial \theta_n} + \frac{\partial^2 \Pi_S}{\partial \theta_m \partial \theta_n} \hat{R}_y \, \Pi_S (\underline{\theta}) \right\} \right] \right\} \qquad (D.2)$$

or, according to the fact that , at convergence, $\hat{\Pi}_S \Pi_B \equiv 0$

$$\Delta f_{mn} \equiv 2\,\mathrm{Re\,al}\mathrm{Tr}\left[\hat{\Pi}_B \frac{\partial \Pi_S}{\partial \theta_m} \hat{R}_y \frac{\partial \Pi_S}{\partial \theta_n} \right] \qquad (D.3)$$

Let us compute the derivative of $\Pi_S(\theta)$. After reordering the terms, it can be written in the following form (variable θ is omitted for clarity) :

$$\frac{\partial \Pi_S}{\partial \theta_m} = S(S^\dagger S)^{-1} \dot{S}_m^\dagger (I - \Pi_S) + (I - \Pi_S) \dot{S}_m (S^\dagger S)^{-1} S^\dagger$$

$$\Pi_S = S(S^\dagger S)^{-1} S \qquad (D.4)$$

$$\dot{S}_m = \frac{\partial S}{\partial \theta_m}$$

By substitution in (D.3) and considering that, at convergence $S^\dagger \Pi_B \approx 0$ only one of the two terms of each derivative remains in the expression of Δf_{mn}, so that :

$$\Delta f_{mn} \cong 2\,\mathrm{Re\,al}\left\{ \mathrm{Tr}\left[\hat{\Pi}_B (I - \Pi_s)\, \dot{S}_m \,(S^\dagger S)^{-1} S^\dagger \hat{R}_y S (S^\dagger S)^{-1}\, \dot{S}_n^\dagger \,(I - \Pi_s) \right] \right\}$$

$$\Delta f_{mn} \cong 2\,\mathrm{Re\,al}\left\{ \mathrm{Tr}\left[\hat{\Pi}_B (I - \Pi_s)\, \dot{S}_m \hat{P}\, \dot{S}_n \,(I - \Pi_s) \right] \right\} \qquad \text{(D.5)}$$

$$\hat{P} = (S^\dagger S)^{-1} S^\dagger \hat{R}_y S (S^\dagger S)^{-1}$$

Introducing an approximation in the gradient is more questionable since it may modify the convergence point. Neverthless let us substitute (D.4) in (D.1) . We find (after conjugation of the 2nd term):

$$\nabla f_m \cong 2\,\mathrm{Re\,al}\left\{ \mathrm{Tr}\left[\hat{\Pi}_B (I - \Pi_s)\, \dot{S}_m \,(S^\dagger S)^{-1} S^\dagger \hat{R}_y S (S^\dagger S)^{-1} S^\dagger \right] \right\}$$

$$+ 2\,\mathrm{Re\,al}\left\{ \mathrm{Tr}\left[\hat{R}_y (I - \Pi_s)\, \dot{S}_m \,(S^\dagger S)^{-1} S^\dagger \hat{\Pi}_B S (S^\dagger S)^{-1} S^\dagger \right] \right\} \qquad \text{(D.6)}$$

In order to save computation time, we have considered that the second term could be neglected without loss of performance . The justification is that for a small perturbation of Π_B, the first term turn out to be of the 1st order in the perturbation, while the other one is of the 3rd order. Let us demonstrate this point shortly .

We denote by U the N.M matrix of the M first source eigenvectors, and D the diagonal matrix of the first M eigenvalues:

$$U = [U_1, .., U_M]$$

$$D = \mathrm{diag}\,[l_1, .., l_M] \qquad \text{(D.7)}$$

According to the analysis of section IV. B, in the first order approximation, the matrix U of the M perturbed source vectors may be written:

$$\hat{U} \approx U + \delta U_\perp$$

$$\Pi_S \delta U_\perp = 0 \qquad \text{(D.8)}$$

The small perturbation δU may always be written in terms of a normalized matrix Δ defined by:

$$\begin{aligned} &\delta U_{\perp} = \varepsilon \Delta \\ &Tr[\Delta^2] = M \\ &\Pi_S \Delta = 0 \\ &\varepsilon << 1 \end{aligned} \qquad (D.9)$$

where ε gives a measure in radian of the average random rotation of the eigenvectors .

Using these notations we etablish that:

$$\begin{aligned} \hat{\Pi}_B &= I - \hat{U}\hat{U}^{\dagger} \cong \Pi_B - \varepsilon[(U\Delta^{\dagger} + \Delta U^{\dagger})] \\ R_y &= \hat{U}D\hat{U}^{\dagger} \cong UDU + \varepsilon[UD\Delta^{\dagger} + \Delta DU^{\dagger}] \end{aligned} \qquad (D.10)$$

We will calculate the gradient at the exact sources position so that $\Pi_S(\theta)=\Pi_S$. Note that:

$$S^{\dagger}\hat{\Pi}_B S = S^{\dagger}\hat{\Pi}_B\hat{\Pi}_B S^{\dagger} \cong \varepsilon^2 (S^{\dagger}U)\Delta^{\dagger}\Delta(U^{\dagger}S) \qquad (D.11)$$

By substitution in (D.6) after all reduction and reordering we find for the two terms the expressions:

$$\begin{aligned} \nabla f_m \cong\ & 2\,Real\left\{ Tr\left[-\varepsilon\hat{\Pi}_B \dot{S}_m (S^{\dagger}S)^{-1}(S^{\dagger}U)\, I_M\, D\Delta^{\dagger} \right] \right\} \\ & + 2\,Real\left\{ Tr\left[\varepsilon^3\hat{\Pi}_B \dot{S}_m (S^{\dagger}S)^{-1}(S^{\dagger}U)\, \Delta^{\dagger}\Delta\, D\Delta^{\dagger} \right] \right\} \end{aligned} \qquad (D.12)$$

The two matrices are almost identical, the MM identity matrix I_M being replaced by the MM matrix $\Delta^{\dagger}\Delta$,with the same norm equal to M according to (D.9). The difference is in the coefficient, -ε for

the first, ϵ^3 for the second . This justify our approximation of the gradient by the first term of (D.6) which may be written:

$$\nabla f_m \equiv 2\mathrm{Re\,al}\left\{\mathrm{Tr}\left[S^{\dagger}\hat{\Pi}_B(I-\Pi_S)\,\dot{S}_m\,\hat{P}\right]\right\} \qquad (D.13)$$

$$\hat{P} = (S^{\dagger}S)^{-1}S^{\dagger}R_y S(S^{\dagger}S)^{-1}$$

REFERENCES

[1] R. Schmit, "A signal subspace approach to multiple emitter location and spectral estimation", P.H.D. Dissertation, Stanford, Nov., 1981.

[2] G. Bienvenue , L Kopp, " Principe de la goniométrie passive adaptative", Proc. Colloque GRETSI, pp 106-116 , Nice, juin 1979 .

[3] B. Porat , B Friedlander , " Analysis of the asymptotic relative efficiency of the MUSIC algorithm", IEEE Trans. Acous., Speech, and Signal Processing VASSP 36 N° 4 April 1988,pp. 532-543 .

[4] A. Ouamri, S.Tressens ,H. Clergeot ," Comparison of high resolution spectral methods based on SVD" , Proc of EUSIPCO-86,Sept 86, pp.337-340 .

[5] A. Ouamri,"Etude de performances des méthodes à haute résolution et application à l'identification des échos par une antenne linèaire . Thèse d'Etat, Orsay 1986.

[6] P. Stoica, A. Nehoari, "MUSIC,Maximum Likelihood and Cramer Rao Bounds" , IEEE Trans. Acous., Speech, and Signal Processing VASSP 37-N° 5, May 1989, pp. 720-741.

[7] T.J. Shan, T. Kailath, "New adaptive processor for coherent signal interferences", in Proc of ICASP-84, pp.3351-3354, 1984 .

[8] H. Clergeot,A.Ouamri,S. Tressens, "High resolution spectral methods for spatial discrimination of closely spaced correlated sources", Proc ICASSP-85, pp. 560-563,1985 .

[9] D.C. Rife and R.R. Boorstyn," Multiple tone parameters estimation from discrete time observations", Bell System Tech. Journal Vol 55,pp 1389-1410 , Nov. 1976 .

[10] R. Kumaresan, L. L. Scharf , A. K Shaw, "An algorithm for pole-zero Modeling and Spectral Analysis",IEEE Trans. Acous., Speech, and Signal Processing VASSP 34 N° 3, June 1986,pp. 637-640.

[11] S.Tressens , H. Clergeot , A. Ouamri ,M. Buvet, " More on the optimality of EVD methods for bearing estimation", Proc. ICASSP-88 , pp 2881-2884 .

[12] H. Clergeot , S.Tressens , A. Ouamri, "A new Maximum Likelihood Method for estimation of correlated sources. Comparison with existing methods".Proc of EUSIPCO-87, Sept 87, Florence pp. 71-75 .

[13] H. Clergeot , S.Tresssens , A. Ouamri, " Performances of high resolution spectral methods compared to the Cramer Rao Bounds", to be published nov 1989 , IEEE Trans. Acous., Speech. and Signal Processing .

[14] D.R. Brillinger , Time Series-Data Analysis Theory . New York ,Holt ,Reinhart and Winston, 1975 .

FIGURE CAPTION

Table I- Theoretical variances.

Fig.1 -Variation of cos β_{12} as a function of source separation compared to the Fourier resolution.

Fig. 2 -Theoretical variances for $\rho=0$. Comparison with Monte Carlo results.

Fig.3 - Theoretical variances for $\rho=0.9$, $\Delta f = 1$. Comparison with Monte Carlo results.

Fig.4 - Theoretical variances for $\rho=0.9$, $\Delta f = 0.063$. Comparison with Monte Carlo results

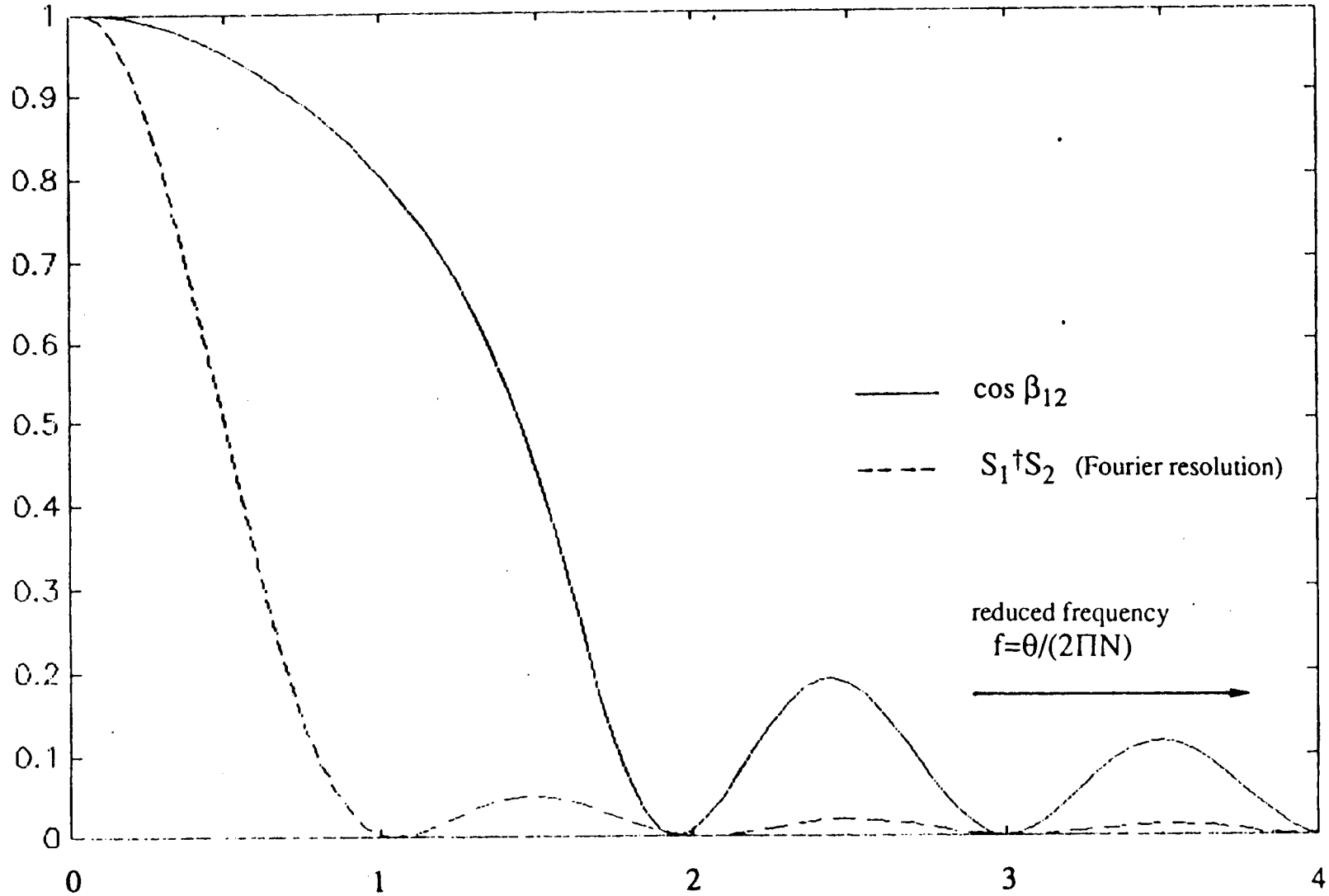
Fig.5 - Contour plots without noise for AMLM, RAMLM, $\rho=0.9$. For $\rho=0$: the same as in c and d ($\alpha = \Pi/2$) . For MUSIC, any ρ for α: the same as d .

Fig.6 - Convergence of MUSIC, RAMLM,AMLM for three initialization points.

Fig.7 - Typical contours plots in presence of noise ($\Delta f=0.063$, SNR = 50 dB , $\rho=0.9$)

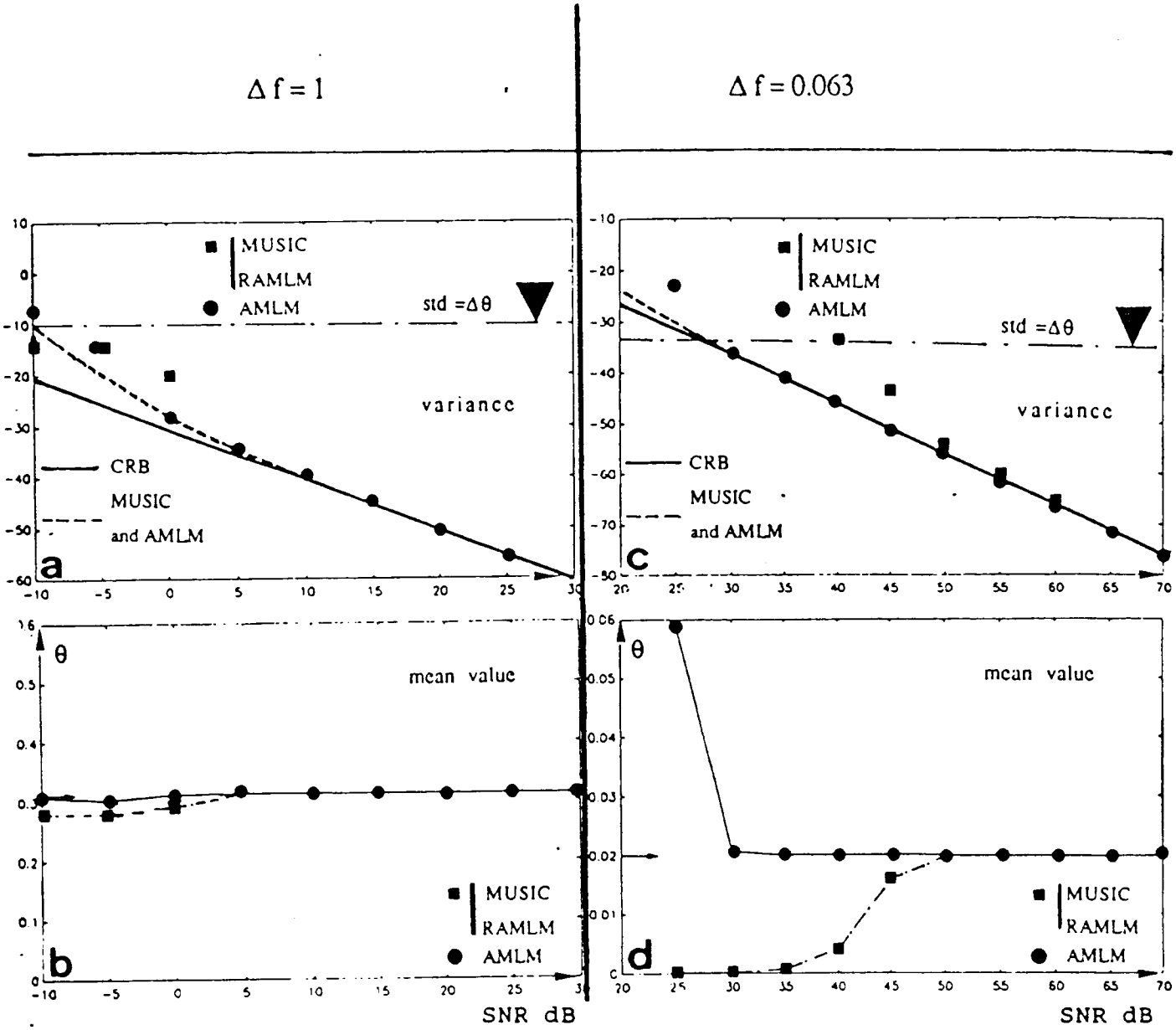
Fig.8 - Distribution of (θ_1, θ_2) estimates for 100 Monte Carlo trials ($\rho=0.9$, SNR= 50 dB, $\Delta f=0.063$)

Fig.9 -Performance of AMLM for one snapshot (Q=1).



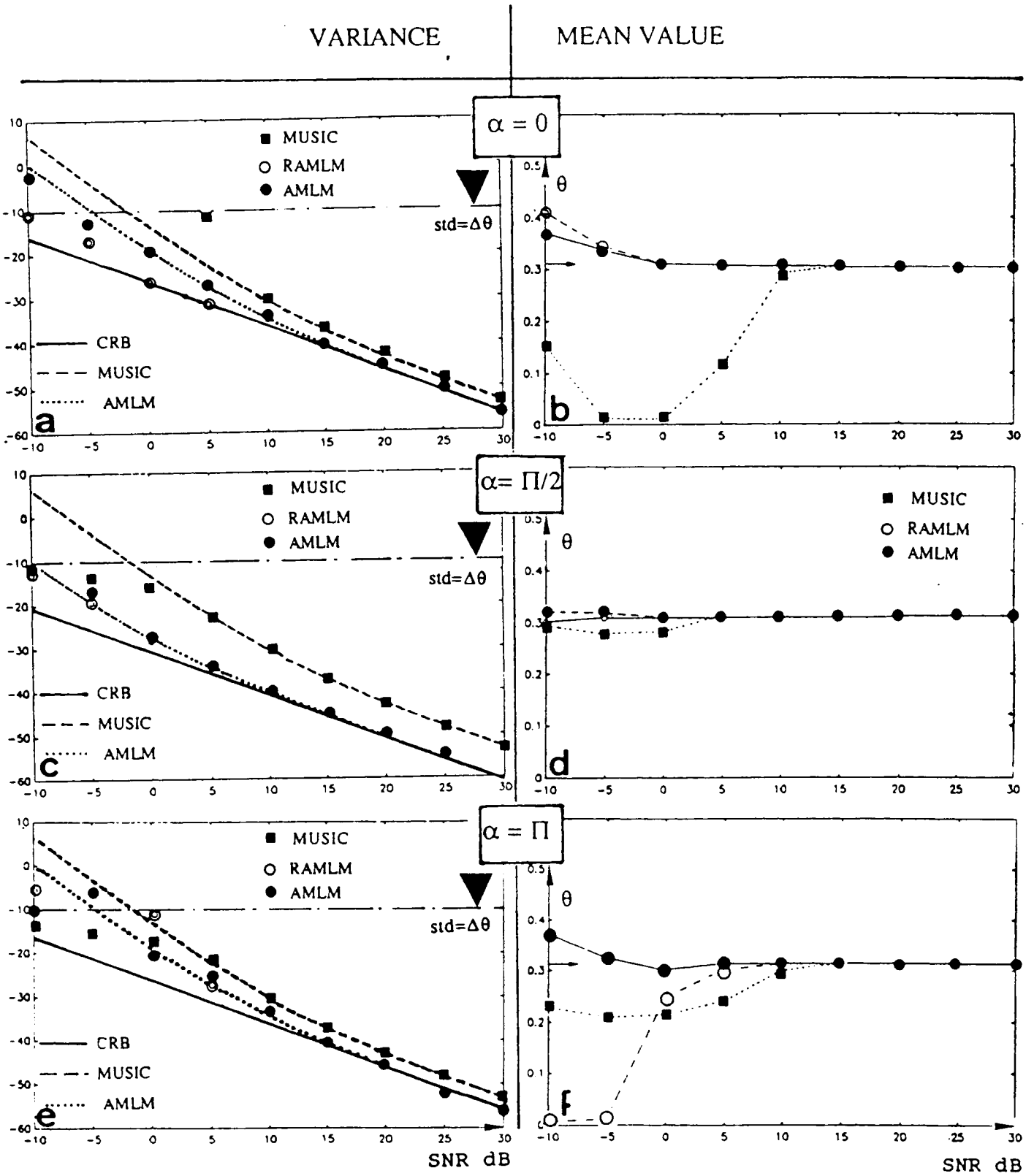
Variation of cos β_{12} as a function of source separation compared to the Fourier resolution

Figure 1



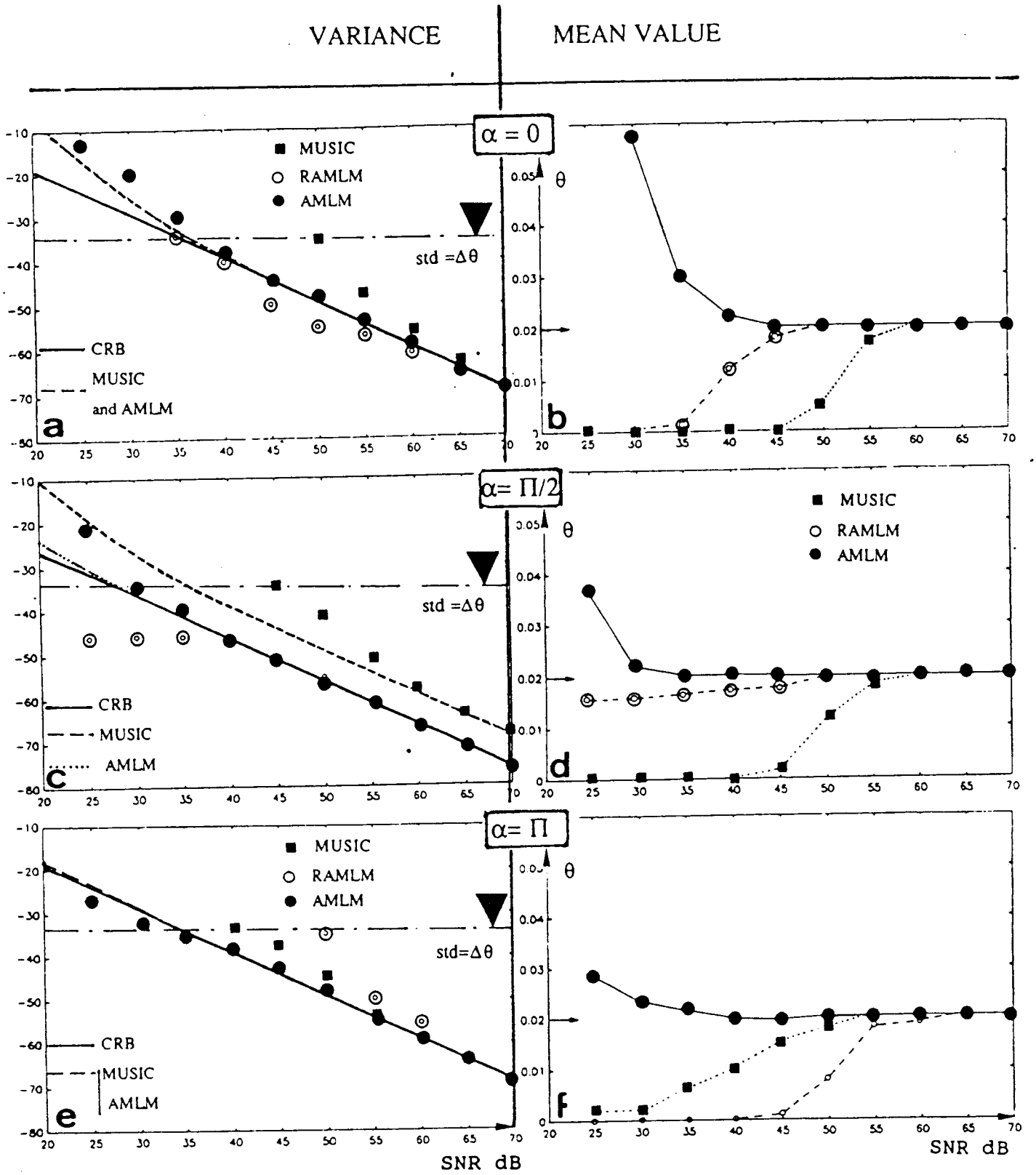
Theoretical variances for ρ=0 . Comparis on with Monte Carlo results

Figure 2



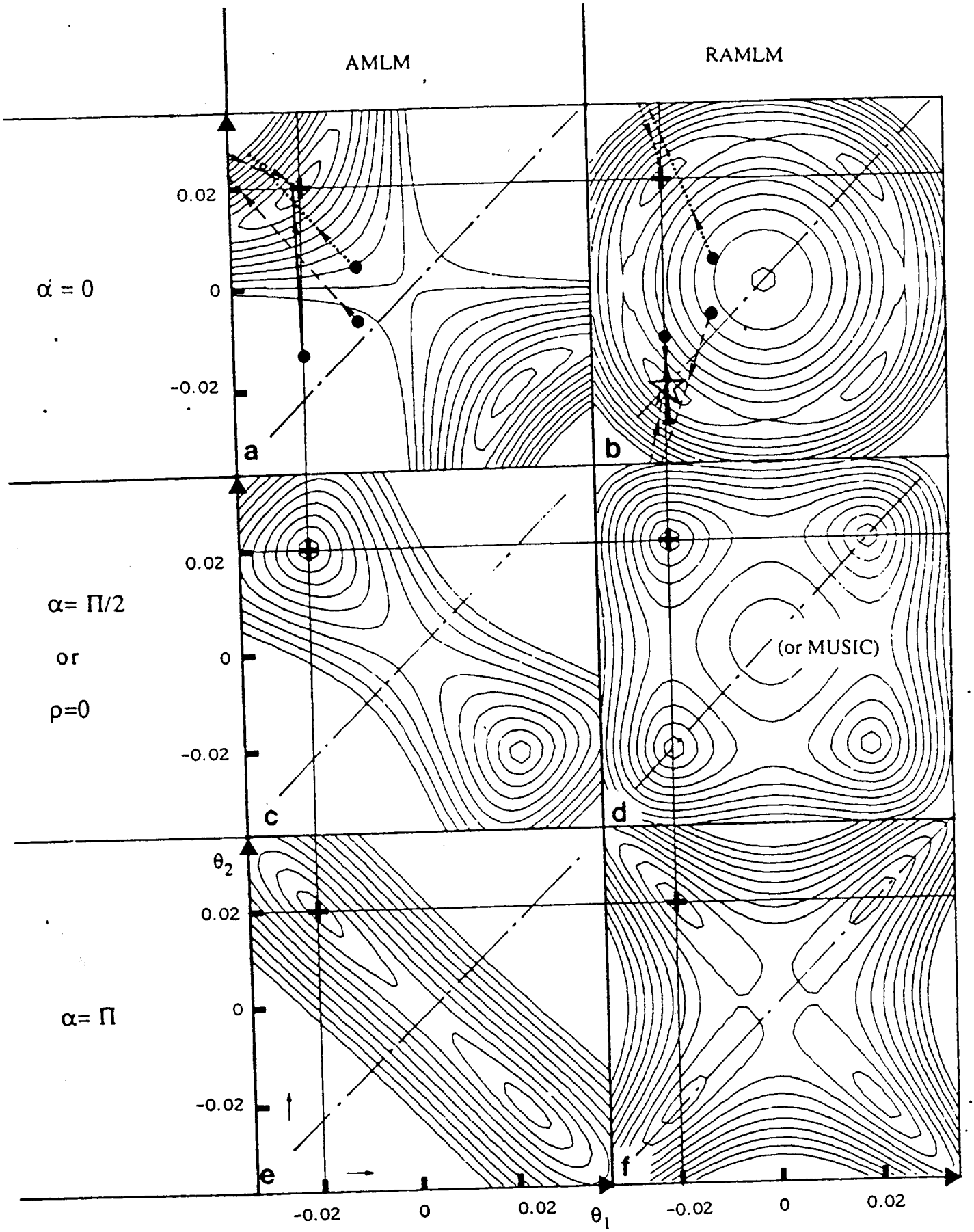
Theoretical variances for ρ=0.9 , Δ f =1. Comparis on with Monte Carlo results

Figure 3



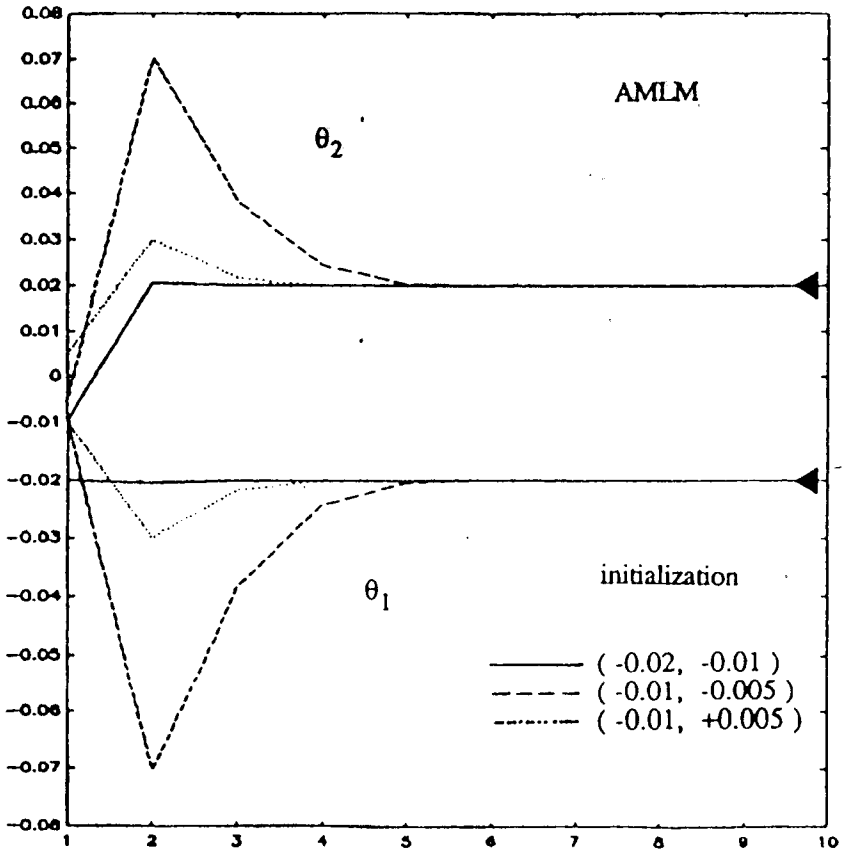
Theoretical variances for ρ=0.9 , Δ f =0.063. Compar ;ion with Monte Carlo results

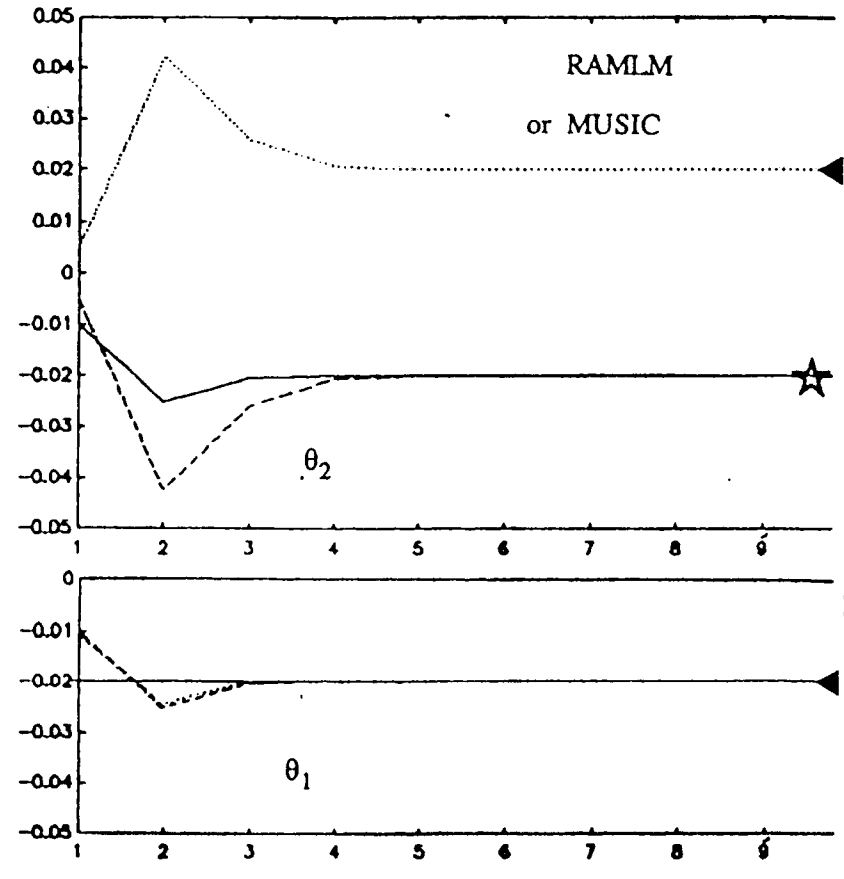
Figure 4



Contour plots without noise for AMLM, RAMLM, ρ=0.9 . For ρ=0: the same as in c and d (α= Π/2) . For MUSIC, any ρ or α : the same as d..

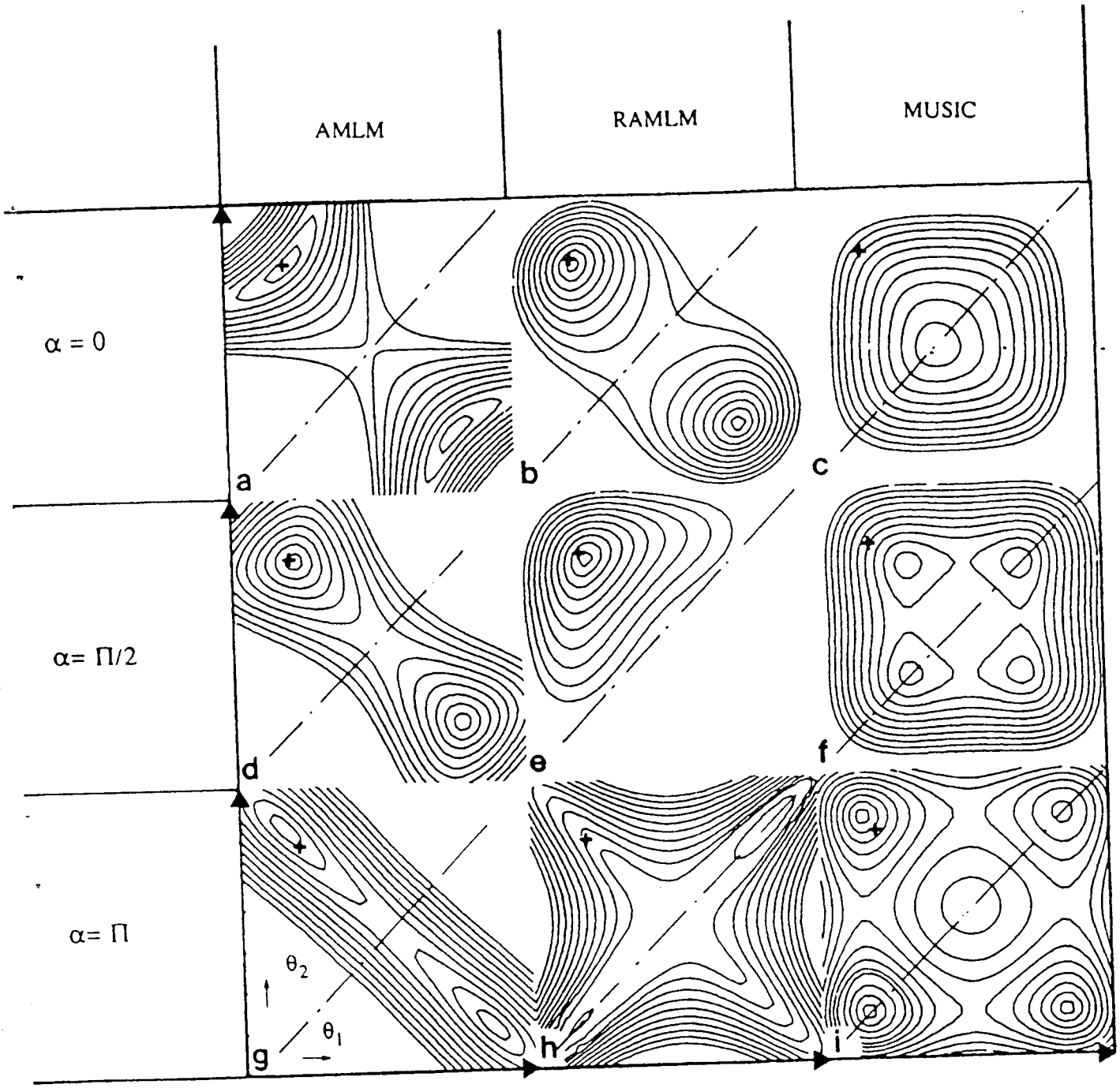
Figure 5





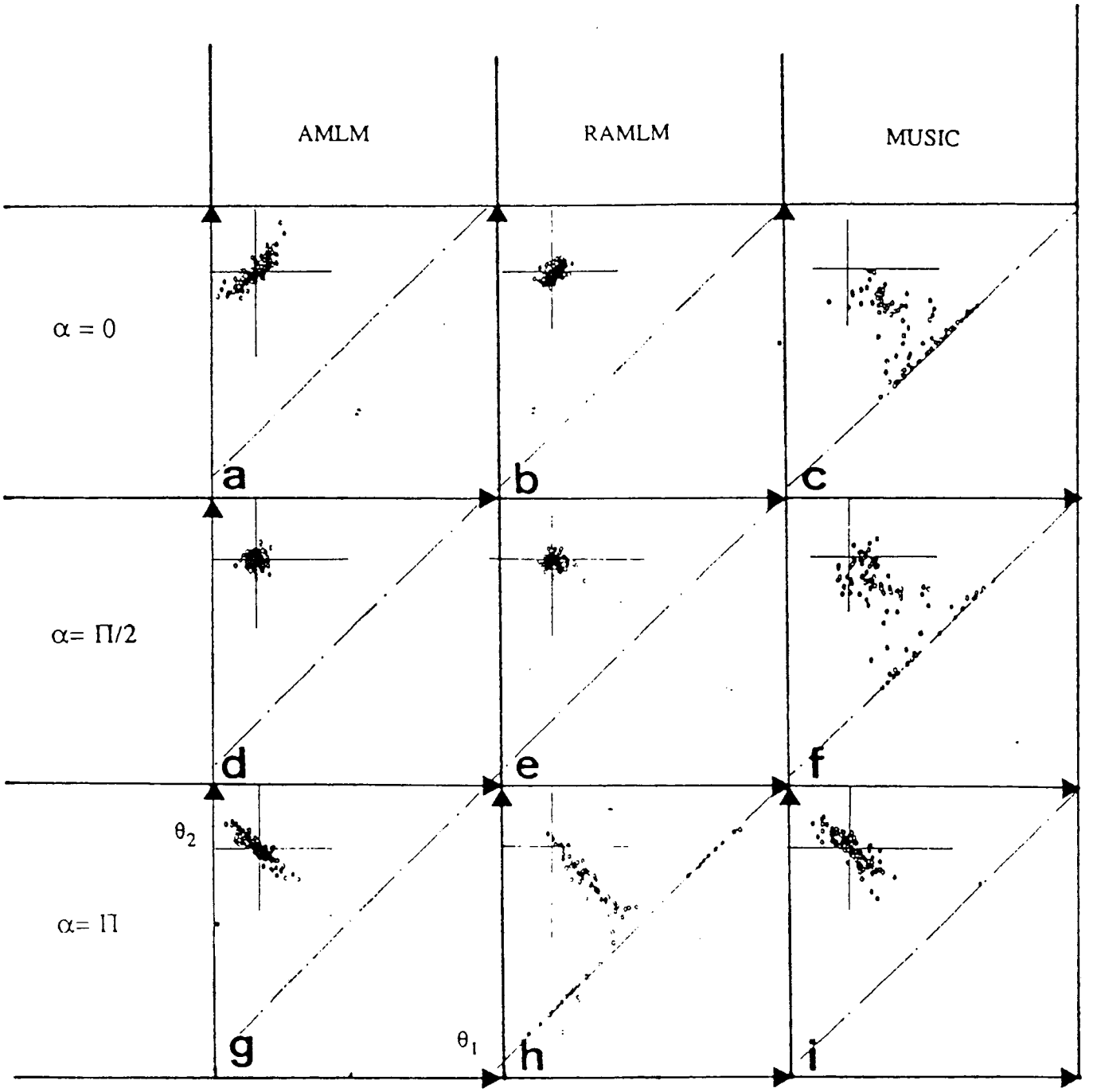
Convergence of MUSIC, RAMLM,AMLM for three initialization points.

Figure 6



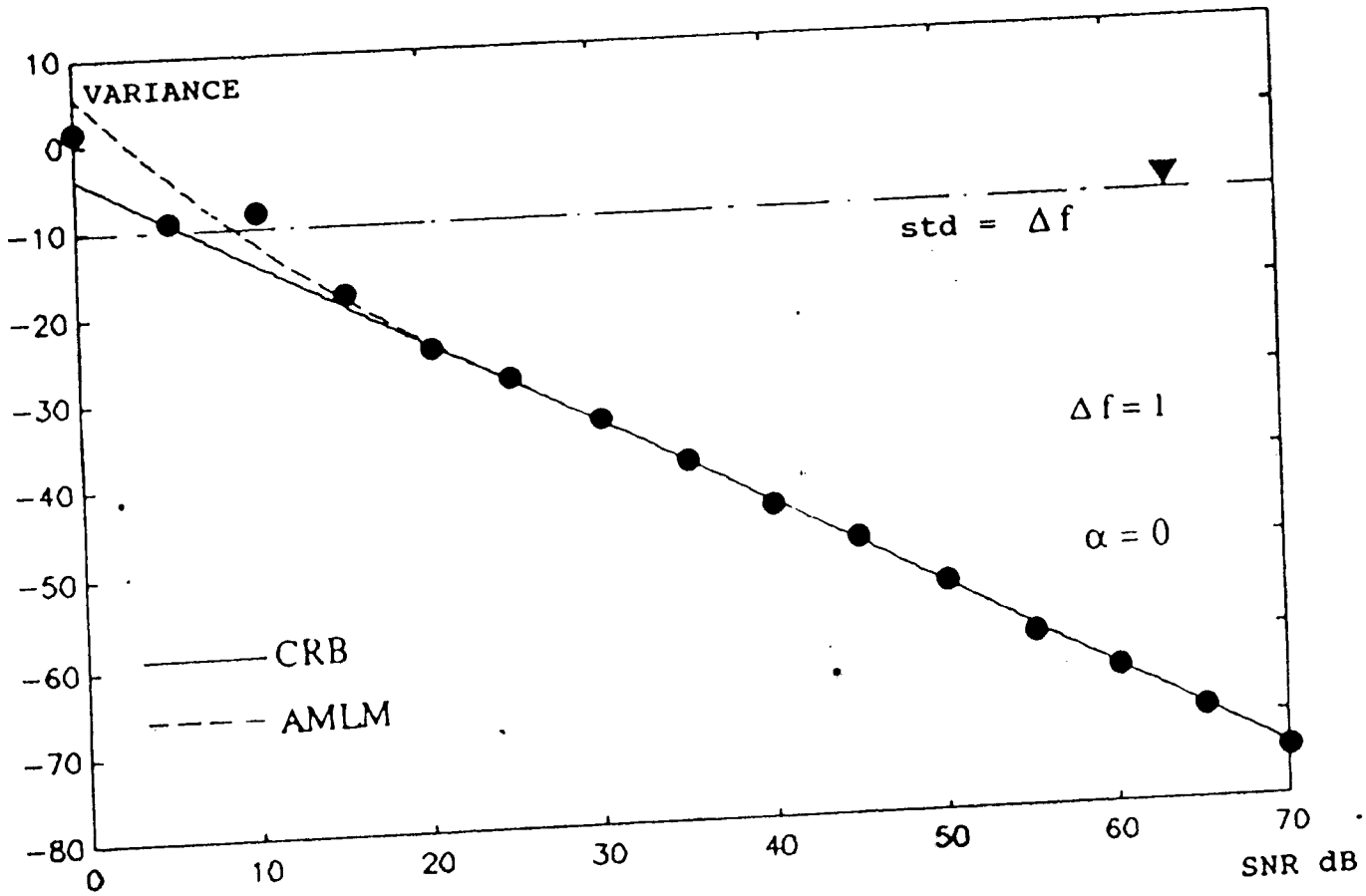
Typical contours plots in presence of noise (Δf=0.063 , SNR = 50 dB , ρ=0.9)

Figure 7



Distribution of (θ_1, θ_2) estimates for 100 Monte Carlo trials (ρ=0.9, SNR= 50 dB, Δf=0.063)

Figure 8



Performance of AMLM for one snapshot (Q=1)

Figure 9

ARTICLE 3

An asymptotically unbiased estimator for the poles of a rational transfer function,

par S.MAYRARGUE

soumis en Septembre 1989 à IEEE Trans. ASSP

AN ASYMPTOTICALLY UNBIASED ESTIMATOR FOR THE POLES OF A RATIONAL TRANSFER FUNCTION

Sylvie Mayrargue
CNET/PAB/RPE
38-40 rue du Général Leclerc
92131 Issy-les-Moulineaux
France

Abstract A recent correspondence compares several methods aiming to decrease the bias in the Kumaresan-Tufts (KT) algorithm for estimating the parameters of rational transfer functions in additive white noise [1] . By means of Monte-Carlo simulations, it was shown that the best method in most cases was the so-called "Bias Compensated KT method" (BCKT) .

We show that another variant of KT, the Improved Pisarenko (IP) method [2] is theoretically asymptotically unbiased. We also show that the difference between KT and IP is that a linear system is solved in the least square (LS) sense for KT, and in the total least square (TLS) [3],[4] sense for IP.This last result was already partially shown in [5].We show that BCKT and IP are asymptotically identical.We present simulation results for the three methods.

I-INTRODUCTION

This paper deals with estimating the parameters of damped exponentials, or equivalently the poles of rational transfer functions, when the signal is buried in white noise.

Several methods performing this estimation exist, among which the KT method [2], [6]. However, it has been acknowledged that this method is biased. Several attempts have been made to decrease the bias, giving rise to a number of variants. A recent correspondence compares these variants [1]. By means of Monte-Carlo simulations, the best method in most cases was shown to be the "Bias Compensated Kumaresan and Tufts method" (BCKT) [7].

In this correspondence, we first prove theoretically the existence of the bias of the KT method. We show that it is due to the use of LS method for solving a linear system both sides of which are corrupted by noise. We then recall the IP method [2], which is a variant of KT and was not included in the comparisons of [1]. We show that it is asymptotically unbiased due to the fact that IP solves the same linear system as KT, using TLS instead of LS.That TLS could be applied to solve the linear system in KT was already noticed by [5]. However, the authors did not realize that the resulting method was the IP method of [2]. Then we use properties of TLS [4] to show that IP is asymptotically identical to BCKT.

We compare KT, BCKT and IP by Monte-Carlo simulations, using the example given in [1]. As expected, KT is biased, while BCKT and IP are both almost unbiased.

II-PROBLEM POSITION

Let us assume that N data points of the following form are given :

$y_k = h_k + \varepsilon_k \qquad 1 \le k \le N$

where ε_k is a zero-mean Gaussian noise with variance σ^2, and h_k is the impulse response of the rational transfer function

$$H(z) = \frac{d_1 z^{M-1} + \ldots + d_M}{z^M + a_1 z^{M-1} + \ldots + a_M} = \frac{d(z)}{\prod_{i=1}^{M}(z-\mu_i)} \qquad (1)$$

(note that the poles μ_i are assumed to be simple).

Since the degree of the numerator is smaller than the degree of the denominator, we can write, with λ_i the residue of H(z) at μ_i .

$$H(z) = \sum_{i=1}^{M} \frac{\lambda_i}{(z-\mu_i)}$$

We have :

$$H(z) = \sum_{i=1}^{M} \lambda_i z^{-1} \sum_{n=0}^{\infty} (\mu_i z^{-1})^n$$

Hence ,

$$h_k = \sum_{i=1}^{M} \lambda_i \mu_i^{k-1}$$

Thus we can see that the initial problem of retrieving the poles of a rational transfer function from a finite sample of its impulse response embedded into white noise is identical to the identification of damped exponentials embedded into white noise. Since H(z) is assumed to be a stable filter, the μ_k will lie inside the unit circle so that the h_k are samples of damped exponentials.

III-TUFTS AND KUMARESAN'S (KT) METHOD

III-1 Description of the Method

KT method applied to damped exponentials has been presented [2] , [6]. We briefly recall this method.

Consider the following linear system (backward direction), with $M \leq L \leq N-M$:

$$\frac{1}{\sqrt{N-L}} \begin{bmatrix} y_2 & y_3 & \ldots & y_{L+1} \\ y_3 & y_4 & \ldots & y_{L+2} \\ & & & \\ y_{N-L+1} & & & y_N \end{bmatrix} \begin{bmatrix} \beta_1 \\ \\ \\ \beta_L \end{bmatrix} = - \frac{1}{\sqrt{N-L}} \begin{bmatrix} y_1 \\ \\ \\ y_{N-L} \end{bmatrix} \qquad (2a)$$

or : $Ab = -\alpha$ **(2b)**

(The reason for the normalization factor $1/\sqrt{N-L}$ will appear later).

Without noise, this system would be :

$$\frac{1}{\sqrt{N-L}} \begin{bmatrix} h_2 & h_3 & \ldots & h_{L+1} \\ h_3 & h_4 & \ldots & h_{L+2} \\ & & & \\ h_{N-L+1} & & & h_N \end{bmatrix} \begin{bmatrix} \beta_1^0 \\ \\ \\ \beta_L^0 \end{bmatrix} = - \frac{1}{\sqrt{N-L}} \begin{bmatrix} h_1 \\ \\ \\ h_{N-L} \end{bmatrix} \qquad (3a)$$

or : $A_0 b^0 = - \alpha_0$ (3b)

It is well-known that A_0 has rank M. Hence A_0 is singular. Therefore (3) has an infinite number of solutions, from which the one with minimum Euclidean norm is selected by the KT method:

$b_{KT} = - A_0^{\#} \alpha_0$ (# : pseudo-inverse).

Let $A_0 = \sum_{k=1}^{\inf(L,N-L)} \sigma_k^0 u_k^0 v_k^{0+}$

(+ means conjugate transpose)

be the singular value decomposition (SVD) of A_0 . σ_k^0 are the singular values, v_k^0 and u_k^0 the right and left singular vectors, respectively. Since $\sigma_{M+1}^0 = \ ..... \ = \sigma_{\inf(L,N-L)}^0 = 0$, we can write

$$A_0 = \sum_{k=1}^{M} \sigma_k^0 u_k^0 v_k^{0+}$$

and $$b_{KT} = - \sum_{k=1}^{M} \sigma_k^{0\,-1} \ (u_k^{0+} \ \alpha_0) \ v_k^0$$

Let $b_{KT} = [\beta_1^{KT} , \ ..... \ , \ \beta_L^{KT}]$, and consider the polynomial B(z) given by

$$B(z) = z^L + \beta_1^{KT} \ z^{L-1} + \ .... \ + \beta_L^{KT} .$$

B(z) has exactly M roots outside the unit circle (i.e. $\{ \mu_k^{-1} \}$). The remaining L-M roots lie within the unit circle.

In the case of noisy data, A is generally full-rank. In order to take into account the underlying rank M structure of A_0 , A is replaced by the nearest (in the Frobenius sense) rank M matrix, which is (Eckart-Young theorem) :

$$\hat{A} = \sum_{k=1}^{M} \sigma_k u_k v_k^{+}$$

where σ_k are the M largest singular values of A, v_k (resp. u_k)the associated right (resp. left) singular vectors.

(2b) thus becomes $\hat{A} b = - \alpha$ (4)

The minimum-norm solution of (4) is given by

$$\hat{b}_{KT} = - \sum_{k=1}^{M} \sigma_k^{-1} \ (u_k^{+} \ \alpha_0) \ v_k \qquad (5)$$

A polynomial $\hat{B}(z)$ is built as above using the coefficients of $\hat{b}_{KT}$. The M largest roots are selected . They are taken as estimates of $\{ \mu_k^{-1} \}$ and denoted by $\{ \hat{\mu}_k^{-1} \}$.

III-2 $\hat{b}_{KT}$ is an asymptotically biased estimator of b_{KT}

Let us first provide another expression for $\hat{b}_{KT}$. From the definition of SVD, we know that

$u_k = A v_k / \sigma_k$

Thus : $$\hat{b}_{KT} = - \sum_{k=1}^{M} \frac{v_k}{\sigma_k^2} \ v_k^{+} \ (A^{+}\alpha) \qquad (6)$$

(Note that $\hat{b}_{KT} = - (\hat{A}^{+}\hat{A})^{\#} A^{+}\alpha = - (\hat{A}^{+}\hat{A})^{\#} \hat{A}^{+}\alpha$)

We prove in Appendix A that , when the amount of data N tends to infinity :

$A^{+}\alpha \ \rightarrow \ A_0^{+}\alpha_0$

$v_k \ \rightarrow \ v_k^0$ k=1, ... , M almost surely (a.s.)

but $\sigma_k^2 \rightarrow (\sigma_k^0)^2 + \sigma^2$

so that $\hat{b}_{KT} \not\rightarrow b_{KT}$

$\hat{b}_{KT}$ is thus an asymptotically biased estimator of b_{KT}, and the roots of $\hat{B}(z)$, the associated polynomial, are biased estimates of μ_k^{-1}. Note that here "asymptotically" means that the number of samples N tends to infinity , while L (the length of b) remains a constant.. This may seem contradictory to the assertion of Kumaresan [8] and other authors , who prescribe to choose L proportional to N in order to optimize the variance of the $\hat{\mu}_k$. However, this contradiction is only apparent, since L is anyway limited by the computational load that would become quickly untractable for large L.

The existence of a bias in the solution of (2) is to be related to a known result for linear regression , namely that the LS solution to a linear system with "errors-in-variables" (i.e. with noisy measurements on both sides) is asymptotically biased [4],[9].

IV-IMPROVED PISARENKO (IP) METHOD

The IP method was first presented by Kumaresan [2] , [10]. Let us give an outline of the method. The range of L is again chosen between M and N-M.

Consider the following homogeneous linear system :

$$\begin{bmatrix} y_1 & y_2 & & y_{L+1} \\ y_2 & y_3 & & y_{L+2} \\ & & & \\ & & & \\ y_{N-L} & & & y_M \end{bmatrix} \begin{bmatrix} 1 \\ \beta_1 \\ \\ \\ \beta_L \end{bmatrix} = [0] \qquad (7a)$$

or $A'b' = 0$ (7b)

where A' denotes the augmented matrix [α | A] with α and A the variables of Section III . Throughout this paper, a vertical bar will be used to separate submatrices or vectors components.

Without noise, this system is :

$$\begin{bmatrix} h_1 & h_2 & & h_{L+1} \\ h_2 & h_3 & & h_{L+2} \\ & & & \\ & & & \\ h_{N-L} & & & h_M \end{bmatrix} \begin{bmatrix} 1 \\ \beta_1 \\ \\ \\ \beta_L \end{bmatrix} = [0] \qquad (8a)$$

or $A'_0 b' = 0$, (8b)

with $A'_0 = [\alpha_0 \mid A_0]$

A'_0 has rank M , hence is singular. Its nullspace is (L+1-M) dimensional. The solution b'_{IP} is taken as the minimum-norm vector having first component equal to 1, lying in this subspace.

Let v'^0_k be the right singular vectors of A'_0, with indices k such that $\{ v'^0_k , \quad k=M+1, \ldots , L+1\}$ span the nullspace of A'_0.

It is easy to see that the solution provided by the IP method is given by [2] , [10]

$$b'_{IP} = \frac{\sum_{k=M+1}^{L+1} \overline{v'^0_k(1)} v'^0_k}{\sum_{k=M+1}^{L+1} \left|v'^0_k(1)\right|^2} = \frac{e_1 - \sum_{k=1}^{M} \overline{v'^0_k(1)} v'^0_k}{1 - \sum_{k=1}^{M} \left|v'^0_k(1)\right|^2}$$

where

$v'^0_k(1)$ is the first component of v'^0_k , the horizontal bar denotes complex conjugation and $e_1 = [\,1, 0, \ldots , 0\,]^T$ where ^{T} means transpose.

Let $b'^T_{IP} = [\,1 \mid b^T_{IP}\,]$.

The definition of b'_{IP} as the minimum-norm solution of (8), leads to

$$b_{IP} = b_{KT} .$$

In the case of noisy data, A' is full-rank. A' is replaced, as was A in Section III, by its nearest rank-M approximation

$$\hat{A}' = \sum_{k=1}^{M} \sigma'_k u'_k v'^+_k$$

where σ'_k are the M largest singular values of $\hat{A}'$, v'_k (resp.u'_k) are the associated right (resp. left) singular vectors.With this modification, (7b) becomes

$$\hat{A}'b' = 0 \qquad (9)$$

The minimum-norm solution of (9) is given by

$$\hat{b}'_{IP} = \frac{\sum_{k=M+1}^{L+1} \overline{v'_k(1)}\, v'_k}{\sum_{k=M+1}^{L+1} \left|v'_k(1)\right|^2} = \frac{e_1 - \sum_{k=1}^{M} \overline{v'_k(1)} v'_k}{1 - \sum_{k=1}^{M} \left|v'_k(1)\right|^2} \qquad (10)$$

(Remark : Let $\hat{b}_{IP}$ be defined by $\hat{b}'^T_{IP} = [1 \mid \hat{b}^T_{IP}]$. $\hat{b}_{IP}$ is not equal to $\hat{b}_{KT}$)

Reasoning as in Appendix A, we can show that $v'_k \rightarrow v'^0_k$ a.s. when N tends to infinity. $\hat{b}_{IP}$ is thus an asymptotically unbiased estimate of b_{IP}.

Solving the homogeneous linear system (7) as performed above, is by definition solving the linear system (2) in the Total Least Square (TLS) sense (see [3],[4] for definitions and properties of TLS, [5] for application of TLS to this present problem.)

V-RELATIONSHIP BETWEEN BCKT AND IP

Let us describe BCKT.

BCKT is derived from KT and uses the fact (Appendix A) that

$\sigma_k^2 \rightarrow \sigma_k^{0\,2} + \sigma^2$ a.s. , in order to reduce the bias.

Firstly, an estimate of σ^2 is computed by

$$\hat{\sigma}^2 = \frac{1}{L-M} \sum_{k=M+1}^{L} \sigma_k^2$$

(This is a consistent estimate of σ^2 [4],[9])

Then, $\hat{b}_{KT}$ of Eq.5 is replaced by

$$\hat{b}_{BCKT} = -\sum_{k=1}^{M} \frac{\sigma_k}{\sigma_k^2 - \hat{\sigma}^2} (u_k^+ \alpha) v_k$$

Let us show why $\hat{b}_{BCKT}$ can be expected to be a less biased estimator of b than $\hat{b}_{KT}$.

Indeed, letting $u_k = A v_k / \sigma_k$, we obtain

$$\hat{b}_{BCKT} = -\sum_{k=1}^{M} \frac{\sigma_k}{\sigma_k^2 - \hat{\sigma}^2} \frac{1}{\sigma_k} v_k^+ (A^+\alpha) v_k = -\sum_{k=1}^{M} \frac{v_k}{\sigma_k^2 - \hat{\sigma}^2} v_k^+ (A^+\alpha) \qquad (11)$$

We can compare (11) to (6).

Since $A^+\alpha \rightarrow A_0{}^+ \alpha_0$

$\sigma_k{}^2 - \sigma^2 \rightarrow \sigma_k{}^{0\,2}$ a.s.

$v_k \rightarrow v_k{}^0$

$\hat{b}_{BCKT}$ is asymptotically unbiased.

Note that $\hat{b}_{BCKT} = -(\hat{A}^+ \hat{A} - \hat{\sigma}^2 I_L)^{\#} A^+\alpha \cong -(\hat{A}^+ \hat{A} - \sigma^2 I_L)^{\#} A^+\alpha$, (where I_L means the L-dimensional identity matrix).

We are going to establish that $\hat{b}_{IP}$ can also be approximated by $-(\hat{A}^+ \hat{A} - \sigma^2 I_L)^{\#} A^+\alpha$.

It has been proven that $(A^+ A - \sigma^2 I_L)\, \hat{b}_{IP} \cong -A^+\alpha$ under the assumption that the L+1-M smaller eigenvalues of $A'^+ A'$ be approximately equal to σ^2 [5]. In this case, we can consider that $A^+ A - \sigma^2 I_L$ has rank M, and is thus equal to $\hat{A}^+ \hat{A} - \sigma^2 I_L$. We show in Appendix B that $\hat{b}_{IP}$ is indeed the minimum-norm solution of $(A^+ A - \sigma^2 I_L)\, b_{IP} \cong -A^+\alpha$.

VI-SIMULATIONS RESULTS

We use the example given in [1].

$$H(z) = \frac{0.5}{z-\mu} + \frac{0.5}{z-\bar{\mu}} \quad , \text{with} \quad \mu = 0.72 + j\,0.54$$

$$H(z) = \frac{z-0.72}{z^2 - 1.44z + 0.81} \qquad (12)$$

Thus, with the notations of Section I, $a_1 = -1.44$ $a_2 = 0.81$

In the following, we will be interested in the bias and standard deviation of a_1 (resp. a_2), which are the estimates of a_1 (resp. a_2) obtained by the different methods.

The number of data points is assumed to be N = 40.

The order L of the polynomial B(z) is assumed to be 8.

The noise root mean square (r.m.s.) value σ is given the different values 0.05, 0.1, 0.2 and 0.3. For each value of σ, we ran 500 Monte-Carlo simulations, first with the original KT method, then with BCKT, and eventually with IP.

The biases and standard deviations of the estimates of a_1 and a_2 are shown in Table I.

TABLE I

Performance Comparison of KT, BCKT and IP methods for example (11)

σ	Method	Bias a_1	Bias a_2	Standard Deviation a_1	Standard Deviation a_2
0.05	KT	0.69×10^{-2}	0.88×10^{-2}	0.0134	0.0122
	BCKT	0.28×10^{-3}	0.54×10^{-3}	0.0136	0.0126
	IP	0.30×10^{-3}	0.57×10^{-3}	0.0135	0.0124
0.1	KT	0.26×10^{-1}	$0.34\ 10^{-1}$	0.0277	0.0235
	BCKT	-0.2×10^{-4}	0.11×10^{-2}	0.0289	0.0261
	IP	-0.33×10^{-3}	0.77×10^{-3}	0.0280	0.0250
0.2	KT	0.97×10^{-1}	0.14	0.200	0.116
	BCKT	0.46×10^{-2}	0.18×10^{-1}	0.155	0.072
	IP	-0.46×10^{-2}	0.12×10^{-2}	0.0719	0.0528

We can see that BCKT and IP exhibit very small bias, even for small size of the data set ,which is not the case of KT. The variances of the three methods are similar for $\sigma = 0.05$ and $\sigma = 0.1$, and BCKT and IP improve upon KT when $\sigma = 0.2$.

APPENDIX A

Let $E=A-A_0$ be the noise matrix

$e = \alpha - \alpha_0$ be the noise vector corrupting α_0 .

1) Let us first prove that $A^+\alpha$ --------> $A_0^+\alpha_0$ a.s.

We have $A^+\alpha = A_0^+\alpha_0 + E^+\alpha_0 + A_0^+e + E^+e$

Consider any element of $E^+\alpha_0$. It is of the following form :

$$\frac{1}{N-L}\sum_{i=1}^{N-L}\varepsilon_{i+k}h_i \qquad \text{for some } k \geq 1$$

To prove the convergence to zero of the above expression,we use a Large Numbers Law, described in the following theorem [11]:

Let X_n be independent variables, with $E(X_n) = 0$ (where E means expectation) .

If $\sum_{n=1}^{\infty} \frac{E(X_n^2)}{n^2} < \infty$ **, then** $\frac{X_1 + \quad .... \quad + X_n}{n} \longrightarrow 0$ **a.s.**

Let $X_n = h_n \varepsilon_{n+k}$. Recall that the noise was assumed to be zero-mean and white, so that the ε_n (and thus the X_n) are independent variables. Obviously, $E(X_n) = 0$, and since $|h_n^2| \leq |\Sigma |\lambda_i| |^2$, the above series is convergent, and the theorem is applicable.

We thus have

$$\frac{1}{N-L} \sum_{i=1}^{N-L} \varepsilon_{i+k} h_i \xrightarrow[a.s.]{} 0 \quad \text{when} \quad N \to \infty$$

(Note the importance here of the normalizing factor $1/\sqrt{N-L}$).

A similar demonstration results in

$A_0^+ e \to 0$ a.s.

As to $H^+ e$, any of its elements has the form

$$\frac{1}{N-L} \sum_{i=1}^{N-L} \varepsilon_{i+k} \varepsilon_i \qquad \text{for } k \geq 1 \qquad (A1)$$

Let $X_i = \varepsilon_i \varepsilon_{i+k}$. We have $E(X_i) = 0$. We cannot use the Strong Law of Large Numbers [11] straightforwardly since the X_i are not independent. However, let us partition the $\{X_i\}$ into k subsequences $\{X_i^{(j)}\}$ as follows :

$X_i^{(j)} = X_{j+(i-1)*j}$ $\qquad j = 1, \ldots , k$.

For a given j, the $X_i^{(j)}$ are independent and equidistributed, hence the Strong Law of Large Numbers holds , so that

$$\frac{X_1^{(j)} + X_2^{(j)} + \ldots + X_N^{(j)}}{N} \dashrightarrow 0 \quad \text{a.s.} \qquad (A2)$$

Since this is true for all j from 1 to k , summing (A2) over the k possible values of j gives

$$\frac{X_1 + X_2 + \ldots + X_N}{N} \dashrightarrow 0 \quad \text{a.s.}$$

which completes the proof.

2) Let us now prove that

$\sigma_k^2 \to \sigma_k^{0\,2} + \sigma^2$ a.s.

$v_k \to v_k^0$

We have $A^+ A - \sigma^2 I - A_0^+ A_0 = E^+ A_0 + A_0^+ H + E^+ E - \sigma^2 I$.

As we saw above , $E^+ A_0$ and $A_0^+ E$ tend to zero a.s. when N tends to infinity.

As to $E^+ E - \sigma^2 I$, the off-diagonal elements are of the form of Eq.(1) , hence they tend to zero.The diagonal elements have the following form : $1/(N-L) \ \Sigma (\varepsilon_i^2 - \sigma^2)$. The Strong Law of Large Numbers ensures that these quantities tend to zero a.s. when N tends to infinity. Hence,

$A^+ A - \sigma^2 I - A_0^+ A_0$ tends to zero a.s. when N tends to infinity. Let $\kappa(N)$ be an upper bound for its elements. $\kappa(N) \dashrightarrow 0$ when $N \dashrightarrow \infty$.

Using a theorem of [12] about symmetric perturbations of symmetric matrices, we can assert that $|\sigma_k^2 - \sigma^2 - \sigma_k^{0\,2}| \leq L\, \kappa(N)$. Thus $\sigma_k^2 \to \sigma_k^{0\,2} + \sigma^2$ a.s.

A theorem of [8] about perturbations of eigenvectors associated to simple eigenvalues, ensures that $v_k \rightarrow v_k^0$ a.s. for k= 1, .. , M . In fact, in the case when some of the σ_k^0 would not be distinct, the demonstration of [8] can be generalized. This completes the proof.

APPENDIX B

$A'^+ A' - \sigma^2 I_{L+1}$ has approximatively rank M, and can be written as $V'_S (\Sigma_S^2 - \sigma^2 I_M) V'^+_S$ where $V'_S = [v'_1 \mid ... \mid v'_M]$ and $\Sigma_S = \mathrm{diag} (\sigma_1 ,..., \sigma_M)$. Remembering that

$$A'^+ A' - \sigma^2 I = \begin{bmatrix} \alpha^+ \\ A^+ \end{bmatrix} [\alpha \;\; A] - \sigma^2 I = \begin{bmatrix} \alpha^+\alpha - \sigma^2 & \alpha^+ A \\ A^+\alpha & A^+A - \sigma^2 I_L \end{bmatrix}$$

we can write $A^+ A - \sigma^2 I_L \equiv V'_S\uparrow (\Sigma_S^2 - \sigma^2 I_M) V'^+_S\uparrow$ (where $V'_S\uparrow$ means V'_S deprived from its first row) and $- A^+\alpha = - V'_S\uparrow (\Sigma_S^2 - \sigma^2 I_M) w^+$ (where $w = [v'_1(1), ... , v'_M(1)]$). Note that w is a row vector.

Since $A^+ A - \sigma^2 I_L \equiv \hat{A}^+ \hat{A} - \sigma^2 I_L$, we have

$- (\hat{A}^+ \hat{A} - \sigma^2 I_L)^\# A^+\alpha = - (V'_S\uparrow)^\# (\Sigma_S^2 - \sigma^2 I_M)^{-1} (V'_S\uparrow)^\# V'_S\uparrow (\Sigma_S^2 - \sigma^2 I_M) w^+$

$= - V'_S\uparrow (V'_S\uparrow^+ V'_S\uparrow)^{-1} (\Sigma_S^2 - \sigma^2 I_M)^{-1} (V'_S\uparrow^+ V'_S\uparrow)^{-1} V'_S\uparrow^+ V'_S\uparrow (\Sigma_S^2 - \sigma^2 I_M) w^+$

$= - V'_S\uparrow (V'_S\uparrow^+ V'_S\uparrow)^{-1} w^+ = V'_S\uparrow (I_M - w^+ w)^{-1} w^+ = - V'_S\uparrow (I_M + w^+ w / (1 - ||w||^2))w^+$ (inversion lemma)

$= - V'_S\uparrow w^+ / (1 - ||w||^2) = \hat{b}_{IP}$ (see Eq. (10)) , which completes the proof.

REFERENCES

[1] D.R.FARRIER and A.R.PRIOR-WANDESFORDE "A Comparison of Bias Reduced Methods for the Estimation of Rational Transfer Functions" *IEEE Trans.ASSP* ,Vol.ASSP 37, n°3, pp.431-433 March 1989

[2]R.KUMARESAN *Estimating the Parameters of Exponentially Damped or Undamped Sinusoidal Signals in Noise.* Thesis . Oct. 1982. Kingston (Rhode Island)

[3] G.H.GOLUB, C.F. VAN LOAN "An analysis of the total-least-squares problem". *SIAM Journal of Num.Analysis.* Vol. 17 n° 6 pp. 883-893 Dec.1980

[4]S. VAN HUFFEL *Analysis of the Total Least Square Problem and its Use in Parameter Estimation.* Doctorate Thesis.Katholieke Universiteit Leuven (Belgium) June 1987.

[5] MD.A. RAMAN and K.B.YU "Total Least Squares Approach fo Frequency Estimation Using Linear Prediction "*IEEE Trans. ASSP* Vol.ASSP 35 n° 10,pp.1440-1454, Oct.1987

[6]R.KUMARESAN, D.W.TUFTS "Estimating the Parameters of Exponentially Damped Sinusoids and Pole-zero Modeling in Noise"*IEEE Trans. on ASSP* Vol.ASSP 30, n° 6, pp.833-840 .Dec. 1982

[7]Z.D.BAI , KRISHNAIAH P.R., ZHAO L.C. "On estimation of the Number of Signals and Frequencies of Multiple Sinusoids". *Proc. ICASSP 1987 Dallas* pp.1308-1311.

[8] D.W.TUFTS, R.KUMARESAN "Estimation of Frequencies of Multiple Sinusoids : Making Linear Prediction Perform Like Maximum Likelihood". *Proc. IEEE* , Vol.70, n°9, pp.975-989. Sept.1982.

[9] L.J.GLESER "Estimation in a Multivariate "errors-in-variables" Regression Model : Large

Sample Results". *The Annals of Statistics*. Vol.9 n°1, pp.24-44 ,1981.

[10]R.KUMARESAN, D.W.TUFTS "Estimating the Angles of Arrival of Multiple Plane Waves". *IEEE Trans. on Aerospace and Electronic Systems* . Vol.19.AES n°1pp.134-139 Jan.1983.

[11]W. FELLER *An Introduction to Probability Theory and its Applications Vol.II* p.239 , Wiley and Sons, 1971.

[12] J.H. WILKINSON *The Algebraic Eigenvalue Problem.* Clarendon Press.Oxford. 1965.

ARTICLE 4

A Fast Exact Least Mean Square Algorithm,
par J. BENESTY et P. DUHAMEL

soumis en Novembre 1989 à IEEE Trans. ASSP

A Fast Exact Least Mean Square Adaptive Algorithm

Jacob Benesty, Pierre Duhamel

CNET/PAB/RPE
38-40, Rue du Général Leclerc
92131 Issy-les-Moulineaux, FRANCE.

ABSTRACT

We present a general block-formulation of the LMS algorithm for adaptive filtering. This formulation has an exact equivalence with the initial LMS, hence retaining the same convergence properties, while allowing a reduction in the arithmetic complexity, even for very small block lengths. Working with small block lengths is very interesting from an implementation point of view (large blocks means large memory and large system delay) and allows nevertheless a signifiant reduction in the number of operations. Furthermore, tradeoffs between number of operations and convergence rate are obtainable, by applying certain approximations to a matrix involved in the algorithm. The usual block-LMS (BLMS) hence appears as a special case, which explains its convergence behaviour according to the type of input signal (correlated or uncorrelated).

I. INTRODUCTION

Adaptive filters are widely used in many applications, including system modeling, adaptive antennas, interference cancelling and so on [1,2,3]. Some of these applications require large FIR adaptive filters (straightforward adaptive acoustic echo cancellation would require filters of about 4000 taps), and it is therefore important to reduce the computational load of these tasks.

Usually, this has been performed by block processing, in which the filter's coefficents remain unchanged during a chosen number of samples N which is the block size. Reduction of the arithmetic complexity is obtained by making use of the redundancy between the successive computations, using the same techniques as the ones used in the fixed coefficient filtering.

Several techniques are known to reduce the arithmetic complexity of fixed coefficient FIR filtering. The usual ones are based on FFT's (most of them) or on aperiodic convolution (possibly) as an intermediate step. The main drawback of these methods is that they require large block processing : N is usually twice as large as the filter's length, which becomes very large for the very applications where a reduction in the arithmetic complexity is required. Furthermore, these classical fast algorithms involve a global exchange of data inside this large vector, a situation always difficult to manage with in actual implementations. On the contrary, the initial computation is a multiply-accumulate operation, which is very efficiently implemented, either in software or in hardware. In some sense, the fast algorithms have lost the structural regularity of the filtering process. Nevertheless, a new class of fast FIR filtering algorithms taking these considerations into account was recently proposed [9,12,13] : for a given block length, which may possibly be small, these algorithms allow a reduction of the arithmetic complexity (of course, the larger the block size is, the smaller the computational complexity per output point is), while retaining partially the usual FIR filtering structure : The multiply-accumulate operation is still a basic building block of these algorithms [19].

And indeed, all the above techniques were already used in the adaptive case, from the classical ones [7,8] to the most recent ones [15] or, even, more exotic ones [18]. The implementations in the time or frequency domain [6], although proposed independently were early recognized to be equivalent [14].

Nevertheless, the requirement for these techniques to be applied in an adaptive filtering scheme (i.e. the "blocking" of the coefficients during N samples) results in a different behaviour of block adaptive algorithms compared to the sample-by-sample LMS, excepted in the very special case of uncorrelated inputs. Indeed, we illustrate on an example in this paper that the block-LMS (BLMS) algorithm has a smaller convergence domain than the LMS algorithm in the case of a correlated input, resulting in a slower overall convergence to ensure stability (this is the reason of the well-known divided by N adaptation step).

In this paper, we show that this drawback is removable, and that the availability of small-block processing techniques allows the derivation of computationally efficient block-adaptive algorithms which behave exactly like their scalar version. Some additional degrees of freedom exist that can even make the block algorithms converge faster than the non-block version.

This paper concentrates on the LMS algorithm, in the FIR case.

First, by working on a simple example, we prove that we can reduce the arithmetic complexity of the LMS algorithm without modifying its behaviour : the algorithm obtained is mathematically equivalent to LMS.

This is generalized in section III where it is shown that the LMS algorithm on a block of data of size N can be turned into a "fixed" filtering with some corrections, plus an "updating" part which generates the next taps. This algorithm is only a rearrangement of the initial equations of the LMS. Using the method "fast FIR filtering" for the fixed FIR part results in a so-called "Fast Exact Least Mean Square" (FELMS) algorithm, in which the total number of operations

(multiplications plus additions) is reduced whatever the block size (N) may be. For example, with N = 2, the number of multiplications can be reduced by about 25 % with a little increase of the number of additions, and this without any approximation. Furthermore, considering larger blocks results in a "mixed radix" LMS adaptive filter with increased arithmetic efficiency, the reduction in the number of both multiplications and additions being commanded by the block size.

On the other hand, we prove that the usual BLMS algorithm [8] is a special case of the FELMS algorithm, with an approximation on a matrix involved in the updating of the coefficients, which is the origin of the difference between the convergence rates of BLMS and LMS for correlated inputs.

Other approximations on that matrix lead to new algorithms (Fast Approximate Least Mean Square - FALMS) where the convergence rate is almost that of LMS with a number of operations still reduced compared to FELMS.

A table comparing the number of operations of the various algorithms is provided.

Finally, we show that this technique can be applied to several variations of the LMS algorithm (such as normalized LMS, sign algorithm).

All these points are illustrated by simulations of an acoustic impulse response identification.

II. AN EXAMPLE OF AN LMS ALGORITHM WITH REDUCED NUMBER OF OPERATIONS

An LMS adaptive structure has the overall organization depicted in fig.1, where :

(1)

$$\hat{y}(n) = \sum_{i=0}^{L-1} x(n-i)h_i(n) = X^t(n)H(n) = H^t(n)X(n)$$

L being the length of the filter, X(n) the observed data vector at time n :

$$X(n) = [x(n), x(n-1), \ldots\ldots, x(n-L+1)]^t$$

and H(n) the filter weight vector at time n :

$$H(n) = [h_0(n), h_1(n), \ldots\ldots, h_{L-1}(n)]^t$$

The algorithm for changing the weights of the LMS adaptive filter is given by :

(2)

a)

$$e(n) = y(n) - X^t(n)H(n)$$

b)

$$H(n+1) = H(n) + \mu e(n)X(n)$$

where μ is a parameter that controls stability and rate of convergence, and e(n) is the system error at time n.

It is well known [4] that if the adaptation constant μ is chosen such that :

$$0 < \mu < \frac{2}{L\,E[x^2(n)]}$$

where E[.] denotes statistical expectation, then the mean of the weight vector H(n) will converge to the optimal solution of Wiener - Hopf.

Our aim in this section is to provide, for a simple example, an exact equivalent of the algorithm of eq. (2) requiring a lower number of operations per output point. It is hoped that this will be obtained by working on small data blocks, in such a way that the overall organization of the algorithm is simple. Let us consider the simplest

case : a block size N=2.

Let us write equations (2) at time n - 1 :

(3)

$$e(n-1) = y(n-1) - X^t(n-1)H(n-1)$$

(4)

$$H(n) = H(n-1) + \mu e(n-1)X(n-1)$$

substituting (4) into (2) a), we obtain :

(5)

$$e(n) = y(n) - X^t(n)H(n-1) - \mu e(n-1)X^t(n)X(n-1)$$
$$= y(n) - X^t(n)H(n-1) - e(n-1)s(n)$$

with :

$$s(n) = \mu X^t(n)X(n-1)$$

Combining eq. (3) and (5) in matrix form results in :

(6)

$$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} - \begin{bmatrix} X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-1) - \begin{bmatrix} 0 & 0 \\ s(n) & 0 \end{bmatrix} \begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix}$$

or

$$\begin{bmatrix} 1 & 0 \\ s(n) & 1 \end{bmatrix} \begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} - \begin{bmatrix} X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-1)$$

hence :

(7)

$$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -s(n) & 1 \end{bmatrix} \left[\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} - \begin{bmatrix} X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-1) \right]$$

The second term of this equation appears to be the computation of two successive outputs of a fixed coefficient filter. Thus, we can apply the same technique as explained in ref. [9] :

(8)

$$\begin{bmatrix} X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-1) = \begin{bmatrix} x(n-1)\ x(n-2) \ldots\ldots x(n-L) \\ x(n)\ \ x(n-1) \ldots\ldots x(n-L+1) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ \vdots \\ h_{L-1} \end{bmatrix} (n-1)$$

$$= \begin{bmatrix} x(n-1)\ x(n-3) \ldots x(n-L+1) & x(n-2)\ x(n-4) \ldots x(n-L) \\ x(n)\ \ x(n-2) \ldots x(n-L) & x(n-1)\ x(n-3) \ldots x(n-L+1) \end{bmatrix} \begin{bmatrix} h_0 \\ h_2 \\ \vdots \\ h_{L-2} \\ h_1 \\ h_3 \\ \vdots \\ h_{L-1} \end{bmatrix} (n-1)$$

in which the even and odd numbered terms of the involved vectors have been grouped. Furthermore, in order to obtain a more compact notation, let us suppose L to be even, and define :

$$A_0 = [x(n) \quad x(n-2) \ldots\ldots x(n-L+2)]$$
$$A_1 = [x(n-1)\ x(n-3) \ldots\ldots x(n-L+1)]$$
$$A_2 = [x(n-2)\ x(n-4) \ldots\ldots\ x(n-L)\ ]$$
$$H_0(n-1) = [h_0\ h_2 \ldots\ldots h_{L-2}]^t(n-1)$$
$$H_1(n-1) = [h_1\ h_3 \ldots\ldots h_{L-1}]^t(n-1)$$

Equation (8) can now be rewritten as :

(9)
$$\begin{bmatrix} X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-1) = \begin{bmatrix} A_1 & A_2 \\ A_0 & A_1 \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} (n-1)$$

The same kind of work can be performed for the updating of the filter taps. First substitute (4) into (2) b) :

(10)
$$H(n+1) = H(n-1) + \mu e(n) X(n) + \mu e(n-1) X(n-1)$$

or, with the above notations :

(11)

$$\begin{bmatrix} H_0 \\ H_1 \end{bmatrix}(n+1) = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}(n-1) + \mu e(n) \begin{bmatrix} A_0^t \\ A_1^t \end{bmatrix} + \mu e(n-1) \begin{bmatrix} A_1^t \\ A_2^t \end{bmatrix}$$

The following set of two equations is now seen to be the exact equivalent of the initial definition of LMS given in (2), for a block of 2 outputs :

(12)

a) $$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -s(n) & 1 \end{bmatrix} \left[\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} - \begin{bmatrix} A_1 & A_2 \\ A_0 & A_1 \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}(n-1) \right]$$

b) $$\begin{bmatrix} H_0 \\ H_1 \end{bmatrix}(n+1) = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}(n-1) + \mu \begin{bmatrix} A_1^t & A_0^t \\ A_2^t & A_1^t \end{bmatrix} \begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix}$$

Now, the reduction in arithmetic complexity can take place, by rewriting (12) as :

(13)

a) $$\begin{bmatrix} e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -s(n) & 1 \end{bmatrix} \left[\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} - \begin{bmatrix} A_1(H_0+H_1)+(A_2-A_1)H_1 \\ A_1(H_0+H_1)-(A_1-A_0)H_0 \end{bmatrix}(n-1) \right]$$

b) $$\begin{bmatrix} H_0 \\ H_1 \end{bmatrix}(n+1) = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}(n-1) + \mu \begin{bmatrix} A_1^t(e(n-1)+e(n)) - (A_1-A_0)^t e(n) \\ A_1^t(e(n-1)+e(n)) + (A_2-A_1)^t e(n-1) \end{bmatrix}$$

Considerations similar to the ones in [9] will allow to evaluate precisely the number of arithmetic operations involved in (13) :

The filtering operation $A_1(H_0 + H_1)$ is common between the two terms of (13) a), and the overall arithmetic computation is now that of 3 length N/2 filters, instead of 4, like in eq. (12). Two of them are applied on combinations of the input samples, namely :

(14)

$$A_2 - A_1 = [x(n-2) - x(n-1), \ldots \ldots, x(n-L) - x(n-L+1)]$$
$$A_1 - A_0 = [x(n-1) - x(n), \ldots \ldots, x(n-L+1) - x(n-L+2)]$$

The apparent number of additions involved in (14) can be reduced by noticing that the previous set of scalar products ($X^t(n-2)H(n-3)$, $X^t(n-3)H(n-3)$) already required nearly the same

operations, which were stored in the filtering process and that only two new additions are to be computed : (x(n-2)-x(n-1)) and (x(n-1)-x(n)).

This results in the overall organization of the algorithm as depicted in fig. (2).

Note also that s(n), although being defined as a scalar product of length L, can be computed recursively from s(n-2) as :

(15)

$$s(n) = s(n-2) + \mu[x(n-1)(x(n)+x(n-2)) - x(n-L-1)(x(n-L)+x(n-L-2))]$$

The second expression in the brackets was already calculated at time n-L, thus (15) requires only one multiplication and three additions. Furthermore, if the adaptation step μ is chosen as a negative power of 2, multiplication by μ will be considered as a shift, instead of a general multiplication.

The comparison of the arithmetic complexities is now as follows :

Computation of the LMS algorithm, as expressed by (2) and (3) for two successive outputs requires :

4 L multiplications
4 L additions,

the proposed algorithm, as expressed by (13) requires :

3 L + 2 multiplications
4 L + 8 additions

for the same computation. This means that the number of multiplications has been reduced (about 25 % improvement) with only 4 additions per output more, and without any approximation in the initial equations describing the LMS algorithm.

The above approach is different from the one that has been used previously for describing block adaptive algorithms [8,15]. In the usual approach, the initial equations are rewritten in vector instead of scalar form without changing anything else. We shall see in the

following that this fact has a number of consequences on the behaviour of the resulting algorithms. On the contrary, in our approach, the reduction in the number of operations is obtained only by a rearrangement of the initial equations, and there is an exact mathematical equivalence between the initial algorithm and our block version of it.

III. GENERALISATION TO ARBITRARY N

The algorithm explained above for two successive outputs can easily be generalized to an arbitrary block size N. We shall proceed in the same way as in section II : we first establish an exact block formulation of the LMS, on which a reduction of the arithmetic complexity is performed.

III.1. Exact block formulation of the LMS algorithm

Let us assume that the block length N is a factor of the length of the filter : L = NM (for a filter length that is not divisible by N, it is zero extended to the smallest multiple of N to satisfy the assumption), and let us write the LMS error equations at time n-N+1, n-N+2,....., n-1, n :

(16)

$$\begin{bmatrix} e(n-N+1) \\ e(n-N+2) \\ \vdots \\ \vdots \\ e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} y(n-N+1) \\ y(n-N+2) \\ \vdots \\ \vdots \\ y(n-1) \\ y(n) \end{bmatrix} - \begin{bmatrix} X^t(n-N+1) \\ X^t(n-N+2) \\ \vdots \\ \vdots \\ X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-N+1) - S(n) \begin{bmatrix} e(n-N+1) \\ e(n-N+2) \\ \vdots \\ e(n-1) \\ e(n) \end{bmatrix}$$

Nx1 Nx1 NxL Lx1 NxN Nx1

with matrix S defined as follows :

(17)

$$S(n)=\begin{bmatrix} 0 & 0 & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots 0 \\ s_1(n-N+2) & 0 & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots 0 \\ s_2(n-N+3) & s_1(n-N+3) & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ s_{N-1}(n) & s_{N-2}(n) & \cdots\cdots\cdots\cdots\cdots\cdots\cdots s_1(n) \quad 0 \end{bmatrix}$$

where :

$$s_i(n)=\mu X^t(n)X(n-i) \quad , \; i=1,2,\ldots\ldots,N-1.$$

The filter weight will then be updated once per data block (instead of once per data sample) in such a way that these weights are equal to those that would be found in the initial LMS algorithm at the same time. This will be performed by the following equation :

(18)

$$H(n+1)=H(n-N+1)+\mu[X(n-N+1)\; X(n-N+2)\ldots\ldots X(n)]\begin{bmatrix} e(n-N+1) \\ e(n-N+2) \\ \vdots \\ \vdots \\ e(n) \end{bmatrix}$$

Lx1 Lx1 LxN Nx1

Both equations are stated in a more convenient form as follows :

(19)

a) $\underline{e}(n)=\underline{y}(n)-\underline{X}(n)H(n-N+1)-S(n)\underline{e}(n)$

b) $H(n+1)=H(n-N+1)+\mu\underline{X}^t(n)\underline{e}(n)$

eq (19) a) can be rewritten as :

(20)

$$[S(n)+I]\underline{e}(n)=\underline{y}(n)-\underline{X}(n)H(n-N+1)$$

$$\underline{e}(n)=[S(n)+I]^{-1}[\underline{y}(n)-\underline{X}(n)H(n-N+1)]$$
$$=G(n)[\underline{y}(n)-\underline{X}(n)H(n-N+1)]$$

Obtaining G (n) may seem intricate to compute, but, recalling that G (n) is lower triangular, it is easily seen to be obtained by :

$$G(n) = G_{N-1}(n)\,G_{N-2}(n)\ldots\ldots G_1(n)$$

with :

(21)

$$G_i(n) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & & \vdots & \vdots \\ \vdots & \vdots & & 0 & \vdots \\ -s_i(n-N+i+1) & -s_{i-1}(n-N+i+1) & \cdots & 1 & \vdots \\ 0 & & & 0 & \vdots \\ \vdots & & & \vdots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \leftarrow \text{line } i+1$$

And, as a result, eq. (22) is now seen to be an exact block representation of the LMS :

(22)

a) $\underline{e}(n) = G(n)\,[\underline{y}(n) - \underline{X}(n)\,H(n-N+1)]$

b) $H(n+1) = H(n-N+1) + \mu\,\underline{X}^t(n)\,\underline{e}(n)$

Eq.(22) a) is easily seen to contain a fixed coefficient filtering, during the period N.

Using the techniques described in [11], we can therefore apply a reduction of the arithmetic complexity of this filtering during the time its coefficients remain unchanged. This is obtained by first performing a proper reordering of $\underline{X}$ and H :

Let us define :

(23)

$$A_j = [x(n-j)\;\; x(n-N-j)\;\; \ldots\ldots x(n-Ni-j)\;\; \ldots\ldots x(n-L+N-j)]$$

a row vector (1 x L/N), for j = 0,1,.....,2N-2; i=0,1,.....,(L/N)-1 and

(24)

$$H_k(n-N+1) = [h_k, h_{k+N}, \ldots\ldots, h_{k+Ni}, \ldots\ldots, h_{k+L-N}]^t(n-N+1)$$

for k=0,1,.....,N-1.

Then, $\underline{X}(n)$ expressed in terms of these polyphase components turns out to be a block Toeplitz matrix.

(25)

$$\underline{X}(n)H(n-N+1) = \begin{bmatrix} A_{N-1} & A_N \cdots\cdots\cdots\cdots\cdots\cdots A_{2N-3} & A_{2N-2} \\ A_{N-2} & A_{N-1} & A_{2N-3} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ A_1 & \vdots & A_N \\ A_0 & A_1 \cdots\cdots\cdots\cdots\cdots\cdots A_{N-2} & A_{N-1} \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ \vdots \\ H_{N-2} \\ H_{N-1} \end{bmatrix} (n-N+1)$$

This block-Toeplitz matrix can be seen as the representation of length-N FIR filtering where all the coefficients involved (x and h) are replaced by blocks. Hence, fast FIR filtering, as explained in [9] can apply. Note that, in the context of block-adaptive filtering, all the fast FIR algorithms should be used in their overlap-save version, rather than overlap-add (in other terms, following [9] they should be used in the version based on the transposition of polynomial products). This is due to the fact that any reuse of partial computations that has been performed in the previous blocks is to be avoided if it involves a filtering.

Let us now consider more precisely the computation of each term of eq. (22) :

As explained for the example N = 2, the computation of $S(n) = \{s_i(n)\}$ can be performed recursively as follows :

A first equation (26) provides the expression of the first column of matrix S ($s_i(n - N+i+1)$) in terms of the last row of this matrix ($s_i(n-N)$) during the previous block :

(26)

$$s_i(n-N+i+1) = s_i(n-N) + \mu\Big[\sum_{j=0}^{i} x(n-N+i-j+1)\,x(n-N-j+1) - \sum_{j=0}^{i} x(n-L-N+i-j+1)\,x(n-L-N-j+1)\Big]$$

for i=0,1,.....,N-1

and eq. (27) provides the computations to be performed along the sub-diagonals :

(27)

$$s_i(n+1) = s_i(n) + \mu[x(n+1)x(n-i+1) - x(n-L+1)x(n-i-L+1)]$$

Taken altogether, there are N(N - 1)/2 such equations, requiring a total of N (N - 2) multiplications and (3/2)N (N - 1) additions.

Note that one multiplication can be saved in eq. (26) because x (n - N +1) appears twice. This fact has been taken into account in our operation counts.

It is also possible to show in eq. (22) that the combinations of the input samples required by the fast FIR filtering process in eq. (22) a) can be reused for the updating of the filter coefficients (22)b):

In fact, all fast FIR algorithms can be seen as a "diagonalization" of the filtering matrix as follows : If **M** is the number of multiplications required by the length N (short) length FIR filter, eq. (22) turns to :

(28)

a) $\underline{e}(n) = G(n)[\underline{y}(n) - A\underline{X}_d(n)(BH(n-N+1))]$

b) $H(n+1) = H(n-N+1) + \mu B^t \underline{X}^t_d(n) A^t \underline{e}(n)$

where A is an N x **M** block - matrix, and B an (**M**L/N) x N matrix, both of them involving only additions. $\underline{X}_d(n)$ is block - diagonal, each "block-coefficient" involving linear combinations of the sub-sequences defined by (23). Eq. (28) clearly shows that the update of H(n+1) involves the same input combination $\underline{X}^t_d(n)$ as the computation of the error vector.

Let us illustrate these points on the special case N = 3 :

The error vector is provided by :

(29)

$$\begin{bmatrix} e(n-2) \\ e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -s_2(n) & -s_1(n) & 1 \end{bmatrix} \left\{ \begin{bmatrix} 1 & 0 & 0 \\ -s_1(n-1) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y(n-2) \\ y(n-1) \\ y(n) \end{bmatrix} - \begin{bmatrix} X^t(n-2) \\ X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-2) \right\}$$

and the computation of the second term of (29) is performed by a length-3 fast FIR filter, as given in [19] :

(30)

$$\begin{bmatrix} X^t(n-2) \\ X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-2) = \begin{bmatrix} A_2 & A_3 & A_4 \\ A_1 & A_2 & A_3 \\ A_0 & A_1 & A_2 \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \\ H_2 \end{bmatrix} (n-2)$$

$$= \begin{bmatrix} A_2(H_0+H_1+H_2)+(A_3-A_2)(H_1+H_2)+(A_4-A_3)H_2 \\ A_2(H_0+H_1+H_2)-(A_2-A_1)(H_0+H_1)+(A_3-A_2)(H_1+H_2)-[(A_3-A_2)-(A_2-A_1)]H_1 \\ A_2(H_0+H_1+H_2)-(A_2-A_1)(H_0+H_1)-(A_1-A_0)H_0 \end{bmatrix}$$

The equivalence with eq. (28) is as follows :

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\Sigma_d(n) = \mathrm{diag}[A_2, A_3 - A_2, A_4 - A_3, A_2 - A_1, (A_3 - A_2) - (A_2 - A_1), A_1 - A_0]$$

(resulting in an 6 x (6L/3))

$$B = \begin{bmatrix} I & I & I \\ O & I & I \\ O & O & I \\ -I & -I & O \\ O & -I & O \\ -I & O & O \end{bmatrix}$$

((6L/3) x L)

I and O being identity and null matrices of size (L/3) x (L/3).

The elements $s_i(n)$ are computed recursively, as explained in the general case by eq. (26) and (27) :

(31)

$$s_1(n-1) = s_1(n-3) + \mu[\, x(n-2)(x(n-1) + x(n-3)) - x(n-L-2)(x(n-L-1) + x(n-L-3))]$$

$$s_1(n) = s_1(n-1) + \mu[\, x(n)x(n-1) - x(n-L)x(n-L-1)]$$

$$s_2(n) = s_2(n-3) + \mu[\, x(n-2)(x(n) + x(n-4)) + x(n-1)x(n-3) - x(n-L-2)(x(n-L) + x(n-L-4)) - x(n-L-1)x(n-L-3)]$$

Now, we adjust the weight vector as :

(32)

$$\begin{bmatrix} H_0 \\ H_1 \\ H_2 \end{bmatrix}(n+1) = \begin{bmatrix} H_0 \\ H_1 \\ H_2 \end{bmatrix}(n-2) + \mu \begin{bmatrix} A_2^t & A_1^t & A_0^t \\ A_3^t & A_2^t & A_1^t \\ A_4^t & A_3^t & A_2^t \end{bmatrix} \begin{bmatrix} e(n-2) \\ e(n-1) \\ e(n) \end{bmatrix}$$

$$= \begin{bmatrix} H_0 \\ H_1 \\ H_2 \end{bmatrix}(n-2) + \mu B^t \underline{X}_d^t(n)\, A^t \begin{bmatrix} e(n-2) \\ e(n-1) \\ e(n) \end{bmatrix}$$

$$= \begin{bmatrix} H_0 \\ H_1 \\ H_2 \end{bmatrix}(n-2) + \mu \begin{bmatrix} A_2^t(e(n)+e(n-1)+e(n-2)) - (A_2 - A_1)^t(e(n)+e(n-1)) - (A_1 - A_0)^t e(n) \\ A_2^t(e(n)+e(n-1)+e(n-2)) - (A_2 - A_1)^t(e(n)+e(n-1)) + (A_3 - A_2)^t(e(n-1)+e(n-2)) - ((A_3 - A_2)^t - (A_2 - A_1)^t)e(n-1) \\ A_2^t(e(n)+e(n-1)+e(n-2)) + (A_3 - A_2)^t(e(n-1)+e(n-2)) + (A_4 - A_3)^t e(n-2) \end{bmatrix}$$

This algorithm requires, for three successive iterations 4 L+7 multiplications and 5 L + 23 additions instead of 6 L multiplications and 6 L additions for the LMS algorithm.

Note that, compared to the case N = 2, working with 3 outputs at a time is now seen to be more efficient : the total number of operations (multiplications plus additions) has been reduced by 25 %.

The main result of this section is that, provided that some caution is taken (the S matrix of eq. (17)), nearly any fast FIR

filtering algorithm can be used in an LMS algorithm to reduce the arithmetic complexity, without modifying its convergence properties, since there is an exact equivalence between both versions of the algorithm.

III.2. Two special cases of interest

III.2.1. $N = 2^n$

This is an important case, since simple and efficient fixed-coefficient fast FIR filtering algorithms are known for this type of length. A significant reduction of the arithmetic complexity is thus feasible, even with small to moderate block lengths. Straightforward application of the results of section III.1. results in the following operation counts for a block of $N = 2^n$ outputs of a filter of length L=NM.

Total number of operations :

(33)

$2\ (3/2)^n\ L + 2^n\ (3\ 2^n - 5)/2 + 1$ multiplications

(34)

$2\ (2\ (3/2)^n - 1)\ L + 2^{n+1}\ (2^n - 3) + 4\ 3^n$ additions,

or, if we consider the number of operations per output :

(35)

$2\ (3/2)^n\ M + (3\ 2^n - 5)/2 + 1/2^n$ multiplications

(36)

$2\ (2\ (3/2)^n - 1)\ M + 4\ (3/2)^n + 2\ (2^n - 3)$ additions,

to be compared with 2 L multiplications and 2 L additions per output required by the LMS algorithm.

It is easily verified that, as long as $M \geq 2$, FELMS will require fewer operations than LMS. Note also that the second term of (35) and (36) involves $N=2^n$, instead of NM in the LMS. This term (2^n) is due to the computation of matrix S(n). This is the point where the

availability of small block-processing made this approach feasible : In fact, if N was of the same order of magnitude as L, FELMS would require even more operations than LMS. The important point now is that arithmetic complexities involve a term growing with N (updating of S), and another one diminishing with N (fast FIR). Hence, eq. (35) has a minimum : the zero of the derivative of (35) results in the following "optimum" which provides the least number of multiplications :

(37)

$$n_{opt} \approx -0.6 + 0.7 \log_2 L$$

Table 1 provides the number of operations required by both LMS and FELMS for various filter lengths, and a blocksize of the order of the one given by (37). It is seen that an adaptive filter of length 128 can be implemented with half as many operations per point than the LMS algorithm, with a block length of only 16. A reduction by a factor near 4 is obtained for a filter of length 1024, and a block length equal to 64. Compared with classical fast algorithms [14] our approach offers an easier implementation, and compared with other recent approaches [15], it allows a faster convergence in the case of correlated inputs, at the cost of a slight increase of the computational complexity (see table 1).

Note that the most interesting case in our approach is found when the block size is smaller than the filter's length. This is clearly seen from eq.(37).

The same kind of approach can be applied to different criteria (e.g. total number of operations...) as was performed in the fixed coefficient case [19].

III.2.2. FFT- based implementations

One of the possible fixed-coefficient FIR schemes uses the FFT as an intermediate step [12, 10], in which case matrices A and B in eq. (28) turn out to be parts of Fourier matrices. Note that, in this case, the Fourier transform is used only for "block - diagonalizing" the block Toeplitz matrix of eq. (25), and that the usual constraint of

the size of the FFT being twice the filter's length does not hold. With the notations of this paper, the length of the FFT is twice the block size.

The important point is that some of the properties of FFT are known to be useful in the context of block - processing [15,16,17]. In particular, the possibility of increased speed of convergence by using different adaptation steps on the Fourier coefficients of the impulse response of the filter is usually thought to be linked with the orthogonality property of the DFT [21]. Hence, these possibilities of increased rate of convergence should be kept in the implementation proposed in this section, by using different adaption steps for each subfilter (there are several possibilities for this task and a paper is in preparation).

III.3. Simulations

The above algorithms have been programmed for an acoustic impulse response identification, in a scheme as depicted in fig. (3).

The system to be identified is described by an impulse response measured in a real room (500 ms duration, sampled to 16000 Hz) with a slow movement of a reflector during the experience. The model is an FIR filter of length 1024.

Fig. (4) provides the error curves of several algorithms in the case of white noise input, and fig. (5), (6), (7) in the case of USASI noise input (USASI noise is a correlated noise with the same spectrum as speech). For the purpose of drawing, error samples are averaged over 128 points to smooth the curves and normalized by the corresponding input energy.

III.3.1. FELMS versus LMS

Curves of fig. (5) a) and (7) a) refer to both LMS and FELMS : In both cases of simulations, they exactly superimpose. Since they have an exact mathematical equivalence, this is not astonishing. Nevertheless, this gives an indication on the accuracy issue of FELMS:

the updating of coefficients $s_i(n)$ being performed recursively, one may wonder if this computation could introduce any major drawback. Curves of fig. (5) a) and 7) a) show that, after 9000 iterations, for a simple precision (32 bits) floating point implementation, the two curves could not be distinguished. It is further shown in Appendix A that, in the case of fixed point implementations, this way of computing only result in a slight increase of the residual error. It is also shown that, in any realistic case, these errors cannot result in an instability of the algorithm.

III.3.2. FELMS versus BLMS

FELMS and BLMS are both block adaptive algorithms. The number of operations of the first one is hardly greater than the second one (see table 1). The question now is about their relative convergence :

Simulations were first performed with the same adaptation step μ for both algorithms (the same μ as used in the LMS). Note that, compared to the adaptation step recommended in the classical paper on BLMS, this μ is N times larger.

In the uncorrelated case, the convergence curves were almost identical for any block lengths from N = 2 to N = 128, exhibiting only tiny differences : fig. (4) b) depicts the first BLMS curve that can visually be distinguished from the LMS - FELMS case (curves for FELMS are of course always identical to the one of fig. (4) a) whatever the block size N may be).

In the case of USASI noise input, the situation is much different : fig. (6) provides the error curve of BLMS with the same step μ as LMS and FELMS algorithm (fig. (5) a)) : Divergence is seen to occur quickly. The solution to this problem is well-known : the adaptation step in BLMS should be N times smaller than in LMS in order to ensure stability. In this case, the algorithm converges, as shown in fig. (5) b). Nevertheless, it is seen that the convergence is now slower than in the LMS case, this drawback being stronger for increasing N (see the N = 16 curve of fig. (5) c)).

From these simulations, it is clearly seen that BLMS performs best for uncorrelated inputs, in which case it is equivalent to LMS provided that the adaptation step is chosen N times larger than usually. Correlated inputs result in a reduced convergence region. The "better" convergence of BLMS reported elsewhere [7] in the case of frequency domain implementation seems to be due only to the use of a different adaptation step on each coefficient, which is also feasible for the proposed algorithm (see section III.2.2.).

This has to be compared with FELMS which always has the same error curve as LMS, whatever the input signal and block size may be. Eq. (22) is used in the next section to explain precisely why this behaviour of BLMS occurs.

IV. TRADING CONVERGENCE SPEED FOR ARITHMETIC COMPLEXITY

In fact, (22) can be seen to be the same set of equations as the definition of BLMS [8,15], but for the term G (n) : Taking G(n)=I, the identity matrix, turns FELMS to BLMS, together with an eventual change of μ. The role of this matrix (or matrix S(n) in eq.(19)) is thus seen to be crucial in these algorithms.

IV.1. Interpretation of matrix S

A straightforward calculation shows that, under the following conditions :

- the input signal is ergodic
- the filter's length L is large enough to provide acceptable average of the statistics of the signal.

We have as a result :

(38)

$$s_1(n) = \mu \sum_{i=0}^{L-1} x(n-i)x(n-1-i) \rightarrow \mu L\, E[x(n)\,x(n-1)] = \mu L\, r(1)$$

$$\vdots$$

$$\vdots$$

$$s_{N-1}(n) = \mu \sum_{i=0}^{L-1} x(n-i)x(n-N+1-i) \rightarrow \mu L\, E[x(n)\,x(n-N+1)] = \mu L\, r(N-1)$$

where r is the autocorrelation function of the input signal. Hence, S converges to :

(39)

$$S_A = \mu L \begin{bmatrix} 0 & 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & 0 \\ r(1) & 0 & \vdots \\ r(2) & r(1) & \vdots \\ \vdots & r(2) & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ r(N-1) & r(N-2) \ldots\ldots\ldots r(2) \quad r(1) & 0 \end{bmatrix}$$

Note that since :

$$0 < \mu < 2 / L\, r(0)$$

and :

$$r(0) \rangle |r(i)| \quad \forall\, i \neq 0$$

we obtain :

$$\mu L\, |r(i)| \langle 2 \quad \forall\, i \neq 0$$

which gives an indication on the range of the s_i.

IV.2. Various approximations on S :

IV.2.1. BLMS

We have already seen that BLMS can be seen as an FELMS where the S - matrix has been taken equal to zero. The results of section IV.1. explain why this does make sense whenever the input is

uncorrelated, since in this case the autocorrelation coefficients are equal to zero.

Nevertheless, when the input is correlated, this may be a drastic approximation : fig. (6) shows that, when used with the same adaptation step as the initial LMS, BLMS diverges soon, even for such small blocks as N = 4.

IV.2.2. FALMS

Hence, the difference between FELMS and BLMS is seen to depend only on the statistics of the input signal. In many cases, if N is large enough, it will not be necessary to take into account the high order correlations of the input signal. It is therefore reasonable to keep only a few sub-diagonals in matrix S, up to the point where the correlation coefficients are known to be small enough. This will result in a Fast Approximate Least Mean Square (FALMS) algorithm which will have a faster convergence compared to BLMS, and an arithmetic complexity still reduced compared to FELMS. Table 1 provides the corresponding number of operations, in the case where $\log_2 N$ subdiagonals are kept. Note that this number of sub-diagonals should not be used as such : The number to be kept clearly depends on the correlation of the input signal. This rule-of-thumb was used in our example to provide a precise evaluation of the number of operations.

We have simulated this so-called FALMS algorithm under the same conditions as previously. Using USASI noise input, $\log_2 N$ subdiagonals were enough to make FALMS behave much like LMS : fig. (7) a) gives the adaptation curve of FELMS for N = 64, which is exactly the same one as would have been provided the LMS algorithm. Fig. (7) b) provides the error curve of FALMS for the same blocklength, and the same adaptation step μ, but using only six subdiagonals. Both curves are seen to be quite similar. The error curve of BLMS, with an adaptation step ensuring stability would be completely out of scale, and has not been plotted.

Let us point out that the main motivation for using an

approximate S-matrix is not so much with arithmetic complexity, since it is seen in table 1 that FELMS, FALMS, and BLMS require comparable number of operations. Main advantages should be lower memory requirements and easier organization of the resulting program.

Table 1, together with fig.7 shows that FALMS is a good tradeoff between algorithm complexity and speed of convergence.

Let us also point out that in the simulations of fig.7 BLMS saved 60 % computation time compared with LMS, but with a slower convergence, while FELMS saved 50 %, with exactly the same behaviour as LMS - These timings should not be taken as definitive ratios, the best implementation of FELMS is a subject of further study.

Another remark is that FELMS keeps another advantage of BLMS: the filter weights are updated once per data block, which reduces the amount of data flow. This is useful in DSP or VLSI implementations.

IV.2.3. Blocking the S matrix

An important practical situation is the case when one knows that the input is stationary, but one does not know its statistics.

FELMS can be used to obtain the signal autocorrelation, and when the values of S are stabilized, the recursions on the elements s_i can be stopped, and the remainder of the algorithm will use the obtained values.

Fig.7-c provides such a simulation where the s_i were estimated during 2048 samples (twice the filter's length) using the FELMS equation (19) and then blocked to the obtained values. The resulting error curve is seen to behave much like the initial LMS.

V. VARIATIONS OF THE LMS ALGORITHM

Our purpose in this section is to show that, although derived

for the straightforward LMS algorithm, the technique described above is of more general application. As examples, we have chosen the normalized LMS and the sign algorithms. Of course, we do not fully derive the algorithms, but only provide their block formulation, on which the fast FIR algorithms can easily be applied.

V.1. The normalized LMS algorithm

In this algorithm, the adaptation step is normalized by the input energy, and this results in a better convergence rate compared with the initial LMS. The normalized LMS equations are :

(40)

$$H(n+1) = H(n) + \mu(n) X(n) e(n)$$

$$e(n) = y(n) - X^t(n) H(n)$$

$$\mu(n) = \frac{\alpha}{X^t(n) X(n)}, \quad \text{with} \quad 0 < \alpha < 2$$

And, using the techniques described above, it is possible to obtain an exact block formulation of the normalized LMS :

(41)

$$\begin{bmatrix} e(n-N+1) \\ e(n-N+2) \\ \vdots \\ \vdots \\ e(n-1) \\ e(n) \end{bmatrix} = \begin{bmatrix} y(n-N+1) \\ y(n-N+2) \\ \vdots \\ \vdots \\ y(n-1) \\ y(n) \end{bmatrix} - \begin{bmatrix} X^t(n-N+1) \\ X^t(n-N+2) \\ \vdots \\ \vdots \\ X^t(n-1) \\ X^t(n) \end{bmatrix} H(n-N+1) - S(n) \begin{bmatrix} \bar{e}(n-N+1) \\ \bar{e}(n-N+2) \\ \vdots \\ \vdots \\ \bar{e}(n-1) \\ \bar{e}(n) \end{bmatrix}$$

S(n) being defined as in eq.(17), but with a different definition of $s_i(n)$:

$$s_i(n) = X^t(n) X(n-i)$$

$\bar{e}(n)$ and $\mu(n)$ are defined as follows:

(42)

$$\bar{e}(n) = \mu(n) e(n)$$

$$\mu(n) = \frac{\alpha}{X^t(n) X(n)} = \frac{\alpha}{s_0(n)}$$

The block - adaptation is as follows :

(43)

$$H(n+1) = H(n-N+1) + [X(n-N+1) \ldots\ldots\ldots X(n)] \begin{bmatrix} \bar{e}(n-N+1) \\ \vdots \\ \vdots \\ \vdots \\ \bar{e}(n) \end{bmatrix}$$

Note that a slight change of notation has been necessary, compared to the LMS, (μ is no more included in the S matrix) , in order to allow the same techniques to be applied : recursive computation of s_i (n) and fast FIR computation, that can be partially reused in the updating of H.

V.2. The sign algorithm

The purpose of this algorithm is not to increase convergence rate, but to reduce the number of operations to be performed for updating the coefficients :

(44)

$$H(n+1) = H(n) + \mu \, \mathrm{sgn}(e(n)) \, X(n)$$

$$e(n) = y(n) - X^t(n) H(n)$$

where:

$$\mathrm{sgn}(\theta) = +1 \text{ if } \theta \geq 0$$
$$= -1 \text{ if } \theta < 0$$

The block formulation of the algorithm is performed exactly in the same manner as for the LMS, with e(n) replaced by sgn(e(n)) in eq. (22).

Note that the resulting fast algorithm involves some terms of the form :

$$\Sigma \, \mathrm{sgn}(e(n))$$

which are no more equal to ± 1. A multiplication by such a quantity is nevertheless much simpler for small N than a general multiplication.

Other variations of the LMS, such as delayed LMS, can be treated by the same techniques

In fact, all these techniques are of much more general application than even the variations of LMS, since we were able to derive an exact block formulation of the Constant Modulus Algorithm (CMA),[17] which was thought to be impossible by the usual techniques. This work will be reported elsewhere [23].

VI. CONCLUSION

In this paper, we provided an algorithm allowing a reduction of the number of operations required by the LMS adaptive filter, without modifying any of its convergence properties. This reduction is significant whatever the block size may be, the usual relationship between the length of the filter and the block size not being compulsory any more. However, we showed that for a given filter length L, there is an optimum block size which is always smaller than L. This is an important point that makes the implementation of this algorithm efficient.

Furthermore, we showed that different approximations on some matrix led to a whole family of algorithms, the BLMS being one of its members. This fact has been used to explain precisely the convergence behaviour of BLMS. In the case of correlated input signal, our algorithm was shown to converge faster than the BLMS, with only a slight increase of the computational complexity.

Although explained in the special case of the LMS algorithm, these techniques can be applied to many different algorithms in the LMS family. We provided an outline of their application to two variations of the LMS.

We also applied these tools to the Constant Modulus Algorithm [23].

Work is continuing to implement this new algorithm in frequency-domain and to improve its convergence rate.

AKNOWLEDGMENTS

The authors would like to thank M. Vetterli, A. Gilloire, and Z.J. Mou for useful discussions on the subject of this paper. Simulation data were kindly provided by A. Gilloire. We also wish to thank the reviewers for their helpful comments and suggestions in improving the earlier version of the paper.

APPENDIX A Errors due to finite precision arithmetic

The theoretical analysis of the errors due to finite precision arithmetic in an adaptive algorithm is usually based on the assumption that the input samples are zero mean uncorrelated white Gaussian random variables. Nevertheless, in this case, the "autocorrelation" matrix S is zero, and there is no difference between FELMS and BLMS algorithms. So, a detailed analysis of FELMS is very difficult, and we shall only provide in the following some indications on the behaviour of FELMS under fixed point arithmetic.

Throughout this appendix, we use unprimed and primed symbols to represent quantities of infinite and finite precision, respectively. Another assumption is that a scalar product is computed with full precision, and quantized afterwards. This corresponds to the kind of implementation that is found in many digital signal processors.

We address separately the problem of computing the updating of the filter taps and that of the recursive computation of s_j. Another limitation of our analysis is that we consider only the influence of finite precision on the adaptive aspects, and assume that the fast FIR technique is applied in such a way that the resulting accuracy is comparable with that obtained in usual implementations. Preliminary results show that this is obtained by using $(1/2)\log_2 N$ additional bits in the fast FIR computation [22].

Let

σ_x^2

be the input signal power, and

$\sigma^2_{e'}(n)$

the error power at time n. Based on [20], the condition for the i^{th} component of the weight vector not to be updated in the LMS algorithm is :

(A1)

$$|\mu x(n-i)e'(n)| < 2^{-B_{LMS}-1}$$

where B_{LMS} is the number of bits used for representing the coefficients. Squaring both sides of (A1) , and with the assumption that n is large enough, so that the filter is near convergence, [20] shows that a reasonable approximation of (A1) is :

(A2)

$$\mu^2 \sigma_x^2 \sigma^2_{e'_{LMS}}(n) < \frac{2^{-2B_{LMS}}}{4}$$

Hence :

(A3)

$$B_{LMS} < -\frac{1}{2}[\log_2(\mu^2 \sigma_x^2) + \log_2(\sigma^2_{e'_{LMS}}(n)) + 2]$$

Concerning the FELMS algorithm, the same kind of calculation holds : (A4) is the condition for the i^{th} component of the weight vector not to be updated in the FELMS algorithm :

(A4)

$$\left| \mu \sum_{j=0}^{N-1} x(n-j-i)e'(n-j) \right| < 2^{-B_{FELMS}-1}$$

which gives :

(A5)

$$\mu^2 N \sigma_x^2 \sigma^2_{e'_{FELMS}}(n) < \frac{2^{-2B_{FELMS}}}{4}$$

Hence :

(A6)

$$B_{FELMS} < -\frac{1}{2}[\log_2(\mu^2 \sigma_x^2) + \log_2(\sigma^2_{e'_{FELMS}}(n)) + \log_2(N) + 2]$$

A scaling by 1/N in (A4) has not been taken into account, the multiplication by μ (which is very small usually) playing more than necessary this role.
And, if both algorithms converge, we have as a result :
(A7)

$$B_{FELMS} + \frac{1}{2}\log_2(N) \geq B_{LMS}$$

It is seen that, from the strict point of view of adaptation, FELMS would require (1/2) $\log_2$ N bits less than LMS.

Nevertheless, the main problem in FELMS is certainly the recursive computation of s_i (n), and a necessary condition for stability of FELMS is the stability of s_i , because their computation is independent of the remainder of the algorithm.

Let

$$d'_i(n) = s'_i(n) - s_i(n)$$

where d'_i is the error between the fixed-point estimate s'_i and the true (infinite precision) s_i.

A straightforward computation shows that :
(A8)

$$\sigma^2_{d'_i}(n) = \frac{n(N-i)}{N}\sigma^2_\beta$$

$$\text{where } \sigma^2_\beta = \frac{2^{-2b}}{12},$$

b being the number of bits used for the computation of s_i.

The worst case is seen to occur for s_1 , in which case :
(A9)

$$\sigma^2_{d'_1}(n) \approx n\,\sigma^2_\beta$$

We have seen in section IV.1. that

$$s_1(n) \approx \mu L\, r(1)$$

for large n (r being the autocorrelation function).

Furthermore, it is well known that

(A10)

$$|r(i)| \leq r(0) \quad \forall i \geq 0$$

So, we can write the condition such that, at time t, the time of convergence of the algorithm, the recursive computation of $s_i(n)$ will meet inequality (A10) :

(A11)

$$|s'_1(t)| \leq \mu L \sigma_x^2$$

which can be rewritten as :

(A12)

$$|s'_1(t) - s_1(t)| \leq 2 \mu L \sigma_x^2$$

And, taking the expectation of the square of (A12) :

(A13)

$$t \sigma_\beta^2 \leq 4 (\mu L \sigma_x^2)^2$$

Hence :

(14)

$$b \geq \frac{1}{2} \log_2(t) - \log_2(\mu L \sigma_x^2) - 3$$

This number of bits, required for eq. (A11) to be met, is to be compared to the number of bits required for a usual implementation of LMS, as given in [5] :

(A15)

$$b_{LMS} \approx 2 + \frac{1}{2} \log_2\left(\frac{\sigma_x^2}{\sigma_y^2}\right) + \log_2(G_S) + \frac{1}{2} \log_2(t)$$

with the assumption that

$$\sigma_x^2 \geq \sigma_y^2$$

where

σ_y^2

is the reference signal power, G_S^2 the system gain :

(A16)

$$G_S^2 = \frac{E[y^2(n)]}{E[e^2(n)]}$$

t being the time of convergence of LMS, which is the same one as that of FELMS, if the system is well designed.

Comparison of (A14) and (A15) shows that the use in FELMS of a number of bits equal to that of LMS, as provided in (A15) never results in errors greater than the difference between the correlations coefficients.

In summary, the (few) results we have obtained for the analysis of FELMS under finite precision arithmetic are :

- from the strict point of view of the updating of the coefficients, FELMS should require (1/2) $\log_2$ N bits less than LMS for representing the coefficients,

- from the point of view of the quantization noise at the output of the filter, FELMS should require (1/2) $\log_2$ N bits more than LMS for representing the data.

- updating the s_i with the same number of bits as that used in the LMS never results in unrealistic values of the correlation coefficients, even if the recursive computation is performed up to the time of convergence. Furthermore, we have seen (fig. 6.e) that this recursive computation could be stopped much earlier in the case of stationary inputs without modifying the convergence of the algorithm.

Note that, as usual, all these results are asymptotic.

As a conclusion, we think that FELMS can be implemented with same number of bits as LMS, FELMS exhibiting possibly an increased residual error.

Note also that these finite precision problems should always remain manageable, since our approach requires the use of very small block lengths : Implementing efficiently a filter length $2^{10}=1024$ by our algorithm requires only a block size equal to $2^6=64$.

REFERENCES :

[1] S. Haykin, "Adaptive filter theory", Prentice Hall, Englewood cliffs, N.J., 1986.

[2] B. Widrow et al., "A comparison of adaptive algorithms based on the methods of steepest descent and random search", IEEE Trans. Antennas Propagat., vol. AP-24, Sept. 1976, pp. 616 - 637.

[3] O. Macchi, "Le filtrage adapatatif en Télécommunications", Annales des Télécommunications, vol. 36, n° 11-12, 1981.

[4] B. Widrow et al., "Stationary and nonstationary learning characteristics of the LMS adaptive filter", Proc. IEEE, vol. 64, pp. 1151-1162, Aug. 1976.

[5] M.G. Bellanger, "Adaptive digital filters and signal analysis", Marcel Dekker Inc., New-York, 1987.

[6] E.R. Ferrara, "Fast implementation of LMS adaptive filters", IEEE Trans. Acoust. , speech, Signal, vol. ASSP-28, pp.474-475, Aug. 1980.

[7] D. Mansour, A.H. Gray, Jr., "Unconstrained frequency-domain adaptive filter", IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-30, PP. 726 - 734, oct. 1982.

[8] G.A. Clark, S.K. Mitra, S.R. Parker, "Block implementation of adaptive digital filters", IEEE Trans. on CAS., vol. 28, pp. 584-592, June 1988.

[9] Z.J. Mou, P. Duhamel, "Fast FIR filtering : algorithms and implementation", Signal Processing, Dec. 1987, pp. 377-384.

[10] Z.J. Mou, P. Duhamel, "A unified approach to the fast FIR filtering algorithms", Proc. ICASSP' 88, pp. 1914-1917.

[11] P. Duhamel, Z.J. Mou, J. Benesty, " Une présentation unifiée du filtrage rapide fournissant tous les intermédiaires entre traitements temporels et fréquentiels", 12th GRETSI Conf., Juan-les-Pins, June 1989, pp. 37-40.

[12] H.K. Kwan, M.T. Tsim, "High speed 1-D FIR digital filtering architecture using polynomial convolution", Proc. ICASSP' 87, pp. 1863-1866.

[13] M. Vetterli, "Running FIR and IIR filtering using multirate filter banks", IEEE Trans. on ASSP, Vol. 36, n° 5, May 1988, pp. 730-738.

[14] G.A. Clark, S.R. Parker, and S.K. Mitra, "A unified approach to time and frequency domain realization of FIR adaptive digital filters", IEEE Trans. on ASSP, vol. 31, n°5, pp. 1073-1083, oct. 1983.

[15] A.O. Ogunfunmi, A.M. Peterson "Fast direct implementation of block adaptive FIR filtering", Proc. ICASSP' 89, pp. 920 - 923.

[16] J.C. Lee, C.K. Un , "Block realization of multirate Adaptive Digital Filters", IEEE Trans. on ASSP , vol. 34, n°1, pp. 105 - 117, Feb. 1986.

[17] J.R. Treichler, S.L. Wood, M.G. Larimore, "Convergence Rate limitations in certain frequency-domain adaptive filters", Proc. ICASSP'89, pp. 960 - 963.

[18] G. Panda, P.M. Grant, "Rectangular Transform band adaptive filter", Electronics Letters, Vol. 21, n°7, pp. 301-303, March 1985.

[19] Z.J. Mou, P. Duhamel , "Short-length FIR filters and their use in fast FIR filtering" submitted to IEEE Trans. on ASSP.

[20] C. Caraiscos, B. Liu , "A roundoff error analysis of the LMS adaptive algorithm", IEEE Trans. on ASSP, vol. 32, n°1, Feb. 1984, pp.

34-41.

[21] J.C. Lee, C.K. Un, "Performance of transform-domain LMS adaptive digital filters", IEEE Trans. on ASSP, vol. 34, June 1986, pp. 499-510.

[22] Z.J. Mou, "Fast FIR filtering : algorithms and architectures", Doctorate thesis, University of Paris Sud, Sept. 1989.

[23] J. Benesty, P. Duhamel, "A fast constant modulus adaptive algorithm", to appear, EUSIPCO-1990.

Figure captions

Figure 1 : LMS adaptive structure

Figure 2 : An FELMS structure for block length N=2

Figure 3 : Adaptive structure used for system identification

Figure 4 : The error curves of LMS and BLMS algorithms when the input signal is a white noise and with the same adaptation step :
a) LMS algorithm
b) BLMS algorithm with a block size N=128

Figure 5 : The error curves of LMS and BLMS algorithms in the case of USASI noise input, μ being chosen to insure convergence :
a) LMS algorithm
b) BLMS algorithm with N=4. μ is divided by N
c) BLMS algorithm when N=16 and with μ divided by 16

Figure 6 : Error curve of BLMS, N=4, in the case of USASI noise input, the adaptation step is the same one as in the LMS case

Figure 7 : Error curves of the proposed algorithms in the case of USASI noise and with the same μ than LMS. The block size N is equal to 64 :
a) FELMS algorithm
b) First approximation (computation of matrix S only with 6 subdiagonals instead of 63)
c) Second approximation (blocking of the S matrix after 2048 samples)

Table caption

Table 1 : Comparison of the number of operations per output point required by the various algorithms

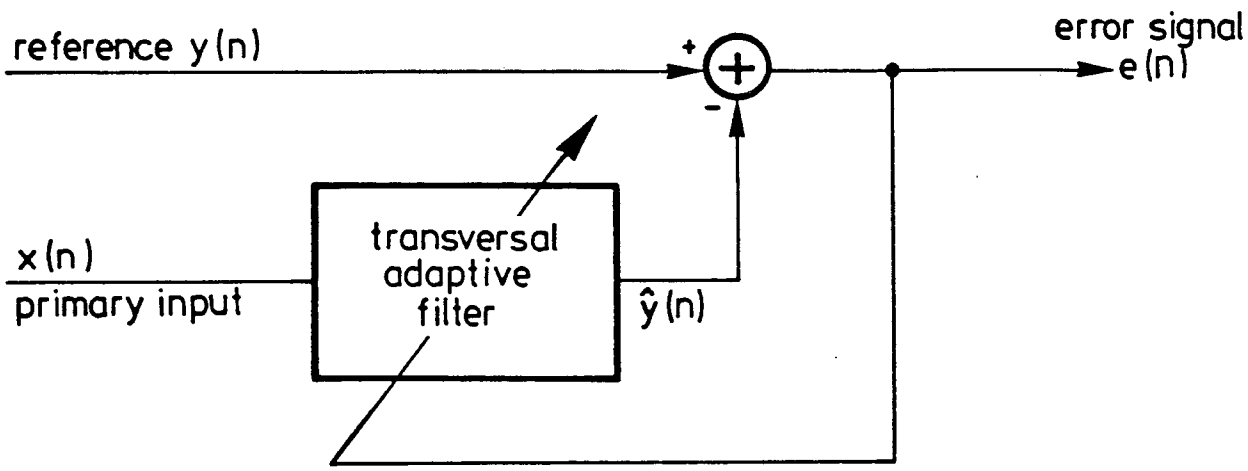


Fig.1: LMS adaptive structure

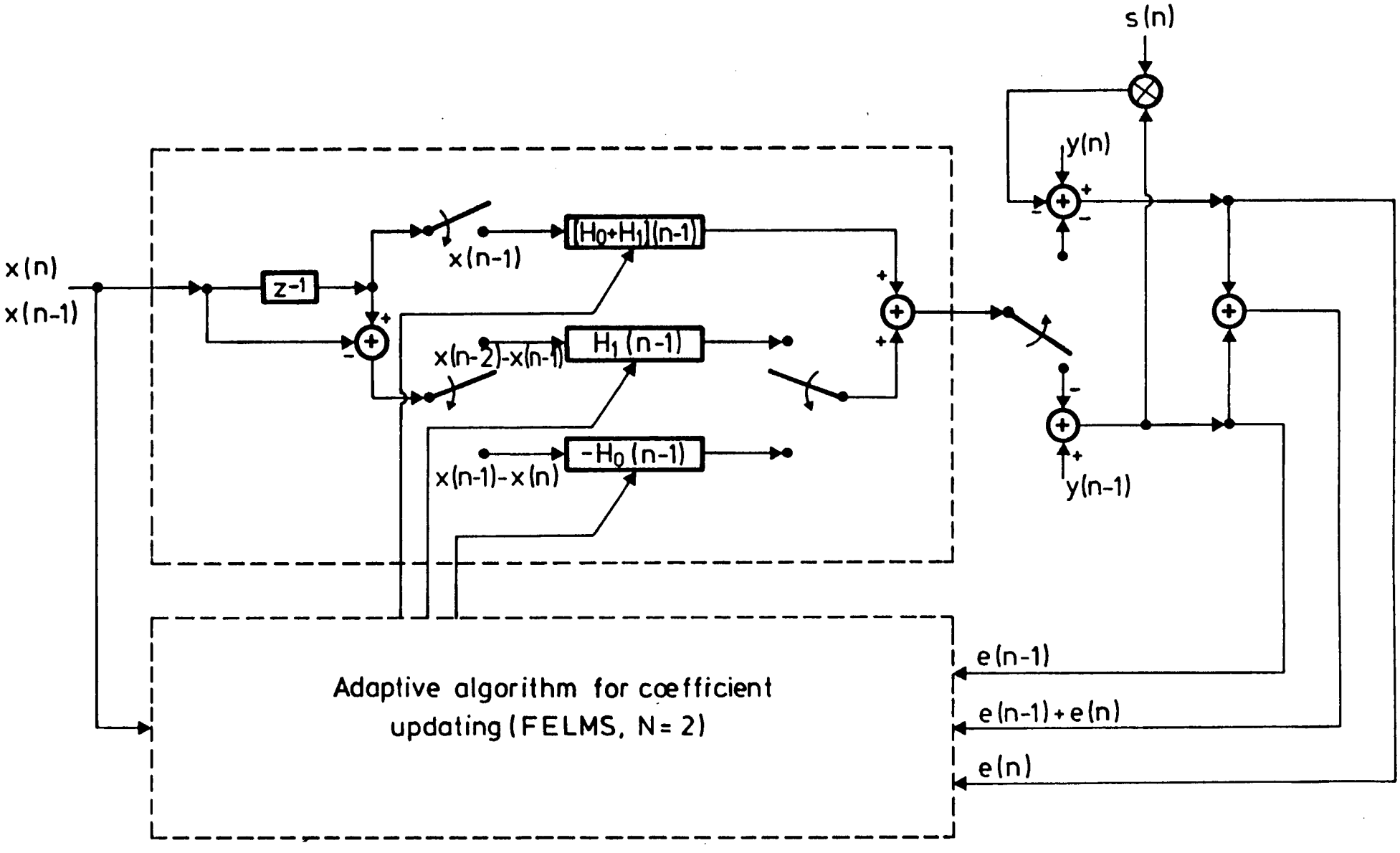


Fig.2: An FELMS structure for block length N = 2

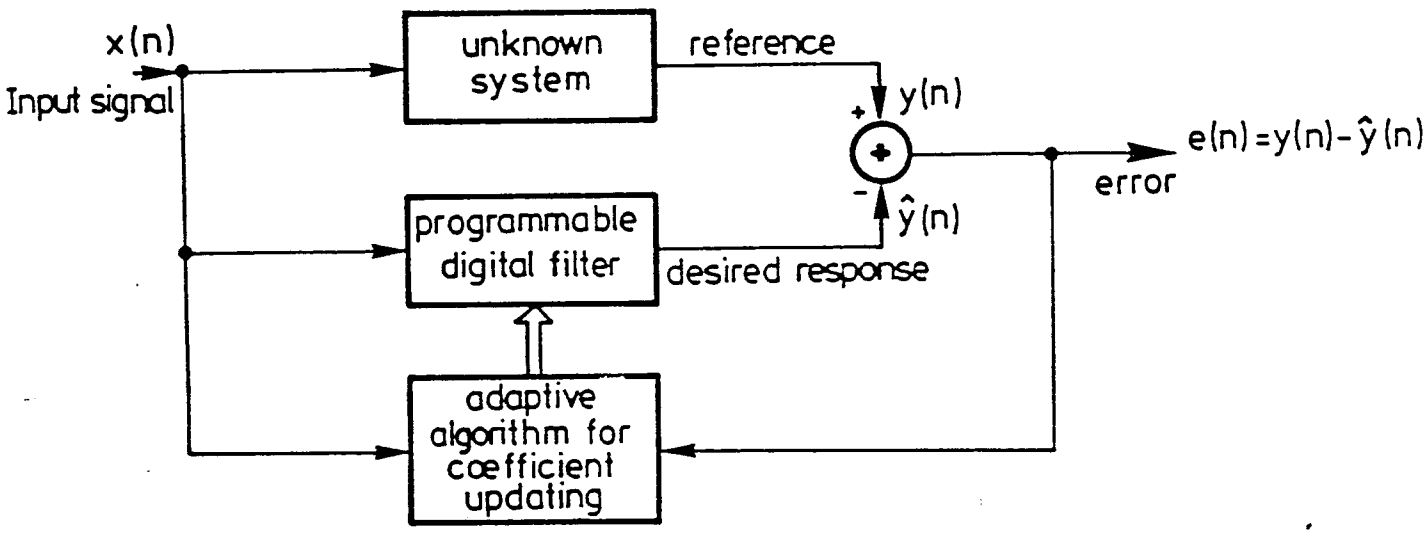


Fig.3: Adaptive structure used for system identification

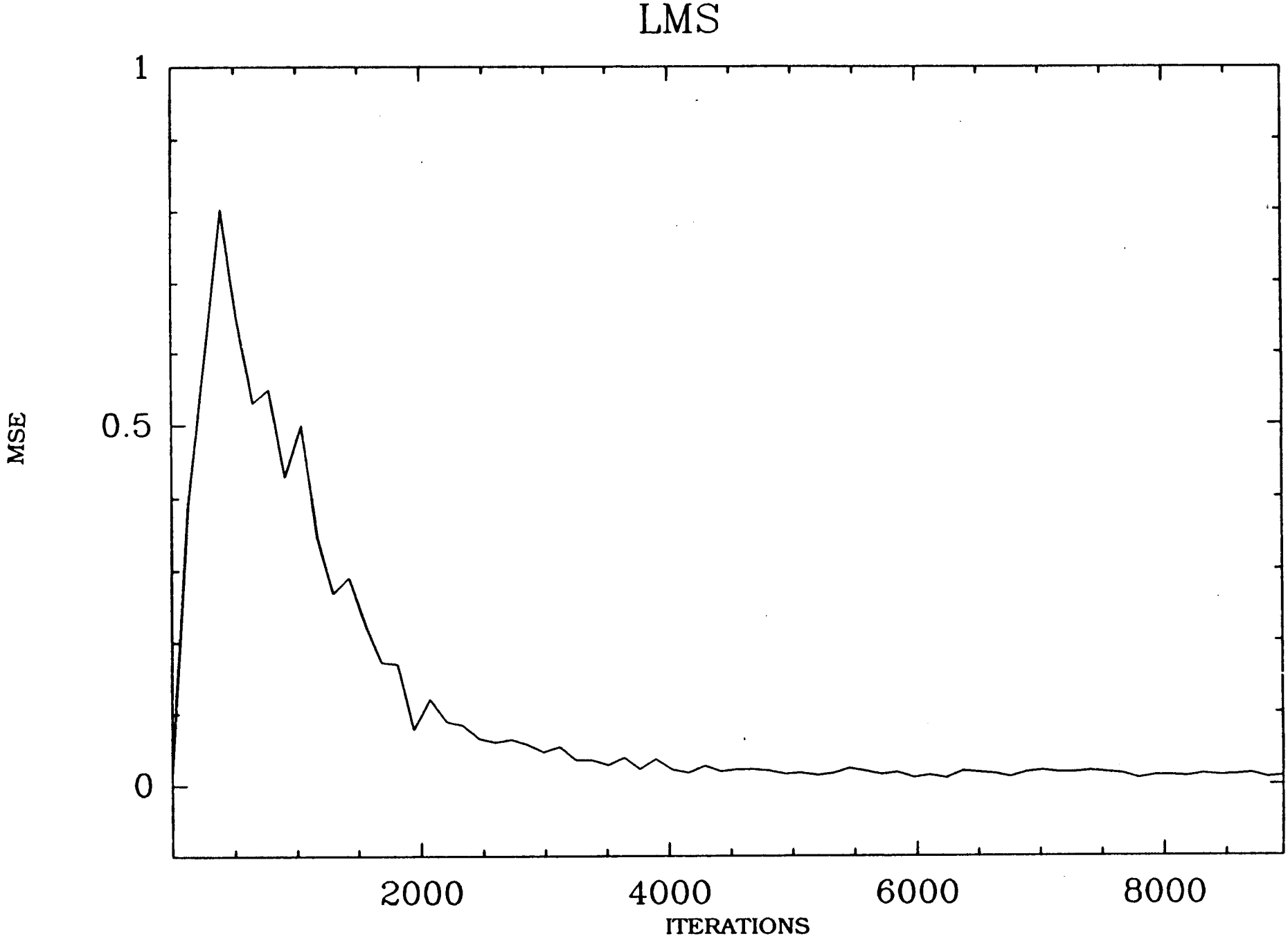
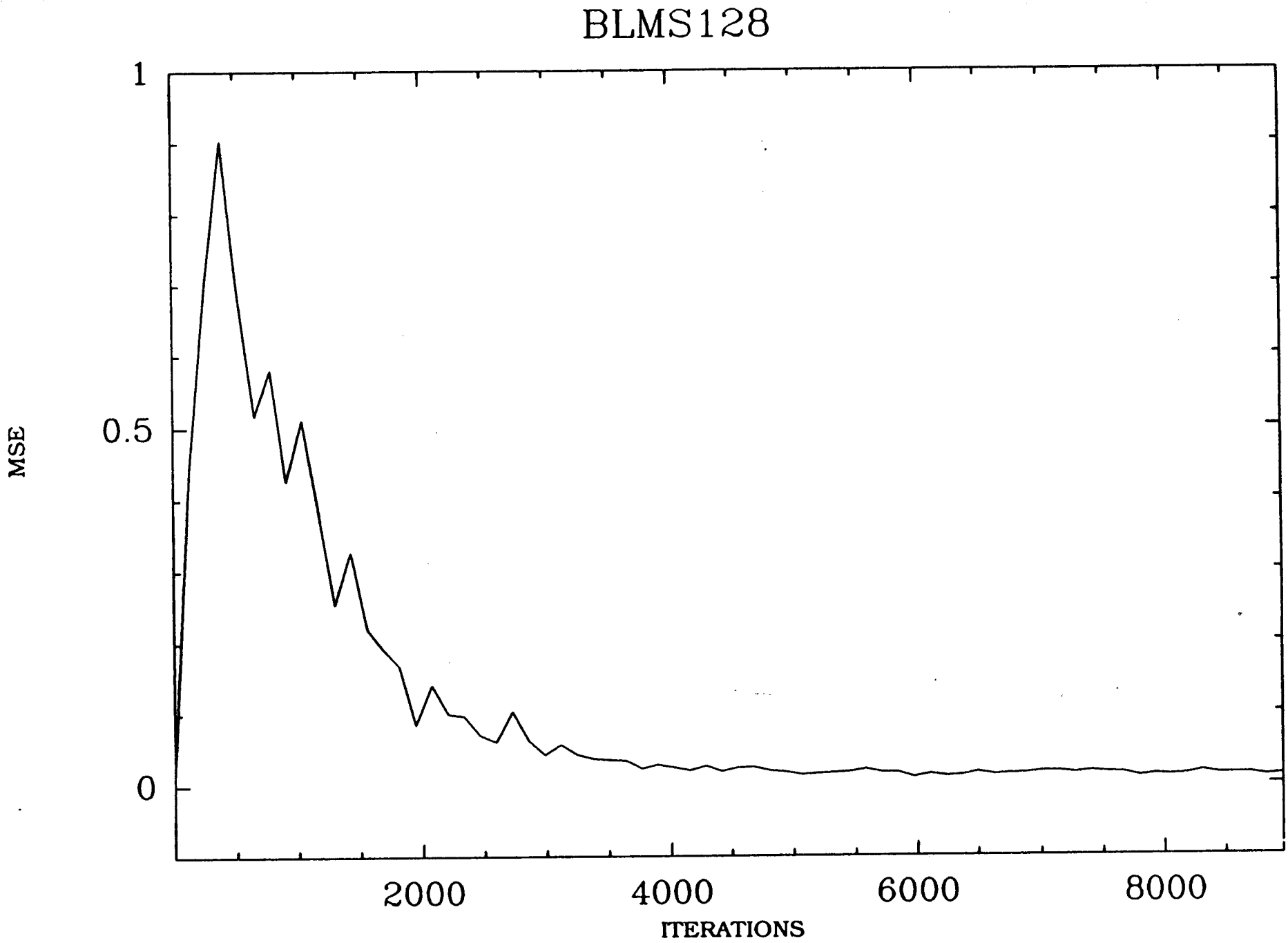

LMS
1
0.5
0
MSE
2000
4000
6000
8000
ITERATIONS
Benesty 19-JUL-1989

Fig. (4) a)


BLMS128
1
0.5
0
MSE
2000
4000
6000
8000
ITERATIONS
Benesty 19-JUL-1989

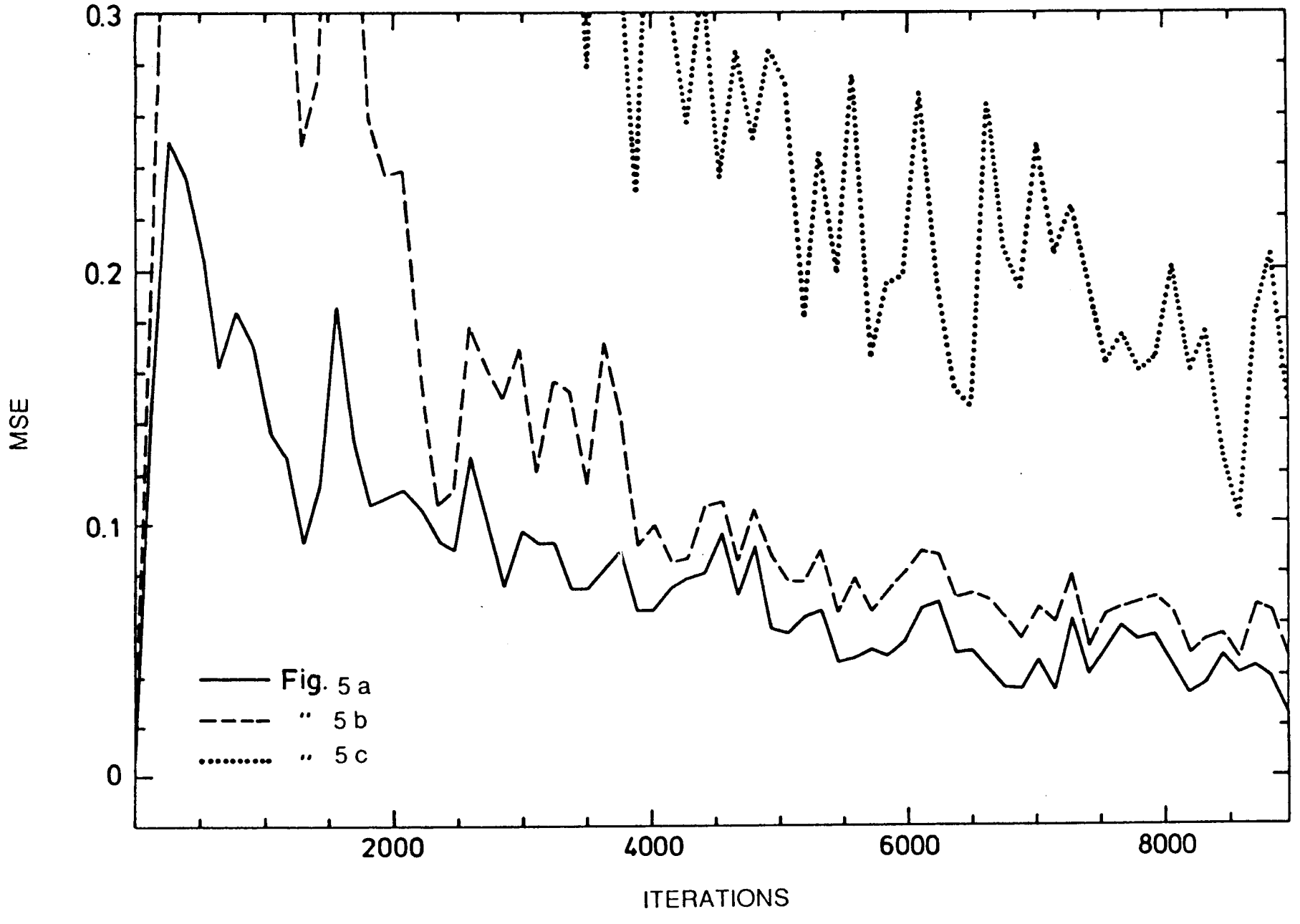
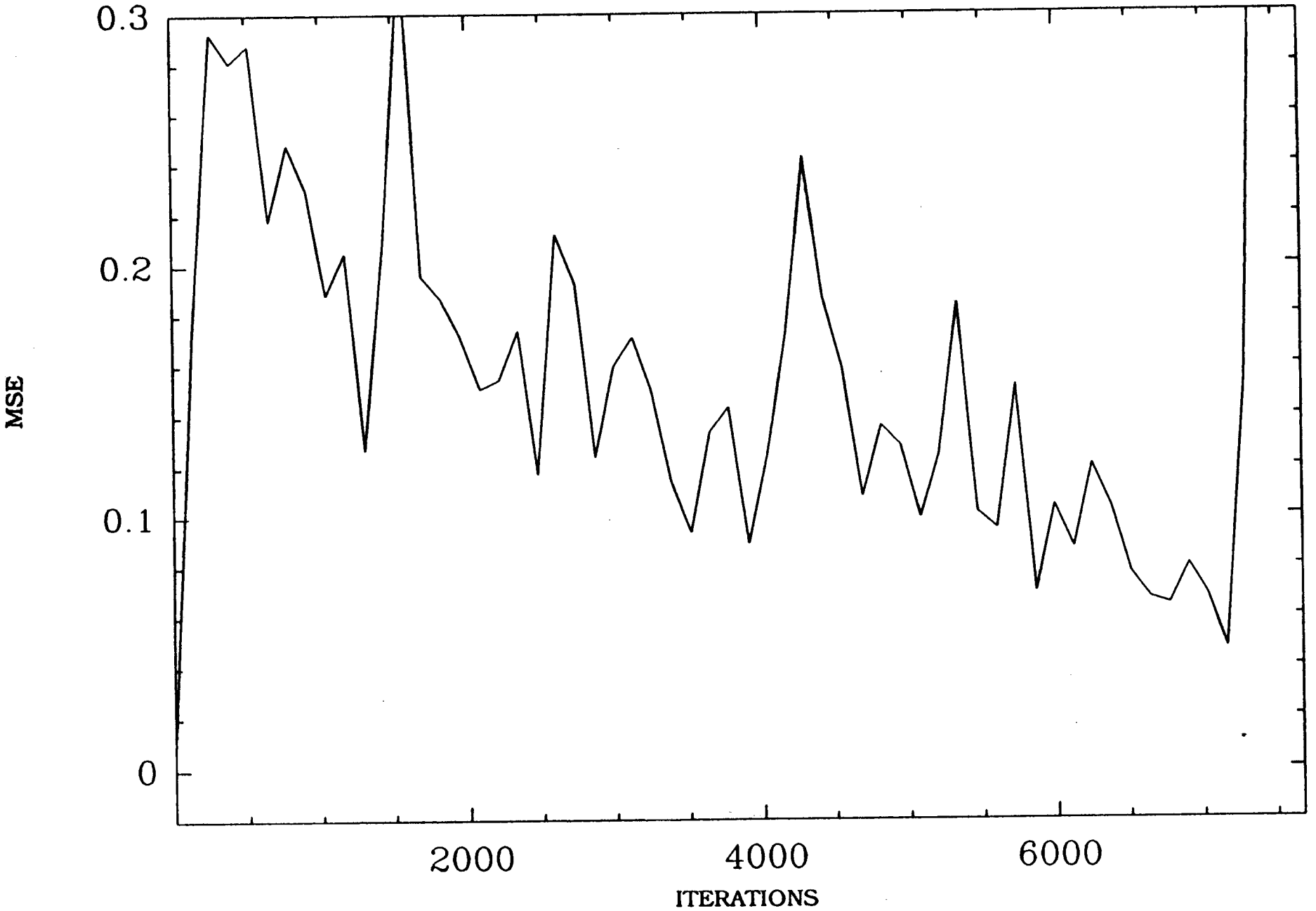


Fig. 5


BLMS4
0.3
0.2
0.1
0
MSE
2000
4000
6000
ITERATIONS
Benesty 26-APR-1989

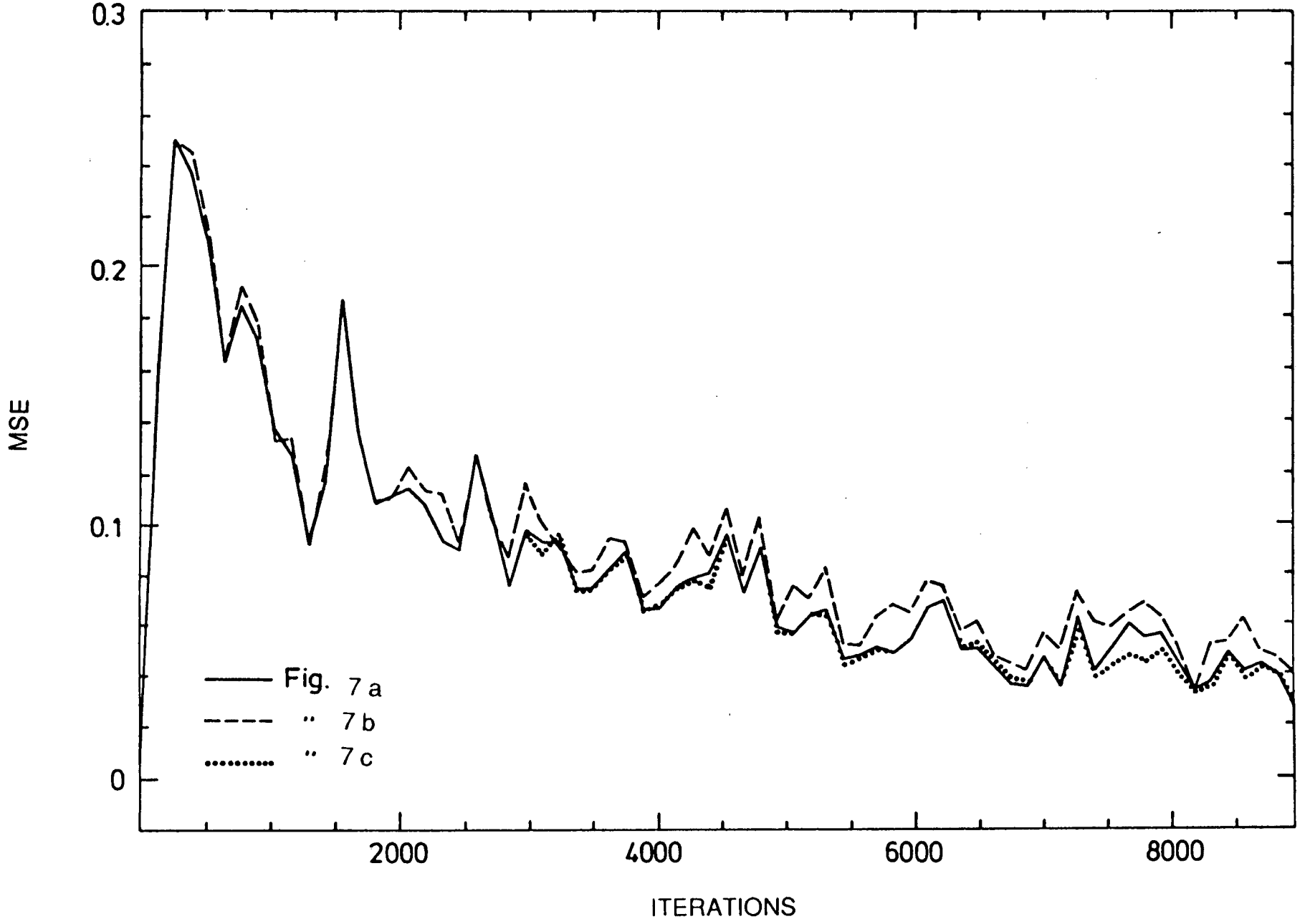


Fig. 7

		LMS algorithm		FELMS algorithm		FALMS algorithm		BLMS algorithm	
Filter length L	block length N	number of additions	number of multiplications	number of additions	number of multiplications	number of additions	number of multiplications	number of additions	number of multiplications
32	8	64	64	70	37	63	32	56	27
64	8	128	128	116	64	109	59	102	54
128	16	256	256	192	103	173	88	162	81
256	32	512	512	315	167	267	131	253	122
512	32	1024	1024	542	289	494	252	480	243
1024	64	2048	2048	865	458	756	376	739	365

Table 1 : comparison of the number of operations per output point required by the various algorithms

ARTICLE 5

A description of the central result of TAM (Toeplitz Approximation Method) leading to an improved TAM

par S. MAYRARGUE

soumis en Avril 1990 à IEEE Trans. ASSP

A Description of The Central Result of TAM (Toeplitz Approximation Method), Leading to an Improved TAM

S. MAYRARGUE and J.P. JOUVEAU

CNET/PAB/RPE
38-40, rue du Géneral Leclerc
92131 Issy-les-Moulineaux (France)

ABSTRACT : TAM is a recently developed high-resolution method for harmonic retrieval . TAM was inspired by the state-space representation in system theory. A demonstration of the central result of TAM is given, without resorting to control theory, thus relating TAM to other high-resolution methods.This demonstration leads to a modification of TAM resulting in the disappearance of spurious peaks when estimating two very close frequencies.

1.INTRODUCTION

Time series harmonic analysis is a crucial problem in a number of practical situations. Very often, data lengths are short, so that the separation between the frequencies to be retrieved may happen to be less than the Fourier resolution limit. "High-resolution" methods are thus needed. Various such methods have been devised (for instance [1], [2] , [3]) . One of the most recent is S.Y.Kung's TAM ([4], [5]).

TAM was inspired by the state-space identification problem in system theory : if a linear rational model (i.e. an ARMA (auto-regressive, moving-average) model) is assumed for a system, how to identify the model from the impulse response samples ? Classical algorithms ([6],[7]) exist to solve this problem. They make use of a Hankel matrix (H) built from the impulse response samples and compute the state-space parameters from this matrix. Indeed, the rank of this matrix is equal to p, the number of poles of the model. It can thus be decomposed into a product of two factors (the observability (O) and controllability (C) matrices), each factor having one dimension equal to p. The poles are then determined from any of these two matrices, for instance O : it has been be shown in ([6],[7]) that they are the eigenvalues of a matrix F, itself solution of a multidimensional linear system, both sides of which are obtained from O by deleting resp. the first and the last row.

In the case of noisy data, H becomes full rank. In TAM, the determination of the number of poles, and of the observalibility and controllability factors is obtained by a Principal Component approach, i.e. by a Singular Value Decomposition (SVD) ([8]) of H

giving a low-rank approximation of H. The principle of the method remains unchanged. Note that the idea of combining SVD with algorithms such as ([6],[7]) dates back to ([9]).

Another application of the above method is the retrieval of damped sinusoids from noisy data, since such a signal can be viewed as the impulse response of a system with poles inside the unit circle.

As to the harmonic (i.e. undamped sinusoids) retrieval problem, TAM uses the fact that a sum of sine waves can be viewed as the impulse response of a special ARMA model whose poles lie on the unit circle . Moreover the poles are directly related to the pulsations : if the pulsations are $\{\omega_1 , \omega_2,.....\omega_p\}$, the poles are $\{ e^{j\omega_1} ,....., e^{j\omega_p}\}$. It is also possible to use a Toeplitz matrix R built from the covariance sequences instead of the Hankel matrix of the data samples. In this case, and for noisy data, a (non-Toeplitz) low-rank approximation of R will be obtained , hence the name TAM (Toeplitz Approximation Method).

Note that TAM can also be applied in the context of antenna arrays ([10]).

In this paper, we first give a slightly different presentation of TAM, without calling for any reference to system theory, thus relating TAM to other high-resolution methods : we note that the low-rank approximation of H (or of R) obtained by SVD leads to the computation of the so-called "signal subspace" of high-resolution methods.

In the second part of the paper, we modify TAM in order to improve the miss ratio when retrieving very close frequencies. The idea is to artificially increase the frequencies spacing by multiplying each frequency by n, n being a prescribed constant.

In this aim, we modify the multidimensional linear system whose solution was F : instead of deleting resp. the first and last row of O, we delete resp. the first and last n rows. We show that F is thus replaced by F^n, and the eigenvalues of F, i.e. the $\{ e^{j\omega_i} \}$ by those of F^n : $\{ e^{jn\omega_i} \}$. Of course, this method works only if the data are oversampled, which is a realistic hypothesis in this kind of problems.

Tufts-Kumaresan's, TAM and improved TAM methods are compared on simulated data, as well as on experimental data. In both cases, improved TAM gives the smallest number of outliers.

2. KUNG's TOEPLITZ APPROXIMATION METHOD

We present the method, following the lines of the demonstration given in [11].

2.1 The Noiseless Case :

Let y be a signal composed of a sum of m complex exponentials , observed on N samples :

$y(k) = \sum_{i=1}^{m} c_i \exp [j (\omega_i k + \phi_i)]$, for k = 0, ... , N-1 (1)

where c_i , ω_i and ϕ_i are the amplitudes, pulsations and phases of the i-th complex exponential. The c_i and ϕ_i 's are real-valued , while the ω_i 's are either real-valued in the case of undamped sinusoids, or complex with a negative imaginary part for damped sinusoids. The ω_i 's are assumed to be distinct.

Let Y be a LxN-L+1 Hankel matrix built from the samples , both dimensions of Y being required to be greater than m, the number of complex exponentials . L is moreover required to be strictly greater than m :

$m+1 \leq L$, and $m \leq N-L+1$, i.e. $m+1 \leq L \leq N+1-m$

(the reasons for the additional constraint on L will become apparent below):

$$Y = \begin{bmatrix} y_0 & y_1 & & y_{N-L} \\ y_1 & y_2 & & y_{N-L+1} \\ & & & \\ & & & \\ y_{L-1} & & & y_{N-1} \end{bmatrix}$$

Let S_K be the K×m Van der Monde matrix :

$$S_K = \begin{bmatrix} 1 & \cdots & 1 \\ e^{j\omega_1} & & e^{j\omega_m} \\ & & \\ & & \\ e^{j(K-1)\omega_1} & & e^{j(K-1)\omega_m} \end{bmatrix}$$

It can be shown that $Y = S_L\ D\ S_{N-L+1}^T$ (2)

where the superscript T means transpose

and $D = \text{diag}(\ c_i \exp(j\,\phi_i)\ )$

The conditions on L compared to the number of exponentials and the fact that the ω_i are distinct ensure that S_L and S_{N-L+1} both have rank m. Hence, Y being the product of three rank-m matrices has also rank m, and its columns span the same subspace as those of S_L.

On another hand, we can write the SVD of Y (see ([8]) for definition and properties of the SVD):

$$Y = U\Sigma V^+ = \sum_{i=1}^{L} \sigma_i\, u_i\, v_i^{\,+}$$

where the superscript $^+$ means conjugate transpose, U (resp.V) is the unitary matrix containing the left (resp. right) singular vectors u_i (resp.v_i) , Σ a LxN-L+1 "diagonal" matrix containing the singular values σ_i in decreasing order.

Since Y has rank m , only m singular values are non-zero, which enables us to rewrite the decomposition of Y as follows :

$$Y = \sum_{i=1}^{m} \sigma_i\, u_i\, v_i^{\,+} = U_S \Sigma_S V_S^{+} \qquad (3)$$

where U_S is the Lxm (V_S the N-L+1xm)matrix containing the m left (resp.right) singular vectors associated to the m non-zero singular values , and Σ_S is the diagonal matrix of size m containing these singular values.The columns of U_S span the so-called "signal subspace" well-known in high-resolution methods.

The columns of Y thus span the same subspace as the columns of U_S .Since (as was seen above), these columns span the same subspace as those of S_L, we can therefore write :

$$S_L = U_S\, P \qquad (4)$$

with P a non-singular mxm matrix.

Let us now write the fundamental relationship :

$$\underset{S\downarrow}{\begin{bmatrix} 1 & \cdots & 1 \\ e^{j\omega_1} & & e^{j\omega_m} \\ & & \\ e^{j(L-2)\omega_1} & & e^{j(L-2)\omega_m} \end{bmatrix}} \underset{D}{\begin{bmatrix} e^{j\omega_1} & & \\ & \ddots & \\ & & e^{j\omega_m} \end{bmatrix}} = \underset{S\uparrow}{\begin{bmatrix} e^{j\omega_1} & \cdots & e^{j\omega_m} \\ e^{2j\omega_1} & \cdots & e^{2j\omega_m} \\ & & \\ e^{j(L-1)\omega_1} & & e^{j(L-1)\omega_m} \end{bmatrix}} \qquad (5)$$

Let us note that $S\downarrow$ (resp. $S\uparrow$) is obtained from S_L by deleting the last (resp. first) row of S_L.

Deleting the first (resp. the last) row of Eq.(4) gives rise to two new equations :

$$S\downarrow = Us\downarrow P \qquad (6)$$

and $$S\uparrow = Us\uparrow P \qquad (7)$$

Expressing (5) in terms of $Us\downarrow$ and $Us\uparrow$ with the aid of (6) and (7) gives :

$$Us\downarrow (PDP^{-1}) = Us\uparrow \qquad (8)$$

This last equation is the crux of TAM. Indeed, TAM can be summed up as follows : in the noiseless experiment, a linear relationship exists between $Us\downarrow$ and $Us\uparrow$:

$$Us\downarrow F = Us\uparrow \qquad (9)$$

and the eigenvalues of F are the $\{ e^{j\omega_i} \}$.

Since Us can be computed by Eq. (3), F and therefore the $\{ e^{j\omega_i} \}$ will be known provided that the linear system (9) is not under-determined. The m columns of $Us\downarrow$ being independent, a sufficient condition for (9) not to be under-determined is to have at least m rows : this is the case, because L was assumed to be at least equal to m+1. The unique solution F of (9) is then given by : $F = Us\downarrow^{\#} Us\uparrow$. (# means pseudo-inverse) .

2.2 The Case of a Noisy Signal

If noise is added to the signal y , we obtain a noisy signal x :

$x(k) = y(k) + n(k)$ $\qquad k = 0, \ldots , N-1$

n(k) being a complex, zero-mean, circular gaussian white noise of variance σ^2.

We define X analogously to Y :

$$X = \begin{bmatrix} x_0 & x_1 & & x_{N-L} \\ x_1 & x_2 & & x_{N-L+1} \\ & & & \\ & & & \\ x_{L-1} & & & x_{N-1} \end{bmatrix}$$

Because of noise on the data, X is now full rank.

However, X being a perturbed version of Y, m of its singular values will be far larger than the other ones, and the left singular vectors associated to these singular values will provide an estimate Us of Us. (It can be shown that these properties are asymptotically true, even for low signal-to-noise ratios, and moreover that Us is an asymptotically unbiased estimate of Us when N tends to infinity (see [12])).

An estimate of F will then be given by the lms solution of the equation $\hat{U}s\downarrow\hat{F} = \hat{U}s\uparrow$ i.e. $\hat{F}=\hat{U}s\downarrow^{\#}\hat{U}s\uparrow$ .The eigenvalues of $\hat{F}$ will be approximations of $\{e^{j\omega_1}, ..., e^{j\omega_m}\}$.

2.3 Remarks :

1) The above reasoning can be applied as well to the retrieval of damped or undamped sinusoids, as explained in the introduction.

2) In the case of undamped sinusoids, the same approach works with the data reversed and conjugated. Hence, more accurate results will be obtained by considering simultaneously both sets of data. This means using the matrix X' defined by :

$$X' = \begin{bmatrix} x_0 & x_1 & x_{N-L} & \bar{x}_{L-1} & \bar{x}_{N-1} \\ x_1 & x_2 & x_{N-L+1} & \bar{x}_{L-2} & \bar{x}_{N-2} \\ & & & & \\ & & & & \\ x_{L-1} & & x_{N-1} & \bar{x}_1 & \bar{x}_{N-L} \end{bmatrix}$$

instead of X.

This is the well-known Forward-Backward approach.([13]).

3) In the case of stochastic data (for instance if the ϕ_i are random variables), and if the autocorrelation function of the signal is perfectly known, (hypothesis H1), it is possible to replace the y(k) by the correlation coefficients $r_Y(k)$. Indeed, it is easily shown that in this case :

$$r_Y(k) = \sum_{i=1}^{m} c_i^2 \exp [j (\omega_i k)] \qquad (10)$$

A Toeplitz matrix is then built :

$$R_Y = \begin{bmatrix} r_Y(0) & r_Y(-1) & r_Y(L-1) \\ r_Y(1) & r_Y(0) & r_Y(L-2) \\ & & \\ & & \\ r_Y(L-1) & r_Y(L-2) & r_Y(0) \end{bmatrix}$$

Since it clearly results from (10) that :

$$R_Y = S_L \, \mathrm{diag} (c_i^2) \, S_L^+$$

it is easily seen that the same reasoning as was held with Y still holds with R_Y.

If the data are noisy, the noise n(k) being defined as above, we will have :

$r_X(k) = r_Y(k)$ for $k \neq 0$

$r_X(0) = r_Y(0) + \sigma^2$.

Let R_X be a Toeplitz matrix defined similarly to R_Y . R_X is full rank. Its eigenvalues are equal to those of R_Y, translated by σ^2. The eigensubspaces of R_X and R_Y are equal. The above reasoning still holds, provided that the U_S are now taken to be the eigenvectors of R_X associated to its m largest eigenvalues.

If the covariance has to be estimated, i.e. if only an approximation $\hat{R}_X$ of R_X is known, then the eigenvectors of $\hat{R}_X$ associated to its m largest eigenvalues will be taken as estimates $\hat{U}_S$ of U_S (as was done above when dealing with X instead of Y).

It is to be noted here that estimating the autocorrelation of a signal obeying hypothesis (H1) is untractable, since it implies the knowledge of different realizations of the signal (corresponding to different realizations of ϕ_i, which is clearly impossible. This hypothesis is nonetheless classically made, in order to give a sense to the computation of the autocorrelation of the signal given by Eq. (1), since such a signal is not stationary. However, the autocorrelation of y can be given a different meaning, involving the "second-order ergodicity" of y : indeed, define $r_Y(k)$ as

$$\lim_{N\to\infty} \sum_{n=k}^{N-L+k} \frac{y(n)\overline{y((n-k)}}{N-L+1} \qquad (11)$$

We prove in Appendix A that the two definitions of ry (k) are identical. This results from the fact that : $\lim_{N\to\infty} \frac{Y_N Y_N^+}{N-L+1} = R_Y$.

Thus, in practice, only an approximation of R_Y (and thus of R_X) will be available. $\hat{R}_X = X_N X_N^+ /(N-L+1)$ is such an approximation (the so-called "covariance" approximation). Since the left singular vectors of X_N are identical to the eigenvectors of $X_N X_N^+ /(N-L+1)$, applying TAM to $\hat{R}_X$ or to X_N will yield the same estimates, except when numerical stability is at stake : it is well-known that the condition number of $\hat{R}_X$ is the square of the condition number of X_N, so that working with X_N will give better numerical results than working with $\hat{R}_X$.

A better estimate is obtained using the above mentioned forward-backward approach : $\breve{R} = D_N D_N^+ / (N-L+1)$

Another approximation of R_X is $\tilde{\tilde{R}}_X$, the so-called "correlation" approximation, which is Toeplitz.

Simulation results of [6] show that $\tilde{\tilde{R}}_X$ is more robust than $\breve{R}_X$ for low signal to noise ratio, but $\tilde{\tilde{R}}_X$ brings in more bias than $\breve{R}_X$.

4) The property of X of being Hankel or of R of being Toeplitz is NEVER used in the method.

3.MODIFICATION OF TAM

We now show that a modification of TAM, arising from the structure of the demonstration in §2, results in an improvement of the miss ratio when retrieving very close frequencies. The idea is to increase artficially the frequencies spacing by multiplying each of them by n, with n a prescribed constant. Of course the data are assumed to be at least n times oversampled.For the estimation of these increased frequencies, we use a relationship similar to (5) :

$$\underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ e^{j\omega_1} & & e^{j\omega_m} \\ & & \\ e^{j(L-n-1)\omega_1} & & e^{j(L-n-1)\omega_m} \end{bmatrix}}_{S(\downarrow n)} \underbrace{\begin{bmatrix} e^{jn\omega_1} & & \\ & \ddots & \\ & & e^{jn\omega_m} \end{bmatrix}}_{D^n} = \underbrace{\begin{bmatrix} e^{jn\omega_1} & \cdots & e^{jn\omega_m} \\ e^{2j(n+1)\omega_1} & \cdots & e^{2j(n+1)\omega_m} \\ & & \\ e^{j(L-1)\omega_1} & & e^{j(L-1)\omega_m} \end{bmatrix}}_{S(\uparrow n)} \quad (12)$$

As in §2 , we can write :

$$S(\downarrow^n) = U_S(\downarrow^n)\, P \qquad (13)$$

and $$S(\uparrow^n) = U_S(\uparrow^n)\, P \qquad (14)$$

Expressing (11) in terms of $U_S(\downarrow^n)$ and $U_S(\uparrow^n)$ with the aid of (13) and (14) gives : $U_S(\downarrow^n)(P\, D^n\, P^{-1}) = U_S(\uparrow^n)$

Let $P\, D^n\, P^{-1}$ be denoted by F_n ($F_n = F^n$) . The eigenvalues of F_n are the $\{ e^{jn\omega_i} \}$.

In a noiseless experiment F_n , being the solution of $U_S(\downarrow^n) F_n = U_S(\uparrow^n)$, is given by : $F_n = U_S(\downarrow^n) \# U_S(\uparrow^n)$.

If the data are noisy, then we will use estimates of U_S as in §2, and we will take for F_n the lms solution of the equation $\hat{U}_S(\downarrow^n) F_n = \hat{U}_S(\uparrow^n)$ i.e. $\hat{F}_n = \hat{U}_S(\downarrow^n)^{\#} \hat{U}_S(\uparrow^n)$. The eigenvalues will be approximations of $\{e^{jn\omega_1}, \ldots, e^{jn\omega_m}\}$.

Here onwards, n will be referred to as the "shift", and the present modification of TAM as shifted-TAM.

Taking for n a value greater than 1 amounts to spacing the frequencies wider apart in a n ratio, (which should improve the resolution power) , but at the expense of a loss in the number of equations involved in the linear system to be solved in the lms.

We can precise these notions.

Let f(z) be the characteristical polynomial of F.

We have : $f(z_i) = 0$ with $z_i = e^{j\omega_i}$ for i =1, ... , m (15)

Now, if F is corrupted by noise, so is f and so are its roots. Let $\hat{F}$ be the estimate of F, $\hat{f}$ its characteristical polynomial and $\hat{z}_i$ its roots.We have $\hat{f}(\hat{z}_i) = 0$, which can be written as a first-order approximation. $(f+df)(z_i + dz_i) = 0$, or $f'(z_i)\, dz_i = - df(z_i)$

Therefore : $$dz_i = -\frac{df(z_i)}{f'(z_i)} = -\frac{df(z_i)}{\prod_{j\neq i}(z_i - z_j)} \qquad (16)$$

Let $\hat{f}_n(z)$ be the characteristical polynomial of $\hat{F}_n$. We have $\hat{f}_n(\hat{z}_i^n) = 0$, and as in Eq. (15) we can write :

$$d(z_i^n) = n\, z_i^{n-1} dz_i = -\frac{df_n(z_i^n)}{\prod_{j\neq i}(z_i^n - z_j^n)}$$

Let us take for instance m = 2 and assume z_1 and z_2 very close. Then :

$z_1^n - z_2^n \cong n\,(z_1 - z_2)\, z_1^{n-1}$, so that :

$$dz_1 \cong -\frac{1}{n^2}\frac{df_n(z_1^n)}{(z_1 - z_2)\, z_1^{2(n-1)}} \qquad (17)$$

Since the modulus of z_1 is 1 , a comparison between (16) and (17) shows that the modulus of dz_1 is reduced by a factor n^2 as long as both numerators : $df(z_1)$ and $df_n(z_1^n)$ keep the same order of magnitude, which can only be conjectured.

Generally speaking, increasing the value of the shift n has a drastic effect on the quality of the estimation as it increases the spacing between frequencies, though at the cost of a lesser precision in the coeficients of F_n compared to those of F, due to the loss of n rows in the linear system determining it. A trade-off is then to be expected between these two effects, leading to an optimal choice for the shift.

5. SIMULATIONS

We performed simulations using two undamped sine waves with very closely spaced frequencies embedded into additive gaussian white noise .

$$x(k) = \exp(2\pi j k \tau_1) + 0.8 \exp(2\pi j k \tau_2) + w(k)$$

$k = 1, \ldots, 64$

$\tau_1 = 0.$

$\tau_2 = 0.01$

The real and imaginary parts of the noise have a variance equal to 1.

L was taken to be 48.

TAM and modified -TAM were applied for n = 1 , 2 .... , 12. (The case n =1 corresponds obviously to the original TAM) .Tufts and Kumaresan's method ([2]) was also applied.

Let $\hat{\tau}_1$ (resp. $\hat{\tau}_2$) be the obtained estimation of τ_1 (resp.τ_2).

In order to have a meaningful estimate of the bias and variances for the different methods, we defined two failure cases : the "outliers" and the "single frequency" cases.

We defined an outlier by : $|\hat{\tau}_1 - \hat{\tau}_2| > 5|\tau_1 - \tau_2|$.

We decided that a single frequency was found if : $|\hat{\tau}_1 - \hat{\tau}_2| < .5|\tau_1 - \tau_2|$.

For 100 different pseudo-random choices of the noise samples , we obtained :

	outliers	single frequencies	mean	variance
KUMARESAN	14	4	0.0116	0.0064
TAM				
n = 1	11	7	0.0114	0.0075
n = 2	10	12	0.0114	0.0075
n = 3	8	12	0.0099	0.0048
n = 4	9	12	0.0106	0.0060
n = 5	5	14	0.0100	0.0051
n = 6	4	15	0.0107	0.0059
n = 7	1	13	0.0116	0.0080
n = 8	3	11	0.0112	0.0064
n = 9	0	14	0.0108	0.0068
n = 10	0	14	0.0115	0.0063
n = 11	0	16	0.0106	0.0063
n = 12	0	19	0.0104	0.0058

(mean and variance are computed taking into account the single frequencies, but not the outliers)

We can see that for $n \geq 9$, there are no more outliers.

For $n \geq 10$, the number of single frequency cases has a tendency to rise.

The optimum value of n can be 11 or 12 according to the criterion chosen: less many single frequency cases or : best variance and bias.

The spacing of the frequencies was under the methods resolution limit : in some cases, the program would retrieve one single frequency, in other cases, outliers. We noticed that these cases very often corresponded to eigenvalues of $\hat{F}$ very far from the unit circle. (The reason for this is clear for the single frequency cases : the characteristical polynomial of F without noise has its roots on the unit circle, therefore its coefficients have a hermitian symmetry. This symmetry is more or less conserved when noise is added. However, the roots of a symmetrical polynomial are not necessarily on the unit circle. They are inverse conjugated $z=1/\bar{z}$, which means that one of them may have a large modulus, both having the same argument).

6.EXPERIMENTAL DATA

We used 1GHz bandwidth atmospheric complex transfer functions obtained from the PACEM experiment ([14]), aimed at studying the characteristics of multiple paths affecting microwave links. Since we model these functions as superpositions of rays, the problem can be stated as follows :

$$x(k) = \sum_{i=1}^{m} a_i \, e^{2\pi j k \tau_i} + w(k) \qquad k = 1, \ldots, 63$$

where : m is the (unknown) number of rays

(a_i , τ_i) the (unknown) amplitudes and delays of the rays.

w is a complex noise which will be supposed to be white and gaussian (though the true noise, due to the tape - recorder, is in fact additive in dB) .

Physical considerations hinted that the delays were less than 1ns (Fourier resolution) .

We built a forward-backward X' matrix with L rows.

We used Tufts-Kumaresan's method with m from 8 to 16 and L = 25,32,48. The best least-mean-square fit was obtained with L = 32 , m = 15 (Fig.1).

We used TAM with m from 11 to 13 and L = 32,48. The best lms. fit was obtained with L = 48 , m = 11 (Fig. 2).

We used TAM-modified with m from 11 to 13, L = 32 , and the shift n from 1 to 8 . The best lms. fit was obtained with m = 11 , n = 6 (Fig. 3).

We can conjecture that the improvement obtained with TAM-modified and n = 6 , comes from the disappearance of outliers , as was noted in the simulations.

7. CONCLUSION

We first give a presentation of TAM , not calling for notions of system theory. This demonstration, slightly modified, gave rise to a modified TAM, thus improving the miss ratio of TAM when retrieving very close frequencies. This was illustrated by application of these methods to both simulated and real data.

APPENDIX A

We are going to prove that $r_Y(k)$ defined by Eq.(11) is identical to $r_Y(k)$ of matrix R_Y.

To start with, note that $\lim_{N\to\infty} \sum_{n=k}^{N-L+k} \frac{y(n)\overline{y((n-k)}}{N-L+1}$ is the element of the k-th row, first column of $\frac{Y_N Y_N^+}{N-L+1}$, where the index N stresses the dependency of Y on N.

Therefore, we must show that : $\lim_{N\to\infty} \frac{Y_N Y_N^+}{N-L+1} = R_Y$

Consider Eq(2) , which we rewrite : $Y_N = S_L D S_{N-L+1}^T$.

We thus have $\frac{Y_N Y_N^+}{N} = S_L D \text{ conjg}(\frac{S_{N-L+1}^+ S_{N-L+1}}{N-L+1}) \text{ conjg}(D) S_L^+$

Well-known properties of infinite length VanderMonde vectors imply :

$$\lim_{N\to\infty} \frac{S_{N-L+1}^+ S_{N-L+1}}{N-L+1} = I_m \qquad (I_m \text{ Identity matrix of dimension m})$$

wherefrom : $$\lim_{N\to\infty} \frac{Y_N Y_N^+}{N-L+1} = S_L \left|D^2\right| S_L^+ = R_Y$$

This completes the proof since the element of the k-th row, 1st column of $\lim_{N\to\infty} \frac{Y_N Y_N^+}{N-L+1}$ is precisely $r_Y(k)$ according to (11) .

REFERENCES

[1] D.W.TUFTS, R.KUMARESAN Estimation of Frequencies of Multiple Sinusoids : Making Linear Prediction Perform Like Maximum Likelihood. Proc. of the IEEE, Vol.70, N°9, Sept.1982

[2] R.KUMARESAN , D.W.TUFTS Estimating the Angles of Arrival of Multiple Plane Waves. IEEE Trans. on Aerospace Vol AES-19 N° 1 Jan.83

[3] R.SCHMIDT Multiple Emitter Location and Signal Parameter Estimation. Proc. RADC Spectral Estimation Workshop (Rome, NY, 1979) pp. 243-256

[4] S.Y. KUNG, K.S. ARUN, D.V. BHASKAR RAO, State-space and singular-value decomposition-based approximation methods for the harmonic retrieval problem. Vol. 73 n°12, Dec. 83. Journal Opt. Soc. Am.

[5] S.Y. KUNG, D.V. BHASKAR RAO, K.S. ARUN. New state space and SVD based approximate modeling methods for harmonic retrieval. IEEE 2nd ASSP Workshop on spectral Estimation, Tampa, Florida, 1983.

[6]B.L.HO and R.E.KALMAN Effective Construction of Linear State-Variable Models from Input/Output Data.Proc. Third Allerton Conference, Urbana, Illinois, 1966 pp.449-459.

[7] J.RISSANEN Recursive Identification of Linear Systems SIAM Journal CONTROL Vol.9, N°3, Aug.1971

[8] G.H.GOLUB, C.F.VAN LOAN Matrix Computations, North Oxford Academic 1983.

[9] H.P.ZEIGER, A.J.McEWEN Approximate Linear Realizations of Given Dimension Via Ho's Algorithm.IEEE Trans.Automatic Control (Tech. Notes and Corresp.) Vol.AC-19 p.153, Apr.1974. and H.P.ZEIGER (Author's Reply) IEEE Trans.Automatic Control Vol.AC-20, p.302, April 1975.

[10] J.P. LE CADRE, P.RAVAZZOLA Utilisation de modélisation d'état en traitement d'antennes. pp.385-388 GRETSI 1987

[11] S.MAYRARGUE , J.P. JOUVEAU A new application of singular-value-decomposition to harmonic retrieval. Proceedings of the International Workshop on SVD and Signal Processing 21-23 Sept.1987 Les Houches France Ed.E.F.Deprettere . North-Holland (to be published in 1988)

[12] S.MAYRARGUE An Asymptotically Unbiased Estimate For The Poles of a Rational Transfer Function submitted to IEEE Trans. on ASSP.

[13] T.J.ULRYCH , R.W.CLAYTON Time Series Modelling and Maximum Entropy, Phys. Earth Planet. Interiors, Vol.12, pp.188-200, Aug.1976.

[14] M . SYLVAIN et al. The PACEM Experiment on Line of Sight Multipath Propagation, Proc. of URSI 1983, Louvain, Belgium, ESA SP-194

FIGURES CAPTIONS

Fig.1 Amplitude of an Atmospheric Transfer Function
Bold line : experimental data .
Dotted line : reconstruction by a high-resolution method.

1a .Tufts and Kumaresan's method, 15 sources, model order = 32

1b.TAM, 11 sources, model order = 48

1c.shifted-TAM, shift = 6, 11 sources , model order = 32

TABLE CAPTION

Comparison of methods on data simulations of two very close frequencies.
Number of outliers, single frequency case. Bias and root mean square of the estimate of one of the two frequencies by each method.

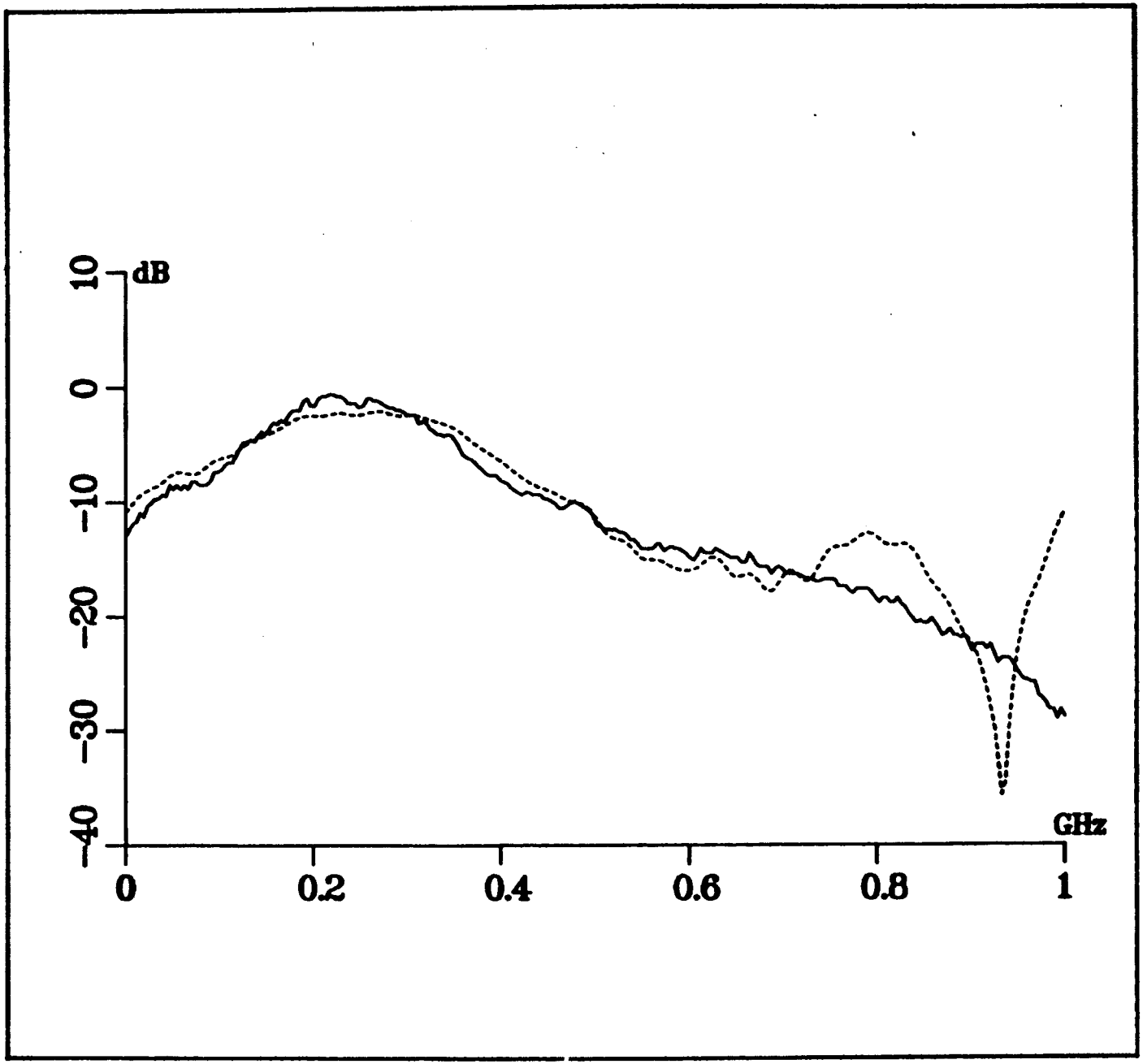


Fig 1a Tufts and Kumaresan's method

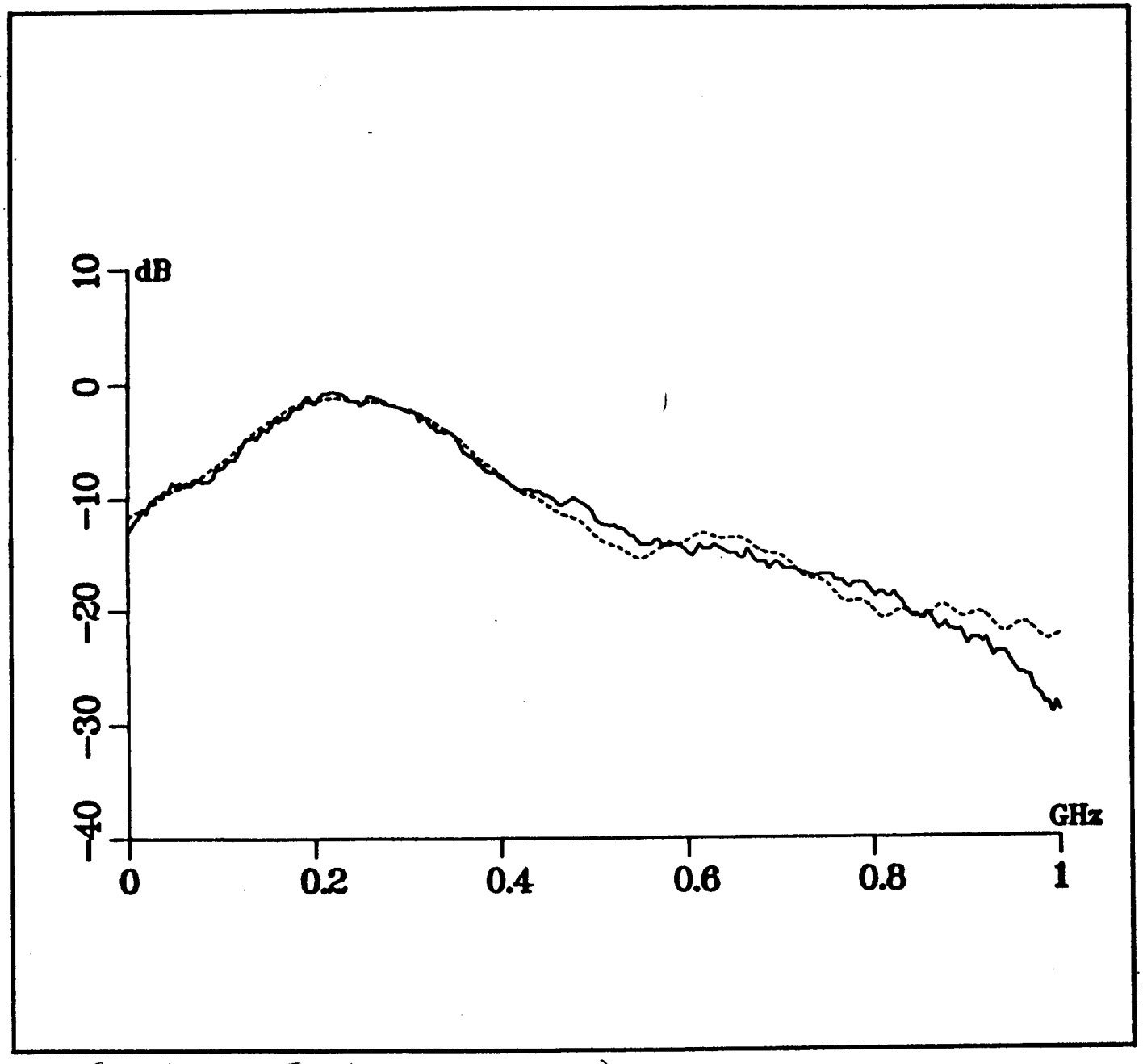


Fig 16 TAM (shift = 1)

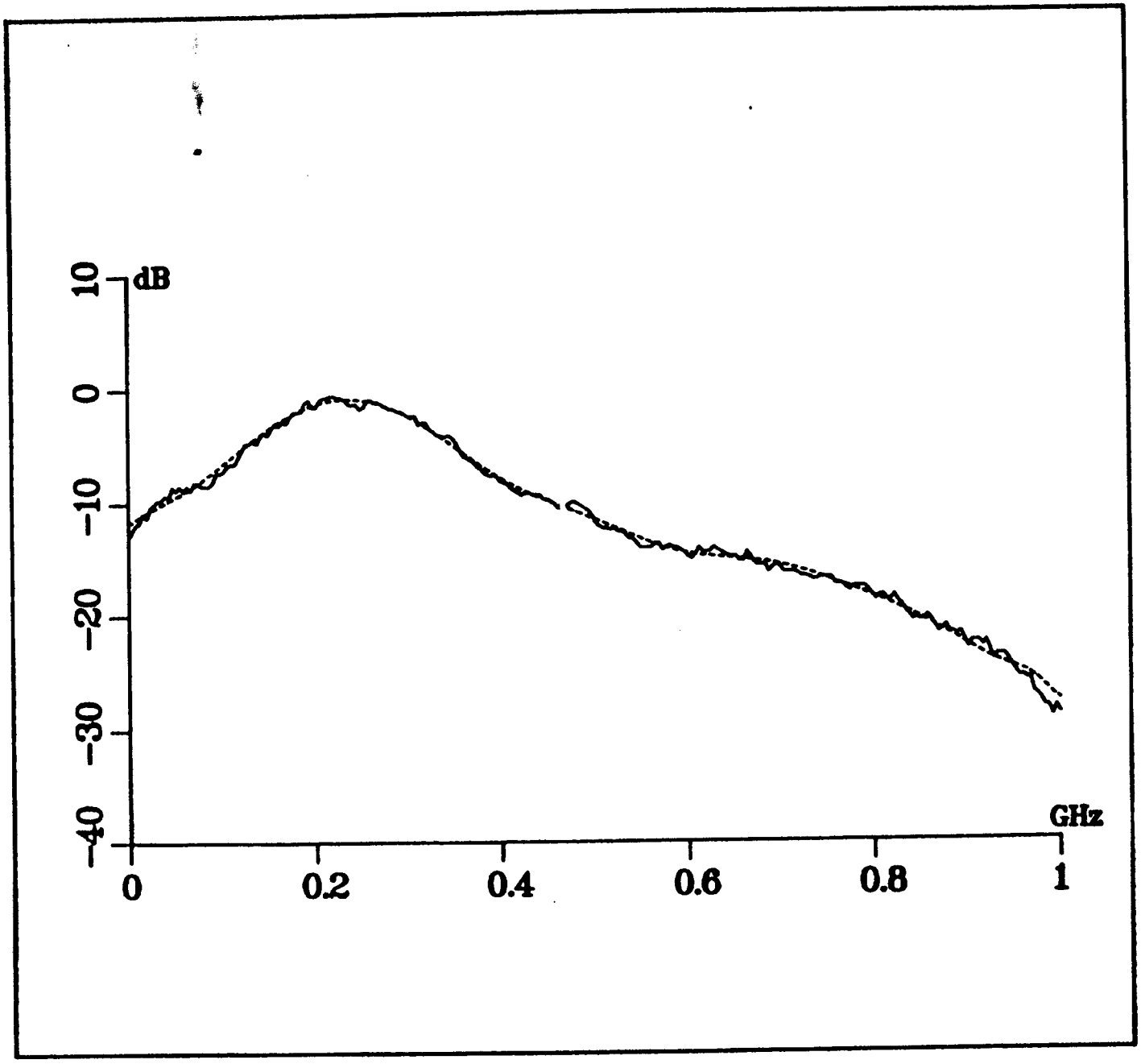


Fig 1c shifted - TAM n = 6

ARTICLE 6

Wavelets and Time-Scale Energy Distribution
par O. RIOUL et P. FLANDRIN

soumis en Mai 1990 à IEEE Trans. ASSP

WAVELETS AND TIME-SCALE ENERGY DISTRIBUTIONS

Olivier Rioul ([1]) and Patrick Flandrin ([2])

([1]) CNET/PAB/RPE, 38-40 Rue du Général Leclerc 92131 Issy-les-Moulineaux France
([2]) LTS-ICPI (URA 346 CNRS), 31 Place Bellecour 69288 Lyon Cedex 02 France

ABSTRACT

This paper presents a new general class of energetic representations of signals depending on both time and scale. Time-scale analysis has been introduced recently as a powerful tool through linear representations called (continuous) Wavelet Transforms (WTs), a concept of which we give an exhaustive bilinear generalization. Although time-scale is presented as an alternative method to time-frequency, strong links relating the two are emphasized, thus combining both descriptions into a unified perspective. We provide a full characterization of the new class: the result is expressed as an affine smoothing of the Wigner-Ville distribution, on which interesting properties may be further imposed through proper choices of the smoothing function parameters. Not only specific choices allow to recover known definitions (such as the Bertrands'), but also provide, *via* separable smoothing, a continuous transition between spectrograms and scalograms (squared modulus of the WT) *via* Wigner-Ville. This «do it yourself» property makes of time-scale representations a very flexible tool for nonstationary signal analysis.

EDICS number: 5.1.3 (*Time-varying spectral analysis*)

INTRODUCTION

A new method for time-varying signal analysis, called the *Wavelet Transform* (WT), has come under investigation during the past five years, which gives a new description of spectral decompositions *via* the *scale* concept [1]. The fundamental paper by Goupillaud, Grossmann and Morlet [2] was the first which clearly described linear time-scale decomposition of signals by means of *wavelets* of constant shape. This concept is attractive in that it is not another formulation of time-frequency ideas (where the Fourier duality is used to introduce the time-varying local frequency parameter), but a new formalism where the basic operator acting on the signal is of time-scaling nature rather than making use of the Fourier variable.

Although the first applications one could then think of were high resolution seismic analysis and coherent states representations in Quantum Mechanics, representations of signals' characteristics spread over the time-scale plane seem to be useful in a variety of fields: analysis of speech and sound signals, turbulence flows, to name but a few [1]. Because of its constant-Q type, the WT selectively matches, by means of a scalar product, transient features characterized by unknown locations and time extents. It is this property that makes it relevant for many nonstationary Signal Processing tasks, and especially for time-varying analysis. Section I briefly covers a few of these concepts.

The mathematics of the WT has been developed by Grossmann and Morlet, mostly in the language of Quantum Mechanics [3], which certainly didn't look attractive at first for the Signal Processing scientific community. Various degrees of discretization were formalized by several mathematicians such as Meyer [4], giving rise to new ideas relating discrete WTs to QMF filter bank structures, relevant for applications such as image coding. In this context, papers by Daubechies [5] and Mallat [6] are among the first that widely caught the attention to the Signal Processing people, although there had been sustantial prior work on this approach.

Even in the early stages of the theory, emphasis has been put on the fact that a relevant graphical time-scale representation of signals should include the squared modulus (or energetic) representation [7]. Although sole information on the modulus of the transform is not enough to reconstruct the general signal (phase information is also needed), the interest in the squared modulus representation (referred throughout this paper to as *scalogram*) is to provide a graphical picture of the energy of the signal spread over the time-scale plane, in a similar way as a spectrogram spreads the energy over the time-frequency plane. All such energetic representations have of course a bilinear dependence on the signal.

Other time-scale energy distributions, *viz.* bilinear forms covariant with time and scale shifts have been investigated by the Bertrands [8]. The resulting definitions are very specialized and somewhat complicated. Similar definitions occur in Grossmann's [9] and Unterberger's [10] work.

The purpose of this paper is to provide a full generalization of time-scale energy representations that gives new insights into the work of the Bertrands and better explains the relationship between scalograms and spectrograms. The general framework we present for this new approach of time-scale analysis is also expected to serve as a basis for future extensions.

Our approach follows the same lines as the derivations of the most general formulation of time-frequency energy representations, referred to as the Cohen's class [11]. We first describe the scalogram by means of members of this class and insist on similar properties handled by spectrograms. This is done in Section II.B. We then give in Section III a complete characterization of time-scale energy distributions involving affine smoothing of the Wigner-Ville Distribution (WVD). Several equivalent formulations of this may be considered, more or less compatible with additional requirements one may impose on the representation. The flexibility of this approach is finally illustrated in Section III.C by recovering several known definitions as special cases, and by deriving a new subclass of (separable) affine smoothed WVDs. This new class offers a great versatility to balance time-frequency resolution and cross-terms reduction from the WVD. A good illustration of the possibilities of our approach is the following: using Gaussian windows, we construct a sole class of time-frequency and time-scale representations in which the three well-known distributions: the WVD, the scalogram and the spectrogram are members. It is then possible, as shown in Section III-D, to pass from one well-known representation to the other *via* a continuum of this class.

I - The Short-Time Fourier Transform and the Wavelet Transform

Given a finite energy signal $x(t)$ and a sliding window $h(t)$, a classical linear *time-frequency* representation can be obtained by computing the *short-time Fourier transform* (STFT) [12]

$$F_x(t, \nu) \triangleq \int_{-\infty}^{+\infty} x(u)\, h^*(u - t)\, e^{-i2\pi\nu u}\, du \quad . \tag{1}$$

This can be interpreted as a sequence of spectral decompositions applied to successive short-time segments of the signal, as seen through the sliding window $h(t)$. Provided that this window is of finite energy and normalized such that

$$E_h \triangleq \int_{-\infty}^{+\infty} |h(t)|^2 \, dt = \int_{-\infty}^{+\infty} |H(\nu)|^2 \, d\nu = 1 \, ,$$

the squared modulus of the STFT may be interpreted as a density of energy since

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} |F_x(t, \nu)|^2 \, dt \, d\nu = E_x \quad .$$

Also, there is a one-to-one correspondence with the original signal and the exact inversion formula reads

$$x(t) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F_x(u, n) \, h(t - u) \, e^{i2\pi nt} \, du \, dn \quad . \tag{2}$$

In recent years, an alternative representation, called the *wavelet transform* (WT), has been widely addressed in the literature [1]. The fundamental idea here is to replace the frequency *shifting* operation which occurs in the STFT by a time (or frequency) *scaling* operation. The resulting definition is

$$T_x(t, a) \triangleq \frac{1}{\sqrt{|a|}} \int_{-\infty}^{+\infty} x(u) \, h^*\left(\frac{u - t}{a}\right) du \quad , \tag{3}$$

where the function $h(t)$ (called the *analyzing wavelet*) is supposed to have some localization properties in time. The explicit dependence of this definition on the dilation/compression (or *scale*) parameter a makes the WT a *time-scale* representation rather than a time-frequency one [13]. It is to be noted that, in the most general case, negative scale parameters are allowed. Their role is therefore similar to the one played by negative frequencies in Fourier analysis.

Again, $|T_x(t, a)|^2$ may be interpreted as a density of energy since

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} |T_x(t, a)|^2 \, \frac{dt \, da}{a^2} = E_x \quad ,$$

but provided that the Fourier transform $H(\nu)$ of the basic analyzing waveform $h(t)$ satisfies the so-called «admissibility condition» [3]

$$\int_{-\infty}^{+\infty} |H(\nu)|^2 \, \frac{d\nu}{|\nu|} = 1 \quad . \tag{4}$$

This basically means that $h(t)$ is necessarily the impulse response of some band-pass filter. In the time domain, its mean value must be zero, which implies that $h(t)$ will

oscillate, whence the name *wavelet*. Although a large number of wavelets can be considered, a typical choice is a *modulated Gaussian* [2].

As for the STFT, the WT satisfies then an inversion formula which reads [3]

$$x(t) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} T_x(u, a)\, \frac{1}{\sqrt{|a|}}\, h\left(\frac{t-u}{a}\right) \frac{du\, da}{a^2} \quad . \tag{5}$$

Note that, in a similar way to the STFT, the signal is reconstructed from the (possibly complex) values of the transform, *i.e.* both information on the modulus and the phase of the transform is needed.

Both the STFT and the WT analyze signals by means of an inner product with analyzing waveforms depending on two parameters. In the WT case, the waveforms onto which the signal is decomposed are of the form $1/\sqrt{|a|}\; h((u-t)/a)$ and are generated from the analyzing wavelet $h(t)$ by time-shift (t) and dilation (a) operations. It is these waveforms that are referred to as *wavelets*.

The main difference between the STFT and the WT is related to the structure of their respective analyzing waveforms. The STFT uses modulated versions of a *low-pass* filter to explore the spectral content of the analyzed signal (*uniform* filterbank). This is clearly evidenced when rewriting (1) as

$$F_x(t, \nu) = \int_{-\infty}^{+\infty} [X(n)\, e^{i2\pi nt}]\, H^*(n-\nu)\, dn \quad .$$

This amounts, in the time-domain, to using an analyzing waveform of constant envelope with an increasing number of oscillations as higher frequencies are analyzed.

The WT uses dilated or compressed versions of a *band-pass* filter, whose relative bandwidths are constant (*constant-Q* filterbank). This structure is evidenced when rewriting (3) as

$$T_x(t, a) = \int_{-\infty}^{+\infty} [X(n)\, e^{i2\pi nt}]\, \sqrt{|a|}\, H^*(an)\, dn \; .$$

Therefore, time evolutions of signals are analyzed by means of a waveform whose envelope is narrowed as higher frequencies are analyzed, whereas its number of oscillations, hence its shape, remains constant. As a consequence, the WT is a particular implemention of a constant-Q short-time spectral analysis.

In order to get a better understanding of the mathematical relationships between those two linear transforms, one can imagine to deduce STFTs from WTs, and *vice-versa*, using inversion formulæ (2) and (5). Precisely, inserting (2) (resp. (5)) in (3) (resp. (1)), we get the following transform pair [13]

$$T_x(t, a) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F_x(u, n)\, G_h(u, n; t, a)\, du\, dn \quad ;$$

$$F_x(t, \nu) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} T_x(u, a)\, G_h^*(t, \nu; u, a)\, \frac{du\, da}{a^2} \quad ,$$

with

$$G_h(u, n; t, a) = \int_{-\infty}^{+\infty} h(w - u)\, e^{i2\pi nw}\, [\frac{1}{\sqrt{|a|}}\, h(\frac{w - t}{a})]^*\, dw \quad .$$

The above transform kernel G_h is the inner product between the modulated windows of the STFT and the wavelets of the WT. In other words, it corresponds to the WT of the analyzing wavelet itself, when shifted in time and frequency, or, conversely, to the (complex conjugate of) the STFT of the corresponding analyzing window, when shifted in time and dilated or compressed.

II - SPECTROGRAMS AND SCALOGRAMS

A. Definitions and comparison

Owing to their definition, STFTs and WTs are complex-valued functions and they convey both modulus and phase informations. For some applications [7], these latter can be of interest but a description based only on the squared modulus, providing an energy density distribution, is often preferred. Indeed, the *spectrogram* $|F_x(t, \nu)|^2$, defined as the energy distribution associated to the STFT, has been widely used for many signal processing tasks. A similar quantity, $|T_x(t, a)|^2$, can be defined in the case of the WT: we propose to refer to it as a *scalogram*.

A classical time and frequency resolution trade-off underlies the structure of the STFT: the choice of an analyzing window of short duration ensures a good time localization, but at the expense of a poor frequency resolution (by Fourier duality), and *vice-versa*. Moreover, once an analyzing window has been chosen, the resolution capabilities of the spectrogram remain fixed all over the time-frequency plane. The situation is different for scalograms: owing to the constant-Q structure described above, resolution capabilities are *frequency-dependent*. This follows from the fact that, for a band-pass filter $H(\nu)$ of central frequency ν_0 and bandwidth $\Delta\nu_h$, changing the scale parameter corresponds to explore the frequency axis with a *relative* bandwidth $\Delta\nu_h / \nu_0$

kept constant and equal to a quantity referred to as the inverse $1/Q$ of the quality factor of the filter.

A symbolic comparison of spectrograms and scalograms resolutions is provided in Figure 1.

- Figure 1 -

In the narrowband limit for which, at a given scale, only the vicinity of the corresponding central frequency is analyzed, scale and frequency are inversely proportional

$$\nu/\nu' = a'/a \quad \Rightarrow \quad \nu a = \text{constant} = \nu_0 \quad .$$

In such a case, a scalogram behaves, in a first approximation, as a spectrogram whose local frequency resolution would be fitted to the analyzed frequency:

$$|T_x(t, a)|^2 \,|_{a = \nu_0/\nu} \cong |F_x(t, \nu)|^2 \,|_{\Delta\nu_h = \nu/Q} \quad .$$

B. Smoothing interpretation within the Cohen's class

Both spectrograms and scalograms have a bilinear dependence on the analyzed signal. It is the purpose of this subsection to give a simple interpretation for spectrograms and scalograms within the general class of bilinear time-frequency (shift-covariant) energy distributions. Recall that this class, referred to as *Cohen's* [11], is given by

$$C_x(t, \nu; \Pi) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_x(u, n)\ \Pi(u - t, n - \nu)\ du\ dn \quad , \qquad (6)$$

where $\Pi(t, \nu)$ is some arbitrary time-frequency function and where

$$W_x(t, \nu) = \int_{-\infty}^{+\infty} x(t + \frac{\tau}{2})\ x^*(t - \frac{\tau}{2})\ e^{-i2\pi\nu\tau}\ d\tau$$

is the so-called *Wigner-Ville distribution* (WVD). If $\Pi(t, \nu)$ behaves like a low-pass function in the time-frequency plane, the general class (6) may be considered as composed of *smoothed* versions of the WVD (we have chosen the smoothing operation to be a *correlation* one).

It is convenient to introduce 2D Fourier transformations in (6). Changing variables accordingly yields a dual characterization

$$C_x(t, \nu; \Pi) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(n, \tau)\, A_x(n, \tau)\, e^{-i2\pi(nt + \tau\nu)}\, dn\, d\tau \quad , \tag{7}$$

where the *weighting function* is defined as the 2D Fourier transform of $\Pi(t, \nu)$

$$f(n, \tau) \triangleq \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \Pi(t, \nu)\, e^{-i2\pi(nt + \tau\nu)}\, dt\, d\nu$$

and where the (narrowband) *ambiguity function* is defined as the inverse 2D Fourier transform of the WVD

$$A_x(n, \tau) = \int_{-\infty}^{+\infty} x(u + \frac{\tau}{2})\, x^*(u - \frac{\tau}{2})\, e^{i2\pi nu}\, du \quad .$$

The interest of the Cohen's class is threefold: 1) it allows to recover most of the known time-frequency energy distributions as special cases, through proper choices of characterization functions (f or Π); 2) properties of distributions within the class can be handled directly *via* properties of their associated characterization functions; 3) specific definitions can be derived by imposing constraints and translating them into requirements to be fulfilled by characterization functions.

Structure constraints of spectrograms and scalograms can be made explicit using members of the Cohen's class. The following propositions illustrate this fact. Proposition 1 is a well-known result applying to spectrograms [14] whereas Proposition 2 gives a similar specification for scalograms [15, 16]. The proofs are given in Appendices A and B, respectively.

Proposition 1. *For time-frequency energy distributions characterized by a weighting function of modulus unity, a spectrogram results from the smoothing of the signal distribution by the window distribution:*

$$\text{if } |f(n, \tau)| = 1, \text{ then } |F_x(t, \nu)|^2 = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} C_x(u, n; \Pi)\, C_h^*(u - t, n - \nu; \Pi)\, du\, dn.$$

Proposition 2. *For time-frequency energy distributions characterized by a weighting function of modulus unity which depends on its variables only through their product, a scalogram results from the affine smoothing of the signal distribution by the wavelet distribution:*

if $f(n, \tau)$ is of the form $\varphi(n\tau)$ and $|\varphi(n\tau)| = 1$, then

$$|T_x(t, a)|^2 = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} C_x(u, n; \Pi)\, C_h^*\left(\frac{u - t}{a}, an; \Pi\right) du\, dn \ . \qquad (8)$$

Several distributions satisfy the two conditions of Proposition 2. A fairly general class is that of *generalized Wigner distributions* [14] which depends on a free scalar parameter α, associated to the choice $f_\alpha(n, \tau) = e^{i2\pi\alpha n\tau}$

$$C_x(t, \nu; \Pi_\alpha) \triangleq W_x^{(\alpha)}(t, \nu) = \int_{-\infty}^{+\infty} x(t - (\alpha - \tfrac{1}{2})\tau)\, x^*(t - (\alpha + \tfrac{1}{2})\tau)\, e^{-i2\pi\nu\tau}\, d\tau \ . \qquad (9)$$

This class generalizes the WVD which is associated to the particular value $\alpha = 0$ and it includes as a special case the *Rihaczek's* distribution

$$R_x(t, \nu) \triangleq W_x^{(1/2)}(t, \nu) = x(t)\, X^*(\nu)\, e^{-i2\pi\nu t} \ .$$

Other distributions associated to complex exponentials depending on $n|\tau|$ (such as *Page's* distribution [11]) or $|n|\tau$, also satisfy the unit modulus condition and the product condition in the case of positive a's, although they do not enter directly the class (9).

III - TIME-SCALE ENERGY DISTRIBUTIONS

A. A general class and its interpretation

In order to derive the general formulation of time-scale energy distributions, it is appropriate, at this point, to interpret Proposition 2 in the restrictive case where WVDs are used: *a scalogram results from the affine smoothing of the WVD of the analyzed signal by the WVD of the analyzing wavelet.* However, scalograms are only a special case of time-scale energy distributions: it is this affine smoothing concept that enables us to generalize scalograms to general time-scale energy distributions, in a similar way as spectrograms are generalized to the Cohen's class. More precisely, consider the affine transformation:

$$[\mathbf{L}_A(t, a)h](u) = \frac{1}{\sqrt{|a|}}\, h\left(\frac{u - t}{a}\right) \ , \qquad (10)$$

where the factor $1/\sqrt{|a|}$ is introduced for normalization purposes. The main result of this paper is the following.

Proposition 3. *If a bilinear time-scale distribution $\Omega_x(t, a)$ is covariant to affine transformations, i.e. if*

$$\Omega_{L_A(\theta,\alpha)x}(t, a) = \Omega_x(\frac{t - \theta}{\alpha}; \frac{a}{\alpha}) \ ,$$

then, it is necessarily parameterized as:

$$\Omega_x(t, a; \Pi) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_x(u, n)\ \Pi(\frac{u - t}{a}, an)\ du\ dn \quad , \tag{11}$$

where $\Pi(t, \nu)$ is some arbitrary time-frequency function. Eq.(11) characterizes the general class of time-scale energy distributions.

The proof is given in Appendix C.

A similar approach has been investigated by the Bertrands [8]. Precise links between our formulation and theirs will be given in subsection III-C. It can be noted for the moment that (11) better reveals the affine smoothing concept underlying time-scale distributions and certainly is more suited for combining time-scale and time-frequency into a unified perspective.

Alternative characterizations of the class (11) may be given, depending on the domain in which the arbitrary function characterizing the distribution is expressed. For instance, if we define a bi-frequency characterization function π as

$$\pi(n, m) \triangleq \int_{-\infty}^{+\infty} \Pi(t, m)\ e^{-i2\pi nt}\ dt \ , \tag{12}$$

the equivalent formulation is

$$\Omega_x(t, a; \Pi) = \frac{1}{|a|}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \pi(n, m)\ X(\frac{1}{a}(m - \frac{n}{2}))\ X^*(\frac{1}{a}(m + \frac{n}{2}))\ e^{-i2\pi(t/a)n}\ dn\ dm. \tag{13}$$

If instead we want to characterize the class (11) with the help of the (frequency-time) weighting function f, the equivalent formulation (to be compared to (7)) reads

$$\Omega_x(t, a; \Pi) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(an, \frac{\tau}{a})\ A_x(n, \tau)\ e^{-i2\pi nt}\ dn\ d\tau \ ,$$

B. Properties

The general formulation (11) enables us to find distributions satisfying various specific requirements. This approach, which closely parallels the one used for the Cohen's class, is illustrated on some examples in the following.

1) Energy. The terminology «energy distribution» is justified by the following equality

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \Omega_x(t, a; \Pi) \frac{dt\, da}{a^2} = \left[\int_{-\infty}^{+\infty} \pi(0, m) \frac{dm}{|m|} \right] E_x \quad , \tag{14}$$

with π as in (12). This means that energy is properly spread over the time-scale plane if the quantity into brackets in the r.h.s. of (14) is unity.

2) Marginal in frequency. The spectral energy density of x is recovered from the marginal in frequency, *i.e.*

$$\int_{-\infty}^{+\infty} \Omega_x(t, a; \Pi)\, dt = \left| X\left(\frac{\nu_0}{a}\right) \right|^2 \tag{15}$$

as long as $$f(0, \tau) = e^{-i2\pi\nu_0\tau} \quad .$$

3) Marginal in time. Similarly, the instantaneous power of x is obtained as time marginal, *i.e.*

$$\int_{-\infty}^{+\infty} \Omega_x(t, a; \Pi) \frac{da}{a^2} = |x(t)|^2$$

as long as $$\int_{-\infty}^{+\infty} f(an, \frac{\tau}{a}) \frac{da}{a^2} = \delta(\tau), \quad \text{for any } n \, .$$

4) Moyal-type formula. Finally, a Moyal-type formula relating inner products of signals and distributions may be obtained as

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \Omega_x(t, a; \Pi)\, \Omega_y^*(t, a; \Pi) \frac{dt\, da}{a^2} = \left| \int_{-\infty}^{+\infty} x(t)\, y^*(t)\, dt \right|^2 \tag{16}$$

if $$\int_{-\infty}^{+\infty} f(an, \frac{\tau}{a})\, f^*(an, \frac{\tau'}{a})\, \frac{da}{a^2} = \delta(\tau - \tau') , \quad \text{for any } n . \tag{17}$$

C. Special cases

We have just seen that, in a similar way as for the Cohen's class, specific structures of characterization functions (f, Π or π) permit to investigate properties of the associated time-scale distributions. They also allow us to obtain specific definitions as special cases within the same class: some of them will now be reviewed.

1) Product kernels. Just as the WT uses band-pass filters, the smoothing function Π is preferrably chosen to be band-pass as a function of frequency. We thus define:

$$\Pi(t, \nu) \triangleq \Pi_0(t, \nu - \nu_0) \quad \Leftrightarrow \quad f(n, \tau) \triangleq f_0(n, \tau)\, e^{-i2\pi\nu_0\tau} ,$$

where ν_0 is some non-zero frequency. Using this notation, an interesting identification between time-scale and time-frequency distributions may be found, provided that the weighting function $f_0(n, \tau)$ depends only on the product $n\tau$

$$f_0(n, \tau) = \varphi(n\tau) \quad \Rightarrow \quad \Omega_x(t, a; \Pi) = C_x(t, \frac{\nu_0}{a}; \Pi_0) \quad . \tag{18}$$

As a special case, it can be easily checked that

$$\Pi_W(t, \nu) = \delta(t)\, \delta(\nu - \nu_0) \quad \Rightarrow \quad \Omega_x(t, a; \Pi_W) = W_x(t, \frac{\nu_0}{a}) ,$$

which means that the usual WVD can be recovered as the «no smoothing» limit case, with the identification «frequency = inverse of scale». The condition under which this identification holds is met by numerous distributions. In addition to the class of generalized Wigner distributions (9), we can mention the *Choï-Williams'* distribution [17], which has recently received a special attention and which is associated to the choice $f_0(n, \tau) = e^{-(n\tau)^2/\alpha}$, where α is some (positive) real-valued parameter.

2) Scalograms. A simple example is the scalogram which, according to (8), can be seen as the affine smoothing of the WVD of the analyzed signal by the WVD of the analyzing wavelet [18]

$$|T_x(t, a)|^2 = \Omega_x(t, a; W_h) \ .$$

Because the smoothing function associated to the scalogram is itself a WVD, it cannot be perfectly concentrated in both time and frequency: this results in bias on, *e.g.*, marginals. We get for instance

$$\int_{-\infty}^{+\infty} |T_x(t, a)|^2 dt = \int_{-\infty}^{+\infty} \left| X(\tfrac{n}{a}) \right|^2 |H_0(n - \nu_0)|^2 \ dn \ ,$$

if we make explicit the dependence of the band-pass filter transfer function on its central frequency ν_0 by letting $H(n) = H_0(n - \nu_0)$. This is to be compared to (15): clearly, satisfy marginal in frequency would be obtained with a highly frequency-selective analyzing wavelet. This corresponds necessarily to a large amount of time spreading and, hence, to some loss in time resolution.

Note that, in the case of the scalogram, the admissibility condition (4) is recovered from the energy condition (14).

3) Bertrands' class. Another choice yields the *Bertrands'* class [8]

$$\Omega_x(t, a; \Pi_B) = \frac{1}{|a|} \int_{-\infty}^{+\infty} \mu(u)\, X(\tfrac{1}{a}\lambda(u))\, X^*(\tfrac{1}{a}\lambda(-u))\, e^{-i2\pi(t/a)(\lambda(u)-\lambda(-u))}\, du \ , \qquad (19)$$

where $\lambda(u)$ and $\mu(u)$ are two arbitrary functions.

Several points should be noted here. 1) Eq.(19) is explicitly written in terms of time and *scale*, whereas the Bertrands' formulation uses *frequency* as a formal parameter playing the role of the inverse of scale. 2) Bertrands' approach puts emphasis on *analytic* signals and the integration in (19) is therefore limited to values of u for which $\lambda(u)$ is non-negative and to non-negative scale parameters a. Nevertheless, and for a sake of homogeneity, the above formulation will be retained throughout this paper.

Given this definition, it turns out that it enters the framework of (11) if the weighting function is chosen as

$$f_B(n, \tau) = \int_{-\infty}^{+\infty} \mu(u)\, \delta(n + [\lambda(u) - \lambda(-u)])\, e^{-i2\pi(\tau/2)(\lambda(u)+\lambda(-u))}\, du \ .$$

In order to support that claim, it is worthwhile to recall where Bertrands' class comes from. As mentioned previously, it stems from a covariance requirement (with respect to affine transformations) applied to bilinear forms of the signal. The parameterization they retain reads [8]

$$P_x(t, a; K_B) = \frac{1}{|a|} \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} K_B(n, m)\, X(\frac{1}{a} n)\, X^*(\frac{1}{a} m)\, e^{-i2\pi(t/a)(m-n)}\, dn\, dm \quad , \qquad (20)$$

where the bi-frequency kernel is proposed to be of the form :

$$K_B(n, m) = \int_{-\infty}^{+\infty} \mu(u)\, \delta(n - \lambda(u))\, \delta(m - \lambda(-u))\, du \quad . \qquad (21)$$

Therefore, the result (20) is identical to the frequency-domain formulation (13) of the general class (11), up to the reparameterization :

$$\pi(n, m) = K_B(m - \frac{n}{2}, m + \frac{n}{2}) \quad . \qquad (22)$$

The associated (frequency-time) weighting function f is then obtained by plugging (21) into (22) and by taking the partial Fourier transform over the second variable.

Since Bertrands' class can be viewed as a special case of (11), we can apply any of the specific choices corresponding to (21). In particular, we can obtain the following distribution [8]

$$B_x(t, a) = \frac{1}{|a|} \int_{-\infty}^{+\infty} \frac{(n/2)}{\sinh(n/2)}\, X(\frac{1}{a}\frac{(n/2)\, e^{-(n/2)}}{\sinh(n/2)})\, X^*(\frac{1}{a}\frac{(n/2)\, e^{+(n/2)}}{\sinh(n/2)})\, e^{-i2\pi(t/a)n}\, dn \, , \quad (23)$$

which is associated to

$$\lambda(u) = \frac{(u/2)\, e^{-(u/2)}}{\sinh(u/2)} \quad , \quad \mu(u) = \frac{(u/2)}{\sinh(u/2)} \quad .$$

However, if we are to end up with distributions such as (23), it appears that the formulation (21-22) is unnecessarily complicated. This situation will be considered and simplified in the following subsection.

4) Localized bi-frequency kernels. A useful sub-class of (11) consists in characterization functions which are perfectly localized on some curve $m = F(n)$ in their bi-frequency representation:

$$\pi_\delta(n, m) \triangleq G(n)\, \delta(m - F(n)) \quad \Leftrightarrow \quad f_\delta(n, \tau) \triangleq G(n)\, e^{-i2\pi F(n)\tau} \, , \qquad (24)$$

where $G(n)$ is an arbitrary function. The associated time-scale distributions then read

$$\Omega_x(t, a; \Pi_\delta) = \frac{1}{|a|} \int_{-\infty}^{+\infty} G(n)\, X(\frac{1}{a}[F(n) - \frac{n}{2}])\, X^*(\frac{1}{a}[F(n) + \frac{n}{2}])\, e^{-i2\pi(t/a)n}\, dn \quad .$$

Within this formulation, it can be checked that specifying

$$G(n) = \frac{(n/2)}{\sinh(n/2)} \quad ; \quad F(n) = (n/2)\coth(n/2)$$

allows to recover the particular Bertrands' distribution (23). More important is the fact that this specific definition may be constructed starting from a localized bi-frequency kernel by imposing *a priori* requirements and by retaining the characterization function which fulfills the associated structure constraints. This is detailed in Appendix D, making use of the results of subsection III-B to take into account the requirements (namely *time-localization* and a *Moyal-type* formula) which led the Bertrands to (23).

Furthermore, a number of additional properties of distributions can be directly checked by inspection of (24). For instance, the energy distribution condition (14) becomes

$$\int_{-\infty}^{+\infty} \pi_\delta(0, m)\, \frac{dm}{|m|} = \frac{G(0)}{|F(0)|} = 1 \quad .$$

This clearly holds for the Bertrands' case (23), as well as the marginal condition (15) with the arbitrary normalization $\nu_0 \equiv 1$ Hz.

5) Separable kernels and affine smoothed Wigner-Ville. It is known [19] that the trade-off underlying the time and frequency behaviors of the spectrogram can be overcome if we replace the associated WVD smoothing by a smoothing function which is *separable* in time and frequency

$$\Pi_0(t, \nu) = g(t)\, H_0(\nu).$$

The resulting distribution (called the *smoothed pseudo-WVD*) reads

$$C_x(t, \nu; \Pi_0) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_x(u, n)\, g(u - t)\, H_0(n - \nu)\, du\, dn$$

$$= \int_{-\infty}^{+\infty} h_0(\tau) \left[\int_{-\infty}^{+\infty} g(u - t)\, x(u + \frac{\tau}{2})\, x^*(u + \frac{\tau}{2})\, du \right] e^{-i2\pi\nu\tau}\, d\tau. \qquad (25)$$

This offers a great versatility for balancing *e.g.* time-frequency resolution and cross-terms reduction [19], although this is necessarily at the expense of the loss of other properties such as marginals.

We propose a similar approach for time-scale distributions and define the *affine smoothed WVD* by taking

$$\Pi_S(t, \nu) = \Pi_0(t, \nu - \nu_0) = g(t)\, H_0(\nu - \nu_0)\,.$$

The resulting definition is

$$\Omega_x(t, a; \Pi_S) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_x(u, n)\, g(\frac{u - t}{a})\, H_0(an - \nu_0)\, du\, dn \quad . \tag{26}$$

For practical calculations, an equivalent form, which parallels that of (25) and which relies on some prior affine smoothing pertaining to the Wigner-Ville kernel, is to be preferred.

$\Omega_x(t, a; \Pi_S) =$

$$\int_{-\infty}^{+\infty} \frac{1}{\sqrt{|a|}} h_0(\frac{\tau}{a}) \left[\int_{-\infty}^{+\infty} \frac{1}{\sqrt{|a|}} g(\frac{u - t}{a})\, x(u + \frac{\tau}{2})\, x^*(u + \frac{\tau}{2})\, du \right] e^{-i2\pi(\nu_0/a)\tau}\, d\tau.$$

D. From spectrograms to scalograms via Wigner-Ville

The smoothing functions acting on the WVD to obtain spectrograms on one hand and scalograms on the other hand are found, by Propositions 1 and 2, to be of the form of a WVD. This suggests a continuous transition from spectrograms to scalograms *via* the WVD by suitably controlling the evolution of the (affine) smoothing function between a WVD and a delta function. The following Proposition shows that this can be achieved using separable kernels, which allow an independent control of the time and frequency (or scale) behaviors of the associated distributions.

Proposition 4. *A continuous passage from spectrograms to scalograms via Wigner-Ville is possible by means of separable kernels if and only if these latter are Gaussian :*

$$\Pi_S(t, \nu) = \frac{\sqrt{\alpha\beta}}{\pi} e^{-\alpha t^2}\, e^{-\beta(\nu - \nu_0)^2}\,,$$

where α, β and ν_0 are (positive) real-valued parameters.

The proof is given in Appendix E.

The transition is controlled by the parameter $\mu = 2\pi/\sqrt{\alpha\beta}$, which runs from 0 (WVD) to 1 (spectrogram/scalogram). An example of this transition is illustrated in Figure 2 which presents several analyses of a computed example (consisting of three Gaussian wave packets) resembling the one symbolically used in Figure 1.

- Figure 2 -

CONCLUSION

The material presented in this paper is based on similar properties of two notions that are both defined by covariance requirements: 1) *local frequency*, covariant under modulations (or Fourier-frequency shifts); 2) *time scaling* (or frequency scaling by Fourier duality), covariant under dilations or contractions.

This similarity is evidenced by describing general time-frequency and time-scale energy distributions in a unified way as the result of some 2D correlation acting on the WVD. The WVD thus appears to play a central role in both analyses since it is (among a few other members of the Cohen's class, see Section II.B) covariant under frequency shifts as well as under scale dilations. Our specific choice of putting emphasis on the WVD is motivated, among other things, by the fact that it is real-valued.

The WVD thus belongs to both classes of time-frequency and time-scale distributions with the simple identification «scale = inverse of frequency». This is well illustrated by the last result presented in this paper, which shows a continuous transition from spectrograms to scalograms with the WVD as a middle step. In light of this, we recommend that various properties of time-frequency and time-scale methods be compared keeping in mind that both result from a smoothing (more generally, a correlation) operation acting on a common kernel (the WVD), the difference being related to the nature of the smoothing operation used: time-frequency or affine (time-scale) smoothing. Moreover, this continuous transition permits to balance time-frequency resolution and cross-terms reduction in the time-scale representation, in a similar (but different) way as for the smoothed pseudo-WVD. Other specific requirements (such as energy normalization, time marginal, *etc.*) and associated parameterizations of the representation were also studied in this paper. This results in a great versatility for the choice of representations appropriate for various applications.

Since a large class of time-scale and time-frequency representations is now available, with many possible (and sometimes, exclusive) properties, some analysis should be done on the analysis tool itself in order to express particular needs: starting from the most general formulation, one can, for instance, build a subset of time-scale energy representations, suitable for a given application, by imposing specific requirements. Controlling a few parameters on this set of analyses should help in many ways, *e.g.* for determining which representation best reveals a given time-scale signature.

APPENDIX A
Proof of Proposition 1

It is known that Cohen's class can be derived from a shift-covariance requirement with respect to both time and frequency [20]. Therefore, if we introduce the time-frequency shift operator

$$[\mathbf{L}_{\mathrm{WH}}(t, \nu)h](u) = h(u - t)\, e^{i2\pi\nu u} \quad , \tag{A1}$$

one has

$$C_{\mathbf{L}_{\mathrm{WH}}(t, \nu)h}(u, n; \Pi) = C_h(u - t; n - \nu; \Pi).$$

On another hand, *Moyal's formula* [11] guarantees that, for any two finite energy signals,

$$\left| \int_{-\infty}^{+\infty} x(t)\, y^*(t)\, dt \right|^2 = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_x(u, n)\, W_y(u, n)\, du\, dn$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} A_x(n, \tau)\, A_y^*(n, \tau)\, dn\, d\tau \quad . \tag{A2}$$

Using (7) and applying Parseval's equality, we obtain

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} C_x(u, n; \Pi)\, C_y^*(u, n; \Pi)\, du\, dn =$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} |f(n, \tau)|^2\, A_x(n, \tau)\, A_y^*(n, \tau)\, dn\, d\tau$$

and, hence, a generalized Moyal's formula holds for all members of Cohen's class which are characterized by a weighting function of modulus unity [14]. This is true especially if $y(t)$ is chosen as in (A1): in such a case, the left-hand side of (A2) identifies to the spectrogram, which completes the proof. ■

APPENDIX B
Proof of Proposition 2

According to the definition (7), one has

$$C_h(\frac{u-t}{a}, an; \Pi) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(m, \tau)\, A_h(m, \tau)\, e^{-i2\pi(m(u-t)/a + \tau an)}\, dm\, d\tau$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(am, \frac{\tau}{a})\, [A_h(am, \frac{\tau}{a})\, e^{i2\pi mt}]\, e^{-i2\pi(mu + \tau n)}\, dm\, d\tau..$$

It can be easily checked that the quantity into brackets corresponds to the ambiguity function of $h(t)$ after action of the affine transformation (10). Therefore, we obtain

$$C_h(\frac{u-t}{a}, an; \Pi) = C_{L_{A(t,a)}h}(u, n; \Pi)\,,$$

provided that

$$f(an, \frac{\tau}{a}) = f(n, \tau) \qquad \text{(B1)}$$

for any a. Weighting functions satisfying (B1) are functions of the product of their variables. The proof is completed by considering again the unit modulus condition under which Moyal's formula holds. ■

APPENDIX C
Proof of Proposition 3

Assume the following covariance requirement relative to affine transformations (10) is imposed to a time-scale distribution Ω_x

$$\Omega_{L_{A(\theta,\alpha)}x}(t, a) = \Omega_x(\frac{t-\theta}{\alpha}; \frac{a}{\alpha})\,, \qquad \text{(C1)}$$

If the desired distribution Ω_x is supposed to be constructed as a bilinear form in x characterized by some kernel K, then

$$\Omega_x(t, a) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} K(u, u'; t, a)\, x(u)\, x^*(u')\, du\, du'$$

and it follows from the covariance requirement (C1) that

$$\alpha\, K(\alpha u + \theta, \alpha u' + \theta; t, a) = K(u, u'; \frac{t - \theta}{\alpha}, \frac{a}{\alpha})$$

for any α, any θ and any a, or, equivalently,

$$K(u, u'; t, a) = \frac{1}{a} K(\frac{u - \theta}{\alpha}, \frac{u' - \theta}{\alpha}; t - \theta, a) \ .$$

If we fix $\theta = t$ and $\alpha = a$, this implies that the kernel must verify

$$K(u, u'; t, a) = \frac{1}{a} K(\frac{u - t}{a}, \frac{u' - t}{a}; 0, 1).$$

Therefore, one has

$$\Omega_x(t, a) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \frac{1}{a} K(\frac{u - t}{a}, \frac{u' - t}{a}; 0, 1)\, x(u)\, x^*(u')\, du\, du'$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \frac{1}{a} K(\frac{\theta - t}{a} + \frac{\tau}{2a}, \frac{\theta - t}{a} - \frac{\tau}{2a}; 0, 1)\, x(\theta + \frac{\tau}{2})\, x^*(\theta - \frac{\tau}{2})\, d\theta\, d\tau$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_x(u, n)\, \Pi(\frac{u - t}{a}, an)\, du\, dn \quad ,$$

with the smoothing function Π defined as

$$\Pi(t, \nu) \triangleq \int_{-\infty}^{+\infty} K(t + \frac{\tau}{2}, t - \frac{\tau}{2}; 0, 1)\, e^{i2\pi\nu\tau}\, d\tau \ .$$

This completes the proof. ■

APPENDIX D
Derivation of Bertrands' definition

Assume our time-scale distributions are characterized by a kernel function whose bi-frequency representation is localized according to :

$$\pi_\delta(n, m) \triangleq G(n)\, \delta(m - F(n)) \quad . \tag{D1}$$

Further specifications of this kernel function can be obtained by imposing specific requirements to the corresponding distribution.

1) Time localization. Imposing time localization (in Bertrands' sense [8])

$$X(\nu) = \frac{1}{\sqrt{|\nu|}} e^{-i2\pi\nu t_0} \ , \ \Rightarrow \ \Omega_x(t, a; \Pi_\delta) = |a|\ \delta(t - t_0) \ ,$$

yields

$$\Omega_x(t, a; \Pi_\delta) = |a| \int_{-\infty}^{+\infty} \frac{G(an)}{\sqrt{F^2(an) - (an/2)^2}} e^{i2\pi n(t-t_0)}\, dn$$

and, hence, the condition :

$$G^2(n) = F^2(n) - (n/2)^2 \quad . \tag{D2}$$

2) Moyal-type formula. If we impose the condition (17) for the Moyal-type formula (16) to hold, we obtain, within the structure (D1),

$$\int_{-\infty}^{+\infty} f(an, \frac{\tau}{a}) f^*(an, \frac{\tau'}{a}) \frac{da}{a^2} = \int_{-\infty}^{+\infty} G^2(an)\, e^{i2\pi \frac{F(an)}{a}(\tau - \tau')} \frac{da}{a^2}$$

$$= \int_{-\infty}^{+\infty} \frac{G^2(an)}{a^2 \frac{d}{da}(\frac{F(an)}{a})} e^{i2\pi \frac{F(an)}{a}(\tau - \tau')}\, d(\frac{F(an)}{a}) \ .$$

Therefore, (17) is satisfied if

$$G^2(n) = F(n) - n\,\frac{dF}{dn}(n) \quad . \tag{D3}$$

The simultaneous requirements of 1) time localization (D2) and 2) Moyal-type formula (D3) lead to the differential equation

$$F(n) - n\,\frac{dF}{dn}(n) = F^2(n) - (n/2)^2 \quad ,$$

which is equivalent to

$$U(n)\,\frac{dV}{dn}(n) - V(n)\,\frac{dU}{dn}(n) = U(n)\,V(n) \quad , \tag{D4}$$

after introducing the auxiliary functions

$$U(n) \triangleq F(n) - (n/2) \quad ; \quad V(n) \triangleq F(n) + (n/2) \quad .$$

Moreover, if both sides of (D4) are divided by $V(n)$ and if we introduce a new function $W(n) \triangleq U(n)/V(n)$, we get

$$\frac{dW}{dn}(n) = W(n) \quad ,$$

whose solution reads

$$W(n) = \frac{F(n) - (n/2)}{F(n) + (n/2)} = c\,e^n \quad . \tag{D5}$$

It follows from (D5) that $W(0) = 1$. This implies $c = 1$ which, in turn, implies

$$F(n) = (n/2)\coth(n/2) \quad . \tag{D6}$$

We then deduce from (D3) and (D6) that

$$G^2(n) = ((n/2)\frac{e^n + 1}{e^n - 1} - (n/2))\;((n/2)\frac{e^n + 1}{e^n - 1} + (n/2) = \frac{n^2\,e^n}{e^n - 1} \tag{D7}$$

and, hence,
$$G(n) = \frac{(n/2)}{\sinh(n/2)} \quad .$$

This completes the proof. ■

APPENDIX E
Proof of Proposition 4

For a sake of convenience, let us introduce the notation

$$H(\nu) \triangleq H_0(\nu - \nu_0)$$

which allows us to consider (25) and (26) in a common framework, with either H_0 for the smoothed pseudo-WVD or H for the affine smoothed WVD. If both spectrograms and scalograms are supposed to be attainable through separable kernels, then their associated smoothing function, which is a WVD, must necessarily be itself a separable function of time and frequency. However, if we impose to a WVD to be separable, *e.g.*

$$W_x(t, \nu) = g(t)\, H_0(\nu) \ , \tag{E1}$$

we readily obtain

$$|X(\nu)|^2 = \int_{-\infty}^{+\infty} W_x(t, \nu)\, dt = G(0)\, H_0(\nu)$$

and

$$|x(t)|^2 = \int_{-\infty}^{+\infty} W_x(t, \nu)\, d\nu = g(t)\, h_0(0) \ .$$

Therefore, a separable WVD is necessarily of the form

$$W_x(t, \nu) = \frac{|x(t)|^2\, |X(\nu)|^2}{G(0)\, h_0(0)} \ . \tag{E2}$$

From (E1), it follows that

$$G(0)\, h_0(0) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} W_x(t, \nu)\, dt\, d\nu = E_x \geq 0$$

and, hence, (E2) is a non-negative quantity:

$$W_x(t, \nu) \geq 0 \ .$$

This means that, if separable WVDs exist, they are necessarily everywhere non-negative [21]. The non-negativity condition being imposed, we know from Hudson's theorem [22] that the only signals which are admissible are exponentials of quadratic forms in t (with possibly complex-valued coefficients) such that

$$x(t) = e^{-(\alpha t^2+\beta t+\gamma)} \quad , \mathrm{Re}\{\alpha\} > 0 \ .$$

However, since separability is imposed too, no coupling between time and frequency is allowed, which restricts the class of solutions to

$$x(t) = (2\alpha/\pi)^{1/4}\, e^{-\alpha t^2}\, e^{-i2\pi\zeta t}\, e^{i\psi} \quad \Rightarrow \quad W_x(t,\nu) = 2\, e^{-2\alpha t^2}\ e^{-(2\pi^2/\alpha)(\nu-\zeta)^2} \ ,$$

where α and ζ are real-valued, ψ is a pure phase factor and the normalization has been chosen for ensuring energy conservation. Therefore, a suitable choice of separable smoothing functions which allows a continuous passage from Wigner-Ville to spectrograms or spectrograms is of the form of a (normalized) product of Gaussians, *i.e.*

$$\Pi_S(t,\nu) = \frac{\sqrt{\alpha\beta}}{\pi} e^{-\alpha t^2}\, e^{-\beta(\nu-\nu_0)^2} \ .$$

This completes the proof. ■

ACKNOWLEDGEMENTS

The authors are indebted to Th. Doligez and B. Vidalie who developed the software used for producing the pictures in Figure 2. Useful discussions, at an early stage of this work, with I. Daubechies and Th. Paul are also greatfully acknowledged.

REFERENCES

[1] J.M. Combes, A. Grossmann and Ph. Tchamitchian (*eds.*), Wavelets, Time-Frequency Methods and Phase Space, Springer-Verlag, Berlin, 1989.

[2] P. Goupillaud, A. Grossmann and J. Morlet, «Cycle-octave and related transforms in seismic signal analysis», *Geoexploration*, vol. **23**, pp. 85-102, 1984.

[3] A. Grossmann and J. Morlet, «Decomposition of Hardy functions into square integrable wavelets of constant shape», *SIAM J. Math. Anal.*, vol. **15**, n° 4, pp. 723-736, 1984.

[4] Y. Meyer, «Orthonormal wavelets», in [1], pp. 21-37.

[5] I. Daubechies, «Orthonormal bases of wavelets with finite support - Connection with discrete filters», in [1], pp. 38-66. *See also* I. Daubechies, «The wavelet transform, time-frequency localization and signal analysis», Preprint AT&T Bell Labs, 1987.

[6] S.G. Mallat, «A theory for multiresolution signal decomposition: the wavelet representation», *IEEE Trans. Pattern Anal. Machine Intell.*, vol. **PAMI-11**, n° 7, pp. 674-693, 1989.

[7] A. Grossmann, R. Kronland-Martinet and J. Morlet, «Reading and understanding continuous wavelet transforms», in [1], pp. 2-20.

[8] J. Bertrand and P. Bertrand, «Time-frequency representations of broad-band signals», in [1], pp. 164-171.

[9] A. Grossmann, *private communication.*

[10] A. Unterberger, «The calculus of pseudo-differential operators of Fuchs type», *Comm. Part. Diff. Eq.*, vol. **9**, n° 12, pp. 1179-1236, 1984.

[11] L. Cohen, «Time-frequency distribution - A review», *Proc. IEEE*, vol. **77**, n° 7, pp. 941-981, 1989.

[12] J.B. Allen and L.R. Rabiner, «A unified approach to short-time Fourier analysis and synthesis», *Proc. IEEE*, vol. **65**, n° 11, pp. 1558-1564, 1977.

[13] P. Flandrin, «Time-frequency and time-scale», *IEEE 4th ASSP Workshop on Spectrum Estimation and Modeling*, Minneapolis (MN), pp. 77-80, 1988.

[14] A.J.E.M. Janssen, «On the locus and spread of pseudo-density functions in the time-frequency plane», *Philips J. Res.*, vol. **37**, n° 3, pp. 79-110, 1982.

[15] P. Flandrin and O. Rioul, «Affine smoothing of the Wigner-Ville distribution», *IEEE Int. Conf. on Acoust., Speech and Signal Proc. ICASSP-90*, Albuquerque (NM), 1990.

[16] T. Posch, «Wavelet transform and time-frequency distributions», *private communication*, 1989.

[17] H.I. Choï and W.J. Williams, «Improved time-frequency representation of multicomponent signals using exponential kernels», *IEEE trans. on Acoust., Speech and Signal Proc.*, vol. **ASSP-37**, n° 6, pp. 862-871, 1989.

[18] O. Rioul, «Wigner-Ville representations of signals adapted to shifts and dilatations», Tech. Memo. AT&T Bell Labs, 1988.

[19] P. Flandrin, « Some features of time-frequency representations of multicomponent signals », *IEEE Int. Conf. on Acoust., Speech and Signal Proc. ICASSP-84*, San Diego (CA), pp. 41B.4.1-41B.4.4, 1984.

[20] J.G. Krüger and A. Poffyn, «Quantum mechanics in phase space, I. Unicity of the Wigner distribution function», *Physica*, vol. **85A**, pp. 84-100, 1976.

[21] C. Nahum, «Etude de la transformation de Wigner», Mémoire ENST-SYC, 1984.

[22] R.L. Hudson, «When is the Wigner quasi-probability density non-negative ?», *Rep. Math. Phys.*, vol. **6**, n° 2, pp. 249-252, 1974.

FIGURE CAPTIONS

Figure 1

Compared time-frequency resolutions of spectrograms and scalograms

(a) Spectrogram with medium resolution in time and frequency
(b) Spectrogram with low (resp. high) resolution in time (resp. frequency)
(c) Spectrogram with high (resp. low) resolution in time (resp. frequency)
(d) Scalogram with frequency-dependent resolution

Figure 2

From spectrograms to scalograms via Wigner-Ville (time: $\rightarrow$, frequency: $\uparrow$, $\mu = 2\pi/\sqrt{\alpha\beta}$)

(a) Spectrogram ($\mu = 1$)
(b) Smoothed pseudo-Wigner-Ville ($\mu = 0.6$)
(c) Smoothed pseudo-Wigner-Ville ($\mu = 0.25$)
(d) Wigner-Ville ($\mu = 0$)
(e) Affine smoothed Wigner-Ville ($\mu = 0.25$)
(f) Affine smoothed Wigner-Ville ($\mu = 0.6$)
(g) Scalogram ($\mu = 1$)

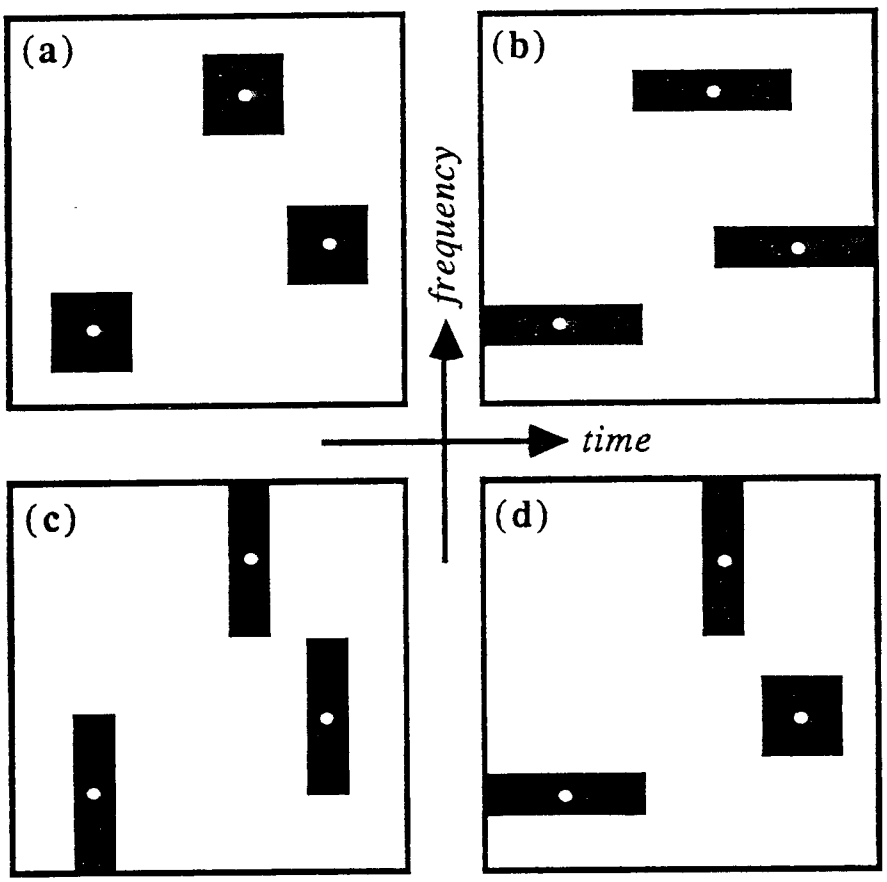


Figure 1

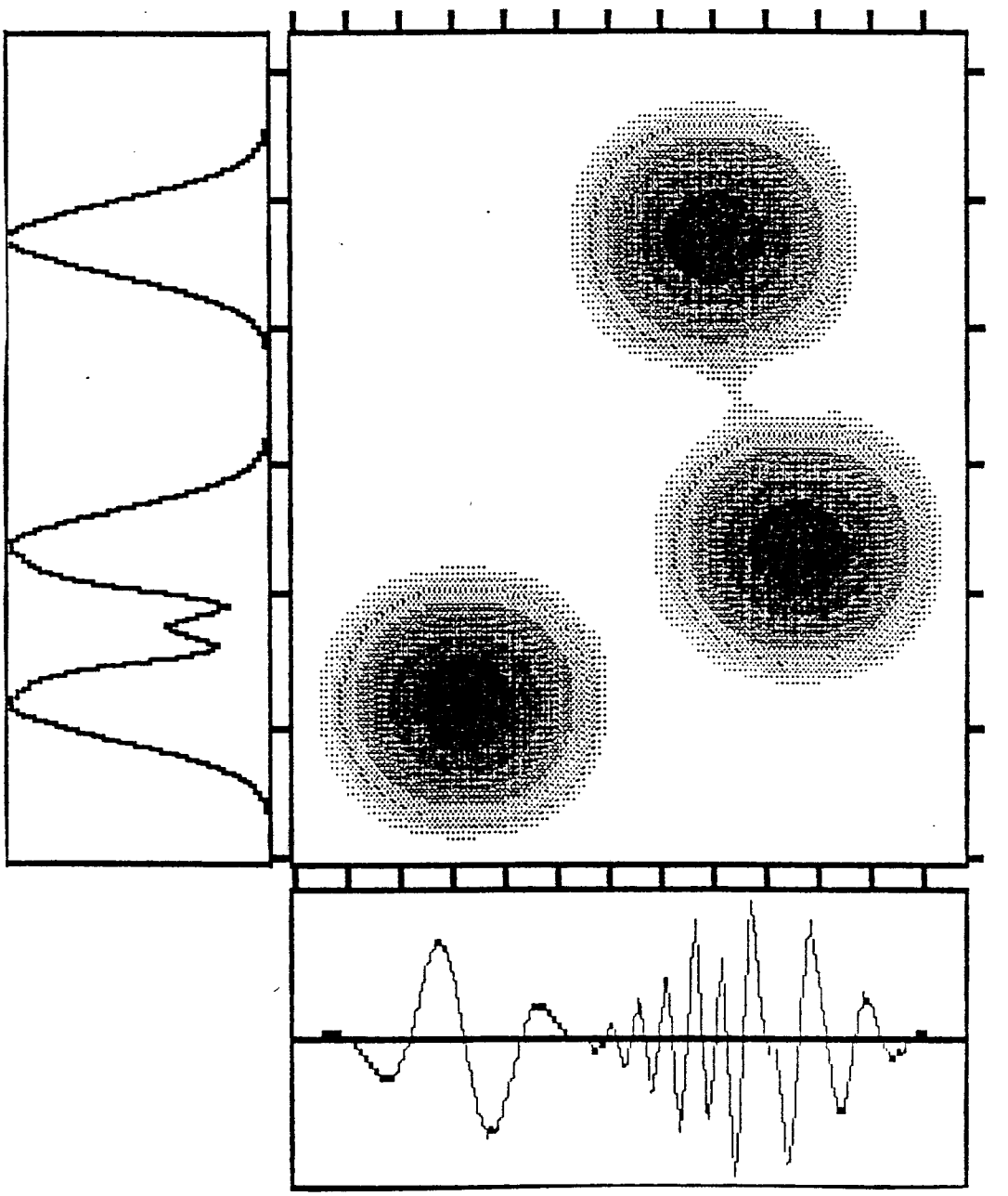


Figure 2-a

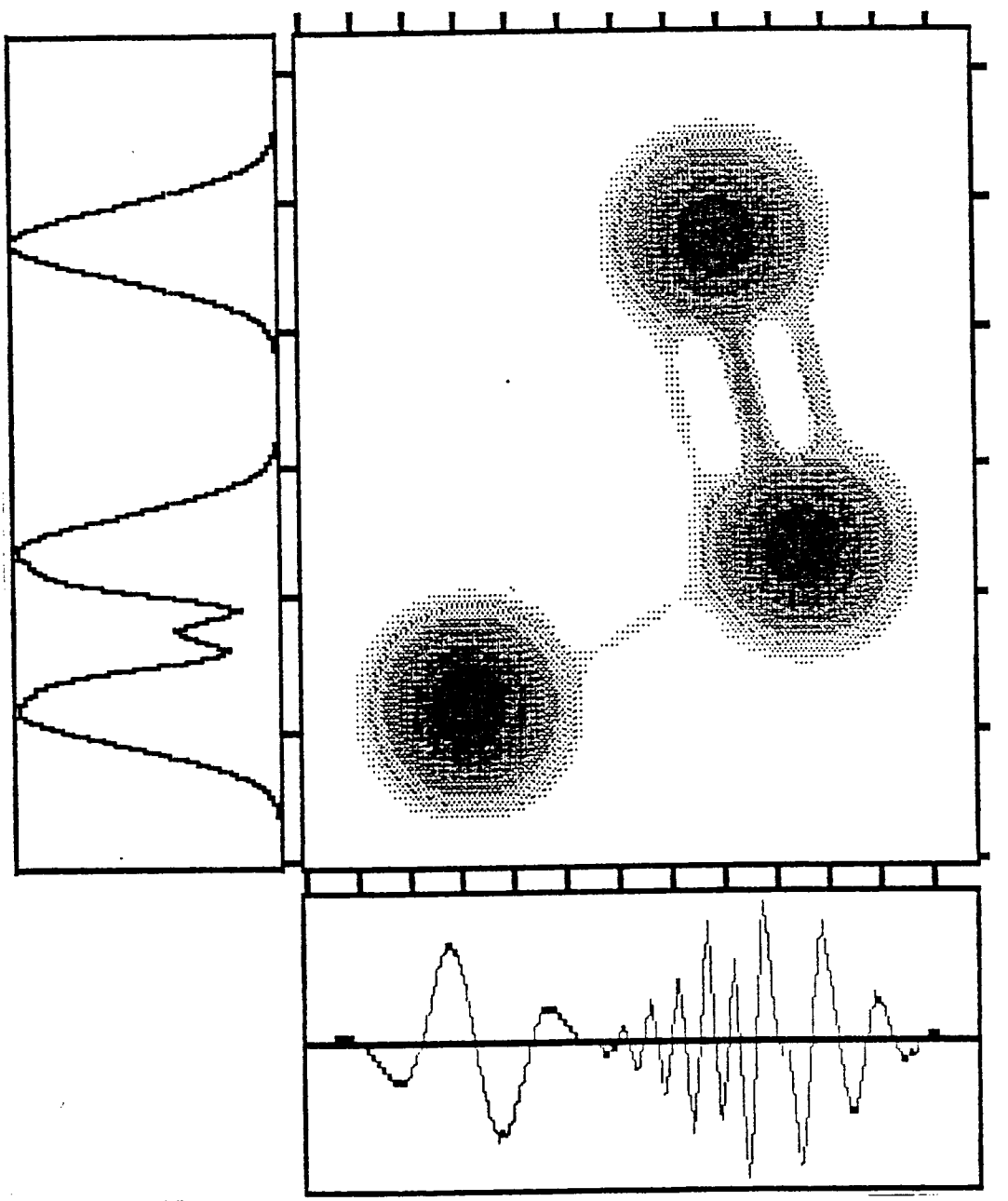


Figure 2-b

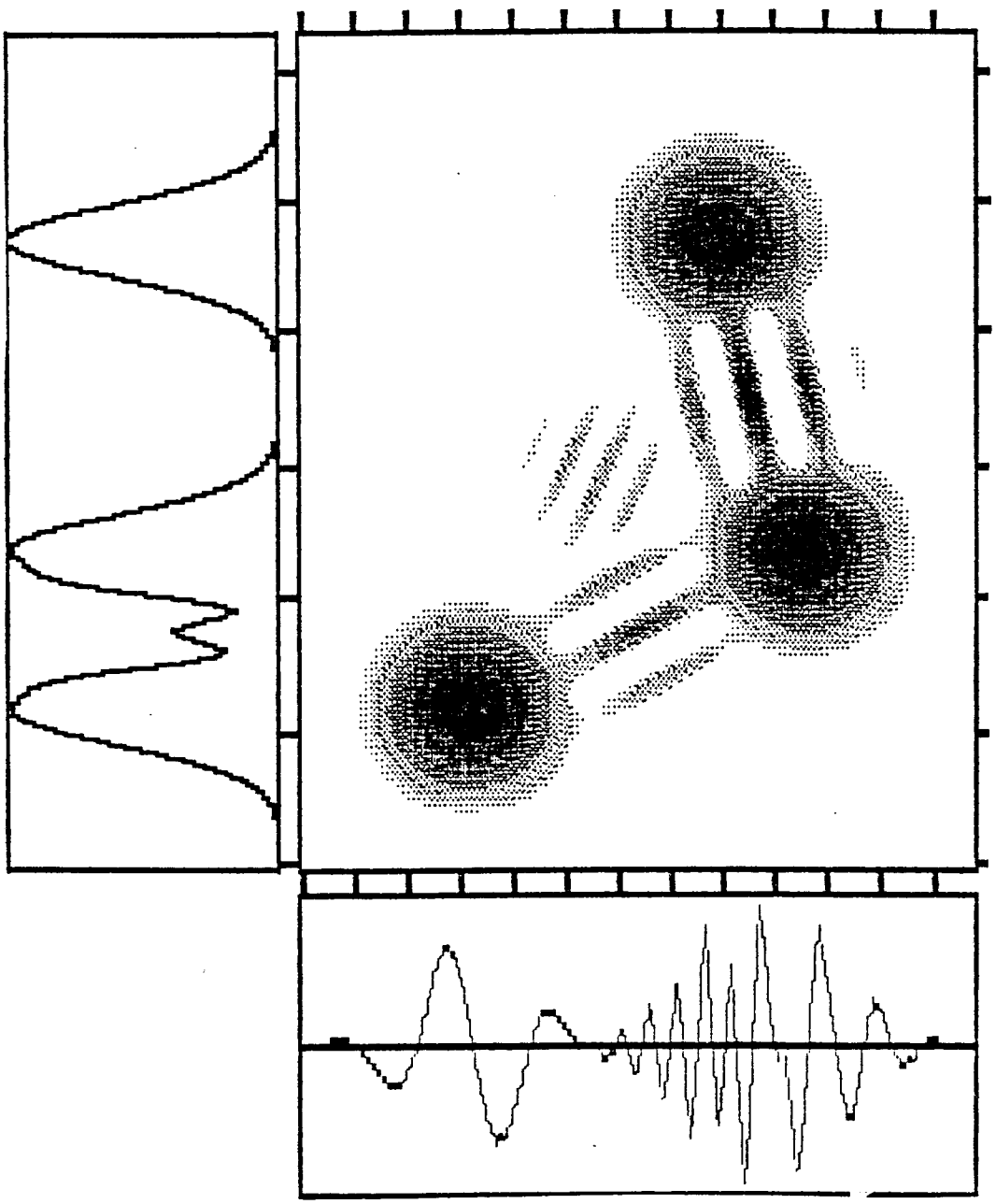


Figure 2-c

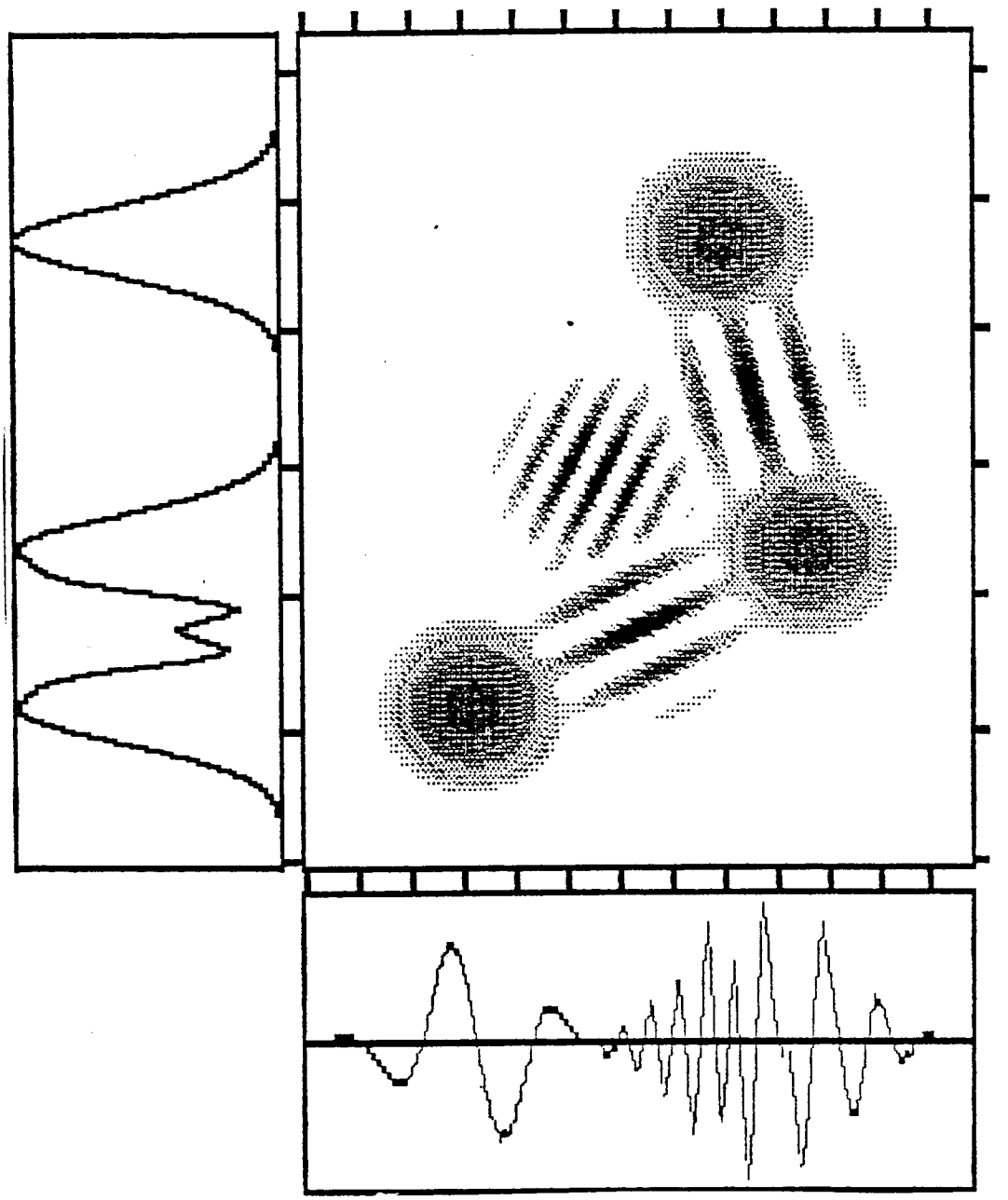


Figure 2-d

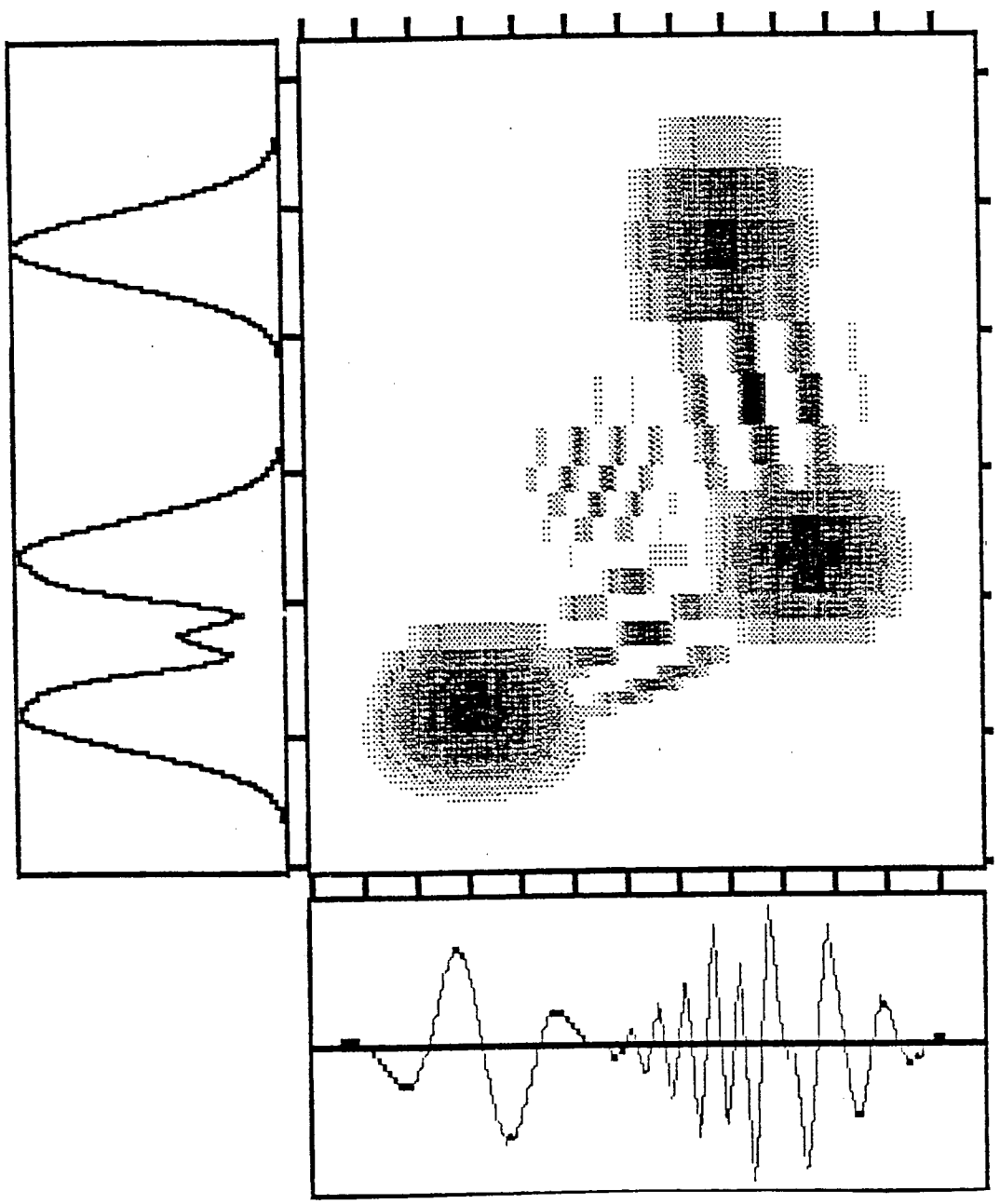


Figure 2-e

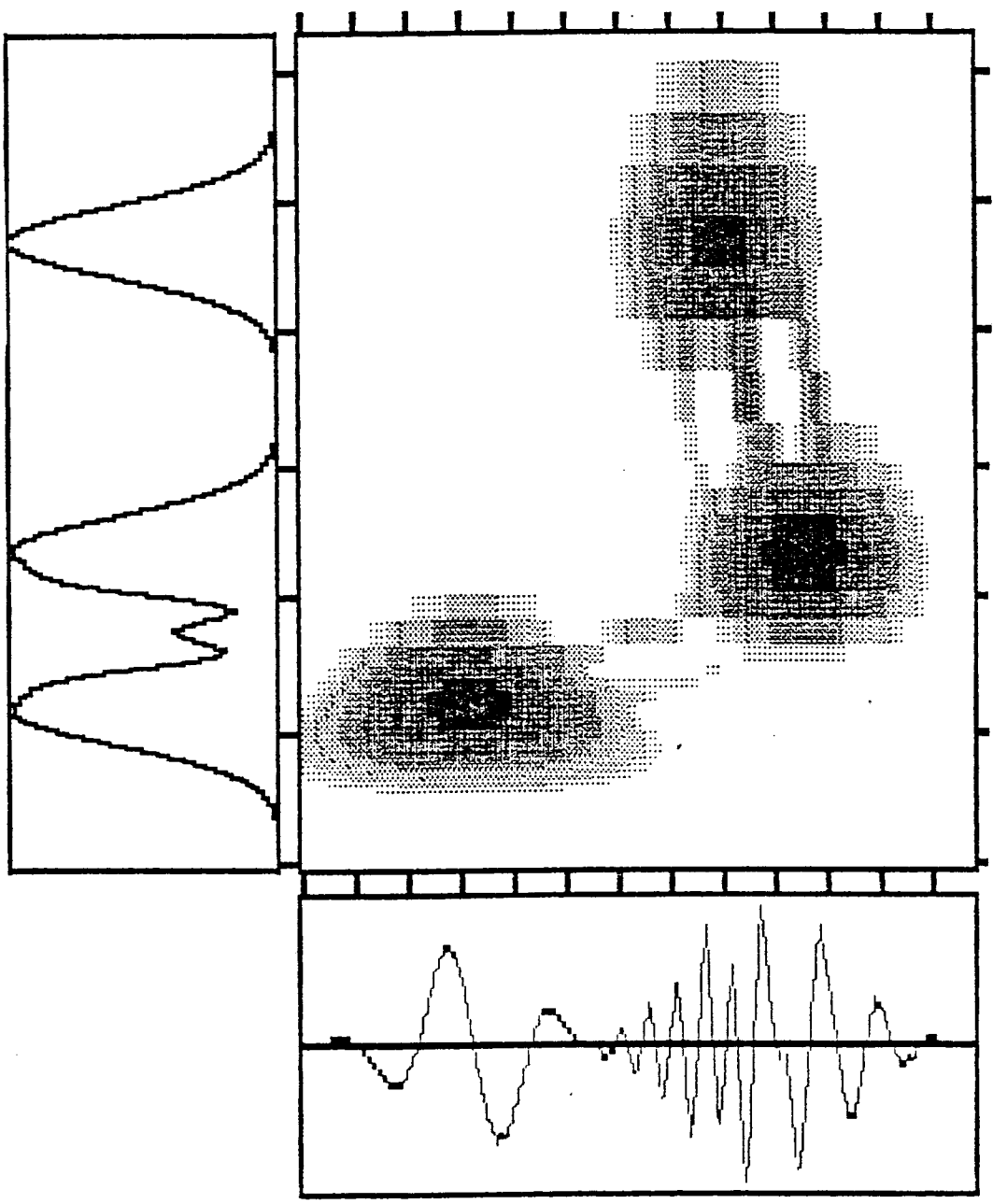


Figure 2-f

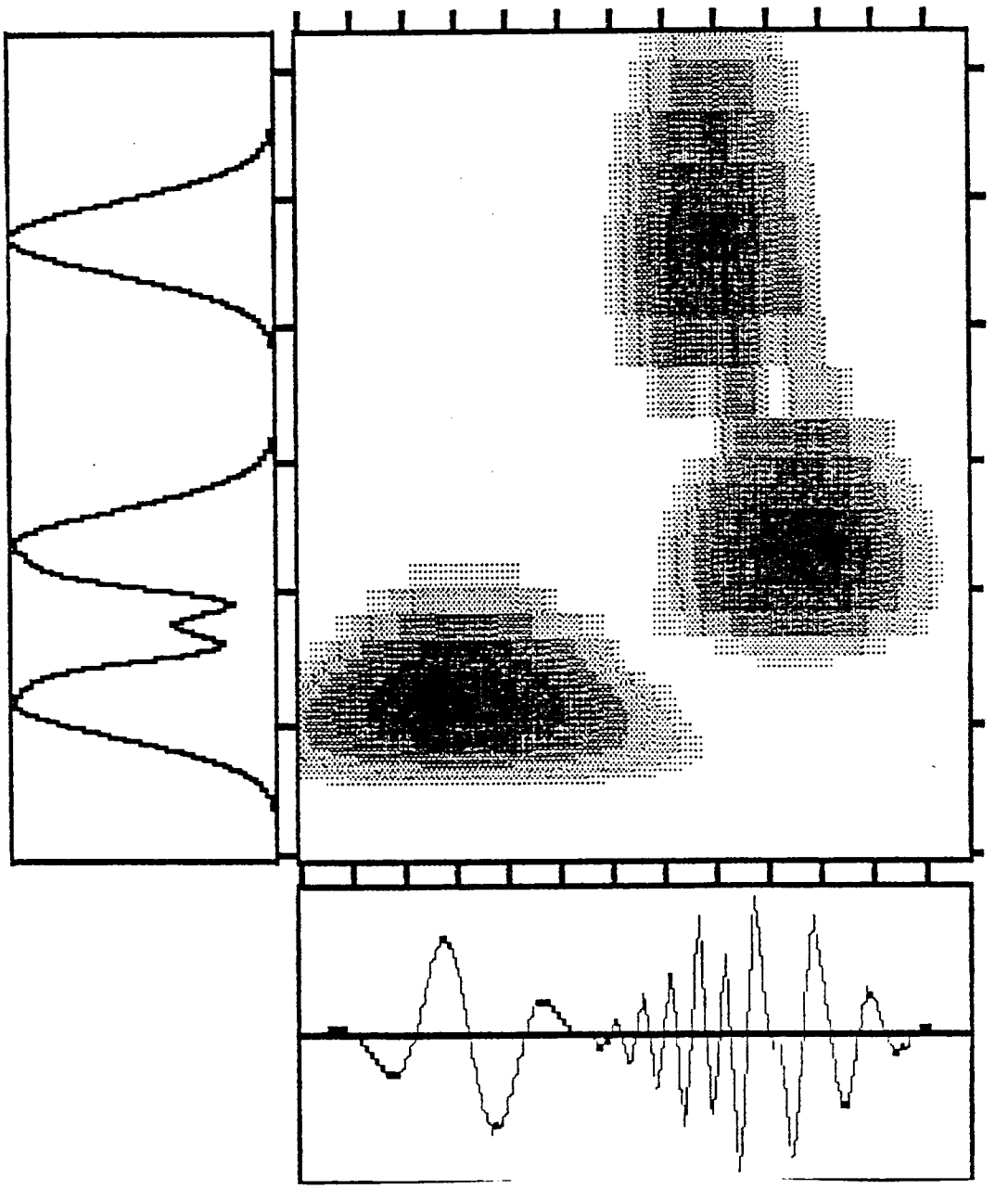


Figure 2-g

ARTICLE 7

A Fast Constant Modulus Adaptive Algorithm
par J. BENESTY et P. DUHAMEL

soumis en Mai 1990 à IEE. Proc. F. Special Issue on Adaptive Filtering.

A Fast Constant Modulus Adaptive Algorithm

J. BENESTY, P. DUHAMEL

CNET/PAB/RPE
38-40, rue du Général Leclerc
92131 ISSY-LES-MOULINEAUX
FRANCE

ABSTRACT

In this paper, an exact block formulation of the Constant Modulus Algorithm (CMA) is presented, on which a reduction of arithmetic complexity is achieved. Due to the equivalence between the original CMA formulation and ours, the convergence properties of the CMA are maintained, which is not the case in the Treichler et *al* . implementation in frequency domain of this algorithm. Furthermore, our approach allows the use of very small block lengths (e.g., N = 2), the reduction of the arithmetic complexity increasing with the block size.

I. INTRODUCTION

The Constant Modulus Adaptive Algorithm (CMA) is a special case of a more general algorithm that was first proposed by D.N. Godard [1] as a method for blind equalization for data modems. Another similar algorithm may be obtained from [10]. The CMA was extensively studied by Treichler et *al* . [2,3] for a communication application. Indeed, in many modulation schemes, such as frequency modulation (FM) and phase modulation (PM), the signal to be transmitted possesses the constant envelope property. The received signal, however, has lost this property due to multipath and interference effects. The CMA restores the constant envelope property of the signal and increases the SNR. This algorithm thus employs just the a priori knowledge about the envelope of the transmitted signal and has the nice characteristic that no reference signal is required.

Nevertheless, the CMA has some shortcomings. First, it involves the minimization of a nonconvex cost function [1]. This non-convexity implies the existence of local minima, and a satisfactory convergence of the algorithm does not imply a true minimization of the cost function. Second, the algorithm may capture a constant modulus interferer rather than the constant modulus signal of interest [3]. These two problems can be overcome by a simple filter initialization [1,3] and will not be considered here. Another drawback is the large number of arithmetic operations required for this algorithm. In order to reduce this load, Treichler et *al* . [4] proposed to compute the nonlinear error in the time - domain while updating weigths and filtering in the frequency-domain. Unfortunately, this algorithm is only an approximation of the initial one, and has been observed to have very slow convergence [4].

The main result of this paper is that it is possible to both reduce the arithmetic complexity of the CMA -by working in blocks that may be very small, if required- and maintain convergence properties : from the mathematical point of view, the algorithm thus obtained is strictly equivalent to the CMA.

Section II briefly recalls the initial version of the CMA, and provides an evaluation of the arithmetic complexity in two cases of implementation.

Section III provides the basis of our approach : merging the computations of two successive CMA outputs permits to reduce the required number of operations per output point.

This is generalized in section IV in which we establish an exact block formulation of the CMA on which a reduction of the arithmetic complexity is feasible by using the "Fast FIR" technique [5,9]. Two special cases are studied with more details : The first one is the recursive application of the fast FIR of length 2, which has the advantage of allowing an improvement of the arithmetic complexity even for very small block lengths. The second one, which uses FFT as an intermediate step, is more efficient for larger blocks.

Note that the blocksize never depends on the filter's length, and that the usual constraint that the FFT length should be at least twice the filter's length does not hold here. This fact has a lot of advantages when thinking of memory requirements or overall system delay.

II. THE INITIAL CMA

II.1. Derivation of the algorithm

The Constant Modulus Algorithm's organization is depicted in Fig.1, where y(n) is the output of a complex FIR filter :

(1)

$$y(n) = X_n^t H = H^t X_n = \sum_{i=0}^{L-1} x(n-i)h_i$$

L being the length of the filter, X_n the vector of the past L complex data at time n , and H the vector of complex weights :

$$X_n = [x(n)\ x(n-1) \ldots\ldots\ x(n-L+1)]^t$$

$$H = [h_0\ h_1 \ldots\ldots\ h_{L-1}]^t$$

The purpose of the adaptation process is to find a weight vector H that minimizes fluctuations in the complex envelope of the output y(n). Hence, a natural criterion J measures the distance between the modulus of y(n) and the constant modulus of the transmitted signal :

(2)

$$J = \frac{1}{4} E\left\{\left[|y(n)|^2 - 1\right]^2\right\}$$

where E denotes statistical expectation and where the modulus of the signal is assumed to be

equal to 1.

A possible algorithm for the coefficients updating is as follows :

(3)

$$H_{n+1} = H_n - \mu \nabla J(n)$$

where μ is a positive step size and ∇ the gradient operator :

(4)

$$\nabla J(n) = \frac{\partial J(n)}{\partial H_n}$$

$$= E\left\{\left[|y(n)|^2 - 1\right] y(n) X_n^*\right\}$$

where * denotes the complex conjugate.

Of course, since eq.(4) involves a mathematical expectation, it cannot be used as it is. It has been proposed in [1,2] to replace it by an instantaneous gradient estimate, as given in (5) (note that similar approach led to the LMS algorithm) :

(5)

$$\hat{\nabla} J(n) = \left[|y(n)|^2 - 1\right] y(n) X_n^*$$

The CMA is thus described by the following set of equations :

(6)

a) $y(n) = X_n^t H_n$

b) $\alpha(n) = \mu\left[|y(n)|^2 - 1\right] y(n)$

c) $H_{n+1} = H_n - \alpha(n) X_n^*$

II. 2. Arithmetic complexity of the CMA

II.2.1. Initial version (CMA1)

Assuming the usual 4 mult - 2 add complex multiplication scheme is used, the arithmetic complexity of the initial CMA, as described by eq.(6), is as follows : Eq.(6a) requires 4L real multiplications, and (4L - 2) real additions. The computation of (6b) requires 5 real multiplications and 2 real additions, while eq.(6c) requires 4L real

multiplications and 4L real additions. This results in the following total number of real operations per output point :

(7)

8L+5 multiplications

(8)

8L additions

Note that since μ has not been chosen as a negative power of 2, it appears in this count.

II.2.2. "Fast" complex multiply-based version

It is well known that a complex mutliplication can be computed as follows :

(9)

a) $$(x_r + jx_i)(h_r + jh_i) = \left[(x_r + x_i)h_r - x_i(h_r + h_i) \right] + j\left[(x_r + x_i)h_r - x_r(h_r - h_i) \right]$$

b) $$= \left[(h_r + h_i)x_r - h_i(x_r + x_i) \right] + j\left[(h_r + h_i)x_r - h_r(x_r - x_i) \right]$$

In the case of fixed coefficient FIR filtering, eq. (9a) is preferred, because $h_r \pm h_i$ can be precomputed, so that the overall computational load is 3 mults and 3 adds, which results in an exchange of one multiplication for one addition. When h_r and h_i are not fixed, the apparent cost is (3 mults, 5 adds). However, it is shown in the following that the use of (9b) in the CMA equations (6) allows a reduction in the total number of operations compared to the initial algorithm. Using (9b), (6) is rewritten as follows :

(10)

a) $$y(n) = \left[(H_n^r + H_n^i)^t X_n^r - (X_n^r + X_n^i)^t H_n^i \right] + j\left[(H_n^r + H_n^i)^t X_n^r - (X_n^r - X_n^i)^t H_n^r \right]$$

b) $$\alpha(n) = \mu\left[|y(n)|^2 - 1 \right] y(n)$$

c) $$H_{n+1}^r = H_n^r - \left[(\alpha_r(n) - \alpha_i(n))X_n^r + \alpha_i(n)(X_n^r + X_n^i) \right]$$

$$H_{n+1}^i = H_n^i - \left[(\alpha_r(n) - \alpha_i(n))X_n^r - \alpha_r(n)(X_n^r - X_n^i) \right]$$

A straightforward operation count would be as follows : 3L mults and 6L - 1 adds in (10a), 5 mults and 2 adds in (10b) and 3L mults and 4 L + 1 adds in (10c).

Nevertheless, remembering that $X^r_n \pm X^i_n$ has a single additional term compared to $X^r_{n-1} \pm X^i_{n-1}$ and that identical terms are stored in the filtering process, it is easily seen (Fig.2) that 2(L - 1) additions can be saved.

The overall CMA process, based on a complex FIR scheme as depicted in Fig.2 hence requires :

(11)

6L+5 mults

(12)

8L+4 adds

which reduces the total number of operations by about 2L. In other words, one fourth of the number of multiplications has been saved, at the cost of 25 % additional memory locations in some implementations. This second algorithm will be referred to as CMA2.

With these two versions of the CMA as starting points, we shall derive in the remaining of this paper an exact block formulation of this algorithm, which allows a reduction of the arithmetic complexity. The tools we use are the same ones as in the fixed coefficients case [9], and this derivation follows the same lines as for the LMS case [6,7], on which similar work was already performed.

III. AN EXAMPLE OF CMA WITH REDUCED NUMBER OF OPERATIONS

Let us first consider the required computations in the CMA at two successive time samples n - 1 and n. By appropriately re-arranging the corresponding equations, we shall obtain an exact equivalent of eq.(6) requiring a lower number of operations per output point.

Consider eq.(6), written at time n - 1 :

(13)

a) $y(n-1) = X^t_{n-1} H_{n-1}$

b) $\alpha(n-1) = \mu\left[\,|y(n-1)|^2 - 1\right] y(n-1)$

c) $H_n = H_{n-1} - \alpha(n-1) X^*_{n-1}$

Substituting (13c) into (6a) results in :

(14)

$$y(n) = X_n^t H_{n-1} - \alpha(n-1) X_n^t X_{n-1}^*$$
$$= X_n^t H_{n-1} - \alpha(n-1)\, s(n)$$

where :

$$s(n) = X_n^t X_{n-1}^*$$

Equations (14), (13a) can be combined in matrix form to give :

(15)

$$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} X_{n-1}^t \\ X_n^t \end{bmatrix} H_{n-1} - \begin{bmatrix} 0 & 0 \\ s(n) & 0 \end{bmatrix} \begin{bmatrix} \alpha(n-1) \\ \alpha(n) \end{bmatrix}$$

The first term of this equation appears to be the computation of two successive outputs of a fixed coefficient filter. Thus, we can apply on the fixed part of eq.(15) the same techniques as explained in [9] :

(16)

$$\begin{bmatrix} X_{n-1}^t \\ X_n^t \end{bmatrix} H_{n-1} = \begin{bmatrix} x(n-1) & x(n-2).....x(n-L) \\ x(n) & x(n-1).....x(n-L+1) \end{bmatrix} \begin{bmatrix} h_0(n-1) \\ h_1(n-1) \\ \vdots \\ \vdots \\ h_{L-1}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} x(n-1) & x(n-3)...x(n-L+1) & x(n-2) & x(n-4)...x(n-L) \\ x(n) & x(n-2)...x(n-L+2) & x(n-1) & x(n-3)...x(n-L+1) \end{bmatrix} \begin{bmatrix} h_0(n-1) \\ h_2(n-1) \\ \vdots \\ h_{L-2}(n-1) \\ \\ h_1(n-1) \\ h_3(n-1) \\ \vdots \\ h_{L-1}(n-1) \end{bmatrix}$$

where the even and odd terms of the involved vectors have been grouped. Furthermore, in order to obtain a more compact notation, assume L is even, and define :

$$A_0 = [x(n)\ x(n-2)\ \ldots\ldots\ x(n-L+2)]^t$$

$$A_1 = [x(n-1)\ x(n-3)\ \ldots\ldots\ x(n-L+1)]^t$$

$$A_2 = [x(n-2)\ x(n-4)\ \ldots\ldots\ x(n-L)]^t$$

$$H^0_{n-1} = [h_0(n-1)\ h_2(n-1)\ \ldots\ldots\ h_{L-2}(n-1)]^t$$

$$H^1_{n-1} = [h_1(n-1)\ h_3(n-1)\ \ldots\ldots\ h_{L-1}(n-1)]^t$$

Equation (16) is now rewritten as :

(17)

$$\begin{bmatrix} X^t_{n-1} \\ X^t_n \end{bmatrix} H_{n-1} = \begin{bmatrix} A^t_1 & A^t_2 \\ A^t_0 & A^t_1 \end{bmatrix} \begin{bmatrix} H^0_{n-1} \\ H^1_{n-1} \end{bmatrix}$$

The same kind of work can be performed for the updating of the filter taps. First substitute (13c) into (6c) :

(18)

$$H_{n+1} = H_{n-1} - \alpha(n) X^*_n - \alpha(n-1) X^*_{n-1}$$

Or, with the above notations :

(19)

$$\begin{bmatrix} H^0_{n+1} \\ H^1_{n+1} \end{bmatrix} = \begin{bmatrix} H^0_{n-1} \\ H^1_{n-1} \end{bmatrix} - \alpha(n) \begin{bmatrix} A^*_0 \\ A^*_1 \end{bmatrix} - \alpha(n-1) \begin{bmatrix} A^*_1 \\ A^*_2 \end{bmatrix}$$

Now, the following set of equations is exactly equivalent to the definition of the CMA, for a block of two outputs :

(20)

a) $$\begin{bmatrix} y'(n-1) \\ y'(n) \end{bmatrix} = \begin{bmatrix} A^t_1 & A^t_2 \\ A^t_0 & A^t_1 \end{bmatrix} \begin{bmatrix} H^0_{n-1} \\ H^1_{n-1} \end{bmatrix}$$

b) $$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} y'(n-1) \\ y'(n) \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ s(n) & 0 \end{bmatrix} \begin{bmatrix} \alpha(n-1) \\ \alpha(n) \end{bmatrix}$$

c) $$\begin{bmatrix} H^0_{n+1} \\ H^1_{n+1} \end{bmatrix} = \begin{bmatrix} H^0_{n-1} \\ H^1_{n-1} \end{bmatrix} - \begin{bmatrix} A^*_1 & A^*_0 \\ A^*_2 & A^*_1 \end{bmatrix} \begin{bmatrix} \alpha(n-1) \\ \alpha(n) \end{bmatrix}$$

Note that from a computational point of view, eq.(20b) states some problem : the computation of y(n) seems to require the knowledge of $\alpha(n)$ which itself is defined in terms of y(n). Nevertheless, since the matrix involved in eq.(20b) is strictly lower triangular, this

equation can be solved as follows : y(n-1) is readily obtained, then compute $\alpha(n-1)$ by its definition (13b), then obtain y(n) by the second line of (20b), and finally compute $\alpha(n)$ from y(n). Although $\alpha(n)$ and y(n) are related in a non-linear manner, this kind of equation will always be solvable by substitution, due to the nature of the matrix involved in eq.(20b) (strictly lower triangular).

Now, reduction of arithmetic complexity can take place, by rewriting (20) as :

(21)

a) $$\begin{bmatrix} y'(n-1) \\ y'(n) \end{bmatrix} = \begin{bmatrix} A_1^t (H_{n-1}^0 + H_{n-1}^1) + (A_2 - A_1)^t H_{n-1}^0 \\ A_1^t (H_{n-1}^0 + H_{n-1}^1) - (A_1 - A_0)^t H_{n-1}^1 \end{bmatrix}$$

b) $$\begin{bmatrix} y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} y'(n-1) \\ y'(n) \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ s(n) & 0 \end{bmatrix} \begin{bmatrix} \alpha(n-1) \\ \alpha(n) \end{bmatrix}$$

c) $$\begin{bmatrix} H_{n+1}^0 \\ H_{n+1}^1 \end{bmatrix} = \begin{bmatrix} H_{n-1}^0 \\ H_{n-1}^1 \end{bmatrix} - \begin{bmatrix} A_1^* (\alpha(n-1) + \alpha(n)) - (A_1 - A_0)^* \alpha(n) \\ A_1^* (\alpha(n-1) + \alpha(n)) + (A_2 - A_1)^* \alpha(n-1) \end{bmatrix}$$

The reduction of the number of operations has mainly been obtained in the first equation of the set : (21a) involves only three different length L/2 inner products instead of 4 inner products of the same size in (20a).

We shall now more precisely explain the organization of the computation, step by step, and evaluate the arithmetic complexity :

step a) computation of y'(n - 1) and y'(n) by eq.(21a)

step b) recursive computation of s(n) :

(22)

$$s(n) = s(n-2) + [x(n)x^*(n-1) + x(n-1)x^*(n-2) - x(n-L)x^*(n-L-1) - x(n-L-1)x^*(n-L-2)]$$

step c) y(n - 1) = y'(n - 1). Compute $\alpha(n - 1)$ by (13b), then substitute in (21b) to get y (n), and finally use (6b) for obtaining $\alpha(n)$ - eq.(21b)

step d) compute the update of H - eq. (21c)

step e) incrementation of n by 2 then go to step a)

Several considerations similar to the ones in [6,7] , allow to precisely evaluate the number of complex arithmetic operations involved in (21) :

Step a) The filtering operation $A^t_0 (H^0_{n-1} + H^1_{n-1})$ is common between the two terms of (21a), which requires 3 length L/2 filters, two of which are applied to combinations of the input samples, namely :

(23)

$$A_2 - A_1 = [x(n-2) - x(n-1), \cdots\cdots, x(n-L) - x(n-L+1)]^t$$

$$A_1 - A_0 = [x(n-1) - x(n), \cdots\cdots, x(n-L+1) - x(n-L+2)]^t$$

Just like in section II, the apparent number of additions involved in (23) can be reduced by noticing that the previous set of scalar products involved in the computation of y(n-3), y(n-2) already required nearly the same combinations of the input samples, and that only two new complex additions are to be computed : (x(n-2) - x (n-1)) and (x(n-1) - x(n)).

Step b) involves two complex multiplications and two complex additions (half the complex mults were already computed previously).

Step c) involves the computation of y(n) with one complex addition and one complex multiplication, plus two equations of the type (13b), which require a total of six real mults.

Finally, step d) requires the complex product $A^*_1 (\alpha(n-1)+\alpha(n))$, which is common between the two equations (21c). Moreover, $(A_1 - A_0)$ and $(A_2 - A_1)$ were already calculated in (21a).

The above considerations allow to evaluate the reduction in the number of complex operations per output point required for implementing the CMA. Nevertheless, the precise improvement in terms of real operations depends on the way the complex multiplications are performed (see section II-2).

When the complex multiplications are performed with the usual 4 mult-2 add scheme, the total number of operations for computing two outputs is :

(24)

12L+22 real multiplications

(25)

14L+22 real additions

We denote this algorithm by FCMA1, for a block length N = 2.

When the "fast" complex multiply scheme is used for computing the length L/2 filters, as explained in section II.2.2, the corresponding algorithm is called FCMA2, and its computational load for a blocklength N = 2 is :

(26)

9L+22 real multiplications

(27)

14L+36 real additions

Table 1 gives the number of operations per output point for all the algorithms explained up to now : CMA1, CMA2, FCMA1 and FCMA2.

It can be observed that each "fast" algorithm reduces by about 25 % the number of multiplications compared to their initial counterpart, while slightly reducing the number of additions. The total number of real operations is seen to be 20 % less than the one required by a straightforward implementation of the CMA.

Note that this reduction is obtained only by a re-arrangement of the initial equations, and that there is an exact equivalence, mathematically speaking, between the initial algorithm and our block version of it. Hence, all these algorithms have the same convergence rate.

This method has been explained in a rather specific manner, by merging the computations of two successive CMA outputs. We show in the next section that this approach is much more general : grouping the computations of more outputs results in greater computational savings.

IV. GENERALISATION TO ARBITRARY N

We shall follow the same lines as in section III : First, we provide an exact block formulation of the CMA for arbitrary N, which is in the form of a fixed filtering, followed by a correction of the outputs, and an update of the coefficients that are to be used in the next block. Concerning the fixed coefficients filtering, we shall refer essentially to ref. [5,9], and only recall some results. We shall rather concentrate on the adaptive part of this algorithm.

IV.1. Exact block formulation of the CMA

Let us assume that L = NM, M a positive integer, and let us write the fixed FIR filter output equations at time n-N+1, n-N+2,....., n-1, n :

(28)

$$\begin{bmatrix} y'(n-N+1) \\ y'(n-N+2) \\ \vdots \\ \vdots \\ y'(n-1) \\ y'(n) \end{bmatrix} = \begin{bmatrix} X_{n-N+1}^t \\ X_{n-N+2}^t \\ \vdots \\ \vdots \\ X_{n-1}^t \\ X_n^t \end{bmatrix} H_{n-N+1}$$

In the same manner as with the example N = 2, we may write the exact output equations at time n - N + 1, n- N + 2, ...n-1, n, of the CMA :

(29)

$$\begin{bmatrix} y(n-N+1) \\ y(n-N+2) \\ \vdots \\ \vdots \\ y(n-1) \\ y(n) \end{bmatrix} = \begin{bmatrix} y'(n-N+1) \\ y'(n-N+2) \\ \vdots \\ \vdots \\ y'(n-1) \\ y'(n) \end{bmatrix} - S(n) \begin{bmatrix} \alpha(n-N+1) \\ \alpha(n-N+2) \\ \vdots \\ \vdots \\ \alpha(n-1) \\ \alpha(n) \end{bmatrix}$$

with

(30)

$$S(n) = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ s_1(n-N+2) & 0 & & \vdots \\ s_2(n-N+3) & s_1(n-N+3) & \cdots & \vdots \\ \vdots & \vdots & & \\ \vdots & \vdots & & \\ s_{N-1}(n) & s_{N-2}(n) & \cdots s_1(n) & 0 \end{bmatrix}$$

where

$$s_i(n) = X_n^t X_{n-i}^* \quad , \quad i = 1,2,.....,N-1$$

The remaining part of the algorithm is the coefficients updating which is expressed as follows :

(31)

$$H_{n+1} = H_{n-N+1} - \left[X^*_{n-N+1} \quad X^*_{n-N+2} \quad \ldots\ldots \quad X^*_{n-1} \quad X^*_n \right] \begin{bmatrix} \alpha(n-N+1) \\ \alpha(n-N+2) \\ \vdots \\ \vdots \\ \alpha(n-1) \\ \alpha(n) \end{bmatrix}$$

(28), (29) and (31) can be written in matrix form as follows :

(32)

a) $Y'_n = \underline{X}(n) H_{n-N+1}$

b) $Y_n = Y'_n - S(n)\boldsymbol{\alpha}_n$

c) $H_{n+1} = H_{n-N+1} - \underline{X}^{\dagger}(n)\boldsymbol{\alpha}_n$

where † denotes the transpose conjugate, Y_n' is a vector of N successive outputs of a fixed filter, Y_n represents N successive outputs of the complex CMA filter, $\underline{X}(n)$ is a matrix (NxL) of the N last input vectors and $\boldsymbol{\alpha}_n$ the vector (Nx1) formed from $\alpha(n-i)$ for i=0 to i=N-1.

The set of equations (32) with the expressions relating $\alpha(n)$ and y(n) (eq.(6b) at time n-N+1,.....,n) form an exact equivalent of (6) for a whole block of outputs y(n-N+1),.....,y(n). Eq.(28) is an FIR filtering, whose coefficients remain unchanged during the whole block of outputs, and is thus amenable to a reduction of the arithmetic complexity through the techniques explained in [9].

Eq. (29) requires more inspection, since both vectors on each side of the equation depend on the same unknowns Y_n through eq.(6b). Nevertheless, since S(n) is strictly lower triangular, (29) can be solved in a manner strictly parallel to the computation of the solution of a linear system with a lower-triangular matrix : First initialize y(n-N+1)=y'(n-N+1) then obtain α(n-N+1) by (6b) at time n-N+1, solve in y(n-N+2) using the second line of (29) from which α(n-N+1) is obtained (6b). Then, α(n-N+1) and α(n-N+2) allow the computation of y(n-N+3) by the third line of (29). Iterating the process provides both Y_n and $\boldsymbol{\alpha}_n$ in (32b). Eq. (32c) then provides the values of H to be used in the next iteration. It is seen that the filter weights are updated once per data block instead of once per data sample. Nevertheless, this updating is performed in such a manner that the weights are equal to those that could have been found in the initial CMA at the same time.

In this way, expressions (32) are an exact block formulation of the CMA with the advantage that arithmetic complexity can be saved by using the same techniques as described in [6,7].

IV.2. Block Toeplitz formulation

A formulation of (28-31) using subsampled versions of the different signals involved allows the derivation of the fast algorithm for any N :

Define :

(33)

$$A_j = [\, x(n-j) \quad x(n-N-j) \quad \dots \quad x(n-Ni-j) \quad \dots \quad x(n-L+N-j) \,]^t$$

$$j = 0,1,\dots\dots,2N-2$$

$$i = 0,1,\dots\dots,(L/N)-1$$

a vector of length L/N ; and

(34)

$$H^k_{n-N+1} = [\, h_k \quad h_{k+N} \quad \dots\dots \quad h_{k+Ni} \quad \dots\dots \quad h_{k+L-N} \,]^t (n-N+1)$$

$$k = 0,1,\dots\dots,N-1$$

Then, eq.(28) becomes :

(35)

$$Y'_n = \begin{bmatrix} A^t_{N-1} & A^t_N & \dots\dots\dots\dots & A^t_{2N-3} & A^t_{2N-2} \\ A^t_{N-2} & A^t_{N-1} & \dots\dots\dots\dots & & A^t_{2N-3} \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ A^t_1 & & & & A^t_N \\ A^t_0 & A^t_1 & \dots\dots\dots\dots & & A^t_{N-1} \end{bmatrix} \begin{bmatrix} H^0_{n-N+1} \\ H^1_{n-N+1} \\ \vdots \\ \vdots \\ H^{N-2}_{n-N+1} \\ H^{N-1}_{n-N+1} \end{bmatrix}$$

and (31) gives :

(36)

$$\begin{bmatrix} H^0_{n+1} \\ H^1_{n+1} \\ \vdots \\ \vdots \\ H^{N-2}_{n+1} \\ H^{N-1}_{n+1} \end{bmatrix} = \begin{bmatrix} H^0_{n-N+1} \\ H^1_{n-N+1} \\ \vdots \\ \vdots \\ H^{N-2}_{n-N+1} \\ H^{N-1}_{n-N+1} \end{bmatrix} - \begin{bmatrix} A^*_{N-1} & A^*_{N-2} & \cdots & A^*_1 & A^*_0 \\ A^*_N & A^*_{N-1} & \cdots & A^*_1 & \\ \vdots & & & \vdots & \\ \vdots & & & \vdots & \\ A^*_{2N-3} & & & A^*_{N-2} & \\ A^*_{2N-2} & A^*_{2N-3} & \cdots & A^*_{N-1} & \end{bmatrix} \begin{bmatrix} \alpha(n-N+1) \\ \alpha(n-N+2) \\ \vdots \\ \vdots \\ \alpha(n-1) \\ \alpha(n) \end{bmatrix}$$

Expressions (35) and (36) play a key role for reducing the arithmetic complexity of the algorithm, since they can be seen as a filtering equation with elements replaced by vectors. Hence, all fast FIR filtering algorithms apply on this kind of matrix-vector products [5,9]. Eq. (29) has not been changed by this block formulation, and it seems that the most efficient way to compute the matrix S is the use of the following recursions :

A first equation (38) provides the expression of the first column of matrix S :

(38)

$$s_i(n-N+i+1) = s_i(n-N) + \sum_{j=0}^{i} x(n-N+i-j+1)x^*(n-N-j+1) - \sum_{j=0}^{i} x(n-L-N+i-j+1)x^*(n-L-N-j+1)$$

$$i = 0,1,\ldots\ldots,N-1$$

Equation (39) provides the computations to be performed along the diagonals :

(39)

$$s_i(n+1) = s_i(n) + x(n+1)x^*(n-i+1) - x(n-L+1)x^*(n-L-i+1)$$

The above considerations allow to evaluate precisely the number of arithmetic operations required for operating this block-CMA, whatever the block size N is. It should be noted that the number of operations to be performed per output point can be decomposed in two terms : the first one is due to the "fixed" coefficient filtering and to the update of H. This term decreases with N : working with larger blocks results in more efficient algorithms. The second one is due to the computation of matrix S, and this term increases with N. Therefore, for a given filter length, there exist an optimum blocksize that will also depend on the type of fast algorithm that is used. The next two sections study two special cases of interest, where both the block-size and the filter's length are powers of 2.

IV.3. FCMA based on short-length FIR algorithms

The first case of interest is the recursive application of the computation we used in section III. If the block length is a power of 2 ($N = 2^n$), a fast FIR algorithm can be obtained by applying **n** times the decomposition (21a), thus resulting in 3^n subfilters of size L/N, the inputs and outputs of which are sub-sampled by a factor N compared to the input of the system.

A precise evaluation of the number of operations required by this algorithm for computing a block of $N = 2^n$ outputs is provided in appendix A.1. It is shown that, if the 4 mult-2 add complex multiply scheme is used in the filtering part (FCMA1), the number of operations to be performed per output point for a filter of length $L = 2^n M$ is :

(37)

$8 (3/2)^n M + 6\, 2^n - 1$ multiplications

(38)

$4 (3 (3/2)^n - 1) M + 7\, 2^n + 8 (3/2)^n - 15$ additions

If the "fast" complex mutliply scheme is used (FCMA2, see section II.2.2.), the resulting number of operations per output point are :

(39)

$6 (3/2)^n M + 6\, 2^n - 1$ multiplications

(40)

$4 (3 (3/2)^n - 1) M + 7\, 2^n + 12 (3/2)^n - 13 - 2/2^n$ additions

Note that as long as $M \geq 4$ (i.e. the filter is at least 4 times as long as the block length) and whatever the blocksize may be, the FCMA requires fewer operations than the CMA. This means that a reduction of the arithmetic complexity is feasible even for such short filters as L=8.

Furthermore, if we suppose that $M = 2^n$,we see that the arithmetic complexity of FCMA varies with $0(3^n)$ instead of $0(4^n)$ for the CMA . This shows the efficiency of this approach.

The important point concerning these numbers is that the precise arithmetic complexity involves a term growing with N (updating of S) and another one diminishing with N (fast FIR). Hence, equations (37) and (39) have a minimum. The zeroes of the

derivations of these functions provide the approximate value of the optimum block length :

(41)

$\mathbf{n} \approx -0.6 + 0.7 \log_2 L$ for the FCMA1

(42)

$\mathbf{n} \approx -0.9 + 0.7 \log_2 L$ for the FCMA2

Table 2 provides a comparison of the number of operation per output point required by the various algorithms for the approximate optimum blocksize given by (41) and (42). A reduction by a factor of 2 of the total number of operations is seen to be very easily obtained for filters longer than L = 64, and a block length as small as N = 8.

IV.4. FFT-based implementation of FCMA

It is well known that the FFT can be used for a fast implementation of an FIR filter, through the use of overlap-add or overlap-save techniques. Since (35) has the form of an FIR filter equation, the FFT technique can be applied. The main differences with the classical technique [4] are that the FFT length is twice the blocklength instead of twice the filter's length, and that sufficient care has been taken in the block formulation of the algorithm in order to maintain the rate of convergence of the CMA.

A simple way of understanding this method consists in extending the size of the block-Toeplitz matrix of eq. (35) in such a way that the resulting matrix is cyclic. The resulting equation is :

(43)

$$\begin{bmatrix} Y_n' \\ Y_n'' \end{bmatrix} = \begin{bmatrix} T(n) & T'(n) \\ T'(n) & T(n) \end{bmatrix} \begin{bmatrix} H_{n-N+1} \\ O \end{bmatrix}$$

where T (n) is the block-Toeplitz matrix of eq.(35), O a null vector of size N, Y_n'' a set of outputs that do not need to be computed (overlap-save technique).

T'(n) is choosen (44) in order to give the block-cyclic property to the above matrix :

(44)

$$T'(n) = \begin{bmatrix} A_{N-1}^t & A_0^t & \cdots & A_{N-3}^t & A_{N-2}^t \\ A_{2N-2}^t & A_{N-1}^t & \cdots & & A_{N-3}^t \\ \vdots & & & & \vdots \\ \vdots & & & & \vdots \\ A_{N+1}^t & & & & A_0^t \\ A_N^t & & \cdots & & A_{N-1}^t \end{bmatrix}$$

eq.(43) involves inner products of the form $A_i^t H^k$. Let us denote by $C_i(n)$ the matrix (2Nx2N) made from the i^{th} term of the blocks of matrix of eq.(43). When developing all inner products in term of the individual components, (43) is rewritten as :

(45)

$$\begin{bmatrix} Y_n' \\ Y_n'' \end{bmatrix} = \sum_{i=0}^{(L/N)-1} C_i(n) \begin{bmatrix} \tilde{H}_{n-N+1}^i \\ O \end{bmatrix}$$

where

$$\tilde{H}_{n-N+1}^i = [\, h_{Ni} \quad h_{Ni+1} \quad \cdots\cdots \quad h_{Ni+N-2} \quad h_{Ni+N-1} \,]^t (n-N+1)$$
$$i = 0,1,2,\ldots\ldots,(L/N)-1$$

Each matrix $C_i(n)$ is cyclic, hence can be diagonalized by a Fourier matrix of size 2N :

(46)

$$\begin{bmatrix} Y_n' \\ Y_n'' \end{bmatrix} = F_{2N}^{-1} \left[\sum_{i=0}^{(L/N)-1} \left[F_{2N}\, C_i(n) F_{2N}^{-1} \right] F_{2N} \begin{bmatrix} \tilde{H}_{n-N+1}^i \\ O \end{bmatrix} \right]$$

where

$$F_{2N}\, C_i(n) F_{2N}^{-1} = D_i(n)$$

is a diagonal matrix, whose elements are the DFT of the first row of $C_i(n)$.

Furthermore, it can easily be seen that $D_i(n)=D_{i-1}(n-N)$. This implies that eq.(46) represents 2N complex filters of length L/N in the Fourier domain, subsampled by a factor 1/N. The overall organization of this scheme is provided in Fig.3. It is seen to require (L/N)+2 FFT of length 2N per block of data of size N, plus 2N complex filters of length L/N which run at a rate divided by N.

The same kind of work has to be performed for the updating of the coefficients : eq.(36) is first extended to become block-cyclic. Considering separately each i^{th} term of the vector H^k and A_i results in the following set of equations :

(47)

$$\begin{bmatrix} \tilde{H}^i_{n+1} \\ O \end{bmatrix} = \begin{bmatrix} \tilde{H}^i_{n-N+1} \\ O \end{bmatrix} - W C_i^{\dagger}(n) \begin{bmatrix} \alpha_n \\ O \end{bmatrix}$$

$$i = 0,1,2,.....,(L/N)-1$$

$$W = Diag\{1,1,.....,1,0,0,.....,0\}$$

and finally :

(48)

$$\begin{bmatrix} \tilde{H}^i_{n+1} \\ O \end{bmatrix} = \begin{bmatrix} \tilde{H}^i_{n-N+1} \\ O \end{bmatrix} - W F_{2N}^{-1} D_i^*(n) F_{2N} \begin{bmatrix} \alpha_n \\ O \end{bmatrix}$$

$$i = 0,1,2,.....,(L/N)-1$$

where $D_i^*(n)$ is the complex conjugate of the matrix $D_i(n)$.

Eq.(46),(48) , together with (29) are seen to represent the FFT-based implementation of the FCMA. A precise count of the required number of operations is provided in Appendix A.2., and compared in table 2 for a number of filter lengths of interest. It is seen that this method requires the lowest number of multiplications among all considered methods for lengths greater than 32, and a lower number of operations (adds plus mults) above L=128. For a filter of length 512, the FFT-based implementation requires a number of operations divided by 4.5 compared to the initial algorithm. Note that this performance is obtained for quite small blocklengths. The table 2 makes the distinction between two versions of FFT-based CMA, depending on the algorithm chosen for the complex filters of length L/N, as seen in section II.2.2. Note that the number of operations for both FFT-based algorithms are similar. Hence, the choice between them will rely on structural considerations.

V. SIMULATIONS

Some of these algorithms have been simulated, to verify our affirmations concerning :

a) The exact equivalence between the CMA and the FCMA,

b) the speed of our algorithm in relation to the initial one.

The complex input signal is (see Fig.1) :
$x(n)=s(n)+s_M(n)+b(n)$

where s(n) is the constant modulus transmitted signal, $s_M(n)=\beta\, s(n-\tau)$ the signal due to the effects of multipath propagation and b(n) a zero-mean white noise.
The objective is to use the CMA to provide an output y(n) which is an estimation of the transmitted signal s(n) :
$y(n)=\hat{s}(n)$

Our aim here is not to concentrate on properties of the CMA itself, but to check the equivalence between the initial and fast versions of this algorithm. Fig.4 provide the convergence curves (J(n) averaged on 64 points, normalized by the corresponding input energy) of algorithms CMA1 and FCMA1 in the case of a complex FIR filter of length L=256 and a blocksize of N=32.These curves are clearly seen to be identical. As for the speeding up of the algorithm, we observed that the FCMA1 saved 40% computation time compared with the CMA1, which is nearly the ratio of the total number of operations, while exhibiting exactly the same convergence behaviour. This gives an indication on the accuracy issue of the FCMA : The updating of coefficients s_i being performed recursively, one may wonder if this recursivity could introduce any major drawback. We have shown that in the LMS case [7], and for a fixed-point computation, this way of computing only results in a slight increase of the residual error. It is also shown that, in any realistic case, these errors cannot result in an instability of the algorithm [7]. The same demonstration holds for the CMA case, and will not repeated here.

It is interesting to note that the matrix S depends only of the input signal, and some approximations are feasible [6,7].

VI. CONCLUSION

In this paper, we provided a new algorithm wich allows a reduction of arithmetic complexity of the CMA. This reduction is possible whatever the blocksize is, and even for the smallest blocklength (N=2).

Furthermore, we showed that it was also possible to work in the frequency-domain and that the obtained algorithm is strictly equivalent to the initial CMA.

All these algorithms share the same advantages : same convergence as CMA with a lower arithmetic complexity, and small blocksize. The small blocksize allows the memory requirements to remain reasonable, and reduces the overall system delay.

The algorithms based on short-length FIR algorithms are efficient for very small blocklengths, while FFT-based algorithms are more efficient for medium size ones.

Furthermore, there is a possibility that the convergence rate can be improved using this technique. This work is under consideration and will be reported.

AKNOWLEDGMENTS

The authors would like to thank S. Mayrargue and O. Rioul from the CNET for useful discussions on the subjects of this paper.

APPENDIX A

Evaluation of the arithmetic complexity of the proposed algorithms for blocklengths $N=2^n$.

A.1. Based on short-length FIR filters

This algorithm turns the fixed coefficient filtering (28) of length L into 3^n complex filters of length L/N, that are used for computing a block of N outputs. The precise number of real operations required by these filters depends on the chosen type of implementation (see section II.2.2.). This transformation is obtained by linear combinations of subsampled input sequences (recursive application of (35))which cost a total of :
$4\,(3^n - N)$ complex additions.

The second step is the correction of Y_n' in order to obtain Y_n. This first requires the computation of the elements of the matrix S(n), by application of eq.(38) and (39). This requires :
$2N\,(N-1)$ complex multiplications,
$5N\,(N-1)/2$ complex additions.
Once S(n) is obtained, eq.(29) is solved by substitution, as explained in section IV.1., which requires a total of :
N real multiplications,
$N\,(N+1)$ complex multiplications,
N^2 complex additions.

The final step is the updating of H to be used in the next block computation. This first requires the computation of $\underline{X}^{\dagger}(n)\,\alpha_n$, which is computed in the same manner as Y_n'. By taking into account the fact that the combinations of the input samples need not to be computed again, this requires :
$2\,3^n\,L/N$ complex multiplications,
$3^{n+1}\,L/N - 2L + 3^n - N$ complex additions.

Finally, once the impulse response is obtained, the linear combinations of the coefficients H^k need to be computed. This requires :
$3^n\,L/N - L$ complex adds,

and the final computation of (36) requires L complex additions more.

When the 4-mult 2-add complex multiplication sheme is used in the FIR filtering, the resulting algorithm (FCMA1) requires a total of :

$8\ 3^n\ L/N + N\ (\ 6N - 1\ )$ real mults,

$12\ 3^n\ L/N - 4L + 8\ 3^n + N\ (\ 7N - 15\ )$ real adds.

Finally, for the so-called FCMA2, where the complex filter scheme is that of Fig.2, we obtain the following number of operations :

$6\ 3^n\ L/N + N\ (\ 6N - 1\ )$ real mults,

$12\ 3^n\ L/N - 4L + 12\ 3^n + N\ (\ 7N - 13\ ) - 2$ real adds.

A.2. FFT-based implementation

The overall organization of the algorithm is the same one as before, the differences being found in the complex filtering sheme and in the updating of the coefficients:

In fact, the FFT scheme transforms the length-L filter into 2N complex filters of length L/N, at the cost of L/N length-2N FFT's for computing the weights, one FFT for the determination of $D_0(n)$ (note that $D_i(n)$, i=1,2,...,(L/N)-1, have already been computed, since $D_i(n)=D_{i-1}(n-N)$) and one length-2N inverse FFT for recovering the outputs.

As for the updating of the weigths, the overall computation, as given in (48) requires (L/N)+1 length-2N FFT's, plus 2L complex multiplications and 3L complex additions.

Furthermore, let us assume that the FFT is computed using the split radix algorithm [11], we obtain as a result :

For the FFT-based FCMA1 :

$4L\ (\ \log_2 N + 2\ ) + 8\ L/N + 6\ N^2 + 6N \log_2 N - 13N + 12$ real mults,

$2L\ (\ 6 \log_2 N + 7\ ) + 8\ L/N + 7\ N^2 + 18N \log_2 N - 9N + 12$ real adds,

and for the FFT-based FCMA2 :

$4L\ (\ \log_2 N + 1\ ) + 8\ L/N + 6\ N^2 + 6N \log_2 N - 13N + 12$ real mults,

$2L\ (\ 6 \log_2 N + 7\ ) + 8\ L/N + 7\ N^2 + 18N \log_2 N - N + 12$ real adds.

REFERENCES:

[1] D.N. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communication systems", IEEE Trans. COM-28, N°11, Nov. 1980, pp.1867-1875.

[2] J.R. Treichler and B.G. Agee, "A new approach to multipath correction of constant modulus signals", IEEE Trans. ASSP-31, N° 2, April 1983, pp.459-471.

[3] J.R.Treichler and M.G. Larimore, "The tone capture properties of CMA-based interference suppresors", IEEE Trans. ASSP-33, N°4, Aug. 1985, pp.946-958.

[4] J.R. Treichler, S.L. Wood and M.G. Larimore, "Convergence rate limitations in certain frequency-domain adaptive filters", ICASSP-1989, pp.960-963.

[5] Z.J. Mou and P. Duhamel, "Short-length FIR filters and their use in fast FIR filtering", submitted to IEEE Trans. on ASSP.

[6] J. Benesty and P. Duhamel, "A fast exact least mean square adaptive algorithm", to appear, ICASSP-1990.

[7] J. Benesty and P. Duhamel, "A fast exact least mean adaptive algorithm", submitted to IEEE Trans. on ASSP.

[8] R.E. Blahut, Fast Algorithms for Signal Processing, Addison-Wesley, Reading, MA, 1985.

[9] Z.J. Mou, P. Duhamel, "Fast FIR filtering : algorithms and implementation", Signal Processing, Dec. 1987, pp.377-384.

[10] A. Benveniste, M. Goursat and G. Ruget, "Robust identification of a nonminimum phase system : Blind adjustment of a linear equalizer in data communications", IEEE Trans. AC-25, N°3, June 1980, pp.385-399.

[11] P. Duhamel, "Implementation of split-radix FFT algorithm for complex, real and real-symmetric data", IEEE Trans. ASSP-34, N°2, April 1986, pp.285-295.

Figure captions

Fig.1 : Overall organization of the CMA.

Fig.2 : Efficient implementation of the complex filter found in the CMA in terms of real operations.

Fig.3 : Implementation of the complex filter based on shorter FFT's.

Fig.4 : Error curve of the
a) Constant Modulus Algorithm
b) Fast Constant Modulus Algorithm.

Table captions

Table 1 : Comparison of the arithmetic complexity per output point of the CMA and the FCMA for a blocksize of N=2.

Table 2 : Comparison of the number of operations per output point required by the various algorithms.

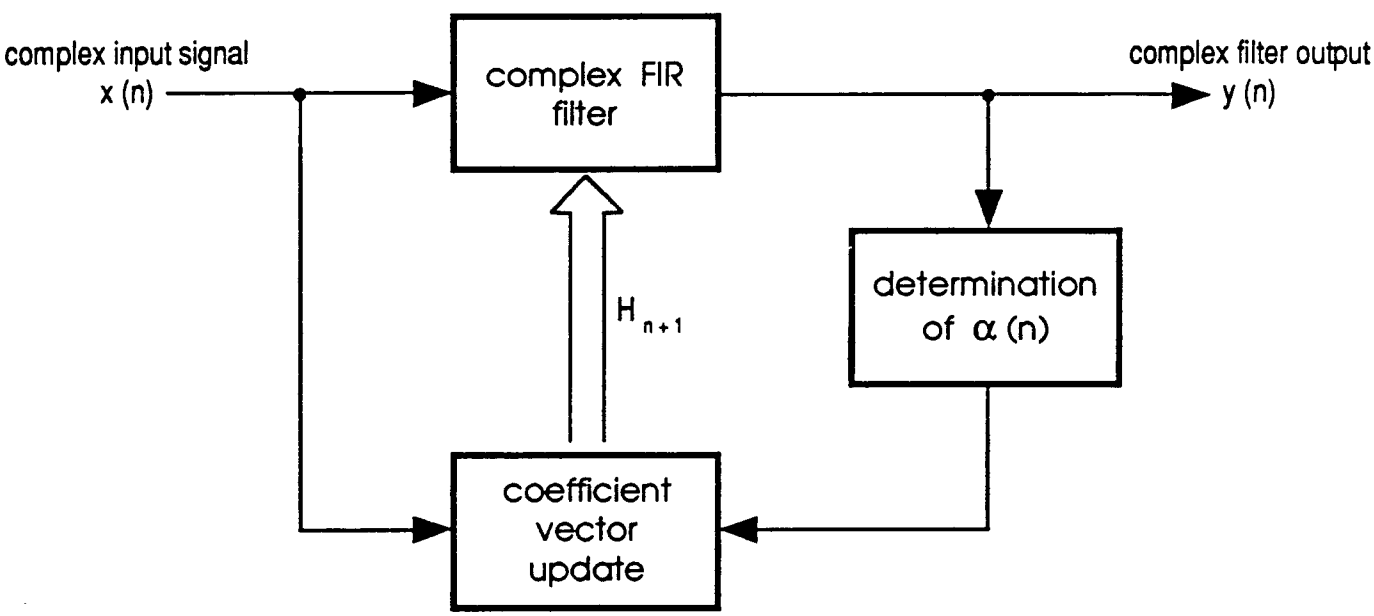


Fig. 1 : The Constant Modulus Algorithm.

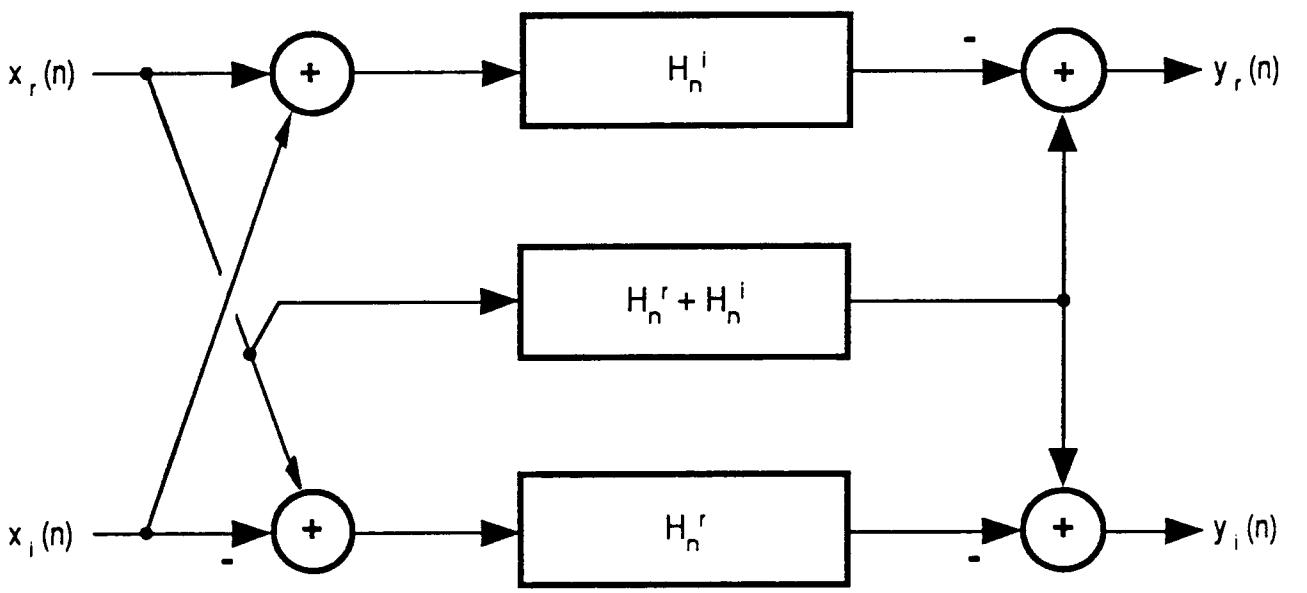


Fig. 2 : Complex FIR filtering using three real FIR filters.

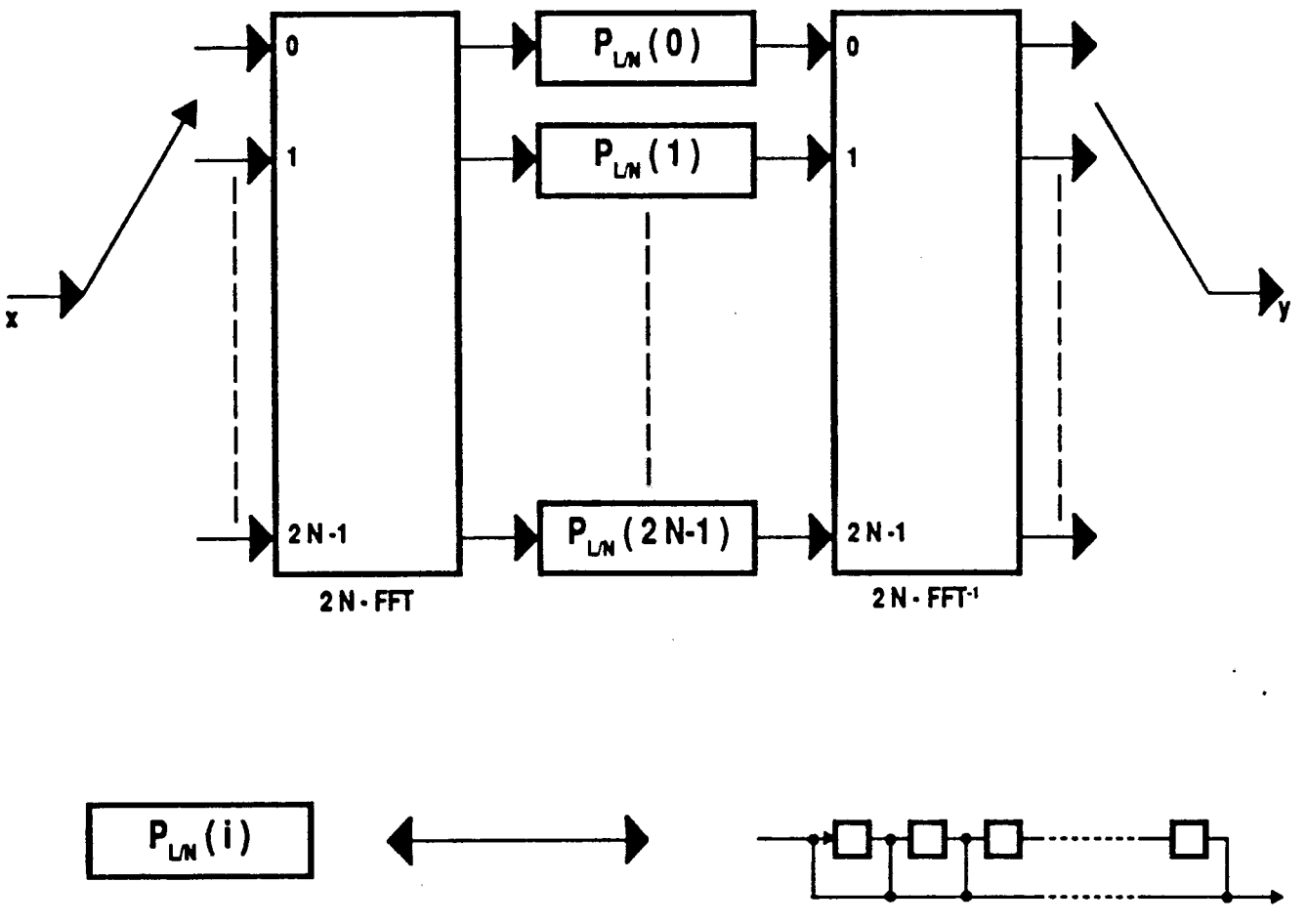


Fig. 3 : Shorter FFT - based FIR filtering scheme.

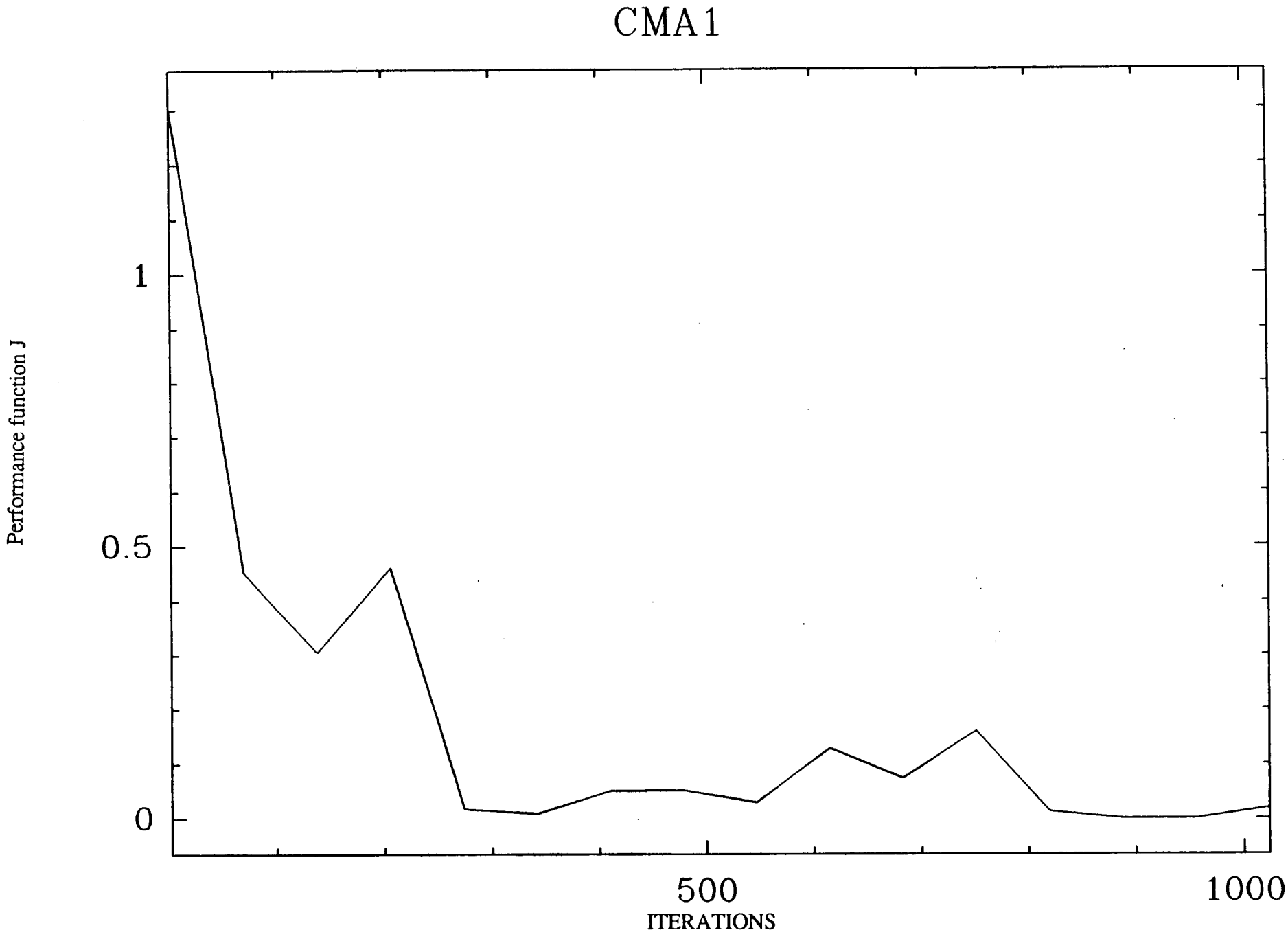


Fig.4 a) Error curve of the CMA

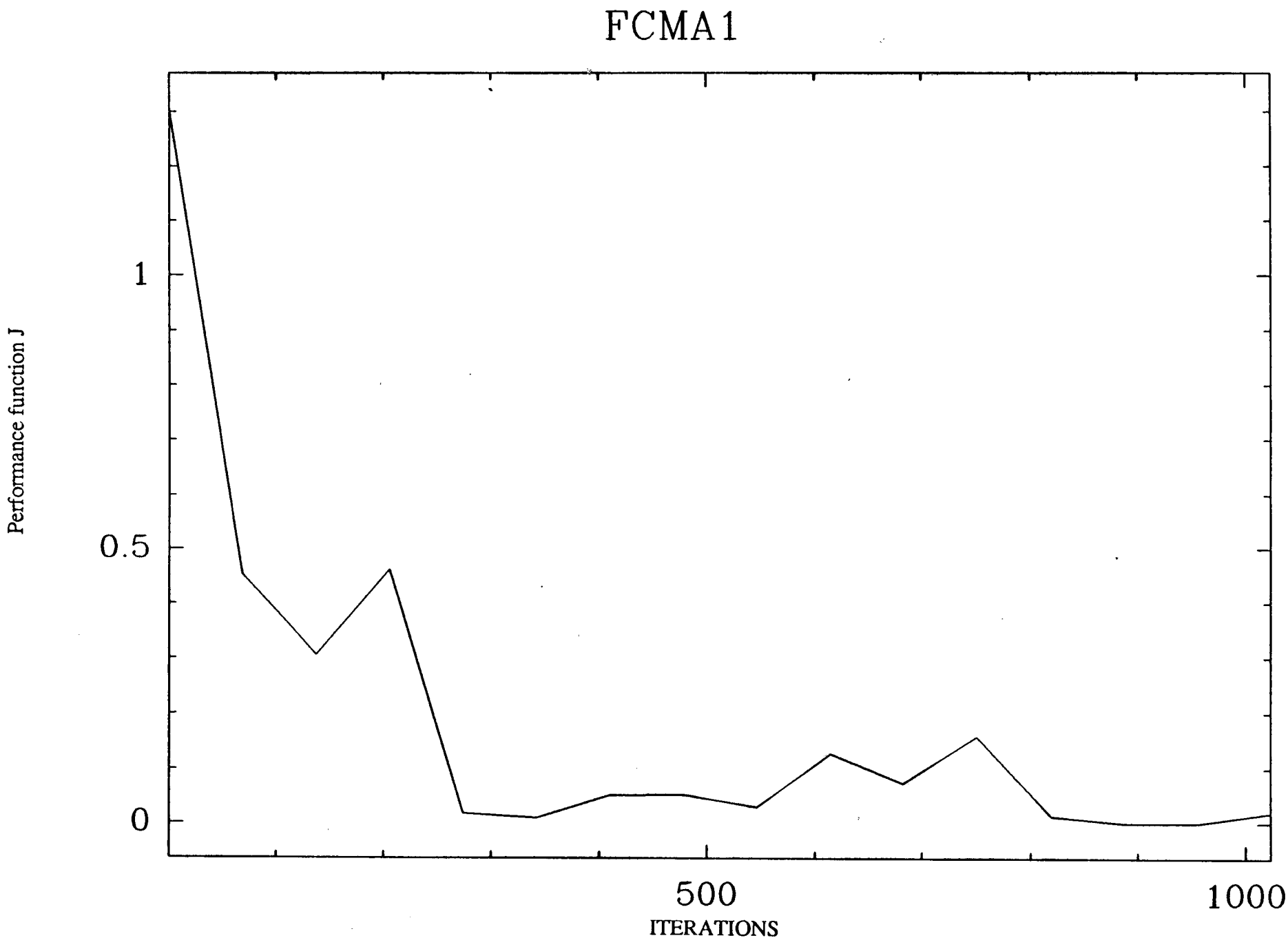


Fig.4 b) Error curve of the FCMA

Algorithm / Number of operations	CMA1	CMA2	FCMA1	FCMA2
Number of additions	8 L	8 L + 4	7 L + 11	7 L + 18
Number of multiplications	8 L + 5	6 L + 5	6 L + 11	4,5 L + 11
Total number of operations	16 L + 5	14 L + 9	13 L + 22	11,5 L + 29

Table 1 : Comparison of the number of operations per output point of the CMA and the FCMA for a blocksize of N = 2

	CMA1		CMA2		FCMA1			FCMA2			FFT - CMA1			FFT-CMA2		
filter length L	number of additions	number of multi-plications	number of additions	number of multi-plications	block length N	number of additions	number of multipli-cations	block length N	number of additions	number of multipli-cations	block length N	number of additions	number of multipli-cations	block length N	number of additions	number of multi pli-cations
32	256	261	260	197	8	214	155	8	229	128	8	275	143	8	283	127
64	512	517	516	389	8	346	263	8	361	209	16	378	206	16	386	190
128	1024	1029	1028	773	16	591	419	16	614	338	16	628	304	16	636	272
256	2048	2053	2052	1541	32	967	677	32	999	555	32	839	435	32	847	403
512	4096	4101	4100	3077	64	1586	1112	32	1696	920	64	1164	664	64	1172	632

Table 2 : Comparison of the number of operations per output point required by the various algorithms.

ARTICLE 8

On the methods for solving Yule-Walker equations
par H.M ZHANG et P. DUHAMEL

soumis en Mai 1990 à IEEE Trans. on ASSP

ON THE METHODS FOR SOLVING YULE-WALKER EQUATIONS

Hui-Min ZHANG and Pierre DUHAMEL
CNET/PAB/RPE
38-40 Rue du Général Leclerc
92131 Issy-Les-Moulineaux
FRANCE

ABSTRACT

We review the three well known fast algorithms for the solution of Yule-Walker (YW) equations: Levinson algorithm, Euclidean algorithm and Berlekamp-Massey algorithm, and show the relation between each of them and the Padé approximation problem. This connection has already been noticed for some of them, but we intend here to offer a synthetic view of these fast algorithms.

We classify the algorithms solving YW equations with reference to three criteria, namely:

-the path they follow in the Padé table

-the organization of the computation: we distinguish between one-pass and two-pass algorithms.

-the auxiliary variables used: some algorithms use the backward predictor of same degree as intermediate variable for computing the forward predictor (or vice versa), while others use two predictors of the same type but of successive degrees.

This classification shows that the set of known classical algorithms is not complete, and we propose the missing variants. With these variants of the Berlekamp-Massey and Euclid algorithms, we are able to obtain both forward and backward predictors without additional cost.

Furthermore, we give a unified representation of the two-pass algorithms, in such a way that the application of the divide and conquer strategy becomes straightforward. A general doubling algorithm which represents all the associated doubling algorithms in an exhaustive way is provided.

EDICS Category: 5.1.6 Computational Algorithms

* Permission to publish this abstract separately is granted

1. INTRODUCTION

Speeding up the solution of Yule-Walker equations is very important, since this problem appears frequently in various fields such as error-correcting codes, digital signal processing, to name only a few. For example, the key equation for decoding Bose-Chaudhuri-Hocquenghem (BCH) / Reed-Solomon (RS) / Goppa codes [4] and the equation involved in the identification of coefficients of an autoregressive model are Yule-Walker equations. The Levinson algorithm, Berlekamp-Massey algorithm, and Euclidean algorithm are the three well known fast algorithms for this purpose, each of them being linked initially with one special application, although their usefulness has been recognized for the other ones later on.

By exploiting the Toeplitz structure of the coefficient matrix, the Levinson algorithm [18] solves a Yule-Walker equation with $O(n^2)$ arithmetic operations, where n is the size of the matrix. The Berlekamp-Massey algorithm was developed by Berlekamp [4] for solving precisely the key equation for decoding BCH codes, and was interpreted by Massey as a linear feedback shift-register synthesis having the shortest length [19]. The relation between a Yule-Walker equation and a Padé approximant [6] makes also possible the Euclidean algorithm [17,20] for solving the Yule-Walker equation. This algorithm requires also $O(n^2)$ arithmetic operations.

Although proposed for solving the same equation, these algorithms are based initially on various concepts. Our main purpose is therefore to compare them and to offer a synthetic view of these fast algorithms.

First, we recall that all these algorithms can be interpreted as Padé approximation algorithms [6,7,15], which provides a better understanding of the essential of the problem. This interpretation clarifies the relation between different algorithms, gives an explanation of the diversity of algorithms, and provides possibilities for the development of new ones.

Then, we show that all these algorithms have two different versions, belonging respectively to the following classes defined in [8]:

- the one-pass algorithms, that evaluate directly the denominators of Padé approximants;
- the two-pass algorithms, that compute first the numerators or the remainders of Padé approximants via a sequence of elementary operations and then repeat the same sequence of operations to obtain the denominators.

It is shown that the one-pass algorithms are computationally more efficient than the two-pass algorithms, while the two-pass algorithms which provide more information, have the essential advantage of allowing the derivation of very powerful signal processing algorithms, namely the doubling algorithms ("superfast algorithms") and the parallel algorithms [24]. Thus, for each algorithm belonging to one class, we develop the corresponding one belonging to the other class. The Levinson algorithm being of the one-pass type, we obtain its two-pass equivalent by a simple combination of this algorithm with that of Schur [25], which is already known. For the Berlekamp-Massey and Euclidean algorithms, we develop two new corresponding algorithms: the two-pass Berlekamp-Massey algorithm and the one-pass Euclidean algorithm. We complete therefore these algorithms in both classes.

When the polynomials to be processed in the algorithms are seen as forward or backward prediction polynomials in the prediction theory, it is seen that the Levinson algorithm works on both forward and backward prediction polynomials, while the Euclidean's works only on the forward one, and the Berlekamp-Massey's works only on the backward one. We develop for each of the last two algorithms a variant which is able to provide both types of predictors while maintaining the computational complexity almost unchanged.

Throughout the paper, the arithmetic complexity is evaluated for each algorithm. We show that the algorithms in the same class require nearly the same number of arithmetic operations, and that the Levinson algorithm is the only one for which the arithmetic complexity can be reduced when the matrix is symmetric.

Finally, we study the common structure of the two-pass algorithms allowing to apply the divide and conquer strategy. The application of this strategy converts the classical algorithms into the class of so-called doubling algorithms which is faster asymptotically [1,2,3,15,16,22,23]. We then present a general doubling algorithm which represents all the doubling algorithms in an exhaustive manner.

The remaining of the paper is divided into six sections as follows:

2. Preliminaries
3. Classical algorithms
4. Summary of the classical algorithms
5. Development of the missing variants
6. Doubling algorithms

7. Conclusion

The preliminary section recalls the Padé approximation and describes the Yule-Walker equation as a Padé approximation problem. The section on classical algorithms is divided into three subsections, presenting successively the three algorithms for which we discuss the following points:

- the Padé approximants to be evaluated
- the number of multiplications required
- the class to which the algorithm belongs

In section 4, we give a summary of these algorithms and point out the missing variants to be developed in section 5. Then, based on the common structure of the classical two-pass algorithms, section 6 gives a general principe for the application of the doubling strategy to the classical algorithms. Finally, we conclude the paper with a short summary.

2. PRELIMINARIES

Solving a Yule-Walker equation is shown to be closely related to the Padé approximation of a formal power series [see ref. 6]. We recall the definition of the Padé approximation problem and its connection with the Yule-Walker equation.

2.1. Padé approximant

Definition [13,14,21]: Let $C(z) = c_0 + c_1 z + c_2 z^2 + ...$ be a formal power series with $c_0 \neq 0$. A rational function of the form $u(z)/v(z)$ is an (m, n) Padé approximant for $C(z)$ if

$$\deg(u(z)) \leq m \tag{1}$$

$$\deg(v(z)) \leq n \tag{2}$$

$$C(z)v(z) - u(z) = O(z^{m+n+1}) \tag{3}$$

For a formal power series, there always exists a Padé approximant of order (m, n) represented by the rational function $r_{m,n}(z)$:

$$r_{m,n}(z) = \frac{p_{m,n}(z)}{a_{m,n}(z)} \tag{4}$$

with $p_{m,n}(z)$ and $a_{m,n}(z)$ relatively prime and $p_{m,n}(0) = c_0$, $a_{m,n}(0) = 1$.

The Padé table of the formal power series C(z) is a doubly infinite array of the uniquely determined rational function $r_{m,n}(z) = p_{m,n}(z)/a_{m,n}(z)$, which can be represented as follows:

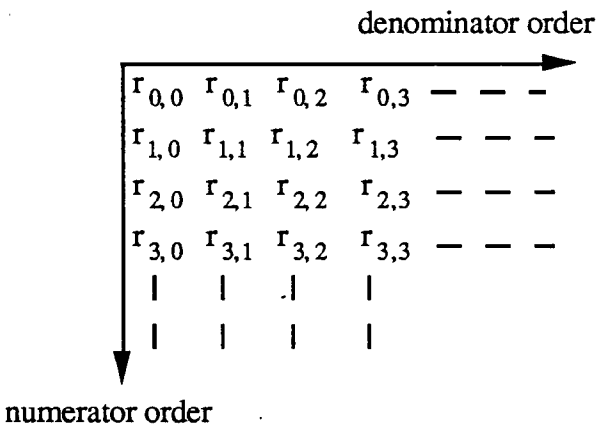


Table 1: Padé table

By definition of $r_{m,0}(z)$, the first column contains the partial sums of C(z):

$$r_{m,0}(z) = \sum_{k=0}^{m} c_k z^k$$

The power series C(z) is said to be normal if, for each pair (m, n), the Maclaurin expansion of $r_{m,n}(z)$ agrees with C(z) exactly up to the power z^{m+n}. The Padé approximant $r_{m,n}$ is normal if it occurs exactly once in the Padé table. The power series C(z) is normal if all its Padé approximants are normal; that is, no two are equal [14].

2.2. Yule-Walker equation

The Yule-Walker (YW) equation is defined as a system of linear equations:

$$\begin{bmatrix} c_n & & c_0 \\ & \ddots & \\ c_{2n} & & c_n \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \alpha_n \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} c_n & & c_0 \\ & \ddots & \\ c_{2n} & & c_n \end{bmatrix} \begin{bmatrix} b_0 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \alpha_n \end{bmatrix} \qquad (5)$$

where a(z) and b(z) ($a(z) = a_0 + a_1z + ... + a_nz^n$, $b(z) = b_0 + b_1z + ... + b_nz^n$) are seen as the forward and backward prediction polynomials of the corresponding system. It is shown in [6] that the solution of the YW equations can be described as a Padé approximation problem. This interpretation is recalled below.

The equation (5) is completed as follows:

$$\begin{bmatrix} c_0 & & \\ & \ddots & 0 \\ & & \ddots \\ c_n & & c_0 \\ & \ddots & \\ & & \ddots \\ c_{2n} & & c_n \\ & \ddots & \\ & & \ddots \\ & & c_{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \bar{p}_0 \\ \vdots \\ \vdots \\ \bar{p}_n \\ 0 \\ \vdots \\ 0 \\ \bar{q}_0 \\ \vdots \\ \bar{q}_{n-1} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} c_0 & & \\ & \ddots & 0 \\ & & \ddots \\ c_n & & c_0 \\ & \ddots & \\ & & \ddots \\ c_{2n} & & c_n \\ & \ddots & \\ & & \ddots \\ & & c_{2n} \end{bmatrix} \begin{bmatrix} b_0 \\ \vdots \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} p_0 \\ \vdots \\ p_{n-1} \\ 0 \\ 0 \\ \vdots \\ q_0 \\ \vdots \\ \vdots \\ q_n \end{bmatrix} \qquad (6)$$

with $\bar{p}_n = q_0 = \alpha_n$. By noting $c(z) = c_0 + c_1z + ... + c_{2n}z^{2n}$, $C(z) = c(z) + O(z^{2n+1})$, $\bar{p}(z) = \bar{p}_0 + \bar{p}_1z + ... + \bar{p}_nz^n$, and $\bar{q}(z) = \bar{q}_0 + \bar{q}_1z + ... + \bar{q}_{n-1}z^{n-1}$, $p(z) = p_0 + p_1z + ... + p_{n-1}z^{n-1}$, and $q(z) = q_0 + q_1z + ... + q_nz^n$, we have the polynomial description of (6):

$$\begin{aligned} C(z)a(z) &= \bar{p}(z) + z^{2n+1}\bar{q}(z) \\ C(z)b(z) &= p(z) + z^{2n}q(z) \end{aligned} \qquad (7)$$

Comparing (7) with the definition of the Padé approximant, we see that $\bar{p}(z)/a(z)$ and $p(z)/b(z)$ with $\deg(\bar{p}(z)) = n$, $\deg(a(z)) = n$, $\deg((p(z)) = n-1$, $\deg(b(z)) = n$ are respectively the Padé approximants of order (n, n) and (n-1, n) for C(z). The solution of the Yule-Walker equation is thus described as a Padé approximation problem.

This simple consideration already provides an interesting information: a_n is the forward predictor, while b_n is the backward one. The understanding of a_n and b_n as Padé approximants of order (n,n) and (n-1,n) respectively shows that the backward predictor is certainly easier to obtain, since it is an approximant of lower order than the forward one.

3. CLASSICAL ALGORITHMS

We present here three classical algorithms: the Levinson algorithm (and Schur algorithm), Euclidean algorithm and Berlekamp-Massey algorithm. Our goal is not to describe the details of the algorithms, but to: 1) interprete them as Padé approximation algorithms; 2) group them together in two classes, namely one-pass algorithms and two-pass algorithms; 3) compare their arithmetic complexity.

Considering that the algorithms require almost as many additions as multiplications, we evaluate only the latter for each algorithm. The number will be noted after each step in a square bracket.

3.1. Levinson Agorithm

We recall the Levinson algorithm as well as the Schur algorithm. For these algorithms, we can find in [7] their interpretation as Padé approximation algorithms. The Levinson algorithm [12,18] solves the following equations:

$$\begin{bmatrix} c_n & & c_0 \\ & \ddots & \\ c_{2n} & & c_n \end{bmatrix} \begin{bmatrix} & \\ \underline{a}_n & \underline{b}_n \\ & \end{bmatrix} = \begin{bmatrix} \alpha_n & 0 \\ 0 & \ddots \\ \ddots & \ddots \\ 0 & \alpha_n \end{bmatrix} \tag{8}$$

which are Yule-Walker equations with $\underline{a}_n$ and $\underline{b}_n$ respectively the forward and backward prediction vectors in the prediction theory. Let C_i be the principal minor of dimension i+1 of the matrix in the above equation, which is supposed to be non singular. The prediction vectors of two succesive orders obey the following recursion relationships:

$$C_i\left\{\begin{bmatrix} a_{i-1} \\ \\ 0 \end{bmatrix} + K_{a,i}\begin{bmatrix} 0 \\ \\ \underline{b}_{i-1} \end{bmatrix}\right\} = \begin{bmatrix} \alpha_{i-1} \\ \underline{0} \\ \beta_{a,i-1} \end{bmatrix} + K_{a,i}\begin{bmatrix} \beta_{b,i-1} \\ \underline{0} \\ \alpha_{i-1} \end{bmatrix} = \begin{bmatrix} \alpha_i \\ \underline{0} \\ \\ \end{bmatrix} \tag{9}$$

$$C_i\left\{\begin{bmatrix} 0 \\ \\ \underline{b}_{i-1} \end{bmatrix} + K_{b,i}\begin{bmatrix} a_{i-1} \\ \\ 0 \end{bmatrix}\right\} = \begin{bmatrix} \beta_{b,i-1} \\ \underline{0} \\ \alpha_{i-1} \end{bmatrix} + K_{b,i}\begin{bmatrix} \alpha_{i-1} \\ \underline{0} \\ \beta_{a,i-1} \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \\ \alpha_i \end{bmatrix} \tag{10}$$

These two equations constitute the main recursion relationships of the Levinson algorithm. The algorithm is summarized below.

ALGORITHM 1 (LEVINSON)

$a_0(z) = b_0(z) = 1, \alpha_0 = c_n$

For $i = 1, ..., n$

$$\beta_{b,i-1} = \sum_{j=0}^{i-1} b_{i-1,j}\, c_{n-1-j} \qquad [i-1]$$

$$\beta_{a,i-1} = \sum_{j=0}^{i-1} a_{i-1,j}\, c_{n+i-j} \qquad [i-1]$$

$$K_{b,i} = -\beta_{b,i-1} / \alpha_{i-1} \qquad [1]$$

$$K_{a,i} = -\beta_{a,i-1} / \alpha_{i-1} \qquad [1]$$

$$\begin{bmatrix} b_i(z) \\ a_i(z) \end{bmatrix} = \begin{bmatrix} z & K_{b,i} \\ K_{a,i} z & 1 \end{bmatrix} \begin{bmatrix} b_{i-1}(z) \\ a_{i-1}(z) \end{bmatrix} \qquad [2(i-1)]$$

$$\alpha_i = \alpha_{i-1} (1 - K_{a,i} K_{b,i}) = \alpha_{i-1} + \beta_{b,i-1} K_{a,i} \qquad [1]$$

We can easily evaluate the number of multiplications required by this algorithm, which is equal to $2n^2 + n$. In the case where the matrix C is symmetric, we have $\beta_{a,i} = \beta_{b,i}$, $K_{a,i} = K_{b,i}$, and $a_i(z) = b_i^{\#}(z) = z^i b_i(z^{-1})$, this number is reduced to $n^2 + n$.

In the above algorithm, the parameters $K_{a,i}$ and $K_{b,i}$, known as reflection coefficients, are computed via the scalar products $\beta_{a,i}$ and $\beta_{b,i}$. These parameters provide the recursion relationship for the prediction polynomials $a_i(z)$ and $b_i(z)$, and the only recursion is carried out on these prediction polynomials. This algorithm is thus classified as a one-pass algorithm.

The Schur algorithm [25] provides an alternative method for computing the reflection coefficients { $K_{a,i}$, $K_{b,i}$ } via the recursion of the polynomials $p_i(z)$, $q_i(z)$, $\overline{p_i}(z)$ and $\overline{q_i}(z)$:

ALGORITHM 2 (SCHUR)

$$\begin{bmatrix} p_i(z) \\ \bar{p}_i(z) \end{bmatrix} = \begin{bmatrix} z & K_{b,i} \\ K_{a,i}z & 1 \end{bmatrix} \begin{bmatrix} p_{i-1}(z) \\ \bar{p}_{i-1}(z) \end{bmatrix}$$

$$z\begin{bmatrix} q_i(z) \\ z\bar{q}_i(z) \end{bmatrix} = \begin{bmatrix} z & K_{b,i} \\ K_{a,i}z & 1 \end{bmatrix} \begin{bmatrix} q_{i-1}(z) \\ z\bar{q}_{i-1}(z) \end{bmatrix}$$

with

$$K_{b,i+1} = -p_{i,n-1}/\bar{p}_{i,n}$$
$$K_{a,i+1} = -\bar{q}_{i,0}/q_{i,0}$$

where

$$p_i(z) = \sum_{j=0}^{n-1} p_{i,j} z^j, \qquad \bar{p}_i(z) = \sum_{j=0}^{n} \bar{p}_{i,j} z^j$$
$$q_i(z) = \sum_{j=0}^{n} q_{i,j} z^j, \qquad \bar{q}_i(z) = \sum_{j=0}^{n-1} \bar{q}_{i,j} z^j.$$

The variables $p_{i,j}$, $q_{i,j}$, $\bar{p}_{i,j}$, $\bar{q}_{i,j}$ involved in the algorithm are found in the following linear equations:

$$\begin{bmatrix} c_0 & & & \\ & \ddots & & 0 \\ & & \ddots & \\ c_n & & & c_0 \\ & \ddots & & \\ & & \ddots & \\ c_{2n} & & & c_n \\ & \ddots & & \\ 0 & & \ddots & \\ & & & \ddots \end{bmatrix} \left[\begin{array}{ccc|ccc} b_{0,0} & & b_{n,0} & a_{0,0} & & a_{n,0} \\ & \ddots & & & \ddots & \\ 0 & & b_{i,i} & 0 & & a_{i,i} \\ & & & & & \\ & & b_{n,n} & & & a_{n,n} \end{array}\right] = \left[\begin{array}{ccc|ccc} p_{0,0} & & p_{n,0} & \bar{p}_{0,0} & & \bar{p}_{n,0} \\ & & & & & \\ p_{0,n-1} & & p_{n,n-1} & & & \\ q_{0,0} & & & \bar{p}_{0,n} & & \bar{p}_{n,n} \\ & \ddots & 0 & \bar{q}_{0,0} & & \\ & & & & \ddots & 0 \\ q_{0,n} & & q_{n,0} & \bar{q}_{0,n-1} & & \\ & \ddots & & & \ddots & \bar{q}_{n,0} \\ 0 & & & 0 & & \\ & \ddots & & & \ddots & \end{array}\right] \tag{11}$$

Obviously, we have $p_{i,n-1} = \beta_{b,i}$, $\bar{q}_{i,0} = \beta_{a,i}$, and $q_{i,0} = \bar{p}_{i,n} = \alpha_i$.

The Levinson type two-pass algorithm is obtained by combining the computation of the reflection coefficients $K_{a,i}$ and $K_{b,i}$ in Schur's manner, and the prediction polynomials $a_i(z)$

and $b_i(z)$ in Levinson's manner, in such a way that the recursive iterations are performed altogether on $p_i(z)$, $q_i(z)$, $\bar{p}_i(z)$, $\bar{q}_i(z)$ and $a_i(z)$, $b_i(z)$. The combined algorithm called Levinson/Schur algorithm is written below:

ALGORITHM 3 (LEVINSON/SCHUR)

First pass:

$$p_0(z) = c_0 + c_1 z + \ldots + c_{n-1} z^{n-1}, \quad \bar{p}_0(z) = c_0 + c_1 z + \ldots + c_n z^n,$$
$$q_0(z) = c_n + c_{n+1} z + \ldots + c_{2n} z^n, \quad \bar{q}_0(z) = c_{n+1} + \ldots + c_{2n} z^{n-1},$$

For i = 1, ..., n

$$K_{b,i} = - p_{i-1,n-1} / \bar{p}_{i-1,n} \qquad [1]$$
$$K_{a,i} = - \bar{q}_{i-1,0} / q_{i-1,0} \qquad [1]$$

$$\begin{bmatrix} p_i(z) \\ \bar{p}_i(z) \end{bmatrix} = \begin{bmatrix} z & K_{b,i} \\ K_{a,i} z & 1 \end{bmatrix} \begin{bmatrix} p_{i-1}(z) \\ \bar{p}_{i-1}(z) \end{bmatrix} \qquad [2(n-i)]$$

$$z \begin{bmatrix} q_i(z) \\ z\bar{q}_i(z) \end{bmatrix} = \begin{bmatrix} z & K_{b,i} \\ K_{a,i} z & 1 \end{bmatrix} \begin{bmatrix} q_{i-1}(z) \\ z\bar{q}_{i-1}(z) \end{bmatrix} \qquad [2(n-i)]$$

Second pass:

$$a_0(z) = b_0(z) = 1$$

For i = 1, ..., n

$$\begin{bmatrix} b_i(z) \\ a_i(z) \end{bmatrix} = \begin{bmatrix} z & K_{b,i} \\ K_{a,i} z & 1 \end{bmatrix} \begin{bmatrix} b_{i-1}(z) \\ a_{i-1}(z) \end{bmatrix} \qquad [2(i-1)]$$

The number of multiplications required for evaluating this algorithm is $3n^2$-n, which is more than the corresponding one-pass algorithm (Levinson algorithm). If the matrix is symmetric, this number may be reduced straightforwardly to $1.5n^2$-0.5n.

As seen in section 2, the ratio $p_i(z)/b_i(z)$ and $\overline{p}_i(z)/a_i(z)$, with $p_i(z) = p_{i,0} + p_{i,1}\, z + ... + p_{i,n-1}\, z^{n-1}$, and $\overline{p}_i(z) = \overline{p}_{i,0} + \overline{p}_{i,1}\, z + ... + \overline{p}_{i,n}\, z^n$, are respectively the Padé approximants of types (n-1, i) and (n, i) for C(z). Thus, for i from 0 to n, it is seen that this algorithm computes the Padé approximants of types (n-1, 0), (n, 0), (n-1, 1), (n, 1), ..., (n-1, n), (n, n), which has the form of a sawtooth related to a row in the Padé table (see fig.1). This is a result given in [7].

Based on a conventional three-term recursion for polynomials which are orthogonal on the unit circle, a variant of the Levinson algorithm has been proposed in [26] . This variant computes only the polynomials $a_i(z)$ or $b_i(z)$, the forward or backward predictor, but not both of them. As a consequence, only the Padé approximants of types (n-1,0), (n-1,1), ..., (n-1,n) (or (n,0), (n,1), ..., (n,n)) are computed, which constitute a simple row in the Padé table.

We note that all these types of algorithms (Levinson, Schur, Levinson/Schur) require that all the principal minors of matrix C be not singular, that is, the polynomial c(z) must be normal. The generalization of this algorithm to the case where the matrix C has any rank profile can be found in [9, 26].

3.2. Euclidean Algorithm

The Euclidean algorithm was originally developed for computing the greatest common divisor (GCD) of two entries or two polynomials [5,20]. The application of this algorithm to the computation of Padé approximants is an original suggestion of Kronecker [17]. Using this algorithm for solving a YW equation is related directly to its application for Padé approximation [5]. We first recall the original algorithm for the computation of the GCD of two polynomials, and its extension for solving YW equation. The relation between this algorithm and the Padé approximation has been described in [6].

3.2.1. Extended Euclidean algorithm

Let us consider the problem of computing the GCD of two polynomials s(z) and t(z): GCD(s, t), and the "comultiplicators" u(z) and v(z):

$$s(z)\, u(z) + t(z)\, v(z) = GCD(s, t) \qquad (12)$$

Suppose that $\deg(s(z)) \geq \deg(t(z))$, the Euclidian algorithm computes GCD(s, t) by a sequence of recursive divisions: let $s_0(z) = s(z)$, $s_1(z) = t(z)$, we construct a sequence of remainder polynomials $s_i(z)$ for $1 \leq i \leq n$ by Euclidean division:

$$s_{i-1}(z) = qu_i(z)\, s_i(z) + s_{i+1}(z) \qquad (13)$$

The recursion being continued till i = n when $s_{n+1}(z) = 0$, the resulting $s_n(z)$ is the GCD(s, t).

Once the quotients $qu_i(z)$ are known, we can evaluate u(z) and v(z). The algorithm involving these two passes (computation of GCD then evaluation of the comultiplicators) is called "Extended Euclidean Algorithm" [8], and is summarized below:

ALGORITHM 4 (EXTENDED EUCLIDEAN)

First pass:

$s_0(z) = s(z)$; $s_1(z) = t(z)$

For $1 \le i \le n$

$qu_i(z)$ = quotient of the Euclidean division $s_{i-1}(z)/s_i(z)$

$$\begin{bmatrix} s_i(z) \\ s_{i+1}(z) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -qu_i(z) \end{bmatrix} \begin{bmatrix} s_{i-1}(z) \\ s_i(z) \end{bmatrix}$$

$$\begin{bmatrix} s_n(z) \\ s_{n+1}(z) \end{bmatrix} = \begin{bmatrix} GCD(s, t) \\ 0 \end{bmatrix}$$

Second pass:

$$\begin{bmatrix} u_0 & v_0 \\ u_1 & v_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

For i= 1, ... n

$$\begin{bmatrix} u_i & v_i \\ u_{i+1} & v_{i+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -qu_i(z) \end{bmatrix} \begin{bmatrix} u_{i-1} & v_{i-1} \\ u_i & v_i \end{bmatrix}$$

3.2.2. Extended Euclidean algorithm for solving YW equation

To apply the extended Euclidean algorithm for solving a YW equation, we consider the polynomial description of the problem, i.e. the first equation of (7):

$$-z^{2n+1}\ \overline{q}(z) + c(z)\, a(z) = \overline{p}(z), \qquad (14)$$

By taking $s(z) = -z^{2n+1}$, $t(z) = c(z)$, we remark that (11) and (13) are of the same form with $\bar{q}(z) = u(z)$ and $a(z) = v(z)$, if $\bar{p}(z)$ is the GCD of $s(z)$ and $t(z)$. In fact, if we apply the extended Euclidean algorithm to the polynomials $s(z) = -z^{2n+1}$ and $t(z)=c(z)$, the GCD would be found as a result of the recursion when $\deg(\bar{p}(z)) = 0$. However, we are interested in the solution of the YW equations. If $c(z)$ is normal, this solution is found at the n-th recursion, the solution $a(z)$ is of degree n, and $\deg(\bar{p}(z))$ is also equal to n. If $c(z)$ is not normal, then the solution of the YW equations is found at an intermediate step where $\deg(a(z)) \leq n$ and $\deg(\bar{p}(z)) \leq n$. Thus the Euclidean algorithm for solving equation (13) is as follows (see Ref. [5] for a detailed demonstration):

ALGORITHM 5 (EUCLIDEAN)

First pass:

$$\bar{p}_{-1}(z) = -z^{2n+1}, \quad \bar{p}_0(z) = c(z)$$

For i = 0, ... , r

$qu_i(z)$ = quotient of the Euclidean division $\bar{p}_{i-1}(z) / \bar{p}_i(z)$ [3]

$$\begin{bmatrix} \bar{p}_i(z) \\ \bar{p}_{i+1}(z) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -qu_i(z) \end{bmatrix} \begin{bmatrix} \bar{p}_{i-1}(z) \\ \bar{p}_i(z) \end{bmatrix} \qquad [4n-4i-2]$$

$$\deg(\bar{p}_r(z)) \geq n+1$$
$$\deg(\bar{p}_{r+1}(z)) \leq n$$

Second pass:

$$a_{-1}(z) = 0, \; a_0(z) = 1$$

For i = 0, ..., r

$$\begin{bmatrix} a_i(z) \\ a_{i+1}(z) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -qu_i(z) \end{bmatrix} \begin{bmatrix} a_{i-1}(z) \\ a_i(z) \end{bmatrix} \qquad [2(i+1)]$$

Solution

$$a(z) = a_{r+1}(z) / a_{r+1}(0)$$

In the case where $c(z)$ is normal, r = n-1, and the number of multiplications required by this algorithm is $3n^2+4n$.

The evolution of the parameters involved in this algorithm is shown below in matrix form:

$$\begin{bmatrix} c_0 & & & \\ & \ddots & 0 & \\ & & \ddots & \\ c_n & & & c_0 \\ & \ddots & & \\ & & \ddots & \\ c_{2n} & & & c_n \\ & \ddots & & \\ 0 & & \ddots & \\ & & & c_{2n} \end{bmatrix} \begin{bmatrix} a_{0,0} & & & a_{n,0} \\ & \ddots & & \\ 0 & & a_{i,i} & \\ & & & a_{n,n} \end{bmatrix} = \begin{bmatrix} \overline{p}_{0,0} & & & \overline{p}_{n,0} \\ \cdot & & & \\ & & & \overline{p}_{n,n} \\ \cdot & & \cdot & 0 \\ & \cdot & & \\ \overline{p}_{0,2n} & & & 0 \\ & \cdot & & \overline{q}_{n,0} \\ 0 & & \cdot & \\ & & & \overline{q}_{n,n-1} \end{bmatrix} \qquad (15)$$

This algorithm is of the two-pass type. It computes successively the Padé approximants (2n, 0), (2n-1, 1), (2n-2, 2), ..., (n, n), which form an antidiagonal of the Padé table (fig.1), a result having been given in [6]. If c(z) is a normal power series, all these approximants are distinct and deg(a(z)) = deg($\overline{p}$(z)) = n.

We observe that, this algorithm computes successively the solution of the following linear equations in which $a_i(z) = a_{i0} + a_{i1} z + ... + a_{ii} z^i$ are seen as the forward prediction polynomials.

$$\begin{bmatrix} c_{2n-i} & & c_{2n-2i} \\ & \ddots & \\ c_{2n} & & c_{2n-i} \end{bmatrix} \begin{bmatrix} a_{i,0} \\ \vdots \\ a_{i,i} \end{bmatrix} = \begin{bmatrix} \overline{p}_{i,2n-i} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad (16)$$

It will be shown in section 5 that it is possible to obtain in addition the backward prediction polynomials without increasing the computational complexity (Algorithm 8 below).

3.3. Berlekamp-Massey Algorithm

This algorithm [4] was originally developed for decoding the BCH codes. It was interpreted as the synthesis of an autoregressive filter [19], with the following system of equations:

$$\begin{bmatrix} c_{n-1} & & c_0 \\ & \ddots & \\ c_{2n-2} & & c_{n-1} \end{bmatrix} \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} -c_n \\ \vdots \\ -c_{2n-1} \end{bmatrix} \qquad (17)$$

which is equivalent to

$$\begin{bmatrix} c_n & & c_0 \\ & \ddots & \\ c_{2n} & & c_n \end{bmatrix} \begin{bmatrix} 1 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \alpha_n \end{bmatrix} \qquad (18)$$

This algorithm computes therefore the backward prediction vector in the prediction theory, while the Euclidean algorithm computes the forward one, and the Levinson algorithm computes both of them. Nevertheless, changing the initial algorithm to obtain the forward one is trivial since we need only to carry out the algorithm on the reverse of the polynomial c(z). A variant which computes both forward and backward prediction vectors is developed in section 5 (algorithm 10).

We give here only the algorithm. For its derivation, the reader is referred to [5,4,19].

ALGORITHM 6 (BERLEKAMP-MASSEY)

$b_0(z) = t_0(z) = 1,$
$L=0,\ dt=1$

For r = 0, ..., 2n-1

$$d_r = \sum_{j=0}^{L} b_j c_{r-j} \qquad [L]$$

if (d_r=0), then

$$\{ \begin{bmatrix} b_{r+1}(z) \\ t_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & z \end{bmatrix} \begin{bmatrix} b_r(z) \\ t_r(z) \end{bmatrix} \};$$

if (($d_r \neq 0$) and (2L>r)), then

$$\{ \begin{bmatrix} b_{r+1}(z) \\ t_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & -d_r dt^{-1} z \\ 0 & z \end{bmatrix} \begin{bmatrix} b_r(z) \\ t_r(z) \end{bmatrix} \} \qquad [L]$$

else (($d_r \neq 0$) and ($2L \leq r$))
{ $L = r+1-L$

$$\begin{bmatrix} b_{r+1}(z) \\ t_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & -d_r dt^{-1} z \\ 1 & 0 \end{bmatrix} \begin{bmatrix} b_r(z) \\ t_r(z) \end{bmatrix} \qquad [L]$$

$dt = d_r$ }

Solution $b(z) = b_r(z)$.

We give here some explanation about the parameters used in the algorithm: L is the order of polynomials $b_r(z)$, which is different from the recursion order r. The recursions are carried out on polynomials $b_r(z)$, r = 0, 1, ..., 2n-1, and some of them are stored in the auxiliary polynomials $t_r(z)$ when they are denominators of a Padé approximants of c(z). In the case where c(z) is normal, only the polynomials at even recursion orders will be stored. It is easily seen that this algorithm is of the one-pass type, since the polynomials $b_r(z)$ are obtained in a non recursive manner.

The number of multiplications required by this algorithm in the case where c(z) is normal, is $2n^2+2n$. We remark that it is of the same order as that of the other one-pass algorithm (Levinson algorithm) stated before.

We emphasize the fact that all the $b_r(z)$ are not denominators of Padé approximants of c(z), but only the ones that are stored in $t_r(z)$ are the denominators of the Padé approximant of order (r/2, r/2) with r even in the case where c(z) is normal. Thus, for r = 0, ..., 2n, this algorithm computes the denominators of the Padé approximants of types (0,1), ..., (n-1,n), which constitutes a main diagonal in the Padé table.

4. SUMMARY OF THE CLASSICAL ALGORITHMS

As seen above most of the results provided in section 3 have been explained elsewhere in the literature, at least partially. Nevertheless, only a global view of YW equations as a Padé approximation problem will allow the derivation of new results:

1) All these three algorithms evaluate the denominator of the Padé approximant of type (n,n) or (n-1,n). They cover various paths in the Padé table:

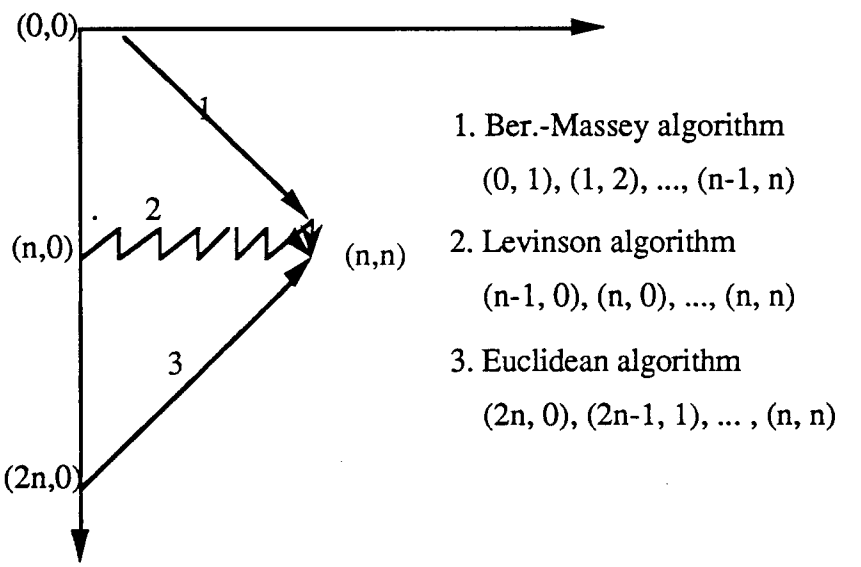


fig. 1 evolution of the various algorithms in the Padé table.

2) The Levinson algorithm is of the one-pass type. We obtain the corresponding two-pass version by a combination with the Schur algorithm. The Euclidean algorithm is shown to be of the two-pass type, its one pass version is missing. The Berlekamp-Massey algorithm is of the one-pass type, its two-pass version is to be developed.

3) The Levinson algorithm computes both forward and backward predictors of the associated system. The Euclidean algorithm and the Berlekamp-Massey algorithm compute respectively the forward and backward predictor, the variants of these two algorithms for computing both predictors altogether are missing.

The following table shows what are the algorithms already known, and what are the missing variants to be developed in the next section.

	One-pass	Two-pass	Forward or Backward	Forward and Backward
Levinson	Known	Known	Known	Known
Euclidean	Missing	Known	Known	Missing
Berlekamp	Known	Missing	Known	Missing

Table 2 Variants of the algorithms known or missing

5. DEVELOPMENT OF THE MISSING VARIANTS

5.1. The variants of the Euclidean algorithm

By comparing the Euclidean algorithm with that of Levinson/Schur, we remark that the first pass looks like Schur's algorithm, and the second one looks like Levinson's. We are therefore motivated to develop a one-pass algorithm which would be similar to that of Levinson, by hoping that it will be more efficient than the initial (two-pass) Euclidean algorithm.

As for the Levinson algorithm, we suppose that the formal power series c(z) associated with the Toeplitz matrix is normal. The idea is to replace the computation of quotients $qu_i(z)$ in the initial algorithm by a scalar product which is a non recursive computation, just like the computation of the reflection coefficients in the Levinson algorithm. For this, we examine the explicit expression of $qu_i(z)$:

$$qu_i(z) = \text{quotient of the Euclidean division } \bar{p}_{i-1}(z) / \bar{p}_i(z) \qquad (19)$$

with $\bar{p}_{i-1}(z) = \bar{p}_{i-1,0} + \bar{p}_{i-1,1} z + \ldots + \bar{p}_{i-1,2n-i+1} z^{2n-i+1}$

$\bar{p}_i(z) = \bar{p}_{i,0} + \bar{p}_{i,1} z + \ldots + \bar{p}_{i,2n-i} z^{2n-i}$

c(z) is normal, then $\bar{p}_{i,2n-i} \neq 0$, we obtain

$$qu_i(z) = qu_{i,0} + qu_{i,1} z \qquad (20)$$

with $qu_{i,1} = \bar{p}_{i-1,2n-i+1} / \bar{p}_{i,2n-i}$

$qu_{i,0} = (\bar{p}_{i-1,2n-i} - qu_{i,1} \bar{p}_{i,2n-i-1}) / \bar{p}_{i,2n-i}$

And, the recursion of $\overline{p}_i(z)$ is replaced by the computation of its last two elements:

$$\overline{p}_{i,2n-i} = [\, c_{2n-i}, \ldots, c_{2n-2i} \,]\, [\, a_{i,0}, \ldots, a_{i,i} \,]^T \qquad (21)$$

$$\overline{p}_{i,2n-i-1} = [\, c_{2n-i-1}, \ldots, c_{2n-2i-1} \,]\, [\, a_{i,0}, \ldots, a_{i,i} \,]^T \qquad (22)$$

This algorithm is summarized below:

ALGORITHM 7 (ONE-PASS EUCLIDEAN)

$$\begin{bmatrix} \overline{p}_{-1,2n} \\ \overline{p}_{-1,2n+1} \end{bmatrix} \leftarrow \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} a_{-1}(z) \\ a_0(z) \end{bmatrix} \leftarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

For i = 0, ..., n-1

$$\overline{p}_{i,2n-i} = \sum_{j=0}^{i} a_{i,j} c_{2n-i-j} \qquad [i+1]$$

$$\overline{p}_{i,2n-i-1} = \sum_{j=0}^{i} a_{i,j} c_{2n-i-1-j} \qquad [i+1]$$

$$q u_{i,1} = \overline{p}_{i-1,2n-i+1} / \overline{p}_{i,2n-i} \qquad [1]$$

$$q u_{i,0} = (\overline{p}_{i-1,2n-i} - q u_{i,1}\, \overline{p}_{i,2n-i-1}) / \overline{p}_{i,2n-i} \qquad [2]$$

$$a_{i+1}(z) = a_{i-1}(z) - (q u_{i,0} + q u_{i,1} z) a_i(z) \qquad [2i+2]$$

Solution:

$$a(z) = a_n(z) / a_n(0)$$

The resulting algorithm requires $(2n^2+5n)$ multiplications. Indeed, this one-pass algorithm is computationally more efficient than the initial two-pass algorithm. It should be noted that this algorithm supposes that the polynomial c(z) is normal, but it can be generalized for the anormal case without difficulty.

As we have mentioned before, the algorithm computes only the forward predictor solution. The variant which provides both the forward and backward one is developed below.

First, we introduce an auxiliary polynomial $b_i(z) = b_{i0} + b_{i1}\, z + \ldots + b_{ii}\, z^i$, which represents a backward prediction polynomial. It is the solution of the following linear equations:

$$\begin{bmatrix} c_{2n-i} & & c_{2n-2i} \\ & \ddots & \\ c_{2n} & & c_{2n-i} \end{bmatrix} \begin{bmatrix} b_{i,0} \\ \vdots \\ b_{i,i} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ q_{i,0} \end{bmatrix} \qquad (23)$$

Then, we organize the variables involved in the algorithm in the following matrices:

$$\begin{bmatrix} c_0 & & & \\ & \ddots & 0 & \\ & & & \\ c_n & & c_0 & \\ & \ddots & & \\ & & & \\ c_{2n} & & c_n & \\ & \ddots & & \\ 0 & & \ddots & \\ & & & c_{2n} \end{bmatrix} \begin{bmatrix} a_{0,0} & & & a_{n,0} & b_{0,0} & & b_{n,0} \\ & \ddots & & & & \ddots & \\ 0 & a_{i,i} & & 0 & b_{i,i} & & \\ & & a_{n,n} & & & & b_{n,n} \end{bmatrix} = \begin{bmatrix} \overline{p}_{0,0} & & \overline{p}_{n,0} & p_{0,0} & & & p_{n,0} \\ \ddots & & \ddots & \ddots & & & \ddots \\ & & & & & & p_{n,n-1} \\ & & \overline{p}_{n,n} & & & \ddots & 0 \\ & \ddots & 0 & & \ddots & & \\ \ddots & & & p_{0,2n-1} & 0 & & 0 \\ p_{0,2n} & 0 & 0 & q_{0,0} & \ddots & \ddots & q_{n,0} \\ & \ddots & \overline{q}_{n,0} & \ddots & & & \\ 0 & \ddots & & 0 & \ddots & & \\ & & \overline{q}_{n,n-1} & & & & q_{n,n} \end{bmatrix}$$

(24)

The recursions for $a_i(z)$ and $b_i(z)$ are the same ones as for $\overline{p}_i(z)$ and $p_i(z)$, which are easily deduced by decreasing at each step one degree of the polynomials $\overline{p}_i(z)$ and $p_i(z)$. This recursion depends only on the last coefficients of the $\overline{p}_i(z)$ and $p_i(z)$ ($\overline{p}_{i,2n-i}$ and $p_{i,2n-i+1}$). For example, it may be as follows:

$$\begin{aligned} a_i(z) &= -(p_{i-1,2n-i} / \overline{p}_{i-1,2n-i+1}) a_{i-1}(z) + z\, b_{i-1}(z) \\ b_i(z) &= -(p_{i-1,2n-i} / \overline{p}_{i,2n-i}) a_i(z) + b_{i-1}(z) \end{aligned} \qquad (25)$$

If these last coefficients are computed via the recursion of the polynomials $\overline{p}_i(z)$ and $p_i(z)$, then the resulting algorithm is of the two-pass type. If they are computed via scalar products, we obtain a one-pass algorithm. We give here only the one-pass version.

ALGORITHM 8 (Forward and backward EUCLIDEAN)

$$\begin{bmatrix} a_0(z) \\ b_0(z) \end{bmatrix} \longleftarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$p_{0,2n} = c_{2n}$$

For i = 1, ..., n

$$p_{i-1,2n-i} = \sum_{j=0}^{i-1} b_{i-1,j} c_{2n-i-j} \qquad [i]$$

$$a_i(z) = -(p_{i-1,2n-i} / \overline{p}_{i-1,2n-i+1}) a_{i-1}(z) + z\, b_{i-1}(z) \qquad [i+1]$$

$$\overline{p}_{i,2n-i} = \sum_{j=0}^{i} a_{i,j} c_{2n-i-j} \qquad [i+1]$$

$$b_i(z) = -(p_{i-1,2n-i} / \overline{p}_{i,2n-i}) a_i(z) + b_{i-1}(z) \qquad [i+2]$$

Solution:

$$a(z) = a_n(z)/a_n(0)$$

$$b_n(z) = b_n(z) / b_n(0)$$

The resulting algorithm requires $(2n^2 + 6n)$ multiplications, which is almost the same number as that obtained for the one pass Euclidean algorithm which provides only the forward prediction polynomial, i.e., the backward predictor is given nearly for free in this new algorithm.

5.2. The variants of the Berlekamp-Massey algorithm

A two-pass Berlekamp-Massey algorithm is developed for completing the classical algorithms in both classes, and also for obtaining an algorithm which makes the development of its associated doubling algorithm easier. Observe that the parameter d_r is the first element of the vector shown in the following matrix description:

$$\begin{bmatrix} c_0 & & \\ & \ddots & 0 \\ & & \\ c_n & & c_0 \\ & \ddots & \\ & & \\ c_{2n} & & c_n \end{bmatrix} \begin{bmatrix} 1 & 1 & & & 1 \\ & b_{1,1} & b_{1,2} & & \ddots \\ & 0 & & b_{3,2} & \ddots \\ & & & & b_{2n,n} \end{bmatrix} = \begin{bmatrix} d_0 & & & \ddots \\ & d_1 & & \\ \ddots & & & \\ & & d_r & 0 \\ & & \ddots & \\ & & & \\ d_{0,2n} & & d_{r,2n-r} & d_{2n} \end{bmatrix}$$

$$\Uparrow \qquad \Uparrow$$

$$\underline{d}_0 \qquad \underline{d}_r \qquad (26)$$

Hence, the two-pass algorithm is obtained by computing these parameters via the recursion of the vectors $\underline{d}_r$, which constitutes the first pass of the algorithm. The recursion relationship is exactly the same one as that for $\underline{b}_r$ (or $b_r(z)$). The algorithm is described below:

ALGORITHM 9 (TWO-PASS BERLEKAMP-MASSEY)

First pass:

$d_0(z) = c(z)$, $dt = 1$, $d_0'(z) = c(z)$, $L = 0$

For $r = 0, 1, \ldots, 2n-1$

$d_r = d_{r,r}$ (the first element of $d_r(z)$)

if ($d_r = 0$), then

$$\{ \begin{bmatrix} d_{r+1}(z) \\ d'_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & z \end{bmatrix} \begin{bmatrix} d_r(z) \\ d'_r(z) \end{bmatrix} \}$$

if [($d_r \neq 0$) and ($2L > r$)], then

$$\{ \begin{bmatrix} d_{r+1}(z) \\ d'_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & -d_r dt^{-1} z \\ 0 & z \end{bmatrix} \begin{bmatrix} d_r(z) \\ d'_r(z) \end{bmatrix} \} \qquad [2n-r+1]$$

if [($d_r \neq 0$) and ($2L \leq r$)], then

$$\{ \begin{bmatrix} d_{r+1}(z) \\ d'_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & -d_r dt^{-1} z \\ 1 & 0 \end{bmatrix} \begin{bmatrix} d_r(z) \\ d'_r(z) \end{bmatrix} \qquad [2n-r+1]$$

$L = r+1-L$

$dt = d_r$ }

Second pass:

$b_0(z) = t_0(z) = 1,$

$L=0,\ dt=1$

For $r = 0, ..., 2n-1$

if $(d_r=0)$, then

$$\{ \begin{bmatrix} b_{r+1}(z) \\ t_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & z \end{bmatrix} \begin{bmatrix} b_r(z) \\ t_r(z) \end{bmatrix} \};$$

if ($(d_r \neq 0)$ and $(2L>r)$), then

$$\{ \begin{bmatrix} b_{r+1}(z) \\ t_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & -d_r dt^{-1} z \\ 0 & z \end{bmatrix} \begin{bmatrix} b_r(z) \\ t_r(z) \end{bmatrix} \} \qquad \text{[L-1]}$$

else ($(d_r \neq 0)$ and $(2L \leq r)$)

{ $L = r+1-L$

$$\begin{bmatrix} b_{r+1}(z) \\ t_{r+1}(z) \end{bmatrix} = \begin{bmatrix} 1 & -d_r dt^{-1} z \\ 1 & 0 \end{bmatrix} \begin{bmatrix} b_r(z) \\ t_r(z) \end{bmatrix} \qquad \text{[L-1]}$$

$dt = d_r$ }

It is easily checked out that the number of multiplications required for evaluating this algorithm is equal to $3n^2+2n$ in the case where c(z) is normal. It is thus of the same order of magnitude as that of the other two-pass algorithms described before.

As for the Euclidean algorithm, we can also modify a Berlekamp-Massey algorithm for computing both the forward and backward prediction polynomials. The development of this algorithm follows the same lines and will not be repeated. The one-pass version is provided below:

ALGORITHM 10 (Forward and backward BERLEKAMP-MASSEY)

$$\begin{bmatrix} a_0(z) \\ b_0(z) \end{bmatrix} \longleftarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$d_{0,0} = c_0,\ r = 0$

For i = 1, ..., n

$r = r+1$

$$\bar{d}_{i-1,0} = \sum_{j=0}^{i-1} b_{i-1,j} c_{r-j} \qquad [i-1]$$

$$a_i(z) = b_{i-1}(z) - (\bar{d}_{i-1,0} / d_{i-1,0}) z a_{i-1}(z) \qquad [i]$$

$r = r+1$

$$d_{i,0} = \sum_{j=0}^{i} a_{i,j} c_{r-j} \qquad [i]$$

$$b_i(z) = a_i(z) - (d_{i,0} / \bar{d}_{i-1,0})\ z\ b_{i-1}(z) \qquad [i]$$

Solution:

$a(z) = a_n(z)$

The number of multiplications required is $(2n^2 + n)$, which is even less than that of the original algorithm, while providing a set of additional forward prediction polynomials ($a_0(z)$, ..., $a_n(z)$).

We have thus completed the missing variants for the three classical algorithms. In table 3, we give the number of multiplications for each version of the algorithms described before.

	LEVINSON		EUCLIDEAN		BERLEKAMP	
	One-pass	Two-pass	One-pass	Two-pass	One-pass	Two-pass
a/b	$2n^2$	$3n^2$-3n	$2n^2$+5n	$3n^2$+4n	$2n^2$+2n	$3n^2$+2n
a+b	$2n^2$+n	$3n^2$-n	$2n^2$+6n	$3n^2$+3n	$2n^2$+n	$3n^2$

Table 3: Numbers of multiplications required for each algorithm

where a/b: forward or backward predictor to be computed

a+b: forward and backward predictors to be computed

We remark that the algorithms providing both forward and backward predictors have almost the same computational complexity as those providing only the forward or backward predictor. All the algorithms of one-pass type require fewer arithmetic operations than the two-pass algorithms ($O(2n^2)$ versus $O(3n^2)$). Nevertheless, the structure of the two-pass algorithms is fundamental for the development of the doubling algorithms which we will describe in the next section.

Some differences are to be mentioned between the three classical algorithms. First, the Levinson algorithm is the only one which can be simplified for a symmetric matrix, since the other algorithms perform the recursions on partial matrices that are not symmetric, excepted at the very last step. Next, these algorithms will have different finite precision properties since the recursion relationships are related to the inverse of different minors of the Toeplitz matrix (the principal minors for Levinson algorithm, the minors along the second diagonal for Euclidean's (from bottom to top) and Berlekamp's (from top to bottom)), and the condition number of different minors are different. The detailed analysis is out of the scope of this paper.

6. DOUBLING ALGORITHMS

All the algorithms that we have discussed so far require a number of multiplications which is proportional to n^2. In this section we show how the divide and conquer strategy [5] can be used to reduce the computational complexity of all these algorithms for large n. Our goal is not to study each algorithm in detail, but to retrieve the essential of the problem, and to try to give a unified presentation of the doubling algorithms.

6.1. A Unified Presentation of The Classical Algorithms

Despite the diversity of the classical algorithms, they have a common structure on which the doubling strategy relies.

First, we remark that the development of a doubling algorithm based on a two-pass algorithm is straightforward (e.g. the development of the doubling Euclidean algorithm [5]). As for a one-pass algorithm, it relies implicitly on the corresponding two-pass version, since a scalar product in the algorithm can be seen as an element of some polynomial. The computation of the scalar product is then replaced by the computation of the corresponding polynomial, which can be found in the 2-pass version of the algorithm. A practical example is the development of the doubling Berlekamp-Massey algorithm (cf. The Recursive Berlekamp-Massey Algorithm in [5]).

Then, we note that the recursion relationship is the same one for the polynomials in the first pass as for the second pass. Moreover, the recursion in each pass is carried out either on one polynomial sequence involving three terms (three same type of polynomials of successive order) or on two polynomial sequences involving two terms. This means that the recursion is always in the following general form:

$$\begin{bmatrix} s_i(z) \\ t_i(z) \end{bmatrix} = F_i(z) \begin{bmatrix} s_{i-1}(z) \\ t_{i-1}(z) \end{bmatrix} \qquad (27)$$

with $F_i(z)$ a polynomial matrix of dimensions 2x2. We have $t_i(z) = s_{i-1}(z)$ in the case where three terms recursion is used.

Thus, we have picked up the common structure of the classical algorithms for the development of the corresponding doubling algorithms. The following general presentation of the algorithms summarizes this stucture:

First pass:

Initialization of $u_0(z)$ and $v_0(z)$

For i = 1, 2, ..., n

$F_i(z)$ = function of $(u_{i-1}(z), v_{i-1}(z))$

$$\begin{bmatrix} u_i(z) \\ v_i(z) \end{bmatrix} = F_i(z) \begin{bmatrix} u_{i-1}(z) \\ v_{i-1}(z) \end{bmatrix}$$ /* outgoing recursion */

Second pass:

Initialisation of $a_0(z)$ and $b_0(z)$

For i = 1, 2, ..., n

$$\begin{bmatrix} a_i(z) \\ b_i(z) \end{bmatrix} = F_i(z) \begin{bmatrix} a_{i-1}(z) \\ b_{i-1}(z) \end{bmatrix}$$ /* incoming recursion */

Now, this general representation allows us to explain the principle of the doubling algorithms for all these classical algorithms in a unified manner.

6.2. Application of The Doubling Strategy

We start with the definition of a polynomial matrix of dimensions 2x2:

$$F_{k,l}(z) = \prod_{i=k}^{l} F_i(z) \tag{28}$$

Using this definition, the polynomials $a_i(z)$ and $b_i(z)$ in the general algorithm can be expressed as follows:

$$\begin{bmatrix} a_k(z) \\ b_k(z) \end{bmatrix} = F_{k,1}(z) \begin{bmatrix} a_0(z) \\ b_0(z) \end{bmatrix} \tag{29}$$

For splitting the algorithm, we suppose that n is even, thus we have:

$$F_{n,1}(z) = F_{n,n/2+1}(z) F_{n/2,1}(z) \tag{30}$$

Then,

$$\begin{bmatrix} a_k(z) \\ b_k(z) \end{bmatrix} = F_{n,n/2+1}(z) \begin{bmatrix} a_{n/2}(z) \\ b_{n/2}(z) \end{bmatrix} \tag{31}$$

The second pass has been splitted in two parts, the first one being the computation of $a_{n/2}(z)$ and $b_{n/2}(z)$, while the second half provides $F_{n,n/2+1}(z)$.

The second pass depends on the result of the first pass, $F_i(z)$, i = 1, 2, ..., n, and the computation of these polynomial matrices should be also splitted. For this, we examine the expression of $u_i(z)$ and $v_i(z)$ for $i > n/2$:

$$\begin{aligned} \begin{bmatrix} u_i(z) \\ v_i(z) \end{bmatrix} &= F_{i,n/2+1}(z) F_{n/2,1}(z) \begin{bmatrix} u_0(z) \\ v_0(z) \end{bmatrix} \\ &= F_{i,n/2+1}(z) \begin{bmatrix} u_{n/2}(z) \\ v_{n/2}(z) \end{bmatrix} \end{aligned} \tag{32}$$

with

$$\begin{bmatrix} u_{n/2}(z) \\ v_{n/2}(z) \end{bmatrix} = F_{n/2,1}(z) \begin{bmatrix} u_0(z) \\ v_0(z) \end{bmatrix} \tag{33}$$

We note that the recursion is initialized at order n/2, by $u_{n/2}(z)$ and $v_{n/2}(z)$. The recursion here is of outgoing type: the orders of the polynomials $u_i(z)$ and $v_i(z)$ decrease when i increases. Therefore, the orders of $u_{n/2}(z)$ and $v_{n/2}(z)$ are lower than those of $u_0(z)$ and $v_0(z)$. As for $F_i(z)$, i = 1, 2, ..., n/2, it is easily seen that it does not depend on every coefficient of $u_0(z)$ and $v_0(z)$. In fact, the initial polynomials for the first half are of the same orders as that of $u_{n/2}(z)$ and $v_{n/2}(z)$, which are the initial polynomials of the second half.

Therefore, we have also splitted the second pass. The resulting algorithm is thus entirely splitted. It is summarized below:

THE SPLITTED GENERAL ALGORITHM

First half:

Initialization:

$$\begin{bmatrix} u'_0(z) \\ v'_0(z) \end{bmatrix} \leftarrow \begin{bmatrix} \text{a part of } u_0(z) \\ \text{a part of } v_0(z) \end{bmatrix}$$

$$F'_{0,1}(z) \leftarrow I$$

For i = 1, 2, ..., n/2

$F'_i(z)$ = function of $(u'_{i-1}(z), v'_{i-1}(z))$

$$\begin{bmatrix} u'_i(z) \\ v'_i(z) \end{bmatrix} = F'_i(z) \begin{bmatrix} u'_{i-1}(z) \\ v'_{i-1}(z) \end{bmatrix}$$

$$F'_{i,1}(z) = F'_i(z)\, F'_{i-1,1}(z)$$

Output: $F'_{n/2,1}(z)$

$$\begin{bmatrix} a_{n/2}(z) \\ b_{n/2}(z) \end{bmatrix} = F'_{n/2,1}(z) \begin{bmatrix} a_0(z) \\ b_0(z) \end{bmatrix}$$

Second half:

Initialisation:

$$\begin{bmatrix} u''_0(z) \\ v''_0(z) \end{bmatrix} \leftarrow F'_{n/2,1}(z) \begin{bmatrix} u_0(z) \\ v_0(z) \end{bmatrix}$$

$$F''_{0,1}(z) \leftarrow I$$

For i = 1, 2, ..., n/2

$F''_i(z)$ = function of $(u''_{i-1}(z), v''_{i-1}(z))$

$$\begin{bmatrix} u''_i(z) \\ v''_i(z) \end{bmatrix} = F''_i(z) \begin{bmatrix} u''_{i-1}(z) \\ v''_{i-1}(z) \end{bmatrix}$$

$$F''_{i,1}(z) = F''_i(z) F''_{i-1,1}(z)$$

Output: $F''_{n/2,1}(z)$

Combination:

$$\begin{bmatrix} a_n(z) \\ b_n(z) \end{bmatrix} = F''_{n/2,1}(z) \begin{bmatrix} a_{n/2}(z) \\ b_{n/2}(z) \end{bmatrix}$$

If we compare each half of the splitted algorithm with the initial algorithm, we remark that the latter computes directly the polynomials $a_i(z)$ and $b_i(z)$, while the former computes the polynomial matrices $F'_{i,1}(z)$ (or $F''_{i,1}(z)$), which are of dimensions 2x2 and contain 4 polynomials instead of 2. The computational complexity is therefore increased. However, this initial increase in the arithmetic complexity makes the application of the divide and conquer strategy feasible, which allows a large compensation for the additional computations.

The following figure shows the structure of the splitted algorithm:

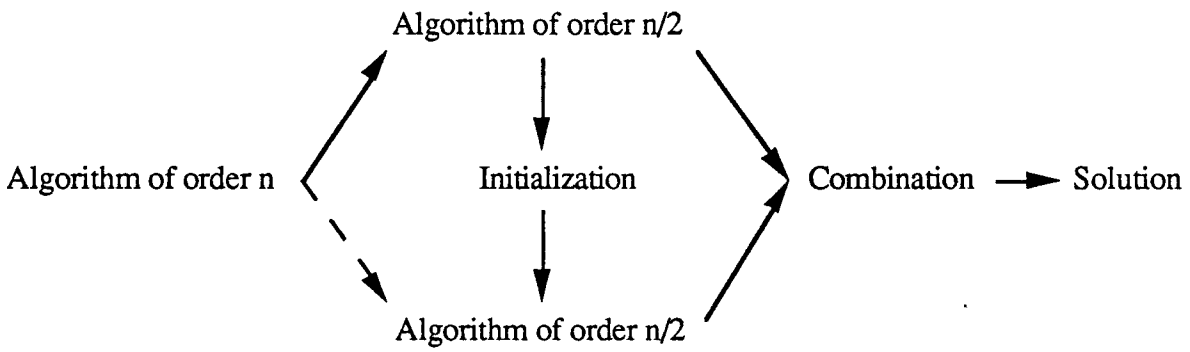


fig.2 organization of the divide and conquer algorithms

We remark that the splitted algorithm contains twice the same algorithms of half order, with some additional computations for the initialization of the second half and the combination of the partial results obtained in each half. The number of multiplications required for this algorithm, we note it M(n), follows the relationship below:

$$M(n) = 2\,M(n/2) + Ms(n) \qquad (34)$$

where Ms(n) is refered as the number of multiplications required for the additional computations (initialization and combination).

It is easily seen that the initialization of the second part and the final combination are in fact polynomial products. This type of operation is essentially a linear convolution. Therefore, we can use any fast convolution algorithm, for example, the FFT method [10,11], and we obtain the computational complexity of the splitted algorithm:

$$Ms(n) = C^{te}\, n\log_2 n \qquad (35)$$

If we continue this splitting operation for each half of the algorithm, (this is feasible, since the splitted algorithm preserves the same structure as that of the initial one), for $n = 2^v$, we obtain

$$M(n) = A\, n(\log_2 n)^2 \qquad (36)$$

with A a given constant.

We have thus succeeded in applying the doubling strategy to the general algorithm, and obtain a doubling algorithm which requires a number of multiplications proportional to $n(\log_2 n)^2$. Since all the classical algorithms belong to the class described in section 6-1 on which the divide and conquer strategy has been applied, this doubling algorithm represents exhaustively their associated doubling algorithms. A precise derivation of these doubling

algorithms is easily obtained by applying the strategy explained in section 6 on any of the two-pass algorithms, either computing both forward and backward predictors, or based on a three-term recurrence.

The case of the doubling Levinson/Schur algorithm has been explained with some detail in [1,2,3] by Ammar and Gragg, called "Generalized Schur Algorithm". In [27], we have developed the algorithm from the above general framework, and the resulted algorithm has got a more concise presentation, and it has a lower arithmetic complexity in small dimension cases. Since the above approach can be applied to any two-pass algorithm, it can also be applied to the split-Levinson algorithm which allows the computational complexity to be further divided by two when the Toeplitz matrix is symmetric.

7. CONCLUSION

In this paper, we have provided a general framework for the description of the classical algorithms solving Yule-Walker equations. We provide an exhaustive presentation of these algorithms in terms of Padé approximants, which includes some partial results already published. This presentation allows the obtention of new versions of these algorithms. The arithmetic complexity of all these algorithms is seen to be nearly equivalent inside each of the two classes we have distinguished, and in any case is proportional to N^2. They will nevertheless have different properties in terms of error accumulation.

This unified presentation allows to understand the common structure underlying these algorithms, and we could thus obtain a general description of the associated doubling algorithms, which require a number of multiplications proportional to $N \log_2^2 N$.

REFERENCES:

[1] G.S.Ammar and W.B.Gragg, "Implementation and Use of the Generalized Schur Algorithm", in Computational and Combinatirial Methods in Systems Theory, C.I.Byrnes and A.Lindquist, eds., North-Holland, Amsterdam, pp.265-280, 1986.

[2] G.S.Ammar and W.B.Gragg, "The Generalized Schur Algorithm for the Superfast Solution of Toeplitz Systems", in Rational Approximation and its Applications in Mathematics and Physics, J.Gilewicz, M.Pindor, and W.Siemaszko, eds., Lecture Notes in Matematics 1237, Spinger-Verlag, New York, Berlin, pp.315-330, 1987.

[3] G.S.Ammar and W.B.Gragg, "Superfast Solution of Real Positive Definite Toeplitz Systems", SIAM J. Matrix Anal. Appl., vol.9, No. 1, January 1988, pp. 61-76.

[4] E.R.Berlekamp, Algebraic Coding Theory. New York: McGraw-Hill, 1968.

[5] R.E.Blahut, Fast Algorithms for Digital Signal Processing, Addison-Wesley, Reading, MA 1985.

[6] R.P.Brent, F.G.Gustavson, and D.Y.Y.Yun, "Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximations", J.Algorithms, vol.1, pp.259-295, 1980.

[7] A.Bultheel and P.Dewilde, "On the Relation Between Pade Approximation Algorithms and Levinson/Schur Recursive Methods", Signal Processing: Theories and Applications, M. Kunt and F. de Coulon (editors), North-Holland Publishing Company, EURASIP, 1980.

[8] J.M.Delosme, Algorithms for Finite Schift-Rank Processes, Ph.D dissertation, Stanford University, 1982.

[9] P.Delsarte, Y.Genin, and Y.G.Kamp, " A Generalization of The Levinson Algorithm for Hermitian Toeplitz Matrices with Any Rank Profile", IEEE Trans. on ASSP, vol. ASSP-33, No. 4, August 1985.

[10] P.Duhamel, "Implementation of "Split-Radix" FFT Algorithms for Complex, Real, and Real-Symmetric Data, IEEE Trans. Acoust. Speech Signal Processing, 34 (1986), pp.285-295.

[11] P.Duhamel, M.Vetterli, "Improved Fourier and Hartley Transform Algorithms: Application to Cyclic Convolution of Real Data", IEEE Trans. on Acoust., Speech, and Signal Processing, vol. ASSP-35, No.6, June 1987.

[12] J.Durbin, "The Fitting of Time Series models", Rev. Inst. Int. de stat. vol. 28, pp. 233-244, 1960.

[13] G.Frobenius, Uber Relationem Zwischen den Naherungsbruchen von Potenzreihen, J. fur Math. 90 (1881), 1-17.

[14] W.B.Gragg, "The Padé Table and Its Relation to Certain Algorithms of Numerical Analysis", SIAM Rev. 14, No.1 pp.1-62, 1972.

[15] F.G.Gustavson and D.Y.Y.Yun, " Fast Algorithms for Rational Hermite Approximation and Solution of Toeplitz System", IEEE Trans. Circuits Syst., vol. CAS-26, pp.750-755, Sep.1979.

[16] F.De Hoog, "A New Algorithm for Solving Toeplitz Systems of Equations", Linear Algebra Appl., 88/89 (1987), pp.122-138.

[17] L.Kronecker, "Zur Theorie der Elimination einer Variabelen aus zwei algebraischen Gleichungen", Monastsb. Konigl. Preuss. Akad. Wiss. Berlin pp.735-600, 1881.

[18] N.Levinson, "The Wiener (Root Mean Square) error criterion in filter design and Prediction", J.Math. Phys., vol.25, pp. 261-278, 1947.

[19] J.L.Massey, "Shift-Register Synthesis and BCH Decoding", IEEE Trans. Inform. Theory, vol. IT-15, pp.122-127, jan. 1969.

[20] R.T.Moenck, "Fast Computation of GCDs", in Proc. 5th Annu. ACM Symp. Theory of computing, pp.142-151, 1973.

[21] H.Padé, "Sur La Representation Approchée d'une Fonction par des Fractions Rationnelles", Thesis, Ann. Ecole Nor. (3), 9, pp.1-93, suppl.

[22] Y.Sugiyama, M.Kasahara, S.Hirasawa, and T.Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes", Inform. Contr., vol. 27, pp.87-99, Jan. 1975.

[23] Y.Sugiyama, "An Algorithm for Estimating AR Coefficients Based upon Euclid Algorithm", in Proc. 1981 National. Conf. Information Science and Technology, Inst. Electron. Commun. Eng. of Japan, vol. 10, p.10. Oct.1981.

[24] R.P.Brent, "Old and New Algorithms for Toeplitz Systems", in NATO ASI 1988, Louvent, Belgiem.

[25] I.Gohberg, "I. Schur Methods in Operator Theory and Signal Processing", ed. by I. Goberg [Vol. ed. office Nathan and Lily Silver]: Basel, Boston, Stuttgart, Birkhäuser, 1986.

[26] S.Pombra, H.Lev-Ari, T.Kailath, "Levinson and Schur Algorithms for Toeplitz Matrices with Singular Minors", Processing of the ICASSP,1988.

[27] H.M. Zhang, P. Duhamel, "Doubling Levinson/Schur algorithm and its implementation" Proc. of the ICASSP 89, pp.1115-1118.