INRIA

CENTRE DE RENNES
IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France

Tél. (1) 39 63 55 11

Rapports de Recherche

N° 570

STABILIZATION OF FAST RECURSIVE LEAST-SQUARES TRANSVERSAL FILTERS

Jean-Luc BOTTO
Georges V. MOUSTAKIDES

Septembre 1986

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CÉDEX
FRANCE
Téléphone : 99 36 20 00
Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

STABILIZATION OF FAST RECURSIVE LEAST-SQUARES TRANSVERSAL FILTERS

STABILISATION DE L'ALGORITHME TRANSVERSAL RAPIDE DES MOINDRES CARRES

Jean-Luc Botto and George V. Moustakides

Publication Interne n° 306 - Août 1986
60 pages

Résumé.- Les propriétés numériques des algorithmes des moindres carrés rapides sont importantes, en raison du rôle que ces algorithmes jouent dans les traitements adaptatifs. Ce papier étudie la propagation des erreurs numériques dans les filtres transversaux rapides non normalisés, et montre l'origine des instabilités numériques et leur manifestation sous deux formes différentes. On montre que l'introduction d'un léger surcroît de redondance dans ces algorithmes permet de les stabiliser. La même méthode, appliquée à la forme normalisée du filtre transversal rapide, donne un algorithme numériquement stable et de complexité comparable à l'algorithme d'origine.

.../...

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L. A. 227)
UNIVERSITÉ DE RENNES 1 I. N. S. A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

STABILIZATION OF
FAST RECURSIVE LEAST-SQUARES TRANSVERSAL FILTERS

Jean-Luc Botto*

and

George V. Moustakides, Member, IEEE**

Abstract.- Numerical properties of fast recursive least-squares algorithms are important because of the use of these algorithms in various adaptive schemes. In this paper, we first investigate how numerical errors propagate in unnormalized fast transversal filter (FTF) algorithms and show how they create numerical instability, resulting in two different modes of divergence. We then try to correct these bad numerical properties by introducing some redundancy in the equations of these algorithms. Because the obtained algorithms are still unstable, but less unstable than the original algorithms, we use the redundancy to derive a stabilized FTF algorithm. The same method is also applied for the stabilization of normalized fast transversal filter (NFTF) algorithms. The stabilized NFTF algorithm has a computational cost comparable to the original unstable algorithm.

---

*Jean-Luc BOTTO is with TRT, 5, Avenue Réaumur, 92350 Le Plessis Robinson, France
He was with IRISA, Campus de Beaulieu, 35042 Rennes Cédex, France

**George Moustakides is with IRISA, Campus de Beaulieu, 35042 Rennes Cédex, France

I. INTRODUCTION

Recursive least-squares identification algorithms play an increasing role in many adaptive control and signal processing problems. Adaptive and computationally efficient versions of the recursive least-squares (RLS) algorithm |1| have recently been implemented under the form of fast transversal filters (FTF) |2| - |7|. These fast least-squares algorithms require a number of arithmetic operations proportional to the number N of parameters to be estimated, which is comparable to the suboptimal gradient type techniques. A useful classification of the different existing fast sequential least-squares algorithms can be found in |8|.

Unfortunately, all these FTF algorithms are known to exhibit an unstable behaviour and a sudden divergence due to the accumulation of round-off errors in finite precision computation. This numerical instability remains the main drawback of these algorithms and certainly limits their extensive use in adaptive control or signal processing applications. Therefore, a better understanding of their poor numerical properties and modifications to improve these algorithms are of great interest.

Several techniques have been proposed to overcome the problem of instability. In |6|, |9|, the algorithms are applied in spite of their numerical instability, in parallel with a detection procedure that tries to detect the divergence as soon as possible in order to reinitialize their internal variables before the estimated parameters become significantly

corrupted by the accumulation of numerical errors. The algorithm is then periodically restarted by using more or less complex procedures. However, the effect of such reinitializations is most of the time apparent (for example in the prediction error), especially when small exponential forgetting factors are used.

This is the reason why other stabilization techniques have been recently presented |10|, |11|. They can be understood as regularization techniques, used to stabilize least-squares algorithms (whatever their computational organization may be) when the exponential forgetting factor is not compatible with the characteristics of the input signal (leading to the computation of ill-conditionned autocorrelation matrices). However, these modified fast algorithms are always suboptimal in the sense that the identification gain that they recursively compute with time is no more equal to the theoretical least-squares gain, thus reducing the tracking capability of the algorithm. Moreover, since these methods do not try to handle the numerical instability resulting from the specific transversal structure of FTF algorithms, they fail to stabilize these algorithms for specific input sequences (such as white noise for example). This was confirmed in our simulations.

From a theoretical point of view, the numerical instability of the Fast Kalman algorithm |1|, |2| has been analytically pointed out in |12| for a first-order filter and a specific input signal.

The paper is organized as follows. In the first part, we explain a few surprising facts observed in simulations. For this purpose, the propagation of numerical errors in the FTF algorithm derived in |6| is investigated. In the

second part, we derive a stabilized FTF algorithm, which coincides with the theoretical algorithm when no round-off errors propagate, i.e. in the ideal case of infinite computational precision.

The FTF algorithm, derived in [6], is briefly recalled in section 2, as well as the notations we use. Section 3 is devoted to the numerical error propagation analysis of the fastest FTF algorithms which are shown to be unstable. The same analysis is also applied to these algorithms when computing the backward scalar residual in a non optimal fashion. The computational redundancy of these algorithms is then used in section 4 to modify and stabilize the FTF algorithms. We also present the normalized version of this stabilized FTF algorithm. From a computational point of view, it has roughly the same complexity as the unstable normalized fast transversal filter (NFTF) algorithm, when computing the normalized backward scalar residual in a non optimal fashion. As illustrated by simulation results given in section 5, these stabilized unnormalized and normalized FTF algorithms prove to be stable for various input sequences and exponential forgetting factors (no reinitialization is needed) when using floating point or fixed point arithmetic.

II. THE FTF ALGORITHMS

The notations used in this paper are similar to those in [6], except for the vectors which are now column vectors. The aim of finite impulse response identification algorithms is to determine the filter parameter vector $H_{N,T}$ which minimizes at each time T the cost :

$$V_N(T) = \sum_{k=0}^{T} \lambda^{T-k} \{y(k) - H_{N,T}^T X_N(k)\}^2 \qquad (2.1)$$

where $\lambda(0<\lambda<1)$ is the exponential forgetting factor, $y(T)$ is the signal to be linearly estimated, $X_N(T) = [x(T-1),\ldots,x(T-N)]^T$ regroups the N last samples of the input signal $x(T)$ and $H_{N,T} = [H_{N,T}^1,\ldots,H_{N,T}^N]^T$ is the filter coefficient vector. To obtain the known fastest FTF algorithms, we also assume (prewindowed signal case) :

$$y(T) = x(T-1) = 0 \quad \text{for } T \leqslant 0$$

The filter coefficient vector $H_{N,T}$ is recursively updated in time according to the classical relations [1] :

$$H_{N,T} = H_{N,T-1} - \varepsilon_N^P(T)\, C_{N,T}$$

where $$\varepsilon_N^P(T) = y(T) - H_{N,T-1}^T \quad X_N(T)$$

is the a-priori scalar residual at time T, and $C_{N,T}$ the (Nx1) Kalman gain vector defined as :

$$C_{N,T} = - R_{N,T}^{-1} X_N(T) \qquad (2.2)$$

where

$$R_{N,T} = \sum_{k=0}^{T} \lambda^{T-k} X_N(k)\, X_N(k)^T \qquad (2.3)$$

is the short-term autocorrelation matrix of the input sequence. Associated with $\varepsilon_N^P(T)$, we also define the a-posteriori scalar residual at time T :

$$\varepsilon_N(T) = y(T) - H_{N,T}^T X_N(T)$$

The FTF algorithms compute the Kalman gain vector $C_{N,T}$ (and also $H_{N,T}$) using 4 sets of variables. The first set which has been previously defined is related to the linear prediction of y(T) knowing the N values {x(T-1),...,x(T-N)}. The second (respectively third) set is related to the linear prediction of x(T-1) knowing its last N values {x(T-2),...,x(T-N-1)} (resp. x(T-N-1) knowing its N coming values {x(T-1),...,x(T-N)}). We denote it as the forward (resp. backward) variables set. As shown in |8|, the Kalman gain vector $C_{N,T}$ and its associated variables can be related to the linear prediction of some specific sequence y(T) knowing {x(T-1),...,x(T-N)}. Each set consists of the (Nx1) identified filter coefficient vector (such as $H_{N,T}$), the a-priori and a-posteriori scalar residuals (such as $\varepsilon_N^P(T)$ and $\varepsilon_N(T)$) and the corresponding energy residual (such as $V_N(T)$). These variables are summarized in Table 1. For each set of variables, we can derive their analytical expressions in the same way we did for the first set. For more details, the interested reader is referred to the references.

In order to save N multiplications per time step, the dual Kalman gain vector $\tilde{C}_{N,T}$ defined as :

$$\tilde{C}_{N,T} = -\frac{1}{\lambda} R_{N,T-1}^{-1} X_N(T) \qquad (2.4)$$

is recursively updated instead of the Kalman gain vector $C_{N,T}$. As an example, the FTF algorithm derived in |6| is summarized in Table 2 (where X^i denotes

the i^{th} component of the vector $X = [x^1,\ldots,x^i,\ldots]^T$). Because its computation requires about 7N multiplications per time step, we denote it the FTF (7N) algorithm (throughout this paper, the number of operations is only approximate ; nevertheless, the general dependency of computational complexity on N obtained by including only multiplications is sufficiently accurate to permit comparisons between the different FTF algorithms).

III. NUMERICAL ERROR PROPAGATION PROPERTIES OF FTF ALGORITHMS

The FTF algorithms are known to exhibit an unstable behaviour and a sudden divergence, due to round-off noise in finite precision implementation, whatever the input sequence x(T) and the number N of estimated parameters are. In fact, one shoud not speak of the "divergence" of the FTF algorithms (an algorithm is said to diverge when the difference between the theoretical and computed variables of the algorithm is increasing with time) since there are actually two different modes of divergence.

The first mode is well known and occurs when $\gamma_N(T)$ exceeds unity (its theoretical maximum value) shortly before $\theta_N(T)$ becomes negative and the coefficients of the $A_{N,T}$, $B_{N,T}$, $C_{N,T}$ and $H_{N,T}$ filters diverge to infinity |6|, |9|. It has immediate effects (when $\lambda < 1$) since the computed a priori scalar residual $\epsilon_N^P(T)$ also tends to infinity.

The second mode is less known and occurs when $\gamma_N(T)$ diverges towards zero together with the Kalman gain vector coefficients $C_{N,T}^i$ ($1 \leq i \leq N$). For this case, because the identification gain vector coefficients become very small, the tracking capability of the diverging algorithm is reduced and tends to zero ! In fixed-point arithmetic, the computed Kalman gain vector $C_{N,T}$ can even be exactly equal to the zero vector after some time |13|. This mode of divergence is more difficult to detect because no variable escapes from its theoretical domain of variation : it is just as if the algorithm is slowly "freezing". We give in section 5 an example where these two modes of divergence have been observed one after the other.

Let us now show that these divergences are due to the accumulation of numerical errors in finite precision arithmetic and explain why, for a given processor accuracy, one mode of divergence is more likely to occur than the other. Let ψ denote the theoretical value of a variable of the algorithm, $\bar{\psi}$ the corresponding computed value and $\Delta\psi$ the error between ψ and $\bar{\psi}$ such that $\bar{\psi} = \psi + \Delta\psi$. Let us consider the FTF (7N) algorithm of Table 2 and assume that errors are introduced at some step $T = T_0$ of the algorithm so that :

$$\left\{\begin{array}{l} \bar{\alpha}_N(T_0) = \alpha_N(T_0) + \Delta\alpha_N(T_0) \\ \bar{\gamma}_N(T_0) = \gamma_N(T_0) + \Delta\gamma_N(T_0) \\ \bar{\beta}_N(T_0) = \beta_N(T_0) + \Delta\beta_N(T_0) \\ \bar{A}_{N,T_0} = A_{N,T_0} + \Delta A_{N,T_0} \\ \bar{B}_{N,T_0} = B_{N,T_0} + \Delta B_{N,T_0} \\ \bar{\tilde{C}}_{N,T_0} = \tilde{C}_{N,T_0} + \Delta\tilde{C}_{N,T_0} \end{array}\right. \qquad (3.1)$$

Let us study how these errors propagate in the FTF(7N) algorithm for $T > T_0+1$, assuming an infinite precision computation.

Introducing the scalar variables :

$$\begin{cases} \delta_N(T) = \dfrac{\beta_N(T)}{\lambda\beta_N(T-1)} - 1 \gtrless 0 & (3.2) \\ \delta'_N(T) = 2\,\dfrac{\lambda\alpha_N(T-1)}{\alpha_N(T)} - 1 \lessgtr 1 & (3.3) \end{cases}$$

and following a first order error analysis, after some standard but tedious calculations (summarized in Appendix A), the error propagation equations can be written in a state-form as :

$$\Xi(T) = F(T)\,\Xi(T-1) + V_1(T) + V_2(T) \qquad (3.4)$$

where

$$\Xi(T) = \begin{bmatrix} \dfrac{\Delta\alpha_N(T)}{\alpha_N(T)} & \dfrac{\Delta\gamma_N(T)}{\gamma_N(T)} - \dfrac{\Delta\alpha_N(T)}{\alpha_N(T)} \end{bmatrix}^T \qquad (3.5)$$

$$F(T) = \begin{bmatrix} 1 & \dfrac{1}{2}\,(1-\delta'_N(T)) \\ 3\delta_N(T) & \delta'_N(T) + \dfrac{1}{2}\,\delta_N(T)\,(3+\delta'_N(T)) \end{bmatrix} \qquad (3.6)$$

$$V_1(T) = 2\delta_N(T)\,v_1(T) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{with} \quad v_1(T) = s_0 + \frac{\Delta\tilde{C}^{N+1}_{N+1,T}}{\tilde{C}^{N+1}_{N+1,T}} - \frac{e_N(T)}{\alpha_N(T)}\,\Delta e^P_N(T) \qquad (3.7)$$

$$V_2(T) = 2v_2(T) \begin{bmatrix} 1 \\ -2 \end{bmatrix} \qquad \text{with } v_2(T) = \frac{e_N(T)}{\alpha_N(T)}\,\Delta e^P_N(T) \qquad (3.8)$$

$$s_0 = \frac{1}{2} \left\{ \frac{\Delta\beta_N(T_0)}{\beta_N(T_0)} - \frac{\Delta\gamma_N(T_0)}{\gamma_N(T_0)} - \frac{\Delta\alpha_N(T_0)}{\alpha_N(T_0)} \right\} \qquad (3.9)$$

We first make the following hypothesis regarding the computed identified filter parameter vector $A_{N,T}$ and the $(N+1)^{th}$ component $\tilde{C}^{N+1}_{N+1,T}$ of the computed $(N+1)^{th}$ order dual Kalman gain vector $\tilde{C}_{N+1,T}$:

Hypothesis (H1) : there are no errors on $\overline{A_{N,T}}$ and $\overline{\tilde{C}^{N+1}_{N+1,T}}$ for $T \geqslant T_0$

This hypothesis implies for $T \geqslant T_0+1$:

$$\Delta e^P_N(T) = \Delta \tilde{C}^{N+1}_{N+1,T} = 0 \tag{3.10}$$

so that the error propagation equations (3.4) reduces to :

$$\Xi(T) = F(T)\ \Xi\ (T-1) + V_1(T) \tag{3.11}$$

where now $$V_1(T) = 2s_0\ \delta_N(T) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3.12}$$

Let us consider the trajectory of the state vector $\Xi(T)$ in the state space $\mathbb{R}^2$. In practice, we can always assume that the input sequence $x(T)$ satisfies :

$$\alpha_N(T) \leqslant (2\lambda)\ \alpha_N(T-1) \qquad \text{for} \quad T \geqslant T_0+1$$

and

there exists $\varepsilon > 0$ such that $\beta_N(T) \geqslant \lambda(1+\varepsilon)\ \beta_N(T-1)$ for $T \geqslant T_0+1$

As a consequence, the scalar variables $\delta_N(T)$ and $\delta'_N(T)$ defined in (3.2) and (3.3) satisfy :

$$0 < \varepsilon \leqslant \delta_N(T) \tag{3.13}$$

and $$0 \leqslant \delta'_N(T) \leqslant 1 \tag{3.14}$$

and the four entries of F(T), according to their definition (3.6), are positive real scalars (usually since $\alpha_N(T) \simeq \alpha_N(T-1)$ and $\beta_N(T) \simeq \beta_N(T-1)$, we have $\delta'_N(T) \simeq 1$ and $\delta_N(T) \simeq \varepsilon$).

If we now define two sectors S_p and S_n of $\mathbb{R}^2$ as :

$$S_p = \{\Xi = [u\ v]^T \ /\ u \geqslant 0 \text{ and } v \geqslant 0\}$$

$$S_n = \{\Xi = [u\ v]^T \ /\ u \leqslant 0 \text{ and } v \leqslant 0\}$$

and if we assume :

$$\Xi(T-1) \in S_p \text{ and } s_0 \geqslant 0 \tag{3.15}$$

then, according to (3.13), (3.14) and (3.11), (3.15) we have :

$$F(T)\ \Xi(T-1) \in S_p \text{ and } \Xi(T) \in S_p \tag{3.16}$$

If now $\|\cdot\|$ denotes the following norm :

$$\|\Xi\| = 5u^2 + 4uv + v^2 \text{ for any } \Xi = [u\ v]^T \in \mathbb{R}^2$$

then after some long but straightforward calculations, we can show that |13| :

$$\|\Xi(T)\|^2 \geqslant \{1 + \tfrac{1}{2}\,\delta_N(T)\}\ \|\Xi(T-1)\|^2 \text{ for } T \geqslant T_0 + 1$$

According to (3.13), we thus have :

$$\|\Xi(T)\|^2 \geqslant \{1 + \tfrac{1}{2}\,\varepsilon\}\ \|\Xi(T-1)\|^2 \text{ for } T \geqslant T_0 + 1 \tag{3.17}$$

Similarly, it can be shown that :

"if $\Xi(T-1) \in S_n$ and $s_0 < 0$, then $\Xi(T) \in S_n$ and (3.17) is also verified" (3.18)

This demonstration is sketched in Figure 1. From (3.16), (3.17) and (3.18), we conclude that the system in (3.11) is unstable. **The FTF (7N) algorithm is thus unstable with respect to numerical error propagation.** Moreover, the mode of divergence of the algorithm depends directly on the initial errors (3.1) More precisely, if at time T_0 :

- $\Xi(T_0) \in S_p$ and $s_0 > 0$, then $\Xi(T)$ diverges in the sector S_p i.e.

$$\overline{\gamma}_N(T) \to +\infty \quad , \quad \overline{\alpha}_N(T) \to +\infty$$

(in fact, $\overline{\gamma}_N(T)$ diverges to infinity until $\overline{\theta}_N(T)$ (cf. Table 2) becomes negative, which causes $\overline{\gamma}_N(T)$ to be also negative)

This is what we called the first mode of divergence.

- $\Xi(T_0) \in S_n$ and $s_0 < 0$, then $\Xi(T)$ diverges in the sector S_n i.e.

$$\overline{\gamma}_N(T) \to 0_+ \quad , \quad \overline{\alpha}_N(T) \to 0+$$

(since, according to the equations given in Table 2, the computed variables $\overline{\alpha}_N(T)$ and $\overline{\gamma}_N(T)$ are always positive scalars as long as $\overline{\theta}_N(T)$ does not become negative, which is the case here)

This is the second mode of divergence.

If the hypothesis (H1) is now extended to :

Hypothesis (H2) : $\Delta A_{N,T}$, $\tilde{\Delta C}^{N+1}_{N+1,T}$ are "small" with respect to s_0 and $\frac{\Delta\alpha_N(T)}{\alpha_N(T)}$ for each time $T \gg T_0$

then, it can be shown |13| that the previous result is still valid. Let us simply note that (H2) is satisfied in practice : if at time T_d, $\overline{\gamma}_N(T_d) > 1$ (or $\overline{\theta}_N(T_d) < 0$), i.e. $\Delta\gamma_N(T_d) \gg 0$, it is well known that the computed identified filter parameter vectors $\overline{A}_{N,T_d}$ and $\overline{H}_{N,T_d}$ are not significantly corrupted by round-off noise (with respect to $\overline{\gamma}_N(T_d)$, $\overline{\alpha}_N(T_d)$), as is illustrated by the reinitialization procedures taking $\overline{H}_{N,T_d-1}$ as initial condition |6|, |9|.

The results presented above have been verified in simulation. The two modes of divergence have been observed corresponding to round-off errors on $\alpha_N(T_0)$ and $\beta_N(T_0)$ that are positive ($\Xi(T_0) \in S_p$, $s_0 > 0$) or negative ($\Xi(T_0) \in S_n$, $s_0 < 0$) |13|. We have thus shown that the FTF(7N) algorithm of Table 2 was unstable with respect to numerical errors, the mode of divergence depending on the values of these errors. In fact, the same analysis applied to the FAEST(7N) algorithm |4|, |5| (which computes $\gamma_N(T)^{-1}$ instead of $\gamma_N(T)$) shows that this algorithm has **exactly** the same numerical properties as the FTF(7N) algorithm |13|.

The previous results (in particular, the inequality (3.17)) are true whether or not $\alpha_N(T)$ is assumed to be computed with a finite or an infinite precision (i.e. $\Delta\alpha_N(T) = 0$ for $T > T_0$). Thus a natural way to try to improve the robustness of the FTF algorithms is to try to compute the two energy residuals $\gamma_N(T)$ and $\beta_N(T)$ in a better way. However, the computation of $\beta_N(T)$ considered alone is much more unstable than the computation of $\gamma_N(T)$. This is the reason why the relation of Table 2 :

$$r_N^P(T) = -\lambda\, \beta_N(T-1)\, \tilde{C}_{N+1,T}^{N+1} \tag{3.19}$$

is replaced by the N-terms convolution :

$$r_N^P(T) = x(T-N-1) - B_{N,T-1}^{T}\; X_N(T) \tag{3.20}$$

thus avoiding the computation of $\beta_N(T)$. This algorithm is now denoted as the FTF(7N+N) algorithm since (3.20) now requires N additional multiplications per time step. In fact, when the architecture of the signal processor is optimized for convolution, it is much more easier (and sometimes faster !) to compute (3.20) than (3.19). Unfortunately, the same analysis as before shows that this FTF(7N+N) algorithm is still numerically unstable, but less unstable than before (the key characteristics of the F(T) matrix in (3.4) are not modified). This appears clearly on simulations.

This is the reason why we propose, in the next section, a new method to stabilize both normalized and unnormalized FTF algorithms in order to make them suitable for practical implementation and contribute to the extension of their application fields.

IV. STABILIZATION OF FTF ALGORITHMS

It is widely known that two independent facts can cause the instability of any exponentially windowed least-squares identification algorithm |6|, |11|. First, if the exponential forgetting factor value λ is not compatible with the input sequence characteristics (possibly time-varying) then the algorithm can be unstable, independently of the way it is implemented (recursive least-squares, FTF algorithms, ladder algorithms...). Sufficient conditions for a safe choice of the exponential forgetting factor value have been estimated in |11|. A second fact is that for some least-squares algorithms, round-off errors accumulate until the algorithm diverges. As we have shown in the previous section, this is the case for the FTF algorithms but it is not true for the recursive least-squares algorithm, if it is correctly implemented |10|, or for least-squares ladder algorithms |12|. These two facts prevent the wide use of FTF algorithms. However, we have to distinguish the first fact which is common to all exponentially windowed least-squared identification algorithms and which can be overcomed by taking a better value for the exponential forgetting factor λ from the second (which is the major drawback of FTF algorithms only). Gur goal here is to resolve the numerical instability of the exponentially windowed FTF algorithms, assuming implicitly that the exponential forgetting factor value is compatible with the characteristics of the input sequence.

For the sake of simplicity, we shall only consider the case of the FTF(7N+N) algorithm discussed previously. However, it is quite obvious that

our method of stabilization can be easily extended to any FTF algorithm in which the backward a-priori scalar residual $r_N^P(T)$ is computed according to the convolution (3.20) of the backward transversal filter $B_{N,T}$ with the input sequence $x(T)$.

Indeed, the stabilization of any FTF algorithm cannot be efficient when there is no way to "measure" the divergence of the computed algorithm (with respect to its theoretical behaviour), allowing with some adequate feed-back to correct the erroneous variables. Now, if we add the computation of the backward energy residual $\beta_N(T)$ to the previous FTF(7N+N) algorithm, we introduce some redundancy in the algorithm. We can then compute, the "control variable" $\xi_N(T)$ defined as :

$$\bar{\xi}_N(T) = \bar{r}_N^P(T) + \lambda\, \bar{\beta}_N(T-1)\, \bar{\tilde{c}}_{N+1,T}^{N+1} \tag{4.1}$$

i.e.

$$\bar{\xi}_N(T) = \{x(T-N-1) - \bar{B}_{N,T-1}^T\, X_N(T)\} + \lambda\bar{\beta}_N(T-1)\, \{\bar{\tilde{C}}_{N,T-1}^N + \frac{\bar{e}_N^P(T)}{\lambda\bar{\alpha}_N(T-1)}\, \bar{A}_{N,T-1}^N\}$$

$$= f(\bar{A}_{N,T-1}\ ;\ \bar{B}_{N,T-1}) \tag{4.2}$$

from the equations of Table 2. As before, $\bar{\psi}$ denote the computed variable corresponding to the theoretical variable ψ. According to (3.19) and (3.20) this control variable $\bar{\xi}_N(T)$ is exactly equal to zero when the equations of the algorithm are propagated with an infinite precision ($\xi_N(T) = 0$). In fact, computer simulations show that $|\bar{\xi}_N(T)|$ grows exponentially with time. Instead of using $\bar{\xi}_N(T)$ in a passive way to detect the divergence of the algorithm ($\bar{\xi}_N(T)$ acts sooner than $\bar{\gamma}_N(T)$), we propose now to use this control

variable to correct the other computed variables and prevent any divergence. Having computed, at any time T, a control variable $\bar{\xi}_N(T)$ different from zero, there are multiple ways to correct the variables of the algorithm. This is the reason why we need to state clearly the modification brought to the original FTF(7N+N) algorithm. This modification can be viewed as a way of implementing the following procedure :

At each time T, **given** the previously computed filter parameter vectors $\bar{A}_{N,T-1}$, $\bar{B}_{N,T-1}$, $\bar{C}_{N,T-1}$ and their corresponding energy residuals $\bar{\alpha}_N(T-1)$, $\bar{\beta}_N(T-1)$, $\bar{\gamma}_N(T-1)$ and

having computed the control variable $\bar{\xi}_N(T)$ (non equal to zero because of round-off errors),

replace $\{\bar{A}_{N,T-1};\ \bar{e}_N^P(T)\}$ and $\{\bar{B}_{N,T-1};\ \bar{r}_N^P(T)\}$ respectively by

$\{A'_{N,T-1};\ e_N^{P'}(T)\}$ and $\{B'_{N,T-1};\ r_N^{P'}(T)\}$ where :

$$\begin{cases} e_N^{P'}(T) = x(T-1) - A'^{T}_{N,T-1}\, X_N(T-1) & (4.3) \\ r_N^{P'}(T) = x(T-N-1) - B'^{T}_{N,T-1}\, X_N(T) & (4.4) \end{cases}$$

and where the (Nx1) transversal filters $A'_{N,T-1}$ and $B'_{N,T-1}$ are obtained by minimizing the quadratic error criterion :

$$\begin{aligned} W_N(T) = {} & \lambda(A'_{N,T-1} - \bar{A}_{N,T-1})^T R_{N,T-2} (A'_{N,T-1} - \bar{A}_{N,T-1}) \\ & + \lambda(B'_{N,T-1} - \bar{B}_{N,T-1})^T R_{N,T-1} (B'_{N,T-1} - \bar{B}_{N,T-1}) \\ & + \rho\, \xi'_N(T)^2 \end{aligned} \qquad (4.5)$$

where ρ is a given strictly positive scalar and $\xi'_N(T)$ is defined as the control variable $\bar{\xi}_N(T)$ except it depends on the modified filter parameter vectors $A'_{N,T-1}$, $B'_{N,T-1}$:

$$\xi'_N(T) = f(A'_{N,T-1};\ B'_{N,T-1}) \tag{4.6}$$

where the function $f(A;B)$ has been defined in (4.2).

Since the theoretical autocorrelation matrix $R_{N,T}$ is positive definite, minimizing the cost $W_N(T)$ with respect to $A'_{N,T-1}$ and $B'_{N,T-1}$ can also be viewed as minimizing a norm between $A'_{N,T-1}$ and $\bar{A}_{N,T-1}$ on one side, $B'_{N,T-1}$ and $\bar{B}_{N,T-1}$ on the other side, but penalizing also this minimization by the variable $\xi'_N(T)$ which we like to be small. Speaking in a different way, the modified transversal filters $A'_{N,T-1}$ and $B'_{N,T-1}$ have to minimize the control variable $\xi'_N(T)$, but staying also close to the initially computed transversal filters $\bar{A}_{N,T-1}$ and $\bar{B}_{N,T-1}$ respectively, the closeness being relative to the input $x(T)$. According to (4.6), we verify that, in the ideal case where the algorithm is propagated with an infinite precision, the modified procedure stated above does not change the algorithm since $A'_{N,T-1} = \bar{A}_{N,T-1}$, $B'_{N,T-1} = \bar{B}_{N,T-1}$ and $\xi'_N(T) = \bar{\xi}_N(T) = \xi_N(T) = 0$.

If we now define the scalar variable :

$$\bar{k}_N(T-1) = \frac{\bar{\beta}_N(T-1)}{\bar{\alpha}_N(T-1)} \tag{4.7}$$

then, the function f(A;B) defined in (4.2) can be written as :

$$f(A;B) = \{x(T-N-1) - B^T X_N(T)\} + \lambda\bar{\beta}_N(T-1)\,\bar{\tilde{C}}^N_{N,T-1} + \bar{k}_N(T-1)A^N\{x(T-1)-A^T X_N(T-1)\} \tag{4.8}$$

Since f(A;B) is linear with respect to B but non-linear with respect to A, we next linearize f(A;B) with respect to A, around $A=\bar{A}_{N,T-1}$. The linearized approximation of f(A;B) takes the form :

$$f(A;B)\simeq\{x(T-N-1)-B^T X_N(T)\}+\lambda\bar{\beta}_N(T-1)\bar{\tilde{C}}^N_{N,T-1}+\bar{k}_N(T-1)A^{N-P}\bar{e}^P_N(T)$$
$$-\bar{k}_N(T-1)\bar{A}^N_{N,T-1}(A-\bar{A}_{N,T-1})^T X_N(T-1) \tag{4.9}$$

Using this equality for expressing $\xi'_N(T)$ in the cost $W_N(T)$, we can easily express the minimizing filters $A'_{N,T-1}$ and $B'_{N,T-1}$ in terms of $\bar{A}_{N,T-1}$, $\bar{B}_{N,T-1}$ and some other (theoretical) variables [13] :

$$\begin{cases} B'_{N,T-1} = \bar{B}_{N,T-1} - \rho\,\xi'_N(T)\,\tilde{C}_{N,T} & (4.10) \\ A'_{N,T-1} = \bar{A}_{N,T-1} - \rho\,\bar{k}_N(T-1)\,\bar{A}^N_{N,T-1}\,\xi'_N(T)\,\tilde{C}_{N,T-1} - \rho\,\bar{k}_N(T-1)\,\xi'_N(T)\bar{e}^P_N(T)d_{N,T-2} & (4.11) \end{cases}$$

where :

$$d_{N,T} = \frac{1}{\lambda}\,R^{-1}_{N,T}\,[0\ \dots\ 0\ 1]^T \tag{4.12}$$

Transposed and convolved with the input sequence x(T) and according to the definitions (4.3) and (4.4), the relations (4.10) and (4.11) give :

$$\begin{cases} r^{P\,\prime}_N(T) = \bar{r}^P_N(T) - \rho\left(\frac{1}{\gamma_N(T)} - 1\right)\xi'_N(T) & (4.13) \\ e^{P\,\prime}_N(T) = (1-\rho\bar{k}_N(T-1)\tilde{C}^N_{N,T-1}\xi'_N(T))\bar{e}^P_N(T)-\rho\bar{k}_N(T-1)\bar{A}^N_{N,T-1}\left(\frac{1}{\gamma_N(T-1)} - 1\right)\xi'_N(T) & (4.14) \end{cases}$$

Since the only "measured" variable is $\bar{\xi}_N(T)$, while $\xi'_N(T)$ is unknown, we have to use relations (4.13) and (4.14) to express $\xi'_N(T)$, given in (4.6) and (4.8), in terms of $\bar{\xi}_N(T)$:

$$\xi'_N(T) = \{1 + \rho(\frac{1}{\gamma_N(T)} - 1) + \rho \bar{k}_N(T-1)^2 \bar{A}^{N^2}_{N,T-1}(\frac{1}{\gamma_N(T-1)} - 1)$$

$$+ 2\rho\, \bar{A}^N_{N,T-1}\, \bar{k}_N(T-1)^2\, \tilde{C}^N_{N,T-1}\, \bar{e}^P_N(T)$$

$$+ \rho\, \bar{k}_N(T-1)^2\, d^N_{N,T-2}\, \bar{e}^P_N(T)^2\}^{-1}\, \bar{\xi}_N(T) \qquad (4.15)$$

In fact, it can be shown that the terms including $d_{N,T}$ in relations (4.11) and (4.15) can be neglected, without any harm to the algorithm. For the theoretical variables $\gamma_N(T)$, $\gamma_N(T-1)$, $\tilde{C}_{N,T}$ and $\tilde{C}_{N,T-1}$ used in the previous relations, we now assume that the variables of the stabilized algorithm, once corrected using the modified procedure stated above, are a good approximation of the corresponding theoretical variables. Thus, the theoretical variables at time (T-1) can be approximated according to :

$$\tilde{C}_{N,T-1} \simeq \bar{\tilde{C}}_{N,T-1} \qquad (4.16)$$

and

$$\gamma_N(T-1) \simeq \bar{\gamma}_N(T-1) \qquad (4.17)$$

In order to save one division per time step, we can also note that :

$$\bar{k}_N(T-1) = \frac{\bar{\beta}_N(T-1)}{\bar{\alpha}_N(T-1)} \simeq \frac{\beta_N(T-1)}{\alpha_N(T-1)} = \lambda^{-N}\gamma_N(T-1) \simeq \lambda^{-N}\bar{\gamma}_N(T-1)$$

If now $\tilde{C}'_{N+1,T}$ denotes the $(N+1)^{th}$ order dual Kalman gain vector computed from $A'_{N,T-1}$, $e_N^{P'}(T)$ and $\bar{\tilde{C}}_{N,T-1}$:

$$\tilde{C}'_{N+1,T} = \begin{bmatrix} 0 \\ \bar{\tilde{C}}_{N,T-1} \end{bmatrix} - \frac{e_N^{P'}(T)}{\lambda \bar{\alpha}_N(T-1)} \begin{bmatrix} 1 \\ -A'_{N,T-1} \end{bmatrix} \tag{4.18}$$

then, assuming $A'_{N,T-1}$ and $B'_{N,T-1}$ close to their theoretical values $A_{N,T-1}$ and $B_{N,T-1}$, we obtain :

$$\begin{bmatrix} \tilde{C}_{N,T} \\ 0 \end{bmatrix} = \tilde{C}'_{N+1,T} - \tilde{C}^{N+1}_{N+1,T} \begin{bmatrix} -B_{N,T-1} \\ 1 \end{bmatrix}$$

$$\simeq \tilde{C}'_{N+1,T} - \tilde{C}^{N+1'}_{N+1,T} \begin{bmatrix} -B'_{N,T-1} \\ 1 \end{bmatrix} \tag{4.19}$$

Substituting this expression of $\tilde{C}_{N,T}$ in (4.10), we obtain :

$$\begin{bmatrix} B'_{N,T-1} \\ -1 \end{bmatrix} = \frac{1}{1+\rho \tilde{C}^{N+1'}_{N+1,T}\ \xi'_N(T)} \left\{ \begin{bmatrix} \bar{B}_{N,T-1} \\ -1 \end{bmatrix} - \rho\ \xi'_N(T)\ \tilde{C}'_{N+1,T} \right\} \tag{4.20}$$

The theoretical energy residual $\gamma_N(T)$ can also be approximated in terms of $\bar{\gamma}_N(T)$ and $\xi'_N(T)$ using (4.18). However, in practice the same results can be obtained when it is simply assumed that :

$$\gamma_N(T) \simeq \bar{\gamma}_N(T) \tag{4.21}$$

We have summarized in Table 3 the complete set of equations for the stabilized FTF algorithm, in which the transversal filters $A_{N,T-1}$ and $B_{N,T-1}$ and their corresponding a-priori scalar residuals are corrected at each time T, in relation to the numerical errors measured through the control variable $\overline{\xi}_N(T)$. For notational simplicity, all computed variables are denoted as the corresponding theoretical variables.

Computer simulations have clearly shown that the correction of $\overline{A}_{N,T-1}$ does not contribute much to the stabilization mechanism : the algorithm is numerically stable whether we use $A'_{N,T-1}$ or $\overline{A}_{N,T-1}$. This confirms the experimentally known fact that the computation of $A_{N,T-1}$ is numerically stable while the one of $B_{N,T-1}$ is unstable. Hence, the relation (4.11) can be skipped, thus reducing by N the number of additional multiplications required at each time step. With respect to the classical FTF(7N+N) algorithm, the additional complexity essentially comes from the relation (4.20) giving $B'_{N,T-1}$ in terms of $\overline{B}_{N,T-1}$ and $\tilde{C}_{N+1,T}$, requiring (2N) multiplications per time step. This is the reason why the algorithm of Table 3 is denoted as the stabilized FTF(7N+N+2N) algorithm. Its complexity is comparable to the one of the Fast Kalman algorithm |1|, |3|, while its robustness with respect to numerical errors is much better (cf. section 5).

The theoretical analysis of the numerical error propagation in the stabilized FTF(7N+N+2N) algorithm seems unfortunately quite complicated. Fortunately, this theoretical analysis is not absolutely necessary since the computer simulations clearly show this algorithm is numerically stable (cf. section 5).

In the same way, one might think that this modified algorithm is suboptimal. We have tried to address this problem using computer simulations by comparing at each time T, the dual Kalman gain vectors $\tilde{C}_{N,T}$ computed on one side using the definition (2.4) (the autocorrelation matrix $R_{N,T}$ being computed recursively in time according to the classical recursive least-squares algorithm |1|) and, on the other side using the stabilized FTF(7N+N+2N) algorithm summarized in Table 3. These simulations show that these two computed identification gain vectors are **identical** (modulo of course the computer accuracy, i.e. 10^{-7}) for each time T and different types of input sequences. The algorithm of Table 3 is thus both numerically stable and optimal in the least-squares sense.

Before giving some simulation results showing this algorithm is robust with respect to numerical errors, we now present briefly the normalized version of the stabilized FTF(7N+N+2N) algorithm. The numerical properties of NFTF algorithms are comparable to the properties of the unnormalized algorithms |6|. They are thus unstable with respect to round-off errors and it is interesting to apply to the normalized case the stabilization technique used before. Indeed, the simplest way to obtain a numerically stable NFTF algorithm is to "normalize" the relations previously presented, using the derivation of |6|. It can be shown |13| that the relations (4.10) to (4.14) can be included into the NFTF (11N+N) algorithm derived in |6| (in which the backward a-posteriori normalized scalar residual $\hat{r}_N(T)$ is computed using the normalized version of the convolution (3.20)) without increasing significantly the computational cost of the stabilized algorithm. The reason is that it is possible to bypass the computation of the corrected transversal filters $A'_{N,T-1}$ and $B'_{N,T-1}$ by directly expressing $\hat{A}_{N,T}$ and $\hat{B}_{N,T}$, i.e. the normalized versions of the $(N+1)^{th}$ order vectors $\begin{bmatrix}1 & -A_{N,T}\end{bmatrix}^T$ and $\begin{bmatrix}-B_{N,T} & 1\end{bmatrix}^T$, in terms of $\hat{A}_{N,T-1}$ and $\hat{B}_{N,T-1}$ respec-

tively, modifying only some multiplicative coefficients in the usual relations. The stabilized NFTF(11N+N) algorithm, derived in |13|, is summarized in Table 4. All computed variables take their values in $[-1, +1]$. Since the computations of $\hat{A}_{N,T}$, $\hat{B}_{N,T}$, $\hat{C}_{N,T}$ and $\hat{C}_{N+1,T}$ are not basically modified, the additional computational cost due to the stabilization procedure is **independent of the number N of estimated parameters**, which is a very interesting feature, and is roughly equal to twenty multiplications per time step for the computation of the control variable $\bar{\varepsilon}_N(T)$. This is the reason why this stabilized algorithm, like the original unstable version (obtained by taking $\rho = 0$ in the equations of Table 4) is denoted as the stabilized NFTF(11N+N) algorithm.

We now briefly comment in the next section some of the results obtained in simulating the normalized and unnormalized stabilized algorithms of Tables 3 and 4.

V. SIMULATIONS

In this section, computer simulations are presented in order to verify the results previously stated. These simulations have been conducted using 1977-FORTRAN with 32-bit single precision, floating-point arithmetic on a VAX 11/750 minicomputer system under a VMS operating system. These results are particularly interesting when the stabilized FTF (or NFTF) algorithm is compared to the classical FTF (or NFTF) algorithms.

The parameter μ used in these simulations is the one defined in |6| to initialize the FTF algorithms with arbitrary initial conditions (here equal to zero) so that :

$$\alpha_N(T = 0) = \mu \lambda^N$$

The integer W also denotes the "observation window" of the algorithm and is chosen as a multiple of the number N of estimated parameters, so that the exponential forgetting factor value λ computed according to :

$$\lambda = 1 - \frac{1}{W}$$

is then related to the number N.

As is noted in section 3, we now give an example showing both modes of divergence. The FTF(8N) algorithm given in |6| (it is very similar to the FTF(7N) algorithm summarized in Table 2, except it recursively computes the Kalman gain vector $C_{N,T}$ instead of the dual Kalman gain vector $\tilde{C}_{N,T}$) has been simulated for N = 10, W = 10N, μ = 0.1 and a centered white noise with unit variance as input sequence x(T).

Let us point out that these selected parameters, as well as the input signal x(T), can be considered as "favourable" since the exponential forgetting factor value λ is, with respect to N, close to unity (since W is much larger than N) and the input signal stationary, with the identity as autocorrelation matrix. In spite of these conditions, we can see on Figure 2, where we have plotted the value of the energy residual $\gamma_N(T)$ versus time T, that $\gamma_N(T)$ becomes strictly greater than its theoretical maximum value, i.e. unity, three times between T=1 and T=4000 (each one of these times is followed by the reinitialization procedure presented in |6|), and then suddenly diverges definitely to zero, along with the energy residuals $\alpha_N(T)$ and $\beta_N(T)$ (whose theoretical asymptotic values are equal to $(1-\lambda)^{-1}=100$). For this second mode of divergence, we have not tried to reinitialize the algorithm. With respect to the theoretical analysis of section 3, we have to note that, since $\alpha_N(T)$ and $\beta_N(T)$ are here computed in such a way that the state vector $\Xi(T)$ does not belong to S_p nor to S_n (the numerical errors on $\alpha_N(T)$ and $\beta_N(T)$ are of opposite signs at each time T), the sequential occurence of the two modes of divergence of the algorithm shows the importance of "local" conditions, i.e. the value of $\alpha_N(T)$, $\gamma_N(T)$... at time T, in such a case.

As is noted in section 3, the substitution of the convolution (3.20) to equation (3.19) for the computation of the backward a-priori scalar residual $r_N^P(T)$ greatly improves the robustness of the FTF algorithms with respect to round-off errors, without however making them numerically stable. We have simulated most of the known FTF algorithms, in their initial versions as well as in their modified versions (denoted FTF(-+N)) where $r_N^P(T)$ is computed according to (3.20). We give in Table 5 the average number of sampling steps between two consecutive divergences of these algorithms, for various exponential forgetting factor values λ, N=10 and a centered white noise with unit

variance as input signal x(T). Each algorithm is reinitialized when $\gamma_N(T)$ first exceeds unity. We apply the qualification "very unstable" to algorithms for which the average number of sampling steps between two consecutive divergences is of the same order of magnitude as the number N of estimated parameters. It is interesting to observe that even though the Fast Kalman (10N) algorithm |1|, |3| has a higher computational cost, it does not have a better robustness with respect to round-off errors.

We now compare the stabilized FTF(7N+N+2N) algorithm of Table 3 with the classical FTF(7N) and FTF(7N+N) algorithms, for N=500, W=3N, μ=10 and a centered white noise with unit variance as input signal x(T). We point out that the exponential forgetting factor value λ is, with respect to N, very small and, in particular, very inferior to the minimal values estimated in |11|. Observing the values of $\gamma_N(T)$ computed by the FTF(7N) algorithm |6| and plotted Figure (3.a), we see that this algorithm diverges very quickly, even before having reached its stationary state. As a consequence, it cannot be used in practice with such a value of λ. In Figure (3.b), we give the values of $\xi_N(T)$ and $\gamma_N(T)$ computed by the FTF(7N+N) algorithm. This figure gives a typical example of the exponential growth of $|\xi_N(T)|$ which is only used here to reinitialize the algorithm at the first time where $|\xi_N(T)| > 1$, i.e. before the estimated parameters $H_{N,T}$ are too significantly corrupted by round-off errors. The comparison of Figures (3.a) and (3.b) clearly demonstrates the spectacular improvement of the average number of sampling steps between two consecutive reinitializations when the FTF(7N+N) algorithm is implemented. The Figure (3.c) is similar to the Figure (3.b) except it has been obtained for the stabilized FTF(7N+N+2N) algorithm derived in section 4. We note again that the forward transversal filter $A_{N,T}$ is not corrected (i.e. $A'_{N,T} = A_{N,T}$) so that the computational cost of this stabilized algorithm is roughly similar

to the one of the Fast Kalman algorithm |1|, |3|. The parameter ρ, weighting $\xi'_N(T)^2$ in the quadratic cost $W_N(T)$ defined in (4.5), is equal to unity. In fact, this value has no importance provided it is not equal (or close) to zero (we obtain for $\rho = 0$ the previously simulated FTF(7N+N) algorithm). Figure (3.c) clearly demonstrates that this stabilized FTF(7N+N+2N) algorithm is numerically stable : the energy residual $\gamma_N(T)$ does not diverge to zero or infinity while the values of $\xi_N(T)$, plotted in the same scale as in Figure (3.b), cannot be distinguished from zero. More precisely, $|\xi_N(T)|$ is close to 10^{-5} at the beginning of the simulation (N+1 < T < 10000) and then decreases to 10^{-6}, 10^{-7}, i.e. to the accuracy of the computer. This phenomenon shows the stabilization mechanism of the stabilized FTF(7N+N+2N) algorithm. Since the input signal x(T) is in fact periodic, the period being 10000 samples of a white gaussian noise signal, the periodicity of $\gamma_N(T)$ observed on Figure (3.c) shows that the algorithm works well from the beginning to the end (the simulation has been conducted up to $T=10^6$ without any problem).

These conclusions remain valid when N is small (N=10, λ=0.95) and for stationary (white noise, autoregressive process...) or non-stationary (speech sequences) input signals |13|. We have also the same results when simulating the NFTF(11N), unstabilized (ρ=0) or stabilized ($\rho \neq 0$) NFTF(11N+N) algorithms in floating-point arithmetic |13|.

Finally, we would like to give one example of how the stabilized NFTF (11N+N) algorithm works in fixed-point arithmetic with a 16-bit wordlength. These normalized algorithms have been simulated without any particular care, following simply the equations of Table 4 with ρ=0 (original NFTF(11N+N) algorithm |6|) or ρ=1 (stabilized NFTF (11N+N) algorithm). Each computed variable takes its values in $[-1 \; +1[$ and the simulated accuracy is close

to 10^{-5}. The simulations have been done for N=10, W=2N (λ=0.95!), μ=1 and a centered white noise of variance 0.9 as input signal. We have plotted in Figures (4.a) and (4.b) the values of the normalized control variable $\hat{\xi}_N(T)$ |13|, and of $\gamma_N(T)^{1/2}$ for the original NFTF(11N+N) algorithm and the stabilized NFTF(11N+N) algorithm respectively. Figure (4.a) confirms an already known fact |14| : the NFTF(11N+N) algorithm does not work when the exponential forgetting factor value λ is strictly less than unity for a coding dynamic of 16 bits in fixed-point arithmetic. Indeed, all variables diverge towards zero, as well as $\hat{\xi}_N(T)$. In particular, the normalized Kalman gain vector $\hat{C}_{N,T}$ also diverges towards the zero vector (for an accuracy close to 10^{-4}, $\hat{C}_{N,T}$ is exactly the zero vector after some time !). As a consequence, this algorithm has approximately the same properties as a well-known suboptimal gradient type algorithm. On the contrary, it appears on Figure (4.b) (where T is now varying from 1 to 5000 000) that the stabilized NFTF(11N+N) algorithm summarized in Table 4 works well. The same results have also be obtained with speech signals |13|.

VI. CONCLUSION

We have first presented a simple investigation of the numerical instability of the FTF algorithms, which allows us to explain the two modes of divergence observed in simulations. Then we have derived a new stabilization method, whose theoretical basis is very simple : a control variable is generated by introducing an adequate redundancy in the algorithm, and is used to correct, at each time step, the computed variables when its non-zero value indicates the divergence of the algorithm. As we saw in the simulations, this technique can be applied to normalized and unnormalized FTF algorithms and make them numerically stable, without modifying their optimality. The additional computational cost to be paid for the stabilized algorithms is so small (especially in the normalized case where it is independent of the number N of estimated parameters) that these stabilized algorithms are attractive. Overally, the results given here will help to extend the application fields of FTF algorithms.

APPENDIX A

DERIVATION OF THE ERROR PROPAGATION EQUATIONS FOR THE FTF(7N) ALGORITHM

In this Appendix, the error propagation equations (3.4), (3.9) are derived, following a first order error analysis from the equations of the FTF(7N) algorithm summarized in Table 2.

Introducing :

$$0 < \theta'_N(T) = \frac{\lambda \alpha_N(T-1)}{\alpha_N(T)} \leqslant 1 \qquad (A1)$$

and

$$0 < \theta_N(T) = \frac{\lambda \beta_N(T-1)}{\beta_N(T)} \leqslant 1 \qquad (A2)$$

and keeping only first order errors, we obtain from each equation of Table 2 the following error equations :

- $e_N^P(T) = x(T-1) - A_{N,T-1}^T \; X_N(T-1)$

$$\Delta e_N^P(T) = - \Delta A_{N,T-1}^T \; X_N(T-1) \qquad (A3)$$

- $\alpha_N(T) = \lambda\alpha_N(T-1) + \gamma_N(T-1)\, e_N^P(T)^2$

$$\frac{\Delta\alpha_N(T)}{\alpha_N(T)} = \theta'_N(T)\,\frac{\Delta\alpha_N(T-1)}{\alpha_N(T-1)} + (1-\theta'_N(T))\,\frac{\Delta\gamma_N(T-1)}{\gamma_N(T-1)} + 2\,\frac{e_N(T)}{\alpha_N(T)}\,\Delta e_N^P(T) \qquad (A4)$$

- $\gamma_{N+1}(T) = \dfrac{\lambda\alpha_N(T-1)}{\alpha_N(T)}\ \gamma_N(T-1)$

Using (A4), we have :

$$\frac{\Delta\gamma_{N+1}(T)}{\gamma_{N+1}(T)} = (1-\theta'_N(T))\,\frac{\Delta\alpha_N(T-1)}{\alpha_N(T-1)} + \theta'_N(T)\,\frac{\Delta\gamma_N(T-1)}{\gamma_N(T-1)} - 2\,\frac{e_N(T)}{\alpha_N(T)}\,\Delta e_N^P(T) \qquad (A5)$$

- $r_N^P(T) = -\lambda\beta_N(T-1)\ \hat{C}_{N+1,T}^{N+1}$

$$\frac{\Delta r_N^P(T)}{r_N^P(T)} = \frac{\Delta\beta_N(T-1)}{\beta_N(T-1)} + \frac{\Delta\hat{C}_{N+1,T}^{N+1}}{\hat{C}_{N+1,T}^{N+1}} \qquad (A6)$$

- $\theta_N(T) = 1+\gamma_{N+1}(T)\ r_N^P(T)\ \hat{C}_{N+1,T}^{N+1}$

$$\frac{\Delta\theta_N(T)}{\theta_N(T)} = -\left(\frac{1}{\theta_N(T)} - 1\right)\left\{\frac{\Delta r_N^P(T)}{r_N^P(T)} + \frac{\Delta\hat{C}_{N+1,T}^{N+1}}{\hat{C}_{N+1,T}^{N+1}} + \frac{\Delta\gamma_{N+1}(T)}{\gamma_{N+1}(T)}\right\} \qquad (A7)$$

- $\gamma_N(T) = \dfrac{\gamma_{N+1}(T)}{\theta_N(T)}$

$$\frac{\Delta\gamma_N(T)}{\gamma_N(T)} = \frac{\Delta\gamma_{N+1}(T)}{\gamma_{N+1}(T)} - \frac{\Delta\theta_N(T)}{\theta_N(T)}$$

Using the previous equalities, we have :

$$\frac{\Delta\gamma_N(T)}{\gamma_N(T)} = \frac{(1-\theta'_N(T))}{\theta_N(T)} \frac{\Delta\alpha_N(T-1)}{\alpha_N(T-1)} + \frac{\theta'_N(T)}{\theta_N(T)} \frac{\Delta\gamma_N(T-1)}{\gamma_N(T-1)} + \left(\frac{1}{\theta_N(T)} - 1\right) \frac{\Delta\beta_N(T-1)}{\beta_N(T-1)}$$

$$- \frac{2}{\theta_N(T)} \frac{e_N(T)}{\alpha_N(T)} \Delta e^P_N(T) + 2\left(\frac{1}{\theta_N(T)} - 1\right) \frac{\Delta \tilde{C}^{N+1}_{N+1,T}}{\tilde{C}^{N+1}_{N+1,T}} \qquad (A8)$$

$$\beta_N(T) = \lambda\beta_N(T-1) + \gamma_N(T)\, r^P_N(T)^2$$

Using (A2), we have after some calculations :

$$\frac{\Delta\beta_N(T)}{\beta_N(T)} = \left(\frac{1}{\theta_N(T)} - 1\right)(1-\theta'_N(T)) \frac{\Delta\alpha_N(T-1)}{\alpha_N(T-1)} + \theta'_N(T)\left(\frac{1}{\theta_N(T)} - 1\right) \frac{\Delta\gamma_N(T-1)}{\gamma_N(T-1)}$$

$$+ \frac{1}{\theta_N(T)} \frac{\Delta\beta_N(T-1)}{\beta_N(T-1)} - 2\left(\frac{1}{\theta_N(T)} - 1\right) \frac{e_N(T)}{\alpha_N(T)} \Delta e^P_N(T) + 2\left(\frac{1}{\theta_N(T)} - 1\right) \frac{\Delta \tilde{C}^{N+1}_{N+1,T}}{\tilde{C}^{N+1}_{N+1,T}}$$

(A9)

According to relations (A4), (A8) and (A9), we are now able to compute at each time $T \geq (T_0+1)$, the relative errors on the energy residuals $\alpha_N(T)$, $\gamma_N(T)$ and $\beta_N(T)$. Let us now concentrate on these numerical errors and study how they evolve with time.

We first notice (after some straightforward calculations) that if $s_N(T)$ denotes the sum :

$$s_n(T) = \frac{1}{2}\left\{ \frac{\Delta\beta_N(T)}{\beta_N(T)} - \frac{\Delta\gamma_N(T)}{\gamma_N(T)} - \frac{\Delta\alpha_N(T)}{\alpha_N(T)} \right\} \qquad \text{(A10)}$$

then

$$s_n(T) = s_n(T-1) = s_0 \qquad \text{for } T \geq (T_0+1)$$

where

$$s_0 = s(T_0) = \frac{1}{2}\left\{ \frac{\Delta\beta_N(T_0)}{\beta_N(T_0)} - \frac{\Delta\gamma_N(T_0)}{\gamma_N(T_0)} - \frac{\Delta\alpha_N(T_0)}{\alpha_N(T_0)} \right\} \qquad \text{(A11)}$$

Replacing $\frac{\Delta\beta_N(T-1)}{\beta_N(T-1)}$ by $\{2s_0 + \frac{\Delta\gamma_N(T-1)}{\gamma_N(T-1)} + \frac{\Delta\alpha_N(T-1)}{\alpha_N(T-1)}\}$ in (A8), we obtain :

$$\frac{\Delta\gamma_N(T)}{\gamma_N(T)} = \left(\frac{2}{\theta_N(T)} - \frac{\theta'_N(T)}{\theta_N(T)} - 1\right)\frac{\Delta\alpha_N(T-1)}{\alpha_N(T-1)} + \left(\frac{\theta'_N(T)}{\theta_N(T)} + \frac{1}{\theta_N(T)} - 1\right)\frac{\Delta\gamma_N(T-1)}{\gamma_N(T-1)}$$

$$- \frac{2}{\theta_N(T)}\frac{e_N(T)}{\alpha_N(T)}\Delta e_N^P(T) + 2\left(\frac{1}{\theta_N(T)} - 1\right)\frac{\Delta\tilde{C}^{N+1}_{N+1,T}}{\tilde{C}^{N+1}_{N+1,T}} + 2\left(\frac{1}{\theta_N(T)} - 1\right)s_0 \qquad \text{(A12)}$$

Introducing the state vector :

$$\Xi(T) = \left[\frac{\Delta\alpha_N(T)}{\alpha_N(T)} \quad \frac{\Delta\gamma_N(T)}{\gamma_N(T)} - \frac{\Delta\alpha_N(T)}{\alpha_N(T)}\right]^T \qquad \text{(A13)}$$

and denoting

$$\begin{cases} \delta_N(T) = \dfrac{1}{\theta_N(T)} - 1 & \text{(A14)} \\ \\ \delta'_N(T) = 2\theta'_N(T) - 1 & \text{(A15)} \end{cases}$$

we obtain from relations (A4) and (A12), the error propagations (3.4), (3.9) analyzed in section 3.

REFERENCES

|1| L. Ljung and T. Söderström : "Theory and Practice of Recursive Identification". **MIT Press** (1983).

|2| L. Ljung, M. Morf and D. Falconer : "Fast calculation of gain matrices for recursive estimation schemes". **Int. Journal of Control**, vol.27, pp.1-19, Janv. 1978.

|3| D.-D. Falconer and L. Ljung : "Application of Fast Kalman estimation to adaptive equalization". **IEEE Trans. on Comm.**, vol.COMM-26, pp.1439-1446, Oct. 1978.

|4| G. Carayannis, D. Manolakis and N. Kalouptsidis : "A fast sequential algorithm for least squares filtering and prediction". **IEEE Trans. on ASSP**, Dec. 1983.

|5| G. Carayannis, D. Manolakis and N. Kalouptsidis : "Fast Kalman type algorithms for sequential signal processing". **Proc. ICASSP** 1983, Boston, PP.186-189.

|6| J. Cioffi and T. Kailath : "Fast recursive least-squares transversal filters for adaptive filtering". **IEEE Trans. on ASSP**, vol.32, n°2, pp.304-337, April 1984.

|7| J. Cioffi and T. Kailath : "Windowed fast transversal filters adaptive algorithms with normalization". **IEEE Trans. on ASSP**, vol.33, n°3, PP.607-625, June 1985.

|8| D. Manolakis, G. Carayannis and N. Kalouptsidis : "On the computational organization of fast sequential algorithms". **Proc. of ICASSP** 1984, San Diego, pp.43-7-1.

|9| D.-W. Lin : "On digital implementation of the fast Kalman algorithm". **IEEE Trans. on ASSP**, vol.32, n°5, pp.998-1005, Oct. 1984.

|10| P. Fabre and C. Gueguen : "Fast recursive least-squares algorithms : preventing divergence". **Proc. of ICASSP**, 1984, San Diego, PP.30-2.

|11| M. Bellanger and R. Lamberti : "Stability conditions for fast weighted least squares algorithms". **Submitted to ASSP**

|12| S. Ljung and L. Ljung : "Error propagation properties of recursive least-squares adaptation algorithms". **Automatica**, vol.21, n°2, pp.157-167, 1985.

|13| J.-L. Botto : "Etude des algorithmes transversaux rapides : application à l'annulation d'échos acoustiques pour l'audioconférence". **Thèse de Docteur Ingénieur**, Université de Rennes I, IRISA (France), Mai 1986.

|14| P. Fabre and C. Gueguen : "Improvement of fast RLS algorithms via normalization : a comparative study". **IEEE Trans. on ASSP**, vol.34, n°2, pp.296-308, April, 1986.

LIST OF TABLES

Table 1

Definition of FTF variables

definition / set	identified filter coefficient (Nx1) vector	a-priori scalar residual	a-posteriori scalar residual	energy residual
output variables	$H_{N,T}$	$\varepsilon_N^P(T)$	$\varepsilon_N(T)$	$V_N(T)$
forward variables	$A_{N,T}$	$e_N^P(T)$	$e_N(T)$	$\alpha_N(T)$
backward variables	$B_{N,T}$	$r_N^P(T)$	$r_N(T)$	$\beta_N(T)$
Kalman variables	$C_{N,T}$		$\gamma_N(T)$	$\gamma_N(T)$

Table 2

The FTF(7N) algorithm [6]

→ variables available at time T : $\alpha_N(T-1)$, $\beta_N(T-1)$, $\gamma_N(T-1)$

$A_{N,T-1}$, $B_{N,T-1}$, $\tilde{C}_{N,T-1}$ and $H_{N,T-1}$

→ new information at time T : $y(T)$, $x(T-1)$

Computation of the dual Kalman gain vector $\tilde{C}_{N,T}$

$$e_N^P(T) = x(T-1) - A_{N,T-1}^T X_N(T-1)$$

$$e_N(T) = \gamma_N(T-1)\, e_N^P(T)$$

$$\alpha_N(T) = \lambda\alpha_N(T-1) + e_N(T)\, e_N^P(T)$$

$$\gamma_{N+1}(T) = \frac{\lambda\alpha_N(T-1)}{\alpha_N(T)}\ \gamma_N(T-1)$$

$$\tilde{C}_{N+1,T} = \begin{bmatrix} 0 \\ \tilde{C}_{N,T-1} \end{bmatrix} - \frac{e_N^P(T)}{\lambda\alpha_N(T-1)} \begin{bmatrix} 1 \\ -A_{N,T-1} \end{bmatrix}$$

$$A_{N,T} = A_{N,T-1} - e_N(T)\, \tilde{C}_{N,T-1}$$

$$r_N^P(T) = -\lambda\beta_N(T-1)\, \tilde{C}_{N+1,T}^{N+1}$$

$$\theta_N(T) = 1 + \gamma_{N+1}(T)\, r_N^P(T)\, \tilde{C}_{N+1,T}^{N+1}$$

$$\gamma_N(T) = \frac{\gamma_{N+1}(T)}{\theta_N(T)}$$

$$r_N(T) = \gamma_N(T)\, r_N^P(T)$$

$$\beta_N(T) = \lambda\beta_N(T-1) + r_N(T)\, r_N^{\Gamma}(T)$$

$$\begin{bmatrix} \tilde{C}_{N,T} \\ \\ 0 \end{bmatrix} = \tilde{C}_{N+1,T} - \tilde{C}_{N+1,T}^{N+1} \begin{bmatrix} -B_{N,T-1} \\ \\ 1 \end{bmatrix}$$

$$B_{N,T} = B_{N,T-1} - r_N(T)\, \tilde{C}_{N,T}$$

Filtering of the y(T) signal

$$\varepsilon_N^P(T) = y(T) - H_{N,T-1}^T\, X_N(T)$$

$$\varepsilon_N(T) = \gamma_N(T)\, \varepsilon_N^P(T)$$

$$H_{N,T} = H_{N,T-1} - \varepsilon_N(T)\, \tilde{C}_{N,T}$$

Table 3

A stabilized FTF (7N+N+2N) algorithm

→ variables available at time T : $\alpha_N(T-1)$, $\beta_N(T-1)$, $\gamma_N(T-1)$

$A_{N,T-1}$, $B_{N,T-1}$, $\tilde{C}_{N,T-1}$ and $H_{N,T-1}$

→ new information at time T : $y(T)$, $x(T-1)$

Computation of the control variable $\xi_N(T)$

$$e_N^P(T) = x(T-1) - A_{N,T-1}^T X_N(T-1)$$

$$r_N^P(T) = x(T-N-1) - B_{N,T-1}^T X_N(T)$$

$$\gamma_{N+1}(T) = \frac{\lambda\alpha_N(T-1)}{\lambda\alpha_N(T-1) + \gamma_N(T-1)\, e_N^P(T)^2}\ \gamma_N(T-1)$$

$$\gamma_N(T) = \{1 + \gamma_{N+1}(T) r_N^P(T) (\tilde{C}_{N,T-1}^N + \frac{e_N^P(T)}{\lambda\alpha_N(T-1)}\ A_{N,T-1}^N)\}^{-1}\ \gamma_{N+1}(T)$$

$$k_N(T-1) = \lambda^{-N}\gamma_N(T-1)$$

$$\xi_N(T) = r_N^P(T) + A_{N,T-1}^N k_N(T-1) e_N^P(T) + \lambda\beta_N(T-1)\ \tilde{C}_{N,T-1}^N$$

$$\xi_N'(T) = \Big\{1 + \rho(\frac{1}{\gamma_N(T)} - 1) + \rho\ A_{N,T-1}^{N\,2} k_N(T-1)^2 (\frac{1}{\gamma_N(T-1)} - 1)$$

$$+ 2\rho\ A_{N,T-1}^N\ k_N(T-1)^2\ \tilde{C}_{N,T-1}^N\ e_N^P(T)\Big\}^{-1}\ \xi_N(T)$$

Correction of the transversal filters $A_{N,T-1}$ and $B_{N,T-1}$

$$\begin{cases} A'_{N,T-1} = A_{N,T-1} \\ e_N^{P\prime}(T) = (1 - \rho k_N(T-1)\ \tilde{C}_{N,T-1}^N \xi_N'(T))\ e_N^P(T) - \rho\lambda^{-N} A_{N,T-1}^N (1-\gamma_N(T-1))\ \xi_N'(T) \end{cases}$$

$$\tilde{C}_{N+1,T} = \begin{bmatrix} 0 \\ \tilde{C}_{N,T-1} \end{bmatrix} - \frac{e_N^P(T)}{\lambda \alpha_N(T-1)} \begin{bmatrix} 1 \\ -A'_{N,T-1} \end{bmatrix}$$

$$\left\{ \begin{bmatrix} B'_{N,T-1} \\ 1 \end{bmatrix} = \frac{1}{1+\rho C^{\tilde{N+1}}_{N+1,T}\, \xi'_N(T)} \left\{ \begin{bmatrix} B_{N,T-1} \\ -1 \end{bmatrix} - \rho \xi'_N(T)\, \tilde{C}_{N+1,T} \right\} \right.$$

$$\left. r^{P'}_N(T) = r^P_N(T) - \rho \left(\frac{1}{\gamma_N(T)} - 1\right) \xi'_N(T) \right\}$$

Classical FTF algorithm

$$e_N(T) = \gamma_N(T-1)\, e^{P'}_N(T)$$

$$\alpha_N(T) = \lambda \alpha_N(T-1) + e_N(T)\, e^{P'}_N(T)$$

$$A_{N,T} = A'_{N,T-1} - e_N(T)\, \tilde{C}_{N,T-1}$$

$$r_N(T) = \gamma_N(T)\, r^{P'}_N(T)$$

$$\beta_N(T) = \lambda \beta_N(T-1) + r_N(T)\, r^{P'}_N(T)$$

$$\begin{bmatrix} \tilde{C}_{N,T} \\ 0 \end{bmatrix} = \tilde{C}_{N+1,T} - C^{\tilde{N+1}}_{N+1,T} \begin{bmatrix} -B_{N,T-1} \\ 1 \end{bmatrix}$$

$$B_{N,T} = B'_{N,T-1} - r_N(T)\, \tilde{C}_{N,T}$$

Filtering of the y(T) signal

$$\varepsilon^P_N(T) = y(T) - H^T_{N,T-1}\, X_N(T)$$

$$\varepsilon_N(T) = \gamma_N(T)\, \varepsilon^P_N(T)$$

$$H_{N,T} = H_{N,T-1} - \varepsilon_N(T)\, \tilde{C}_{N,T}$$

Table 4

A stabilized NFTF (11N+N) algorithm

→ variables available at time T : $\gamma_N(T-1)^{1/2}$

$\hat{C}_{N,T-1}$, $\hat{A}_{N,T-1}$, $\hat{B}_{N,T-1}$ and $H_{N,T-1}$

→ new information at time T : $y(T)$, $x(T-1)$

Computation of the control variable $\xi_N(T)$

$$\hat{e}_N^P(T) = \hat{A}_{N,T-1}^T \; X_{N+1}(T)$$

$$\hat{r}_N^P(T) = \hat{B}_{N,T-1}^T \; X_{N+1}(T)$$

$$v = \lambda^{-1/2} \gamma_N(T-1)^{1/2} \; \hat{e}_N^P(T)$$

$$\hat{e}_N^c(T) = (1 + v^2)^{-1/2}$$

$$\hat{e}_N(T) = v \quad \hat{e}_N^c(T)$$

$$\gamma_{N+1}(T)^{1/2} = \hat{e}_N^c(T) \; \gamma_N(T-1)^{1/2}$$

$$\hat{r}_N(T) = \lambda^{-1/2} \gamma_{N+1}(T)^{1/2} \; \hat{r}_N^P(T)$$

$$\hat{r}_N^c(T) = (1 - \hat{r}_N^2(T))^{1/2}$$

$$\gamma_N(T)^{1/2} = \hat{r}_N^c(T)^{-1} \; \gamma_{N+1}(T)^{1/2}$$

$$\hat{\xi}_N(T) = \hat{r}_N(T) - \left(\frac{\hat{A}_{N,T-1}^{N+1}}{\hat{B}_{N,T-1}^{N+1}} \right) \hat{e}_N(T) + \lambda^{1/2} \left(\frac{\hat{C}_{N,T-1}^{N}}{\hat{B}_{N,T-1}^{N+1}} \right) \hat{e}_N^c(T)$$

$$\xi_N(T) = \lambda^{1/2} \; \gamma_{N+1}(T)^{-1/2} \left(\frac{\hat{\xi}_N(T)}{\hat{B}_{N,T-1}^{N+1}} \right)$$

$$k_N(T-1) = \left(\frac{\hat{A}_{N,T-1}^{1}}{\hat{B}_{N,T-1}^{N+1}} \right)^2 \quad ; \quad k_N'(T-1) = - \left(\frac{\hat{A}_{N,T-1}^{N+1}}{\hat{A}_{N,T-1}^{1}} \right) \; k_N(T-1)$$

$$\xi'_N(T) = \left\{1+\rho\left(\frac{1}{\gamma_N(T)} - 1\right) + \rho k'_N(T-1)^2 \left(\frac{1}{\gamma_N(T-1)} - 1\right)\right.$$

$$\left. +2\rho k_N(T-1)\ k'_N(T-1)\ \gamma_N(T-1)^{-1/2}\ \hat{e}^P_N(T) \begin{pmatrix} \hat{C}^N_{N,T-1} \\ \hat{A}^1_{N,T-1} \end{pmatrix}\right\}^{-1} \xi_N(T)$$

$$\xi^A_N(T) = \rho\ k_N(T-1)\ \gamma_N(T-1)^{-1/2}\ \xi'_N(T)$$

$$\xi^B_N(T) = \rho\ \hat{B}^{N+1}_{N,T-1}\ \gamma_N(T)^{-1/2}\ \xi'_N(T)$$

Correction of scalar normalized residuals $\hat{e}^P_N(T)$, $\hat{r}^P_N(T)$

$$\begin{cases} \hat{e}^{P'}_N(T) = (1 - \hat{C}^N_{N,T-1}\ \xi^A_N(T))\ \hat{e}^P_N(T) + \hat{A}^{N+1}_{N,T-1}(\gamma_N(T-1)^{-1/2} - \gamma_N(T-1)^{1/2})\xi^A_N(T) \\ \hat{r}^{P'}_N(T) = \hat{r}^P_N(T) - (\gamma_N(T)^{-1/2} - \gamma_N(T)^{1/2})\ \xi^B_N(T) \end{cases}$$

Classical FTF algorithm

$$v = \lambda^{-1/2}\gamma_N(T-1)^{1/2}\ \hat{e}^{P'}_N(T)$$

$$\hat{e}^c_N(T) = (1 + v^2)^{-1/2}$$

$$\hat{e}_N(T) = v\ \ \hat{e}^c_N(T)$$

$$\gamma_{N+1}(T)^{1/2} = \hat{e}^c_N(T)\ \gamma_N(T-1)^{1/2}$$

$$\hat{C}_{N+1,T} = \{\hat{e}^c_N(T) + \lambda^{-1/2}\hat{e}_N(T)\ \hat{A}^{N+1}_{N,T-1}\ \xi^A_N(T)\} \begin{bmatrix} 0 \\ \sim \\ C_{N,T-1} \end{bmatrix} - \lambda^{-1/2}\hat{e}_N(T)\hat{A}_{N,T-1}$$

$$\hat{A}_{N,T} = \lambda^{-1/2}\ \hat{e}^c_N(T)\ \hat{A}_{N,T-1} + \{\hat{e}_N(T) - \lambda^{-1/2}\ \hat{e}^c_N(T)\ \hat{A}^{N+1}_{N,T-1}\ \xi^A_N(T)\} \begin{bmatrix} 0 \\ \hat{C}_{N,T-1} \end{bmatrix}$$

$$\hat{r}_N(T) = \lambda^{-1/2}\ \gamma_{N+1}(T)^{1/2}\ \hat{r}^{P'}_N(T)$$

$$\hat{r}^{C}_{N}(T) = (1 - \hat{r}^{2}_{N}(T))^{1/2}$$

$$\gamma_{N}(T)^{1/2} = \hat{r}^{C}_{N}(T)^{-1} \quad \gamma_{N+1}(T)^{1/2}$$

$$\begin{bmatrix} \hat{C}_{N,T} \\ 0 \end{bmatrix} = \frac{1}{\hat{r}^{C}_{N}(T) - \lambda^{-1/2}\, \hat{r}_{N}(T)\, \xi^{B}_{N}(T)} \{\hat{C}_{N+1,T} + \lambda^{-1/2}\, \hat{r}_{N}(T)\, \hat{B}_{N,T-1}\}$$

$$\hat{B}_{N,T} = \lambda^{-1/2}\, \hat{r}^{C}_{N}(T)\, \hat{B}_{N,T-1} + \{\hat{r}_{N}(T) + \lambda^{-1/2}\, \hat{r}^{C}_{N}(T) \xi^{B}_{N}(T)\} \begin{bmatrix} \hat{C}_{N,T} \\ 0 \end{bmatrix}$$

Filtering of the y(T) signal

$$\varepsilon^{P}_{N}(T) = y(T) - H^{T}_{N,T-1}\, X_{N}(T)$$

$$H_{N,T} = H_{N,T-1} - (\gamma_{N}(T)^{1/2}\, \varepsilon^{P}_{N}(T))\, \hat{C}_{N,T}$$

Table 5

Average number of sampling steps
Between two consecutive divergences

For N = 10 and a centered white noise of unit variance as input signal

forgetting factor λ / Algorithm	λ=0.99 (W=10N)	λ=0.98 (W=5N)	λ=0.95 (W=2N)
FAEST (7N)	1 000	very unstable	very unstable
FTF (7N)	1 000	very unstable	very unstable
FTF (8N)	1 000	very unstable	very unstable
Fast Kalman (10N)	20 000	9 000	1 400
FTF (7N+N)	30 000	10 000	1 500
FTF (8N+N)	28 000	10 000	1 500

List of Figures

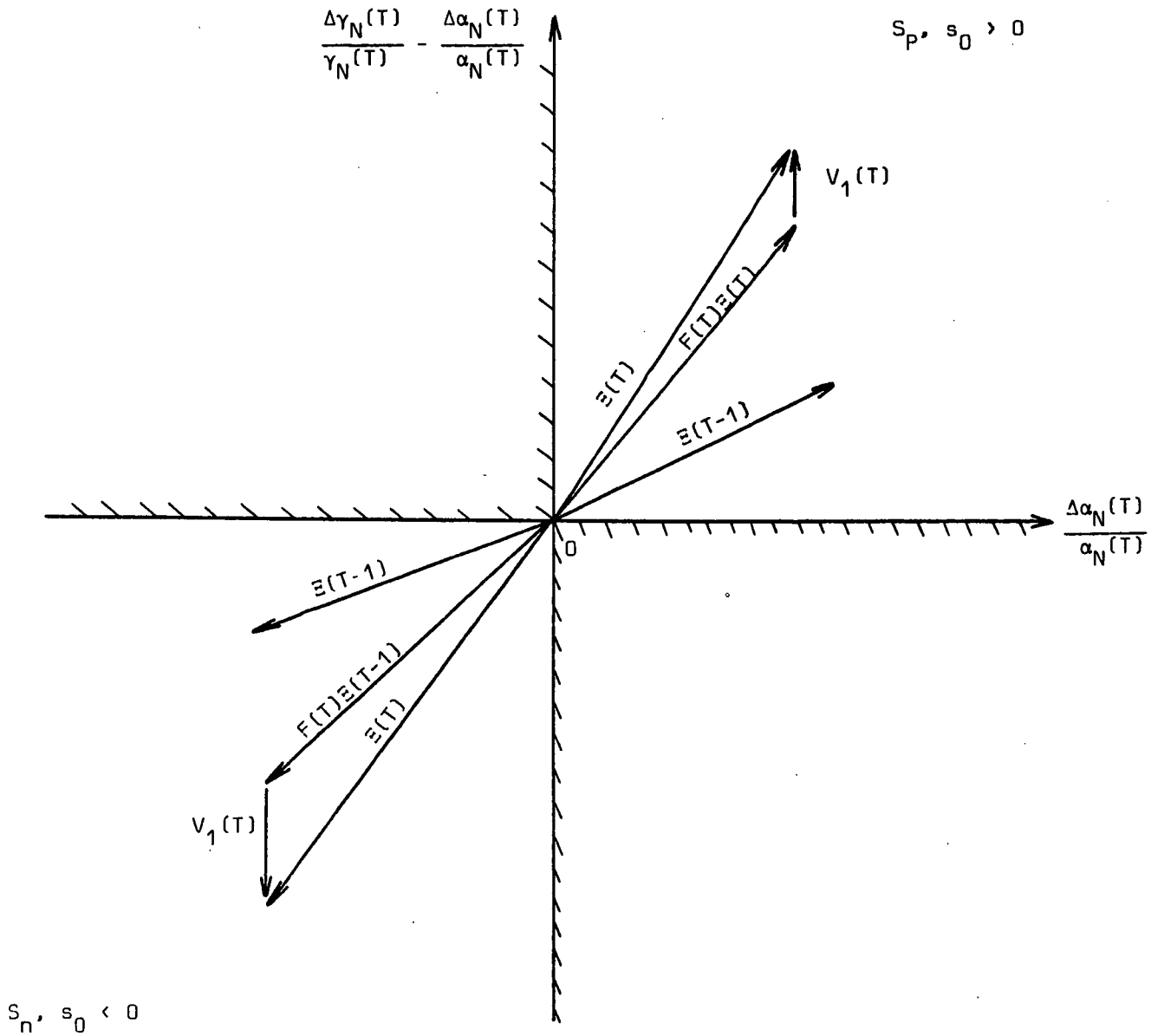

Figure 1: Numerical error propagation analysis for the FTF(7N) algorithm

$(\Xi(T) = F(T)\Xi(T-1) + V_1(T))$

Figure 2: $\gamma_N(T)$ for the FTF(8N) algorithm

(N=10, W=10N, μ =0.1, x(T) white noise)

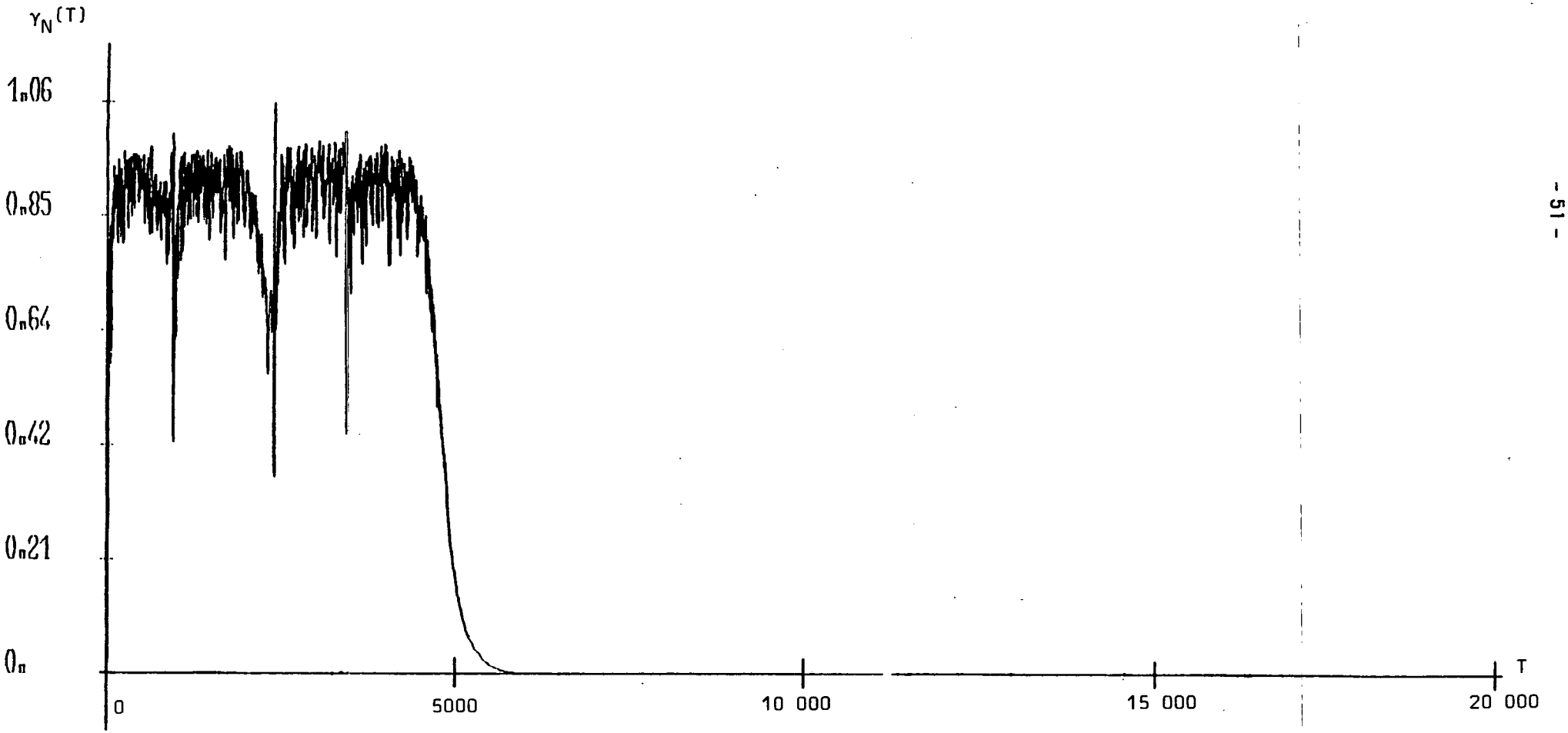

Figure (3.a): $\gamma_N(T)$ for the FTF(7N) algorithm

(N=500, W=3N, μ =10, x(T) white noise)

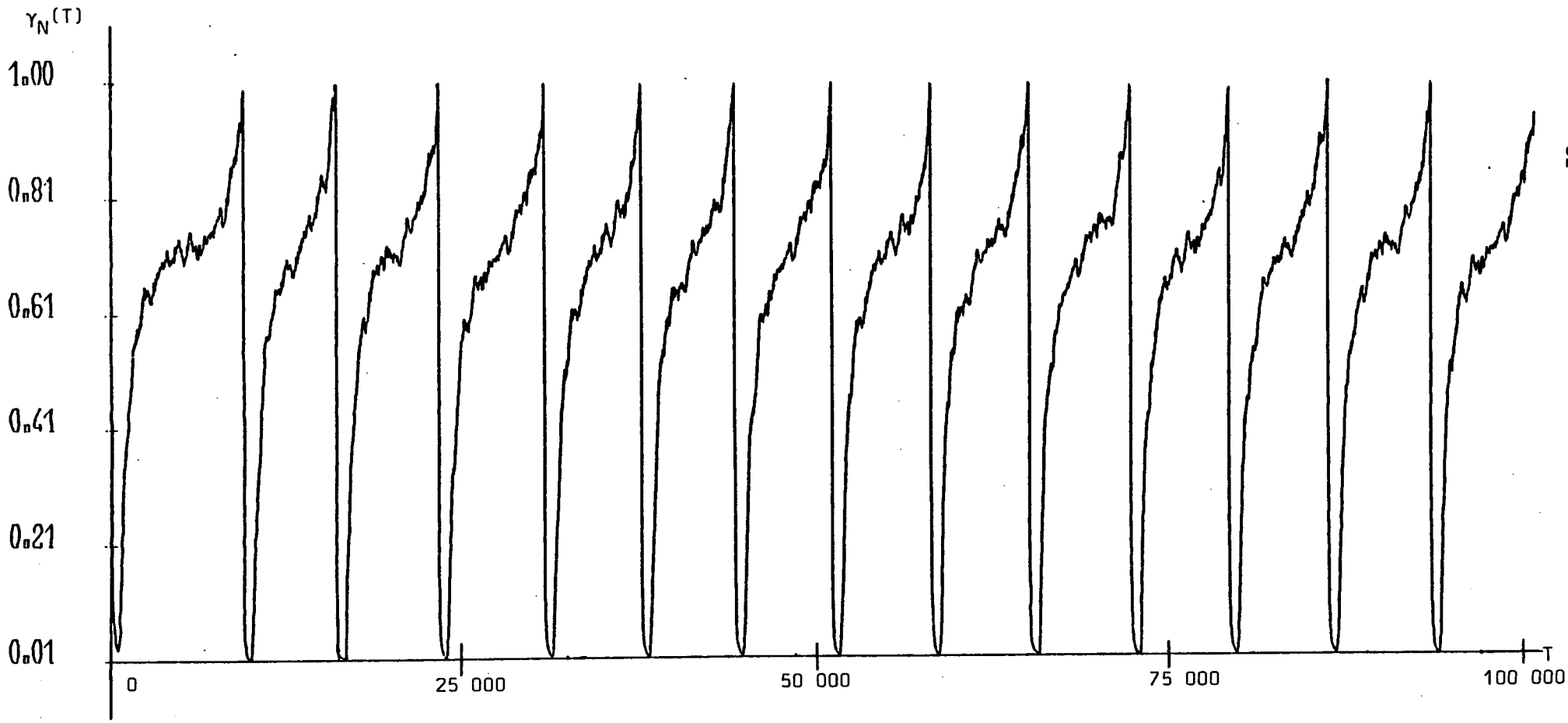

Figure (3.b): $\gamma_N(T)$ and $\xi_N(T)$ for the FTF(7N+N) algorithm
(N=500, W=3N, μ =10, x(T) white noise)

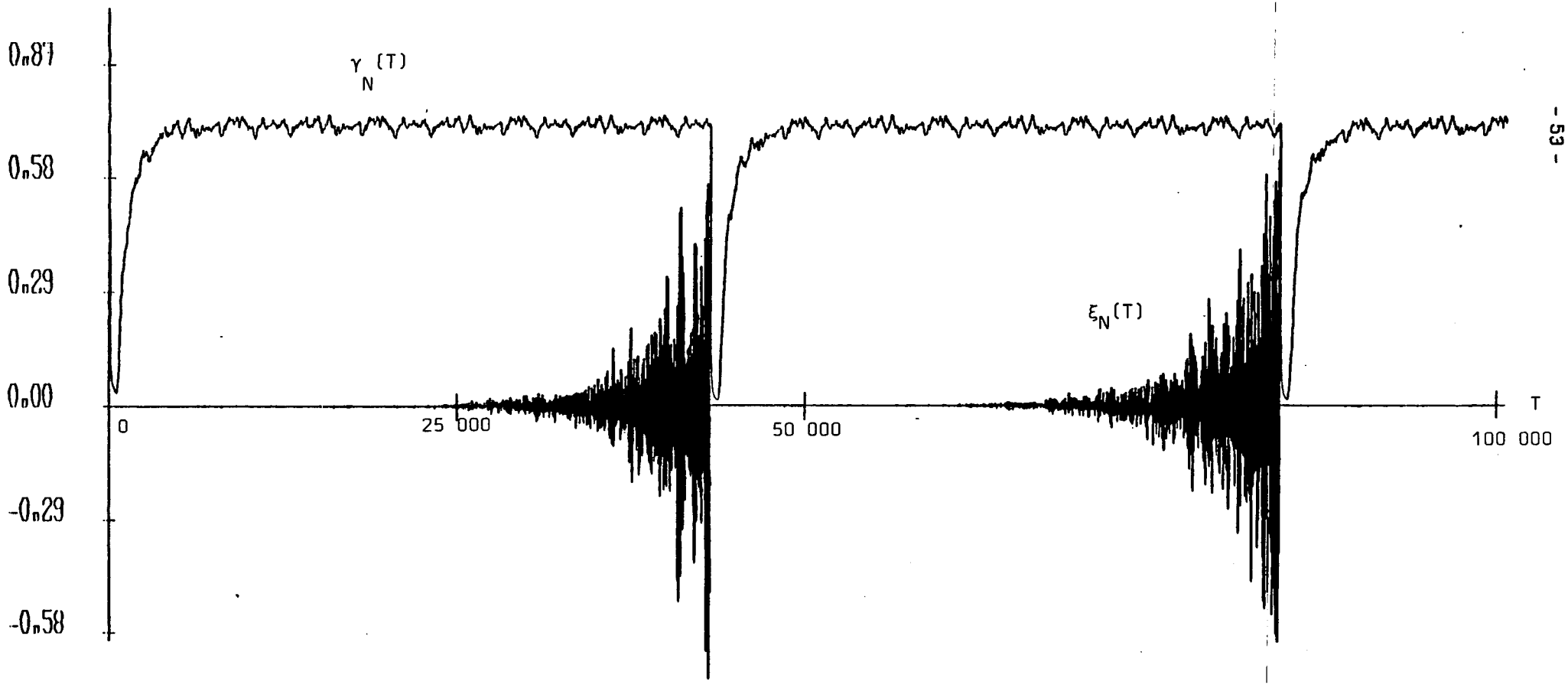

Figure (3.c): $\gamma_N(T)$ and $\xi_N(T)$ for the stabilized FTF(7N+N+2N) algorithm
(N=500, W=3N, μ =10, x(T) white noise)

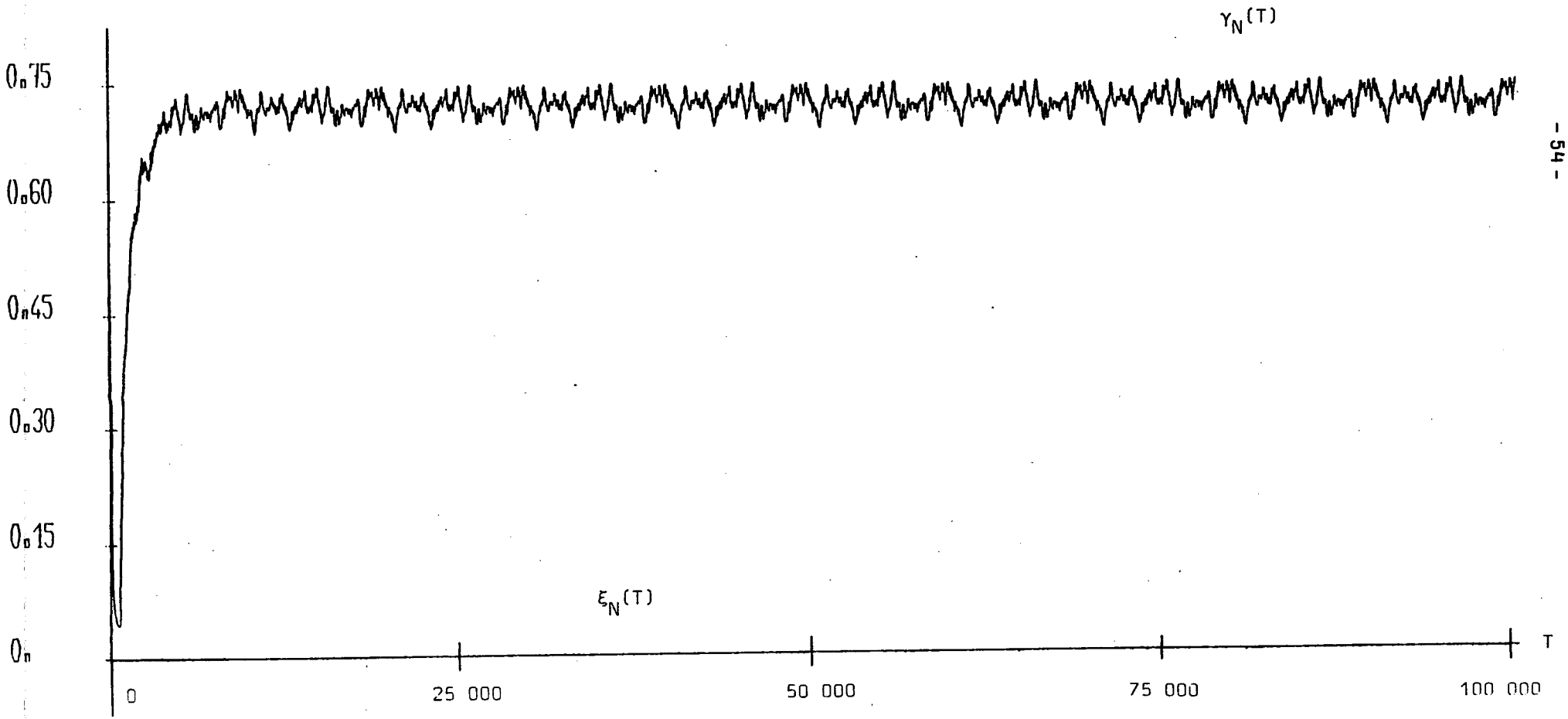

Figure (4.a): $\gamma_N(T)^{1/2}$ and $\hat{\xi}_N(T)$ for the NFTF(11N+N) algorithm with a 16-bit fixed-point arithmetic

(N=10, W=2N, μ =1, ρ=0, x(T) white noise)

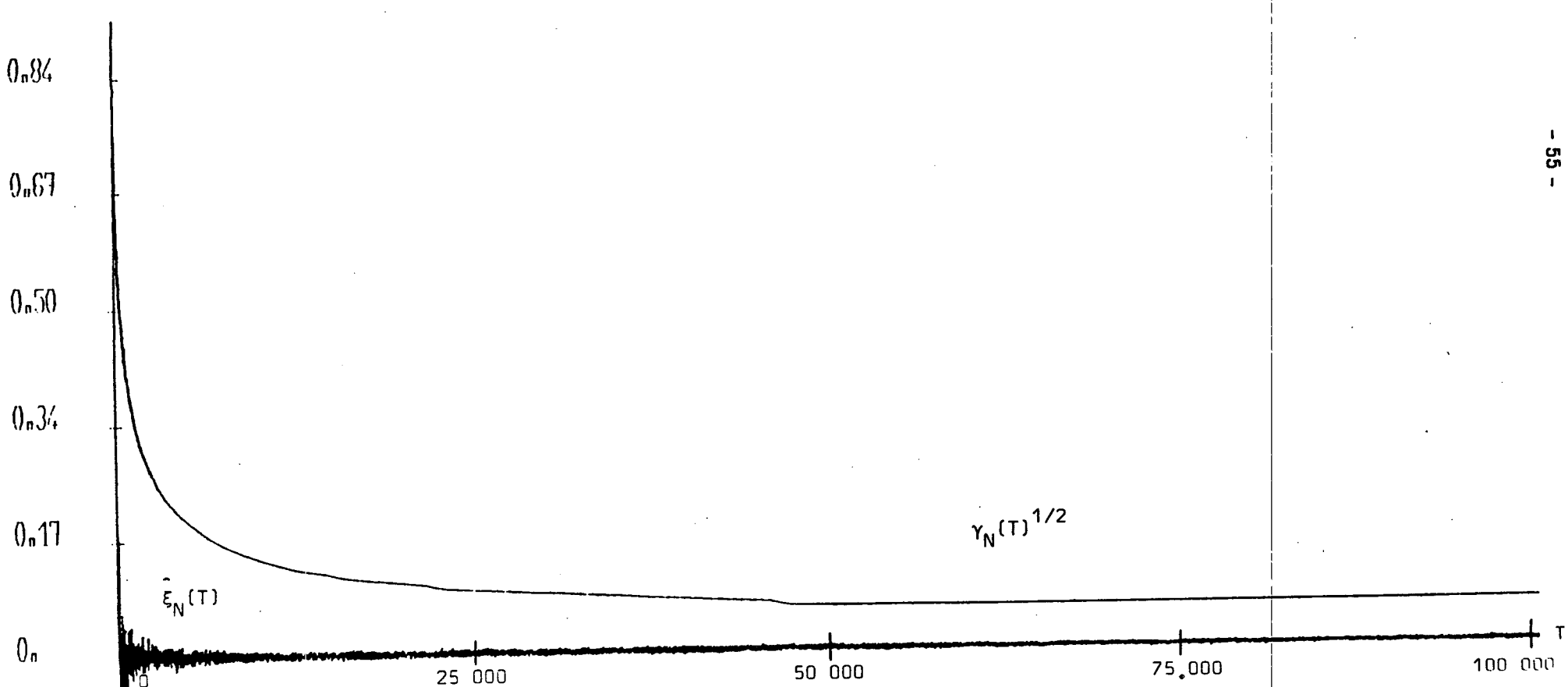

Figure (4.b): $\gamma_N(T)^{1/2}$ and $\hat{\xi}_N(T)$ for the stabilized NFTF(11N+N) algorithm with a 16-bit fixed-point arithmetic

(N=10, W=2N, μ=1, ρ=1, x(T) white noise)