Stabilizing the Fast Kalman Algorithms

JEAN-LUC BOTTO AND GEORGE V. MOUSTAKIDES, MEMBER, IEEE

Abstract—Fast Kalman algorithms are algorithms that solve, in a very efficient way, the recursive least-squares estimation problem. Unfortunately they are known to exhibit a very unstable behavior, due basically to the accumulation of roundoff errors. It is the structure of the algorithms that favors this accumulation, which is present even when the data are well behaved. In this paper, by introducing a redundant equation, that is, by computing a specific quantity of the algorithms in two different ways, we use the difference of these two ways as a measure of the accumulation of the roundoff errors. This difference is consequently used to correct the variables of the algorithm at every time step in order to stabilize it. The correction is defined as the solution of a specific minimization problem. The resulting algorithm still has the nice complexity properties of the original algorithm (linear in the number of parameters to be estimated), but has a much more stable behavior.

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 [12] have been implemented under the form of Fast Kalman Algorithms (FKA) [3]-[6], [10], [13]. These fast least-squares algorithms require a number of arithmetic operations which is proportional to the number N of parameters to be estimated per sample. This is comparable to the suboptimal gradient-type techniques. A useful classification of the different existing fast sequential least-squares algorithms can be found in [15].

Unfortunately, all exponentially weighted FKA are known to exhibit an unstable behavior and a sudden divergence due to the accumulation of roundoff errors in finite precision computation. This numerical instability remains the main drawback of these algorithms and limits their use in practical problems. Several techniques have been proposed to overcome this problem. In [5] and [11], the algorithms are applied in spite of their numerical instability, in parallel with a detection procedure. As soon as the detector detects instability, then the algorithm is reinitialized. This method leads to a more or less periodic reinitialization, with effects that are quite apparent (for example, in the prediction error).

Other stabilization techniques have been recently presented in [1] and [8]. They are basically regularization techniques, used to stabilize the algorithm for cases where the sample covariance matrix of the data is ill conditioned. These algorithms are always suboptimal in the sense that the identification gain that they compute is not close to the theoretical optimum gain, and thus have a reduced tracking capability. Because the above methods do not take into account the unstable structure of the FKA, they fail to stabilize the algorithms for specific well-behaved sequences (as white noise). This was confirmed in our simulations [2]. In [7] there is a presentation of the behavior of most realizations of the least-squares with respect to roundoff errors.

From a theoretical point of view, the numerical instability of the FKA of [12] and [13] has been analytically pointed out in [14] for a first-order filter and a specific input signal. Also, in [2], using simple assumptions regarding the input signal and the errors in the filters, the unstable behavior of the algorithms was shown. It was shown that the FKA exhibit two different modes of divergence—divergence toward infinity and toward zero. The first is immediately apparent since most of the variables of the algorithm blow to infinity. The second mode is less known since it is difficult to observe. It consists basically of a gain tending to zero, with the filters having usually reasonable values. In the stationary case, this is not apparent even in the prediction error. For the nonstationary case, this becomes a serious problem, because even though the prediction error is large, the gain is so small that the algorithm cannot track the signal efficiently.

In this paper we introduce a quantity that has the property of being a measure of the accumulation of the roundoff errors. This quantity is used to correct the variables of the algorithm *at every time step*. This results in an increase in the complexity of the original algorithm, but still keeping the complexity linear with respect to the number of parameters to be identified. The resulting algorithm has a much more stable behavior. We performed simulations for different orders N and signals (white, AR, speech). In all the cases we tried, our algorithm did not diverge in any sense (we tried up to 500 000 points) [2]. In this paper, as an example, we present the case $N = 10$ and we perform a simulation for 100 000 points (for other examples, see [2]).

Clearly, having an algorithm that does not diverge (or at least not as quickly as the FKA) does not mean that we have an algorithm that solves the Least-Squares (LS) problem. It is possible that the modifications we introduce lead the variables away from their correct values. To check this point, in Section IV, with simulations, we compare at every time step the Kalman gain computed

Manuscript received February 17, 1988; revised November 7, 1988.

J.-L. Botto is with TRT, 5 Av. Reaumur, 92350 Le Plessis Robinson, France.

G. V. Moustakides is with the Computer Technology Institute, Box 1122, 26110 Patras, Greece.

IEEE Log Number 8929365.

with our algorithm, with the Kalman gain computed with RLS. We use RLS because if it is performed in its most nonfast way, it is a very stable method to solve the LS problem. The relative difference of the two gains turns out to be of the order of the machine accuracy. This means that our algorithm introduces corrections in the right sense. Finally, since all corrections depend on a quantity that is a measure of the accumulation of the roundoff errors, with infinite accuracy all corrections are zero and thus we get the original algorithm.

We would like to point out that in this paper we do not prove stability of our algorithm in any sense. This is a very difficult problem. By "stabilizing" we only mean that the stable life of the algorithm is increased by a very important factor. This was basically justified with computer simulations that were performed with different orders and signals [2]. Since there is always the risk that the algorithm will possibly diverge, we also introduce a new divergence detector that detects efficiently the two modes of divergence. As we said, however, in all our simulations our algorithm did not diverge in any sense, thus, the detector was not really used.

The price we have to pay for having an algorithm with a more stable behavior is another $3N$ multiplications, thus raising the total from $7N$ to $10N$.

II. The Fast Kalman Algorithms

Before briefly presenting the FKA, let us introduce our notation. With lower case italic letters we will denote scalars, and with upper case italic letters we denote vectors. Vectors will usually have two indexes—the first denoting their length. Also, if X is a vector, then X^i denotes the ith element of X. The problem we would like to solve is the following: we are given sequentially pairs $(y(T), x(T))$. At every time instant T, we would like to determine a filter $H_{N,T}$ that solves the LS problem, i.e., that minimizes $u_N(T)$ defined as

$$u_N(T) = \sum_{k=0}^{T} \lambda^{T-k}(y(T) - H'_{N,T}X_{N,k})^2 \tag{1}$$

where $0 < \lambda \leq 1$ is the exponential forgetting factor, $X'_{N,k} = [x(k), x(k-1), \cdots, x(k-N+1)]$ and "$'$" denotes transpose. We assume that $y(T) = x(T) = 0$ for $T \leq 0$. The recursion that updates $H_{N,T}$ is the well known

$$\epsilon^p_N(T) = y(T) - H'_{N,T-1}X_{N,T}$$
$$\epsilon_N(T) = \gamma_N(T)\,\epsilon^p_N(T)$$
$$H_{N,T} = H_{N,T-1} - \epsilon_N(T)\,\tilde{C}_{N,T} \tag{2}$$

where $\epsilon^p_N(T)$ is the prior scalar residual, $\epsilon_N(T)$ is the posterior, and $\tilde{C}_{N,T}$ is the Nth-order Dual Kalman gain defined as

$$\tilde{C}_{N,T} = -\frac{1}{\lambda} R^{-1}_{N,T-1}X_{N,T} \tag{3}$$

where

$$R_{N,T} = \sum_{k=0}^{T} \lambda^{T-k}X_{N,k}X'_{N,k} \tag{4}$$

also the power $\gamma_N(T)$ is defined as

$$\gamma_N(T) = \frac{1}{1 - \tilde{C}'_{N,T}X_{N,T}}. \tag{5}$$

All FKA basically update the Dual Kalman gain in a very efficient way, with the fastest versions requiring $5N + c$ multiplications (versions FAEST, FTF [3]–[5]). Combining this with (2) gives a total of $7N + c$ multiplications. This is the fastest realization of the least squares (for a given order) up to date. Notice that we use here the Dual Kalman gain because this yields the fastest FKA.

The stabilization method that follows in the next section can be very easily applied to any FKA version, normalized or not. As an example, we will apply it to the Fast Transversal Filters (FTF) version of [5]. Also, without proof, in a table we will give the stabilized version for the Normalized FTF algorithm of [5].

III. Stabilization of the Fast Kalman Algorithms

It is widely known that two independent facts can cause the instability of an exponentially windowed least-squares identification algorithm [6], [11]. First, if the matrix $R_{N,T}$ of (4) is not well behaved because of a small λ and/or of an input sequence $\{x(T)\}$ not consistently excited, then the algorithm is unstable regardless of the way it is implemented (RLS, FKA, Ladder . . .). Sufficient conditions for a safe choice of the exponential forgetting factor were estimated in [1]. Second, for some LS algorithms, roundoff errors have the tendency to accumulate until the algorithm diverges. This is, for example, the case for most versions of the FKA and for a version of RLS [14], but it is not the case for the Ladder least-squares and some other versions of RLS. These two facts prevent the wide use of the FKA. However, we must distinguish the first fact, common to all exponentially windowed algorithms from the second, which is basically a characteristic of the FKA. Our goal in this section is to try to eliminate this drawback of the FKA, assuming that the forgetting factor λ and the signal $x(T)$ are such that the matrix $R_{N,T}$ is well behaved. Before presenting our method, let us introduce some additional notation that we are going to use in the sequel. Boldface italic letters denote theoretical values of variables, that is, variables computed with infinite accuracy. Barred letters denote corrected values of variables. Also, for simplicity, we drop the tilde from the notation of the Dual Kalman gain.

As we said in Section II, we will apply our method to the FTF version of the FKA. This algorithm is presented in Table I. The algorithm of Table I is known to be very unstable. It is possible to increase its stable life by an important factor if we compute the prior backward error $r^p_N(T)$ as $r^p_N(T) = x(T-N) - B'_{N,T-1}X_{N,T}$ instead of using the equations of Table I. Of course, this requires

TABLE I
THE FTF ALGORITHM (COMPLEXITY $7N$)

Available at time T:	$C_{N,T-1}, A_{N,T-1}, B_{N,T-1}, H_{N,T-1}, X_{N,T-1}$ $\gamma_N(T-1), \alpha_N(T-1), \beta_N(T-1)$
New Information:	$y(T), x(T)$

Computation of the Dual Kalman Gain vector $C_{N,T}$.

$$e_N^p(T) = x(T) - A'_{N,T-1}X_{N,T-1}$$
$$e_N(T) = \gamma_N(T-1)e_N^p(T)$$
$$\alpha_N(T) = \lambda\alpha_N(T-1) + e_N(T)e_N^p(T)$$
$$\gamma_{N+1}(T) = \frac{\lambda\alpha_N(T-1)}{\alpha_N(T)}\gamma_N(T-1)$$
$$C_{N+1,T} = \begin{bmatrix} 0 \\ C_{N,T-1} \end{bmatrix} - \frac{e_N^p(T)}{\lambda\alpha_N(T-1)} \begin{bmatrix} 1 \\ -A_{N,T-1} \end{bmatrix}$$
$$A_{N,T} = A_{N,T-1} - e_N(T)C_{N,T-1}$$
$$r_N^p(T) = -\lambda\beta_N(T-1)C_{N+1,T}^{N+1}$$
$$\theta_N(T) = 1 + \gamma_{N+1}(T)r_N^p(T)C_{N+1,T}^{N+1}$$
$$\gamma_N(T) = \frac{\gamma_{N+1}(T)}{\theta_N(T)}$$
$$r_N(T) = \gamma_N(T)r_N^p(T)$$
$$\beta_N(T) = \lambda\beta_N(T-1) + r_N(T)r_N^p(T)$$
$$\begin{bmatrix} C_{N,T} \\ 0 \end{bmatrix} = C_{N+1,T} - C_{N+1,T}^{N+1}\begin{bmatrix} -B_{N,T-1} \\ 1 \end{bmatrix}$$
$$B_{N,T} = B_{N,T-1} - r_N(T)C_{N,T}$$

Filtering of the $y(T)$ signal.

$$\epsilon_N^p(T) = y(T) - H'_{N,T-1}X_{N,T}$$
$$\epsilon_N(T) = \gamma_N(T)\epsilon_N^p(T)$$
$$H_{N,T} = H_{N,T-1} - \epsilon_N(T)C_{N,T}$$

another N multiplications raising the total to $8N$, but the new algorithm has a much more stable behavior compared to the original. Notice that there are two different ways for computing $r_N^p(T)$ and they seem to be quite independent. With no roundoff errors, they yield the same result, but under finite precision this is hardly the case. We can thus use their difference as a measure of the accumulation of the roundoff errors. Let us denote this difference by $\xi_N(T)$. Using the formulas of Table I, we define it as follows:

$$\begin{aligned}\xi_N(T) &= r_N^p(T) + \lambda\beta_N(T-1)C_{N+1,T}^{N+1} \\ &= x(T-N) - B'_{N,T-1}X_{N,T} \\ &\quad + \lambda\beta_N(T-1)C_{N,T-1}^{N} \\ &\quad + \frac{\beta_N(T-1)}{\alpha_N(T-1)}A_{N,T-1}^{N}(x(T) \\ &\quad - A'_{N,T-1}X_{N,T-1}) \\ &= f(A_{N,T-1}, B_{N,T-1}). \end{aligned} \tag{6}$$

Fig. 1 shows the typical behavior of $\xi_N(T)$ and $\gamma_N(T)$. Notice that $\xi_N(T)$ gradually increases in absolute value indicating the accumulation of roundoff errors. Since $\xi_N(T)$ is a measure of divergence of the algorithm, we will use $\xi_N(T)$ to correct the variables of the algorithm of Table I in order to stabilize it. We will mainly concentrate on the upper part of the algorithm of Table I concerning the adaptation of the filters $A_{N,T}$, $B_{N,T}$, and $C_{N,T}$, because it was observed that if this part is stable, then the part concerning $H_{N,T}$ is stable as well. Our goal will be to replace the two filters $A_{N,T-1}$ and $B_{N,T-1}$ with new corrected values $\bar{A}_{N,T-1}$ and $\bar{B}_{N,T-1}$. We would like the corrected values to be as close to the originally computed values as possible, but also to minimize $\bar{\xi}_N(T) = f(\bar{A}_{N,T-1}, \bar{B}_{N,T-1})$, with $f(A, B)$ defined in (6). This double requirement is expressed with the minimization of $w(\bar{A}_{N,T-1}, \bar{B}_{N,T-1})$ defined as

$$\begin{aligned} w(\bar{A}_{N,T-1}, \bar{B}_{N,T-1}) &= [\bar{A}_{N,T-1} - A_{N,T-1}]'\boldsymbol{R}_{N,T-2}[\bar{A}_{N,T-1} - A_{N,T-1}] \\ &\quad + [\bar{B}_{N,T-1} - B_{N,T-1}]'\boldsymbol{R}_{N,T-1}[\bar{B}_{N,T-1} - B_{N,T-1}] \\ &\quad + \rho[f(\bar{A}_{N,T-1}, \bar{B}_{N,T-1})]^2 \end{aligned} \tag{7}$$

where ρ is a constant. Notice that the first term in (7) is nothing but $\Sigma_{k=0}^{T-1} \lambda^{T-1-k}[(x(k) - \bar{A}'_{N,T-1}X_{N,k-1}) - (x(k) - A'_{N,T-1}X_{N,k-1})]^2$, i.e., the norm of the difference of the two (length $T-1$) error vectors. It is in this sense that we define the distance of the two filters $A_{N,T-1}$ and $\bar{A}_{N,T-1}$ (similarly the second term). The nice property of this setting is that the resulting corrections are scale invariant, a property that the original filters have as well. Unfortunately, the minimization in (7) is nonlinear because of a nonlinear term in $f(\bar{A}, \bar{B})$. Assuming that the required corrections are small, we can linearize $f(\bar{A}, \bar{B})$ around A and B, this gives

$$\begin{aligned} f(\bar{A}, \bar{B}) \approx f(A, B) &- [\bar{B} - B]'X_{N,T} + k_N(T-1) \\ &\cdot [\bar{A}^N - A^N]e_N^p(T) \\ &- k_N(T-1)A^N[\bar{A} - A]'X_{N,T-1} \end{aligned} \tag{8}$$

where $k_N(T-1) = \alpha_N(T-1)/\beta_N(T-1)$. Substituting (8) in (7), the resulting minimization is now straightforward and gives

$$\begin{aligned} \bar{A}_{N,T-1} &= A_{N,T-1} - \rho k_N(T-1)\,\bar{\xi}_N(T)\,A_{N,T-1}^N \boldsymbol{C}_{N,T-1} \\ &\quad - \rho k_N(T-1)\,\bar{\xi}_N(T)\,e_N^p(T)\,\boldsymbol{D}_{N,T-2} \end{aligned}$$

$$\bar{B}_{N,T-1} = B_{N,T-1} - \rho\bar{\xi}_N(T)\boldsymbol{C}_{N,T} \tag{10}$$

where $\boldsymbol{D}_{N,T} = (1/\lambda)\boldsymbol{R}_{N,T}^{-1}[\boldsymbol{0} \cdots \boldsymbol{1}]'$ and $\bar{\xi}_N(T) = f(\bar{A}_{N,T-1}, \bar{B}_{N,T-1})$. Using (9) and (10), we can compute

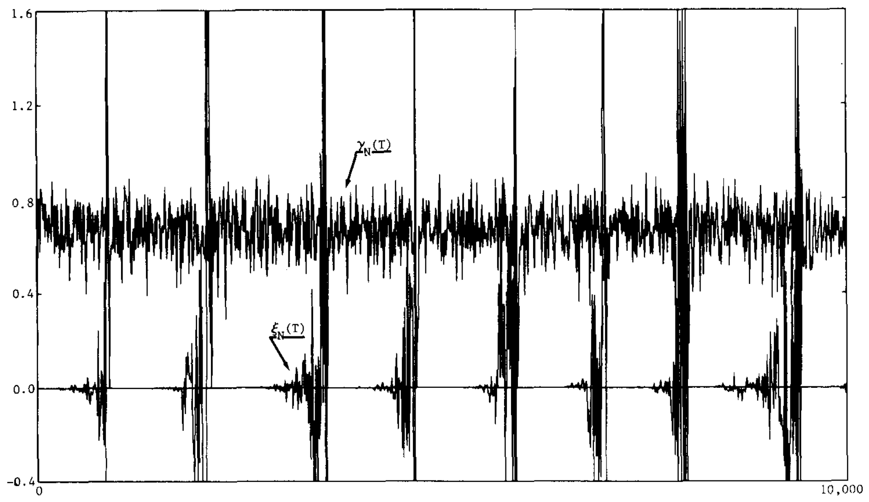


Fig. 1. Typical performance of the variables $\gamma_N(T)$ and $\xi_N(T)$ for the FTF algorithm.

the corrections for the two prior errors

$$\bar{r}^p_N(T) = r^p_N(T) - \rho\left[\frac{1}{\gamma_N(T)} - 1\right]\bar{\xi}_N(T)$$

$$\begin{aligned}\bar{e}^p_N(T) = {} & \left[1 - \rho k_N(T-1)\, \boldsymbol{C}^N_{N,T-1}\bar{\xi}_N(T)\right]e^p_N(T) \\ & - \rho k_N(T-1)\, A^N_{N,T-1}\left(\frac{1}{\gamma_N(T-1)} - 1\right) \\ & \cdot\, \bar{\xi}_N(T)\end{aligned} \tag{12}$$

where

$$\begin{aligned}\bar{\xi}_N(T) = \Bigg[& 1 + \rho\left(\frac{1}{\gamma_N(T)} - 1\right) + \rho\big(k_N(T-1) \\ & \cdot A^N_{N,T-1}\big)^2\left(\frac{1}{\gamma_N(T-1)} - 1\right) \\ & + 2\rho A^N_{N,T-1}k^2_N(T-1)\, \boldsymbol{C}^N_{N,T-1}e^p_N(T) \\ & + \rho k^2_N(T-1)\, \boldsymbol{D}^N_{N,T-2}\left(e^p_N(T)\right)^2\Bigg]^{-1} \xi_N(T).\end{aligned} \tag{13}$$

In the above expressions, we use theoretical values of certain variables. We will approximate these theoretical values as follows: $\boldsymbol{C}_{N,T-1} \approx C_{N,T-1}$, $\gamma_N(T-1) \approx \gamma_N(T-1)$ and $\gamma_N(T) \approx \gamma_N(T)$. Notice that in (10), the correction of the backward filter $\bar{B}_{N,T-1}$ is defined in terms of the theoretical Kalman Gain $\boldsymbol{C}_{N,T}$. We would like to express it in terms of known quantities. Let us first define

$$\bar{C}_{N+1,T} = \begin{bmatrix} 0 \\ C_{N,T-1}\end{bmatrix} - \frac{\bar{e}^p_N(T)}{\lambda\alpha_N(T-1)}\begin{bmatrix} 1 \\ -\bar{A}_{N,T-1}\end{bmatrix} \tag{14}$$

if we use the formulas of Table I, and approximate theoretical values with corrections, then

$$\begin{aligned}\begin{bmatrix} \boldsymbol{C}_{N,T} \\ \boldsymbol{0}\end{bmatrix} &= \boldsymbol{C}_{N+1,T} - \boldsymbol{C}^{N+1}_{N+1,T}\begin{bmatrix} -\boldsymbol{B}_{N,T-1} \\ 1\end{bmatrix} \\ &\approx \bar{C}_{N+1,T} - \bar{C}^{N+1}_{N+1,T}\begin{bmatrix} -\bar{B}_{N,T-1} \\ 1\end{bmatrix}.\end{aligned} \tag{15}$$

Substituting (15) in (10) gives

$$\begin{aligned}\begin{bmatrix} \bar{B}_{N,T-1} \\ -1\end{bmatrix} = {} & \frac{1}{1 + \rho\bar{C}^{N+1}_{N+1,T}\bar{\xi}_N(T)} \\ & \cdot\left\{\begin{bmatrix} B_{N,T-1} \\ -1\end{bmatrix} - \rho\bar{\xi}_N(T)\bar{C}_{N+1,T}\right\}.\end{aligned} \tag{16}$$

Simulations have shown that the performance of the algorithm does not change if we neglect the terms involving $\boldsymbol{D}_{N,T}$ and also if we set $\bar{A}_{N,T-1} \approx A_{N,T-1}$. With these assumptions, the resulting algorithm is given in Table II. We can see from this table that we need N additional multiplications for computing the backward error $r^p_N(T)$ plus another $2N$ for correcting the filter $B_{N,T-1}$. Notice also that we compute $k_N(T-1)$ as $\lambda^{-N}\gamma_N(T-1)$ instead of $\beta_N(T-1)/\alpha_N(T-1)$ (thus avoiding one division) since these two expressions are equivalent for the corresponding theoretical values. Even though we do not need to correct the filter $A_{N,T-1}$, we do correct the forward prior error. We have to point out that the algorithm defined in Table II is very sensitive. If we omit certain corrections or change the way of correcting the vectors, the resulting algorithm most probably will diverge. Notice also that all corrections depend on $\bar{\xi}_N(T)$ which is proportional to $\xi_N(T)$. Thus, all corrections become zero when $\xi_N(T)$ is zero (which is, for example, the case of infinite preci-

TABLE II
Stabilized FTF Algorithm (Complexity $10N$)

Available at time T:	$C_{N,T-1}, A_{N,T-1}, B_{N,T-1}, H_{N,T-1}, X_{N,T-1}$ $\gamma_N(T-1), \alpha_N(T-1), \beta_N(T-1)$
New Information:	$y(T), x(T)$

Computation of the variables $\xi_N(T), \bar{\xi}_N(T)$

$$e^p_N(T) = x(T) - A'_{N,T-1}X_{N,T-1}$$
$$r^p_N(T) = x(T-N) - B'_{N,T-1}X_{N,T}$$
$$\gamma_{N+1}(T) = \frac{\lambda\alpha_N(T-1)}{\lambda\alpha_N(T-1) + \gamma_N(T-1)\big(e^p_N(T)\big)^2}\gamma_N(T-1)$$
$$\gamma_N(T) = \Big\{1 + \gamma_{N+1}(T)r^p_N(T)\big(C^N_{N,T-1} + \frac{e^p_N(T)}{\lambda\alpha_N(T-1)}A^N_{N,T-1}\big)\Big\}^{-1}\gamma_{N+1}(T)$$
$$k_N(T-1) = \lambda^{-N}\gamma_N(T-1)$$
$$\xi_N(T) = r^p_N(T) + A^N_{N,T-1}k_{N,T-1}e^p_N(T) + \lambda\beta_N(T-1)C^N_{N,T-1}$$
$$\bar{\xi}_N(T) = \Big\{1 + \rho\big(\frac{1}{\gamma_N(T)} - 1\big) + \rho\big(A^N_{N,T-1}k_N(T-1)\big)^2\big(\frac{1}{\gamma_N(T-1)} - 1\big) + 2\rho A^N_{N,T-1}k^2_N(T-1)C^N_{N,T-1}e^p_N(T)\Big\}^{-1}\xi_N(T)$$

Correction of the Transversal Filter $B_{N,T-1}$

$$\bar{e}^p_N(T) = \big(1 - \rho k_N(T-1)C^N_{N,T-1}\bar{\xi}_N(T)\big)e^p_N(T) - \rho\lambda^{-N}A^N_{N,T-1}\big(1 - \gamma_N(T-1)\big)\bar{\xi}_N(T)$$
$$C_{N+1,T} = \begin{bmatrix} 0 \\ C_{N,T-1} \end{bmatrix} - \frac{\bar{e}^p_N(T)}{\lambda\alpha_N(T-1)}\begin{bmatrix} 1 \\ -A_{N,T-1} \end{bmatrix}$$
$$\begin{bmatrix} \bar{B}_{N,T-1} \\ -1 \end{bmatrix} = \frac{1}{1 + \rho C^{N+1}_{N+1,T}\bar{\xi}_N(T)}\Big\{\begin{bmatrix} B_{N,T-1} \\ -1 \end{bmatrix} - \rho\bar{\xi}_N(T)C_{N+1,T}\Big\}$$
$$\bar{r}^p_N(T) = r^p_N(T) - \rho\big(\frac{1}{\gamma_N(T)} - 1\big)\bar{\xi}_N(T)$$

Classical FTF algorithm

$$e_N(T) = \gamma_N(T-1)\bar{e}^p_N(T)$$
$$\alpha_N(T) = \lambda\alpha_N(T-1) + e_N(T)\bar{e}^p_N(T)$$
$$A_{N,T} = A_{N,T-1} - e_N(T)C_{N,T-1}$$
$$r_N(T) = \gamma_N(T)\bar{r}^p_N(T)$$
$$\beta_N(T) = \lambda\beta_N(T-1) + r_N(T)\bar{r}^p_N(T)$$
$$\begin{bmatrix} C_{N,T} \\ 0 \end{bmatrix} = C_{N+1,T} - C^{N+1}_{N+1,T}\begin{bmatrix} -\bar{B}_{N,T-1} \\ 1 \end{bmatrix}$$
$$B_{N,T} = \bar{B}_{N,T-1} - r_N(T)C_{N,T}$$

Filtering of the $y(T)$ signal.

$$\epsilon^p_N(T) = y(T) - H'_{N,T-1}X_{N,T}$$
$$\epsilon_N(T) = \gamma_N(T)\epsilon^p_N(T)$$
$$H_{N,T} = H_{N,T-1} - \epsilon_N(T)C_{N,T}$$

TABLE III
Stabilized Normalized FTF Algorithm (Complexity $12N$)

Available at time T:	$\hat{C}_{N,T-1}, \hat{A}_{N,T-1}, \hat{B}_{N,T-1}, H_{N,T-1}, X_{N,T-1}$ $\gamma_N^{1/2}(T-1)$
New Information:	$y(T), x(T)$

Computation of the variables $\xi_N(T)$, $\bar{\xi}_N(T)$.

$$\hat{e}^p_N(T) = \hat{A}'_{N,T-1}X_{N+1,T}$$
$$\hat{r}^p_N(T) = \hat{B}'_{N,T-1}X_{N+1,T}$$
$$v = \lambda^{-1/2}\gamma_N(T-1)^{1/2}\hat{e}^p_N(T)$$
$$\hat{e}^c_N(T) = \big(1 + v^2\big)^{-1/2}$$
$$\hat{e}_N(T) = v\hat{e}^c_N(T)$$
$$\gamma_{N+1}(T)^{1/2} = \hat{e}^c_N(T)\gamma_N(T-1)^{1/2}$$
$$\hat{r}_N(T) = \lambda^{-1/2}\gamma_{N+1}(T)^{1/2}\hat{r}^p_N(T)$$
$$\hat{r}^c_N(T) = \big(1 - \hat{r}^2_N(T)\big)^{-1/2}$$
$$\gamma_N(T)^{1/2} = \hat{r}^c_N(T)^{-1}\ \gamma_{N+1}(T)^{1/2}$$
$$\hat{\xi}_N(T) = \hat{r}_N(T) - \left(\frac{\hat{A}^{N+1}_{N,T-1}}{\hat{B}^{N+1}_{N,T-1}}\right)\hat{e}_N(T) + \lambda^{1/2}\left(\frac{\hat{C}^N_{N,T-1}}{\hat{B}^{N+1}_{N,T-1}}\right)\hat{e}^c_N(T)$$
$$\xi_N(T) = \lambda^{1/2}\gamma_{N+1}(T)^{-1/2}\left(\frac{\hat{\xi}_N(T)}{\hat{B}^{N+1}_{N,T-1}}\right)$$
$$k_N(T-1) = \left(\frac{\hat{A}^1_{N,T-1}}{\hat{B}^{N+1}_{N,T-1}}\right)^2$$
$$\bar{k}_N(T-1) = -\left(\frac{\hat{A}^{N+1}_{N,T-1}}{\hat{A}^1_{N,T-1}}\right)k_N(T-1)$$
$$\bar{\xi}_N(T) = \Big\{1 + \rho\big(\frac{1}{\gamma_N(T)} - 1\big) + \rho\big(\bar{k}_N(T-1)\big)^2\big(\frac{1}{\gamma_N(T-1)} - 1\big) + 2\rho k_N(T-1)\bar{k}_N(T-1)\gamma_N(T-1)^{-1/2}\hat{e}^p_N(T)\left(\frac{\hat{C}^N_{N,T-1}}{\hat{A}^1_{N,T-1}}\right)\Big\}^{-1}\xi_N(T)$$
$$\xi^A_N(T) = \rho k_N(T-1)\gamma_N(T-1)^{-1/2}\bar{\xi}_N(T)$$
$$\xi^B_N(T) = \rho\hat{B}^{N+1}_{N,T-1}\gamma_N(T)^{-1/2}\bar{\xi}_N(T)$$

Correction of normalized forward and backward errors.

$$\bar{\hat{e}}^p_N(T) = \big(1 - \hat{C}^N_{N,T-1}\xi^A_N(T)\big)\hat{e}^p_N(T) + \hat{A}^{N+1}_{N,T-1}\big(\gamma_N(T-1)^{-1/2} - \gamma_N(T-1)^{1/2}\big)\xi^A_N(T)$$
$$\bar{\hat{r}}^p_N(T) = \hat{r}^p_N(T) - \big(\gamma_N(T)^{-1/2} - \gamma_N(T)^{1/2}\big)\xi^B_N(T)$$

Classical FTF algorithm.

$$v = \lambda^{-1/2}\gamma_N(T-1)^{1/2}\bar{\hat{e}}^p_N(T)$$
$$\hat{e}^c_N(T) = \big(1 + v^2\big)^{-1/2}$$
$$\hat{e}_N(T) = v\hat{e}^c_N(T)$$
$$\gamma_{N+1}(T)^{-1/2} = \hat{e}^c_N(T)\gamma_N(T-1)^{-1/2}$$
$$\hat{C}_{N+1,T} = \big(\hat{e}^c_N(T) + \lambda^{-1/2}\hat{e}_N(T)\hat{A}^{N+1}_{N,T-1}\xi^A_N(T)\big)\begin{bmatrix} 0 \\ \hat{C}_{N,T-1} \end{bmatrix} - \lambda^{-1/2}\hat{e}_N(T)\hat{A}_{N,T-1}$$
$$\hat{A}_{N,T} = \lambda^{-1/2}\hat{e}^c_N(T)\hat{A}_{N,T-1}\big(\hat{e}_N(T) - \lambda^{-1/2}\hat{e}^c_N(T)\hat{A}^{N+1}_{N,T-1}\xi^A_N(T)\big)\begin{bmatrix} 0 \\ \hat{C}_{N,T-1} \end{bmatrix}$$
$$\hat{r}_N(T) = \lambda^{-1/2}\gamma_{N+1}(T)^{1/2}\bar{\hat{r}}^p_N(T)$$
$$\hat{r}^c_N(T) = \big(1 - \hat{r}^2_N(T)\big)^{1/2}$$
$$\gamma_N(T)^{1/2} = \big(\hat{r}^c_N(T)\big)^{-1}\gamma_{N+1}(T)^{1/2}$$
$$\begin{bmatrix} \hat{C}_{N,T} \\ 0 \end{bmatrix} = \frac{1}{\hat{r}^c_N(T) - \lambda^{-1/2}\hat{r}_N(T)\xi^B_N(T)}\big(\hat{C}_{N+1,T} + \lambda^{-1/2}\hat{r}_N(T)\hat{B}_{N,T-1}\big)$$
$$\hat{B}_{N,T} = \lambda^{-1/2}\hat{r}^c_N(T)\hat{B}_{N,T-1} + \big(\hat{r}_N(T) + \lambda^{-1/2}\hat{r}^c_N(T)\xi^B_N(T)\big)\begin{bmatrix} \hat{C}_{N,T} \\ 0 \end{bmatrix}$$

Filtering of the $y(T)$ signal.

$$\epsilon^p_N(T) = y(T) - H'_{N,T-1}X_{N,T}$$
$$H_{N,T} = H_{N,T-1} - \big(\gamma_N(T)^{1/2}\epsilon^p_N(T)\big)\hat{C}_{N,T}$$

sion). Finally, notice that we modify the variables of the original algorithm at every time instant.

Since we do not prove stability of our algorithm in any sense, there is always the risk of divergence. Thus, we still need a divergence detector. From Fig. 1 we can see that $\gamma_N(T)$ (used in [5] as divergence detector) does not

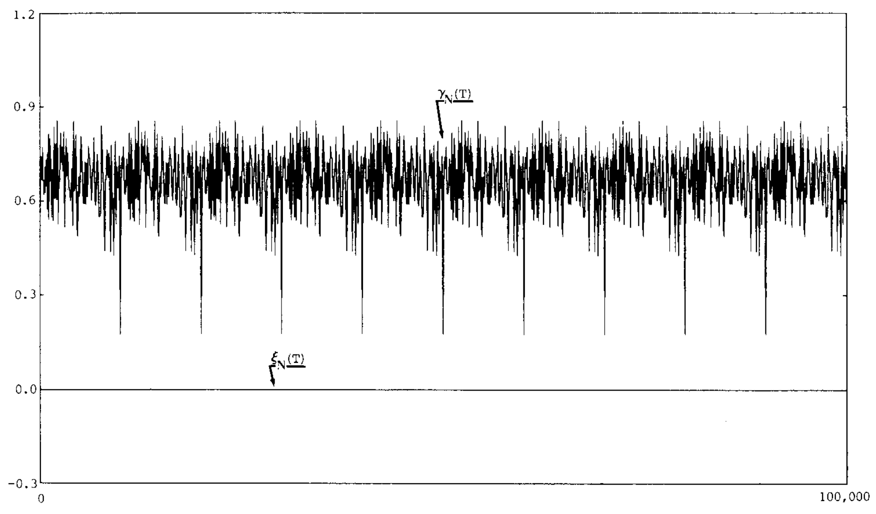


Fig. 2. Typical performance of the variables $\gamma_N(T)$ and $\xi_N(T)$ for the stabilized FTF algorithm.

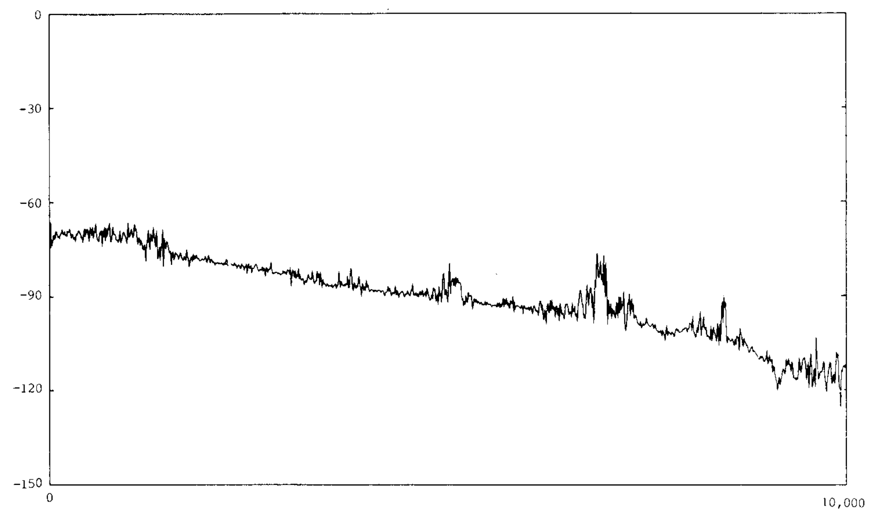


Fig. 3. Relative difference (in decibels) of the Kalman Gains computed by RLS and by the SFTF algorithm of Table II.

really show the accumulation of the roundoff errors (since any nonnegative value less than unity is acceptable). The quantity $\gamma_N(T)$ suddenly becomes larger than unity or negative and this is used in [5] as a criterion for reinitialization. It is possible especially when we have divergence toward zero (in which case $\gamma_N(T) \rightarrow 0$) to pass a large amount of time before this detector detects divergence. We propose here the following detector: reinitialize whenever $\xi_N^2(T) > t\beta_N(T)$, where t is a small constant. By selecting t properly, we can detect divergence earlier than $\gamma_N(T)$ and also both modes.

Before going to simulations, we will briefly say a few things about the stabilization of the normalized FTF (NFTF) algorithm. It can be shown [2] that the corrections in (9) and (10) can be included in the NFTF of [5] without significantly increasing the complexity of the algorithm. We will only need N additional multiplications for the computation of the backward prior error (raising the total from $11N$ to $12N$) plus an additional constant number of multiplications. The reason is that we can bypass the computation of the corrections of the two filters $\hat{A}_{N,T-1}$ and $\hat{B}_{N,T-1}$ by directly expressing $\hat{A}_{N,T}$ and $\hat{B}_{N,T}$ (the normalized versions of the corresponding filters), in terms of $\hat{A}_{N,T-1}$ and $\hat{B}_{N,T-1}$. The resulting algorithm is presented in Table III.

IV. Simulations

In this section we present computer simulations for a 10th-order filter. We use $\lambda = 0.96$, $\rho = 1$, and, as input sequence $\{x(T)\}$, a pseudowhite centered Gaussian noise of unit variance. We initialize the algorithms as in [5] with $\alpha_N(0) = \mu\lambda^N$, $\beta_N(0) = \mu$, $\gamma_N(0) = 1$, with $\mu = 0.1$. The constant μ is also used for the reinitialization of the normal FTF algorithm as described in [5]. The simulations were performed with a Personal Computer and with 32 bit floating-point arithmetic. In Fig. 1 we can see a typical behavior of the FTF algorithm of complexity $8N$ (the algorithm of Table I, but with $r_N^p(T)$ computed as $x(T - N) - B'_{N,T-1}X_{N,T}$). Every time $\gamma_N(T)$ exceeds unity, the algorithm is reinitialized. Notice that we have performed simulation only up to 10 000 points. The corresponding FTF of complexity $7N$ has two times as many reinitializations. We can also see the behavior of the variable $\xi_N(T)$; its absolute magnitude increases with time as the roundoff errors accumulate. In Fig. 2 we have the performance of our algorithm defined in Table II. Here we have performed a 100 000 points simulation. At $\{x(T)\}$ we were recycling a 10 000 samples pseudowhite Gaussian noise sequence. This is the reason why the graph seems periodic. We can see that our algorithm behaves in a much more stable way and that $\xi_N(T)$ is practically zero. As we said in the Introduction, having a stable algorithm does not necessarily mean that we have an LS algorithm. In order to check this point, in Fig. 3 we plot $\| C_{SFTF} - C_{RLS} \| / \| C_{RLS} \|$ (in decibels), the relative difference of the Dual Kalman Gains computed by our Stabilized FTF (SFTF) algorithm and by RLS. We can see that they are very close to each other and that their relative distance decreases with time. This means that the modifications introduced here are such that the resulting algorithm, to all practical considerations, solves the LS problem at every time instant.

V. Conclusions

In this paper we have presented a method for stabilizing Fast Kalman Algorithms. The method was presented with an example by stabilizing the FTF algorithm. We derived a version of the FTF algorithm that has a much more stable behavior compared to the normal FTF. In all the simulations we have performed, this new version did not diverge even for as many as 500 000 points. The price we pay for this performance is another $3N$ multiplications for the unnormalized case, raising the total from $7N$ to $10N$, and only N for the normalized case, raising the total from $11N$ to $12N$. We must also point out that we have not tried to optimize the computations in any sense. It might thus be possible to reduce further the complexity below $10N$. Of course, with the idea used here, it is impossible to go below $8N$ since we always need to compute the backward prior error using N multiplications. For our algorithm we need to define the constant ρ. This constant, as we can see from (9) and (10), is directly related to the amount of correction that we impose on the two filters. It is clear that if it has a small value, then the algorithm will diverge. The same will happen if the value is too large. For orders around 10 or 20, a good selection is $\rho = 1$. For larger orders, a larger value of ρ is required.

References

[1] M. Bellanger, "Engineering aspects of fast least squares algorithms in transversal adaptive filters," in *Proc. ICASSP*, Dallas, TX, 1987, Paper 49.15.

[2] J. L. Botto, "Etude des algorithmes transversaux rapides: Application a l' annulation d' echo acoustiques pour l' audioconférence," These de Docteur Ingénieur, Univerité de Rennes I, IRISA, France, May 1986.

[3] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least squares filtering and prediction," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1394-1402, Dec. 1983.

[4] ——, "Fast Kalman-type algorithms for sequential signal processing," in *Proc. ICASSP*, Boston, MA, 1983, pp. 186-189.

[5] J. M. Cioffi and T. Kailath, "Fast recursive least-squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 304-337, Apr. 1984.

[6] ——, "Windowed fast transversal filters adaptive algorithms with normalization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 607-625, June 1985.

[7] J. M. Cioffi, "Limited precision effects in adaptive filtering," *IEEE Trans. Circuits Syst.*, Special Issue on Adaptive Filtering, vol. CAS-34, pp. 1097-1110, July 1987.

[8] P. Fabre and C. Gueguen, "Fast recursive least-squares algorithms: Preventing divergence," in *Proc. ICASSP*, Tampa, FL, Mar. 1985, Paper 30.2.

[9] ——, "Improvement of fast RLS algorithms via normalization: A comparative study," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 296-308, Apr. 1986.

[10] D. D. Falconer and L. Ljung, "Application of fast Kalman estimation to adaptive equalization," *IEEE Trans. Commun.*, vol. COM-26, pp. 1439-1446, Oct. 1987.

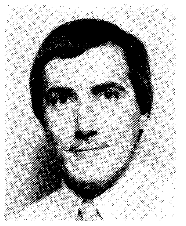
[11] D. W. Lin, "On digital implementation of the fast Kalman algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 998-1005, Oct. 1984.

[12] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*. Cambridge, MA: M.I.T. Press, 1983.

[13] L. Ljung, M. Morf, and D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," *Int. J. Contr.*, vol. 27, pp. 1-19, Jan. 1978.

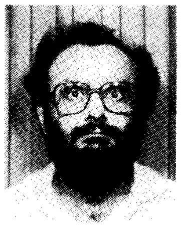
[14] S. Ljung and L. Ljung, "Error propagation properties of recursive least-squares adaptation algorithms," *Automatica*, vol. 21, no. 2, pp. 157-167, 1985.

[15] D. Manolakis, G. Carayannis, and N. Kalouptsidis, "New issues on the computational organization of fast sequential algorithms," in *Proc. ICASSP*, San Diego, CA, 1984, Paper 43.7.



Jean-Luc Botto was born on November 10, 1959. He received the Engineer degree from the Ecole Nationale Supérieure des Télécommunications, Paris, France, in 1982, and the Doctor degree from the University of Rennes in 1986. He received the award for the best thesis in automatic control for his work on acoustic echo cancellation and fast Kalman algorithms in 1987.

He joined the Department of Data Transmission Télécommunications Radioélectriques et Téléphoniques (T.R.T.) Company, Le Plessis-Robinson, France, in January 1986, where he worked on high-speed modem research and development. His area of interest is signal processing for high speed equalization, and ISDN products development.



George V. Moustakides (S'79-M'83) was born in Drama, Greece, on April 16, 1955. He received the diploma in electrical engineering from the National Technical University of Athens, Greece, the M.Sc. degree in systems engineering from the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, in 1979, 1980, and 1983, respectively.

From 1983 to 1986 he held a research position at the Institut de Recherche en Informatique et Systèmes Aléatoires, Rennes, France. From 1986 to 1988 he was with the Greek Air Force for his military service. He is currently with the Computer Technology Institute, Patras, Greece. His interests include detection of signals, detection of changes in systems, fast estimation algorithms, and theory of optimal stopping times.