EURASIP Journal on Applied Signal Processing 2001:3, 169–180

Quantization Effects and Stabilization of the Fast-Kalman Algorithm

Constantin Papaodysseus

Department of Electrical and Computer Engineering, Division of Communications, Electronics and Information Systems, National Technical University of Athens, GR-15773 Athens, Greece
Email: cpapaod@cs.ntua.gr

Constantin Alexiou

Department of Electrical and Computer Engineering, Division of Communications, Electronics and Information Systems, National Technical University of Athens, GR-15773 Athens, Greece
Email: kalex@cs.ntua.gr

George Roussopoulos

Department of Electrical and Computer Engineering, Division of Communications, Electronics and Information Systems, National Technical University of Athens, GR-15773 Athens, Greece
Email: roussop@cs.ntua.gr

Athanasios Panagopoulos

Department of Electrical and Computer Engineering, Division of Communications, Electronics and Information Systems, National Technical University of Athens, GR-15773 Athens, Greece
Email: thpanag@cs.ntua.gr

Received 1 November 2000 and in revised form 23 June 2001

The exact and actual cause of the failure of the fast-Kalman algorithm due to the generation and propagation of finite-precision or quantization error is presented. It is demonstrated that out of all the formulas that constitute this fast Recursive Least Squares (RLS) scheme only three generate an amount of finite-precision error that consistently propagates in the subsequent iterations and eventually makes the algorithm fail after a certain number of recursions. Moreover, it is shown that there is a very limited number of specific formulas that transmit the generated finite-precision error, while there is another class of formulas that lift or "relax" this error. In addition, a number of general propositions is presented that allow for the calculation of the exact number of erroneous digits with which the various quantities of the fast-Kalman scheme are computed, including the filter coefficients. On the basis of the previous analysis a method of stabilization of the fast-Kalman algorithm is developed and is presented here, a method that allows for the fast-Kalman algorithm to follow very difficult signals such as music, speech, environmental noise, and other nonstationary ones. Finally, a general methodology is pointed out, that allows for the development of new algorithms which, intrinsically, suffer far less of finite-precision problems.

Keywords and phrases: Kalman filtering, recursive least squares filtering, adaptive algorithms, quantization error in fast-Kalman algorithm, finite-precision error in RLS algorithms.

1. INTRODUCTION

Adaptive filtering, by means of Recursive Least Squares (RLS) algorithms, finds an exceedingly large number of applications in many areas of automatic control and signal processing, as for example in adaptive control, in model based fast process fault diagnosis, in system identification, stochastic control, adaptive differential encoding and deconvolution, echo cancellation and channel equalization, in line and image enhancement, biological signal processing, frequency domain adaptive filtering, and so forth, (see [1, 2, 3]).

Various fast (i.e., of $O(m)$ computational complexity) algorithms that perform an RLS adaptive filtering by means of an infinite window have been developed (see [1, 4, 5, 6, 7], etc.). Most of these schemes use the fruitful concept of rank displacement, introduced by Morf [8], and suffer from serious numerical problems, due to the finite-precision with which all calculations are made. Actually, it has been demonstrated in

practice, that the faster algorithms, for example, the FAEST 5p [5] introduced by Carayannis, Manolakis, and Kalouptsides or the original FTF version proposed by Cioffi and Kailath [9], have the worst numerical behaviour and fail very quickly. However, even the first fast-Kalman algorithm introduced by Ljung, Morf, and Falconer [4] fails quickly enough, in the sense that it produces completely unreliable results after a fairly small number of iterations (see also [1, 10, 11], etc., and Section 3 of the present work).

A number of authors have dealt with the problem of numerical instability and error propagation concerning the RLS schemes, and/or other related algorithms, as for example, Fabre and Gueguen [1], Ljung and Ljung [10], Lin [12], Botto and Moustakides [13], Glaros and Carayannis [14], and so forth. Slock and Kailath in a well-known paper [9] have proposed a method based on feedback schemes, for stabilizing the numerical behaviour of the fast transversal algorithm.

In this paper, the finite-precision error generation and propagation in the fast-Kalman scheme [4] is examined on the basis of a set of newly introduced principles and by means of an entirely new methodology and philosophy, which are drastically different from the ones undertaken in all the previous aforementioned papers. A primary form of this new methodology and of the related introduced principles, has been used by the authors in [15, 16, 17, 18, 19], in connection with other iterative DSP algorithms. The application of those new principles and of the related methodology allows to find the deeper, actual, and exact cause of the numerical problems and of the numerical instability due to the finite-precision, in the fast-Kalman algorithm; it offers, moreover, the considerable advantage of having an accurate knowledge of the amount of the finite-precision error generated at each recursion, as well as a thorough knowledge of the reliability of the results offered by the algorithm at each iteration. But perhaps most important of all, the knowledge of the exact and deeper cause of the finite-precision problems and of the eventual numerical failure of the fast RLS schemes, shows the way for the construction of algorithms which, intrinsically, suffer far less from finite-precision problems, as well as for the development of new methods for stabilizing algorithms.

2. GENERAL REMARKS

2.1. Some necessary definitions and the notation used

The propositions stated in this paper hold true independently of the radix of the arithmetic system. However, the numerical error generation and propagation will be studied in the decimal representation, because the decimal arithmetic system is far more familiar and clear to the user. In this system, precision comparison between two numbers will be made using the following definitions.

Definition 1. Consider two numbers, n_1, n_2, of the same sign, written in canonical exponential form, with the same number n of decimal digits in the mantissa, that is,

$$\begin{aligned} n_1 &= d_1 \cdot d_2 \cdot d_3 \cdots d_n \times 10^{\tau}, \\ n_2 &= \delta_1 \cdot \delta_2 \cdot \delta_3 \cdots \delta_n \times 10^{\rho}, \\ \tau &\geq \rho. \end{aligned} \tag{1}$$

Then, these two numbers differ by K decimal digits, if and only if

$$\big|\,|n_1| - |n_2|\,\big| = d \times 10^{\tau-(n-k)}, \quad 1 \leq d < 10. \tag{2}$$

Notation 2. For any quantity α expressed in canonical exponential form we write

(i) $\mathrm{man}(\alpha)$ for the mantissa of α,
(ii) $E(\alpha)$ for the exponent of α.

Notation 3. The abbreviation e.d.d. stands for "erroneous decimal digits."

Notation 4. The abbreviation f.p.e. stands for "finite-precision error."

Notation 5. The expression "the algorithm fails" means that (at the iteration at hand and all the subsequent ones) the specific algorithm gives totally unreliable results.

Definition 6. Let all the quantities be written in canonical exponential form, with n decimal digits in the mantissa. Suppose that the correct value an arbitrary quantity α should have, if all calculations were made with infinite precision, is α_c. Then one may define that the quantity α has been computed with precisely the last λ decimal digits erroneous if and only if α and α_c differ by λ digits according to Definition 1.

2.2. Error generation, propagation, or relaxation in a set of fundamental operations

Some of the propositions presented in this paper can be found in [15, 16, 18]. However, they are stated here too, for completeness and easy reference.

Proposition 7. *Let all the involved quantities be computed with finite-precision of n decimal digits in the mantissa, and consider any quantity computed through*

$$x = y \cdot z. \tag{3}$$

Suppose that, due to the previous finite-precision calculations, the quantity y has been computed with precisely the last λ e.d.d., while z has been computed with up to λ e.d.d. In this case the following hold:

(i) *If*

$$\big|\mathrm{man}(y) \cdot \mathrm{man}(z)\big| \geq 10, \tag{4}$$

then x is computed with precisely the last λ or $\lambda - 1$ e.d.d.

(ii) *If*

$$\big|\mathrm{man}(y) \cdot \mathrm{man}(z)\big| < 10, \tag{5}$$

then x is computed with precisely the last λ or $\lambda + 1$ e.d.d.

Proposition 8. *Let all the following quantities be computed with finite-precision of n decimal digits in the mantissa, and consider any quantity computed through a formula of the type*

$$k = \frac{\beta}{\alpha}. \tag{6}$$

Suppose that, due to the previous finite-precision calculations, the quantity β has been computed with precisely the last 1 e.d.d., while α has been computed with up to 1 e.d.d.

Then the following hold:

(i) *If $|\text{man}(\beta)/\text{man}(\alpha)| \geq 1$, then k is computed with precisely the last λ or $\lambda - 1$ e.d.d.*

(ii) *If $|\text{man}(\beta)/\text{man}(\alpha)| < 1$, then k is computed with precisely the last $\lambda + 1$ or λ e.d.d.*

The following proposition will play a crucial role in all the subsequent analysis.

Proposition 9. *Let all the involved quantities be computed with finite-precision of n decimal digits in the mantissa, and consider any quantity calculated through a formula of the type*

$$w = t - r, \qquad t \cdot r > 0, \tag{7}$$

where

$$\begin{aligned} w &= w_1 \cdot w_2 \cdots w_n \times 10^{\delta}, \\ t &= t_1 \cdot t_2 \cdots t_n \times 10^{\tau}, \\ r &= r_1 \cdot r_2 \cdots r_n \times 10^{\rho}, \end{aligned} \tag{8}$$

are such that

$$\delta < \max\{\tau, \rho\} \Longleftrightarrow E(w) < \max\{E(t), E(r)\}. \tag{9}$$

Moreover, let

$$d = |\max\{\tau, \rho\} - \delta| \tag{10}$$

and suppose that, due to the previous finite-precision calculations, the higher order quantity has been computed with precisely the last λ e.d.d., while τ has been computed with a number of e.d.d. equal to or smaller than λ.

Then w is computed with the last $(\lambda + d)$ e.d.d.

Most of the proofs of these propositions can be found in [15, 16].

Proposition 10 (the numerical error relaxation shift). *Let all the involved quantities be computed with finite-precision of n decimal digits in the mantissa, and consider any quantity x computed through a sum of two quantities, that is,*

$$x = y + z. \tag{11}$$

Suppose moreover that z has its last λ e.d.d. and that the exponent of z is by v smaller than the exponent of y, that is,

$$E(y) > E(z), \tag{12}$$

$$v = E(y) - E(z). \tag{13}$$

Then z transfers to x only $\lambda - v$ e.d.d. if $\lambda - v > 0$, or no e.d.d. at all if $\lambda - v < 0$.

Proof. In finite-precision, the addition of the mantissa of y and z, because of the inequality (12) and of the entailed necessary corresponding shift, takes place as follows:

$$\text{man}(y) + 0 \cdot \underbrace{000}_{\substack{v \text{ decimal} \\ \text{places}}} z_1 z_2 \cdots z_{n-v}. \tag{14}$$

Hence, only $\lambda - v$ e.d.d. participate in this addition and they are eventually transferred to the quantity x if $(\lambda - v) > 0$, or no e.d.d. at all if $(\lambda - v) < 0$. □

2.3. Some general results concerning the finite-precision error generated in any recursive algorithm

As will become evident from the subsequent analysis too, all the formulas that constitute a certain iterative algorithm are not equivalent from the point of view of the finite-precision error generation and propagation. On the contrary, in all the recursive computational schemes that have been examined so far (see also [15, 16, 17, 18, 19]) it has been asserted that

(a) There is a limited number of specific formulas, each of which generates the greatest and most decisive part of the numerical error due to the finite word length (for a more extensive analysis see Section 2.4). We will refer to each one of those formulas, with the generic name "Main Source of Finite-Precision Error."

(b) Similarly, there is a class of specific formulas that "transmit" the generated quantization error.

(c) On the other hand, there is another class of formulas that "relax" the f.p.e. that has been generated elsewhere (see Section 2.4).

Clearly, which formula precisely belong to the class (a), (b), or (c), depends on the exact form of the computational scheme in hand, and on the exact nature of the input data each time.

2.4. Understanding and classifying the role of each formula in the quantization effects

Propositions 7, 8, 9, and 10 may offer a clear understanding of the way with which each formula generates quantization error. Moreover, they dynamically offer exact knowledge of the generated number of e.d.d., up to ± 1 digit. However, we must stress that knowledge of the amount of f.p.e., generated by a single formula alone, is not sufficient for understanding, explaining, and classifying its role in the overall quantization effects in an algorithm. For example, suppose that all operations are made with single (float) precision and that the outcome of an operation is the number $x = 1.2\mathbf{345678} \times 10^{-5}$, computed with six e.d.d., shown in bold, where four of these erroneous digits have been generated from the operation in hand, say for the reasons described in Proposition 9. If, in the immediately subsequent operation in the algorithm execution, number x is added to the error-free number $y = 8.7654321 \times 10^{1}$, then the outcome of this operation will be $y = 8.7654332 \times 10^{1}$, which is error free, too.

If this sequence of events occurs statistically often, namely whenever the first operation generates quantization error, the second one "relaxes" it, then the first operation is not a main source of f.p.e.

Therefore, consider any algorithm used for the computation of a set of quantities Q, which is the output obtained by the user at various instances of its execution. To set ideas, for the fast-Kalman algorithm considered in this paper, Q is the set consisting of the filter coefficients c_m, obtained at the end of each iteration. We will use for set Q the name "desired final output," too. By definition, a main source of f.p.e. for this algorithm, is a formula which generates an amount of f.p.e. that directly influences the quantization error of the desired final output systematically. In order to unambiguously test if a certain formula is a main source of quantization error the following technique has been developed.

First, we have developed a software tool for performing the four fundamental operations with arbitrary precision, which may each time be chosen by the user. We have executed the fast-Kalman algorithm first with standard float precision (using about 8 decimal digits in the mantissa) and then, employing the developed tool, we have executed it with any precision using more than 2×8 decimal digits in the mantissa, say with fifty digits. We must stress that for all employed precisions, the very same input sequence has been used. Then, for each quantity, say x, of the algorithm the following three representations have been obtained:

(a) x_8 resulting by executing the algorithm with float precision.

(b) x_{50} resulting by executing the algorithm with fifty decimal digits in the mantissa.

(c) Next, by keeping the eight most significant decimal digits of the mantissa and the proper exponent of the quantity x_{50}, representation $\bar{x}_8$ resulted.

Now, since finite-precision error is generated and propagates from the least significant digit of the mantissa to the most significant one, applying Definition 1 to x_8 and $\bar{x}_8$, one obtains a number K. As far as this number K is smaller than eight, then one may safely state that K is the number of e.d.d. with which quantity x has been generated at this specific algorithm instant. Moreover, if $K < 8$, in other words as far as the algorithm with float execution is not destroyed, one safely concludes that the number $\bar{x}_8$ is an error-free eight digit representation of quantity x.

Subsequently, in order to classify an operation, with quantity x as outcome, from the quantization effects point of view, one may proceed as follows:

(1) One replaces x_8 with $\bar{x}_8$ immediately after the operation execution, simulating the case where the operation in hand offered error-free results.

(2) One computes the exact number of e.d.d. with which the desired final output of the whole algorithm is computed. To set ideas, for the fast-Kalman algorithm, one computes the exact number of e.d.d. with which the filter coefficients c_i, $i = 1, 2, K, m$, where m is the filter order, are computed.

(3) If the desired algorithm final output suffers from the same quantization effects, then one may state that this operation is not a main source of finite-precision error. On the contrary, if (a) the operation in hand statistically generates an output with a number of e.d.d. greater than the maximum number of e.d.d. of its input operands and (b) the substitution of x_8 with $\bar{x}_8$, drastically reduces the number of e.d.d. with which the desired algorithm output is computed at the corresponding iteration, then one states that the operation in hand is a main source of finite-precision error. By definition, the formula that includes this operation is a main source, too.

In an analogous way, one may characterize a formula as a "transmitter" of f.p.e. when, statistically, the number of e.d.d. of its outcome equals the maximum number of e.d.d. of its input operands. When a formula, statistically, reduces the maximum number of e.d.d. of its input operands, then one may characterize it as a "relaxation" formula.

We must stress once more that spotting the actual and deeper cause of each formula behaviour from the quantization effects point of view, demands employing Propositions 7, 8, 9, and 10.

3. ANALYSIS OF THE WAY THE FINITE-PRECISION ERROR IS GENERATED AND PROPAGATED IN THE FAST-KALMAN ALGORITHM, WHEN THE INPUT SEQUENCE $x(n+1)$ IS A WHITE NOISE

The fast-Kalman algorithm is intrinsically unstable, in the sense that in any case, will fail completely after a fairly limited number of iterations. The exact number of recursions after which all the obtained results become totally unreliable, depends on various parameters such as: the nature of the input sequence, the value of the forgetting factor λ, the initial conditions that determine the speed of convergence, the ratio of the maximum (or the mean) value of the output $z(n)$ by the maximum (or the mean) value of the input $x(n)$ (i.e., the "SNR"), the used arithmetic system, and so forth. However, when standard IEEE arithmetics is used, with $\lambda = 0.97$ and SNR in the interval [10 dB, 100 dB], then, one may safely state that, when the input sequence is a white noise or a periodic function, the number of recursions after which the fast-Kalman algorithm totally fails, typically belongs to the interval [3600, 4000], reaching, rarely, under very favorable conditions the 4500 recursions.

3.1. *The main sources of numerical error in this algorithm, for a white noise input*

Out of all the formulas that constitute the fast-Kalman algorithm (see Appendix A for its description, where the notation introduced in [5] is used), only three are the main sources of numerical error, that is, only three formulas generate consistently an amount of finite-precision error that propagates in the other formulas and the subsequent iterations and eventually makes the algorithm fail after a certain number of recursions. These are the following formulae.

(1) The formula (A.3) that computes $\varepsilon_m^f(n+1)$.

(2) The formula (A.5) that performs the computation of $\mathbf{w}_{m+1}^*(n+1)$, together with (A.8) that computes $\mathbf{w}_m^*(n+1)$. Actually those two formulas act as a strongly interrelated couple, as far as the generation and propagation of the finite-

precision error is concerned. In fact, the formula that computes $\mathbf{w}^*_{m+1}(n+1)$ is the main source of f.p.e., while the one for the $\mathbf{w}^*_m(n+1)$ computation fully transmits the f.p.e. generated into the other formula (A.5).

(3) The formula (A.9) for the computation of $\mathbf{b}_m(n+1)$. The error generated by this formula is transmitted to the rest of the algorithm, mainly by means of (A.7) that computes the quantity $e^b_m(n+1)$.

A more extensive analysis of the exact role of each one of these three main sources of f.p.e. error follows immediately.

3.2. Finite-precision error generated in the computation of $\varepsilon^f_m(n+1)$

The analysis of the finite-precision error generation and propagation in connection with the formula that computes $\varepsilon^f_m(n+1)$ may, quite clearly, demonstrate the meaning of the terms "main source of numerical error," "f.p.e. transmitter," and "formula that relaxes the generated f.p.e. error." In fact, prior to $\varepsilon^f_m(n+1)$, consider the $e^f_m(n+1)$ computation. Then, it is possible that at a certain iteration, say the kth iteration, the two terms in the right-hand side of (A.1) that computes $e^f_m(k)$ (1) are of opposite sign and (2) have a number of digits in common, in the sense of Definition 1.

Therefore, at this particular iteration, $e^f_m(k)$ is computed with a corresponding number of e.d.d. These e.d.d., however, do not propagate essentially in the subsequent steps of the algorithm, nor in the following iterations, due to

(a) the relaxation shifts that take place in the performance of the operations in the right-hand sides of (A.2) and (A.4) and

(b) the fact that there is no systematic reason that makes conditions (1) and (2) occur; or, equivalently, there is no systematic reason that forces $e^f_m(n+1)$ to be generated with a certain number of e.d.d. statistically regularly.

On the contrary, for the $\varepsilon^f_m(n+1)$ it holds that

$$\varepsilon^f_m(n+1) = \frac{e^f_m(n+1)}{a_m(n)}, \tag{15}$$

$$a_m(n) = 1 - \mathbf{x}^T_m(n)\mathbf{w}_m(n). \tag{16}$$

But the demand that the system matrix $\mathbf{R}_m(n+1)$ must be positive definite for all time instants n implies, in a straightforward way, that

$$1 < a_m(n) \tag{17}$$

must always hold. Therefore, due to (16), (17), and the specific nature of the input $x(n+1)$, it follows that, statistically, in the computation of $\varepsilon^f_m(n+1)$ by means of

$$\varepsilon^f_m(n+1) = x(n+1) + \mathbf{a}^T_m(n+1)\mathbf{x}_m(n), \tag{18}$$

the exponent of the quantity $\varepsilon^f_m(n+1)$ is repeatedly lower than the maximum of the exponents of $x(n+1)$ and $\mathbf{a}^T_m(n+1)\mathbf{x}_m(n)$. In other words, mainly due to the positive definiteness requirement, it holds that in the right-hand side of (18), the two terms $x(n+1)$ on the one hand and $\mathbf{a}^T_m(n+1)\mathbf{x}_m(n)$ on the other, statistically, (i) are of opposite sign and (ii) have a number of digits in common, in the sense of Definition 1.

Therefore, according to Proposition 9, it follows that at each iteration where the above conditions (i) and (ii) occur, $\varepsilon^f_m(n+1)$ is computed with k additional e.d.d., with $k = p - \tau$, where τ is the exponent of $\varepsilon^f_m(n+1)$, that is,

$$\tau = E(\varepsilon^f_m(n+1)) \tag{19}$$

and p is the maximum of the exponents of $x(n+1)$ and $\mathbf{a}^T_m(n+1)\mathbf{x}_m(n)$, that is,

$$p = \max\{E(x(n+1)), E(\mathbf{a}^T_m(n+1)\mathbf{x}_m(n))\}. \tag{20}$$

It is worthwhile noticing that it is not necessary for the two quantities $x(n+1)$ and $\mathbf{a}^T_m(n+1)\mathbf{x}_m(n)$ to satisfy conditions (i) and (ii) at each iteration; actually, they do not. However, due to the requirement of the positive definiteness of the system matrix $\mathbf{R}_m(n)$ for every n, conditions (i) and (ii) are, statistically, bound to hold at a certain number of iterations. Therefore, at each such iteration an additional amount of finite-precision error is generated and accumulated in $\varepsilon^f_m(n+1)$; this error is transferred to the Kalman gain of order $m+1$, $\mathbf{w}^*_{m+1}(n+1)$ and then, via formula (A.8), to the Kalman gain of order m and subsequently to all the quantities of the algorithm and in particular to the filter coefficients $\mathbf{c}_m(n+1)$.

One may recursively compute the exact, practically up to ± 1, number of e.d.d. with which the various quantities are computed owing exclusively to the finite-precision error generated in formula (A.3), if one uses the previous results and Propositions 7, 8, 9, and 10.

3.3. Numerical error due to finite word length generated by the two formulas for the computation of the Kalman gains $\mathbf{w}^*_{m+1}(n+1)$ and $\mathbf{w}^*_m(n+1)$ together, considered as a unity

Each one of the two formulas (A.5) and (A.8), standing alone, is a source of finite-precision error. The first formula that computes $\mathbf{w}^*_{m+1}(n+1)$ is a comparatively much stronger source of f.p.e., while the one that computes $\mathbf{w}^*_m(n+1)$ is a much more feeble such source. However, the amount of finite-precision error generated at each iteration by one of those two formulas can be propagated only by the other, complementary, formula. To set ideas, suppose that at a certain iteration all the quantities $w^*_{j,m+1}(n+1)$ are generated with one e.d.d., due to the phenomenon described in Proposition 9. If this numerical error is relaxed in the computation of $\mathbf{w}^*_m(n+1)$ by means of the relaxation shift described in Proposition 10, then clearly this error does not propagate in the subsequent iterations. If, on the contrary, the f.p.e. generated in the computation of $\mathbf{w}^*_{m+1}(n+1)$ is actually transmitted to $\mathbf{w}^*_m(n+1)$ by means of formula (A.8), then in the subsequent $(n+2)$th iteration, $\mathbf{w}^*_{m+1}(n+2)$ will be calculated with at least the same amount of finite-precision error, due to the presence of $\mathbf{w}^*_m(n+1)$ in formula (A.5) and due to the fact

that, statistically, the term $[\varepsilon_m^f(n+1)/\alpha_m^f(n+1)]\mathbf{a}_m(n+1)$ is of a smaller order than $\mathbf{w}_m^*(n+1)$.

Now, when the input sequence is a white noise, then it follows that, statistically, in the computation of $w_{j,m+1}^*(n+1)$ the conditions of Proposition 9 often hold indeed, and hence a corresponding amount of f.p.e. is frequently generated, and usually the one corresponding to one or two e.d.d. In other words, the formula for the computation of $\mathbf{w}_{m+1}^*(n+1)$ is a consistent source of finite-precision error. This numerical error is, statistically, fully transmitted to $\mathbf{w}_m^*(n+1)$, since in the numerator of the right-hand side of (A.8), participate all the components of $\mathbf{w}_{m+1}^*(n+1)$.

Similarly, the f.p.e. generated in the computation of $w_{j,m}^*(n+1)$, according to Proposition 9, can be transmitted to the subsequent recursion, by means of formula (A.5), and then, normally, to all the other formulas and iterations. We shall stress once more that (A.5) that computes $\mathbf{w}_{m+1}^*(n+1)$ is a much more crucial and decisive source of f.p.e., while formula (A.8) that computes $\mathbf{w}_m^*(n+1)$ acts mainly as a good and crucial transmitter of the numerical error generated by the complementary formula (A.5).

Once more, it is possible to make a theoretical prediction of the exact, practically up to ± 1, number of e.d.d. with which the various quantities of the fast-Kalman algorithm are computed due exclusively to the quantization error generated in formula (A.5), on the basis of the previous analysis and the one of Section 2.2.

3.4. Finite-precision error generated in the computation of $\mathbf{b}_m(n+1)$

Along very similar lines with Section 3.3 it may be shown that, when the input data is a white noise, then statistically, the two terms $b_{j,m}(n)$ and $w_{j,m}^*(n+1)^* e_m^b(n+1)$ in the right-hand side of (A.9) satisfy the conditions of Proposition 9, hence, a corresponding amount of finite-precision error is generated in the computation of $\mathbf{b}_m(n+1)$. This numerical error is normally transmitted to $\mathbf{w}_m^*(n+2)$, via formulas (A.7) and (A.8) and hereafter to all the quantities of the algorithm.

The number of e.d.d. with which $\mathbf{b}_m(n+1)$ is computed is not drastically great; this means that the formula that calculates $\mathbf{b}_m(n+1)$ is not a dominant source of f.p.e. However, if one wants to develop an error-free RLS algorithm, one must, absolutely, render this or the corresponding formula error-free too, because, if the other two sources of quantization error were made error-free, then, still, the error generated by this formula (A.9) can destroy the algorithm.

3.5. The role of the relaxation shift in the error generation and propagation of the fast-Kalman algorithm

It is worthwhile noticing that the serious numerical error generated in the $\varepsilon_m^f(n+1)$ computation, that has been examined in Section 3.2, does not affect the value of the quantity $\alpha_m^f(n+1)$ the way one might expect. On the contrary, even when $\varepsilon_m^f(n+1)$ is computed with five or six e.d.d., only one or two e.d.d. are transferred to $\alpha_m^f(n+1)$ or even no e.d.d. at all. This "relaxation of the finite-precision error" is due to the fact that the order of the term $\varepsilon_m^f(n+1)e_m^f(n+1)$ is decisively smaller than the one of the quantity $\alpha_m^f(n)$. Actually, it is an immediate consequence of Proposition 10 that the greater the numerical error with which $\varepsilon_m^f(n+1)$ is calculated, the smaller its exponent, and hence the smaller the exponent of the quantity $\varepsilon_m^f(n+1)e_m^f(n+1)$ as compared to the exponent of $\alpha_m^f(n)$. This implies that in the performance of the addition

$$\alpha_m^f(n) + \varepsilon_m^f(n+1)e_m^f(n+1), \tag{21}$$

the quantity $\varepsilon_m^f(n+1)e_m^f(n+1)$ undertakes a corresponding shift which implies that a very restricted number of e.d.d. is actually transferred to $\alpha_m^f(n+1)$, as described in Proposition 10.

Similarly, it turns out that an even considerable amount of f.p.e. may be generated in the computation of the Forward Linear Predictor $\mathbf{a}_m(n+1)$. However, this computation does not constitute a main source of f.p.e., since the generated numerical error does not propagate, all, in the subsequent computations and iterations. In fact, again, it is a straightforward consequence of Proposition 10 that the greater the number of e.d.d. with which one FLP coefficient $a_{j,m}(n+1)$ is generated, the smaller its exponent; therefore, the more clear and decisive the relaxation shift occurring in the performance of the addition

$$w_{j,m}^*(n+1) + \frac{a_{j,m}(n+1)\varepsilon_m^f(n+1)}{\alpha_m^f(n+1)} \tag{22}$$

necessary for the $w_{j,m+1}^*(n+1)$ computation. This favorable "relaxation shift" is strengthened by the fact that $\varepsilon_m^f(n+1)$ has, almost constantly, a (very) small value. Hence, once more, a very restricted number of the e.d.d. generated in the $\mathbf{a}_m(n+1)$ computation is transferred to $\mathbf{w}_m^*(n+1)$ so restricted, that one may safely claim that this error does not influence at all the numerical behaviour of the fast-Kalman algorithm.

3.6. Prediction of the number of erroneous decimal digits generated in the fast-Kalman algorithm

It is stressed once more, that one may recursively compute the exact, up to ± 1, number of e.d.d. with which the various quantities are computed owing to all the factors that have been pointed out in this paper, if one uses the previous results and Propositions 7, 8, 9, and 10.

This is demonstrated in Figure 1, where the results of a system identification experiment are presented. In fact, the continuous "step-like" line in Figure 1 shows the theoretically predicted number of e.d.d. of the "approximated response" $\hat{Z}(n+1)$, while the discontinuous line shows the experimentally confirmed number of e.d.d. This computation of the number of e.d.d. can be performed in parallel with the execution of the main algorithm, with small additional computational complexity. The great advantage of this method is that the user, at any instant of the execution of the algorithm, can have a complete knowledge of the algorithm robustness

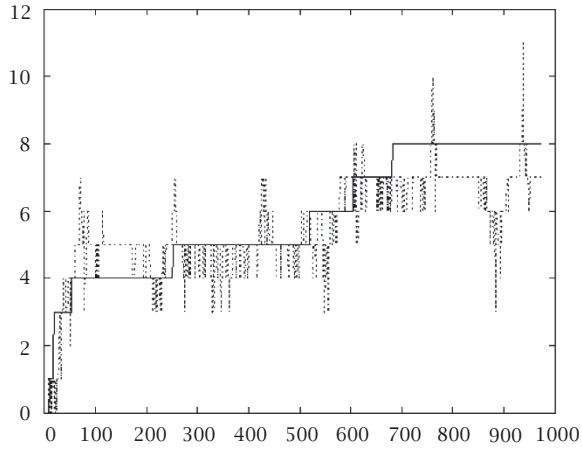


FIGURE 1: Demonstration of the theoretical prediction of the number of the generated e.d.d. The continuous "step-like" line shows the theoretically predicted number of e.d.d. of the "approximated response" $\hat{Z}(n+1)$ in connection with a system identification experiment, while the discontinuous line shows the experimentally confirmed number of e.d.d.

and of the reliability of the obtained results, and that independently of the nature of the input signal and of the employed machine precision.

4. ANALYSIS OF THE FINITE-PRECISION ERROR GENERATION AND PROPAGATION, IN THE CASE THAT THE INPUT SEQUENCE $x(n+1)$ IS A SINUSOIDAL OR PERIODIC SIGNAL

In the case where the nature of the input signal $x(n+1)$ is different than white noise, then it is intrinsically possible that, in every formula of the fast-Kalman algorithm, the relationship of the involved quantities may be different as well. Therefore, in this case, some of the aforementioned formulas may not be a main source of f.p.e. any more. On the other hand, due precisely to the fact that the terms in the various formulas of the algorithm may systematically have a different magnitude relationship, new formulas may emerge, that are the main sources of numerical error.

In general, in connection with any computational organization, the set of formulas that are the main sources of f.p.e. may have different elements or they may even be entirely disjoint, for a different nature of the input signal $x(n+1)$. In fact, in the fast-Kalman algorithm case, the following hold.

In contrast with what happens in the white noise case, the formula (A.5) that computes $\mathbf{w}^*_{m+1}(n+1)$ does not generate a serious amount of f.p.e. any more. On the contrary, the following two formulas are the main sources of f.p.e.

(1) Formula (A.10) that computes $e_m(n+1)$.

(2) Formula (A.8) that performs the computation of $\mathbf{w}^*_m(n+1)$, with main transmitter of this generated f.p.e., formula (A.5) that computes $\mathbf{w}^*_{m+1}(n+1)$. In other words, once more, those two formulas act as a strongly interrelated couple, as regards the generation and propagation of the f.p.e., with their roles interchanged however.

(3) The other main source of f.p.e. is, once more, the formula that computes $\varepsilon^f_m(n+1)$. That is it is the same with the first main source in the white noise case, precisely for the reasons described in Section 3.2 before, namely for the positive definiteness requirement of the system matrix $\mathbf{R}_m(n+1)$, for every n.

Notice, moreover, that there is an essentially lesser source of f.p.e., namely the one that calculates $\mathbf{c}_m(n+1)$.

The above results hold true for the case where the input signal is a sum of sinusoidal functions and hence for a large class of periodic functions.

All the comments regarding the role of the "error relaxation shift," that have been made in the white noise case, are identically valid in the case where the input sequence is a sinusoidal or a periodic function. And in this case too, one may compute, the number of e.d.d. with which the various quantities are computed, due to each man source of finite-precision error, separately and, most important, due to all the error sources together.

5. A GENERAL METHOD FOR STABILIZING THE FAST-KALMAN ALGORITHM AND FOR DEVELOPING ROBUST RECURSIVE LEAST SQUARES ALGORITHMS

The previous analysis demonstrates that the deeper and actual cause of the failure of the fast-Kalman algorithm is the existence of formulas that constitute the main sources of finite-precision error and the existence of formulas that transmit this error. Therefore, one may employ the results of this paper, in order to stabilize the fast-Kalman algorithm as well as to develop new algorithms that have radically improved numerical behaviour. In fact,

(A) Concerning the stabilization of the fast-Kalman algorithm the following method has been applied:

(1) First, one spots the main sources of finite-precision error, as before. These sources are formulas (A.3), (A.8), and (A.9) for white noise input and (A.5), (A.8), and (A.10) for a periodic input.

(2) Then from the resultant conclusions, a piece of code has been developed that predicts the exact amount, up to a selected number of digits, of the finite-precision error the main sources generate at each iteration, as stated in Section 3.6 and shown in Figure 1. This prediction computation is carried out while the algorithm is executed and the additional complexity is negligible compared to the one of the whole scheme.

(3) Finally, a feedback process that corrects the numerical error is executed each time the observed quantization error calculated at the previous step reaches a selected threshold, for example, it reaches five e.d.d. in the mantissa. In fact, in that case, one initiates execution of a second identical implementation of the fast-Kalman algorithm, which keeps running until convergence, simultaneously with the previous one for a limited number of iterations N_i. A good choice for N_i seems

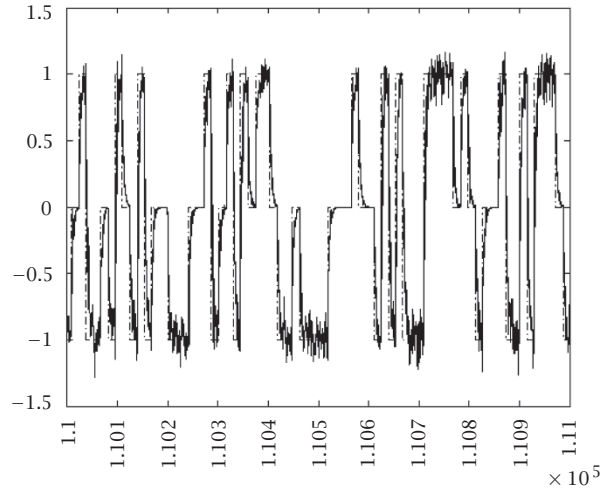


FIGURE 2: Demonstration of the way the stabilized fast-Kalman algorithm tracks a fast varying signal with white noise input. The dotted line depicts the original signal and the continuous line depicts the computed signal. For this experiment, a system order $m = 35$, an SNR 10 dB and a forgetting factor $\lambda = 0.97$ have been chosen.

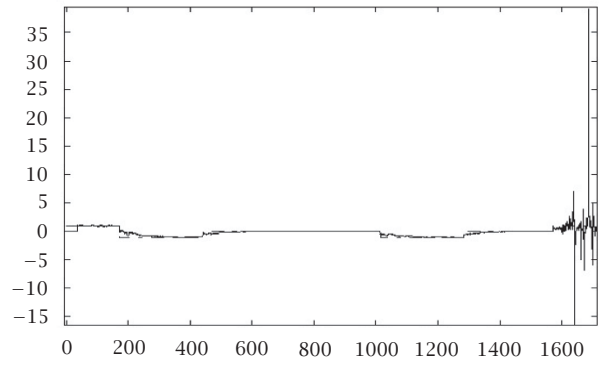
to be a number between 40 and 80 recursions. Immediately after conclusion of these N_i iterations, one can perform one of the following two actions for obtaining a robust scheme:

(a) Let the first implementation stop and let running the second implementation of the fast-Kalman algorithm until its main sources reach the same critical threshold of e.d.d. (say five), in which case the first implementation is called upon again and the present stops running after N_i recursions, and so on.

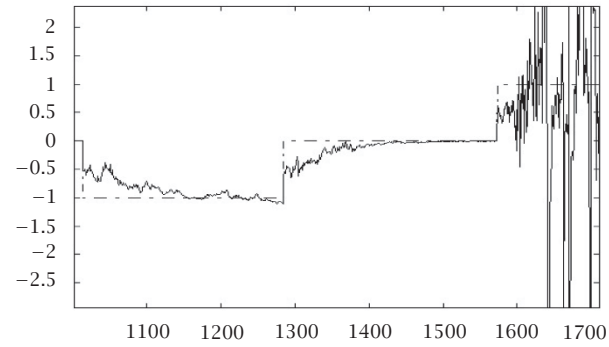
(b) If the second implementation is executed with a higher precision than the first one, the proper number of the decimal digits in the mantissa, as well as the proper exponent for fast-Kalman algorithm quantities are kept and are given as a quantization error-free input to the first implementation which subsequently continues running on its own, offering the desired results to the user.

It must be pointed out that application of the feedback process adds 10% to 15% to the overall computational complexity and that no discontinuity appears in the results of the algorithm.

One very important feature of this approach is that the fast-Kalman algorithm stabilized in this way, can track "very difficult to follow signals" with noticeable success. For example, it can track speech, music, environmental noise, and other nonstationary signals. The aforementioned results have been confirmed by a considerable number of simulation experiments. To set ideas, the stabilized version of the algorithm has been used to find the Linear Prediction Coefficients for various signals-outputs of many musical instruments and extended speech signals and run with complete success for millions of iterations, without showing any sign of breaking.



(a) Demonstration of the way the fast-Kalman algorithm completely fails to track the fast varying signal of Figure 2, due to quantization error, after the very limited number of approximately 1600 iterations. The same parameters with those of Figure 2 have been used.



(b) A detail of Figure 3a, depicting the area where the fast-Kalman algorithm fails completely, due to quantization error.

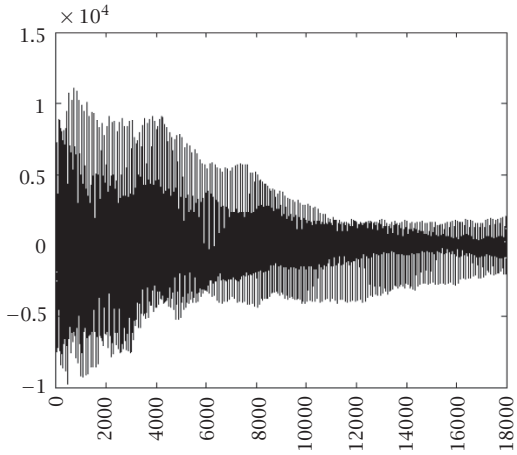
FIGURE 3

Few results of these experiments are depicted in Figures 2, 3, 4, and 5. In fact,

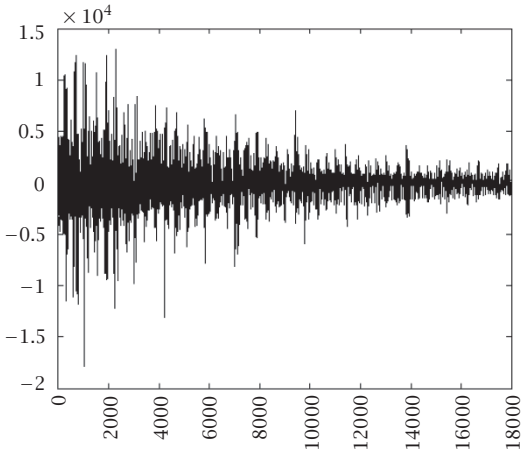
• In Figure 2, the way the stabilized fast-Kalman algorithm tracks a fast varying signal is presented with white noise input. For this experiment, there have been chosen a system order $m = 35$, an SNR 10 dB and a forgetting factor $\lambda = 0.97$, while all operations were made with standard single precision (float), using twenty-four bits for the mantissa and eight bits for the exponent.

• The incapability of the fast-Kalman algorithm to follow this fast varying signal is demonstrated in Figure 3, where the output of this scheme when the very same parameters have been used, is depicted.

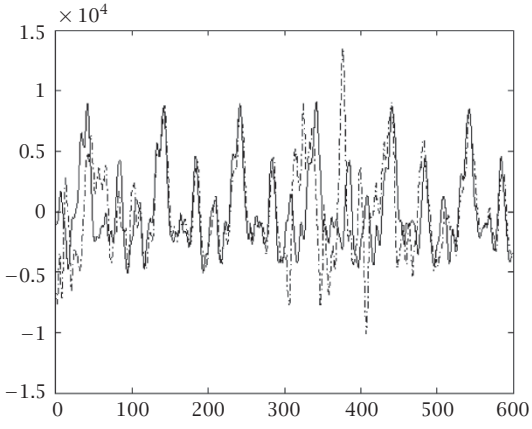
In Figure 4, a very difficult to follow by most RLS algorithms signal is presented. This signal is depicted in Figure 4a, and is the actual digitized recording of a musical instrument (a piano). The stabilized by the introduced method algorithm, has been used to perform the following experiment, very useful for removing environmental noise from a given signal, where, once more, all operations were made with standard single precision (float):



(a) The actual digitized recording of a musical instrument (a piano).

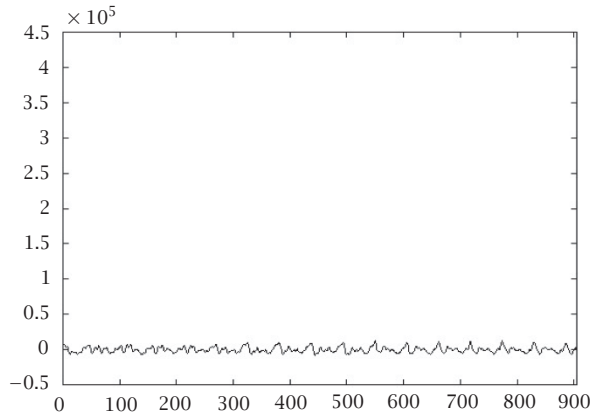


(b) The computed estimated response by the stabilized fast-Kalman algorithm, with input a sample of environmental noise. The forgetting factor λ is 0.97, the SNR is 10 dB, and the system order m is 128.

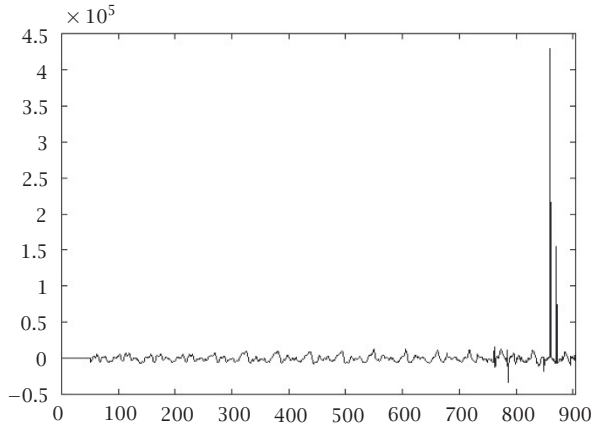


(c) A detail of the signals shown in Figures 3a and 3b superimposed here. The dotted line depicts the computed signal and the continuous line depicts the original signal.

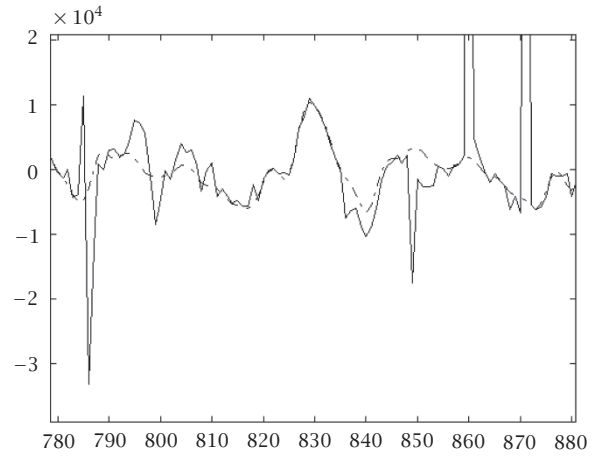
Figure 4



(a) The first 900 samples of the actual digitized recording of a musical instrument (a piano), the same as Figure 4. We have plotted the signal in the same scale with the one of Figure 5b, for comparison.



(b) The computed estimated response by the fast-Kalman algorithm, where the failure of this algorithm, starting approximately at sample 780, is obvious.



(c) A detail of the way the fast-Kalman algorithm fails to approximate the original signal of Figure 5a. It is clear that after sample 850 the fast-Kalman results become totally unreliable.

Figure 5

The input to the algorithm has been a sample of environmental noise.

(1) The output (desired response) of the algorithm has been the signal shown in Figure 4a, obtained from a piano.

(2) The forgetting factor λ is 0.97, the SNR is 10 dB and the system order m is 128. The estimated response is shown in Figure 4b.

(3) A detail of these signals superimposed is shown in Figure 4c.

• In Figures 5a, 5b, and 5c, the incapability of the fast-Kalman algorithm to follow the original signal of Figure 4 is demonstrated, where the output of this scheme when the very same parameters have been used, is depicted.

(B) On the other hand, in order to develop a more robust recursive least square algorithm, one must avoid including (using) formulas that are the main sources of finite word length error and/or formulas that are the main transmitters of quantization error, as they are described in this paper. To set ideas, one must use instead of formulas (A.3), (A.5), (A.8), (A.9), and (A.10), other equivalent that do not manifest the behaviour described in the previous sections, and, hence, that do not generate such a serious amount of quantization error. Similarly, probably at the same time, one must avoid using formulas that transmit the generated finite-precision error.

Such an RLS algorithm, developed by the authors, is presented in Appendix B. The proposed algorithm can be stabilized by means of a quite analogous techniques, while it maintains excellent tracking properties.

The main reason for which the proposed RLS computational scheme is more robust than the fast-Kalman one, lies with formula (B.4) used for the computation of n_m^f, which in a sense is equivalent to formula (A.3) that computes ε_m^f. To be more specific, in this formula, quantity $\mathbf{a}_m(n+1)$ is computed via formula (B.3), while in the fast-Kalman scheme via formula (A.2). The extra term present in (B.3) reduces the frequency of occurrence of events (i) and (ii) in Section 3.2, therefore the amount of quantization error that this main source generates is essentially reduced. The robustness of the introduced RLS algorithm, as compared with the fast-Kalman one, has been extensively experimentally verified, employing the method presented here.

In Figure 6, the way the proposed algorithm tracks a speech signal, a part of the recording of a phrase in American English taken at random from the related database "TIMIT" is depicted, signal, which, too, is very difficult to follow by most RLS algorithms. In this experiment, the system order $m = 20$, the window length $L = 200$ and the forgetting factor $\lambda = 0.99$. The very good tracking properties of the proposed algorithm are obvious in the graphs of the estimated output $z(n)$ as presented in this figure.

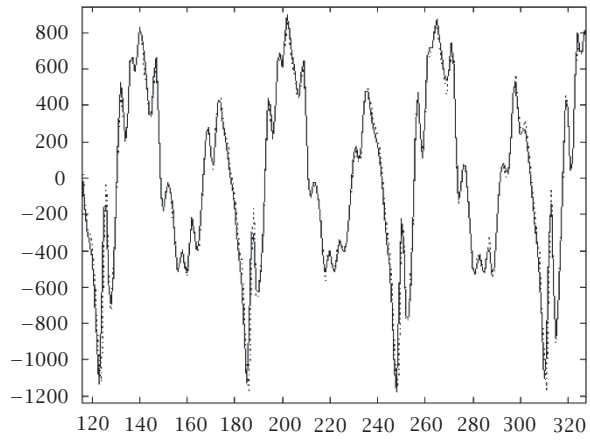


Figure 6: The way the proposed algorithm tracks a speech signal, a part of the recording of a phrase in American English taken at random from the related data base "TIMIT." In this experiment, the system order $m = 20$, the window length $L = 200$ and the forgetting factor $\lambda = 0.99$. The dotted line depicts the computed signal and the continuous line depicts the original signal. The very good tracking properties of the proposed algorithm are obvious in the graphs of the estimated output $\hat{Z}(n)$ as presented in this figure.

6. CONCLUSION

In this paper, it is shown that out of all the formulas that constitute the fast-Kalman algorithm only three are main sources of finite-precision error. Moreover, it is demonstrated that there is a very limited number of specific formulas that transmit the generated finite-precision error, while there is another class of formulas that lift or "relax" this error. In addition, a number of very general propositions is presented that allow for the calculation of the exact number of erroneous digits with which all the quantities of the fast-Kalman scheme are computed. Finally, a general methodology is introduced that allows both for the stabilization of the fast-Kalman algorithm as well as for the development of new algorithms which, intrinsically, suffer less of finite-precision problems.

APPENDICES

APPENDIX A: THE FAST-KALMAN ALGORITHM (WITH THE NOTATION USED IN [5])

(1) Quantities available at time n: $\mathbf{a}_m(n), \mathbf{b}_m(n), \mathbf{c}_m(n), \mathbf{x}_m(n), \mathbf{w}_m(n)$.

(2) New information: $x(n+1), z(n+1)$.

(3) Time updating of the gain vector:

$$e_m^f(n+1) = x(n+1) + \mathbf{a}_m^T(n)\mathbf{x}_m(n), \tag{A.1}$$

$$\mathbf{a}_m(n+1) = \mathbf{a}_m(n) + \mathbf{w}_m^*(n)e_m^f(n+1), \tag{A.2}$$

$$\varepsilon_m^f(n+1) = x(n+1) + \mathbf{a}_m^T\mathbf{x}_m(n), \tag{A.3}$$

$$\alpha_m^f(n+1) = \alpha_m^f(n) + e_m^f(n+1)\varepsilon_m^f(n+1), \tag{A.4}$$

$$\mathbf{w}_{m+1}^*(n+1) = \begin{bmatrix} 0 \\ \mathbf{w}_m^*(n+1) \end{bmatrix} + \left(\frac{\varepsilon_{m(n+1)}^f}{\alpha_{m(n+1)}^f}\right) * \begin{bmatrix} 1 \\ \mathbf{a}_m(n+1) \end{bmatrix}, \tag{A.5}$$

$$\mathbf{w}^*_{m+1}(n+1) = \begin{bmatrix} \mathbf{d}^*_m(n+1) \\ \delta^*_m(n+1) \end{bmatrix}, \tag{A.6}$$

$$e^b_m(n+1) = x(n+1-m) + \mathbf{b}^T_m(n)\mathbf{x}_m(n), \tag{A.7}$$

$$\mathbf{w}^*_m(n+1) = \frac{\mathbf{d}^*_m(n+1) - \delta^*_m(n+1)\mathbf{b}_m(n)}{1+\delta^*_m(n+1)e^b_m(n+1)}, \tag{A.8}$$

$$\mathbf{b}_m(n+1) = \mathbf{b}_m(n) + \mathbf{w}^*_m(n+1)e^b_m(n+1). \tag{A.9}$$

(4) Time updating of the LS FIR filter:

$$e_m(n+1) = z(n+1) + \mathbf{c}^T_m(n)\mathbf{x}_m(n+1), \tag{A.10}$$

$$\mathbf{c}_m(n+1) = \mathbf{c}_m(n) + \mathbf{w}^*_m(n+1)e_m(n+1). \tag{A.11}$$

APPENDIX B: THE $O(m)$ ALGORITHM FOR ADAPTIVE FILTERING VIA A FINITE WINDOW

(1) Definition of useful intermediate quantities

$$\begin{aligned} f^f_m(n+1) &= x(n+1) + \mathbf{a}^T_m(n)\mathbf{x}_m(n), \\ g^f_m(n+1) &= x(M) + \mathbf{a}^T_m(n)\mathbf{x}_m(M-1), \\ \delta^*_{xx}(n+1) &= \mathbf{x}^T_m(n)\mathbf{u}^*_m(n), \\ \delta^*_{yy}(n+1) &= \mathbf{x}^T_m(M-1)\mathbf{u}^*_m(n), \\ \delta^*_{xy}(n+1) &= \mathbf{x}^T_m(M-1)\mathbf{v}^*_m(n). \end{aligned} \tag{B.1}$$

(2) Updating $\Delta_f(n+1)$

$$\begin{aligned} \Delta_f(n+1) = \lambda\Delta_f(n) &+ \left(f^f_m(n+1)\right)^2\left(1+\delta^*_{xx}(n+1)\right) \\ &+ \mu\left(g^f_m(n+1)\right)^2\left(1-\delta^*_{yy}(n+1)\right) \\ &- 2\mu f^f_m(n+1)g^f_m(n+1)\delta^*_{xy}(n+1) \\ &+ \sigma^2. \end{aligned} \tag{B.2}$$

(3) Updating the FLP filter coefficients

$$\mathbf{a}_m(n+1) = \mathbf{a}_m(n) + f^f_m(n+1)\mathbf{u}^*_m(n) - \mu g^f_m(n+1)\mathbf{v}^*_m(n). \tag{B.3}$$

(4) Updating $\mathbf{u}^*_m(n+1)$ and $\mathbf{v}^*_m(n+1)$

(4a) Computation of $\mathbf{u}^*_m(n+1)$

$$n^f_m(n+1) = x(n+1) + \mathbf{a}^T_m(n+1)\mathbf{x}_m(n), \tag{B.4}$$

$$\mathbf{u}^*_{m+1}(n+1) = \begin{Bmatrix} 0 \\ \mathbf{u}^*_{m+1}(n) \end{Bmatrix} - \begin{Bmatrix} 1 \\ \mathbf{a}_m(n+1) \end{Bmatrix} \frac{n^f_m(n+1)}{\Delta_f(n+1)}. \tag{B.5}$$

(4b) Partitioning

$$\mathbf{u}^*_{m+1}(n+1) = \begin{bmatrix} \tilde{\mathbf{u}}^*_m(n+1) \\ u^*_{m+1}(n+1) \end{bmatrix}. \tag{B.6}$$

(4c) Computation of $\mathbf{v}^*_m(n+1)$

$$\begin{aligned} k^f_m(n+1) &= x(M) + \mathbf{a}^T_m(n+1)\mathbf{x}_m(M-1), \\ \mathbf{v}^*_{m+1}(n+1) &= \begin{Bmatrix} 0 \\ \mathbf{v}^*_m(n) \end{Bmatrix} - \begin{Bmatrix} 1 \\ \mathbf{a}_m(n+1) \end{Bmatrix} \frac{k^f_m(n+1)}{\Delta_f(n+1)}. \end{aligned} \tag{B.7}$$

(4d) Partitioning

$$\mathbf{v}^*_{m+1}(n+1) = \begin{bmatrix} \tilde{\mathbf{v}}^*_m(n+1) \\ \mathbf{v}^*_{m+1}(n+1) \end{bmatrix}. \tag{B.8}$$

(4e) Computation of $\mathbf{u}^*_m(n+1)$ and $\mathbf{v}^*_m(n+1)$

$$D = 1 + f^b_m(n+1)u^*_m(n+1) - \mu g^b_m(n+1)v^*_{m+1}(n+1), \tag{B.9}$$

$$\begin{aligned} D_{uu} &= \frac{1-\mu v^*_{m+1}(n+1)g^b_m(n+1)}{D}, \\ D_{uv} &= \frac{\mu u^*_{m+1}(n+1)g^b_m(n+1)}{D}, \\ D_{vu} &= -\frac{v^*_{m+1}(n+1)f^b_m(n+1)}{D}, \\ D_{vv} &= \frac{1+u^*_{m+1}(n+1)f^b_m(n+1)}{D}, \\ d_u &= -\frac{u^*_{m+1}(n+1)}{D}, \qquad d_v = -\frac{v^*_{m+1}(n+1)}{D}, \end{aligned} \tag{B.10}$$

$$\mathbf{u}^*_m(n+1) = D_{uu}\tilde{\mathbf{u}}^*_m(n+1) + D_{uv}\tilde{\mathbf{v}}^*_m(n+1) - d_u\mathbf{b}_n, \tag{B.11}$$

$$\mathbf{v}^*_m(n+1) = D_{vu}\tilde{\mathbf{u}}^*_m(n+1) + D_{vv}\tilde{\mathbf{v}}^*_m(n+1) - d_v\mathbf{b}_n. \tag{B.12}$$

(5) Updating the BLP solution

$$\begin{aligned} f^b_m(n+1) &= x(n+1-m) + \mathbf{b}^T_m(n)\mathbf{x}_m(n+1), \\ g^b_m(n+1) &= x(M-m) + \mathbf{b}^T_m(n)\mathbf{x}_m(M), \\ \mathbf{b}_m(n+1) &= \mathbf{b}_m(n) + f^b_m(n+1)\mathbf{u}^*_m(n+1) \\ &\quad - \mu g^b_m(n+1)\mathbf{v}^*_m(n+1). \end{aligned} \tag{B.13}$$

(6) Compute the filter coefficients

$$\begin{aligned} f_m(n+1) &= z(n+1) + \mathbf{c}^T_m(n)\mathbf{x}_m(n+1), \\ g_m(n+1) &= z(M) + \mathbf{c}^T_m(n)\mathbf{x}_m(M), \\ \mathbf{c}_m(n+1) &= \mathbf{c}_m(n) + f_m(n+1)\mathbf{u}^*_m(n+1) \\ &\quad - \mu g_m(n+1)\mathbf{v}^*_m(n+1). \end{aligned} \tag{B.14}$$

REFERENCES

[1] P. Fabre and C. Gueguen, "Improvement of the fast recursive least-squares algorithms via normalisation: A comparative study," *IEEE Trans. on ASSP*, vol. 34, no. 2, pp. 296–308, 1986.

[2] A. D. Pouliezos and G. S. Stavrakakis, *Real Time Fault Monitoring of Industrial Processes*, Kluwer, Dordrecht, 1994.

[3] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*, Macmillan, NY, 1988.

[4] L. Ljung, M. Morf, and D. Falconer, "Fast calculation of gain matrices for recursive estimation schemes," *Internat. J. Control*, vol. 27, no. 1, pp. 1–19, 1978.

[5] G. Carayannis, D. G. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction," *IEEE Trans. on ASSP*, vol. ASSP-31, no. 6, pp. 1394–1402, 1983.

[6] J. M. Cioffi and T. Kailath, "Fast, recursive-least-squares transversal filters for adaptive filtering," *IEEE Trans. on ASSP*, vol. ASSP-32, no. 2, pp. 304–337, 1984.

[7] H. Lev-Ari, T. Kailath, and J. Cioffi, "Least squares adaptive lattice and transversal filters: A unified theory," *IEEE Trans. on Inform. Theory*, vol. IT-30, no. 2, pp. 346–359, 1989.
[8] M. Morf, T. Kailath, and L. Ljung, "Fast algorithms for recursive identification," in *Proc. 1976 Conf. Decision and Control*, Florida, 1976, pp. 916–921.
[9] D. T. M. Slock and T. Kailath, "Numerically stable fast transversal filters for recursive least squares adaptive filtering," *IEEE Trans. on Signal Processing*, vol. 39, no. 1, pp. 92–114, 1991.
[10] S. Ljung and L. Ljung, "Error propagation properties of recursive least-squares adaptation algorithms," *Automatica J. IFAC*, vol. 21, no. 2, pp. 157–167, 1985.
[11] J. M. Cioffi, "Limited-precision effects in adaptive filtering," *IEEE Trans. on Circuits and Systems*, vol. CAS-34, no. 7, pp. 821–833, 1987.
[12] D. W. Lin, "On digital implementation of the fast kalman algorithms," *IEEE Trans. on ASSP*, vol. ASSP-32, pp. 998–1005, 1984.
[13] J.-L. Botto and G. V. Moustakides, "Stabilizing the fast Kalman algorithms," *IEEE Trans. on ASSP*, vol. ASSP-37, no. 9, pp. 1342–1348, 1989.
[14] N. Glaros and G. Carayannis, "Exact and first-order error analysis of the schur and the split schur algorithms: Theory and practice," *IEEE Trans. on Signal Processing*, vol. 42, no. 8, pp. 1916–1938, 1994.
[15] C. N. Papaodysseus, E. B. Koukoutsis, and C. N. Triantafyllou, "Error sources and error propagation in the levinson-durbin algorithm," *IEEE Trans. on ASSP*, vol. 41, no. 4, pp. 1635–1651, 1993.
[16] C. N. Papaodysseus, E. B. Koukoutsis, and C. Vassilatos, "Error sources, error propagation and methods of error correction in ls fir filtering and 1-step ahead prediction," *IEEE Trans. on Signal Processing*, vol. 42, no. 5, pp. 1097–1108, 1994.
[17] C. Papaodysseus, E. Koukoutsis, C. Triantafyllou, and C. Vassilatos, "Exact monitoring of the numerical error in various speech algorithms," in *Proc. Eurospeech 91*, 1991, vol. 3, pp. 1073–1076.
[18] C. Papaodysseus, E. Koukoutsis, and C. Vassilatos, "Accurate prediction of the numerical error generated in the forward linear prediction algorithms," *ISMM Journal on Microcomputer Applications*, vol. 12, no. 1, pp. 1–6, 1993.
[19] C. Papaodysseus, C. Triantafyllou, E. Koukoutsis, and G. Carayannis, "Error propagation and numerical recovery for a class of parametric dsp algorithms," in *ICASSP-90*, 1990, pp. 1349–1352.

Constantin Papaodysseus was born in Athens, Greece. He received his Diploma in Electrical and Computer Engineering from the National Technical University of Athens (NTUA) and his M.Sc. degree from Manchester University, UK. He received his Ph.D. degree in Computer Engineering from NTUA. Since 1996 he is assistant professor in NTUA, Department of Electrical and Computer Engineering. His research interests include music and speech processing and automatic recognition, image processing, applied mathematics, algorithm robustness, and quantization error analysis, adaptive algorithms, biomedical engineering, and so forth.

Constantin Alexiou was born in Igoumenitsa, Greece, in 1973. He received his Diploma and M.Sc. degree in Electrical and Computer Engineering from National Technical University of Athens in 1996. He is currently persuing the Ph.D. degree in Computer Engineering in the same university. His research interests and recent work are on the following subjects: music and speech processing and automatic recognition, algorithm robustness, algorithms for echo cancellation, biomedical engineering, and so forth.

George Rousopoulos was born in Athens, Greece, in 1971. He received his Diploma in Computer and Software Engineering from Technical University of Patras in 1994. He received his Ph.D. in Computer Engineering from the National Technical University of Athens in 2000. His research interests and recent work are on the following subjects: music and speech processing and automatic recognition, image processing, pattern recognition, algorithm robustness, algorithms for echo cancellation, and so forth.

Athanasios Panagopoulos was born in Athens, Greece, in 1973. He received his Diploma and M.Sc. degree in Electrical and Computer Engineering from National Technical University of Athens in 1996. He is a Ph.D. Computer Engineering student in the same university. His research interests and recent work are on the following subjects: music and speech processing and automatic recognition, image processing, pattern recognition, algorithms for echo cancellation, and so forth.