A NOVEL GIVENS ROTATION BASED FAST SQR-RLS ALGORITHM

Alberto Carini
Dipartimento di Elettrotecnica, Elettronica ed Informatica
Università di Trieste, Via Valerio 10, 34127 Trieste, Italy
Tel: +39.40.676.7127; Fax: +39.40.676.3460
e-mail: carini@imagets.univ.trieste.it

ABSTRACT

A novel Fast RLS Algorithm based on the Givens Rotation and developed from an UDU^T square-root factorization of autocorrelation matrix is discussed. The algorithm presents excellent numerical properties and requires $14N$ multiplications and $6N$ divisions per sampling interval, where N is the linear filter order.

1 INTRODUCTION

Adaptive filters constitute an important and prolific field of signal processing, both for theory and practical applications. The problem addressed in this paper is the classical Fast RLS (Recursive Least Squares) one [2]: we want to identify the linear filter which minimize the exponentially weighted cost function

$$J(n) = \sum_{k=0}^{n} \lambda^{n-k} \big(d(k) - H_n^T X(k)\big)^2 \qquad (1)$$

where $d(k)$ is the desired response signal, H_n is the linear filter coefficient vector at time n, $X(k)$ is the input data vector at time k and λ is a forgetting factor which controls the rate of tracking time varying parameters. Moreover, we want a multiplication/division computational complexity of order N, where N is the filter memory length. A novel solution based on the Givens rotation and developed from an UDU^T square-root factorization [1] of the autocorrelation matrix is presented in this paper.

The proposed algorithm belongs to the class of Fast QR and Lattice RLS algorithms. As in QR algorithms we do have a Q Givens rotation matrix and an R upper triangular matrix which is the Cholesky factor of the autocorrelation matrix. Moreover, as in Fast QR and Lattice algorithms the desired signal estimation is based on different filter order backward prediction errors. Thus, the joint process part of our Fast SQR RLS coincides with that of Fast QR and Lattice algorithms based on backward prediction error vector [5]. But differently from the uppermentioned algorithms, the derivation of this adaptive filter is an algebraic one, based on the relationship between two different UDU^T square-root factorizations of the extended autocorrelation matrix.

Since the joint process part of our algorithm coincides with that of Lattice RLS algorithms, this Fast SQR RLS algorithm does not determine the filter coefficients of a transversal realization (H_n) but those of a Lattice realization. So, even if the vector H_n is not directly determined, the algorithm can still be applied for system identification, as well as adaptive filtering and prediction. Furthermore, differently from all other algorithms which gives a Lattice filter realization (Fast QR and Lattice algorithms), here we have a direct dependency of a priori forward prediction error from the input sample x_n. This direct dependency makes the algorithm suitable for ADPCM applications in signal coding [3] even if it is paid with a not-pipelineable structure.

The algorithm presents excellent numerical stability properties which derives from the exploitation of the UDU^T square-root factorization of the autocorrelation matrix and from the use of numerical robust operations (like Givens rotation) which do not give rise to numerical error accumulation.

This paper is organized as follow. In section 2 the algorithm is described. In section 3 some experimental results are reported and in section 4 some final remarks are discussed.

2 ALGORITHM DERIVATION

Let us consider an UDU^T square-root factorization of the autocorrelation matrix

$$\Omega_n = \sum_{k=0}^{n} \lambda^{n-k} X(k) X^T(k) = R_n^T P_n R_n \qquad (2)$$

where $X(k)$ is the input data vector at time k, R_n is an upper triangular with unit diagonal matrix and P_n is a positive diagonal matrix, such that the uniqueness of the factorization (2) will be assured [1].

As in many fast algorithms we exploit the relationship between forward and backward prediction filters. The following quantities are defined (see [2] for more details):

This work has been partially supported by ESPRIT LTR Project 20229 Noblesse

- extended input vector,

$$\bar{X}(k) = \left[v_k \vdots X^T(k-1)\right]^T = \left[X^T(k) \vdots r_{k-1}\right]^T \quad (3)$$

where $v_k = x_k$ and $r_{k-1} = x_{k-N}$

- forward and backward predictors, A_n and B_n respectively
- forward and backward prediction errors,

$$f_n(k) = v_k + A_n^T X(k-1) \quad (4)$$
$$b_n(k) = r_{k-1} + B_n^T X(k) \quad (5)$$

- autocorrelation of forward and backward prediction errors, α_n and β_n
- the likelihood variable $\gamma_n = 1 - X^T(n)\Omega_n^{-1}X(n)$, which is the estimation error in the evaluation of the pinning sequence from $X(n)$.

Let us define first the vector $D_n = R_n^{-T}X(n)$ which, as we will see, coincides with the backward prediction error vector. In the proposed algorithm D_n will take place of both Kalman gain vector and input data vector of classical Fast RLS and FTF algorithms [2].

The updating of D_n is based on the definition of two different UDU^T square-root factorizations of the extended autocorrelation matrix $\bar{\Omega}_n = \sum_{k=0}^{n} \lambda^{n-k}\bar{X}(k)\bar{X}^T(k)$. Using the two definitions of $\bar{X}(k)$ it is possible, in fact, to derive this two factorizations:

$$\bar{R}_n = \begin{bmatrix} R_n & -Y_n \\ 0^T & 1 \end{bmatrix}, \ \bar{P}_n = \begin{bmatrix} P_n & 0 \\ 0^T & \beta_n \end{bmatrix} \quad (6)$$

and

$$\tilde{R}_n = \begin{bmatrix} 1 & 0^T \\ -Z_n & R_{n-1} \end{bmatrix}, \ \tilde{P}_n = \begin{bmatrix} \alpha_n & 0^T \\ 0 & P_{n-1} \end{bmatrix} \quad (7)$$

where $Y_n = R_n B_n$ and $Z_n = R_{n-1}A_n$.

These two matrix couples do not coincide ($\bar{R}_n$ is upper triangular while $\tilde{R}_n$ is not), but differs from a rotation matrix which can be efficiently computed from the knowledge of $\tilde{P}_n$ and of only the first column of $\tilde{R}_n$:

$$\bar{P}_n^{1/2}\bar{R}_n = Q\,\tilde{P}_n^{1/2}\tilde{R}_n \quad (8)$$

In order to avoid square-roots, we can determine the matrix $\hat{Q} = \bar{P}_n^{-1/2}Q\tilde{P}_n^{1/2}$. $\hat{Q}$ is not a rotation matrix but, by the use of Givens rotations, we can decompose it into N elementary matrices. Let us first decompose Q into N Givens rotations, which set to zero the elements of the first column of $\tilde{P}_n^{1/2}\tilde{R}_n$ and preserve the triangular structure of the remaining columns. We have

$$\begin{aligned} \hat{Q} &= \bar{P}_n^{-1/2}Q_N Q_{N-1}\cdots Q_1 \tilde{P}_n^{1/2} = \\ &= \bar{P}_{n,N}^{-1/2}Q_N\tilde{P}_{n,N}^{1/2}\bar{P}_{n,N-1}^{-1/2}Q_{N-1}\tilde{P}_{n,N}^{1/2}\cdots \\ &\qquad \cdots\bar{P}_{n,1}^{-1/2}Q_1\tilde{P}_{n,1}^{1/2} \end{aligned} \quad (9)$$

$b_0 = \left(\tilde{p}_0^{-1}\right)^{-1}$
FOR $n = N$ TO 1 STEP -1
$\quad b_1 = b_0 + \left(z_{n-1}/\tilde{p}_n^{-1}\right)z_{n-1}$
$\quad \bar{p}_n^{-1} = \left(b_1/b_0\right)\tilde{p}_n^{-1}$
$\quad x_0 = x_0 - z_{n-1}x_n$
$\quad y_n = x_n + \left(z_{n-1}/\tilde{p}_n^{-1}\right)\cdot\left(x_0/b_1\right)$
$\quad b_0 = b_1$
END FOR

$\bar{p}_0^{-1} = b_1^{-1}$
$y_0 = x_0$

Table 1: Algorithm for the computation of $\bar{P}_n^{-1}$ and of $Y = \hat{Q}X$ from the knowledge of Z_n and $\tilde{P}_n^{-1}$.

where $\bar{P}_{n,N} = \bar{P}_n$, $\tilde{P}_{n,1} = \tilde{P}_n$ and $\bar{P}_{n,i} = \tilde{P}_{n,i+1}$. Each one of the elementary matrices $\bar{P}_{n,i}^{-1/2}Q_i\tilde{P}_{n,i}^{1/2}$ annihilates one of the elements of Z_n and keeps the diagonal of $\tilde{R}_n$ set to unit. In Table 1 is reported the algorithm for the computation of $\hat{Q}X$ (X generic vector) and of $\bar{P}_n^{-1}$ from Z_n and $\tilde{P}_n^{-1}$.

Let us now consider

$$\tilde{D}_n = \tilde{R}_n^{-T}\bar{X}(n) = \begin{bmatrix} f_n(n) \\ D_{n-1} \end{bmatrix} \quad (10)$$

and

$$\bar{D}_n = \bar{R}_n^{-T}\bar{X}(n) = \begin{bmatrix} D_n \\ b_n(n) \end{bmatrix}. \quad (11)$$

evidently it is $\bar{D}_n = \hat{Q}\tilde{D}_n$ and, thus, D_n can be trivially updated. Moreover, since $\bar{R}_n$ is an upper triangular matrix as R_n, from (11) we have that D_n is the backward prediction error vector.

From the knowledge of D_n we can compute all other algorithmic parameters. In fact, by manipulation of the classical expressions of the forward a priori prediction error and of the forward predictor updating rule [2], we can obtain

$$f_{n-1}(n) = v_n + D_{n-1}^T\left(T_{n-1}Z_{n-1}\right) \quad (12)$$

$$Z_n = \left(T_{n-1}Z_{n-1}\right) - \left(P_{n-1}^{-1}D_{n-1}\right)f_{n-1}(n). \quad (13)$$

Here T_n is an upper triangular with unit diagonal matrix such that

$$\begin{aligned} T_n U_n T_n^T &= P_n^{-1} + \gamma_n^{-1}\left(P_n^{-1}D_n\right)\left(P_n^{-1}D_n\right)^T \\ &= \left(P_n - D_n D_n^T\right)^{-1} \end{aligned} \quad (14)$$

where U_n is a positive diagonal matrix. It can be demonstrated that T_n satisfy equation (15)

$$R_n = T_n R_{n-1}. \quad (15)$$

sum = 0

FOR $i = N-1$ TO 0 STEP -1
$w = (s_i/c) + d_i^2$
$c = s_i/w$
$y_i = x_i + d_i \cdot \text{sum}$
$\text{sum} = \text{sum} + \frac{d_i}{w} x_i$
END FOR

Table 2: Procedure for the computation of $Y = TX$ product where T is an upper triangular with unit diagonal matrix such that $TUT^T = S + cDD^T$ (U, S are diagonal matrices, D is a vector and c is a scalar).

A fast procedure for the computation of TX product, with X generic vector, has been developed and is reported in Table 2. In this procedure T is an upper triangular with unit diagonal matrix such that $TUT^T = S + cDD^T$, where U, S are positive diagonal matrices, D is a vector and c is a scalar.

As for $f_{n-1}(n)$ and Z_n, the joint process part is given by

$$e_{n-1}(n) = d(n) - D_n^T (T_n W_{n-1}) \quad (16)$$

$$W_n = (T_n W_{n-1}) + \left(P_n^{-1} D_n\right) e_{n-1}(n) \quad (17)$$

where $W_n = R_n H_n$ is the coefficient vector of the adaptive filter lattice realization.

The update of the likelihood variable γ_n is critical for the numerical stability of the algorithm. We can update γ_n with

$$\gamma_n = \gamma_{n-1} - \alpha_n^{-1} f_n^2(n) + \beta_n^{-1} b_n^2(n). \quad (18)$$

But if we update γ_n with (18), a long term numerical instability can arise. This numerical instability is produced by an error accumulation on γ_n and it depends from the wordlength precision of processor. The long term instability and error accumulation can be avoided recomputing γ_n, after a certain amount of steps, as

$$\gamma_n = 1 - D_n^T P_n^{-1} D_n; \quad (19)$$

equation (19), in fact, evaluate directly the correct value of γ_n. The combined use of equations (18) and (19) leads to a numerically stable algorithm without increasing the computational complexity.

The final algorithm is reported in Table 3 with the operation count of each step. The algorithm requires $14N$ multiplications and $6N$ divisions for each sampling interval in case of filtering, $10N$ multiplications and $6N$ divisions in case of prediction. The addition count is comparable to the multiplication one and thus has been neglected.

For what regards the initialization of the algorithm, we have considered $Z_{-1} = D_{-1} = W_{-1} = 0$, $\gamma_{-1} = 1$, $P_{-1}^{-1} = \delta^{-1} I$ and $\alpha_{-1} = \delta$ with $\delta << 1$.

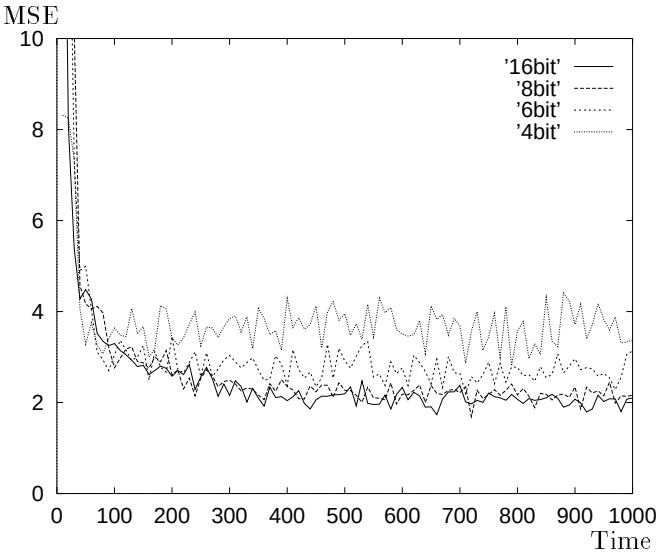

Figure 1: Arithmetic mean of a priori forward prediction mean square error as a function of time. The arithmetic mean is evaluated on ten different non-white-gaussian noise signals while the mean square error is computed on data segments of 10 samples.

3 EXPERIMENTAL RESULTS

The numerical stability of the algorithm has been tested by several experiments with different types of data signals: from deterministic signals, to random noise signals, to voice signals. In these experiments the adaptive filter was used both for prediction and system identification.

A finite precision arithmetic was simulated as in [4] by implementing a floating point arithmetic with a limited precision mantissa, reduced till 4 bits. The longest simulation performed with a 4 bits mantissa value had more than 4 millions samples and in no one of all the considered simulations any instability has been observed.

In Figure 1 is represented, as a function of time, the arithmetic mean of a priori forward prediction mean square error. The arithmetic mean is evaluated on ten different non-white-gaussian noise signals while the mean square error is computed on data segments of 10 samples. All noise signals are obtained by filtering a zero mean, unit variance white gaussian noise $N(n)$ with the cascade of two linear filters given by

$$x(n) = N(n) + 0.9x(n-1) \quad (20)$$

$$y(n) = 2x(n) + x(n-1) - 0.5x(n-2) \quad (21)$$

The different plots refers to different mantissa precision of the processor. Figure 1 illustrates how the wordlength affects the performances of the algorithm and shows the good convergence properties even with a low mantissa precision. The same performance can be obtained both from an 8 bit mantissa wordlength and the standard floating point precision.

Algorithm	Operation Count	
	×	÷
$T_{n-1}Z_{n-1}$	$4N$	$3N$
$f_{n-1}(n) = v_n + D_{n-1}^T (T_{n-1}Z_{n-1})$	$1N$	
$f_n(n) = f_{n-1}(n)\gamma_{n-1}$		
$Z_n = (T_{n-1}Z_{n-1}) - \left(P_{n-1}^{-1}D_{n-1}\right) f_{n-1}(n)$	$1N$	
$\alpha_n = \lambda\alpha_{n-1} + f_{n-1}(n)f_n(n)$		
$\tilde{D}_n = \begin{bmatrix} f_n(n) \\ D_{n-1} \end{bmatrix} \quad \tilde{P}_n^{-1} = \begin{bmatrix} \alpha_n^{-1} & 0^T \\ 0 & P_{n-1}^{-1} \end{bmatrix}$		
$\hat{Q}, \quad \bar{P}_n^{-1} = \begin{bmatrix} P_n^{-1} & 0 \\ 0^T & \beta_n^{-1} \end{bmatrix}, \quad \bar{D}_n = \begin{bmatrix} D_n \\ b_n(n) \end{bmatrix} = \hat{Q}\tilde{D}_n$	$4N$	$3N$
$\begin{cases} \gamma_n = \gamma_{n-1} - \alpha_n^{-1} f_n^2(n) + \beta_n^{-1} b_n^2(n) \\ \gamma_n = 1 - D_n^T \left(P_n^{-1} D_n\right) \end{cases}$	$(1 \sim N)$	
$T_n W_{n-1}$	$2N$	
$e_{n-1}(n) = d(n) - D_n^T (T_n W_{n-1})$	$1N$	
$W_n = (T_n W_{n-1}) + \left(P_n^{-1} D_n\right) e_{n-1}(n)$	$1N$	

Table 3: The Givens Rotation Based Fast SQR-RLS Algorithm. In the second and third column is reported the operation cost in multiplications and divisions respectively.

4 FINAL REMARKS AND CONCLUSION

A novel Fast RLS algorithm based on Givens rotations have been presented. The algorithm operation count is $14N$ multiplications and $6N$ divisions per sampling interval for filtering, $10N$ multiplications and $6N$ divisions for prediction. By the use of an array of processors, however, algorithm adaptation can be performed in a limited O(N) number of machine cycles.

Another derivation of this algorithm has been developed from two different square-root Cholesky factorizations of the autocorrelation matrix. However, this second formulation requires square-root operations.

Robust long term numerical stability of the proposed algorithm has been experimentally verified with different types of data signals and on very long data segments (up to 4 million samples). In none of the performed simulations any instability has been observed even with 4 bit mantissa floating point arithmetic.

The extension of the algorithm to the class of Volterra filters has been performed with a multichannel approach. The algorithm in this extension has retained its properties, i.e. the fast implementation and the numerical stability.

ACKNOWLEDGEMENTS

The author wishes to thank Prof. G. Sicuranza and Dr E. Mumolo for the useful discussions and the support provided during the development of the whole work.

References

[1] G.J.Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic Press, 1977, pp.37-55

[2] S.Haykin, *Adaptive Filter Theory*, Prentice Hall, 1991

[3] E.Mumolo and A.Carini, "Volterra Adaptive Prediction of Speech with Application to Waveform Coding," *European Transactions on Telecomunications*, Vol.6, No.6, November-December 1995, pp.685-693

[4] I.K.Proudler, J.G.McWhirter and T.J.Shepherd, "Computationally efficient QR decomposition approach to Least Squares Adaptive Filtering," *IEE Proceedings-F*, Vol.138, No.4, August 1991, pp.341-353

[5] P.A.Regalia and M.G.Bellanger, "On The Duality Between Fast QR Methods and Lattice Methods in Least Squares Adaptive Filtering" *IEEE Trans. on Signal Processing*, Vol.39, No.4, April 1991, pp.879-891