8

FAST TRANSVERSAL RLS ALGORITHMS

8.1 INTRODUCTION

Among the large number of algorithms that solve the least-squares problem in a recursive form, the fast transversal recursive least-squares (FTRLS) algorithms are very attractive due to their reduced computational complexity [1]-[7].

The FTRLS algorithms can be derived by solving simultaneously the forward and backward linear prediction problems, along with two other transversal filters: the joint-process estimator and an auxiliary filter whose desired signal vector has one as its first and unique nonzero element (i.e., $d(0) = 1$). Unlike the lattice-based algorithms, the FTRLS algorithms require only time-recursive equations. However, a number of relations required to derive some of the FTRLS algorithms can be taken from the previous chapter on LRLS algorithms. The FTRLS algorithm can also be considered a fast version of an algorithm to update the transversal filter for the solution of the RLS problem, since a fixed-order update for the transversal adaptive filter coefficient vector is computed at each iteration.

The relations derived for the backward and forward prediction in the lattice-based algorithms can be used to derive the FTRLS algorithms. The resulting algorithms have computational complexity of order N making them especially attractive for practical implementation. When compared to the lattice-based algorithms, the computational complexity of the FTRLS algorithms is lower due to the absence of order-updating equations. In particular, FTRLS algorithms typically require $7N$ to $11N$ multiplications and divisions per output sample, as compared to $14N$ to $29N$ for the LRLS algorithms. Therefore, FTRLS algorithms are considered the fastest implementation solutions of the RLS problem [1]-[7].

Several alternative FTRLS algorithms have been proposed in the literature. The so-called fast Kalman algorithm [1], which is certainly one of the earlier fast transversal RLS algorithms, has computational complexity of $11N$ multiplications and divisions per output sample. In a later stage of research development in the area of fast transversal algorithms, the fast *a posteriori* error sequential technique (FAEST) [2], and the fast transversal filter (FTF) [3] algorithms were proposed, both requiring an order of $7N$ multiplications and divisions per output sample. The FAEST and FTF algorithms have

P.S.R. Diniz, *Adaptive Filtering*, DOI: 10.1007/978-0-387-68606-6_8,

the lowest complexity known for RLS algorithms, and are useful for problems where the input vector elements consist of delayed versions of a single input signal. Unfortunately, these algorithms are very sensitive to quantization effects and become unstable if certain actions are not taken [5]-[7], [9].

In this chapter, a particular form of the FTRLS algorithm is presented, where most of the derivations are based on those presented for the lattice algorithms. It is well known that the quantization errors in the FTRLS algorithms present exponential divergence [1]-[7]. Since the FTRLS algorithms have unstable behavior when implemented with finite-precision arithmetic, we discuss the implementation of numerically stable FTRLS algorithms, and provide the description of a particular algorithm [8]-[10].

8.2 RECURSIVE LEAST-SQUARES PREDICTION

All fast algorithms explore some structural property of the information data in order to achieve low computational complexity. In the particular case of the fast RLS algorithms discussed in this text, the reduction in the computational complexity is achieved for the cases where the input signal consists of consecutively delayed samples of the same signal. In this case, the patterns of the fast algorithms are similar in the sense that the forward and backward prediction filters are essential parts of these algorithms. The predictors perform the task of modeling the input signal, which as a result allows the replacement of matrix equations by vector and scalar relations.

In the derivation of the FTRLS algorithms, the solutions of the RLS forward and backward prediction problems are required in the time-update equations. In this section, these solutions are reviewed with emphasis on the results that are relevant to the FTRLS algorithms. As previously mentioned, we will borrow a number of derivations from the previous chapter on lattice algorithms. It is worth mentioning that the FTRLS could be introduced through an independent derivation, however the derivation based on the lattice is probably more insightful and certainly more straightforward at this point.

8.2.1 Forward Prediction Relations

The instantaneous *a posteriori* forward prediction error for an Nth-order predictor is given by

$$\begin{aligned} \varepsilon_f(k,N) &= x(k) - \mathbf{w}_f^T(k,N)\mathbf{x}(k-1,N) \\ &= \mathbf{x}^T(k,N+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k,N) \end{bmatrix} \end{aligned} \tag{8.1}$$

The relationship between *a posteriori* and *a priori* forward prediction error, first presented in equation (7.49) and repeated here for convenience, is given by

$$e_f(k,N) = \frac{\varepsilon_f(k,N)}{\gamma(k-1,N)} \tag{8.2}$$

A simple manipulation of equation (7.73), leads to the following relation for the time updating of the minimum weighted least-squares error, which will be used in the FTRLS algorithm:

$$\xi^d_{f_{\min}}(k,N) = \lambda\xi^d_{f_{\min}}(k-1,N) + e_f(k,N)\varepsilon_f(k,N) \tag{8.3}$$

From the same equation (7.73), we can obtain the following equality that will also be required in the FTRLS algorithm:

$$\gamma(k,N+1) = \frac{\lambda\xi^d_{f_{\min}}(k-1,N)}{\xi^d_{f_{\min}}(k,N)}\gamma(k-1,N) \tag{8.4}$$

The updating equation of the forward prediction tap-coefficient vector can be performed through equation (7.40) of the previous chapter, i.e.,

$$\mathbf{w}_f(k,N) = \mathbf{w}_f(k-1,N) + \boldsymbol{\phi}(k-1,N)e_f(k,N) \tag{8.5}$$

where $\boldsymbol{\phi}(k-1,N) = \mathbf{S}_D(k-1,N)\mathbf{x}(k-1,N)$.

As will be seen, the updating of vector $\boldsymbol{\phi}(k-1,N)$ to $\boldsymbol{\phi}(k,N+1)$ is needed to update the backward predictor coefficient vector. Also, the last element of $\boldsymbol{\phi}(k,N+1)$ is used to update the backward prediction *a priori* error and to obtain $\gamma(k,N)$. Vector $\boldsymbol{\phi}(k,N+1)$ can be obtained by post-multiplying both sides of equation (7.56), at instant k and for order N, by $\mathbf{x}(k,N+1) = [x(k)\mathbf{x}^T(k-1,N)]^T$. The result can be expressed as

$$\boldsymbol{\phi}(k,N+1) = \begin{bmatrix} 0 \\ \boldsymbol{\phi}(k-1,N) \end{bmatrix} + \frac{1}{\xi^d_{f_{\min}}(k,N)} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k,N) \end{bmatrix} \varepsilon_f(k,N) \tag{8.6}$$

However, it is not convenient to use the above equation in the FTRLS algorithm because when deriving the backward prediction part, it would lead to extra computation. The solution is to use an alternative recursion involving $\hat{\boldsymbol{\phi}}(k,N+1) = \frac{\boldsymbol{\phi}(k,N+1)}{\gamma(k,N+1)}$ instead of $\boldsymbol{\phi}(k,N+1)$ (see problem 7 for further details). The resulting recursion can be derived after some algebraic manipulations of equations (8.6) and (8.3) to (8.5), leading to

$$\hat{\boldsymbol{\phi}}(k,N+1) = \begin{bmatrix} 0 \\ \hat{\boldsymbol{\phi}}(k-1,N) \end{bmatrix} + \frac{1}{\lambda\xi^d_{f_{\min}}(k-1,N)} \begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix} e_f(k,N) \tag{8.7}$$

The forward prediction tap-coefficient vector should then be updated using $\hat{\boldsymbol{\phi}}(k-1,N)$ as

$$\mathbf{w}_f(k,N) = \mathbf{w}_f(k-1,N) + \hat{\boldsymbol{\phi}}(k-1,N)\varepsilon_f(k,N) \tag{8.8}$$

8.2.2 Backward Prediction Relations

In this subsection, the relations involving the backward prediction problem that are used in the FTRLS algorithm are derived.

The relationship between *a posteriori* and *a priori* backward prediction errors can be expressed as

$$\varepsilon_b(k,N) = e_b(k,N)\gamma(k,N) \tag{8.9}$$

It is also known that the ratio of conversion factors for different orders is given by

$$\frac{\gamma(k,N+1)}{\gamma(k,N)} = \frac{\lambda\xi^d_{b_{\min}}(k-1,N)}{\xi^d_{b_{\min}}(k,N)} \tag{8.10}$$

see equation (7.79) of the previous chapter.

We rewrite for convenience the last equality of equation (7.70), i.e.,

$$\xi^d_{b_{\min}}(k,N) = \lambda\xi^d_{b_{\min}}(k-1,N) + \frac{\varepsilon_b^2(k,N)}{\gamma(k,N)} \tag{8.11}$$

This equation can be rewritten as

$$1 + \frac{\varepsilon_b^2(k,N)}{\lambda\gamma(k,N)\xi^d_{b_{\min}}(k-1,N)} = \frac{\xi^d_{b_{\min}}(k,N)}{\lambda\xi^d_{b_{\min}}(k-1,N)} \tag{8.12}$$

Now we recall that the time updating for the backward predictor filter is given by

$$\begin{aligned}\mathbf{w}_b(k,N) &= \mathbf{w}_b(k-1,N) + \boldsymbol{\phi}(k,N)e_b(k,N)\\ &= \mathbf{w}_b(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)\varepsilon_b(k,N)\end{aligned} \tag{8.13}$$

Following a similar approach to that used to derive equation (8.7), by first post-multiplying both sides of equation (7.59), at instant k and for order N, by $\mathbf{x}(k,N+1) = [\mathbf{x}^T(k,N)\, x(k-N)]^T$, and using relations (8.10), (8.11), and (8.13), we have

$$\begin{bmatrix} \hat{\boldsymbol{\phi}}(k,N) \\ 0 \end{bmatrix} = \hat{\boldsymbol{\phi}}(k,N+1) - \frac{1}{\lambda\xi^d_{b_{\min}}(k-1,N)} \begin{bmatrix} -\mathbf{w}_b(k-1,N) \\ 1 \end{bmatrix} e_b(k,N) \tag{8.14}$$

Note that in this equation the last element of $\hat{\boldsymbol{\phi}}(k,N+1)$ was already calculated in equation (8.7). In any case, it is worth mentioning that the last element of $\hat{\boldsymbol{\phi}}(k,N+1)$ can alternatively be expressed as

$$\hat{\phi}_{N+1}(k,N+1) = \frac{e_b(k,N)}{\lambda\xi^d_{b_{\min}}(k-1,N)} \tag{8.15}$$

By applying equations (8.9), (8.15), and (8.10) in equation (8.12), we can show that

$$1 + \hat{\phi}_{N+1}(k,N+1)\varepsilon_b(k,N) = \frac{\gamma(k,N)}{\gamma(k,N+1)} \tag{8.16}$$

By substituting equation (8.9) into the above equation, we can now derive an updating equation that can be used in the FTRLS algorithm as

$$\gamma^{-1}(k,N) = \gamma^{-1}(k,N+1) - \hat{\boldsymbol{\phi}}_{N+1}(k,N+1)e_b(k,N) \tag{8.17}$$

The updating equations related to the forward and backward prediction problems and for the conversion factor $\gamma(k,N)$ are now available. We can thus proceed with the derivations to solve the more general problem of estimating a related process represented by the desired signal $d(k)$, known as joint-process estimation.

8.3 JOINT-PROCESS ESTIMATION

As for all previously presented adaptive-filter algorithms, it is useful to derive a FTRLS algorithm that can match a desired signal $d(k)$ through the minimization of the weighted squared error. Starting with the *a priori* error

$$e(k,N) = d(k) - \mathbf{w}^T(k-1,N)\mathbf{x}(k,N) \tag{8.18}$$

we can calculate the *a posteriori* error as

$$\varepsilon(k,N) = e(k,N)\gamma(k,N) \tag{8.19}$$

As in the conventional RLS algorithm, the time updating for the output tap coefficients of the joint-process estimator can be performed as

$$\begin{aligned}\mathbf{w}(k,N) &= \mathbf{w}(k-1,N) + \boldsymbol{\phi}(k,N)e(k,N) \\ &= \mathbf{w}(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)\varepsilon(k,N)\end{aligned} \tag{8.20}$$

All the updating equations are now available to describe the fast transversal RLS algorithm. The FRLS algorithm consists of equations (8.1)-(8.3), (8.7)-(8.8), and (8.4) related to the forward predictor; equations (8.15), (8.17), (8.9), (8.11), (8.14), and (8.13) related to the backward predictor and the conversion factor; and (8.18)-(8.20) related to the joint-process estimator. The FTRLS algorithm is in step-by-step form as Algorithm 8.1. The computational complexity of the FTRLS algorithm is $7(N)+14$ multiplications per output sample. The key feature of the FTRLS algorithm is that it does not require matrix multiplications. Because of this, the implementation of the FTRLS algorithm has complexity of order N multiplications per output sample.

The initialization procedure consists of setting the tap coefficients of the backward prediction, forward prediction, and joint-process estimation filters to zero, namely

$$\mathbf{w}_f(-1,N) = \mathbf{w}_b(-1,N) = \mathbf{w}(-1,N) = \mathbf{0} \tag{8.21}$$

Vector $\hat{\boldsymbol{\phi}}(-1,N)$ is set to $\mathbf{0}$ assuming that the input and desired signals are zero for $k < 0$, i.e., prewindowed data. The conversion factor should be initialized as

$$\gamma(-1,N) = 1 \tag{8.22}$$

Algorithm 8.1

Fast Transversal RLS Algorithm

Initialization

$\mathbf{w}_f(-1,N) = \mathbf{w}_b(-1,N) = \mathbf{w}(-1,N) = \mathbf{0}$
$\hat{\boldsymbol{\phi}}(-1,N) = \mathbf{0},\ \gamma(-1,N) = 1$
$\xi^d_{b_{\min}}(-1,N) = \xi^d_{f_{\min}}(-1,N) = \epsilon$ (a small positive constant)

Prediction Part

Do for each $k \geq 0$,

$$e_f(k,N) = \mathbf{x}^T(k,N+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix}$$

$$\varepsilon_f(k,N) = e_f(k,N)\gamma(k-1,N) \tag{8.2}$$

$$\xi^d_{f_{\min}}(k,N) = \lambda\xi^d_{f_{\min}}(k-1,N) + e_f(k,N)\varepsilon_f(k,N) \tag{8.3}$$

$$\mathbf{w}_f(k,N) = \mathbf{w}_f(k-1,N) + \hat{\boldsymbol{\phi}}(k-1,N)\varepsilon_f(k,N) \tag{8.8}$$

$$\hat{\boldsymbol{\phi}}(k,N+1) = \begin{bmatrix} 0 \\ \hat{\boldsymbol{\phi}}(k-1,N) \end{bmatrix} + \frac{1}{\lambda\xi^d_{f_{\min}}(k-1,N)}\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix} e_f(k,N) \tag{8.7}$$

$$\gamma(k,N+1) = \frac{\lambda\xi^d_{f_{\min}}(k-1,N)}{\xi^d_{f_{\min}}(k,N)}\gamma(k-1,N) \tag{8.4}$$

$$e_b(k,N) = \lambda\xi^d_{b_{\min}}(k-1,N)\hat{\boldsymbol{\phi}}_{N+1}(k,N+1) \tag{8.15}$$

$$\gamma^{-1}(k,N) = \gamma^{-1}(k,N+1) - \hat{\boldsymbol{\phi}}_{N+1}(k,N+1)e_b(k,N) \tag{8.17}$$

$$\varepsilon_b(k,N) = e_b(k,N)\gamma(k,N) \tag{8.9}$$

$$\xi^d_{b_{\min}}(k,N) = \lambda\xi^d_{b_{\min}}(k-1,N) + \varepsilon_b(k,N)e_b(k,N) \tag{8.11}$$

$$\begin{bmatrix} \hat{\boldsymbol{\phi}}(k,N) \\ 0 \end{bmatrix} = \hat{\boldsymbol{\phi}}(k,N+1) - \hat{\boldsymbol{\phi}}_{N+1}(k,N+1)\begin{bmatrix} -\mathbf{w}_b(k-1,N) \\ 1 \end{bmatrix} \tag{8.14}$$

$$\mathbf{w}_b(k,N) = \mathbf{w}_b(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)\varepsilon_b(k,N) \tag{8.13}$$

Joint-Process Estimation

$$e(k,N) = d(k) - \mathbf{w}^T(k-1,N)\mathbf{x}(k,N) \tag{8.18}$$

$$\varepsilon(k,N) = e(k,N)\gamma(k,N) \tag{8.19}$$

$$\mathbf{w}(k,N) = \mathbf{w}(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)\varepsilon(k,N) \tag{8.20}$$

End

since no difference between *a priori* and *a posteriori* errors exists during the initialization period. The weighted least-square errors should be initialized with a positive constant ϵ

$$\epsilon = \xi^d_{f_{\min}}(-1,N) = \xi^d_{b_{\min}}(-1,N) \tag{8.23}$$

in order to avoid division by zero in the first iteration. The reason for introducing this initialization parameter suggests that it should be a small value. However, for stability reasons, the value of ϵ should not be small (see the examples at the end of this chapter).

It should be mentioned that there are exact initialization procedures for the fast transversal RLS filters with the aim of minimizing the objective function at all instants during the initialization period [3].

These procedures explore the fact that during the initialization period the number of data samples in both $d(k)$ and $x(k)$ is less than $N + 1$. Therefore the objective function can be made zero since there are more parameters than needed. The exact initialization procedure of [3] replaces the computationally intensive backsubstitution algorithm and is rather simple when the adaptive-filter coefficients are initialized with zero. The procedure can also be generalized to the case where some nonzero initial values for the tap coefficients are available.

As previously mentioned, several fast RLS algorithms based on the transversal realization exist; the one presented here corresponds to the so-called FTF proposed in [3]. A number of alternative algorithms are introduced in the problems.

8.4 STABILIZED FAST TRANSVERSAL RLS ALGORITHM

Although the fast transversal algorithms proposed in the literature provide a nice solution to the computational complexity burden inherent to the conventional RLS algorithm, these algorithms are unstable when implemented with finite-precision arithmetic. Increasing the wordlength does not solve the instability problem. The only effect of employing a longer wordlength is that the algorithm will take longer to diverge. Earlier solutions to this problem consisted of restarting the algorithm when the accumulated errors in chosen variables reached prescribed thresholds [3]. Although the restart procedure would use past information, the resulting performance is suboptimal due to the discontinuity of information in the corresponding deterministic correlation matrix.

The cause for the unstable behavior of the fast transversal algorithms is the inherent positive feedback mechanism. This explanation led to the idea that if some specific measurements of the numerical errors were available, they could conveniently be fed back in order to make the negative feedback dominant in the error propagation dynamics. Fortunately, some measurements of the numerical errors can be obtained by introducing computational redundancy into the fast algorithm. Such a computational redundancy would involve calculating a given quantity using two different formulas. In finite-precision implementation, the resulting values for the quantity calculated by these formulas are not equal and their difference is a good measurement of the accumulated errors in that quantity. This error can then be fed back in an attempt to stabilize the algorithm. The key problem is to determine the quantities where the computational redundancy should be introduced such that the error propagation dynamics can be stabilized. In the early proposed solutions [6]-[7], only a single quantity was chosen to introduce the redundancy. Later, it was shown that at least two quantities are required in order to guarantee the stability of the FTRLS algorithm [9]. Another relevant question is where should the error be fed back inside the algorithm. Note that any point could be chosen without affecting the behavior of the algorithm when implemented with infinite precision, since the feedback error is zero in this case. A natural choice is to feed the error back into the expressions of the quantities that are related to it. That means for each quantity in which redundancy is introduced, its final value is a combination of the two forms of computing it.

The FTRLS algorithm can be seen as a discrete-time nonlinear dynamic system [9]: when finite precision is used in the implementation, quantization errors will rise. In this case, the internal quantities will be perturbed when compared with the infinite-precision quantities. When modeling

the error propagation, a nonlinear system can be described that, if properly linearized, allows the study of the error propagation mechanism. Using an averaging analysis, which is meaningful for stationary input signals, it is possible to obtain a system characterized by its set of eigenvalues whose dynamic behavior is similar to that of the error propagation behavior when $k \to \infty$ and $(1-\lambda) \to 0$. Through these eigenvalues, it is possible to determine the feedback parameters as well as the quantities to choose for the introduction of redundancy. The objective here is to modify the unstable modes through the error feedback in order to make them stable [9]. Fortunately, it was found in [9] that the unstable modes can be modified and stabilized by the introduced error feedback. The unstable modes can be modified by introducing redundancy in $\gamma(k,N)$ and $e_b(k,N)$. These quantities can be calculated using different relations and in order to distinguish them an extra index is included in their description.

The *a priori* backward error can be described in a number of alternative forms such as

$$e_b(k,N,1) = \lambda \xi^d_{b_{\min}}(k-1,N)\hat{\boldsymbol{\phi}}_{N+1}(k,N+1) \tag{8.24}$$

$$e_b(k,N,2) = \left[-\mathbf{w}_b^T(k-1,N)\ \ 1\right]\mathbf{x}(k,N+1) \tag{8.25}$$

and

$$\begin{aligned} e_{b,i}(k,N,3) &= e_b(k,N,2)\kappa_i + e_b(k,N,1)[1-\kappa_i] \\ &= e_b(k,N,1) + \kappa_i[e_b(k,N,2) - e_b(k,N,1)] \end{aligned} \tag{8.26}$$

where the first form was employed in the FTRLS algorithm and the second form corresponds to the inner product implementation of the *a priori* backward error. The third form corresponds to a linear combination of the first two forms where the numerical difference between these forms is fed back to determine the final value of $e_{b,i}(k,N,3)$ which will be used at different places in the stabilized algorithm. For each $\kappa_i, i = 1,2,3$, we choose a different value in order to guarantee that the related eigenvalues are less than one.

The conversion factor $\gamma(k,N)$ is probably the first parameter to show signs that the algorithm is becoming unstable. This parameter can also be calculated through different relations. These alternative relations are required to guarantee that all modes of the error propagation system become stable. The first equation is given by

$$\begin{aligned} \gamma^{-1}(k,N+1,1) &= \gamma^{-1}(k-1,N,3)\frac{\xi^d_{f_{\min}}(k,N)}{\lambda\xi^d_{f_{\min}}(k-1,N)} \\ &= \gamma^{-1}(k-1,N,3)\left[1 + \frac{e_f(k,N)\varepsilon_f(k,N)}{\lambda\xi^d_{f_{\min}}(k-1,N)}\right] \\ &= \gamma^{-1}(k-1,N,3) + \frac{e_f^2(k,N)}{\lambda\xi^d_{f_{\min}}(k-1,N)} \\ &= \gamma^{-1}(k-1,N,3) + \hat{\phi}_0(k,N+1)e_f(k,N) \end{aligned} \tag{8.27}$$

where $\hat{\phi}_0(k,N+1)$ is the first element of $\hat{\boldsymbol{\phi}}(k,N+1)$. The above equalities are derived from equations (8.4), (8.3), (8.2) and (8.7), respectively. The second expression for the conversion factor is derived from equation (8.14) and given by

$$\gamma^{-1}(k,N,2) = \gamma^{-1}(k,N+1,1) - \hat{\boldsymbol{\phi}}_{N+1}(k,N+1)e_{b,3}(k,N,3) \tag{8.28}$$

The third expression is

$$\gamma^{-1}(k,N,3) = 1 + \hat{\boldsymbol{\phi}}^T(k,N)\mathbf{x}(k,N) \tag{8.29}$$

In equation (8.27), the conversion factor was expressed in different ways, one of which was first presented in the FTRLS algorithm of [9]. The second form already uses an *a priori* backward error with redundancy. The third form can be derived from equation (7.48) for the lattice RLS algorithms (see problem 10).

An alternative relation utilized in the stabilized fast transversal algorithm involves the minimum forward least-squares error. From equations (8.3) and (8.7), we can write

$$\begin{aligned}[\xi^d_{f_{\min}}(k,N)]^{-1} &= \lambda^{-1}[\xi^d_{f_{\min}}(k-1,N)]^{-1} - \frac{e_f(k,N)\varepsilon_f(k,N)}{\lambda\xi^d_{f_{\min}}(k-1,N)\xi^d_{f_{\min}}(k,N)} \\ &= \lambda^{-1}[\xi^d_{f_{\min}}(k-1,N)]^{-1} - \frac{\hat{\boldsymbol{\phi}}_0(k,N)\varepsilon_f(k,N)}{\xi^d_{f_{\min}}(k,N)}\end{aligned}$$

From (8.6), we can deduce that

$$\frac{\varepsilon_f(k,N)}{\xi^d_{f_{\min}}(k,N)} = \boldsymbol{\phi}_0(k,N) = \hat{\boldsymbol{\phi}}_0(k,N)\gamma(k,N+1,1)$$

With this relation, we can obtain the desired equation as

$$[\xi^d_{f_{\min}}(k,N)]^{-1} = \lambda^{-1}[\xi^d_{f_{\min}}(k-1,N)]^{-1} - \gamma(k,N+1,1)\hat{\boldsymbol{\phi}}_0^2(k,N+1) \tag{8.30}$$

where the choice of $\gamma(k,N+1,1)$ is used to keep the error-system modes stable [9].

Using the equations for the conversion factor and for the *a priori* backward error with redundancy, we can obtain the stabilized fast transversal RLS algorithm (SFTRLS) whose step-by-step implementation is given as Algorithm 8.2. The parameters κ_i for $i = 1, 2, 3$ were determined through computer simulation search [9] where the optimal values found were $\kappa_1 = 1.5$, $\kappa_2 = 2.5$, and $\kappa_3 = 1$. It was also found in [9] that the numerical behavior is quite insensitive to values of κ_i around the optimal and that optimal values chosen for a given situation work well for a wide range of environments and algorithm setup situations (for example, for different choices of the forgetting factor).

Another issue related to the SFTRLS algorithm concerns the range of values for λ such that stability is guaranteed. Results of extensive simulation experiments [9] indicate that the range is

$$1 - \frac{1}{2(N+1)} \leq \lambda < 1 \tag{8.31}$$

where N is the order of the adaptive filter. It was also verified that the optimal numerical behavior is achieved when the value of λ is chosen as

$$\lambda = 1 - \frac{0.4}{N+1} \tag{8.32}$$

Algorithm 8.2

Stabilized Fast Transversal RLS Algorithm

Initialization

$\mathbf{w}_f(-1,N) = \mathbf{w}_b(-1,N) = \mathbf{w}(-1,N) = \mathbf{0}$
$\hat{\boldsymbol{\phi}}(-1,N) = \mathbf{0},\ \gamma(-1,N,3) = 1$
$\xi^d_{b_{\min}}(-1,N) = \xi^d_{f_{\min}}(-1,N) = \epsilon$ (a small positive constant)
$\kappa_1 = 1.5, \kappa_2 = 2.5, \kappa_3 = 1$

Prediction Part

Do for each $k \geq 0$,

$$e_f(k,N) = \mathbf{x}^T(k,N+1)\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix}$$
$$\varepsilon_f(k,N) = e_f(k,N)\gamma(k-1,N,3) \tag{8.2}$$
$$\hat{\boldsymbol{\phi}}(k,N+1) = \begin{bmatrix} 0 \\ \hat{\boldsymbol{\phi}}(k-1,N) \end{bmatrix} + \frac{1}{\lambda\xi^d_{f_{\min}}(k-1,N)}\begin{bmatrix} 1 \\ -\mathbf{w}_f(k-1,N) \end{bmatrix} e_f(k,N) \tag{8.7}$$
$$\gamma^{-1}(k,N+1,1) = \gamma^{-1}(k-1,N,3) + \hat{\phi}_0(k,N+1)e_f(k,N) \tag{8.27}$$
$$[\xi^d_{f_{\min}}(k,N)]^{-1} = \lambda^{-1}[\xi^d_{f_{\min}}(k-1,N)]^{-1} - \gamma(k,N+1,1)\hat{\phi}_0^2(k,N+1) \tag{8.30}$$
$$\mathbf{w}_f(k,N) = \mathbf{w}_f(k-1,N) + \hat{\boldsymbol{\phi}}(k-1,N)\varepsilon_f(k,N) \tag{8.8}$$
$$e_b(k,N,1) = \lambda\xi^d_{b_{\min}}(k-1,N)\hat{\phi}_{N+1}(k,N+1) \tag{8.15}$$
$$e_b(k,N,2) = \begin{bmatrix}-\mathbf{w}_b^T(k-1,N) & 1\end{bmatrix}\mathbf{x}(k,N+1) \tag{8.25}$$
$$e_{b,i}(k,N,3) = e_b(k,N,2)\kappa_i + e_b(k,N,1)[1-\kappa_i] \ \ for\ i = 1,2,3 \tag{8.25}$$
$$\gamma^{-1}(k,N,2) = \gamma^{-1}(k,N+1,1) - \hat{\phi}_{N+1}(k,N+1)e_{b,3}(k,N,3) \tag{8.28}$$
$$\varepsilon_{b,j}(k,N,3) = e_{b,j}(k,N,3)\gamma(k,N,2)\ \ j = 1,2$$
$$\xi^d_{b_{\min}}(k,N) = \lambda\xi^d_{b_{\min}}(k-1,N) + \varepsilon_{b,2}(k,N,3)e_{b,2}(k,N,3) \tag{8.11}$$
$$\begin{bmatrix} \hat{\boldsymbol{\phi}}(k,N) \\ 0 \end{bmatrix} = \hat{\boldsymbol{\phi}}(k,N+1) - \hat{\phi}_{N+1}(k,N+1)\begin{bmatrix} -\mathbf{w}_b(k-1,N) \\ 1 \end{bmatrix} \tag{8.14}$$
$$\mathbf{w}_b(k,N) = \mathbf{w}_b(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)\varepsilon_{b,1}(k,N,3) \tag{8.13}$$
$$\gamma^{-1}(k,N,3) = 1 + \hat{\boldsymbol{\phi}}^T(k,N)\mathbf{x}(k,N) \tag{8.29}$$

Joint-Process Estimation

$$e(k,N) = d(k) - \mathbf{w}^T(k-1,N)\mathbf{x}(k,N) \tag{8.18}$$
$$\varepsilon(k,N) = e(k,N)\gamma(k,N,3) \tag{8.19}$$
$$\mathbf{w}(k,N) = \mathbf{w}(k-1,N) + \hat{\boldsymbol{\phi}}(k,N)\varepsilon(k,N) \tag{8.20}$$

End

The range of values for λ as well as its optimal value can be very close to one for high-order filters. This can be a potential limitation for the use of the SFTRLS algorithm, especially in nonstationary environments where smaller values for λ are required.

The computational complexity of the SFTRLS algorithm is of order $9N$ multiplications per output sample. There is an alternative algorithm with computational complexity of order $8N$ (see problem 9).

Before leaving this section, it is worth mentioning a nice interpretation for the fast transversal RLS algorithm. The FTRLS algorithm can be viewed as four transversal filters working in parallel and exchanging quantities with each other, as depicted in Fig. 8.1. The first filter is the forward prediction filter that utilizes $\mathbf{x}(k-1,N)$ as the input signal vector, $\mathbf{w}_f(k,N)$ as the coefficient vector, and provides quantities $\varepsilon_f(k,N), e_f(k,N)$, and $\xi^d_{f_{\min}}(k,N)$ as outputs. The second filter is the backward prediction filter that utilizes $\mathbf{x}(k,N)$ as the input signal vector, $\mathbf{w}_b(k,N)$ as the coefficient vector, and provides quantities $\varepsilon_b(k,N), e_b(k,N)$, and $\xi^d_{b_{\min}}(k,N)$ as outputs. The third filter is an auxiliary filter whose coefficients are given by $-\hat{\phi}(k,N)$, whose input signal vector is $\mathbf{x}(k,N)$, and whose output parameter is $\gamma^{-1}(k,N)$. For this filter, the desired signal vector is constant and equal to $[1\ 0\ 0\ldots 0]^T$. The fourth and last filter is the joint-process estimator whose input signal vector is $\mathbf{x}(k,N)$, whose coefficient vector is $\mathbf{w}(k,N)$, and which provides the quantities $\varepsilon(k,N)$ and $e(k,N)$ as outputs.

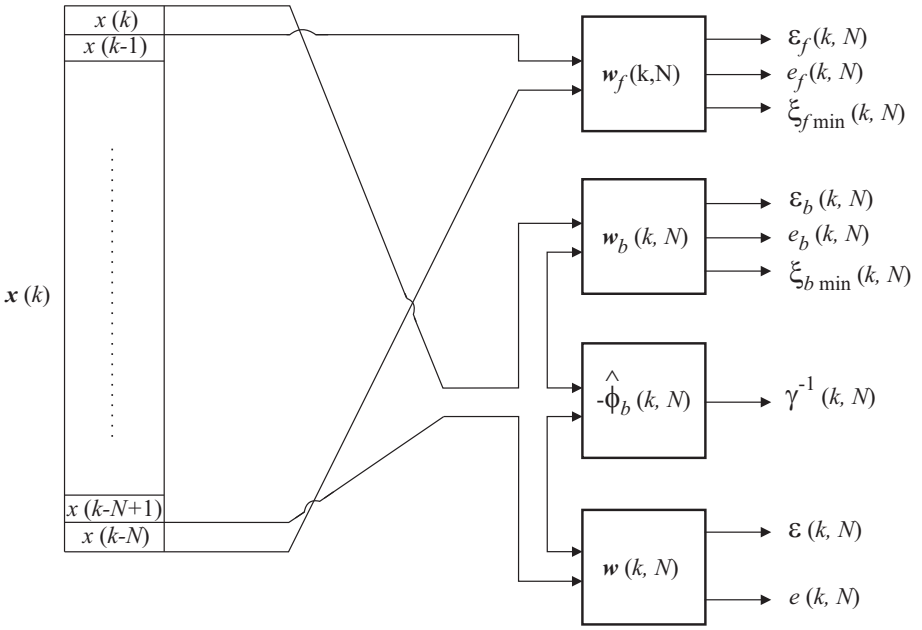

Figure 8.1 Fast transversal RLS algorithm: block diagram.

Example 8.1

The system identification problem described in subsection 3.6.2 is solved using the stabilized fast transversal algorithm presented in this chapter. The main objective is to check the stability of the algorithm when implemented in finite precision.

Solution:

According to equation (8.31), the lower bound for λ in this case is 0.9375. A value $\lambda = 0.99$ is chosen. The stabilized fast transversal algorithm is applied to solve the identification problem and the measured MSE is 0.0432.

Using $\epsilon = 2$, we ran the algorithm with finite precision and the results are summarized in Table 8.1. No sign of instability is found for $\lambda = 0.99$. These results are generated by ensemble averaging 200 experiments. A comparison of the results of Table 8.1 with those of Tables 5.2 and 7.2 shows that the SFTRLS algorithm has similar performance compared to the conventional and lattice-based RLS algorithms, in terms of quantization error accumulation. The question is which algorithm remains stable in most situations. Regarding the SFTRLS, for large-order filters we are left with a limited range of values to choose λ. Also, it was found in our experiments that the choice of the initialization parameter ϵ plays an important role in the performance of this algorithm when implemented in finite precision. In some cases, even when the value of λ is within the recommended range, the algorithm does not converge if ϵ is small. By increasing the value of ϵ, we increase the usual convergence time while keeping the algorithm stable.

□

Table 8.1 Results of the Finite-Precision Implementation of the SFTRLS Algorithm

	$\xi(k)_Q$	$E[\|\|\Delta\mathbf{w}(k)_Q\|\|^2]$
No of bits	**Experiment**	**Experiment**
16	$1.545\ 10^{-3}$	$6.089\ 10^{-5}$
12	$1.521\ 10^{-3}$	$3.163\ 10^{-5}$
10	$1.562\ 10^{-3}$	$6.582\ 10^{-5}$

Example 8.2

The channel equalization example described in subsection (3.6.3) is also used in simulations to test the SFTRLS algorithm. We use a 25th-order equalizer and a forgetting factor $\lambda = 0.99$.

Solution:

In order to solve the equalization problem the stabilized fast transversal RLS algorithm is initialized with $\epsilon = 0.5$. The results presented here were generated by ensemble averaging 200 experiments. The resulting learning curve of the MSE is shown in Fig. 8.2, and the measured MSE is 0.2973. The overall performance of the SFTRLS algorithm for this particular example is as good as any other RLS algorithm, such as lattice-based algorithms.

□

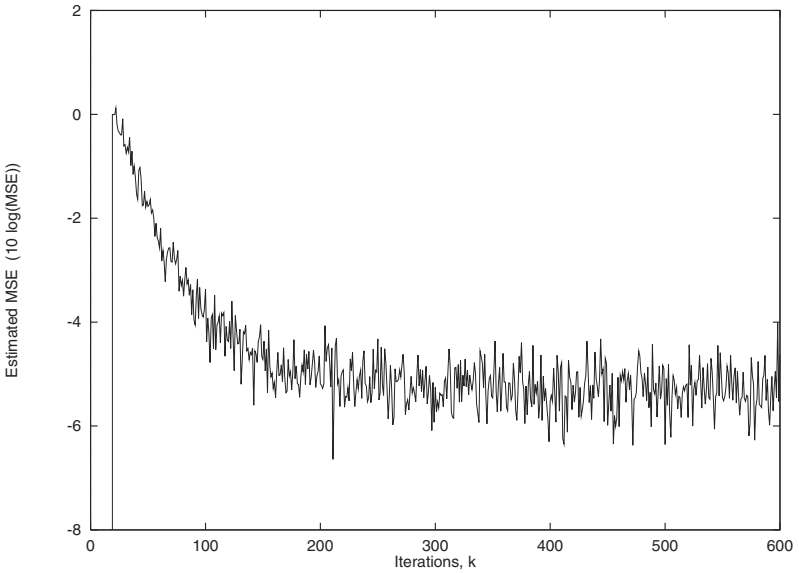

Figure 8.2 Learning curves for the stabilized fast transversal RLS algorithm.

8.5 CONCLUDING REMARKS

In this chapter we have presented some fast transversal RLS algorithms. This class of algorithms is computationally more efficient than conventional and lattice-based RLS algorithms. Some simulation examples were included where the SFTRLS algorithm was employed. The finite-wordlength simulations are of special interest for the reader.

A number of alternative FTRLS algorithms as well as theoretical results can be found in [3]. The derivation of normalized versions of the FTRLS algorithm is also possible and was not addressed in the present chapter, for this result refer to [4]. The most computationally efficient FTRLS algorithms are known to be unstable. The error-feedback approach was briefly introduced that allowed the stabilization of the FTRLS algorithm. The complete derivation and justification for the error-feedback approach is given in [9].

In nonstationary environments, it might be useful to employ a time-varying forgetting factor. Therefore it is desirable to obtain FTRLS algorithms allowing the use of variable λ. This problem was first addressed in [11]. However a computationally more efficient solution was proposed in [8] where the concept of data weighting was introduced to replace the concept of error weighting.

The FTRLS algorithm has potential for a number of applications. In particular, the problem in which the signals available from the environment are noisy version of a transmitted signal and noisy filtered versions of the same transmitted signal is an interesting application. In this problem, both the delay and unknown filter coefficients have to be estimated. The weighted squared errors have to

be minimized while considering both the delay and the unknown system parameters. This problem of joint estimation can be elegantly solved by employing the FTRLS algorithm [12].

8.6 REFERENCES

1. D. D. Falconer and L. Ljung, "Application of fast Kalman estimation to adaptive equalization," *IEEE Trans. on Communications*, vol. COM-26, pp. 1439-1446, Oct. 1978.

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

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

4. J. M. Cioffi and T. Kailath, "Windowed fast transversal filters adaptive algorithms with normalization," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. ASSP-33, pp. 607-627, June 1985.

5. S. Ljung and L. Ljung, "Error propagation properties of recursive least-squares adaptation algorithms," *Automatica*, vol. 21, pp. 157-167, 1985.

6. J.-L. Botto and G. V. Moustakides, "Stabilizing the fast Kalman algorithms," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 37, pp. 1342-1348, Sept. 1989.

7. M. Bellanger, "Engineering aspects of fast least squares algorithms in transversal adaptive filters," *Proc. IEEE Intern. Conf. on Acoust., Speech, Signal Processing*, pp. 49.14.1-49.14.4, 1987.

8. D. T. M. Slock and T. Kailath, "Fast transversal filters with data sequence weighting," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 37, pp. 346-359, March 1989.

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, pp. 92-113, Jan. 1991.

10. J. G. Proakis, C. M. Rader, F. Ling, and C. L. Nikias, *Advanced Digital Signal Processing*, MacMillan, New York, NY, 1992.

11. B. Toplis and S. Pasupathy, "Tracking improvements in fast RLS algorithms using a variable forgetting factor," *IEEE Trans. on Acoust., Speech, and Signal Processing*, vol. 36, pp. 206-227, Feb. 1988.

12. D. Boudreau and P. Kabal, "Joint-time delay estimation and adaptive recursive least squares filtering," *IEEE Trans. on Signal Processing*, vol. 41, pp. 592-601, Feb. 1993.

8.7 PROBLEMS

1. Show that

$$\begin{aligned}\boldsymbol{\phi}(k,N) &= \mathbf{S}_D(k,N)\mathbf{x}(k,N) \\ &= \frac{\mathbf{S}_D(k-1,N)\mathbf{x}(k,N)}{\lambda + \mathbf{x}^T(k,N)\mathbf{S}_D(k-1,N)\mathbf{x}(k,N)}\end{aligned}$$

   Hint: Use the matrix inversion lemma for $\mathbf{S}_D(k,N)$.

2. Show that

$$\boldsymbol{\phi}_N(k-1,N) - \frac{\mathbf{w}_{f,N}(k)\varepsilon_f(k,N)}{\xi^d_{f_{\min}}(k,N)} = \frac{-\varepsilon_b(k,N)}{\xi^d_{b_{\min}}(k,N)} = \boldsymbol{\phi}_{N+1}(k,N+1)$$

   where $\mathbf{w}_{f,N}(k)$ represents the last element of $\mathbf{w}_f(k,N)$.

3. Using a proper mixture of relations of the lattice RLS algorithm based on *a posteriori* and the FTRLS algorithm, derive a fast exact initialization procedure for the transversal filter coefficients.

4. Show that the following relations are valid, assuming the input signals are prewindowed:

$$\frac{\det[\mathbf{S}_D(k,N+1)]}{\det[\mathbf{S}_D(k-1,N)]} = \frac{1}{\xi^d_{f_{\min}}(k,N)}$$

$$\frac{\det[\mathbf{S}_D(k,N+1)]}{\det[\mathbf{S}_D(k,N)]} = \frac{1}{\xi^d_{b_{\min}}(k,N)}$$

5. Show that

$$\gamma^{-1}(k,N) = \frac{\det[\mathbf{R}_D(k,N)]}{\lambda^N \det[\mathbf{R}_D(k-1,N)]}$$

   Hint: $\det[\mathbf{I}+\mathbf{AB}] = \det[\mathbf{I}+\mathbf{BA}]$.

6. Using the results of problems 4 and 5, prove that

$$\gamma^{-1}(k,N) = \frac{\xi^d_{f_{\min}}(k,N)}{\lambda^N \xi^d_{b_{\min}}(k,N)}$$

7. Derive equations (8.7) and (8.14). Also show that the use of $\boldsymbol{\phi}(k,N)$ would increase the computational complexity of the FTRLS algorithm.

8. If one avoids the use of the conversion factor $\gamma(k,N)$, it is necessary to use inner products to derive the *a posteriori* errors in the fast algorithm. Derive a fast algorithm without the conversion factor.

9. By replacing the relation for $\gamma(k, N, 3)$ in the SFTRLS algorithm by the relation

$$\gamma(k, N) = \frac{\lambda^N \xi^d_{b_{\min}}(k, N)}{\xi^d_{f_{\min}}(k, N)}$$

derived in problem 6, describe the resulting algorithm and show that it requires order $8N$ multiplications per output sample.

10. Derive the equation (8.29).

11. The FTRLS algorithm is applied to predict the signal $x(k) = \sin(\frac{\pi k}{4} + \frac{\pi}{3})$. Given $\lambda = 0.98$, calculate the error and the tap coefficients for the first 10 iterations.

12. The SFTRLS algorithm is applied to predict the signal $x(k) = \sin(\frac{\pi k}{4} + \frac{\pi}{3})$. Given $\lambda = 0.98$, calculate the error and the tap coefficients for the first 10 iterations.

13. The FTRLS algorithm is applied to identify a 7th-order unknown system whose coefficients are

$$\mathbf{w}^T = [0.0272 \quad 0.0221 \quad -0.0621 \quad 0.1191 \quad 0.6116 \quad -0.3332 \quad -0.0190 \quad -0.0572]$$

The input signal is Gaussian white noise with variance $\sigma_x^2 = 1$ and the measurement noise is also Gaussian white noise independent of the input signal with variance $\sigma_n^2 = 0.01$.

Simulate the experiment above described and measure the excess MSE for $\lambda = 0.97$ and $\lambda = 0.98$.

14. Repeat problem 13 for the case where the input signal is a first-order Markov process with $\lambda_{\mathbf{x}} = 0.98$.

15. Redo problem 13 using a fixed-point implementation with the FTRLS and SFTRLS algorithms. Use 12 bits in the fractional part of the signal and parameter representations.

16. Suppose a 15th-order FIR digital filter with the multiplier coefficients given below is identified through an adaptive FIR filter of the same order using the FTRLS algorithm. Assuming fixed-point arithmetic, simulate the identification problem described in terms of the following specifications:

Additional noise : white noise with variance	$\sigma_n^2 = 0.0015$
Coefficients wordlength:	$b_c = 16$ bits
Signal wordlength:	$b_d = 16$ bits
Input signal: Gaussian white noise with variance	$\sigma_x^2 = 0.7$
	$\lambda = 0.98$

$$\mathbf{w}_o^T = [0.0219360 \quad 0.0015786 \quad -0.0602449 \quad -0.0118907 \quad 0.1375379$$
$$0.0574545 \quad -0.3216703 \quad -0.5287203 \quad -0.2957797 \quad 0.0002043 \quad 0.290670$$
$$-0.0353349 \quad -0.0068210 \quad 0.0026067 \quad 0.0010333 \quad -0.0143593]$$

Plot the learning curves for the finite- and infinite-precision implementations.

17. Repeat the above problem for the SFTRLS algorithm. Also reduce the wordlength used until a noticeable (10 percent increase) excess MSE is observed at the output.

18. Repeat problem 16 for the SFTRLS algorithm, using $\lambda = 0.999$ and $\lambda = 0.960$. Comment on the results.

19. The SFTRLS algorithm is used to perform the forward prediction of a signal $x(k)$ generated by applying zero-mean Gaussian white noise with unit variance to the input of a linear filter with transfer function given by

$$H(z) = \frac{0.5}{(1 - 1.512z^{-1} + 0.827z^{-2})(1 - 1.8z^{-1} + 0.87z^{-2})}$$

Calculate the zeros of the resulting predictor error transfer function and compare with the poles of the linear filter.

20. Perform the equalization of a channel with impulse response given by

$$h(k) = 0.96^k + (-0.9)^k$$

for $k = 0, 1, 2, \ldots, 15$. The transmitted signal is zero-mean Gaussian white noise with unit variance and the adaptive filter input signal-to-noise ratio is 30 dB. Use the SFTRLS algorithm of order 100.