FAST TECHNIQUES FOR COMPUTING FINITE-LENGTH MMSE DECISION FEEDBACK EQUALIZERS

Ricardo Merched *

Dept. of Electrical Engineering
Federal University of Rio de Janeiro, Brazil
E-mail: merched@lps.ufrj.br

Nabil R. Yousef

Broadband Systems Engineering Group
Broadcom Corporation, Irvine, CA
E-mail: nyousef@broadcom.com

ABSTRACT

The existing computationally efficient methods for computing the finite length minimum mean-square-error decision feedback equalizers (MMSE-DFE) rely on fast methods for Cholesky decomposition, which may face several implementation difficulties. As the demand for broadband communications increases, developing less complex methods is highly motivated. In this paper, we propose new techniques for computing the (MMSE-DFE) coefficients. The new algorithms are obtained by identifying the relationship between the feedforward equalizer computation and well known fast recursive least squares (RLS) adaptive algorithms, and the feedback equalizer as a convolution of the feedforward equalizer with the channel. The proposed algorithms are less complex, more structured, and more stable in finite precision than known methods encountered in the literature.

1. INTRODUCTION

Equalizers are essential building blocks of communication systems, especially in broadband applications where inter-symbol-interference is a critical problem. In many of such systems, the transmitted signal consists of a known training sequence followed by unknown data. An efficient equalization technique in this scenario, is to first estimate the channel impulse responses between the transmitter and receiver using the training sequence, and then use this estimate to compute the optimal decision feedback equalizer (DFE) tap coefficients corresponding to the estimated channel. The computed tap coefficients are then uploaded to the equalizer taps. This procedure should be repeated as often as possible, especially in cases of fast varying channels. Moreover, the received data stream is usually buffered during the period needed for channel estimation and the equalizer tap computation.

In this context, a critical factor for the success of this equalization structure is the complexity of the DFE tap computation. A fast computation technique has the following benefits:

1. It reduces the memory size required to buffer the received sequence during the period required for tap computations.

2. It allows more frequent uploading of new equalizer coefficients, thus enabling the equalizer to track fast channel variations.

3. It simplifies the needed hardware, especially if such computations are performed through structured recursive equations.

*R. Merched is with the Dept. of Electrical Engineering of Federal University of Rio de Janeiro, Brazil. His research is partially funded by CNPq, Brazil.

Now, the current most efficient method for computing the optimal tap coefficients of a MMSE-DFE rely on the use of *generalized Schur* algorithm for fast Cholesky decomposition of the matrices involved in both feedback and feedforward filters computation [1].

In this paper, we propose an efficient method for computing the optimal MMSE-DFE coefficients via some defining relations of fast recursive-least-squares (RLS) adaptive algorithms. The proposed fast algorithm has less overall computational complexity (for the same channel and DFE filter lengths) and appears to be more stable in finite precision than the method in [1]. Moreover, unlike the method in [1], the proposed method relies on the use of a set of structured recursions, which makes it attractive for data-path implementations.

2. FINITE-LENGTH DFE FORMULATION

Consider the discrete-time DFE-channel equalization model depicted in Fig. 1. For simplicity of presentation, we shall assume a

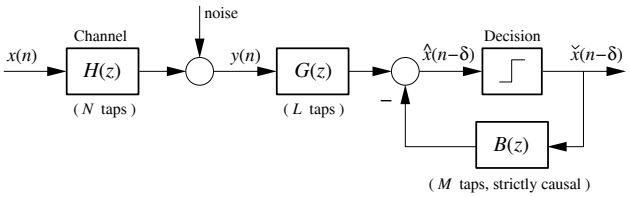

Fig. 1. Discrete symbol-spaced DFE model.

symbol-spaced model for the equalizer. Extension to the fractionally-spaced case is straightforward. In addition, we shall make the following commonly used assumptions (see, e.g., [1]):

- The input sequence $\{x(n)\}$ is complex, iid, with unit power.
- The additive noise $v(n)$ is Gaussian with autocorrelation matrix R_v.
- The decisions $\hat{x}(n-\delta)$ are assumed to be correct, and hence equal to $x(n-\delta)$.
- The feedforward filter $G(z)$ has length L. The number of taps M of the feedback filter $B(z)$ is assumed greater or equal to the channel memory, i.e., $M \geq N-1$.

Our goal is to minimize the mean squared error quantity

$$\xi = \mathsf{E}|x(n-\delta) - \hat{x}(n-\delta)|^2 \,, \qquad (1)$$

where $\hat{x}(n-\delta)$ is the delayed input signal estimate prior to the decision. By collecting the tap coefficients of $G(z)$ and $B(z)$ into

vectors, we can express the received signal $\hat{x}(n-\delta)$ as

$$\hat{x}(n-\delta) = y_n g - \check{x}_n b \ ,$$

where the received vector is given by

$$y_n = x_n H + v_n \ , \tag{2}$$

H is the $(N+L-1)\times L$ convolution matrix associated with the channel matrix coefficients $h(0), h(1), \ldots, h(N-1)$, given by

$$H = \begin{bmatrix} h(0) & 0 & \cdots & 0 \\ h(1) & h(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h(N-1) & h(N-2) & \ddots & h(0) \\ 0 & h(N-1) & \ddots & h(1) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h(N-1) \end{bmatrix} . \tag{3}$$

The row vector x_n is the $1\times(N+L-1)$ input regressor

$$x_n \triangleq \begin{bmatrix} x(n) & x(n-1) & \cdots & x(n-N-L+2) \end{bmatrix}$$

The row vector y_n is the $1\times L$ input regressor to the feedforward filter g, i.e.,

$$y_n \triangleq \begin{bmatrix} y(n) & y(n-1) & \cdots & y(n-L+1) \end{bmatrix}$$

Similarly, $\check{x}_n$ is the $1\times M$ input regressor to the (strictly causal) feedback filter b, i.e.,

$$\check{x}_n \triangleq \begin{bmatrix} x(n-\delta-1) & \cdots & x(n-\delta-M) \end{bmatrix}$$

Also, v_n is the $1\times L$ noise vector process.

2.1. Minimization with respect to $\{g, b\}$

By collecting g and b into a single vector w, the minimization of (1) can be written as

$$\min_w \mathsf{E}\left| x(n-\delta) \ - \ \underbrace{\begin{bmatrix} y_n & -\check{x}_n \end{bmatrix}}_{u} \underbrace{\begin{bmatrix} g \\ b \end{bmatrix}}_{w} \right|^2 .$$

Now, denoting R_u the variance of the augmented input regression vector u, and crossvariance $R_{ux(n-\delta)}$, the well known solution to this smoothing problem is given by [2]

$$w_{\text{opt}} = R_u^{-1} R_{ux(n-\delta)} \ , \tag{4}$$

where

$$R_u \triangleq \mathsf{E}\begin{bmatrix} y_n^* \\ -\check{x}_n^* \end{bmatrix}\begin{bmatrix} y_n & -\check{x}_n \end{bmatrix} = \begin{bmatrix} R_y & -R_{y\check{x}} \\ -R_{\check{x}y} & R_{\check{x}} \end{bmatrix} .$$

and

$$R_{\mathbf{u}x(n-\delta)} = \mathsf{E}\begin{bmatrix} y_n^* x(n-\delta) \\ \check{x}_n^* x(n-\delta) \end{bmatrix} = \begin{bmatrix} R_{yx(n-\delta)} \\ R_{\check{x}x(n-\delta)} \end{bmatrix}$$

Using the channel output model in (2), and the fact that $x(n)$ is iid, we can obtain the following closed form expressions for $\{R_y, R_{y\check{x}}, R_{\check{x}}, R_{yx(n-\delta)}, R_{\check{x}x(n-\delta)}\}$:

$$\begin{aligned} R_y &= \mathsf{E}y_n^* y_n = R_v + H^* H \ , \\ R_{y\check{x}} &= H^*(\mathsf{E}x_n^*\check{x}_n) = H^*\begin{bmatrix} 0_{(\delta+1)\times M} \\ I_M \\ 0_{(N+L-M-\delta)\times M} \end{bmatrix} \triangleq \bar{H}^* \\ R_{\check{x}} &= I_M \\ R_{yx(n-\delta)} &= H^*\mathsf{E}x_n^* x(n-\delta) = H^*\begin{bmatrix} 0_{1\times\delta} \\ I \\ 0 \end{bmatrix} \triangleq h^* \\ R_{\check{x}x(n-\delta)} &= 0 \end{aligned}$$

where $\bar{H}$ is a submatrix of H, such that

$$H = \begin{bmatrix} H_1 \\ \bar{H} \\ H_2 \end{bmatrix} \text{, where } H_1 \text{ is } (\delta+1)\times L$$

Now, with the quantities defined above, Eq. (4) becomes

$$w_{\text{opt}} = \begin{bmatrix} R_v + H^*H & -\bar{H}^* \\ -\bar{H} & I \end{bmatrix}^{-1}\begin{bmatrix} h^* \\ 0 \end{bmatrix} .$$

Using a well known inverse of block matrices, we can write

$$w_{\text{opt}} = \begin{bmatrix} I \\ \bar{H} \end{bmatrix}(R_v + H_1^* H_1 + H_2^* H_2)^{-1} h^* \ .$$

Note further that because we have assumed $M \geq N-1$, the quantities h_δ, H_1 and H_2 are such that

$$\left[\frac{H_1}{H_2}\right] = \left[\frac{H_\delta \quad 0}{0 \quad \tilde{H}}\right] \quad \text{and} \quad h = \begin{bmatrix} h_\delta & 0 \end{bmatrix} .$$

This implies the following expressions for the optimal feedback and feedforward coefficients:

$$\boxed{g_{\text{opt}} = (R_v + H_\delta^* H_\delta)^{-1} h_\delta^*} \tag{5}$$

$$\boxed{b_{\text{opt}} = \bar{H} g_{\text{opt}}} \tag{6}$$

Here, we want to stress that the above expressions are valid for all values of the delay δ. In general, the optimal value for the decision delay δ is within the range $L-1 \leq \delta_{\text{opt}} \leq N+L-2$.

2.2. Prior Art

An alternative approach for optimizing (1) which has been used in [1] and the references there in, is to define an extended vector $[x(n-\delta) \quad \check{x}_n]$, and minimize (1) with respect to g and the augmented coefficient vector $b' \triangleq \begin{bmatrix} 1 \\ b \end{bmatrix}$ such that the minimization of (1) becomes

$$\min_w \mathsf{E}\left|[x(n-\delta) \quad \check{x}_n]b' \ - \ y_n g\right|^2 \tag{7}$$

with the constraint that the first matrix coefficient of b' is equal to 1. When $M = N-1$ and $\delta = L-1$, this approach results in the following expressions for $\{g, b\}$ (see [1]):

$$g_{\text{opt}} = (R_v + H^*H)^{-1} H^* \begin{bmatrix} 0 \\ b'_{\text{opt}} \end{bmatrix} \tag{8}$$

where $\begin{bmatrix} 0 \\ b'_{\text{opt}} \end{bmatrix}$ is obtained from the Cholesky factorization

$$(I + HR_v^{-1}H^*) = \mathcal{L}D\mathcal{L}^* \ . \tag{9}$$

We call the attention of the reader for a few points:

- The current most efficient procedure for finding the optimal DFE coefficients according to (8) and (9) involves two steps: (i) Performing the Cholesky factorization of (9) using the Generalized Schur algorithm; then (ii) computing the feedforward filter via the Levinson's recursion or the *back-substitution* method.
- Equations (8) and (9) are equivalent to (5) and (6) due to the uniqueness of w_{opt} when (1) is minimized. Note, however, that (5) and (6) represent much more compact expressions [for example, b_{opt} is obtained simply by a convolution operation in (6)].

Next, we show that the expressions obtained in (5) and (6) provide alternative methods for computing g_{opt} and b_{opt} in a simpler and more efficient way.

3. FAST COMPUTATION OF g_{opt}

Let us define the coefficient matrix

$$P_\delta \triangleq (R_v + H_\delta^* H_\delta)^{-1} .$$

The optimal solution for the feedforward coefficients is then given by

$$g_{\text{opt}} = P_\delta h_\delta^* .$$

We can readily recognize that the quantity $g_{\text{opt}} = P_\delta h_\delta^*$ corresponds to the definition of the Kalman gain vector, used to update the optimal weights in a regularized RLS problem. More specifically, given an $(n+1) \times L$ data matrix H_n and the corresponding coefficient matrix P_n, the Kalman gain $g_n = P_n h_n^*$ can be time-updated according to the following recursions (see, e.g., [2]):

$$\gamma^{-1}(n) = 1 + h_n P_{n-1} h_n^* , \quad (10)$$

$$g_n = P_{n-1} h_n^* \gamma(n) , \quad (11)$$

$$P_n = P_{n-1} - g_n \gamma^{-1}(n) g_n^* , \quad (12)$$

where $P_{-1} = R_v^{-1}$ and $g_0 = 0$.

On the other hand, the computation of g_n can be done efficiently, via a fast RLS recursion. For example, consider Eq. (11), which can be written as

$$g_n = k_n \gamma(n) ,$$

where $\gamma(n)$ is defined in (10). The quantity $k_n = P_{n-1} h_n^*$ is referred to as the *normalized* Kalman gain matrix in adaptive RLS filtering. One way of achieving a fast recursion is to propagate k_n efficiently, by successive order updates and order downdates of k_n, by means of forward and backward LS prediction problems. The well known fast transversal filters (FTF) [3] is an example where such fast technique is encountered, when $R_v = \sigma_v^2 I$. Note that here, because $M \geq N-1$, the matrix H_δ has a lower triangular structure and the desired signal for the backward prediction problem is always equal to zero. As a result, the least-squares backward prediction solution up to time δ will be equal zero, and the fast algorithm will need to rely only on the forward prediction part. That is, perform the order-update

$$k_{L,n-1} \longrightarrow k_{L+1,n-1}$$

so that

$$k_{L,n} = k_{L+1,n-1}(1:L)$$

Table 1 lists the resulting fast algorithm.[1]

[1]It is well known that the exponentially weighted FTF algorithm can become unstable in finite precision implementation. Hence, one might wonder whether this is also the case for the simplified algorithm, since both have the same essence. There are three advantages, however, that prevent the simplified algorithm from becoming unstable:

(i) The main source of error propagation in the full FTF algorithm arises in the backward prediction section of its recursions [4]. Here, by ruling out the equations associated with the backward prediction problem we are automatically eliminating many of the recursive loops that contribute to the unstable behavior of the full FTF algorithm.

(ii) Another source of instability of the FTF recursions is related to the forgetting factor λ which appears in the exponentially weighted RLS problem. Theoretically with $\lambda < 1$, the redundant components generated by numerical errors would decay to zero as $N \to \infty$. However, an averaging analysis [5] shows that this will lead to unstable modes at $1/\lambda$ which will cause instability in finite precision. Now, note that the above fast recursions deal with the problem of filtering a finite set of data samples of the channel model. In other words, for our purpose, the algorithm must stop when $n = \delta$. Moreover, here the corresponding forgetting factor is always equal to one, in which case the recursions will present much better numerical behavior. This means that even if the simplified fast algorithm were to become unstable, it would not be likely to happen within the first δ iterations. Furthermore, if this was still the case, a simple increase of wordlength would overcome the problem. We have performed extensive simulations under finite precision implementations and observed no sign of instability.

Initialization
$\zeta^f(-1) = \sigma_v^2$ $w_{-1}^f = k_{L,0} = 0$ $\gamma(0) = 1$
For $n = 0$ to δ, repeat: $\alpha_L(n-1) = h(n) - h_{n-1} w_{n-2}^f$ $f(n-1) = \gamma(n-1)\alpha(n-1)$ $k_{L+1,n-1} = \begin{bmatrix} 0 \\ k_{L,n-1} \end{bmatrix} + \frac{f^*(n-1)}{\sigma_v^2} \begin{bmatrix} 1 \\ -w_{n-2}^f \end{bmatrix}$ $\zeta^f(n-1) = \zeta^f(n-2) + \alpha^*(n-1) f(n-1)$ $w_{n-1}^f = w_{n-2}^f + k_{L,n-1} f(n-1)$ $\gamma(n) = \frac{\sigma_v^2}{\zeta^f(n-1)}$ $k_{L,n} = k_{L+1,n-1}(1:L)$
Set $g_{\text{opt}} = k_{L,\delta}\gamma(\delta)$

Table 1. FTF Computation of g_{opt} for the case of white noise.

3.1. Fast Array Computation of g_{opt}

The FTF algorithm propagates explicitly the quantities k_n and $\gamma(n)$, so that at the end of the recursions we can obtain the optimal feedforward filter coefficients as $g_{\text{opt}} = k_\delta \gamma(\delta)$.

On the other hand, its fast array counterpart (see, e.g., [2]), is one that propagates the *square-root normalized* quantities $\tilde{k}_n$ and $\gamma^{-1/2}(n)$, so that the optimal feedforward filter coefficients in the end of the array recursions are computed as $g_{\text{opt}} = \tilde{k}_\delta \gamma^{1/2}(\delta)$. Hence, the fast recursions in Table 1 can be equivalently written in array form, which is listed in Table 2.

Here, Θ_n is a unitary matrix that produces the zero entry in the above post-array, and is computed via a stable circular rotation:

$$\Theta = \frac{1}{\sqrt{1+|\rho|^2}} \begin{bmatrix} 1 & \rho \\ \rho^* & -1 \end{bmatrix} , \quad \text{where} \quad \rho = \frac{[h(n) \quad h_{n-1}] t_n}{\gamma^{-1/2}(n)} .$$

Initialization

$\tilde{k}_0 = 0$

$\gamma^{-1/2}(0) = 1$

$t_0 = \frac{1}{\sigma_v} \begin{bmatrix} 1 & 0_L \\ 0_L & 1 \end{bmatrix}$

For $n = 0$ to δ, repeat:

$$\begin{bmatrix} \gamma^{-\frac{1}{2}}(n) & [h(n) & h_{n-1}]t_n \\ \begin{bmatrix} 0 \\ \tilde{k}_n \end{bmatrix} & & t_n \end{bmatrix} \Theta_n = \begin{bmatrix} \gamma^{-\frac{1}{2}}(n+1) & 0 \\ \begin{bmatrix} \tilde{k}_{n+1} \\ 0 \end{bmatrix} & t_{n+1} \end{bmatrix}$$

Set $g_{\text{opt}} = \tilde{k}_\delta \gamma^{1/2}(\delta)$

Table 2. Fast Array Computation of the feedforward coefficients.

4. FAST COMPUTATION OF FEEDBACK EQUALIZER

The filter convolution that defines the optimal feedback coefficients in Eq. (6) can be computed directly with $LM/2$ multiplications. Alternatively, this can be achieved efficiently via well known fast FFT convolution techniques, by extending the Toeplitz structure of $\bar{H}$ to form a $K \times K$ circulant matrix $C = F^* \Lambda F$, where F the $K \times K$ DFT matrix (K is the smallest power-of-two integer larger than or equal to $(M + L)$). The matrix Λ is diagonal and contains the elements of the DFT of the first column of C. The solution for b_{opt} then becomes

$$b_{\text{opt}} = [I_M \quad 0] F^* \Lambda F \begin{bmatrix} g_{\text{opt}} \\ 0 \end{bmatrix} .$$

The overall complexity simply amounts to $2M + 6M \log_2(2M)$.

5. ADVANTAGES OVER KNOWN PRIOR ART

The previous discussions show that the proposed method has the following differences from the known prior art:

5.1. Lower Computational Complexity

Note that the Cholesky factorization of the prior art in (9) involves the full convolution matrix H defined in (3), which has dimensions $(N + L - 1) \times L$. In this case, the computational complexity required for the generalized Schur algorithm is $\mathcal{O}(NL)$, that is, it depends on the channel length. The total number of complex multiplies of the method in [1] is given there in by

$$6LM + 12L - 4M - 8 + L^2$$

Now, the fast algorithm proposed for the computation of g_{opt} requires $\mathcal{O}(\delta L)$ operations, that is, it is not dependent on the channel length, but on the decision delay δ. This will make considerable difference for long channels, and for small values of δ. The total complex multiplies of the algorithm in Table 1 is given by

$$3(\delta + 1)(L + 1)/2 + \min\{2M + 6M \log_2(2M), LM/2\}$$

Figure 2 shows the reduction in complexity versus L of the proposed method with respect to the method in [1], for $M = 64$ and $\delta = L - 1$. The figure show that the proposed method can reduce from 85 to 95% of the overall tap computation complexity for L within 64 coefficients.

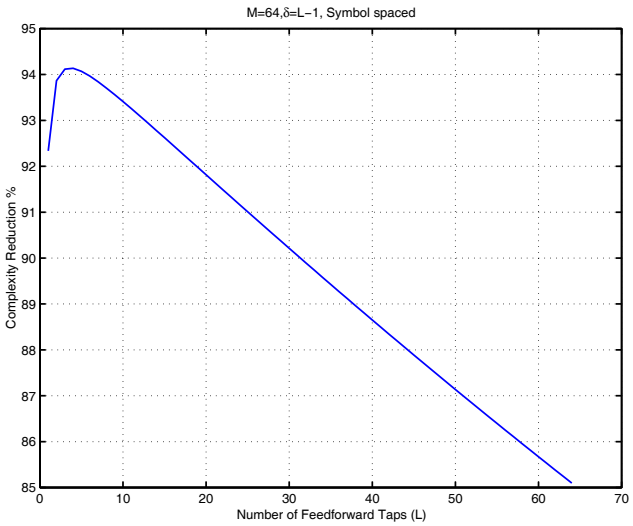

Fig. 2. Complexity reduction compared to the method in [1].

5.2. More Stability in Finite Precision

The feedback b_{opt} is easily obtained via stable fast convolution methods without resorting to any recursive algorithm. This is in contrast to the computation of the feedforward filter in the prior art via Levinson's or the back-substitution algorithm, which may not be reliable under finite precision implementations.

5.3. Simpler Structured Recursions

For FFE tap computations, the proposed method relies on iterating the recursions given in Tables 1 or 2. Note that the updates in Table 1 are very similar to the operations needed to iterate the widely used LMS algorithm, which is well known for its simplicity. This is unlike the prior art which requires iterating non-structured equations which might be more difficult to implement. In addition, the computation of g_{opt} in Table 2 can be efficiently implemented via CORDIC processors.

6. CONCLUSION

We proposed new fast techniques for computing MMSE-DFE coefficients. In comparison with the current known methods, the proposed method is less complex, uses more structured equations, and is more stable in finite precision. The approach of this paper easily accommodates fractionally-spaced models and can be further extended to the case of colored noise.

7. REFERENCES

[1] N. Al-Dhahir and J. M. Cioffi, "Fast Computation of Channel-Estimate Based Equalizers in Packet Data Transmission," *IEEE Trans. on Signal Processing*, vol. 43, no. 11, pp. 2462–2473, Nov. 1995.

[2] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*, Prentice Hall, NJ, 2000.

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

[4] P. A. Regalia, "Numerical stability issues in fast least-squares adaptation algorithms,"*Optical Egineering*, vol. 31, pp. 1144–52, Jun. 1992.

[5] 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.