FRLS COMPLEX DECISION FEEDBACK ADAPTIVE CHANNEL EQUALIZER USING REAL VALUED ARITHMETIC

George-Othon Glentis

TEI of Crete, Branch at Chania
Department of Electronics 3, Romanou Str, Halepa, 73133 Chania, GREECE
Email: gglentis@chania.teicrete.gr

ABSTRACT

This paper considers the implementation of a Fast Recursive Least Squares, complex adaptive Decision Feedback channel equalizer, using real valued arithmetic. An efficient adaptive scheme is established based on novel three-terms recursions for the time updating of the pertinent parameters, resulting in significant computational savings. The proposed algorithm is implemented using real valued arithmetic only, whilst reducing the number of the required real valued multiplication operations by 23%, at the expense of a marginal 1.5% increase in the number of the real valued additions.

1. INTRODUCTION

The design of efficient adaptive Decision Feedback equalizers has been the subject of major research and development, for high-speed digital communication over satellite, microwave, mobile or unshielded twisted pair channels, for DTV receivers etc., [4]. Complex valued signals are encountered in most of of adaptive equalization applications. The *fast* complex valued multiplication method, has been adopted for the design of efficient digital signal processing algorithms, including digital filtering and adaptive filtering. In the context of VLSI signal processing, the *fast* complex valued multiplication method has been introduced as an Algorithmic Strength Reduction (SR) transform, and it has been successfully applied for the design of low power, high speed adaptive filters and equalizers, [1],[3],[5].

The output of symbol spaced Decision Feedback adaptive equalizer is defined as, [4],

$$\begin{aligned} y(n) &= \mathbf{b}_p^H(n-1)\mathbf{u}_p(n) + \mathbf{a}_q^H\mathbf{I}_q(n-1) & (1)\\ I(n) &= \mathcal{D}[y(n)] & (2) \end{aligned}$$

Here, $u(n)$ is a complex valued input signal to the adaptive equalizer and $I(n)$ is the output of the decision device. Vectors $\mathbf{b}_p$ and $\mathbf{a}_q$ carry the coefficients of the feedforward and the feedback part of the DE, respectively. Vectors $\mathbf{u}_p(n)$ and $\mathbf{I}_q(n-1)$ carry the input data associated to the linear regression (subscripts p and q denote the vector size). Eq. (1) is written compactly as

$$y(n) = \mathbf{c}_m^H\mathbf{x}_m(n) \qquad (3)$$

$$\mathbf{c}_m = [\mathbf{b}_p^T \ \mathbf{a}_q^T]^T \qquad \mathbf{x}_m(n) = [\mathbf{u}_p^T(n) \ \mathbf{I}_q^T(n-1)]^T \quad (4)$$

Vectors $\mathbf{c}_m$ and $\mathbf{x}_m(n)$ have dimensions $m \times 1$, where $m = p + q$. The DF equalizer is designed as

$$\mathbf{c}_m(n): \quad \min_{\mathbf{c}_m} \left(\sum_{k=0}^{n} \lambda^{n-k} {e_m^c}^*(k) e_m^c(k) \right) \qquad (5)$$

$$e_m^c(k) = z(k) - \mathbf{c}_m^H(n)\mathbf{x}_m(k). \qquad (6)$$

Here, $z(k)$ denotes the desired response signal. During the training period, $z(k)$ equals to a delayed version of a known transmitted signal, $\delta(n)$, i.e., $z(k) = \delta(n-D)$, while during the decision directed mode, $z(k)$ is set equal to the detected symbol, i.e., $z(k) = I(k)$. $0 << \lambda < 1$ is a parameter that controls the size of the exponentially fading memory of the adaptation mechanism.

The RLS algorithm provides an adaptive method for the solution of the normal equations, resulting by minimization of (6). Indeed,

$$\begin{aligned} \epsilon_m^c(n) &= (z(n) - \mathbf{c}_m^H(n-1)\mathbf{x}_m(n))\,/\alpha_m(n), & (7)\\ \alpha_m(n) &= 1 + \mathbf{w}_m^H(n)\mathbf{x}_m(n), & (8)\\ \mathbf{c}_m(n) &= \mathbf{c}_m(n-1) + \mathbf{w}_m(n)\epsilon_m^{c*}(n). & (9) \end{aligned}$$

Parameter $\mathbf{w}_m(n)$ that appears in the above equations, is the so called Kalman gain vector, defined as, [2],[7],

$$\mathbf{w}_m(n) = \lambda\mathbf{R}_m^{-1}(n-1)\mathbf{x}_m(n). \qquad (10)$$

According to the RLS algorithm, the inverse matrix $\mathbf{R}_m^{-1}(n-1)$ can be updated using an $O(m^2)$ recursive scheme. Fast RLS algorithms go a step further, using fast $O(m)$ adaptive schemes for the time update of the Kalman gain vector, bypassing the need of the inverse matrix adaptation. The key point in the development of fast RLS algorithms is the proper utilization of the shift invariance property of the data matrix, as well as the use of a set of permutation matrices for the reorganization of the block data vectors, [2],[6],[7].

Direct application of a fast complex valued multiplication method to the original FRLS scheme, where each individual complex multiplication is performed using a fast scheme, can reduce the number of the required real multiplications by 23%, at the expense of a 70% increase in the required number of additions. To keep the overhead due to the extra additions low, both the input and the desired response signals, as well as all variables associated with the FRLS algorithm, are treated as pairs of real signals, and operations are re-organized on a real arithmetic basis. Auxiliary signals and filter parameters are introduced, which correspond to the generic signal transformation $s_{\Re}(n) \pm s_{\Im}(n)$, where $s_{\Re}(n)$ and $s_{\Im}(n)$ represent the real and the imaginary part of the complex valued signal $s(n)$, respectively. The algorithmic strength reduction technique is subsequently applied to the transformed FRLS algorithm. In this way, novel real valued three-terms recursions are derived for the efficient time update of the FRLS parameters, resulting in significant computational savings.

2. THE DF SR FRLS ADAPTIVE EQUALIZER

Complex valued arithmetic is required for implementation of the DF FRLS algorithm when the input signals and/or the filter model are represented by complex valued variables. Complex valued addition is realized by a set of two real valued additions, while complex valued multiplication can be realized by the *classic* method, that requires four real valued multiplications and two real valued additions. Alternatively, the *fast* complex valued multiplication method can be applied, where the inherent dependencies of the partial products and sums, are utilized for the reduction of the number of real valued multiplication operations, at the expense of some extra real valued additions, [5]. In this case, three real valued multiplications and five real valued additions, are required. Among others, two possible implementations of a *fast* complex valued multiplication of two complex valued numbers $a + \jmath b$ and $c + \jmath d$, are described by

$$(b(c-d) + (a-b)c) + \jmath\,(a(c+d) - (a-b)c) \quad (11)$$

$$(-(a+b)d + a(c+d)) + \jmath\,(b(c-d) + (a+b)d)\,. \quad (12)$$

The complex regressor vector is written as $\mathbf{x}_m(n) = \mathbf{x}_m^{\Re}(n) + \jmath\mathbf{x}_m^{\Im}(n)$. In a similar way, the filter coefficients complex vector is expressed as $\mathbf{c}_m(n) = \mathbf{c}_m^{\Re}(n) + \jmath\mathbf{c}_m^{\Im}(n)$. Then, $y(n)$ is expanded as

$$\begin{aligned} y(n) \;&=\; \mathbf{c}_m^{\Re,T}(n-1)\mathbf{x}_m^{\Re}(n) + \mathbf{c}_m^{\Im,T}(n-1)\mathbf{x}_m^{\Im}(n) \\ &+\; \jmath\left(\mathbf{c}_m^{\Re,T}(n-1)\mathbf{x}_m^{\Im}(n) - \mathbf{c}_m^{\Im,T}(n-1)\mathbf{x}_m^{\Re}(n)\right). \end{aligned}$$

Let us consider a set of auxiliary filter output signals

$$\widehat{y}(n) = y_{\Re}(n) + y_{\Im}(n), \quad \tilde{y}(n) = y_{\Re}(n) - y_{\Im}(n). \quad (13)$$

Variables $\widehat{y}(n)$ and $\tilde{y}(n)$ are estimated as

$$\widehat{y}(n) \;=\; \tilde{\mathbf{c}}_m^T(n-1)\mathbf{x}_m^{\Re}(n) + \widehat{\mathbf{c}}_m^T(n-1)\mathbf{x}_m^{\Im}(n) \quad (14)$$

$$\tilde{y}(n) \;=\; \widehat{\mathbf{c}}_m^T(n-1)\mathbf{x}_m^{\Re}(n) - \tilde{\mathbf{c}}_m^T(n-1)\mathbf{x}_m^{\Im}(n) \quad (15)$$

Here, vectors $\widehat{\mathbf{c}}_m(n)$ and $\tilde{\mathbf{c}}_m(n)$ are transformed versions of the original filter coefficient vector $\mathbf{c}_m(n)$, defined as

$$\widehat{\mathbf{c}}_m(n) = \mathbf{c}_m^{\Re}(n) + \mathbf{c}_m^{\Im}(n), \tilde{\mathbf{c}}_m(n) = \mathbf{c}_m^{\Re}(n) - \mathbf{c}_m^{\Im}(n). \quad (16)$$

Consider the complex variable $Y(n) = \tilde{y}(n) + \jmath\widehat{y}(n)$. It is computed using eqs. (14) and (15) as

$$Y(n) = \left(\widehat{\mathbf{c}}_m(n-1) + \jmath\tilde{\mathbf{c}}_m(n-1)\right)^T \left(\mathbf{x}_m^{\Re}(n) + \jmath\mathbf{x}_m^{\Im}(n)\right).$$

Application of the SR transform, eq. (11), to the above equation, results in

$$\widehat{y}(n) = \widehat{\mathbf{c}}_m^T(n-1)\widehat{\mathbf{x}}_m(n) - \Delta\mathbf{C}_m(n)^T\mathbf{x}_m^{\Re}(n), \quad (17)$$

$$\tilde{y}(n) = \tilde{\mathbf{c}}_m^T(n-1)\tilde{\mathbf{x}}_m(n) + \Delta\mathbf{C}_m(n)^T\mathbf{x}_m^{\Re}(n). \quad (18)$$

where $\Delta\mathbf{C}_m(n) \equiv (\widehat{\mathbf{c}}_m(n) - \tilde{\mathbf{c}}_m(n))$. The auxiliary regressors $\widehat{\mathbf{x}}_m(n)$ and $\tilde{\mathbf{x}}_m(n)$ appeared above, are compatible defined as

$$\widehat{\mathbf{x}}_m(n) = \mathbf{x}_m^{\Re}(n) + \mathbf{x}_m^{\Im}(n), \tilde{\mathbf{x}}_m(n) = \mathbf{x}_m^{\Re}(n) - \mathbf{x}_m^{\Im}(n).$$

Eqs. (18) and (18) provide a cost effective way of computing variables $\widehat{y}(n)$ and $\tilde{y}(n)$, using $3m$ real valued multiplications and $4m-1$ real valued additions. On the contrary, evaluation of either $\widehat{y}(n)$ and $\tilde{y}(n)$ using eqs. (14) and (15), or $y(n)$ using eq. (13), requires $4m$ real valued multiplications and $4m-2$ real valued additions. Thus, working with the transformed variables $\widehat{y}(n)$ and $\tilde{y}(n)$ instead of the original parameter $y(n)$, results in a significant reduction in the required number of real valued multiplications. If the SR technique had been applied on an individual basis, for each complex valued multiplication involved into the inner product computation $y(n) = \mathbf{c}_m^H(n-1)\mathbf{x}_m(n)$, $3m$ real valued multiplications and $7m$ real valued additions would have been utilized.

Since $\widehat{y}(n)$ and $\tilde{y}(n)$ are estimated using the transformed filters $\widehat{\mathbf{c}}_m(n)$ and $\tilde{\mathbf{c}}_m(n)$, it could be more efficient, if these variables are updated directly, instead of the updating the original $\mathbf{c}_m(n)$. Recall that

$$\begin{aligned} \mathbf{c}_m^{\Re}(n) &= \mathbf{c}_m^{\Re}(n-1) + \mathbf{w}_m^{\Re}(n)\epsilon_m^{c,\Re}(n) + \mathbf{w}_m^{\Im}(n)\epsilon_m^{c,\Im}(n), \\ \mathbf{c}_m^{\Im}(n) &= \mathbf{c}_m^{\Im}(n-1) - \mathbf{w}_m^{\Re}(n)\epsilon_m^{c,\Im}(n) + \mathbf{w}_m^{\Im}(n)\epsilon_m^{c,\Re}(n). \end{aligned}$$

If we insert the above recursions into eq. (16) we get

$$\begin{aligned} \widehat{\mathbf{c}}_m(n) &= \widehat{\mathbf{c}}_m(n-1) + \left(\widehat{\mathbf{w}}_m(n)\epsilon_m^{c,\Re}(n) - \tilde{\mathbf{w}}_m(n)\epsilon_m^{c,\Im}(n)\right), \\ \tilde{\mathbf{c}}_m(n) &= \tilde{\mathbf{c}}_m(n-1) + \left(\tilde{\mathbf{w}}_m(n)\epsilon_m^{c,\Re}(n) + \widehat{\mathbf{w}}_m(n)\epsilon_m^{c,\Im}(n)\right). \end{aligned}$$

The transformed Kalman gain vectors are defined as

$$\widehat{\mathbf{w}}_m(n) = \mathbf{w}_m^{\Re}(n) + \mathbf{w}_m^{\Im}(n), \tilde{\mathbf{w}}_m(n) = \mathbf{w}_m^{\Re}(n) - \mathbf{w}_m^{\Im}(n).$$

Consider the complex valued transformed coefficients vector defined as $\mathbf{C}_m(n) = \widehat{\mathbf{c}}_m(n) - \jmath\tilde{\mathbf{c}}_m(n)$, which is updated as

$$\begin{gathered} \mathbf{C}_m(n) = \mathbf{C}_m(n-1) + \\ \left(\widehat{\mathbf{w}}_m(n) - \jmath\tilde{\mathbf{w}}_m(n)\right)\left(\epsilon_m^{c,\Re}(n) - \jmath\epsilon_m^{c,\Im}(n)\right). \end{gathered}$$

Application of the algorithmic strength reduction transform, eq. (12), to the above equation, results in

$$\begin{aligned} \widehat{\mathbf{c}}_m(n) &= \widehat{\mathbf{c}}_m(n-1) + \widehat{\mathbf{w}}_m(n)\tilde{\epsilon}_m^c(n) + \Delta\mathbf{W}_m(n)\epsilon_m^{c,\Im}(n), \\ \tilde{\mathbf{c}}_m(n) &= \tilde{\mathbf{c}}_m(n-1) + \tilde{\mathbf{w}}_m(n)\widehat{\epsilon}_m^c(n) + \Delta\mathbf{W}_m(n)\epsilon_m^{c,\Im}(n). \end{aligned}$$

where $\Delta\mathbf{W}_m(n) \equiv \widehat{\mathbf{w}}_m(n) - \tilde{\mathbf{w}}_m(n)$. The transformed filtering error variables has as $\widehat{e}_m^c(n) = \widehat{z}(n) + \widehat{y}(n)$, $\tilde{e}_m^c(n) = \tilde{z}(n) - \tilde{y}(n)$ where $\widehat{z}(n) = z_{\Re}(n) + z_{\Im}(n)$, amd $\tilde{z}(n) = z_{\Re}(n) - z_{\Im}(n)$.

The computational complexity for estimating $\widehat{\mathbf{c}}_m(n)$ and $\tilde{\mathbf{c}}_m(n)$ is $3M$ real valued multiplications and $5M$ real valued additions. This figure should be compared with the computational complexity of the original adaptive scheme implied by eq. (9) which is either $4m$ real valued multiplications and $4m$ real valued additions, or $3m$ real valued multiplications and $7m$ real valued additions, depending on the type of complex valued multiplication (i.e., classical, or fast elementwise) is adopted.

The proposed SR FRLS adaptive decision feedback equalization algorithm is summarized in Table 1. The SR transform has been applied to the forward and the backward prediction parameters, that are involved in the computational flow of the FRLS algorithm. Indeed, fast, three term recursions have been derived for the efficient updating of the transformed forward and backward prediction parameters. Efficient three term recursions have also be derived for the updating of the transformed Kalman gain vectors $\hat{\mathbf{w}}_m(n)$ and $\tilde{\mathbf{w}}_m(n)$. Feedback stabilization has been utilized in order to prevail numerical divergence, [6].

The computational complexity of the original complex valued FRLS adaptive DF equalizer, measured in terms of real valued multiplications (MULs) and additions (ADDs), is $60m$ MULs and $60m$ ADDs, where $m = p + q$. When fast multiplication is applied directly at a complex multiplier level, the corresponding complexity reduces to $46m$ MULs and $102m$ ADDs. On the contrary, the proposed efficient SR FLRS DF adaptive equalizer requires $46m$ MULs and $61m$ ADDs. Clearly, the proposed SR FRLS algorithm provide the best low complexity compromise among all the competitive schemes. The proposed scheme is suitable for multiprocessor or VLSI ASIC implementation, achieving reduction of the required DSP units, or in the power dissipation and the silicon area of the fabricated circuit.

3. SIMULATION RESULTS

A QPSK signalling is considered and the ISI is simulated by passing the transmitted signal through a discrete-time linear channel. At the receiver, a DF symbol spaced equalizer is engaged in order to cancel the ISI introduced by the channel, where $p = 9$ and $q = 5$. The channel impulse response is, [4], $\mathbf{h}^2 = [0.227e^{-\jmath\phi_0} + 0.406e^{-\jmath\phi_1} + 0.688e^{-\jmath\phi_2} + 0.406e^{-\jmath\phi_3} + 0.227e^{-\jmath\phi_4}]^T$, where the ϕ_i are random phases. The SNR is set equal to 10 db. The stabilized SR FRLS adaptive algorithm is employed for the channel equalization ($\lambda = 0.99$). Feedback stabilization is employed, where the stabilization parameters are all set equal to $\sigma = 0.5$. The DF equalizer is initially trained using a known data sequence, consisting of 100 data samples. After the training period, it is set to a decision-directed mode. The unequalized signal shown in Fig. 1a, while the equalized signal ($y(n)$) is shown in Fig. 1b. The learning curve is depicted in Fig. 1c. Finally, error feedback signals that are used for algorithm stabilization are shown in Fig. 2.

4. CONCLUSIONS

A novel implementation of the FRLS DF adaptive equalizer has been presented. The proposed method results in a reduced computational complexity adaptive scheme, where all complex valued operations are replaced by real valued counterparts. The proposed algorithm is implemented using real valued arithmetic only, whilst reducing the number of the required real valued multiplications by 23%, at the expense of a marginal 1.5% increase in the number of the real valued additions.

REFERENCES

[1] R.Baghaie, and T.Laakso, Implementation of low-power CDMA RAKE receivers using strength reduction transformation, Proc. IEEE Nordic Sig. Proc Symposium (1998) Denmark, pp. 169-172.

[2] G.Glentis, and N.Kalouptsidis, Fast Adaptive Algorithms for Multichannel Filtering and System Identification, IEEE Trans. Sig. Proc. 40(10) (October 1992) 2433- 2458.

[3] R.Perry, D.Bull, and A.Nix, Efficient adaptive complex filtering algorithm with application to channel equalisation, IEE Proc.-Vis. Image Sig. Proc., 146(2) (February 1999) 57-64.

[4] J.Proakis, Digital Communications, 3rd Ed. McGraw-Hill, 1995.

[5] N.Shanbhag, Algorithmic transformation techniques for low-power wireless VLSI systems design, Int. Jounral of Wireless Information Networks, 5(2), (1998) 147-171.

[6] D.Slock, and T.Kailath, Numerical Stable Fast RLS transversal adaptive filtering, IEEE Trans. ASSP, 39(1) (January 1991) 92- 114.

[7] D.Slock, L.Chisci, H.Lev-Ari, and T.Kailath, Modular and numerically stable fast transversal filters for multichannel and multiexperiment RLS, IEEE Trans. Sig. Proc., 40(4) (April 1992) 784-802.

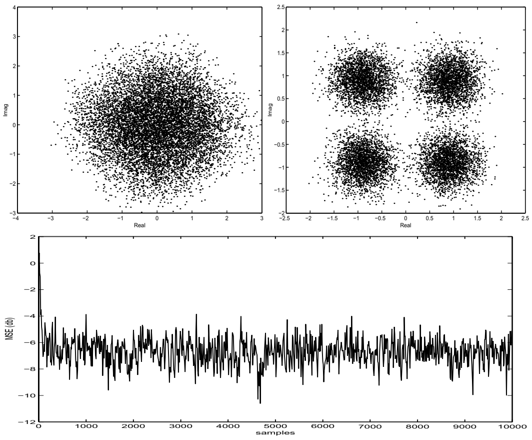

Figure 1. a) Unequalized data. b) Equalized data. c) Learning curve

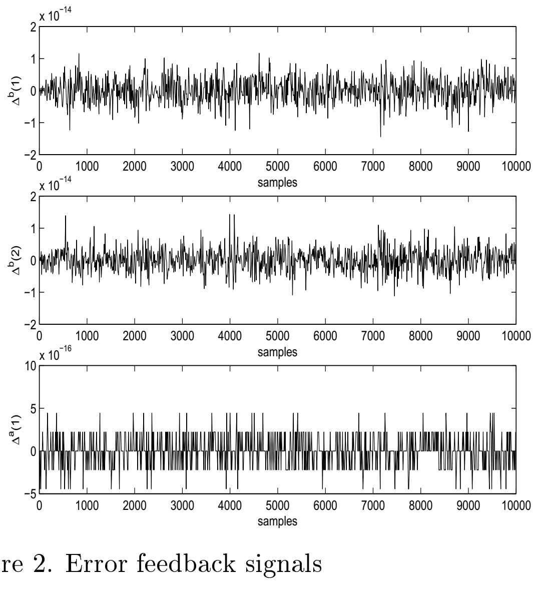

Figure 2. Error feedback signals

Data Preparation
$\hat{u}(n) = u_{\Re}(n) + u_{\Im}(n), \tilde{u}(n) = u_{\Re}(n) - u_{\Im}(n)$ $\hat{I}(n) = I_{\Re}(n) + I_{\Im}(n), \tilde{I}(n) = I_{\Re}(n) - I_{\Im}(n)$ $\hat{\mathbf{x}}_m(n) = [\hat{\mathbf{u}}_p^T(n) \;\; \hat{\mathbf{I}}_q^T(n-1)]^T$ $\tilde{\mathbf{x}}_m(n) = [\tilde{\mathbf{u}}_p^T(n) \;\; \tilde{\mathbf{I}}_q^T(n-1)]^T$ $\hat{\mathbf{x}}_m^{\circ}(n-1) = [\hat{\mathbf{u}}_p^T(n-1) \;\; \hat{\mathbf{I}}_q^T(n-1)]^T$ $\tilde{\mathbf{x}}_m^{\circ}(n-1) = [\tilde{\mathbf{u}}_p^T(n-1) \;\; \tilde{\mathbf{I}}_q^T(n-1)]^T$ $\hat{\mathbf{x}}_m^{f(1)}(n-1) = \hat{\mathbf{x}}_m^{\circ}(n-1), \tilde{\mathbf{x}}_m^{f(1)}(n-1) = \tilde{\mathbf{x}}_m^{\circ}(n-1)$ $\hat{\mathbf{x}}_m^{f(2)}(n-1) = \hat{\mathbf{x}}_m(n-1), \tilde{\mathbf{x}}_m^{f(2)}(n-1) = \tilde{\mathbf{x}}_m(n-1)$ $\hat{\mathbf{x}}_m^{b(1)}(n) = \hat{\mathbf{x}}_m(n), \tilde{\mathbf{x}}_m^{b(1)}(n) = \tilde{\mathbf{x}}_m(n)$ $\hat{\mathbf{x}}_m^{b(2)}(n) = \hat{\mathbf{x}}_m^{\circ}(n-1), \tilde{\mathbf{x}}_m^{b(2)}(n) = \tilde{\mathbf{x}}_m^{\circ}(n-1)$ $\hat{x}^{f(1)}(n) = \hat{u}(n), \tilde{x}^{f(1)}(n) = \tilde{u}(n)$ $\hat{x}^{f(2)}(n) = \hat{I}(n-1), \tilde{x}^{f(2)}(n) = \tilde{I}(n-1)$ $\hat{x}^{b(1)}(n) = \hat{u}(n-p), \tilde{x}^{b(1)}(n) = \tilde{u}(n-p)$ $\hat{x}^{b(2)}(n) = \hat{I}(n-1-q), \tilde{x}^{b(2)}(n) = \tilde{I}(n-1-q)$
Filtering Part
$y_1(n) = \widehat{\mathbf{c}}_m^T(n-1)\widehat{\mathbf{x}}_m(n), y_2(n) = \tilde{\mathbf{c}}_m^T(n-1)\tilde{\mathbf{x}}_m(n)$ $\Delta C_m(n-1) = \widehat{\mathbf{c}}_m(n-1) - \tilde{\mathbf{c}}_m(n-1)$ $y_3(n) = \Delta C_m^T(n-1)\mathbf{x}_m^{\Re}(n)$ $\widehat{e}_m^c(n) = \widehat{z}(n) + (y_1(n) - y_3(n))$ $\tilde{e}_m^c(n) = \tilde{z}(n) + (y_2(n) + y_3(n))$ $\widehat{\epsilon}_m^c(n) = \widehat{e}_m^c(n)/\alpha_m(n), \tilde{\epsilon}_m^c(n) = \tilde{e}_m^c(n)/\alpha_m(n)$ $\epsilon_m^{c,\Im}(n) = 0.5\left(\widehat{\epsilon}_m^c(n) - \tilde{\epsilon}_m^c(n)\right)$ $\mathbf{g}_m^{c,1}(n) = \widehat{\mathbf{w}}_m(n)\tilde{\epsilon}_m^c(n), \mathbf{g}_m^{c,2}(n) = \tilde{\mathbf{w}}_m(n)\widehat{\epsilon}_m^c(n)$ $\mathbf{g}_m^{c,3}(n) = \Delta W_m^T(n)\epsilon_m^{c,\Im}(n)$ $\widehat{\mathbf{c}}_m(n) = \widehat{\mathbf{c}}_m(n-1) + \mathbf{g}_m^{c,1}(n) + \mathbf{g}_m^{c,3}(n)$ $\tilde{\mathbf{c}}_m(n) = \tilde{\mathbf{c}}_m(n-1) + \mathbf{g}_m^{c,2}(n) + \mathbf{g}_m^{c,3}(n)$
Decision Part
$y(n) = 0.5\left((\hat{y}(n) + \tilde{y}(n)) + \jmath\left(\hat{y}(n) - \tilde{y}(n)\right)\right)$ $I(n) = \mathcal{D}[y(n)]$
FOR $i = 2$ **TO** 1 **DO**
Forward Prediction Part
$x_1^{fi}(n) = \widehat{\mathbf{a}}_m^T(n-1)\widehat{\mathbf{x}}_m^{fi}(n-1)$ $x_2^{fi}(n) = \tilde{\mathbf{a}}_m^T(n-1)\tilde{\mathbf{x}}_m^{fi}(n-1)$ $\Delta A_m^i(n-1) = \widehat{\mathbf{a}}_m^i(n-1) - \tilde{\mathbf{a}}_m^i(n-1)$ $x_3^{fi}(n) = \Delta A_m^{iT}(n-1)\mathbf{x}_m^{\Re,fi}(n-1)$ $\widehat{e}_m^{fi}(n) = \widehat{x}^{fi}(n) + \left(x_1^{fi}(n) - x_3^{fi}(n)\right)$ $\tilde{e}_m^{fi}(n) = \tilde{x}^{fi}(n) + \left(x_2^{fi}(n) + x_3^{fi}(n)\right)$ $\widehat{\epsilon}_m^{fi}(n) = \widehat{e}_m^{fi}(n)/\alpha_m(n-1), \tilde{\epsilon}_m^{fi}(n) = \tilde{e}_m^{fi}(n)/\alpha_m(n-1)$ $\epsilon_m^{fi,\Re}(n) = 0.5\left(\widehat{\epsilon}_m^{fi}(n) + \tilde{\epsilon}_m^{fi}(n)\right)$ $\mathbf{g}_m^{fi,1}(n) = \widehat{\mathbf{w}}_m(n-1)\tilde{\epsilon}_m^{fi}(n), \mathbf{g}_m^{fi,2}(n) = \tilde{\mathbf{w}}_m(n-1)\widehat{\epsilon}_m^{fi}(n)$ $\mathbf{g}_m^{fi,3}(n) = \Delta W_m(n-1)\epsilon_m^{fi,\Re}(n)$ $\widehat{\mathbf{a}}_m^i(n) = \widehat{\mathbf{a}}_m^i(n-1) - \mathbf{g}_m^{fi,1}(n) - \mathbf{g}_m^{fi,3}(n)$ $\tilde{\mathbf{a}}_m^i(n) = \tilde{\mathbf{a}}_m^i(n-1) - \mathbf{g}_m^{fi,2}(n) - \mathbf{g}_m^{fi,3}(n)$ $\alpha_m^{fi}(n) = \lambda\alpha_m^{fi}(n-1) +$ $0.5\left(\widehat{e}_m^{fi}(n)\widehat{\epsilon}_m^{fi}(n) + \tilde{e}_m^{fi}(n)\tilde{\epsilon}_m^{fi}(n)\right)$
Kalman Gain Part
$\widehat{k}_{m+1}^{fi}(n) = \lambda^{-1}\widehat{e}_m^{fi}(n)\alpha_m^{fi}(n-1)$ $\tilde{k}_{m+1}^{fi}(n) = \lambda^{-1}\tilde{e}_m^{fi}(n)\alpha_m^{fi}(n-1)$ $\mathbf{g}_{m+1}^{wfi,1}(n) = \begin{bmatrix} 1 \\ \widehat{\mathbf{a}}_m^i(n-1) \end{bmatrix} \widehat{k}_{m+1}^{fi}(n)$ $\mathbf{g}_{m+1}^{wfi,2}(n) = \begin{bmatrix} 1 \\ \tilde{\mathbf{a}}_m^i(n-1) \end{bmatrix} \tilde{k}_{m+1}^{fi}(n)$ $k_{m+1}^{fi,\Im}(n) = 0.5\left(\widehat{k}_{m+1}^{fi}(n) - \tilde{k}_{m+1}^{fi}(n)\right)$ $\mathbf{g}_{m+1}^{wfi,3}(n) = \begin{bmatrix} 0 \\ \Delta A_m^i(n-1) \end{bmatrix} k_{m+1}^{fi,\Im}(n)$ $\mathcal{S}^i\widehat{\mathbf{w}}_{m+1} = \begin{bmatrix} 0 \\ \widehat{\mathbf{w}}_m(n-1) \end{bmatrix} + \mathbf{g}_{m+1}^{wfi,1}(n) - \mathbf{g}_{m+1}^{wfi,3}(n)$ $\mathcal{S}^i\tilde{\mathbf{w}}_{m+1} = \begin{bmatrix} 0 \\ \tilde{\mathbf{w}}_m(n-1) \end{bmatrix} + \mathbf{g}_{m+1}^{wfi,2}(n) - \mathbf{g}_{m+1}^{wfi,3}(n)$ $\alpha_{m+1}(n) = \alpha_m(n-1) +$ $0.5\left(\widehat{k}_{m+1}^{fi}(n)\widehat{e}_m^{fi}(n) + \tilde{k}_{m+1}^{fi}(n)\tilde{e}_m^{fi}(n)\right)$ $\mathcal{T}^i\widehat{\mathbf{w}}_{m+1}(n) = \begin{bmatrix} \widehat{\boldsymbol{\delta}}_m(n) \\ \widehat{\delta}_{m+1}(n) \end{bmatrix}, \mathcal{T}^i\tilde{\mathbf{w}}_{m+1}(n) = \begin{bmatrix} \tilde{\boldsymbol{\delta}}_m(n) \\ \tilde{\delta}_{m+1}(n) \end{bmatrix}$ $\mathbf{g}_m^{wbi,1}(n) = \widehat{\mathbf{b}}_m^i(n-1)\widehat{\delta}_{m+1}(n)$ $\mathbf{g}_m^{wbi,2}(n) = \widehat{\mathbf{b}}_m^i(n-1)\tilde{\delta}_{m+1}(n)$ $\delta_{m+1}^{\Im}(n) = 0.5\left(\widehat{\delta}_{m+1}(n) - \tilde{\delta}_{m+1}(n)\right)$ $\Delta B_m^i(n-1) = \widehat{\mathbf{b}}_m^i(n-1) - \tilde{\mathbf{b}}_m^i(n-1)$ $\mathbf{g}_m^{wbi,3}(n) = \Delta B_m^i(n-1)\delta_{m+1}^{\Im}(n)$ $\widehat{\mathbf{w}}_m(n) = \widehat{\boldsymbol{\delta}}_m(n) - \mathbf{g}_m^{wbi,1}(n) + \mathbf{g}_m^{wbi,3}(n)$ $\tilde{\mathbf{w}}_m(n) = \tilde{\boldsymbol{\delta}}_m(n) - \mathbf{g}_m^{wbi,2}(n) + \mathbf{g}_m^{wbi,3}(n)$ $\widehat{e}_m^{bi}(n) = \lambda\alpha_m^{bi}(n-1)\widehat{\delta}_{m+1}(n)$ $\tilde{e}_m^{bi}(n) = \lambda\alpha_m^{bi}(n-1)\tilde{\delta}_{m+1}(n)$ $\underline{\alpha}_m(n) = 1 + 0.5\left(\widehat{\mathbf{w}}_m^T(n)\widehat{\mathbf{x}}_m^{fi}(n) + \tilde{\mathbf{w}}_m^T(n)\tilde{\mathbf{x}}_m^{fi}(n)\right)$
Backward Prediction Part
$x_1^{bi}(n) = \widehat{\mathbf{b}}_m^{iT}(n-1)\widehat{\mathbf{x}}_m^{bi}(n), x_2^{bi}(n) = \tilde{\mathbf{b}}_m^{iT}(n-1)\tilde{\mathbf{x}}_m^{bi}(n)$ $x_3^{bi}(n) = \Delta B_m^{iT}(n-1)\mathbf{x}_m^{bi,\Re}(n)$ $\widehat{\underline{e}}_m^{bi}(n) = \widehat{x}^{bi}(n) + \left(x_1^{bi}(n) - x_3^{bi}(n)\right)$ $\tilde{\underline{e}}_m^{bi}(n) = \tilde{x}^{bi}(n) + \left(x_2^{bi}(n) + x_3^{bi}(n)\right)$ $\widehat{\Delta}^{bi}(n) = \widehat{\underline{e}}_m^{bi}(n) - \widehat{e}_m^{bi}(n)$ $\tilde{\Delta}^{bi}(n) = \tilde{\underline{e}}_m^{bi}(n) - \tilde{e}_m^{bi}(n)$ $\widehat{e}_m^{bi,\ell}(n) = \widehat{\underline{e}}_m^{bi}(n) + \widehat{\sigma}_\ell^{bi}\widehat{\Delta}^{bi}(n), \quad \ell = 1,2,3$ $\tilde{e}_m^{bi,\ell}(n) = \tilde{\widehat{e}}_m^{bi}(n) + \tilde{\sigma}_\ell^{bi}\tilde{\Delta}^{bi}(n), \quad i = 1,2,3$ $\alpha_m(n) = \alpha_{m+1}(n) -$ $0.5\left(\widehat{e}_m^{bi,1}(n)\widehat{\delta}_{m+1}(n) + \tilde{e}_m^{bi,1}(n)\tilde{\delta}_{m+1}(n)\right)$ $\Delta^{\alpha}(n) = \underline{\alpha}_m(n) - \alpha_m(n)$ $\alpha_m(n) = \underline{\alpha}_m(n) + \sigma^{\alpha}\Delta^{\alpha}(n)$ $\widehat{\epsilon}_m^{bi,\ell}(n) = \widehat{e}_m^{bi,\ell}(n)/\alpha_m(n), \quad \ell = 2,3$ $\tilde{\epsilon}_m^{bi,\ell}(n) = \tilde{e}_m^{bi,\ell}(n)/\alpha_m(n), \quad \ell = 2,3$ $\epsilon_m^{bi,\Im}(n) = 0.5\left(\widehat{\epsilon}_m^{bi,2}(n) - \tilde{\epsilon}_m^{bi,2}(n)\right)$ $\mathbf{g}_m^{bi,1}(n) = \widehat{\mathbf{w}}_m(n)\tilde{\epsilon}_m^{bi,2}(n), \mathbf{g}_m^{bi,2}(n) = \tilde{\mathbf{w}}_m(n)\widehat{\epsilon}_m^{bi,2}(n)$ $\Delta W_m(n) = \widehat{\mathbf{w}}_m(n) - \tilde{\mathbf{w}}_m(n)$ $\mathbf{g}_m^{bi,3}(n) = \Delta W_m(n)\epsilon_m^{bi,\Im}(n)$ $\widehat{\mathbf{b}}_m^i(n) = \widehat{\mathbf{b}}_m^i(n-1) - \mathbf{g}_m^{bi,1}(n) - \mathbf{g}_m^{bi,3}(n)$ $\tilde{\mathbf{b}}_m^i(n) = \tilde{\mathbf{b}}_m^i(n-1) - \mathbf{g}_m^{bi,2}(n) - \mathbf{g}_m^{bi,3}(n)$ $\alpha_m^{bi}(n) = \lambda\alpha_m^{bi}(n-1) +$ $0.5\left(\widehat{e}_m^{bi,3}(n)\widehat{\epsilon}_m^{bi,3}(n) + \tilde{e}_m^{bi,3}(n)\tilde{\epsilon}_m^{bi,3}(n)\right)$
END DO i

Table 1. The Strength Reduced (SR) FRLS adaptive Decision Feedback equalizer.