Chapter 7
Householder-Based RLS Algorithms

Athanasios A. Rontogiannis and Sergios Theodoridis

Abstract This chapter presents recursive least-squares (RLS) algorithms, which are based on numerically robust Householder transformations. Section 7.1 introduces the conventional orthogonal Householder transform and provides its geometrical interpretation. The hyperbolic Householder and the row (orthogonal and hyperbolic) Householder transforms are also briefly described. In Section 7.2, the Householder RLS (HRLS) algorithm is presented, its relation with the other known square-root RLS algorithms is discussed, and two applications, where the HRLS algorithm has been successfully applied, are presented. By considering a block-by-block update approach to the RLS problem, the block exact Householder QRD-RLS algorithm is developed in Section 7.3, which constitutes a generalization of the conventional QRD-RLS algorithm. Section 7.4 presents an inverse QRD-RLS algorithm, which employs block updating via the application of appropriate row orthogonal Householder transformations. Finally, a sliding window block RLS algorithm, which comprises a pair of row Householder transforms, is introduced in Section 7.5.

7.1 Householder Transforms

The Householder transform is an orthogonal matrix transform, named after Alston S. Householder, who has discovered it in the late 1950s [1]. It defines a norm-preserving transformation on a vector as the reflection of the vector with respect to a properly selected hyperplane. More specifically, let $\mathbf{x}$ and $\mathbf{y}$ be $M \times 1$ vectors. The projection of $\mathbf{x}$ onto $\mathbf{y}$ is then defined as follows:

Athanasios A. Rontogiannis
National Observatory of Athens, Athens – Greece
e-mail: tronto@space.noa.gr

Sergios Theodoridis
University of Athens, Athens – Greece
e-mail: stheodor@di.uoa.gr

J.A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering*,
DOI 10.1007/978-0-387-09734-3_7,

$$P_{\mathbf{y}}(\mathbf{x}) = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{y}}{||\mathbf{y}||^2}\mathbf{y} \tag{7.1}$$

By denoting with $P_{\mathbf{y}}^{\perp}(\mathbf{x})$ the projection of $\mathbf{x}$ onto the hyperplane, which is perpendicular to $\mathbf{y}$, we may write

$$\mathbf{x} = P_{\mathbf{y}}^{\perp}(\mathbf{x}) + P_{\mathbf{y}}(\mathbf{x}). \tag{7.2}$$

The Householder transform of $\mathbf{x}$ with respect to $\mathbf{y}$ is then given by [2]

$$T_{\mathbf{y}}(\mathbf{x}) = P_{\mathbf{y}}^{\perp}(\mathbf{x}) - P_{\mathbf{y}}(\mathbf{x}) \tag{7.3}$$

or, from (7.1) and (7.2),

$$T_{\mathbf{y}}(\mathbf{x}) = \left(\mathbf{I} - 2\frac{\mathbf{y}\mathbf{y}^{\mathrm{T}}}{||\mathbf{y}||^2}\right)\mathbf{x}. \tag{7.4}$$

The matrix

$$\mathbf{H} = \mathbf{I} - \beta\mathbf{y}\mathbf{y}^{\mathrm{T}} \tag{7.5}$$

where $\beta = 2/||\mathbf{y}||^2$ is easily seen to be orthogonal and symmetric. It is called Householder matrix and consists a rank-1 update to the identity matrix. The Householder transform is illustrated graphically in Figure 7.1. It can be seen from Figure 7.1 that multiplication of $\mathbf{x}$ with the Householder matrix $\mathbf{H}$ is equivalent to reflecting $\mathbf{x}$ with respect to the hyperplane, which is perpendicular to $\mathbf{y}$.

In many signal processing applications, it turns out to be convenient to transform an initial set of data to a sparse one, which is equivalent to the initial dataset in terms of a certain type of invariance. Mobilizing the Householder transformation in (7.5) and a suitable choice of $\mathbf{y}$ in terms of $\mathbf{x}$, it is possible to compress all the energy of $\mathbf{x}$ in just one element of $T_{\mathbf{y}}(\mathbf{x})$. Indeed, it can be easily verified from (7.4) that, by

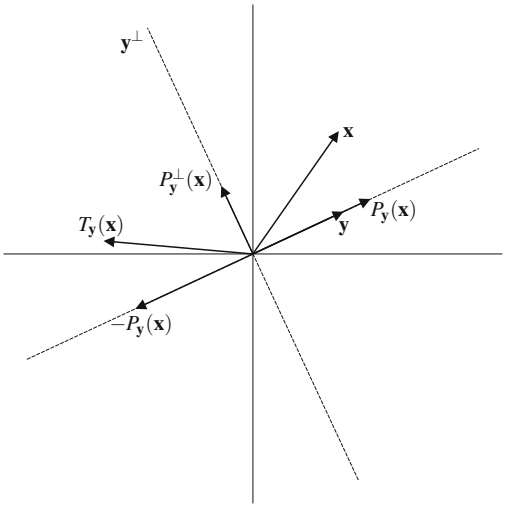

Fig. 7.1 Geometrical representation of the orthogonal Householder transform.

setting

$$\mathbf{y} = \mathbf{x} + ||\mathbf{x}||\mathbf{a}_i, \tag{7.6}$$

then

$$T_{\mathbf{y}}(\mathbf{x}) = \mathbf{H}\mathbf{x} = -||\mathbf{x}||\mathbf{a}_i, \tag{7.7}$$

where $\mathbf{a}_i$ is the ith column of the $M \times M$ identity matrix.[1] That is, a Householder matrix can be designed so that to annul a block of elements by reflecting a vector onto a coordinate axis. This procedure is illustrated in Figure 7.2.

The Householder transformation is a numerically robust approach to produce sparsity in data. Compared to the Givens rotations approach, which is used to zero one vector element at a time, multiple vector elements are zeroed with the application of one Householder reflection. Thus, an $M \times M$ Householder reflection can be considered equivalent to a succession of $M - 1$ Givens rotations, requiring less computations.

Due to its special form, a Householder matrix need not be computed explicitly. For instance, multiplication of $\mathbf{H}$ with an $M \times M$ matrix $\mathbf{A}$ is expressed as

$$\mathbf{H}\mathbf{A} = \mathbf{A} - 2\frac{\mathbf{y}}{||\mathbf{y}||^2}(\mathbf{y}^{\mathrm{T}}\mathbf{A}) \tag{7.8}$$

and thus a matrix-by-matrix multiplication is replaced by matrix-by-vector and vector-by-vector operations.

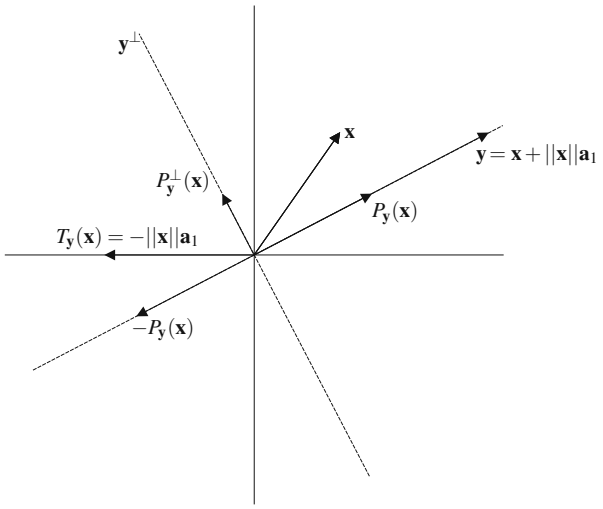

Fig. 7.2 The Householder transform that zeroes entries of vector $\mathbf{x}$.

[1] One could also choose $\mathbf{y} = \mathbf{x} - ||\mathbf{x}||\mathbf{a}_i$, which results in $\mathbf{H}\mathbf{x} = ||\mathbf{x}||\mathbf{a}_i$.

7.1.1 Hyperbolic Householder transforms

A slightly different matrix that can be used to introduce zeros in a vector or a column of a matrix is the hyperbolic Householder transform [3]. Let $\boldsymbol{\Phi}$ be an $M \times M$ diagonal matrix with entries $+1$ and -1. Then, the hyperbolic norm of a vector $\mathbf{x}$ is defined as follows:

$$\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x} = \sum_{i=1}^{M} \phi_i |x_i|^2 \tag{7.9}$$

where we assumed that $\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x} > 0$, ϕ_i is the ith diagonal element of $\boldsymbol{\Phi}$ and x_i the ith element of $\mathbf{x}$.

An $M \times M$ matrix $\mathbf{Q}$ satisfying

$$\mathbf{Q}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{Q} = \boldsymbol{\Phi}, \tag{7.10}$$

is called hypernormal matrix with respect to $\boldsymbol{\Phi}$ and has the property to preserve the hyperbolic norm of a vector, i.e., if $\mathbf{z} = \mathbf{Q}^{\mathrm{T}}\mathbf{x}$ then $\mathbf{z}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{z} = \mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x}$.

A matrix $\mathbf{H}$ defined as

$$\mathbf{H} = \boldsymbol{\Phi} - \beta \mathbf{y}\mathbf{y}^{\mathrm{T}}, \tag{7.11}$$

where $\beta = 2/(\mathbf{y}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{y})$, is called hyperbolic Householder transform. It is not difficult to verify that $\mathbf{H}$ is symmetric and hypernormal with respect to $\boldsymbol{\Phi}$. Similarly to the orthogonal Householder transforms, $\mathbf{y}$ can be properly selected so that application of $\mathbf{H}$ in (7.11) annihilates all but one entries of a vector. Indeed, by setting

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{x} + \left(\frac{x_i}{|x_i|} \sqrt{\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x}} \right) \mathbf{a}_i \tag{7.12}$$

and assuming that $\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x} > 0$ and $\phi_i = 1$, then

$$\mathbf{H}\mathbf{x} = -\left(\frac{x_i}{|x_i|} \sqrt{\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x}} \right) \mathbf{a}_i \tag{7.13}$$

and all hyperbolic energy of $\mathbf{x}$, i.e., $\mathbf{x}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{x}$, is compressed to its ith entry.

7.1.2 Row Householder transforms

The Householder transforms discussed so far are designed to introduce zeros in a *column* of a matrix. As shown in [4], Householder matrices (either orthogonal or hyperbolic) can also be constructed to zero one *row* of a matrix. Let $\mathbf{A}$ be an $(M+1) \times M$ matrix expressed as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{C} \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix} \tag{7.14}$$

where the $M \times M$ matrix $\mathbf{C}$ is assumed to be invertible. Suppose we wish to eliminate the last row $\mathbf{b}^{\mathrm{T}}$ of $\mathbf{A}$ by applying an $(M+1) \times (M+1)$ orthogonal Householder transformation, as the one given in (7.5), i.e.,

$$\mathbf{H} \begin{bmatrix} \mathbf{C} \\ \mathbf{b}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{C}} \\ \mathbf{0}^{\mathrm{T}} \end{bmatrix}. \tag{7.15}$$

It has been proven in [4] that for an $\mathbf{H}$ to satisfy the last equation, its defining $(M+1) \times 1$ vector $\mathbf{y}$ must be constructed as

$$\mathbf{y} = \frac{1}{||\mathbf{b}||} \begin{bmatrix} \mathbf{C}^{-\mathrm{T}}\mathbf{b} \\ -1 - \sqrt{1 + (\mathbf{C}^{-\mathrm{T}}\mathbf{b})^{\mathrm{T}}(\mathbf{C}^{-\mathrm{T}}\mathbf{b})} \end{bmatrix}. \tag{7.16}$$

Note that finding $\mathbf{y}$ requires the computation of the inverse of $\mathbf{C}$.

By following a similar analysis, it can be shown [4] that a hyperbolic Householder matrix, defined as in (7.11), can be constructed to eliminate $\mathbf{b}^{\mathrm{T}}$. Its defining vector will be now given by

$$\mathbf{y} = \frac{1}{||\mathbf{b}||} \begin{bmatrix} \mathbf{C}^{-\mathrm{T}}\mathbf{b} \\ -1 - \sqrt{1 + \phi_{M+1}(\mathbf{C}^{-\mathrm{T}}\mathbf{b})^{\mathrm{T}}\tilde{\boldsymbol{\Phi}}(\mathbf{C}^{-\mathrm{T}}\mathbf{b})} \end{bmatrix}, \tag{7.17}$$

where ϕ_{M+1} is the last diagonal element of the $(M+1) \times (M+1)$ matrix $\boldsymbol{\Phi}$ and $\tilde{\boldsymbol{\Phi}}$ its upper left $M \times M$ diagonal block.

Remarks

In the previous analysis, matrices and vectors with real entries have been assumed. In case of complex entries:

- The analysis of Section 7.1 remains under the condition that the ith element of vector $\mathbf{x}$ is real. The orthogonal Householder transform is now given by

$$\mathbf{H} = \mathbf{I} - \frac{2}{||\mathbf{y}||^2}\mathbf{y}\mathbf{y}^{\mathrm{H}} \tag{7.18}$$

- The hyperbolic Householder transform is expressed as

$$\mathbf{H} = \boldsymbol{\Phi} - \frac{2}{\mathbf{y}^{\mathrm{H}}\boldsymbol{\Phi}\mathbf{y}}\mathbf{y}\mathbf{y}^{\mathrm{H}}, \tag{7.19}$$

  while Equations (7.12) and (7.13) still hold with the hyperbolic energy now defined as $\mathbf{x}^{\mathrm{H}}\boldsymbol{\Phi}\mathbf{x}$.
- The analysis of Section 7.1.2 still holds by replacing simple transposition with conjugate transposition in Equations (7.14), (7.15), (7.16), and (7.17), and constructing Householder matrices from (7.18) and (7.19).

7.2 The Householder RLS (HRLS) Algorithm

In Chapter 3, the conventional and the inverse QR-decomposition recursive least-squares (QRD-RLS) algorithms have been presented. The main characteristic of these algorithms is that the triangular Cholesky factor of the input data correlation matrix (or its inverse) is updated in each iteration based on a sequence of Givens rotations. It is well known that a symmetric positive definite matrix has an infinite number of square-roots. These can be expressed as the product of an arbitrary orthogonal matrix with the respective Cholesky factor. In this section, we present an RLS algorithm, which updates in time an arbitrary square-root of the input data correlation matrix and provides naturally the LS weight vector. It will be shown that such an update is performed by applying a properly constructed data dependent Householder matrix.

Let $\mathbf{S}(k)$ be an arbitrary (not necessarily triangular) $(N+1)\times(N+1)$ square-root factor of the data correlation matrix $\mathbf{R}(k)$ at time k. Then the following relation holds:

$$\mathbf{R}(k)=\mathbf{S}^{\mathrm{T}}(k)\mathbf{S}(k) \tag{7.20}$$

Based on the new data vector $\mathbf{x}(k)$ and the square-root factor of the previous time instant, we define the following vector:[2]

$$\mathbf{u}(k)=\frac{\mathbf{S}^{-\mathrm{T}}(k-1)\mathbf{x}(k)}{\sqrt{\lambda}} \tag{7.21}$$

where λ is the forgetting factor of the exponentially weighted LS cost function. Let now $\mathbf{P}(k)$ be an $(N+2)\times(N+2)$ orthogonal matrix, which performs the following transformation:

$$\mathbf{P}(k)\begin{bmatrix}1\\ \mathbf{u}(k)\end{bmatrix}=\begin{bmatrix}-\delta(k)\\ \mathbf{0}\end{bmatrix}, \tag{7.22}$$

where $\delta(k)$ is a positive scalar. According to (7.22), matrix $\mathbf{P}(k)$ zeros $\mathbf{u}(k)$, which coincides with the last $N+1$ elements of the vector on the left-hand-side of the equation. Due to the orthogonality of $\mathbf{P}(k)$ and (7.21), the positive scalar $\delta(k)$ is given by

$$\delta(k)=\sqrt{1+||\mathbf{u}(k)||^2}=\sqrt{1+\lambda^{-1}\mathbf{x}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}. \tag{7.23}$$

It is most interesting that the orthogonal matrix $\mathbf{P}(k)$, which satisfies (7.22), also updates in time the inverse square-root factor of $\mathbf{R}(k-1)$. Indeed, by formulating the equation

$$\mathbf{P}(k)\begin{bmatrix}\mathbf{0}^{\mathrm{T}} & 1\\ \lambda^{-1/2}\mathbf{S}^{-\mathrm{T}}(k-1) & \mathbf{u}(k)\end{bmatrix}=\begin{bmatrix}\mathbf{z}^{\mathrm{T}}(k) & -\delta(k)\\ \mathbf{Y}(k) & \mathbf{0}\end{bmatrix} \tag{7.24}$$

[2] This vector is reminiscent of the so-called Kalman gain vector, which appears in the conventional RLS algorithm [5]. The Kalman gain vector is obtained by replacing $\mathbf{S}^{-\mathrm{T}}(k-1)$ in (7.21) by $\mathbf{R}^{-\mathrm{T}}(k-1)$.

and multiplying each side of (7.24) by its transpose and equating parts, $\mathbf{z}(k)$ and $\mathbf{Y}(k)$ are obtained from the following set of expressions:

$$\mathbf{z}(k) = -\frac{\mathbf{S}^{-1}(k-1)\mathbf{u}(k)}{\sqrt{\lambda}\,\delta(k)} = -\frac{\mathbf{R}^{-1}(k-1)\mathbf{x}(k)}{\lambda\,\delta(k)} \tag{7.25}$$

and

$$\mathbf{Y}^{\mathrm{T}}(k)\mathbf{Y}(k) = \lambda^{-1}\mathbf{R}^{-1}(k-1) - \mathbf{z}(k)\mathbf{z}^{\mathrm{T}}(k). \tag{7.26}$$

Note that, according to (7.25), $\mathbf{z}(k)$ is a scaled version of the Kalman gain vector [5], which can be used to update the LS weight vector. Moreover, let us consider the well-known LS input correlation matrix update equation

$$\mathbf{R}(k) = \lambda\mathbf{R}(k-1) + \mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k). \tag{7.27}$$

Application of the matrix inversion lemma in (7.27) leads to the expression on the right-hand-side of (7.26). Thus, $\mathbf{R}^{-1}(k) = \mathbf{Y}^{\mathrm{T}}(k)\mathbf{Y}(k)$, verifying that

$$\mathbf{Y}(k) = \mathbf{S}^{-\mathrm{T}}(k). \tag{7.28}$$

Since Equation (7.22) represents an annihilation of a block of elements in a vector, the natural choice for $\mathbf{P}(k)$ is a Householder matrix. According to the analysis of Section 7.1, a Householder matrix $\mathbf{P}(k)$ satisfying (7.22) is expressed as follows:

$$\mathbf{P}(k) = \mathbf{I} - \beta(k)\mathbf{y}(k)\mathbf{y}^{\mathrm{T}}(k), \tag{7.29}$$

where

$$\mathbf{y}(k) = \begin{bmatrix} 1+\delta(k) \\ \mathbf{u}(k) \end{bmatrix}, \tag{7.30}$$

and

$$\beta(k) = \frac{1}{\delta(k)(1+\delta(k))}. \tag{7.31}$$

The Householder RLS (HRLS) algorithm is completed with the formulas required for the computation of the *a priori* LS error and the update of the LS weight vector, respectively, i.e.,

$$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k-1)\mathbf{x}(k), \text{ and} \tag{7.32}$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{e(k)}{\delta(k)}\mathbf{z}(k). \tag{7.33}$$

The basic steps of the HRLS algorithm are summarized in Table 7.1, where the structure of the orthogonal matrix $\mathbf{P}(k)$ has not been taken into consideration. However, by exploiting the special form of the employed Householder transformation, $\mathbf{P}(k)$ need not be explicitly computed. More specifically, from (7.28) to (7.31) and (7.24), the following explicit update equation for $\mathbf{S}^{-1}(k-1)$ is obtained:

Table 7.1 The basic steps of the HRLS algorithm.

HRLS
Initialize $0 \ll \lambda < 1, \mathbf{w}(0) = \mathbf{0}, \varepsilon \approx 1/\sigma_x^2$ and $\mathbf{S}^{-1}(0) = \sqrt{\varepsilon}\mathbf{I}$
for each k
{ Computing $\mathbf{u}(k)$:
$\mathbf{u}(k) = \lambda^{-1/2}\mathbf{S}^{-\mathrm{H}}(k-1)\mathbf{x}(k)$
Obtaining $\mathbf{P}(k)$ and $\delta(k)$:
$\mathbf{P}(k)\begin{bmatrix} 1 \\ \mathbf{u}(k) \end{bmatrix} = \begin{bmatrix} -\delta(k) \\ \mathbf{0} \end{bmatrix}$
Updating $\mathbf{S}^{-1}(k-1)$:
$\mathbf{P}(k)\begin{bmatrix} \mathbf{0}^{\mathrm{T}} \\ \lambda^{-1/2}\mathbf{S}^{-\mathrm{H}}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{z}^{\mathrm{H}}(k) \\ \mathbf{S}^{-\mathrm{H}}(k) \end{bmatrix}$
Obtaining $e(k)$ and $\mathbf{w}(k)$:
$e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$
$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{e^*(k)}{\delta(k)}\mathbf{z}(k)$
}

$$\mathbf{S}^{-1}(k) = \lambda^{-1/2}\mathbf{S}^{-1}(k-1) + \frac{\mathbf{z}(k)\mathbf{u}^{\mathrm{T}}(k)}{1+\delta(k)}. \tag{7.34}$$

Thus, we are led to an analytical form of the HRLS algorithm, as illustrated in Table 7.2.

The HRLS algorithm was originally introduced in [6] and [7] and later rediscovered independently in [8]. The main advantage of the algorithm is its numerically robust behavior in a finite-precision environment. This is clearly shown in Figure 7.3, where the mean squared error of the RLS and HRLS algorithms is depicted for a numerically unstable situation. More specifically, we have considered a system of order $N = 8$ and an input signal is generated as follows:

$$x(k) = \cos(0.05\pi k) + \sqrt{2}\cos(0.3\pi k) + \eta(k) \tag{7.35}$$

Table 7.2 Analytical form of the HRLS algorithm.

HRLS
Initialize $0 \ll \lambda < 1, \mathbf{w}(0) = \mathbf{0}, \varepsilon \approx 1/\sigma_x^2$ and $\mathbf{S}^{-1}(0) = \sqrt{\varepsilon}\mathbf{I}$
for each k
{ $\mathbf{u}(k) = \lambda^{-1/2}\mathbf{S}^{-\mathrm{H}}(k-1)\mathbf{x}(k)$
$\delta(k) = \sqrt{1 + \|\mathbf{u}(k)\|^2}$
$\mathbf{z}(k) = -\frac{\lambda^{-1/2}\mathbf{S}^{-1}(k-1)\mathbf{u}(k)}{\delta(k)}$
$\mathbf{S}^{-1}(k) = \lambda^{-1/2}\mathbf{S}^{-1}(k-1) + \frac{\mathbf{z}(k)\mathbf{u}^{\mathrm{H}}(k)}{1+\delta(k)}$
$e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k-1)\mathbf{x}(k)$
$\mathbf{w}(k) = \mathbf{w}(k-1) - \frac{e^*(k)}{\delta(k)}\mathbf{z}(k)$
}

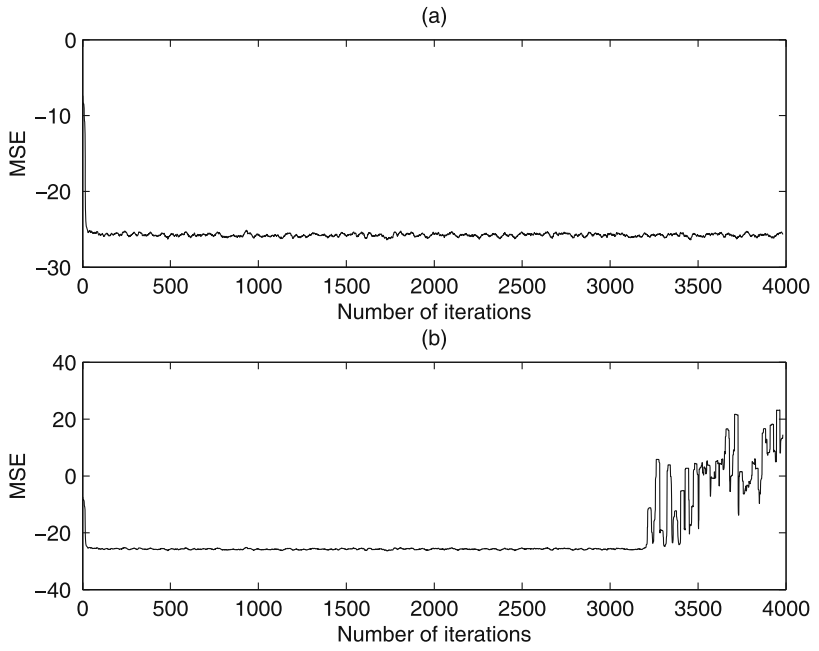

Fig. 7.3 (**a**) Mean squared error of HRLS, (**b**) Mean squared error of RLS.

where $\eta(k)$ is zero-mean Gaussian random noise with variance equal to 10^{-10}. Note that the 8×8 autocorrelation matrix of the input signal is nearly singular. A forgetting factor $\lambda = 0.99$ has been used in both schemes. As observed from Figure 7.3, RLS diverges after a number of iterations, while HRLS retains a numerically robust behavior.

Concerning the computational complexity, the HRLS algorithm requires slightly more arithmetic operations compared to the other square-root RLS schemes (conventional and inverse QRD-RLS algorithms). However, as indicated in the analysis performed in [9], the HRLS is the fastest numerically robust RLS algorithm in terms of MATLAB execution time, irrespective of the problem size N. This is due to the fact that the HRLS algorithm includes simple matrix-by-vector and vector-by-vector operations, which are best suited for implementation in the MATLAB environment.

It should be emphasized that the HRLS algorithm is closely related to the inverse QRD-RLS algorithm. In both algorithms, an orthogonal transformation is constructed by zeroing a vector quantity and then this transformation is applied for the update of an inverse square-root factor of the input data correlation matrix.

As shown in Chapter 3, the orthogonal transformation used in the inverse QRD-RLS algorithm can also be applied for the update of the square-root factor itself,

leading to the QRD-RLS algorithm. It is easily verified that the same holds for the HRLS algorithm. More specifically, as proven in [6], the Householder matrix $\mathbf{P}(k)$ satisfies the following equations:

$$\mathbf{P}(k)\begin{bmatrix} -\mathbf{x}^{\mathrm{T}}(k) \\ \lambda^{1/2}\mathbf{S}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{0}^{\mathrm{T}}(k) \\ \mathbf{S}(k) \end{bmatrix}, \text{ and} \tag{7.36}$$

$$\mathbf{P}(k)\begin{bmatrix} -d(k) \\ \lambda^{1/2}\hat{\mathbf{d}}_{q_2}(k-1) \end{bmatrix} = \begin{bmatrix} e_{q_1}(k) \\ \hat{\mathbf{d}}_{q_2}(k) \end{bmatrix}. \tag{7.37}$$

where $e_{q_1}(k)$ is the rotated estimation error defined in Chapter 3 and $\hat{\mathbf{d}}_{q_2}(k)$ is an orthogonal transformation of the rotated desired signal vector satisfying $\mathbf{S}(k)\mathbf{w}(k) = \hat{\mathbf{d}}_{q_2}(k)$. In the QRD-RLS algorithm, the generation of the orthogonal transformation and the update of the triangular Cholesky factor are performed jointly, without involving the inverse Cholesky factor. Similarly, from (7.36), except for the update of $\mathbf{S}(k-1)$, matrix $\mathbf{P}(k)$ could also be produced as an orthogonal row Householder reflection, which zeroes row $-\mathbf{x}^{\mathrm{T}}(k)$ with respect to the matrix $\lambda^{1/2}\mathbf{S}(k-1)$. However, the analysis of Section 7.1.2 has shown that, in order to construct such a matrix, the inverse of the square-root factor $\mathbf{S}(k-1)$ is also required (see Equation (7.16)). Thus, it seems that a Householder-based RLS algorithm equivalent to the QRD-RLS cannot be derived.

In the analysis presented in [10], it is shown that several variants of the RLS family are closely related to algorithms developed for the Kalman filtering problem. Under this framework, the QRD-RLS and the inverse QRD-RLS schemes are akin to the so-called information and square-root covariance filters, respectively [11, 12]. In a similar way, the HRLS algorithm is related to Potter's square-root covariance filter, which was the first square-root Kalman filter implementation, developed in the early 1960s [13]. It is noteworthy that Potter's square-root filter was the variant of the Kalman filter, which, due to its exceptional numerical properties, has been utilized in the navigation software of the Apollo system [14].

7.2.1 Applications

In the following, we briefly review two specific applications, namely adaptive prewhitening and equalization in wideband code division multiple access (WCDMA) downlink, whereby the HRLS algorithm has been successfully utilized as a component of a larger system.

7.2.1.1 Adaptive pre-whitening

In several applications, such as blind source separation (BSS) or independent component analysis (ICA), it is desirable that the input signal measurements are whitened prior to applying any specific method. This procedure is known as prewhitening [15]. Let us assume that the signal vector $\mathbf{x}(k)$ is wide sense stationary

with zero mean and autocorrelation matrix $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)]$, where $E[\cdot]$ denotes statistical expectation. Then, to pre-whiten $\mathbf{x}(k)$ we need to determine a matrix transformation $\mathbf{T}$ such that, if

$$\mathbf{v}(k) = \mathbf{T}\mathbf{x}(k), \tag{7.38}$$

then

$$E[\mathbf{v}(k)\mathbf{v}^{\mathrm{T}}(k)] = \mathbf{T}\mathbf{R}\mathbf{T}^{\mathrm{T}} = \mathbf{I}. \tag{7.39}$$

Apparently, (7.39) is satisfied if $\mathbf{T}$ is selected as the transpose of a square-root of $\mathbf{R}^{-1}$. Note that, under time-varying conditions, the whitening transformation $\mathbf{T}$ must be properly updated in time. Based on these two observations, a reasonable choice for $\mathbf{T}$ is

$$\mathbf{T} = \mathbf{S}^{-\mathrm{T}}(k-1), \tag{7.40}$$

i.e.,

$$\mathbf{v}(k) = \mathbf{S}^{-\mathrm{T}}(k-1)\mathbf{x}(k). \tag{7.41}$$

This transformation maintains the whitening property with respect to the deterministic autocorrelation matrix for every k, i.e.,

$$\mathbf{S}^{-\mathrm{T}}(k-1)\mathbf{R}(k-1)\mathbf{S}(k-1) = \mathbf{I}. \tag{7.42}$$

From (7.41) the transformed data vector is now $\mathbf{v}(k) = \sqrt{\lambda}\mathbf{u}(k)$, with $\mathbf{u}(k)$ defined in (7.21). Assuming $\mathbf{x}(k)$ to be wide sense stationary, $\mathbf{v}(k)$ enjoys the following property [8]:

$$\lim_{k\to\infty} E[\mathbf{v}(k)\mathbf{v}^{\mathrm{T}}(k)] \approx (1-\lambda)\mathbf{I}. \tag{7.43}$$

The adaptive whitening procedure is summarized by the first four steps of the algorithm in Table 7.2. This procedure is numerically robust, as the update of the whitening transformation stems from the application of a Householder reflection.

7.2.1.2 Equalization in WCDMA downlink

Wideband code division multiple access (WCDMA) has been adopted in several modern telecommunication standards such as the Universal Mobile Telecommunication Systems (UMTS) standard. Due to strong interference from other users, a critical task in such systems is the design of an equalizer in the downlink. Most often, the conventional RAKE receiver is used, which is, however, interference limited. Another solution is to employ a linear minimum mean squares error equalizer, which provides the estimate of the kth symbol of the mth user as [16]

$$\hat{y}_m(k) = \mathbf{c}_m^{\mathrm{T}}(k)\boldsymbol{\Theta}^{\mathrm{T}}(\sigma_y^2\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{T}} + \sigma_\eta^2\mathbf{I})^{-1}\mathbf{x}(k), \tag{7.44}$$

where $\mathbf{x}(k)$ is the vector of the received samples, $\boldsymbol{\Theta}$ is the system channel matrix, $\mathbf{c}_m^{\mathrm{T}}(k)$ is the spreading sequence of user k for symbol m, and $\sigma_y^2, \sigma_\eta^2$ are the variances of the transmitted sequence and the receiver noise, respectively. From (7.44) we can identify that the equalizer consists of two parts; the conventional RAKE receiver

$\mathbf{c}_m^{\mathrm{T}}(k)\boldsymbol{\Theta}^{\mathrm{T}}$ and a preceding filter $(\sigma_y^2\boldsymbol{\Theta}\boldsymbol{\Theta}^{\mathrm{T}}+\sigma_\eta^2\mathbf{I})^{-1}$. It can be easily shown that this prefilter coincides with the inverse autocorrelation matrix $\mathbf{R}^{-1}$ of the received sequence $\mathbf{x}(k)$. By considering a Toeplitz approximation of $\mathbf{R}$, the prefilter coefficients are given by the elements of the middle row of $\mathbf{R}^{-1}$ [16]. If $\mathbf{s}_d^{\mathrm{T}}$ stands for the middle row of the square-root factor $\mathbf{S}^{-1}$ of $\mathbf{R}^{-1}$, the prefilter coefficients will be given by the elements of the following vector:

$$\mathbf{v}=\mathbf{S}^{-1}\mathbf{s}_d. \tag{7.45}$$

Therefore, explicit computation of the inverse autocorrelation matrix is not necessary. Instead, its square-root factor only need to be computed. Moreover, in an adaptive equalization setup, the prefilter coefficients can be updated by applying the HRLS algorithm, which provides, for every time instant the inverse square-root factor $\mathbf{S}^{-1}(k)$ [16], as shown in Table 7.2.

7.3 The Householder Block Exact QRD-RLS Algorithm

In a *block* RLS algorithm, the LS weight vector is updated in a data block-by-block basis, instead of the sample-by-sample updating as it is the case for the conventional RLS algorithms. In [17], a block QRD-RLS algorithm has been developed, where an exponentially block weighting was utilized. In the following, we slightly modify this algorithm, so that the LS weight vector, obtained at each block iteration, coincides with its sample-by-sample counterpart as this is computed by the conventional QRD-RLS algorithm. Such an algorithm is called block exact QRD-RLS algorithm and belongs to a more general class of such algorithms [18].

Let us assume a block length equal to Q. The LS error vector corresponding to $k+1$ blocks can be expressed as follows:

$$\mathbf{e}(k)=\begin{bmatrix}\mathbf{e}_k\\ \mathbf{e}_{k-1}\\ \vdots\\ \mathbf{e}_0\end{bmatrix}=\mathbf{d}(k)-\mathbf{X}(k)\mathbf{w}(k). \tag{7.46}$$

Let us define for $i=0,\ldots,k$ the $Q\times(N+1)$ data block $\mathbf{X}_i^{\mathrm{T}}$ and the $Q\times 1$ desired response block $\mathbf{d}_i$ as follows:

$$\mathbf{X}_i^{\mathrm{T}}=\begin{bmatrix}\mathbf{x}^{\mathrm{T}}((i+1)Q-1)\\ \vdots\\ \mathbf{x}^{\mathrm{T}}(iQ+1)\\ \mathbf{x}^{\mathrm{T}}(iQ)\end{bmatrix}=\begin{bmatrix}\mathbf{x}_{0,i}\ \mathbf{x}_{1,i}\cdots\mathbf{x}_{N,i}\end{bmatrix} \tag{7.47}$$

and

$$\mathbf{d}_i = \begin{bmatrix} d((i+1)Q-1) \\ \vdots \\ d(iQ+1) \\ d(iQ) \end{bmatrix}. \tag{7.48}$$

Then, the input data matrix $\mathbf{X}(k)$ and the desired response vector $\mathbf{d}(k)$ in (7.46) are expressed as follows:

$$\mathbf{X}(k) = \begin{bmatrix} \boldsymbol{\Lambda}\mathbf{X}_k^{\mathrm{T}} \\ \lambda^{Q/2}\boldsymbol{\Lambda}\mathbf{X}_{k-1}^{\mathrm{T}} \\ \vdots \\ \lambda^{kQ/2}\boldsymbol{\Lambda}\mathbf{X}_0^{\mathrm{T}} \end{bmatrix} \tag{7.49}$$

and

$$\mathbf{d}(k) = \begin{bmatrix} \boldsymbol{\Lambda}\mathbf{d}_k \\ \lambda^{Q/2}\boldsymbol{\Lambda}\mathbf{d}_{k-1} \\ \vdots \\ \lambda^{kQ/2}\boldsymbol{\Lambda}\mathbf{d}_0 \end{bmatrix}, \tag{7.50}$$

where $\boldsymbol{\Lambda}$ is a $Q \times Q$ weighting matrix given by

$$\boldsymbol{\Lambda} = \begin{bmatrix} 1 & \cdots & 0 & 0 \\ 0 & \lambda^{1/2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda^{(Q-1)/2} \end{bmatrix}. \tag{7.51}$$

It is not difficult to verify from the above definitions that minimization of the norm of $\mathbf{e}(k)$ given in (7.46) with respect to $\mathbf{w}(k)$, provides the exact exponentially weighted LS solution for time instant $(k+1)Q-1$, i.e., $\mathbf{w}(k)$ minimizes the following cost function:

$$J(\mathbf{w}(k)) = \sum_{j=0}^{(k+1)Q-1} \lambda^{(k+1)Q-1-j}[d(j) - \mathbf{x}^{\mathrm{T}}(j)\mathbf{w}(k)]^2. \tag{7.52}$$

To see how we can obtain this solution adaptively in a block-by-block basis, we observe that $\mathbf{X}(k)$ from (7.49) can be rewritten as

$$\mathbf{X}(k) = \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{T}} \\ \lambda^{Q/2}\mathbf{X}(k-1) \end{bmatrix}, \tag{7.53}$$

where $\tilde{\mathbf{X}}_k^{\mathrm{T}} = \boldsymbol{\Lambda}\mathbf{X}_k^{\mathrm{T}}$. From the last equation, the deterministic data correlation matrix at time $(k+1)Q-1$ can be expressed as follows:

$$\mathbf{R}(k) = \lambda^{Q}\mathbf{R}(k-1) + \tilde{\mathbf{X}}_k\tilde{\mathbf{X}}_k^{\mathrm{T}}, \tag{7.54}$$

where $\mathbf{R}(k-1) = \mathbf{X}^{\mathrm{T}}(k-1)\mathbf{X}(k-1)$ is the data correlation matrix at time $kQ-1$. By expressing $\mathbf{R}(k)$ and $\mathbf{R}(k-1)$ by the respective Cholesky factorizations, (7.54) is written as

$$\mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) = \lambda^{Q}\mathbf{U}^{\mathrm{T}}(k-1)\mathbf{U}(k-1) + \tilde{\mathbf{X}}_k\tilde{\mathbf{X}}_k^{\mathrm{T}}, \tag{7.55}$$

where $\mathbf{U}(k), \mathbf{U}(k-1)$ are assumed to be $(N+1)\times(N+1)$ upper triangular matrices. From (7.55), a block time update of the Cholesky factor of the data correlation matrix can be realized according to the following relation:

$$\tilde{\mathbf{P}}(k)\begin{bmatrix}\tilde{\mathbf{X}}_k^{\mathrm{T}} \\ \lambda^{Q/2}\mathbf{U}(k-1)\end{bmatrix} = \begin{bmatrix}\mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}(k)\end{bmatrix}, \tag{7.56}$$

where $\tilde{\mathbf{P}}(k)$ is an orthogonal matrix. As suggested in (7.56), $\tilde{\mathbf{P}}(k)$ must be properly selected to zero the $Q\times(N+1)$ block $\tilde{\mathbf{X}}_k^{\mathrm{T}}$ with respect to the triangular matrix $\lambda^{Q/2}\mathbf{U}(k-1)$. Moreover, by following similar analysis as in Chapter 3 for the conventional QRD-RLS algorithm, it can be easily shown that $\tilde{\mathbf{P}}(k)$ block updates in time a rotated desired $(N+1)\times 1$ signal vector, $\mathbf{d}_{q_2}(k)$, as in

$$\tilde{\mathbf{P}}(k)\begin{bmatrix}\tilde{\mathbf{d}}_k \\ \lambda^{Q/2}\mathbf{d}_{q_2}(k-1)\end{bmatrix} = \begin{bmatrix}\tilde{\mathbf{e}}_k \\ \mathbf{d}_{q_2}(k)\end{bmatrix}, \tag{7.57}$$

where $\tilde{\mathbf{d}}_k = \boldsymbol{\Lambda}\mathbf{d}_k$ and $\tilde{\mathbf{e}}_k$ being a $Q\times 1$ rotated error block. The exact LS weight vector at time $(k+1)Q-1$ is then given by the following triangular system of equations:

$$\mathbf{U}(k)\mathbf{w}(k) = \mathbf{d}_{q_2}(k), \tag{7.58}$$

which can be easily solved using back-substitution.

As mentioned before, matrix $\tilde{\mathbf{P}}(k)$ must be designed to eliminate the block $\tilde{\mathbf{X}}_k^{\mathrm{T}} = [\tilde{\mathbf{x}}_{0,k}, \tilde{\mathbf{x}}_{1,k}, \cdots, \tilde{\mathbf{x}}_{N,k}]$, where $\tilde{\mathbf{x}}_{n,k} = \boldsymbol{\Lambda}\mathbf{x}_{n,k}$, while retaining the triangular structure of the Cholesky factor. By inspecting (7.56), it is easily shown that $\tilde{\mathbf{P}}(k)$ can be expressed as a product of $N+1$ orthogonal $(N+Q+1)\times(N+Q+1)$ Householder matrices as follows:

$$\tilde{\mathbf{P}}(k) = \tilde{\mathbf{P}}_N(k)\tilde{\mathbf{P}}_{N-1}(k)\cdots\tilde{\mathbf{P}}_0(k). \tag{7.59}$$

Matrix $\tilde{\mathbf{P}}_n(k)$, $n = 0,1,\ldots,N$, zeroes the $(n+1)$-th column of the input data block with respect to the corresponding diagonal element of the upper triangular factor, i.e.,

$$\tilde{\mathbf{P}}_n(k)\begin{bmatrix}\mathbf{0}\cdots\tilde{\mathbf{x}}_{n,k}^{(n)} & \tilde{\mathbf{x}}_{n+1,k}^{(n)} & \cdots\tilde{\mathbf{x}}_{N,k}^{(n)} \\ & \mathbf{U}^{(n)}(k-1) & \end{bmatrix} = \begin{bmatrix}\mathbf{0}\cdots\mathbf{0} & \tilde{\mathbf{x}}_{n+1,k}^{(n+1)} & \cdots\tilde{\mathbf{x}}_{N,k}^{(n+1)} \\ & \mathbf{U}^{(n+1)}(k-1) & \end{bmatrix}, \tag{7.60}$$

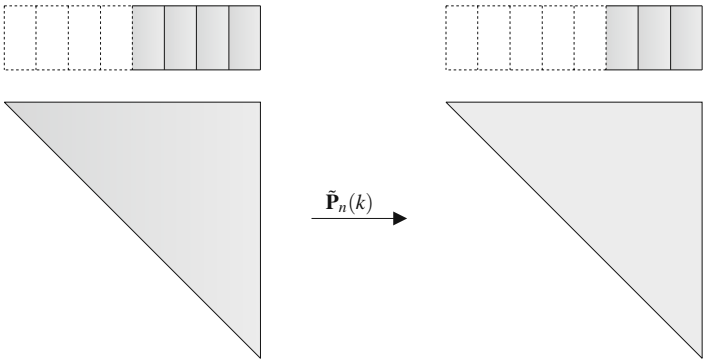

Fig. 7.4 Application of Householder transformations in the block exact QRD-RLS algorithm.

which is also illustrated in Figure 7.4. In (7.60), $\mathbf{U}^{(0)}(k-1) = \lambda^{Q/2}\mathbf{U}(k-1)$, $\mathbf{U}^{(N)}(k-1) = \mathbf{U}(k)$, and $\tilde{\mathbf{x}}_{j,k}^{(n)}, j = n, \ldots, N$ is the $(j+1)$-th column of the data block after the application of matrices $\tilde{\mathbf{P}}_0(k), \ldots, \tilde{\mathbf{P}}_{n-1}(k)$. To be more specific, $\tilde{\mathbf{P}}_n(k)$ is an orthogonal Householder matrix, whose defining vector, according to (7.6), is given by[3]

$$\mathbf{y}_n(k) = \begin{bmatrix} \tilde{\mathbf{x}}_{n,k}^{(n)} \\ \mathbf{0}_n \\ u_{n+1,n+1} + \rho_{n+1} \\ \mathbf{0}_{N-n} \end{bmatrix}, \tag{7.61}$$

where $u_{n+1,n+1}$ is the $(n+1)$-th diagonal element of $\mathbf{U}^{(n)}(k-1)$ and $\rho_{n+1}^2 = u_{n+1,n+1}^2 + ||\tilde{\mathbf{x}}_{n,k}^{(n)}||^2$.

The block exact Householder QRD-RLS algorithm is summarized in Table 7.3. It must be noticed that block updating using Householder transforms results in remarkable reduction in computational complexity compared to sample-by-sample updating using Givens rotations. More specifically, the Householder-based approach requires $QN^2 + \mathcal{O}[QN]$ arithmetic operations, whereas the Givens rotations approach needs $2QN^2 + \mathcal{O}[QN]$ operations, that is, the computational complexity is almost halved.

Moreover, as described in [17], a slightly modified version of the block Householder QRD-RLS algorithm can be implemented in a systolic array architecture, providing a throughput rate similar to that of the Givens rotations method.

[3] To be precise, due to the definition of the Householder transform in (7.5), (7.6), and (7.7), the resulting $\mathbf{U}(k)$ is the Cholesky factor of $\mathbf{R}(k)$ multiplied by -1. This, however, does not affect the analysis of the algorithm.

Table 7.3 The basic steps of the block exact Householder QRD-RLS algorithm.

Block exact Householder QRD-RLS
Initialize $0 \ll \lambda < 1, \varepsilon \approx 1/\sigma_x^2$ and $\mathbf{U}(0) = 1/\sqrt{\varepsilon}\mathbf{I}$
for each k
{ Computing $\tilde{\mathbf{P}}(k)$ and updating $\mathbf{U}(k-1)$:
$\tilde{\mathbf{P}}(k) \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{H}} \\ \lambda^{Q/2}\mathbf{U}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix}$
Updating $\mathbf{d}_{q_2}(k-1)$:
$\tilde{\mathbf{P}}(k) \begin{bmatrix} \tilde{\mathbf{d}}_k^{*} \\ \lambda^{Q/2}\mathbf{d}_{q_2}(k-1) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{e}}_k \\ \mathbf{d}_{q_2}(k) \end{bmatrix}$
Obtaining $\mathbf{w}(k)$:
$\mathbf{w}(k) = \mathbf{U}^{-1}(k)\mathbf{d}_{q_2}(k)$
}

7.4 The Householder Block Exact Inverse QRD-RLS Algorithm

In the block exact QRD-RLS algorithm, the orthogonal matrix $\tilde{\mathbf{P}}(k)$ and the updated Cholesky factor $\mathbf{U}(k)$ are jointly obtained from (7.56). Then, the inverse of $\mathbf{U}(k)$ need to be computed in order to extract the LS weight vector according to (7.58). Clearly, it would be more convenient to be able to work directly with the inverse of the Cholesky factor. It is well known that if $\tilde{\mathbf{P}}(k)$ is orthogonal and satisfies (7.56), then

$$\tilde{\mathbf{P}}(k) \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \lambda^{-Q/2}\mathbf{U}^{-\mathrm{T}}(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{E}^{\mathrm{T}}(k) \\ \mathbf{U}^{-\mathrm{T}}(k) \end{bmatrix}, \tag{7.62}$$

where $\mathbf{E}(k)$ is an $(N+1)\times Q$ matrix. To see this, let us note that we can express an identity matrix as follows:

$$\begin{aligned} \mathbf{I} &= \begin{bmatrix} \mathbf{O}_{(N+1)\times Q} & \lambda^{-Q/2}\mathbf{U}^{-1}(k-1) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{T}} \\ \lambda^{Q/2}\mathbf{U}(k-1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{O}_{(N+1)\times Q} & \lambda^{-Q/2}\mathbf{U}^{-1}(k-1) \end{bmatrix} \tilde{\mathbf{P}}^{\mathrm{T}}(k)\tilde{\mathbf{P}}(k) \begin{bmatrix} \tilde{\mathbf{X}}_k^{\mathrm{T}} \\ \lambda^{Q/2}\mathbf{U}(k-1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E}(k) & \mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix} \end{aligned} \tag{7.63}$$

and thus $\mathbf{W} = \mathbf{U}^{-1}(k)$. From (7.62), the inverse Cholesky factor can be updated. However, in order to compute $\tilde{\mathbf{P}}(k)$ we have to resort to (7.56), which requires the Cholesky factor itself. To avoid using $\mathbf{U}(k-1)$ explicitly, the following lemma, which has been derived in [4], can be employed.

Lemma 1. *Let* $\mathbf{G}(k) = -\lambda^{-Q/2}\mathbf{U}^{-T}(k-1)\tilde{\mathbf{X}}_k$ *and let* $\hat{\mathbf{P}}(k)$ *be an orthogonal matrix such that*

$$\hat{\mathbf{P}}(k)\begin{bmatrix}\mathbf{I}_Q\\ \mathbf{G}(k)\end{bmatrix}=\begin{bmatrix}\boldsymbol{\Delta}(k)\\ \mathbf{O}_{(N+1)\times Q}\end{bmatrix}, \tag{7.64}$$

where $\mathbf{I}_Q$ *is the* $Q\times Q$ *identity matrix and* $\boldsymbol{\Delta}(k)$ *is a* $Q\times Q$ *matrix. Then*

$$\hat{\mathbf{P}}(k)\begin{bmatrix}\tilde{\mathbf{X}}_k^T\\ \lambda^{Q/2}\mathbf{U}(k-1)\end{bmatrix}=\begin{bmatrix}\mathbf{O}_{Q\times(N+1)}\\ \mathbf{V}\end{bmatrix}. \tag{7.65}$$

If $\mathbf{V}$ *is upper triangular, then* $\mathbf{V}=\mathbf{U}(k)$ *and*

$$\hat{\mathbf{P}}(k)\begin{bmatrix}\mathbf{O}_{Q\times(N+1)}\\ \lambda^{-Q/2}\mathbf{U}^{-T}(k-1)\end{bmatrix}=\begin{bmatrix}\mathbf{E}^T(k)\\ \mathbf{U}^{-T}(k)\end{bmatrix}, \tag{7.66}$$

where $\mathbf{E}(k)=\lambda^{-Q/2}\mathbf{U}^{-1}(k-1)\mathbf{G}(k)\boldsymbol{\Delta}^{-1}(k)$.

The crucial task now becomes to determine the orthogonal matrix in (7.64) so that the triangular structures are retained in (7.66). As a generalization of the procedure for $Q=1$ (inverse QRD-RLS algorithm of Chapter 3), $\hat{\mathbf{P}}(k)$ can be constructed as the product of $N+1$ orthogonal row Householder reflections [4], i.e,

$$\hat{\mathbf{P}}(k)=\hat{\mathbf{P}}_N(k)\hat{\mathbf{P}}_{N-1}(k)\cdots\hat{\mathbf{P}}_0(k). \tag{7.67}$$

The row Householder matrix $\hat{\mathbf{P}}_n(k), n=0,1,\ldots,N$, zeros the $(n+1)$-th row of $\mathbf{G}(k)$ as follows:

$$\hat{\mathbf{P}}_n(k)\begin{bmatrix}\boldsymbol{\Delta}_n(k)\\ \mathbf{0}^{\mathrm{T}}\\ \vdots\\ \mathbf{g}_n^{\mathrm{T}}(k)\\ \mathbf{g}_{n+1}^{\mathrm{T}}(k)\\ \vdots\\ \mathbf{g}_N^{\mathrm{T}}(k)\end{bmatrix}=\begin{bmatrix}\boldsymbol{\Delta}_{n+1}(k)\\ \mathbf{0}^{\mathrm{T}}\\ \vdots\\ \mathbf{0}^{\mathrm{T}}\\ \mathbf{g}_{n+1}^{\mathrm{T}}(k)\\ \vdots\\ \mathbf{g}_N^{\mathrm{T}}(k)\end{bmatrix} \tag{7.68}$$

where $\boldsymbol{\Delta}_0(k)=\mathbf{I}_Q$, $\boldsymbol{\Delta}_N(k)=\boldsymbol{\Delta}(k)$, and $\mathbf{g}_n^{\mathrm{T}}(k)$ is the $(n+1)$-th row of $\mathbf{G}(k)$ for $n=0,1,\ldots,N$. This procedure is also graphically illustrated in Figure 7.5. From the analysis on row Householder matrices and (7.68), the defining vector of the row Householder matrix $\hat{\mathbf{P}}_n(k)$ is given by

$$\mathbf{y}_n(k)=\frac{1}{||\mathbf{g}_n(k)||}\begin{bmatrix}\boldsymbol{\Delta}_n^{-\mathrm{T}}(k)\mathbf{g}_n(k)\\ \mathbf{0}_n\\ -1-\sqrt{1+(\boldsymbol{\Delta}_n^{-\mathrm{T}}(k)\mathbf{g}_n(k))^{\mathrm{T}}\boldsymbol{\Delta}_n^{-\mathrm{T}}(k)\mathbf{g}_n(k)}\\ \mathbf{0}_{N-n}\end{bmatrix}. \tag{7.69}$$

Recall that $\hat{\mathbf{P}}_n(k)$ need not be explicitly constructed, but, instead, $\mathbf{y}_n(k)$ is computed from (7.69) and is properly used in (7.68). Thus, the orthogonal matrix $\hat{\mathbf{P}}(k)$ can be constructed from (7.67), (7.68), and (7.69) and then applied in (7.66) in order to

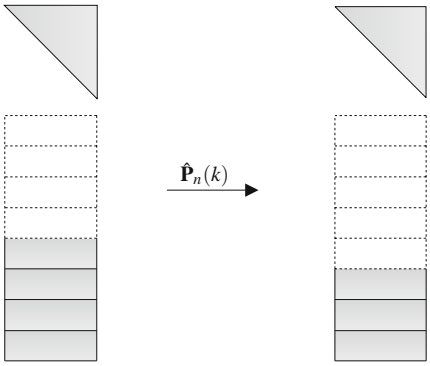

Fig. 7.5 Application of row Householder transformations in the block exact inverse QRD-RLS algorithm.

update directly the inverse Cholesky factor. Furthermore, the LS weight vector can be efficiently updated using the following block recursive formula [4]:

$$\mathbf{w}(k) = \mathbf{w}(k-1) - \mathbf{E}(k)\boldsymbol{\Delta}^{-\mathrm{T}}(k)\left[\tilde{\mathbf{d}}_k - \tilde{\mathbf{X}}_k^{\mathrm{T}}\mathbf{w}(k-1)\right], \tag{7.70}$$

where $\mathbf{w}(k)$ refers to the exponentially weighted LS solution at time $(k+1)Q-1$.

The Householder block exact inverse QRD-RLS algorithm is summarized in Table 7.4.

As mentioned before, in the second step of the algorithm, inversion of the $Q \times Q$ matrices $\boldsymbol{\Delta}_n(k)$ is required. Note, however, that in most applications that employ a block-type recursive scheme, the block length Q is taken to be much smaller than the filter size N [18]. Therefore, the overall burden in complexity, which is due to these inverse matrix calculations, is not considerable.

Table 7.4 The basic steps of the block exact Householder inverse QRD-RLS algorithm.

Block exact Householder inverse QRD-RLS
Initialize $0 \ll \lambda < 1, \mathbf{w}(0) = \mathbf{0}, \varepsilon \approx 1/\sigma_x^2$ and $\mathbf{U}^{-1}(0) = \sqrt{\varepsilon}\mathbf{I}$
for each k
{ Computing $\mathbf{G}(k)$:
$\mathbf{G}(k) = -\lambda^{-Q/2}\mathbf{U}^{-\mathrm{H}}(k-1)\tilde{\mathbf{X}}_k$
Computing $\hat{\mathbf{P}}(k)$ and $\boldsymbol{\Delta}(k)$:
$\hat{\mathbf{P}}(k)\begin{bmatrix}\mathbf{I}_Q \\ \mathbf{G}(k)\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Delta}(k) \\ \mathbf{O}_{(N+1)\times Q}\end{bmatrix}$
Updating $\mathbf{U}^{-1}(k-1)$:
$\hat{\mathbf{P}}(k)\begin{bmatrix}\mathbf{O}_{Q\times(N+1)} \\ \lambda^{-Q/2}\mathbf{U}^{-\mathrm{H}}(k-1)\end{bmatrix} = \begin{bmatrix}\mathbf{E}^{\mathrm{H}}(k) \\ \mathbf{U}^{-\mathrm{H}}(k)\end{bmatrix}$
Computing $\mathbf{w}(k)$:
$\mathbf{w}(k) = \mathbf{w}(k-1) - \mathbf{E}(k)\boldsymbol{\Delta}^{-\mathrm{H}}(k)\left[\tilde{\mathbf{d}}_k^* - \tilde{\mathbf{X}}_k^{\mathrm{H}}\mathbf{w}(k-1)\right]$
}

7.5 Sliding Window (SW) Householder Block Implementation

In the previous analysis, an exponentially weighted LS cost function has been considered, which is more frequently used in practice. In a time-varying environment, an alternative popular approach is to employ a *sliding* window (SW) on the data. To reduce complexity, data can be organized in blocks of size Q, and the LS weight vector can be updated on a block-by-block basis. In a block SW formulation, the LS error vector, whose norm is to be minimized, is given by

$$\mathbf{e}(k) = \begin{bmatrix} \mathbf{e}_k \\ \mathbf{e}_{k-1} \\ \vdots \\ \mathbf{e}_{k-L+1} \end{bmatrix} = \mathbf{d}(k) - \mathbf{X}(k)\mathbf{w}(k), \tag{7.71}$$

where L is the window size,

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{X}_k^{\mathrm{T}} \\ \mathbf{X}_{k-1}^{\mathrm{T}} \\ \vdots \\ \mathbf{X}_{k-L+1}^{\mathrm{T}} \end{bmatrix} \tag{7.72}$$

and

$$\mathbf{d}(k) = \begin{bmatrix} \mathbf{d}_k \\ \mathbf{d}_{k-1} \\ \vdots \\ \mathbf{d}_{k-L+1} \end{bmatrix}. \tag{7.73}$$

The $Q \times (N+1)$ input data matrices $\mathbf{X}_i^{\mathrm{T}}$ and the $Q \times 1$ desired response vectors $\mathbf{d}_i$, $i = k-L+1, \ldots, k$, are given by (7.47) and (7.48), respectively. Let us now define the following augmented quantities:

$$\bar{\mathbf{X}}(k) = \begin{bmatrix} \mathbf{X}(k) \\ \mathbf{X}_{k-L}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}_k^{\mathrm{T}} \\ \mathbf{X}(k-1) \end{bmatrix}, \text{ and} \tag{7.74}$$

$$\bar{\mathbf{d}}(k) = \begin{bmatrix} \mathbf{d}(k) \\ \mathbf{d}_{k-L} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_k \\ \mathbf{d}(k-1) \end{bmatrix}. \tag{7.75}$$

The LS problem can be solved recursively via a two-step procedure. This procedure comprises an update step and a downdate step, as described in the following representation:

$$\mathbf{w}(k-1) \longrightarrow \bar{\mathbf{w}}(k) \longrightarrow \mathbf{w}(k), \tag{7.76}$$

where $\bar{\mathbf{w}}(k)$ is the solution of the LS problem, which results by substituting in (7.71) $\mathbf{X}(k)$ and $\mathbf{d}(k)$ by $\bar{\mathbf{X}}(k)$ and $\bar{\mathbf{d}}(k)$, respectively. The update step can be implemented directly by using one of the algorithms described in Sections 7.3 and 7.4. For the downdate step, the following relation between the involved data correlation matrices holds:

$$\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) = \bar{\mathbf{X}}^{\mathrm{T}}(k)\bar{\mathbf{X}}(k) - \mathbf{X}_{k-L}\mathbf{X}_{k-L}^{\mathrm{T}} \tag{7.77}$$

or

$$\mathbf{R}(k) = \bar{\mathbf{R}}(k) - \mathbf{X}_{k-L}\mathbf{X}_{k-L}^{\mathrm{T}}. \tag{7.78}$$

By expressing the correlation matrices in terms of their Cholesky factors, the last equation is rewritten as follows:

$$\mathbf{U}^{\mathrm{T}}(k)\mathbf{U}(k) = \bar{\mathbf{U}}^{\mathrm{T}}(k)\bar{\mathbf{U}}(k) - \mathbf{X}_{k-L}\mathbf{X}_{k-L}^{\mathrm{T}}. \tag{7.79}$$

In [3], it is shown that there exists a hypernormal matrix $\mathbf{H}(k)$ with respect to the signature

$$\boldsymbol{\Phi} = \begin{bmatrix} -\mathbf{I}_Q & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_{N+1} \end{bmatrix} \tag{7.80}$$

such that

$$\mathbf{H}(k)\begin{bmatrix} \mathbf{X}_{k-L}^{\mathrm{T}} \\ \bar{\mathbf{U}}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}(k) \end{bmatrix}. \tag{7.81}$$

Matrix $\mathbf{H}(k)$ can be constructed as the product of $N+1$ hyperbolic Householder matrices, which annihilate the columns of $\mathbf{X}_{k-L}^{\mathrm{T}}$ with respect to the diagonal elements of $\bar{\mathbf{U}}(k)$. Note that $\bar{\mathbf{U}}(k)$ in (7.81) has been obtained from the initial update step, and thus (7.81) provides the Cholesky factor of the SW RLS problem at time k. However, in order to compute the required LS solution $\mathbf{w}(k)$, we have to resort to the inverse Cholesky factor. It is easily shown that $\mathbf{H}(k)$ can also be used to compute $\mathbf{U}^{-1}(k)$ from $\bar{\mathbf{U}}^{-1}(k)$ as in [4]

$$\mathbf{H}(k)\begin{bmatrix} \mathbf{O}_{Q\times(N+1)} \\ \bar{\mathbf{U}}^{-T}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{F}^{\mathrm{T}}(k) \\ \mathbf{U}^{-T}(k) \end{bmatrix}, \tag{7.82}$$

where $\mathbf{F}(k)$ is an $(N+1)\times Q$ matrix. To compute $\mathbf{H}(k)$, a result similar to that of Lemma 1 can be employed [4]. More specifically, by defining the vector $\bar{\mathbf{G}}(k) = -\bar{\mathbf{U}}^{-\mathrm{T}}(k)\mathbf{X}_{k-L}$, a hypernormal matrix $\mathbf{H}(k)$ such that

$$\mathbf{H}(k)\begin{bmatrix} \mathbf{I}_Q \\ \bar{\mathbf{G}}(k) \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{\Delta}}(k) \\ \mathbf{O}_{(N+1)\times Q} \end{bmatrix}, \tag{7.83}$$

where $\bar{\boldsymbol{\Delta}}(k)$ is a $Q\times Q$ matrix, also satisfies (7.82). To retain the triangular structure of the matrices in (7.82), $\mathbf{H}(k)$ is constructed as a sequence of $N+1$ row hyperbolic Householder reflections $\mathbf{H}_n(k)$, $n=0,1,\ldots,N$, with respect to the signature $\boldsymbol{\Phi}$ given in (7.80). $\mathbf{H}_n(k)$ annihilates the $(n+1)$-th row of $\bar{\mathbf{G}}(k)$ with respect to the upper $Q\times Q$ block of the matrix on the left-hand-side of (7.83). Thus, from the analysis on row Householder matrices and (7.80), its defining vector can be written as

Table 7.5 The basic steps of the sliding window (SW) block HRLS algorithm.

SW block Householder RLS
Run the update step L times to obtain $\mathbf{w}(L)$ and $\mathbf{U}^{-1}(L)$ for each $k > L$ { **Update Step** Computing $\mathbf{G}(k)$: $\mathbf{G}(k) = -\mathbf{U}^{-\mathrm{H}}(k-1)\mathbf{X}_k$ Computing $\mathbf{Q}(k)$: $\mathbf{Q}(k)\begin{bmatrix}\mathbf{I}_Q \\ \mathbf{G}(k)\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Delta}(k) \\ \mathbf{O}_{(N+1)\times Q}\end{bmatrix}$ Updating $\mathbf{U}^{-1}(k-1)$: $\mathbf{Q}(k)\begin{bmatrix}\mathbf{O}_{Q\times(N+1)} \\ \mathbf{U}^{-\mathrm{H}}(k-1)\end{bmatrix} = \begin{bmatrix}\mathbf{E}^{\mathrm{H}}(k) \\ \tilde{\mathbf{U}}^{-\mathrm{H}}(k)\end{bmatrix}$ Computing $\bar{\mathbf{w}}(k)$: $\bar{\mathbf{w}}(k) = \mathbf{w}(k-1) - \mathbf{E}(k)\boldsymbol{\Delta}^{-\mathrm{H}}(k)\left[\mathbf{d}_k^* - \mathbf{X}_k^{\mathrm{H}}\mathbf{w}(k-1)\right]$ **Downdate Step** Computing $\bar{\mathbf{G}}(k)$: $\bar{\mathbf{G}}(k) = -\tilde{\mathbf{U}}^{-\mathrm{H}}(k)\mathbf{X}_{k-L}$ Computing $\mathbf{H}(k)$: $\mathbf{H}(k)\begin{bmatrix}\mathbf{I}_Q \\ \bar{\mathbf{G}}(k)\end{bmatrix} = \begin{bmatrix}\bar{\boldsymbol{\Delta}}(k) \\ \mathbf{O}_{(N+1)\times Q}\end{bmatrix}$ Downdating $\tilde{\mathbf{U}}^{-1}(k)$: $\mathbf{H}(k)\begin{bmatrix}\mathbf{O}_{Q\times(N+1)} \\ \tilde{\mathbf{U}}^{-\mathrm{H}}(k)\end{bmatrix} = \begin{bmatrix}\mathbf{F}^{\mathrm{H}}(k) \\ \mathbf{U}^{-\mathrm{H}}(k)\end{bmatrix}$ **Computing** $\mathbf{w}(k)$: $\mathbf{w}(k) = \bar{\mathbf{w}}(k) - \mathbf{F}(k)\bar{\boldsymbol{\Delta}}^{-\mathrm{H}}(k)\left[\mathbf{d}_{k-L}^* - \mathbf{X}_{k-L}^{\mathrm{H}}\bar{\mathbf{w}}(k)\right]$ }

$$\bar{\mathbf{y}}_n(k) = \frac{1}{||\bar{\mathbf{g}}_n(k)||}\begin{bmatrix} \bar{\boldsymbol{\Delta}}_n^{-\mathrm{T}}(k)\bar{\mathbf{g}}_n(k) \\ \mathbf{0}_n \\ -1-\sqrt{1-(\bar{\boldsymbol{\Delta}}_n^{-\mathrm{T}}(k)\bar{\mathbf{g}}_n(k))^{\mathrm{T}}\bar{\boldsymbol{\Delta}}_n^{-\mathrm{T}}(k)\bar{\mathbf{g}}_n(k)} \\ \mathbf{0}_{N-n}\end{bmatrix}, \tag{7.84}$$

where $\bar{\boldsymbol{\Delta}}_0(k) = \mathbf{I}_Q$, $\bar{\boldsymbol{\Delta}}_N(k) = \bar{\boldsymbol{\Delta}}(k)$, and $\bar{\mathbf{g}}_n^{\mathrm{T}}(k)$ is the $(n+1)$-th row of $\bar{\mathbf{G}}(k)$ for $n = 0, 1, \ldots, N$. Furthermore, the LS weight vector is computed according to the following recursive formula [4]:

$$\mathbf{w}(k) = \bar{\mathbf{w}}(k) + \mathbf{F}(k)\bar{\boldsymbol{\Delta}}^{-\mathrm{T}}(k)(\mathbf{d}_{k-L} - \mathbf{X}_{k-L}^{\mathrm{T}}\bar{\mathbf{w}}(k)). \tag{7.85}$$

The SW block HRLS algorithm is summarized in Table 7.5. The algorithm consists of an update step, implemented with the block inverse QRD-RLS algorithm using a row orthogonal Householder matrix $\mathbf{Q}(k)$, and a downdate step as described in the previous analysis. Note that in the initialization phase the update step is executed L times, before the algorithm switches to the two-step procedure.

7.6 Conclusion

The aim of the chapter was to present various RLS algorithms, whose recursion procedure is based on Householder transforms. Householder transforms are known to possess exceptional numerical properties, and thus the resulting RLS schemes exhibit numerical robustness in finite-precision environments. The HRLS algorithm was first presented, which updates in time an arbitrary square-root of the data correlation matrix using an orthogonal Householder transformation. The algorithm is particularly attractive in several applications, due to its low-computational complexity and numerical robustness. In the sequel, three other Householder-based RLS algorithms have been described. The common characteristic of these schemes is that the weight vector is updated on a block-by-block basis, which calls directly for the application of Householder reflections instead of Givens rotations. The first two algorithms can be considered as generalizations of the conventional sample-by-sample QRD-RLS and inverse QRD-RLS algorithms, respectively. The block exact QRD-RLS scheme provides the per block update of the data correlation matrix Cholesky factor, with the application of a sequence of orthogonal Householder matrices. The block exact inverse QRD-RLS algorithm block updates the inverse Cholesky factor through a sequence of row orthogonal Householder matrices. The third algorithm was a SW block RLS scheme, which also manipulates the inverse Cholesky factor. The algorithm provides the respective LS weight vector by utilizing two Householder transformations, namely a row orthogonal and a row hyperbolic, in order to update and downdate the inverse Cholesky factor, respectively.

References

1. A. O. Householder, The Theory of Matrices in Numerical Analysis. Dover Publications Inc., New York, NY, USA (1964)
2. A. O. Steinhardt, Householder transforms in signal processing. IEEE Signal Processing Magazine, vol. 5, pp. 4–12 (July 1988)
3. C. M. Rader and A. O. Steinhardt, Hyperbolic Householder transformations. IEEE Transactions on Acoustics Speech and Signal Processing, vol. 34, no. 6, pp. 1859–1602 (December 1986)
4. A. W. Bojanczyk, G. G. Nagy and R. J. Plemmons, Block RLS using row Householder reflections. Linear Algebra and its Applications, vol. 188–189, pp. 31–61 (1993)
5. S. Haykin, Adaptive Filter Theory. 4th edition, Prentice-Hall Inc., Upper Saddle River, NJ, USA (2002)
6. A. A. Rontogiannis and S. Theodoridis, On inverse factorization adaptive least squares algorithms. Signal Processing (Elsevier), vol. 52, pp. 35–47 (July 1996)
7. A. A. Rontogiannis and S. Theodoridis, An adaptive LS algorithm based on orthogonal Householder transformations. IEEE International Conference on Electronics, Circuits and Systems, ICECS'96, Rhodes, Greece, pp. 860–863 (October 1996)
8. S. C. Douglas, Numerically robust $O(N^2)$ RLS algorithms using least-squares prewhitening. IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '2000, Istanbul, Turkey (June 2000)
9. S. C. Douglas and R. Losada, Adaptive filters in Matlab: from novice to expert. IEEE Signal Processing Education Workshop, Pine Mountain, USA, pp. 168–173 (October 2002)

10. A. H. Sayed and T. Kailath, A state-space approach to adaptive RLS filtering. IEEE Signal Processing Magazine, vol. 11, pp. 18–60 (July 1994)
11. G. J. Bierman, Factorization Methods for Discrete Sequential Estimation. Academic Press, New York, NY, USA (1977)
12. G. J. Bierman and C. L. Thronton, Numerical comparison of Kalman filter algorithms: orbit determination case study. Automatica, vol. 13, pp. 13–25 (January 1977)
13. R. Batin, Astronautical Guidance. McGraw-Hill Book Company, New York, NY, USA (1964)
14. L. A. McGee and S. F. Schmidt, Discovery of the Kalman filter as a practical tool for aerospace and industry. NASA Technical Memorandum 86847, USA (November 1985)
15. S. C. Douglas, Blind source separation and independent component analysis: a crossroads of tools and ideas. Fourth International Symposium on Independent Component Analysis and Blind Source Separation, Napa, Japan (April 2003)
16. K. Hooli, M. Latva-aho and M. Juntti, Performance evaluation of adaptive chip-level channel equalizers in WCDMA downlink. IEEE International Conference on Communications, ICC'2001, Helsinki, Finland (June 2001)
17. K. R. Liu, S.-F. Hsieh and K. Yao, Systolic block Householder transformation for RLS algorithm with two-level pipelined implementation. IEEE Transactions on Signal Processing, vol. 40, no. 4, pp. 946–958 (April 1998)
18. G.-O. Glentis, K. Berberidis and S. Theodoridis, Efficient least squares adaptive algorithms for FIR transversal filtering. IEEE Signal Processing Magazine, vol. 16, pp. 13–41 (July 1999)