Chapter 5
QRD Least-Squares Lattice Algorithms

Jenq-Tay Yuan

Abstract This chapter presents a full derivation of the square-root-free (SRF) QR-decomposition-based least-squares lattice (QRD-LSL) algorithms in complex form, based on linear interpolation (or two-sided prediction) theory as a generalization of linear prediction theory. The conventionally adopted QRD-LSL prediction algorithm can be derived directly from the QRD-LSL interpolation algorithm and then extended to solve the joint process estimation problem. The QRD-LSL interpolation algorithm that produces interpolation errors (residuals) of various orders may have potential implications for some signal processing and communication problems. Interestingly, the QRD-LSL interpolation algorithm can also be used to calculate the Kalman gain vector to implement the widely known recursive least-squares (RLS) algorithm in a transversal structure to generate the least-squares filter weights at each time step. Therefore, linear interpolation theory may provide a bridge between lattice filters and transversal filters. The chapter is organized as follows. Section 5.1 presents the fundamentals of QRD-LSL algorithms. The LSL interpolator and the LSL predictor are briefly presented in Section 5.2. Section 5.3 presents the SRF Givens rotation with feedback mechanism that is employed to develop the SRF QRD-LSL algorithms. In Section 5.4, the SRF QRD-LSL interpolation algorithm is derived, and then reduced to the SRF QRD-LSL prediction algorithm, which is then extended to develop the SRF joint process estimation. The RLS algorithm in the transversal structure based on the SRF QRD-LSL interpolation algorithm is presented in Section 5.5 followed by some simulation results in Section 5.6. Section 5.7 draws conclusions.

Jenq-Tay Yuan
Fu Jen Catholic University, Taipei, Taiwan – R.O.C.
e-mail: `yuan@ee.fju.edu.tw`

5.1 Fundamentals of QRD-LSL Algorithms

An Nth-*stage lattice filter*, displayed in Figure 5.1, automatically generates all N of the outputs that would be provided by N separate transversal filters of length $1,2,\ldots,N$ [1–6]. In Figure 5.1, $\varepsilon_{f,m}(k)$ and $\varepsilon_{b,m}(k)$ are the forward and backward prediction errors of order m; $\pi_{f,1}(k),\ldots,\pi_{f,N}(k)$ and $\pi_{b,1}(k),\ldots,\pi_{b,N}(k)$ are the forward reflection coefficients and backward reflection coefficients, respectively, which can be obtained by minimizing the sum of weighted prediction error squares at each stage. Optimum higher order lattice filters can be constructed from lower order ones by simply adding more lattice stages, leaving the original stages unchanged. This property is called the *order-recursive property* (or decoupling property) and follows from the fact that lattice filters essentially perform the Gram–Schmidt orthogonalization recursively such that each stage as effectively as possible decorrelates (or orthogonalizes) the inputs that enter it. This order-recursive property of a lattice filter along with its good numerical property is also shared by the QR-decomposition (QRD)-based algorithms. Accordingly, a combination of both the QRD-based algorithm and the least-squares lattice (LSL) filter can be reasonably expected to be a powerful algorithm, known as the *QR-decomposition-based least-squares lattice* (QRD-LSL) *algorithm*, for solving the adaptive least-squares (LS) filtering problem when the corresponding filter weights are not required. The order-recursive property of the QRD-LSL algorithms allows a variable-length filter to be designed, since it permits dynamic assignment, and rapid automatic determination of the most effective filter length. Consequently, in many LS applications, such as acoustic echo cancelation [7], in which only the LS errors of various orders are required to reduce

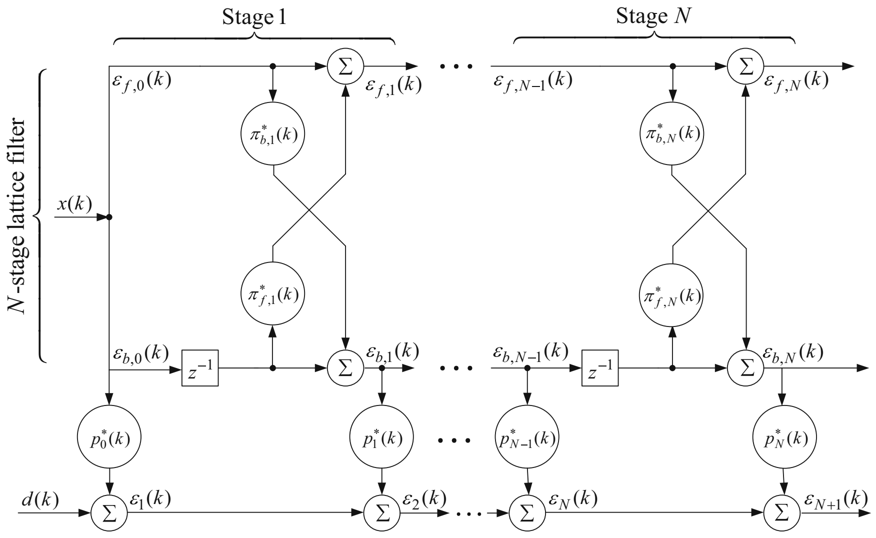

Fig. 5.1 Joint process estimation based on an Nth-*stage* lattice filter.

effectively near-end speech distortion and improve noise robustness without explicitly computing the corresponding filter weights, the QRD-LSL prediction algorithm along with the joint process estimation may play an important role. Even when the corresponding filter weights are required in applications such as system identification, the QRD-LSL interpolation algorithm that is developed in this chapter can still be applied to implement efficiently the well-known *recursive least-squares* (RLS) *algorithm*.

A recursive fast QRD-based LS filtering algorithm (or known as the QRD-based fast Kalman algorithm) of $\mathscr{O}[N]$ complexity was developed by Cioffi [8], where N is the number of taps in the adaptive filter. Although this algorithm is of a fixed-order transversal filter type and thus lacks the flexibility of order recursiveness, it presents for the first time the idea of a fast QRD-based algorithm for RLS estimation. Proudler et al. [9] developed a QRD-LSL prediction algorithm of $\mathscr{O}[N]$ complexity recursively in time and order. They then extended this QRD-LSL prediction algorithm to solve the joint process estimation problem. A similar fast LS lattice algorithm based on Givens rotations was developed independently by Ling [10]. Regalia and Bellanger [11] developed a hybrid QR-lattice algorithm that is closely related to the algorithms of Cioffi [8], Proudler et al. [9], and Ling [10], but they emphasized the relationship between fast QR algorithms and lattice filters. Other related works can also be found in Rontogiannis and Theodoridis [12], who proposed a unified approach for deriving fast QRD algorithms. The QRD-LSL algorithm is now well-known to provide many desirable features such as computational efficiency, fast rate of convergence and fast tracking capability, robustness against and insensitivity to round-off noise, and suitability for very large scale integration (VLSI) implementations. According to the simulations conducted by Yang and Bohme [13], the joint process estimation based on the QRD-LSL prediction algorithm is numerically more stable than the fixed order fast QR algorithms such as those developed by Cioffi [8] and Regalia and Bellanger [11]. The unstable behaviors exhibited by some fast Kalman algorithms may further justify the use of the order-recursive QRD-LSL algorithms.

Almost all of the QRD-LSL algorithms discussed in the literature are designed to solve the linear prediction problem recursively in time and order. This chapter addresses the QRD-LSL problem differently, and presents a complete derivation of the square-root-free (SRF) QRD-LSL algorithms from the perspective of linear interpolation. The derivation of the QRD-LSL interpolation algorithm is motivated by three main considerations. First, the QRD-LSL interpolation algorithm has potential implications for signal processing and communication problems, including data compression, coding and restoration of speech and images, and narrowband interference suppression in spread spectrum communications [14–18]. Secondly, the QRD-LSL prediction algorithm is in effect a special case of the QRD-LSL interpolation algorithm. As will be demonstrated in Section 5.4, the widely known QRD-LSL prediction algorithm can be developed directly from the QRD-LSL interpolation algorithm, because linear interpolation theory is a generalization of linear prediction theory and the former provides a broader interpretation and a more thorough understanding of the latter. Studies [19–22] have demonstrated that linear interpolation may substantially outperform linear prediction in terms of minimum mean squared error, because interpolation makes better use of the correlation between the nearest neighboring samples than does prediction. Thirdly, linear interpolation turns out to form a bridge between an order-recursive lattice filter that generates filter output only and the RLS algorithm, which also generates the adaptive filter weights. Skidmore and Proudler [23] first realized the crucial fact that the Kalman gain vector that is used to compute the RLS algorithm can be calculated as a particular set of normalized LS interpolation errors (residuals). The SRF QRD-LSL interpolation algorithm that produces interpolation errors of various orders can thus be adopted to implement the RLS algorithm. Although the resulting RLS algorithm of $\mathscr{O}[N\log_2 N]$ complexity may exhibit linear error growth with time in a limited-precision environment, it may offer a favorable compromise between some computationally efficient fast transversal filter algorithms of $\mathscr{O}[N]$ complexity, which may exhibit exponential error growth with time in a limited-precision environment, and some numerically stable algorithms of $\mathscr{O}[N^2]$ complexity, which may not be computationally feasible for some real-time applications.

5.2 LSL Interpolator and LSL Predictor

Linear interpolation estimates an unknown data sample based on a weighted sum of surrounding data samples. In one-dimensional signal processing, (p,f)th-*order* linear interpolation is the linear estimation of present input data samples $x(i)$ from its p past and f future neighboring data samples with the pre-windowing condition on the data (i.e., $x(i)=0$, for $i\leq -1$), which is

$$\widehat{x}_{p,f}(i) = -\sum_{\substack{n=-p \\ n\neq 0}}^{f} b^*_{(p,f),n}(k-f)x(i+n), \qquad -f\leq i\leq k-f, \tag{5.1}$$

where $b_{(p,f),n}(k-f)$ is the interpolation coefficient at time k. The (p,f)th-*order* interpolation error at each time unit can thus be written as

$$\varepsilon^I_{p,f}(i) = x(i)-\widehat{x}_{p,f}(i) = \mathbf{b}^{\mathrm{H}}_{p,f}(k-f)\mathbf{x}_{N+1}(i+f), \qquad -f\leq i\leq k-f, \tag{5.2}$$

where $\mathbf{b}^{\mathrm{H}}_{p,f}(k-f) = [b^*_{(p,f),f}(k-f),\ldots,b^*_{(p,f),1}(k-f),1,b^*_{(p,f),-1}(k-f),\ldots,b^*_{(p,f),-p}(k-f)]$ is the interpolation coefficient vector at time k and the $(N+1)\times 1$ input vector is given as $\mathbf{x}_{N+1}(i)=[x(i),x(i-1),\ldots,x(i-N)]^{\mathrm{T}}$ in which $N=p+f$ is assumed implicitly. Notably, the notations used in this chapter are somewhat different from those used in the four previous chapters. In this chapter, we refer to any interpolation filter that operates on the present data sample as well as p past and f future data samples to produce the (p,f)th-*order* interpolation errors at its output as a (p,f)th-*order interpolator*. If the most recent data sample used is $x(k)$, then (5.2) can be written in a matrix form as

$$\boldsymbol{\varepsilon}^{I}_{p,f}(k-f) = \mathbf{X}_{N+1}(k)\mathbf{b}_{p,f}(k-f), \tag{5.3}$$

where $\boldsymbol{\varepsilon}^{I}_{p,f}(k-f) = \left[\, \varepsilon^{I^*}_{p,f}(-f)\ \ldots\ \varepsilon^{I^*}_{p,f}(-1)\ \varepsilon^{I^*}_{p,f}(0)\ \varepsilon^{I^*}_{p,f}(1)\ \ldots\ \varepsilon^{I^*}_{p,f}(k-f) \,\right]^{\mathrm{T}}$, $\mathbf{b}_{p,f}(k-f) =$ $\left[\, b_{(p,f),f}(k-f)\ \ldots\ b_{(p,f),1}(k-f)\ 1\ b_{(p,f),-1}(k-f)\ \ldots\ b_{(p,f),-p}(k-f) \,\right]^{\mathrm{T}}$, and the $(k+1) \times (N+1)$ matrix $\mathbf{X}_{N+1}(k)$ can be written as

$$\left[\underbrace{\begin{matrix} x^*(0) & 0 & \cdots & 0 \\ x^*(1) & x^*(0) & \ddots & \vdots \\ x^*(2) & x^*(1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \vdots & \vdots & \ddots & x^*(0) \\ \vdots & \vdots & \ddots & x^*(1) \\ \vdots & \vdots & \ddots & x^*(2) \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x^*(k) & x^*(k-1) & \cdots & x^*(k-f+1) \end{matrix}}_{f \text{ future data samples}} \quad \underbrace{\begin{matrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ x^*(0) \\ x^*(1) \\ \vdots \\ \vdots \\ \vdots \\ x^*(k-f) \end{matrix}}_{\substack{\text{data sample}\\ \text{to be estimated}}} \quad \underbrace{\begin{matrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ 0 & \ddots & \vdots \\ x^*(0) & \ddots & \vdots \\ \vdots & \ddots & 0 \\ \vdots & \ddots & x^*(0) \\ \vdots & \ddots & \vdots \\ x^*(k-f-1) & \cdots & x^*(k-N) \end{matrix}}_{\substack{p \text{ past}\\ \text{data samples}}}\right].$$

5.2.1 LSL interpolator

The LS solution of the interpolation coefficients can be determined by minimizing the sum of interpolation error squares $\xi(k) = \sum_{i=-f}^{k-f} |\varepsilon^{I}_{p,f}(i)|^2 = \left[\boldsymbol{\varepsilon}^{I}_{p,f}(k-f)\right]^{\mathrm{H}} \boldsymbol{\varepsilon}^{I}_{p,f}(k-f) = \left[\mathbf{X}_{N+1}(k)\mathbf{b}_{p,f}(k-f)\right]^{\mathrm{H}} \left[\mathbf{X}_{N+1}(k)\mathbf{b}_{p,f}(k-f)\right] = \mathbf{b}^{\mathrm{H}}_{p,f}(k-f)\mathbf{R}(k)\mathbf{b}_{p,f}(k-f)$ subject to the constraint $\mathbf{h}^{\mathrm{H}}\mathbf{b}_{p,f}(k-f) = 1$, where $\mathbf{R}(k) = \mathbf{X}^{\mathrm{H}}_{N+1}(k)\mathbf{X}_{N+1}(k)$ is the $(N+1) \times (N+1)$ time-average correlation matrix of the input data samples $x(i)$ and $\mathbf{h}^{\mathrm{H}} = \left[\, \mathbf{0}^{\mathrm{T}}_{f}\ 1\ \mathbf{0}^{\mathrm{T}}_{p} \,\right]$. Using the method of Lagrange, the interpolation coefficient vector can be computed to be $\mathbf{b}_{p,f}(k-f) = \left(\mathbf{h}^{\mathrm{T}}\mathbf{R}^{-1}(k)\mathbf{h}\right)^{-1}\mathbf{R}^{-1}(k)\mathbf{h}$ and the minimum sum of interpolation error squares of order (p,f) can be computed to be $I_{p,f}(k-f) = \xi_{min}(k) = \left(\mathbf{h}^{\mathrm{T}}\mathbf{R}^{-1}(k)\mathbf{h}\right)^{-1}$. The resulting augmented asymmetric interpolation normal equations can thus be written as

$$\mathbf{R}(k)\mathbf{b}_{p,f}(k-f) = \mathbf{i}_{p,f}(k-f), \tag{5.4}$$

where $\mathbf{i}_{p,f}(k-f) = [\,\mathbf{0}^{\mathrm{T}}_{f}\ I_{p,f}(k-f)\ \mathbf{0}^{\mathrm{T}}_{p}\,]^{\mathrm{T}}$ is the $(N+1) \times 1$ vector in which $\mathbf{0}_f$ and $\mathbf{0}_p$ are column vectors of f and p zeros, respectively. A computationally efficient

LSL interpolator developed in [22] requiring only $\mathscr{O}[N]$ operations via "intermediate predictions" can be employed to compute the exact order-updated interpolation errors.

The LSL interpolator computes the order-recursive interpolation errors as an additional *past* data sample [i.e., $(p,f) \rightarrow (p+1,f)$] and an additional *future* data sample [i.e., $(p,f) \rightarrow (p,f+1)$] are taken into account, respectively, as follows:

$$\varepsilon^I_{p+1,f}(k-f) = \varepsilon^I_{p,f}(k-f) - k^*_{p+1,f}(k)\varepsilon_{b,N+1}(k,k-f), \tag{5.5}$$

$$\varepsilon^I_{p,f+1}(k-f-1) = \varepsilon^I_{p,f}(k-f-1) - k^*_{p,f+1}(k)\varepsilon_{f,N+1}(k,k-f-1), \tag{5.6}$$

where both $k_{p+1,f}(k)$ and $k_{p,f+1}(k)$ are the complex-valued coefficients of the interpolator;

$$\varepsilon_{f,N+1}(i,i-f-1) = x(i) + \sum_{\substack{n=1\\ n\neq f+1}}^{N+1} a^*_{N+1,n}(k)x(i-n), \quad 0 \le i \le k \tag{5.7}$$

is referred to as the $(N+1)$th-*order* intermediate forward prediction (IFP) error, as it is the prediction error of $x(i)$ based on a weighted linear combination of its $(N+1)$ previous data samples [i.e., $x(i-1),\ldots,x(i-f),x(i-f-2),\ldots,x(i-N-1)$] without considering present data sample $x(i-f-1)$, where $a_{N+1,n}(k), n = 1,2,\ldots,f,f+2,\ldots,N+1$ are $(N+1)$th-*order* IFP coefficients. Similarly, the $(N+1)$th-*order* intermediate backward prediction (IBP) error is defined as the prediction error of $x(i-N-1)$ based on a weighted linear combination of $x(i-N),x(i-N+1),\ldots,x(i-f-1),x(i-f+1),\ldots,x(i)$ without considering the data sample $x(i-f)$:

$$\varepsilon_{b,N+1}(i,i-f) = x(i-N-1) + \sum_{\substack{n=1\\ n\neq p+1}}^{N+1} c^*_{N+1,n}(k)x(i+n-N-1), \;\; 0 \le i \le k, \tag{5.8}$$

where $c_{N+1,n}(k)$, $n = 1,2,\ldots,p,p+2,\ldots,N+1$ are $(N+1)$th-*order* intermediate backward prediction coefficients. It is worth pointing out that two special cases arise when $f=0$ and when $p=N$, and, as a result, the Nth-*order* IFP error, $\varepsilon_{f,N}(k,k-f)$, is reduced to $\varepsilon_{f,N}(k)$ and $\varepsilon_{f,N-1}(k)$, which are the conventional forward prediction (FP) errors of order N and order $(N-1)$, respectively. Similarly, when $p=0$ and $f=N$, the Nth-*order* intermediate backward prediction error, $\varepsilon_{b,N}(k,k-f)$, is reduced to $\varepsilon_{b,N-1}(k-1)$ and $\varepsilon_{b,N}(k)$, which are the conventional backward prediction (BP) errors of order $(N-1)$ and order N, respectively.

5.2.2 Orthogonal bases for LSL interpolator

To construct an LSL interpolator of order (p, f), Equations (5.5) and (5.6) must be applied p and f times, respectively. However, any sequencing between these two equations is permissible. Consequently, an LSL interpolator of order (p, f) has $C_N^p = C_N^f = \frac{N!}{p!f!}$ permissible realizations. For instance, to construct a (2,2)th-*order* interpolator, a total of $C_4^2 = 6$ permissible realizations may be identified by the sequences BFBF, FBFB, BBFF, FFBB, FBBF, and BFFB of intermediate backward (B) and intermediate forward (F) prediction errors that are used in (5.5) and (5.6), respectively. Six possible *order-recursive* realizations for a $(2,2)$th-*order* LSL interpolator can thus be constructed by employing the six *orthogonal bases*. As an example, the orthogonal basis identified by BFBF sequence may be verified as follows. The sequence BFBF reveals that, to estimate the data sample $x(k-2)$, the data sample immediately prior to $x(k-2)$(i.e., $x(k-3)$) is considered first corresponding to a B followed by the consideration of the data sample immediately future to $x(k-2)$ (i.e., $x(k-1)$) corresponding to an F. The above two operations are then followed by the consideration of one additional past data sample, $x(k-4)$, corresponding to a B, and one additional future data sample, $x(k)$, corresponding to an F. The intermediate prediction error basis, which is an orthogonal basis set, can therefore be generated by

$$\begin{aligned}
&\left[\varepsilon_{b,1}(k-2,k-2)\ \varepsilon_{f,2}(k-1,k-2)\ \varepsilon_{b,3}(k-1,k-2)\ \varepsilon_{f,4}(k,k-2)\ \varepsilon^{I}_{2,2}(k-2)\right]^{\mathrm{T}} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a^*_{2,2}(k) & 1 & 0 & 0 & 0 \\ c^*_{3,1}(k) & c^*_{3,3}(k) & 1 & 0 & 0 \\ a^*_{4,3}(k) & a^*_{4,1}(k) & a^*_{4,4}(k) & 1 & 0 \\ b^*_{(2,2),-1}(k-2) & b^*_{(2,2),1}(k-2) & b^*_{(2,2),-2}(k-2) & b^*_{(2,2),2}(k-2) & 1 \end{bmatrix} \begin{bmatrix} x(k-3) \\ x(k-1) \\ x(k-4) \\ x(k) \\ x(k-2) \end{bmatrix},
\end{aligned} \tag{5.9}$$

The above transformation is known as the Gram–Schmidt orthogonalization procedure. The other five orthogonal basis vectors identified by the sequences FBFB, BBFF, FFBB, FBBF, and BFFB, can be similarly verified. Each of the orthogonal bases identified by the six sequences can be used to construct an *order-recursive* realization for a $(2,2)$th-*order* LSL interpolator.

Both IFP and IBP errors must be computed before the *order-updated* interpolation errors in (5.5) and (5.6) can be computed. They can be computed using the conventional BP and FP errors, as follows.

$$\varepsilon_{b,N+1}(k,k-f) = \varepsilon_{b,N+1}(k) + l^*_{b,N+1}(k)\varepsilon^{I}_{p,f}(k-f), \tag{5.10}$$

$$\varepsilon_{f,N+1}(k,k-f-1) = \varepsilon_{f,N+1}(k) + l^*_{f,N+1}(k)\varepsilon^{I}_{p,f}(k-f-1), \tag{5.11}$$

where $l_{b,N+1}(k)$ and $l_{f,N+1}(k)$ are complex-valued coefficients; $\varepsilon_{b,N+1}(k)$ and $\varepsilon_{f,N+1}(k)$, which are conventional BP and FP errors, respectively, are directly accessible from an $(N+1)$th-*order LSL predictor* that can be embedded into an LSL interpolator.

Notably, both $\varepsilon^{I}_{p,f}(k-f)$ and $\varepsilon^{I}_{p,f}(k-f-1)$ are already computed from the previous interpolation lattice stage of the LSL interpolator. Equations (5.5), (5.6), (5.10), and (5.11) together with the well-known LSL predictor, constitute an *order-recursive LSL interpolator*.

5.2.3 LSL predictor

The well-known exact decoupling property of the LSL predictor can be readily demonstrated to be a special case of the orthogonal basis of the LSL interpolator corresponding to sequence BB…B. By setting $(p,f)=(N,0)$ (i.e., an Nth-*order* LSL predictor), there is one unique orthogonal basis set for use in an LSL interpolator of order $(N,0)$ (since $C^0_N=1$), which is $\left[\varepsilon_{b,1}(k,k),\varepsilon_{b,2}(k,k),\ldots,\varepsilon_{b,N}(k,k),\varepsilon^{I}_{N,0}(k)\right]$. This orthogonal basis set is clearly equivalent to $[\varepsilon_{b,0}(k-1),\varepsilon_{b,1}(k-1),\ldots,\varepsilon_{b,N-1}(k-1),\varepsilon_{f,N}(k)]$, which, in turn, corresponds to $[\varepsilon_{b,0}(k),\varepsilon_{b,1}(k),\ldots,\varepsilon_{b,N-1}(k)]$ that consists of a sequence of N uncorrelated backward prediction errors at all instants of time and forms a unique orthogonal basis set [5].

The widely known LSL predictor can also be derived directly from the LSL interpolator by setting $(p,f)=(N,0)$ and $(p,f)=(0,N)$ in (5.10) and (5.11), respectively, and yields

$$\varepsilon_{b,N+1}(k)=\varepsilon_{b,N}(k-1)-l^{*}_{b,N+1}(k)\varepsilon_{f,N}(k) \tag{5.12}$$

$$\varepsilon_{f,N+1}(k)=\varepsilon_{f,N}(k)-l^{*}_{f,N+1}(k)\varepsilon_{b,N}(k-1). \tag{5.13}$$

Notably, we have used the fact that $\varepsilon_{b,N+1}(k,k)=\varepsilon_{b,N}(k-1)$, $\varepsilon^{I}_{N,0}(k)=\varepsilon_{f,N}(k)$, $\varepsilon_{f,N+1}(k,k-N-1)=\varepsilon_{f,N}(k)$, and $\varepsilon^{I}_{0,N}(k-N-1)=\varepsilon_{b,N}(k-1)$. Interestingly, (5.12) and (5.13) can also be reduced directly from (5.6) and (5.5) by setting $(p,f)=(0,N)$ and $(p,f)=(N,0)$, respectively.

5.3 SRF Givens Rotation with Feedback Mechanism

Consider first a Givens rotation matrix given by $\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix}$, where the two parameters of the Givens rotation are the real cosine parameter c and the complex sine parameter s, such that $c^2 + |s|^2 = 1$. A Givens rotation used to zero out the element at the $(2,1)$ location is an elementary transformation of the form

$$\begin{bmatrix} c & s^* \\ -s & c \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_p \\ \beta_1 & \beta_2 & \dots & \beta_p \end{bmatrix} = \begin{bmatrix} \alpha_1' & \alpha_2' & \dots & \alpha_p' \\ 0 & \beta_2' & \dots & \beta_p' \end{bmatrix}, \tag{5.14}$$

where α_1 and α_1' are defined to be real and non-negative, whereas $\alpha_2 \dots \alpha_p$, $\alpha_2' \dots \alpha_p'$, $\beta_1 \dots \beta_p$, $\beta_2' \dots \beta_p'$ are all complex. Substituting $s = c\beta_1/\alpha_1$ into $c^2 + |s|^2 = 1$ yields $c = \alpha_1/\alpha_1'$, where $\alpha_1' \triangleq \sqrt{\alpha_1^2 + |\beta_1|^2}$. Accordingly,

$$c = \frac{\alpha_1}{\alpha_1'} \text{ and } s = \frac{\beta_1}{\alpha_1'}, \tag{5.15}$$

$$\left(\alpha_1'\right)^2 = \alpha_1^2 + |\beta_1|^2, \tag{5.16}$$

$$\left(\alpha_i'\right)^* = c\alpha_i^* + s\beta_i^*, \quad i = 2, \dots, p, \tag{5.17}$$

$$\left(\beta_i'\right)^* = c\beta_i^* - s^*\alpha_i^*, \quad i = 2, \dots, p. \tag{5.18}$$

Equations (5.15), (5.16), (5.17), and (5.18) summarize the square-root (SR) Givens rotation of a complex version since it requires a SR operation in the generic formulation. The SR operation may occupy a large area in a VLSI chip and many cycles may be required to complete such computations; consequently, the operation is slow. Proudler et al. [24] demonstrated that a finite-precision implementation of an SRF lattice algorithm achieved better numerical results than that of the conventional SR Givens rotation. Hsieh et al. [25] thus proposed a systematic way of generating a unified SRF Givens rotation to avoid the SR operation. In the remaining part of this section, the SRF Givens rotation developed in [25] is generalized to a complex form and extended to include a feedback mechanism, which is known to have a stabilizing effect when errors are made in the QRD-based RLS estimation, because of finite-precision effects [26, 27]. The generalized SRF Givens rotation with a feedback mechanism is then applied to develop the SRF QRD-LSL algorithms that are presented in Section 5.4.

By taking out a scaling factor from each row of the matrices on both sides of (5.14), the two rows, before and after the Givens rotation, are denoted, respectively, by

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_p \\ \beta_1 & \beta_2 & \dots & \beta_p \end{bmatrix} = \begin{bmatrix} \sqrt{k_a} & 0 \\ 0 & \sqrt{k_b} \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \dots & a_p \\ b_1 & b_2 & \dots & b_p \end{bmatrix} \tag{5.19}$$

and

$$\begin{bmatrix} \alpha_1' & \alpha_2' & \dots & \alpha_p' \\ 0 & \beta_2' & \dots & \beta_p' \end{bmatrix} = \begin{bmatrix} \sqrt{k_a'} & 0 \\ 0 & \sqrt{k_b'} \end{bmatrix} \begin{bmatrix} a_1' & a_2' & \dots & a_p' \\ 0 & b_2' & \dots & b_p' \end{bmatrix}, \tag{5.20}$$

where k_a, k_b, k_a', and k_b' are the scaling factors resulting in SRF operations, and α_i' and β_i' are the updated α_i and β_i when β_1 is zeroed out. Replacing $\alpha_i = \sqrt{k_a}a_i$, $\alpha_i' = \sqrt{k_a'}a_i'$, $\beta_i = \sqrt{k_b}b_i$, $i = 1,\dots,p$ and $\beta_i' = \sqrt{k_b'}b_i'$, $i = 2,\dots,p$, in (5.15), (5.16), (5.17), and (5.18) leads to $c = \sqrt{k_a}a_1/\alpha_1'$, $s = \sqrt{k_b}b_1/\alpha_1'$, $a_1' = \alpha_1'/\sqrt{k_a'}$, $a_i' = k_a a_1 a_i + k_b b_1^* b_i/\sqrt{k_a'}\alpha_1'$, $i = 2,\dots,p$, and $b_i' = \frac{\sqrt{k_a k_b}[a_1 b_i - b_1 a_i]}{\sqrt{k_b'}\alpha_1'}, i = 2,\dots,p$, where $\alpha_1' = \sqrt{k_a a_1^2 + k_b |b_1|^2}$. Clearly, if $k_a' = (\alpha_1')^2/\mu^2$ and $k_b' = k_a k_b/\nu^2 (\alpha_1')^2 = k_a k_b/\mu^2\nu^2 k_a'$ are chosen, then the computation of a_1', a_i', and b_i' can avoid SR operation; μ and ν are parameters to be determined later. Substituting the chosen k_a' and k_b' into c, s, a_1', a_i', and, b_i' shown above yields $c = a_1/\mu \cdot \sqrt{k_a/k_a'}$, $s = b_1/\mu \cdot \sqrt{k_b/k_a'}$, $a_1' = \mu$, $a_i' = (k_a a_1 a_i + k_b b_1^* b_i)/\mu k_a', i = 2,\dots,p$, and, $b_i' = \nu[a_1 b_i - b_1 a_i], i = 2,\dots,p$. Evidently, the SR operations in a_1', a_i', and b_i' are eliminated, regardless of the values μ and ν.

Throughout this section, $\mu = 1$ (or $a_1' = 1$), $\nu = 1$, and $a_1 = 1$ were set such that the SRF results would be consistent with those of Gentleman [28] and McWhirter [29]. Accordingly, we have

$$k_a' = k_a + k_b|b_1|^2, \tag{5.21}$$

$$k_b' = \frac{k_a k_b}{k_a'} = \overline{c} \cdot k_b, \tag{5.22}$$

$$a_i' = \overline{c}a_i + \overline{s}^* b_i, \quad i = 2,\dots,p, \tag{5.23}$$

$$b_i' = b_i - b_1 a_i, \quad i = 2,\dots,p, \tag{5.24}$$

where

$$\overline{c} = \frac{k_a}{k_a'} \text{ and } \overline{s} = b_1 \cdot \frac{k_b}{k_a'} \tag{5.25}$$

are the defined generalized rotational parameters. Since $k_a = \alpha_1^2$ and $|\beta_1|^2$ corresponds to $\beta_1 \cdot \beta_1^* = k_b|b_1|^2$, we have $k_a' = k_a + k_b|b_1|^2 = \alpha_1^2 + |\beta_1|^2 = (\alpha_1')^2$. Therefore,

$$\overline{c} = \frac{k_a}{k_a'} = \frac{k_a}{(\alpha_1')^2} = c^2 \tag{5.26}$$

and $s = \sqrt{\frac{k_b}{k_a'}} \cdot b_1 = \sqrt{b_1 \cdot \overline{s}}$. The SRF Givens rotation with feedback mechanism can be derived by substituting (5.24) into (5.23):

$$a_i' = a_i(\overline{c} + \overline{s}^* b_1) + \overline{s}^* b_i' = a_i + \overline{s}^* b_i', \tag{5.27}$$

which is obtained using $\bar{c} + \bar{s}^* b_1 = \frac{k_a}{k'_a} + \frac{k_b b_1^*}{k'_a} b_1 = 1$. For notational convenience, taking the complex conjugate of both sides of (5.24) and (5.27) yields

$$\left(b'_i\right)^* = b_i^* - b_1^* a_i^*, \quad i = 2, \ldots, p, \tag{5.28}$$

$$\left(a'_i\right)^* = a_i^* + \bar{s} \cdot \left(b'_i\right)^*, \quad i = 2, \ldots, p. \tag{5.29}$$

Equations (5.21), (5.22), (5.25), (5.28), and (5.29) summarize the complex version of the *SRF Givens rotation with a feedback mechanism.*

5.4 SRF QRD-LSL Algorithms

This section develops the SRF QRD-LSL interpolation algorithm and the SRF QRD-LSL prediction algorithm. The latter algorithm is then extended to develop the SRF joint process estimation that utilizes information from the prediction lattice filter to generate the LS filtering estimate. More specifically, the joint process estimation problem displayed in Figure 5.1 is the optimal LS estimation of a process $d(k)$, called the desired response, from a related process $x(k)$, called the observations [2, 5].

The SRF QRD-LSL interpolation algorithm comprises six blocks: (a) FP block and BP block (summarized in Table 5.1); (b) IFP block and interpolation [Int(F)] block (summarized in Table 5.2); (c) IBP block and interpolation [Int(P)] block (summarized in Table 5.2) [30]. Overall, the SRF QRD-LSL interpolation algorithm has $\mathscr{O}[N]$ computational complexity per iteration without any SR operation. Notably, both FP and BP blocks, which constitute the SRF QRD-LSL prediction algorithm, are portions of the SRF QRD-LSL interpolation algorithm and must be used to compute both the conventional forward and backward prediction errors of order m [i.e., $e_{f,m}(k)$ and $e_{b,m}(k)$].

The SRF QRD-LSL interpolation algorithm performs adaptive filtering recursively in order and time. As an additional "future" stage is increased [i.e., $(p, f) \rightarrow (p, f+1)$], then both the IFP block and the Int(F) block, which are the QRD implementations of (5.11) and (5.6), respectively, must be used to compute the *a priori* interpolation error, $e^I_{p,f+1}(k-f-1)$. As an additional "past" stage is increased [i.e., $(p, f) \rightarrow (p+1, f)$], then both the IBP block and the Int(P) block, which are the QRD implementations of (5.10) and (5.5), respectively, must be used to compute the *a priori* interpolation error, $e^I_{p+1,f}(k-f)$. Throughout the chapter, the terms "ε" and "e" represent the *a posteriori* and *a priori* versions of estimation errors, respectively, whereas the term "$\overline{\varepsilon}$" represents the "*angle-normalized*" estimation error.

Table 5.1 SRF QRD-LSL algorithm and joint process estimation.

SRF QRD-LSL prediction and filtering
Initialization:
$\overline{\pi}_{f,m}(-1) = \overline{\pi}_{b,m}(-1) = e_{b,m-1}(-1) = 0$, for order $m = 1,2,\ldots,N$,
$\overline{p}_{m-1}(-1) = 0$, for order $m = 1,2,\ldots,N+1$,
$B_m(-2) = B_m(-1) = F_m(-1) = \delta$, for order $m = 0,1\ldots,N$,
For $k = 0,1,2\ldots$, set $e_{f,0}(k) = e_{b,0}(k) = x(k)$, $e_0(k) = d(k)$
For $k = -1,0,1,2\ldots$, set $\gamma_0(k) = 1$
Prediction: For time $k = 0,1,\ldots$, and prediction order $m = 1,2,\ldots,N$.
FP block:
$B_{m-1}(k-1) = \lambda B_{m-1}(k-2) + \gamma_{m-1}(k-1)\|e_{b,m-1}(k-1)\|^2$
$\overline{c}_{b,m-1}(k-1) = \frac{\lambda B_{m-1}(k-2)}{B_{m-1}(k-1)}$
$\overline{s}_{b,m-1}(k-1) = \gamma_{m-1}(k-1)\frac{e^*_{b,m-1}(k-1)}{B_{m-1}(k-1)}$
$e_{f,m}(k) = e_{f,m-1}(k) - e_{b,m-1}(k-1)\overline{\pi}^*_{f,m}(k-1)$
$\overline{\pi}^*_{f,m}(k) = \overline{\pi}^*_{f,m}(k-1) + \overline{s}_{b,m-1}(k-1)e_{f,m}(k)$
$\gamma_m(k-1) = \overline{c}_{b,m-1}(k-1)\gamma_{m-1}(k-1)$
BP block:
$F_{m-1}(k) = \lambda F_{m-1}(k-1) + \gamma_{m-1}(k-1)\|e_{f,m-1}(k)\|^2$
$\overline{s}_{f,m-1}(k) = \gamma_{m-1}(k-1)\frac{e^*_{f,m-1}(k)}{F_{m-1}(k)}$
$e_{b,m}(k) = e_{b,m-1}(k-1) - e_{f,m-1}(k)\overline{\pi}^*_{b,m}(k-1)$
$\overline{\pi}^*_{b,m}(k) = \overline{\pi}^*_{b,m}(k-1) + \overline{s}_{f,m-1}(k)e_{b,m}(k)$
Joint process estimation: For time $k = 0,1,\ldots$, and $m = 1,2,\ldots,N+1$.
$B_{m-1}(k) = \lambda B_{m-1}(k-1) + \gamma_{m-1}(k)\|e_{b,m-1}(k)\|^2$
$\overline{c}_{b,m-1}(k) = \frac{\lambda B_{m-1}(k-1)}{B_{m-1}(k)}$
$\overline{s}_{b,m-1}(k) = \gamma_{m-1}(k)\frac{e^*_{b,m-1}(k)}{B_{m-1}(k)}$
$e_m(k) = e_{m-1}(k) - e_{b,m-1}(k)\overline{p}^*_{m-1}(k-1)$
$\overline{p}^*_{m-1}(k) = \overline{p}^*_{m-1}(k-1) + \overline{s}_{b,m-1}(k)e_m(k)$
$\gamma_m(k) = \overline{c}_{b,m-1}(k)\gamma_{m-1}(k)$

5.4.1 QRD based on interpolation

Before deriving the SRF QRD-LSL interpolation algorithm, we briefly describe a modified QR-decomposition for interpolation. It can be shown that a $(k+1)\times(k+1)$ orthogonal matrix $\overline{\mathbf{Q}}(k)$ can always be constructed from one of the C_N^f orthogonal basis sets described in Section 5.2 such that it applies a generalized orthogonal triangularization to $\mathbf{X}_{N+1}(k)$ in (5.3)

$$\overline{\mathbf{Q}}(k)\mathbf{X}_{N+1}(k) = \begin{bmatrix} \mathbf{Q}_{p,f}(k) \\ \mathbf{S}(k) \end{bmatrix} \mathbf{X}_{N+1}(k) = \begin{bmatrix} \mathbf{R}_{p,f}(k) \\ \mathbf{O}_{(k-N)\times(N+1)} \end{bmatrix}, \tag{5.30}$$

Table 5.2 Summary of SRF QRD-LSL interpolation algorithm.

```
SRF QRD-LSL interpolation algorithm
Initialization:
    $\overline{\Delta}_{f,m}(-1) = \overline{\Delta}_{b,m}(-1) = 0$, for order $m = 1, 2, \ldots, N+1$,
    $\overline{\rho}_{p,f}(-1) = 0$, for all $p$ and $f$,
    $I_{p,f}(k) = \delta$, for $k \leq -1$ and all $p$ and $f$,
    $B_m(-1,k) = F_m(-1,k) = \delta$, for $k \leq -1$ and order $m = 1, 2, \ldots, N+1$
    $e^{I}_{p,f}(k) = 0$ for $k \leq -1$ and all $p$ and $f$, and $\gamma_{0,0}(k) = 1$ for $k \leq -1$,
    For $k = 0, 1, 2, \ldots$, set $e^{I}_{0,0}(k) = x(k)$.
For time $k = 0, 1, \ldots$, starting with $p = f = 0$.
As $(p,f) \rightarrow (p, f+1)$:
  IFP block:
    $I_{p,f}(k-f-1) = \lambda I_{p,f}(k-f-2) + \gamma_{p,f}(k-f-1)|e^{I}_{p,f}(k-f-1)|^2$
    $\overline{c}_{I,N}(k-1) = \frac{\lambda I_{p,f}(k-f-2)}{I_{p,f}(k-f-1)}$ (needed only for deriving QRD-LSL prediction)
    $\overline{s}_{I,N}(k-1) = \gamma_{p,f}(k-f-1)\frac{e^{I*}_{p,f}(k-f-1)}{I_{p,f}(k-f-1)}$
    $e_{f,N+1}(k,k-f-1) = e_{f,N+1}(k) + e^{I}_{p,f}(k-f-1)\overline{\Delta}^{*}_{f,N+1}(k-1)$
    $\overline{\Delta}^{*}_{f,N+1}(k) = \overline{\Delta}^{*}_{f,N+1}(k-1) + \overline{s}_{I,N}(k-1)e_{f,N+1}(k)$
  Int(F) block:
    $F_{N+1}(k,k-f-1) = \lambda F_{N+1}(k-1,k-f-2)$
                        $+\gamma_{p,f}(k-f-1)|e_{f,N+1}(k,k-f-1)|^2$
    $\overline{c}'_{f,N+1}(k) = \frac{\lambda F_{N+1}(k-1,k-f-2)}{F_{N+1}(k,k-f-1)}$
    $\overline{s}'_{f,N+1}(k) = \gamma_{p,f}(k-f-1)\frac{e^{*}_{f,N+1}(k,k-f-1)}{F_{N+1}(k,k-f-1)}$
    $e^{I}_{p,f+1}(k-f-1) = e^{I}_{p,f}(k-f-1) - e_{f,N+1}(k,k-f-1)\overline{\rho}^{*}_{p,f+1}(k-1)$
    $\overline{\rho}^{*}_{p,f+1}(k) = \overline{\rho}^{*}_{p,f+1}(k-1) + \overline{s}'_{f,N+1}(k)e^{I}_{p,f+1}(k-f-1)$
    $\gamma_{p,f+1}(k-f-1) = \overline{c}'_{f,N+1}(k)\gamma_{p,f}(k-f-1)$
As $(p,f) \rightarrow (p+1, f)$:
  IBP block:
    $I_{p,f}(k-f) = \lambda I_{p,f}(k-f-1) + \gamma_{p,f}(k-f)|e^{I}_{p,f}(k-f)|^2$
    $\overline{s}_{I,N}(k) = \gamma_{p,f}(k-f)\frac{e^{I*}_{p,f}(k-f)}{I_{p,f}(k-f)}$
    $e_{b,N+1}(k,k-f) = e_{b,N+1}(k) + e^{I}_{p,f}(k-f)\overline{\Delta}^{*}_{b,N+1}(k-1)$
    $\overline{\Delta}^{*}_{b,N+1}(k) = \overline{\Delta}^{*}_{b,N+1}(k-1) + \overline{s}_{I,N}(k)e_{b,N+1}(k)$
  Int(P) block:
    $B_{N+1}(k,k-f) = \lambda B_{N+1}(k-1,k-f-1) + \gamma_{p,f}(k-f)|e_{b,N+1}(k,k-f)|^2$
    $\overline{c}'_{b,N+1}(k) = \frac{\lambda B_{N+1}(k-1,k-f-1)}{B_{N+1}(k,k-f)}$
    $\overline{s}'_{b,N+1}(k) = \gamma_{p,f}(k-f)\frac{e^{*}_{b,N+1}(k,k-f)}{B_{N+1}(k,k-f)}$
    $e^{I}_{p+1,f}(k-f) = e^{I}_{p,f}(k-f) - e_{b,N+1}(k,k-f)\overline{\rho}^{*}_{p+1,f}(k-1)$
    $\overline{\rho}^{*}_{p+1,f}(k) = \overline{\rho}^{*}_{p+1,f}(k-1) + \overline{s}'_{b,N+1}(k)e^{I}_{p+1,f}(k-f)$
    $\gamma_{p+1,f}(k-f) = \overline{c}'_{b,N+1}(k)\gamma_{p,f}(k-f)$
```

where $\mathbf{Q}_{p,f}(k)$ contains the first $(N+1)$ rows of $\overline{\mathbf{Q}}(k)$, whereas $\mathbf{S}(k)$ contains the remaining rows; $\mathbf{O}_{(k-N)\times(N+1)}$ is a null matrix of order $(k-N)\times(N+1)$. Since C_N^f possible sequences can be used, the $(N+1)\times(N+1)$ matrix $\mathbf{R}_{p,f}(k)$ in (5.30) can display C_N^f different forms, all of which contain one $f \times f$ lower triangular matrix (upper-left submatrix of $\mathbf{R}_{p,f}(k)$) and one $p \times p$ upper triangular matrix (lower-right submatrix of $\mathbf{R}_{p,f}(k)$) with zero elements filling the $(f+1)$st row, except for the $(f+1, f+1)$st element. More specifically, our results indicate that $\mathbf{R}_{p,f}(k)$ using the BFBFBF... sequence can be shown to be

$$\mathbf{R}_{p,f}(k) = \left[\begin{array}{cccc|c|cccc}
\gamma_{F,1} & 0 & \cdots & 0 & \times & \times & \cdots & \cdots & \times \\
\times & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
\vdots & \ddots & \gamma_{F,2} & 0 & \vdots & \vdots & \ddots & \ddots & \vdots \\
\times & \cdots & \cdots & \gamma_{F,3} & \times & \times & \cdots & \cdots & \times \\
\hline
0 & \cdots & \cdots & 0 & I^{\frac{1}{2}}_{p,f}(k-f) & 0 & \cdots & \cdots & 0 \\
\hline
\times & \cdots & \cdots & \times & \times & \gamma_{B,1} & \times & \cdots & \times \\
\vdots & \ddots & \ddots & \vdots & \vdots & 0 & \gamma_{B,2} & \ddots & \vdots \\
\vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \ddots & \ddots & \times \\
\times & \cdots & \cdots & \times & \times & 0 & \cdots & 0 & \gamma_{B,3}
\end{array}\right] \tag{5.31}$$

in which $\gamma_{F,1} = F_N^{\frac{1}{2}}(k,k-f)$, $\gamma_{F,2} = F_4^{\frac{1}{2}}(k-f+2,k-f)$, $\gamma_{F,3} = F_2^{\frac{1}{2}}(k-f+1,k-f)$, $\gamma_{B,1} = B_1^{\frac{1}{2}}(k-f,k-f)$, $\gamma_{B,2} = B_3^{\frac{1}{2}}(k-f+1,k-f)$, $\gamma_{B,3} = B_{N-1}^{\frac{1}{2}}(k-1,k-f)$, where $F_m^{\frac{1}{2}}(k-j,k-f)$ and $B_m^{\frac{1}{2}}(k-j,k-f)$ are the square roots of the minimum sum of mth-*order* intermediate forward and backward prediction error squares, respectively, whereas $I^{\frac{1}{2}}_{p,f}(k-f)$ is the square root of the minimum sum of (p,f)th-*order* interpolation error square; the symbol $\times$ denotes an element whose value is not of direct interest. We refer to the result in (5.30) as the *modified QR-decomposition for interpolation* and refer to the form in $\mathbf{R}_{p,f}(k)$ of (5.31) as the *standard lower/upper (LU) triangular form for* a (p,f)th-*order interpolator* based on the QR-decomposition. For example, a special case of (5.31) identified by sequence BFBF for a $(2,2)$th-*order* LSL interpolator can be expressed as

$$\mathbf{R}_{2,2}(k) = \left[\begin{array}{cc|c|cc}
F_4^{\frac{1}{2}}(k,k-2) & 0 & \times & \times & \times \\
\times & F_2^{\frac{1}{2}}(k-1,k-2) & \times & \times & \times \\
\hline
0 & 0 & I^{\frac{1}{2}}_{2,2}(k-2) & 0 & 0 \\
\hline
\times & \times & \times & B_1^{\frac{1}{2}}(k-2,k-2) & \times \\
\times & \times & \times & 0 & B_3^{\frac{1}{2}}(k-1,k-2)
\end{array}\right] \tag{5.32}$$

Moreover, setting $(p,f) = (N,0)$, the lower-right $p \times p$ upper triangular submatrix of $\mathbf{R}_{p,f}(k)$ in (5.31) yields the following well-known upper triangular matrix after an unitary matrix is used to develop the QRD-LSL prediction algorithm.

$$\mathbf{R}_{N,0}(k-1) = \begin{bmatrix} B_0^{\frac{1}{2}}(k-1) & \times & \times & \dots & \times \\ & B_1^{\frac{1}{2}}(k-1) & \times & \dots & \times \\ & & B_2^{\frac{1}{2}}(k-1) & \dots & \times \\ & \mathbf{O} & & \ddots & \vdots \\ & & & & B_{N-1}^{\frac{1}{2}}(k-1) \end{bmatrix} \quad (5.33)$$

5.4.2 SRF QRD-LSL interpolation algorithm

The SRF QRD-LSL interpolation algorithm is derived according to the following seven stages. In each stage, either a single Givens rotation or a sequence of Givens rotations is applied. In the derivation, matrices are represented in uppercase boldface type and column vectors in lowercase boldface type, whereas scalars appear in plain text type. The dimensions of matrices and vectors appear as subscripts. For example, $\mathbf{A}_{m\times k}$ and $\mathbf{p}_m$ represent a $m \times k$ matrix and a $m \times 1$ column vector, respectively.

1. We first write (5.3), at time $k-2$, as $\boldsymbol{\varepsilon}^I_{p,f}(k-f-2) = \mathbf{X}_{N+1}(k-2)\mathbf{b}_{p,f}(k-f-2)$ and pre-multiply both sides of the equation by $\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)$, where $\boldsymbol{\Lambda}(k-2) = diag[\lambda^{k-2}, \lambda^{k-3}, \dots, 1]$ is the $(k-1)\times(k-1)$ exponential weighting matrix in which $0 \ll \lambda \leq 1$ is the forgetting factor. Next, we apply a sequence of N Givens rotations that define the $(k-1)\times(k-1)$ orthogonal matrix $\overline{\mathbf{Q}}(k-2) = \begin{bmatrix} \mathbf{Q}_{p,f}(k-2) \\ \mathbf{S}(k-2) \end{bmatrix}$ such that matrix $\mathbf{Q}_{p,f}(k-2)$ transforms data matrix $\mathbf{X}_{N+1}(k-2)$ into matrix

$$\mathbf{R}_{p,f}(k-2) = \begin{bmatrix} \mathbf{L}_{f\times f}(k-2) & \mathbf{p}_f(k-2) & \mathbf{B}_{f\times p}(k-2) \\ \mathbf{0}^{\mathrm{T}}_f & I^{\frac{1}{2}}_{p,f}(k-f-2) & \mathbf{0}^{\mathrm{T}}_p \\ \mathbf{A}_{p\times f}(k-2) & \mathbf{p}_p(k-2) & \mathbf{U}_{p\times p}(k-2) \end{bmatrix}, \quad (5.34)$$

   which is in the standard LU triangular form for a (p,f)th-*order* interpolator as shown in (5.31) with $x(k-2)$ being the most recent data sample used. Notably, $\mathbf{L}_{f\times f}(k-2)$ is the $f\times f$ lower triangular matrix and $\mathbf{U}_{p\times p}(k-2)$ is the $p\times p$ upper triangular matrix. We may thus write

$$\begin{aligned} &\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)\overline{\mathbf{Q}}(k-2)\boldsymbol{\varepsilon}^I_{p,f}(k-f-2) \\ &\quad = \boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)\begin{bmatrix} \mathbf{R}_{p,f}(k-2) \\ \mathbf{O}_{(k-N-2)\times(N+1)} \end{bmatrix}\mathbf{b}_{p,f}(k-f-2). \end{aligned} \quad (5.35)$$

2. We may apply the transformations produced by $\begin{bmatrix} 1 & \\ & \overline{\mathbf{Q}}(k-2) \end{bmatrix}$ and $\overline{\mathbf{Q}}(k-2)$ to the data vectors $\mathbf{x}_k(k-1) = [x^*(0), x^*(1), \dots, x^*(k-1)]^{\mathrm{T}}$ and $\mathbf{x}_{k-1}(k-N-3) = [0, \dots, 0, x^*(0), \dots, x^*(k-N-3)]^{\mathrm{T}}$, respectively. The two data vectors containing the next future and next past data samples for consideration, respectively, will serve the purpose of deriving the order-updated recursions for the interpolation error in the later stages. We may thus write

$$\begin{bmatrix} 1 & \\ & \overline{\mathbf{Q}}(k-2) \end{bmatrix} \boldsymbol{\Lambda}^{\frac{1}{2}}(k-1)\mathbf{x}_k(k-1)$$
$$= [\lambda^{\frac{k-1}{2}} x^*(0), \mathbf{f}_f^{\mathrm{T}}(k-1), \Delta_{f,N+1}(k-1), \mathbf{f}_p^{\mathrm{T}}(k-1), \mathbf{f}_{k-N-2}^{\mathrm{T}}(k-1)]^{\mathrm{T}} \tag{5.36}$$

and

$$\overline{\mathbf{Q}}(k-2)\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)\mathbf{x}_{k-1}(k-N-3)$$
$$= [\mathbf{b}_f^{\mathrm{T}}(k-2), \Delta_{b,N+1}(k-2), \mathbf{b}_p^{\mathrm{T}}(k-2), \mathbf{b}_{k-N-2}^{\mathrm{T}}(k-2)]^{\mathrm{T}}, \tag{5.37}$$

where $\Delta_{f,N+1}(k-1)$ and $\Delta_{b,N+1}(k-2)$ are auxiliary parameters which will be used later to obtain the intermediate prediction errors. All vectors appearing on the right-hand-side of (5.36) and (5.37) are defined merely for convenience of presentation; their elements are not of direct interest. By appending both transformed data vectors obtained in (5.36) and (5.37), respectively, as the leftmost column and rightmost column of the matrix defined by $\boldsymbol{\Lambda}^{\frac{1}{2}}(k-2)$ $\begin{bmatrix} \mathbf{R}_{p,f}(k-2) \\ \mathbf{O}_{(k-N-2)\times(N+1)} \end{bmatrix}$ in (5.35) [see the box in (5.38)], together with the new data sample vector for time k at the bottom row, we obtain the following expanded matrix $\mathbf{D}(k)$, which can be written as

$$\left[\begin{array}{c|ccc|c}
\lambda^{\frac{k}{2}} x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\ \cline{2-4}
\lambda^{\frac{1}{2}} \mathbf{f}_f(k-1) & \lambda^{\frac{1}{2}} \mathbf{L}_{f\times f}(k-2) & \lambda^{\frac{1}{2}} \mathbf{p}_f(k-2) & \lambda^{\frac{1}{2}} \mathbf{B}_{f\times p}(k-2) & \lambda^{\frac{1}{2}} \mathbf{b}_f(k-2) \\
\lambda^{\frac{1}{2}} \Delta_{f,N+1}(k-1) & \mathbf{0}_f^{\mathrm{T}} & \lambda^{\frac{1}{2}} I_{p,f}^{\frac{1}{2}}(k-f-2) & \mathbf{0}_p^{\mathrm{T}} & \lambda^{\frac{1}{2}} \Delta_{b,N+1}(k-2) \\
\lambda^{\frac{1}{2}} \mathbf{f}_p(k-1) & \lambda^{\frac{1}{2}} \mathbf{A}_{p\times f}(k-2) & \lambda^{\frac{1}{2}} \mathbf{p}_p(k-2) & \lambda^{\frac{1}{2}} \mathbf{U}_{p\times p}(k-2) & \lambda^{\frac{1}{2}} \mathbf{b}_p(k-2) \\
\lambda^{\frac{1}{2}} \mathbf{f}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}_{k-N-2}(k-2) \\ \cline{2-4}
x^*(k) & \ldots & x^*(k-f-1) & \ldots & x^*(k-N-2)
\end{array}\right]. \tag{5.38}$$

3. Next, we apply a sequence of Givens rotations to annihilate all elements in the bottom row of the matrix $\mathbf{D}(k)$ except for the $(k+1,1)$th, $(k+1,f+2)$th, and $(k+1,N+3)$th elements. These rotations include an appropriate combination of a sequence of f Givens rotations proceeding leftwards from the $(k+1,f+1)$th element to the $(k+1,2)$th element and a sequence of p Givens rotations proceeding rightwards from the $(k+1,f+3)$th element to the $(k+1,N+2)$th element with the order in which the elements are annihilated in accordance with the sequencing chosen (e.g., BFBFBF…) to preserve the standard LU triangular form for interpolator in the transformed matrix. Note that any sequencing between F and B is permissible. Accordingly, there are C_N^f possible sequences. For example, if the sequence BFBFBF… is chosen, then the elements at the bottom row of $\mathbf{D}(k)$ in the following order: $(k+1,f+3)$, $(k+1,f+1)$, $(k+1,f+4)$, $(k+1,f)$, $(k+1,f+5)$, $(k+1,f-1)$, …, $(k+1,N+2)$, $(k+1,2)$ will be annihilated successively. Such a sequence of N Givens rotations defines the $(k+1)\times(k+1)$ orthogonal matrix $\mathbf{L}(k)$ that transforms the matrix $\mathbf{D}(k)$ to the matrix $\mathbf{E}(k)$ as follows:

$$\mathbf{L}(k)\mathbf{D}(k) = \mathbf{E}(k), \tag{5.39}$$

where $\mathbf{E}(k)$ can be written as

$$\begin{bmatrix} \lambda^{\frac{k}{2}} x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\ \mathbf{f}_f(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_f(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_f(k-1) \\ \lambda^{\frac{1}{2}} \Delta_{f,N+1}(k-1) & \mathbf{0}_f^{\mathrm{T}} & \lambda^{\frac{1}{2}} I_{p,f}^{\frac{1}{2}}(k-f-2) & \mathbf{0}_p^{\mathrm{T}} & \lambda^{\frac{1}{2}} \Delta_{b,N+1}(k-2) \\ \mathbf{f}_p(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_p(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_p(k-1) \\ \lambda^{\frac{1}{2}} \mathbf{f}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}_{k-N-2}(k-2) \\ \overline{\varepsilon}^*_{f,N+1}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & \overline{\varepsilon}^* & \mathbf{0}_p^{\mathrm{T}} & \overline{\varepsilon}^*_{b,N+1}(k-1,k-f-1) \end{bmatrix}.$$

Since $x^*(k-f-1)$ at the bottom row of $\mathbf{D}(k)$ is the present data sample to be estimated by its p past and f future neighboring data samples, it was not annihilated by $\mathbf{L}(k)$ in the above transformation. Consequently, a non-zero quantity, $\overline{\varepsilon}^*$, was generated at the bottom row of $\mathbf{E}(k)$. By using the fact that orthogonal rotations are norm preserving, one can show that $\overline{\varepsilon}^*$ is actually the complex conjugate of the (p,f)th-*order* angle-normalized interpolation error, $\overline{\varepsilon}^{I^*}_{p,f}(k-f-1)$, with $x(k-1)$ being the most recent data sample used. Notably, due to the non-zero quantity $\overline{\varepsilon}^*$ at the bottom row of $\mathbf{E}(k)$, the $(k+1,1)$th and $(k+1,N+3)$th elements of $\mathbf{E}(k)$ in (5.39) are $\overline{\varepsilon}^*_{f,N+1}(k,k-f-1)$ and $\overline{\varepsilon}^*_{b,N+1}(k-1,k-f-1)$, which are the angle-normalized intermediate forward and backward prediction errors of order $N+1$, respectively.

4. Both the conventional angle-normalized FP error, $\overline{\varepsilon}_{f,N+1}(k)$, and the angle-normalized backward prediction error, $\overline{\varepsilon}_{b,N+1}(k-1)$, appear if the $(k+1,f+2)$th element of $\mathbf{E}(k)$ is annihilated. This can be accomplished by applying a single Givens rotation to $\mathbf{E}(k)$. We may thus write

$$\mathbf{J}_{I,N}(k-1)\mathbf{E}(k) = \mathbf{F}(k), \tag{5.40}$$

where $\mathbf{J}_{I,N}(k-1) = \begin{bmatrix} \mathbf{I}_{f+1} & & & \\ & c_{I,N}(k-1) & & s^*_{I,N}(k-1) \\ & & \mathbf{I}_{k-f-2} & \\ & -s_{I,N}(k-1) & & c_{I,N}(k-1) \end{bmatrix}$ and $\mathbf{F}(k)$ can be written as

$$\begin{bmatrix} \lambda^{\frac{k}{2}} x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\ \mathbf{f}_f(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_f(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_f(k-1) \\ \Delta_{f,N+1}(k) & \mathbf{0}_f^{\mathrm{T}} & I_{p,f}^{\frac{1}{2}}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & \Delta_{b,N+1}(k-1) \\ \mathbf{f}_p(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_p(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_p(k-1) \\ \lambda^{\frac{1}{2}} \mathbf{f}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}_{k-N-2}(k-2) \\ \overline{\varepsilon}^*_{f,N+1}(k) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & \overline{\varepsilon}^*_{b,N+1}(k-1) \end{bmatrix}$$

5. The annihilation in (5.40) has the effect of computing both the intermediate forward and backward prediction errors from the conventional forward and backward prediction errors. By taking out a scaling factor [i.e., the square root term of each row of $\mathbf{E}(k)$ and $\mathbf{F}(k)$] from each row of matrices $\mathbf{E}(k)$ and $\mathbf{F}(k)$ [31], the rows before and after the Givens rotation in (5.40) is denoted, respectively, by

$$\begin{aligned}\mathbf{E}(k) &= \mathbf{E}_1(k)\mathbf{E}_2(k)\\ &= diag\left[\sqrt{\lambda^k},\ldots,\sqrt{\lambda I_{p,f}(k-f-2)},\ldots,\sqrt{\gamma_{p,f}(k-f-1)}\right]\end{aligned}$$
$$\begin{bmatrix} x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\ \bar{\mathbf{f}}_f(k) & \bar{\mathbf{L}}_{f\times f}(k-1) & \bar{\mathbf{p}}_f(k-1) & \bar{\mathbf{B}}_{f\times p}(k-1) & \bar{\mathbf{b}}_f(k-1) \\ \overline{\Delta}_{f,N+1}(k-1) & \mathbf{0}_f^{\mathrm{T}} & 1 & \mathbf{0}_p^{\mathrm{T}} & \overline{\Delta}_{b,N+1}(k-2) \\ \bar{\mathbf{f}}_p(k) & \bar{\mathbf{A}}_{p\times f}(k-1) & \bar{\mathbf{p}}_p(k-1) & \bar{\mathbf{U}}_{p\times p}(k-1) & \bar{\mathbf{b}}_p(k-1) \\ \bar{\mathbf{f}}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \bar{\mathbf{b}}_{k-N-2}(k-2) \\ e^*_{f,N+1}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & e^{I^*}_{p,f}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & e^*_{b,N+1}(k-1,k-f-1) \end{bmatrix}, \tag{5.41}$$

and

$$\begin{aligned}\mathbf{F}(k) &= \mathbf{F}_1(k)\mathbf{F}_2(k)\\ &= diag\left[\sqrt{\lambda^k},\ldots,\sqrt{I_{p,f}(k-f-1)},\ldots,\sqrt{\gamma'_{p,f}(k-f-1)}\right]\end{aligned}$$
$$\begin{bmatrix} x^*(0) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & 0 \\ \bar{\mathbf{f}}_f(k) & \bar{\mathbf{L}}_{f\times f}(k-1) & \bar{\mathbf{p}}_f(k-1) & \bar{\mathbf{B}}_{f\times p}(k-1) & \bar{\mathbf{b}}_f(k-1) \\ \overline{\Delta}_{f,N+1}(k) & \mathbf{0}_f^{\mathrm{T}} & 1 & \mathbf{0}_p^{\mathrm{T}} & \overline{\Delta}_{b,N+1}(k-1) \\ \bar{\mathbf{f}}_p(k) & \bar{\mathbf{A}}_{p\times f}(k-1) & \bar{\mathbf{p}}_p(k-1) & \bar{\mathbf{U}}_{p\times p}(k-1) & \bar{\mathbf{b}}_p(k-1) \\ \bar{\mathbf{f}}_{k-N-2}(k-1) & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \bar{\mathbf{b}}_{k-N-2}(k-2) \\ e^*_{f,N+1}(k) & \mathbf{0}_f^{\mathrm{T}} & 0 & \mathbf{0}_p^{\mathrm{T}} & e^*_{b,N+1}(k-1) \end{bmatrix}. \tag{5.42}$$

Some elements of the matrices in (5.41) and (5.42) and the corresponding elements of the matrices in (5.19) and (5.20), respectively, can be related as follows: $k_a = \lambda I_{p,f}(k-f-2)$, $k_b = \gamma_{p,f}(k-f-1)$, $a_1 = 1$, $a_2 = \overline{\Delta}_{f,N+1}(k-1)$, $a_3 = \overline{\Delta}_{b,N+1}(k-2)$, $b_1 = e^{I^*}_{p,f}(k-f-1)$, $b_2 = e^*_{f,N+1}(k,k-f-1)$, $b_3 = e^*_{b,N+1}(k-1,k-f-1)$, and $k'_a = I_{p,f}(k-f-1)$, $k'_b = \gamma'_{p,f}(k-f-1)$, $a'_1 = 1$, $a'_2 = \overline{\Delta}_{f,N+1}(k)$, $a'_3 = \overline{\Delta}_{b,N+1}(k-1)$, $b'_1 = 0$, $b'_2 = e^*_{f,N+1}(k)$, $b'_3 = e^*_{b,N+1}(k-1)$. Notably, "e" represents the *a priori* estimation error. Substituting k_a, k'_a, k_b, b_1, a_2, a'_2, b_2, b'_2, a_3, a'_3, b_3, and b'_3 into (5.21), (5.25), (5.28), and (5.29) (by letting $i = 2$ and $i = 3$) yields

$$I_{p,f}(k-f-1) = \lambda I_{p,f}(k-f-2) + \gamma_{p,f}(k-f-1)|e^I_{p,f}(k-f-1)|^2, \tag{5.43}$$

$$\bar{s}_{I,N}(k-1) = \gamma_{p,f}(k-f-1)\frac{e^{I^*}_{p,f}(k-f-1)}{I_{p,f}(k-f-1)}, \tag{5.44}$$

$$e_{f,N+1}(k) = e_{f,N+1}(k,k-f-1) - e^I_{p,f}(k-f-1)\overline{\Delta}^*_{f,N+1}(k-1), \tag{5.45}$$

$$\overline{\Delta}^*_{f,N+1}(k) = \overline{\Delta}^*_{f,N+1}(k-1) + \bar{s}_{I,N}(k-1)e_{f,N+1}(k), \tag{5.46}$$

$$e_{b,N+1}(k) = e_{b,N+1}(k,k-f) - e^I_{p,f}(k-f)\overline{\Delta}^*_{b,N+1}(k-1), \tag{5.47}$$

$$\overline{\Delta}^*_{b,N+1}(k) = \overline{\Delta}^*_{b,N+1}(k-1) + \bar{s}_{I,N}(k)e_{b,N+1}(k). \tag{5.48}$$

Note that the *a priori* IFP error $e_{f,N+1}(k,k-f-1)$ in (5.45) is still unknown whereas the *a priori* FP error $e_{f,N+1}(k)$ in (5.45) has already been computed by using the SRF QRD-LSL prediction algorithm (see Table 5.1), which will be derived in Section 5.4.3. In order to compute the *a priori* IFP error, we "reverse" (5.45) in formulation such that, given the *a priori* FP error, the *a priori* IFP error can be computed as

$$e_{f,N+1}(k,k-f-1) = e_{f,N+1}(k) + e^{I}_{p,f}(k-f-1)\overline{\Delta}^{*}_{f,N+1}(k-1). \tag{5.49}$$

Similarly, (5.47) can be reformulated as

$$e_{b,N+1}(k,k-f) = e_{b,N+1}(k) + e^{I}_{p,f}(k-f)\overline{\Delta}^{*}_{b,N+1}(k-1), \tag{5.50}$$

which also corresponds to the "reversed" formulation. Notably, (5.49) and (5.50) correspond to (5.11) and (5.10), respectively, and these computed intermediate prediction errors are then used to compute the order-updated interpolation errors in the following stages, as $(p,f) \to (p,f+1)$ and as $(p,f) \to (p+1,f)$.

6. To obtain the order-updated interpolation error of order $(p,f+1)$ from that of the current order (p,f) as an additional *future* data sample is used, we proceed with by transforming the matrix $\mathbf{E}(k)$ into the standard LU triangular form for a $(p,f+1)$st-*order* interpolator. This transformation can be achieved by initially applying an $(k+1)\times(k+1)$ orthogonal matrix $\mathbf{P}(k)$ to $\mathbf{E}(k)$. The matrix $\mathbf{P}(k)$ represents the combined transformation produced by a sequence of $(k-N-2)$ Givens rotations, which has the effect of annihilating all the $(k-N-2)$ elements of vector $\lambda^{\frac{1}{2}}\mathbf{f}_{k-N-2}(k-1)$ in the first column of matrix $\mathbf{E}(k)$. We may thus write

$$\mathbf{P}(k)\mathbf{E}(k) = \mathbf{G}(k), \tag{5.51}$$

where $\mathbf{G}(k)$ can be written as

$$\left[\begin{array}{ccccc}
\lambda^{\frac{1}{2}} F^{\frac{1}{2}}_{N+1}(k-1) & \mathbf{0}^{\mathrm{T}}_{f} & 0 & \mathbf{0}^{\mathrm{T}}_{p} & \times \\
\mathbf{f}_{f}(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_{f}(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_{f}(k-1) \\
\lambda^{\frac{1}{2}} \Delta_{f,N+1}(k-1) & \mathbf{0}^{\mathrm{T}}_{f} & \lambda^{\frac{1}{2}} I^{\frac{1}{2}}_{p,f}(k-f-2) & \mathbf{0}^{\mathrm{T}}_{p} & \lambda^{\frac{1}{2}} \Delta_{b,N+1}(k-2) \\
\mathbf{f}_{p}(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_{p}(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_{p}(k-1) \\
\mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}'_{k-N-2}(k-2) \\
\overline{\varepsilon}^{*}_{f,N+1}(k,k-f-1) & \mathbf{0}^{\mathrm{T}}_{f} & \overline{\varepsilon}^{I*}_{p,f}(k-f-1) & \mathbf{0}^{\mathrm{T}}_{p} & \overline{\varepsilon}^{*}_{b,N+1}(k-1,k-f-1)
\end{array}\right]$$

where $\lambda F_{N+1}(k-1) = \lambda^{k}|x(0)|^{2} + \|\lambda^{\frac{1}{2}}\mathbf{f}_{k-N-2}(k-1)\|^{2}$ is the minimum weighted sum of the FP error square. This transformation is followed by rotating the element $\lambda^{\frac{1}{2}}\Delta_{f,N+1}(k-1)$ in the first column of matrix $\mathbf{G}(k)$ into the $(1,f+2)$th element of the matrix such that the resulting matrix $\mathbf{C}(k)$ in (5.52) will be in standard LU triangular form that can be used to compute the order-updated interpolation error. We may thus write

$$\begin{bmatrix} c_\Delta(k-1) & & s^*_\Delta(k-1) & \\ & \mathbf{I}_f & & \\ -s_\Delta(k-1) & & c_\Delta(k-1) & \\ & & & \mathbf{I}_{k-f-1} \end{bmatrix} \mathbf{G}(k) = \mathbf{C}(k), \tag{5.52}$$

where $\mathbf{C}(k)$ can be written as

$$\begin{bmatrix} \gamma_F & \mathbf{0}_f^{\mathrm{T}} & \lambda^{\frac{1}{2}} \rho_{p,f+1}(k-1) & \mathbf{0}_p^{\mathrm{T}} & \times \\ \mathbf{f}_f(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_f(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_f(k-1) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & \gamma_I & \mathbf{0}_p^{\mathrm{T}} & \times \\ \mathbf{f}_p(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_p(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_p(k-1) \\ \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}'_{k-N-2}(k-2) \\ \overline{\varepsilon}^*_{f,N+1}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & \overline{\varepsilon}^{I*}_{p,f}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & \overline{\varepsilon}^*_{b,N+1}(k-1,k-f-1) \end{bmatrix}$$

in which $\gamma_I = \lambda^{\frac{1}{2}} I^{\frac{1}{2}}_{p,f+1}(k-f-2)$; the $(1,1)$th element of $\mathbf{C}(k)$, $\gamma_F = \lambda^{\frac{1}{2}} F^{\frac{1}{2}}_{N+1}$ $(k-1,k-f-2)$, is the square root of the minimum weighted sum of the IFP error square and $\rho_{p,f+1}(k-1)$ is the interpolation auxiliary parameter that will be used in stage 7. Notably, the $(f+2,f+2)$th element of $\mathbf{C}(k)$ becomes $\lambda^{\frac{1}{2}} I^{\frac{1}{2}}_{p,f+1}(k-f-2)$. Clearly, the upper-left $(N+2)\times(N+2)$ submatrix of $\mathbf{C}(k)$ denotes the standard LU triangular form for a $(p,f+1)$st order interpolator and is used in the next stage to compute the order-updated interpolation error of order $(p,f+1)$.

7. We are now positioned to obtain the order-updated recursion $(p,f)\to(p,f+1)$ for the interpolation error as an additional future data sample is used by applying one single Givens rotation to $\mathbf{C}(k)$ so as to annihilate $\overline{\varepsilon}^*_{f,N+1}(k,k-f-1)$ at the bottom row of $\mathbf{C}(k)$. We thus obtain

$$\mathbf{J}'_{F,N+1}(k)\mathbf{C}(k) = \mathbf{J}(k), \tag{5.53}$$

where $\mathbf{J}'_{F,N+1}(k) = \begin{bmatrix} c'_{f,N+1}(k) & s'^*_{f,N+1}(k) \\ & \mathbf{I}_{k-1} \\ -s'_{f,N+1}(k) & c'_{f,N+1}(k) \end{bmatrix}$ and $\mathbf{J}(k)$ is

$$\begin{bmatrix} F^{\frac{1}{2}}_{N+1}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & \rho_{p,f+1}(k) & \mathbf{0}_p^{\mathrm{T}} & \times \\ \mathbf{f}_f(k) & \mathbf{L}_{f\times f}(k-1) & \mathbf{p}_f(k-1) & \mathbf{B}_{f\times p}(k-1) & \mathbf{b}_f(k-1) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & \lambda^{\frac{1}{2}} I^{\frac{1}{2}}_{p,f+1}(k-f-2) & \mathbf{0}_p^{\mathrm{T}} & \times \\ \mathbf{f}_p(k) & \mathbf{A}_{p\times f}(k-1) & \mathbf{p}_p(k-1) & \mathbf{U}_{p\times p}(k-1) & \mathbf{b}_p(k-1) \\ \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \lambda^{\frac{1}{2}} \mathbf{b}'_{k-N-2}(k-2) \\ 0 & \mathbf{0}_f^{\mathrm{T}} & \overline{\varepsilon}^{I*}_{p,f+1}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & \overline{\varepsilon}^*_{b,N+2}(k,k-f-1) \end{bmatrix}.$$

Similar to the procedure in (5.41) and (5.42), scalar factors in the first row and the bottom row in matrices $\mathbf{C}(k)$ and $\mathbf{J}(k)$ can be taken out:

$$
\begin{aligned}
\mathbf{C}(k) &= \mathbf{C}_1(k)\mathbf{C}_2(k) \\
&= diag\left[\sqrt{\lambda F_{N+1}(k-1,k-f-2)},\ldots,\sqrt{\gamma_{p,f}(k-f-1)}\right] \\
&\left[\begin{array}{ccccc}
1 & \mathbf{0}_f^{\mathrm{T}} & \overline{\rho}_{p,f+1}(k-1) & \mathbf{0}_p^{\mathrm{T}} & \times \\
\bar{\mathbf{f}}_f(k) & \overline{\mathbf{L}}_{f\times f}(k-1) & \overline{\mathbf{p}}_f(k-1) & \overline{\mathbf{B}}_{f\times p}(k-1) & \overline{\mathbf{b}}_f(k-1) \\
0 & \mathbf{0}_f^{\mathrm{T}} & \times & \mathbf{0}_p^{\mathrm{T}} & \times \\
\bar{\mathbf{f}}_p(k) & \overline{\mathbf{A}}_{p\times f}(k-1) & \overline{\mathbf{p}}_p(k-1) & \overline{\mathbf{U}}_{p\times p}(k-1) & \overline{\mathbf{b}}_p(k-1) \\
\mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \overline{\mathbf{b}}'_{k-N-2}(k-2) \\
e^*_{f,N+1}(k,k-f-1) & \mathbf{0}_f^{\mathrm{T}} & e^{I^*}_{p,f}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & e^*_{b,N+1}(k-1,k-f-1)
\end{array}\right],
\end{aligned}
\tag{5.54}
$$

and

$$
\begin{aligned}
\mathbf{J}(k) &= \mathbf{J}_1(k)\mathbf{J}_2(k) \\
&= diag\left[\sqrt{F_{N+1}(k,k-f-1)},\ldots,\sqrt{\gamma_{p,f+1}(k-f-1)}\right] \\
&\left[\begin{array}{ccccc}
1 & \mathbf{0}_f^{\mathrm{T}} & \overline{\rho}_{p,f+1}(k) & \mathbf{0}_p^{\mathrm{T}} & \times \\
\bar{\mathbf{f}}_f(k) & \overline{\mathbf{L}}_{f\times f}(k-1) & \overline{\mathbf{p}}_f(k-1) & \overline{\mathbf{B}}_{f\times p}(k-1) & \overline{\mathbf{b}}_f(k-1) \\
0 & \mathbf{0}_f^{\mathrm{T}} & \times & \mathbf{0}_p^{\mathrm{T}} & \times \\
\bar{\mathbf{f}}_p(k) & \overline{\mathbf{A}}_{p\times f}(k-1) & \overline{\mathbf{p}}_p(k-1) & \overline{\mathbf{U}}_{p\times p}(k-1) & \overline{\mathbf{b}}_p(k-1) \\
\mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times f} & \mathbf{0}_{k-N-2} & \mathbf{O}_{(k-N-2)\times p} & \overline{\mathbf{b}}'_{k-N-2}(k-2) \\
0 & \mathbf{0}_f^{\mathrm{T}} & e^{I^*}_{p,f+1}(k-f-1) & \mathbf{0}_p^{\mathrm{T}} & e^*_{b,N+2}(k,k-f-1)
\end{array}\right].
\end{aligned}
\tag{5.55}
$$

Some elements of matrices $\mathbf{C}_1(k)$ and $\mathbf{C}_2(k)$ in (5.54) and matrices $\mathbf{J}_1(k)$ and $\mathbf{J}_2(k)$ in (5.55) and the corresponding elements of the matrices in (5.19) and (5.20), respectively, can be related as follows: $k_a = \lambda F_{N+1}(k-1,k-f-2)$, $k_b = \gamma_{p,f}(k-f-1)$, $a_1 = 1$, $a_2 = \overline{\rho}_{p,f+1}(k-1)$, $b_1 = e^*_{f,N+1}(k,k-f-1)$, $b_2 = e^{I^*}_{p,f}(k-f-1)$, and $k'_a = F_{N+1}(k,k-f-1)$, $k'_b = \gamma_{p,f+1}(k-f-1)$, $a'_1 = 1$, $a'_2 = \overline{\rho}_{p,f+1}(k)$, $b'_1 = 0$, $b'_2 = e^{I^*}_{p,f+1}(k-f-1)$. Substituting k_a, k'_a, k_b, b_1, a_2, a'_2, b_2, and b'_2 into (5.21), (5.22), (5.25), (5.28), and (5.29) (by letting $i = 2$) yields

$$
\begin{aligned}
F_{N+1}(k,k-f-1) = \lambda F_{N+1}(k-1,k-f-2) \\
+ \gamma_{p,f}(k-f-1)|e_{f,N+1}(k,k-f-1)|^2,
\end{aligned}
\tag{5.56}
$$

$$
\overline{c}'_{f,N+1}(k) = \frac{\lambda F_{N+1}(k-1,k-f-2)}{F_{N+1}(k,k-f-1)},
\tag{5.57}
$$

$$
\overline{s}'_{f,N+1}(k) = \gamma_{p,f}(k-f-1)\frac{e^*_{f,N+1}(k,k-f-1)}{F_{N+1}(k,k-f-1)},
\tag{5.58}
$$

$$
e^I_{p,f+1}(k-f-1) = e^I_{p,f}(k-f-1) - e_{f,N+1}(k,k-f-1)\overline{\rho}^*_{p,f+1}(k-1),
\tag{5.59}
$$

$$
\overline{\rho}^*_{p,f+1}(k) = \overline{\rho}^*_{p,f+1}(k-1) + \overline{s}'_{f,N+1}(k)e^I_{p,f+1}(k-f-1),
\tag{5.60}
$$

$$
\gamma_{p,f+1}(k-f-1) = \overline{c}'_{f,N+1}(k)\gamma_{p,f}(k-f-1).
\tag{5.61}
$$

Equations (5.43), (5.44), (5.49), (5.46) (summarized in the IFP block of Table 5.2) and (5.56), (5.57), (5.58), (5.59), (5.60), and (5.61) (summarized in the Int(F) block of Table 5.2) constitute the *SRF QRD-LSL interpolation algorithm* as $(p,f) \to (p,f+1)$. However, $e^I_{p,f+1}(k-f-1)$ computed in (5.59) is actually the *a priori* interpolation error of order $(p,f+1)$. The *a posteriori* interpolation error of order $(p,f+1)$ can then be computed by

$$\varepsilon^I_{p,f+1}(k-f-1) = \gamma_{p,f+1}(k-f-1)e^I_{p,f+1}(k-f-1). \tag{5.62}$$

The derivation of the order-updated interpolation error of order $(p,f) \to (p+1,f)$ can be similarly obtained.

5.4.3 SRF QRD-LSL prediction algorithm and SRF joint process estimation

The widely known SRF QRD-LSL prediction algorithm, which consists of both forward prediction (FP) block and backward prediction (BP) block summarized in Table 5.1, is actually a special case of the SRF QRD-LSL interpolation algorithm. The SRF QRD-LSL prediction algorithm in FP block and BP block can be directly derived by setting $(p,f) = (0,N)$ in the IFP block and $(p,f) = (N,0)$ in the IBP block, respectively. In deriving the SRF QRD-LSL prediction algorithm, the following results have been used. $e^I_{0,N}(k-N-1) = e_{b,N}(k-1)$, $e_{f,N+1}(k,k-N-1) = e_{f,N}(k)$, $I_{0,N}(k-N-1) = B_N(k-1)$, $e^I_{N,0}(k) = e_{f,N}(k)$, $e_{b,N+1}(k,k) = e_{b,N}(k-1)$, $I_{N,0}(k) = F_N(k)$. The FP block and the BP block can also be reduced directly from Int(P) block and Int(F) block by setting $(p,f) = (N,0)$ and $(p,f) = (0,N)$, respectively.

The SRF QRD-LSL prediction algorithm provides the mathematical foundation for the joint process estimation displayed in Figure 5.1 and is used as a sub-system to solve the joint process estimation problem where two optimal estimations are performed jointly. The two optimal estimations are (a) the forward reflection coefficients $\overline{\pi}_{f,m}(k)$ in the FP block and the backward reflection coefficients $\overline{\pi}_{b,m}(k)$ in the BP block both of which characterize a multistage lattice predictor with input $x(k)$ in the LS sense; (b) the regression coefficients $\overline{p}_m(k)$ that characterize a LS estimator of $d(k)$ to be developed below.

The SRF joint process estimation is developed by first considering a special case of (5.30) by setting $(p,f) = (m,0)$, which transforms the data matrix $\mathbf{X}_m(k-1)$

into the following upper triangular form that clearly is related to the QRD-LSL prediction, with $x(k-1)$ being the most recent data sample used,

$$\overline{\mathbf{Q}}(k-1)\boldsymbol{\Lambda}^{\frac{1}{2}}(k-1)\mathbf{X}_m(k-1) = \begin{bmatrix} \mathbf{R}_{m-1,0}(k-1) & \mathbf{p}_{b,m-1}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B^{\frac{1}{2}}_{m-1}(k-1) \\ \mathbf{O}_{(k-m)\times(m-1)} & \mathbf{0} \end{bmatrix}, \tag{5.63}$$

where $\mathbf{R}_{m-1,0}(k-1)$ is a $(m-1)\times(m-1)$ upper triangular matrix as shown in (5.33) and $\mathbf{p}_{b,m-1}(k-1)$ is a $(m-1)\times 1$ vector whose elements are not of direct interest. The same $k\times k$ unitary matrix $\overline{\mathbf{Q}}(k-1)$ is also applied to a weighted *desired signal vector* to obtain

$$\overline{\mathbf{Q}}(k-1)\boldsymbol{\Lambda}^{\frac{1}{2}}(k-1)\mathbf{d}(k-1) = \begin{bmatrix} \mathbf{p}(k-1) \\ p_{m-1}(k-1) \\ \mathbf{v}(k-1) \end{bmatrix}, \tag{5.64}$$

where $\mathbf{d}^{\mathrm{T}}(k-1) = [d^*(0), d^*(1), \ldots, d^*(k-1)]$, $\mathbf{p}^{\mathrm{T}}(k-1) = [p_0(k-1), p_1(k-1), \ldots, p_{m-2}(k-1)]$, and $\mathbf{v}(k-1)$ is a vector containing the remaining $(k-m)$ elements. Subtracting (5.64) from (5.63), the latter of which has been first post-multiplied by a $m\times 1$ tap weight vector $\mathbf{w}(k-1)$, yields the following transformed estimation error vector

$$\begin{aligned}\overline{\mathbf{Q}}(k-1)&\boldsymbol{\Lambda}^{\frac{1}{2}}(k-1)\boldsymbol{\varepsilon}(k-1) \\ &= \begin{bmatrix} \mathbf{p}(k-1) \\ p_{m-1}(k-1) \\ \mathbf{v}(k-1) \end{bmatrix} - \begin{bmatrix} \mathbf{R}_{m-1,0}(k-1) & \mathbf{p}_{b,m-1}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B^{\frac{1}{2}}_{m-1}(k-1) \\ \mathbf{O}_{(k-m)\times(m-1)} & \mathbf{0} \end{bmatrix}\mathbf{w}(k-1),\end{aligned} \tag{5.65}$$

where $\boldsymbol{\varepsilon}(k-1) = [\varepsilon^*(0), \varepsilon^*(1), \ldots, \varepsilon^*(k-1)]^{\mathrm{T}} = \mathbf{d}(k-1) - \mathbf{X}_m(k-1)\cdot\mathbf{w}(k-1)$ is the error vector whose element $\varepsilon^*(k-1)$ is the error in estimating $d^*(k-1)$ of the desired response from a linear combination of $x^*(k-m), \ldots, x^*(k-1)$ when using the tap weight vector $\mathbf{w}(k-1)$ at time $(k-1)$.

The optimum tap weight vector, if desired, can be obtained from $\begin{bmatrix} \mathbf{p}(k-1) \\ p_{m-1}(k-1) \end{bmatrix}$ $= \begin{bmatrix} \mathbf{R}_{m-1,0}(k-1) & \mathbf{p}_{b,m-1}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B^{\frac{1}{2}}_{m-1}(k-1) \end{bmatrix}\mathbf{w}_o(k-1)$ using back-substitution and this choice of $\mathbf{w}_o(k-1)$ gives a minimized error vector as

$$\min_{\mathbf{w}_o(k-1)} \|\boldsymbol{\varepsilon}(k-1)\| = \|\mathbf{v}(k-1)\|. \tag{5.66}$$

Given new observation $x^*(k)$ along with the new desired response $d^*(k)$ received at time k and by appending the transformed desired data vector in (5.64) as the rightmost column of (5.63) followed by a sequence of $(m-1)$ Givens rotations proceeding rightwards from the $(k+1,1)$th element to the $(k+1,m-1)$th element that are used to annihilate the bottom row vector $\mathbf{x}^{\mathrm{H}}_{m-1}(k) = [x^*(k), x^*(k-1), \ldots, x^*(k-m+2)]$ yields

$$\mathbf{T}_{m-1}(k)\begin{bmatrix} \lambda^{\frac{1}{2}}\mathbf{R}_{m-1,0}(k-1) & \mathbf{p}_{b,m-1}(k-1) & \lambda^{\frac{1}{2}}\mathbf{p}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B_{m-1}^{\frac{1}{2}}(k-1) & \lambda^{\frac{1}{2}}p_{m-1}(k-1) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\mathbf{v}(k-1) \\ \mathbf{x}_{m-1}^{\mathrm{H}}(k) & x^*(k-m+1) & d^*(k) \end{bmatrix}$$
$$= \underbrace{\begin{bmatrix} \mathbf{R}_{m-1,0}(k) & \mathbf{p}_{b,m-1}(k) & \mathbf{p}(k) \\ \mathbf{0}^{\mathrm{T}} & \lambda^{\frac{1}{2}}B_{m-1}^{\frac{1}{2}}(k-1) & \lambda^{\frac{1}{2}}p_{m-1}(k-1) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\mathbf{v}(k-1) \\ \mathbf{0}^{\mathrm{T}} & \overline{\varepsilon}_{b,m-1}^*(k) & \overline{\varepsilon}_{m-1}^*(k) \end{bmatrix}}_{\mathbf{L}(k)}, \tag{5.67}$$

where the sequence of Givens rotations used in $\mathbf{T}_{m-1}(k) = J_0(k) \cdot J_1(k) \cdot \ldots \cdot J_{m-2}(k)$ turns out to be the same cosine and sine parameters used in the FP block. As a result of this transformation, both the $(m-1)$th-*order* angle-normalized backward prediction error $\overline{\varepsilon}_{b,m-1}^*(k)$ and the $(m-1)$th-*order* angle-normalized joint process estimation error $\overline{\varepsilon}_{m-1}^*(k)$ are produced.

The order-updated angle-normalized joint process estimation error $\overline{\varepsilon}_m^*(k)$ can be obtained by applying the following Givens rotation to matrix $\mathbf{L}(k)$ to annihilate the $(k+1,m)$th element and yields

$$\begin{bmatrix} \mathbf{I}_{m-1} & & & \\ & c_{b,m-1}(k) & & s_{b,m-1}^*(k) \\ & & \mathbf{I}_{k-m} & \\ & -s_{b,m-1}(k) & & c_{b,m-1}(k) \end{bmatrix}\mathbf{L}(k) = \underbrace{\begin{bmatrix} \mathbf{R}_{m-1,0}(k) & \mathbf{p}_{b,m-1}(k) & \mathbf{p}(k) \\ \mathbf{0}^{\mathrm{T}} & B_{m-1}^{\frac{1}{2}}(k) & p_{m-1}(k) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\mathbf{v}(k-1) \\ \mathbf{0}^{\mathrm{T}} & 0 & \overline{\varepsilon}_m^*(k) \end{bmatrix}}_{\mathbf{H}(k)}. \tag{5.68}$$

By taking out a scaling factor from each row of matrices $\mathbf{L}(k)$ and $\mathbf{H}(k)$, the rows before and after the Givens rotation in (5.68) is denoted, respectively, by

$$\mathbf{L}(k) = \mathbf{L}_1(k)\mathbf{L}_2(k) = diag\left[\ldots, \sqrt{\lambda B_{m-1}(k-1)}, \ldots, \sqrt{\gamma_{m-1}(k)}\right]$$
$$\begin{bmatrix} \overline{\mathbf{R}}_{m-1,0}(k) & \overline{\mathbf{p}}_{b,m-1}(k) & \overline{\mathbf{p}}(k) \\ \mathbf{0}^{\mathrm{T}} & 1 & \overline{p}_{m-1}(k-1) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\overline{\mathbf{v}}(k-1) \\ \mathbf{0}^{\mathrm{T}} & e_{b,m-1}^*(k) & e_{m-1}^*(k) \end{bmatrix}, \tag{5.69}$$

and

$$\mathbf{H}(k) = \mathbf{H}_1(k)\mathbf{H}_2(k) = diag\left[\ldots, \sqrt{B_{m-1}(k)}, \ldots, \sqrt{\gamma_m(k)}\right]$$
$$\begin{bmatrix} \overline{\mathbf{R}}_{m-1,0}(k) & \overline{\mathbf{p}}_{b,m-1}(k) & \overline{\mathbf{p}}(k) \\ \mathbf{0}^{\mathrm{T}} & 1 & \overline{p}_{m-1}(k) \\ \mathbf{O} & \mathbf{0} & \lambda^{\frac{1}{2}}\overline{\mathbf{v}}(k-1) \\ \mathbf{0}^{\mathrm{T}} & 0 & e_m^*(k) \end{bmatrix}. \tag{5.70}$$

Some elements of the matrices in (5.69) and (5.70) and the corresponding elements of the matrices in (5.19) and (5.20), respectively, can be related as follows: $k_a = \lambda B_{m-1}(k-1)$, $k_b = \gamma_{m-1}(k)$, $a_1 = 1$, $a_2 = \overline{p}_{m-1}(k-1)$, $b_1 = e_{b,m-1}^*(k)$, $b_2 = e_{m-1}^*(k)$, $k_a' = B_{m-1}(k)$, $k_b' = \gamma_m(k)$, $a_1' = 1$, $a_2' = \overline{p}_{m-1}(k)$, $b_1' = 0$, and

$b_2' = e_m^*(k)$. Substituting k_a, k_a', k_b, b_1, a_2, a_2', b_2, and b_2' into (5.22), (5.25), (5.28), and (5.29) (by letting $i = 2$) yields the following SRF joint process estimation, which is also summarized in Table 5.1.

SRF joint process estimation:

$$e_m(k) = e_{m-1}(k) - e_{b,m-1}(k)\overline{p}_{m-1}^*(k-1), \tag{5.71}$$

$$\overline{p}_{m-1}^*(k) = \overline{p}_{m-1}^*(k-1) + \overline{s}_{b,m-1}(k)e_m(k), \tag{5.72}$$

where $\overline{c}_{b,m-1}(k) = \frac{\lambda B_{m-1}(k-1)}{B_{m-1}(k)}$ and $\overline{s}_{b,m-1}(k) = \gamma_{m-1}(k)\frac{e_{b,m-1}^*(k)}{B_{m-1}(k)}$ in which $\gamma_m(k) = \gamma_{m-1}(k)\overline{c}_{b,m-1}(k)$. The *a posteriori* estimation error can be computed by $\varepsilon_m(k) = \gamma_m(k)e_m(k)$.

5.5 SRF (QRD-LSL)-Based RLS Algorithm

For many applications such as system identification, adaptive equalization, and active noise control, wherein the filter weights of the corresponding LS algorithm are required. The SRF QRD-LSL interpolation algorithm can be applied to implement the RLS algorithm in the transversal structure that generates the desired weight vector and the resulting algorithm is referred to as the SRF (QRD-LSL)-based RLS algorithm. The $(N+1)$st-*order* RLS algorithm with the tap weight vector at time k, $\mathbf{w}_{N+1}(k)$, can be calculated recursively in time using $\mathbf{w}_{N+1}(k) = \mathbf{w}_{N+1}(k-1) + \mathbf{k}_{N+1}(k)e^*(k)$ (see Table 2.5 in Chapter 2), where $\mathbf{k}_{N+1}(k) = \mathbf{R}^{-1}(k)\cdot\mathbf{x}_{N+1}(k)$ is the Kalman gain vector.

The Kalman gain vector can also be calculated as a particular set of normalized least-squares interpolation errors [23]:

$$\mathbf{k}_{N+1}^{\mathrm{T}}(k) = \left[\frac{\varepsilon_{N,0}^I(k)}{I_{N,0}(k)}, \ldots, \frac{\varepsilon_{p+1,f-1}^I(k-f+1)}{I_{p+1,f-1}(k-f+1)}, \frac{\varepsilon_{p,f}^I(k-f)}{I_{p,f}(k-f)}, \right.$$
$$\left. \frac{\varepsilon_{p-1,f+1}^I(k-f-1)}{I_{p-1,f+1}(k-f-1)}, \ldots, \frac{\varepsilon_{0,N}^I(k-N)}{I_{0,N}(k-N)}\right]. \tag{5.73}$$

Equation (5.73) can be proved as follows. From (5.4), we have $\mathbf{b}_{p,f}(k-f) = \mathbf{R}^{-1}(k)\mathbf{i}_{p,f}(k-f) = I_{p,f}(k-f)\cdot\mathbf{R}^{-1}(k)\left[\mathbf{0}_f^{\mathrm{T}}\ 1\ \mathbf{0}_p^{\mathrm{T}}\right]^{\mathrm{T}}$. Taking the Hermitian on both sides of the above equation and realizing that the correlation matrix is Hermitian [i.e., $\mathbf{R}^{\mathrm{H}}(k) = \mathbf{R}(k)$], we have

$$\left[\mathbf{0}_f^{\mathrm{T}}\ 1\ \mathbf{0}_p^{\mathrm{T}}\right] \cdot \mathbf{R}^{-1}(k) = \frac{1}{I_{p,f}(n-f)} \cdot \mathbf{b}_{p,f}^{\mathrm{H}}(n-f). \tag{5.74}$$

We then post-multiply both sides of the (5.74) by $\mathbf{x}_{N+1}(k)$. Using $\mathbf{k}_{N+1}(k) = \mathbf{R}^{-1}(k) \cdot \mathbf{x}_{N+1}(k)$ and (5.2) [by setting $i = k - f$ in (5.2)] yields the $(f+1)$st element of the Kalman gain vector.

$$[\mathbf{k}_{N+1}(k)]_{f+1} = \left[\mathbf{0}_f^{\mathrm{T}}\ 1\ \mathbf{0}_p^{\mathrm{T}}\right] \cdot \mathbf{R}^{-1}(k) \cdot \mathbf{x}_{N+1}(k) \tag{5.75}$$

$$= \frac{1}{I_{p,f}(k-f)} \cdot \mathbf{b}_{p,f}^{\mathrm{H}}(k-f) \cdot \mathbf{x}_{N+1}(k) = \frac{\varepsilon_{p,f}^{I}(k-f)}{I_{p,f}(k-f)} \tag{5.76}$$

By adjusting the values of p and f with $p + f = N$, all the elements of the Kalman gain vector in (5.73) can be calculated. Equation (5.74) reveals the connection between linear interpolation and the inverse of the time-average correlation matrix that is required to obtain the Kalman gain vector.

The Kalman gain vector is the most decorrelated version of the normalized input vector, because each element of the Kalman gain vector in (5.73) is the optimum two-sided prediction residual of each corresponding element of the input vector $\mathbf{x}_{N+1}(k)$. Accordingly, the Kalman gain vector corresponds to the least redundant version of the input vector. Therefore, the elements of $\mathbf{k}_{N+1}(k)$ may be responsible for the fast convergence of the RLS algorithm. The SRF QRD-LSL interpolation algorithm can be used to calculate $\mathbf{k}_{N+1}(k)$ in (5.73) in an *order-recursive* manner through a divide and conquer method [23]; the resulting SRF (QRD-LSL)-based RLS algorithm requires $\mathscr{O}[N \log_2 N]$ operations for a transversal filter of order N.

5.6 Simulations

All simulations were run on a PC, in a floating-point environment, with 10-tap weights. The effective number of mantissa bits in the floating-point representation is reduced to observe the finite-precision effects by truncating the mantissa at a predefined position without affecting the exponent. Three algorithms were applied to perform adaptive prediction of an autoregressive (AR) process that closely follows the one presented in [13], and their numerical robustness were evaluated in this scenario. The observation is $x(k) = x_{AR}(k) + w_1(k)$, where $x_{AR}(k) = -\sum_{i=1}^{10} a_i x_{AR}(k-i) + w_2(k)$ is an AR process of order 10 with an eigenvalue spread approximately 1020. Both $w_1(k)$ and $w_2(k)$ are independent white noise processes with zero mean. The variance of $w_2(k)$ is set to cause $x_{AR}(k)$ to have a variance of 0 dB, whereas the variance of $w_1(k)$ is chosen such that the signal-to-noise ratio is 30 dB. The desired response is therefore $x(k)$ while the observations are $[x(k-1), x(k-2), \ldots, x(k-10)]$.

Figure 5.2 compares three learning curves obtained by ensemble-averaging the prediction error squares over 200 independent experimental trials. The three learning

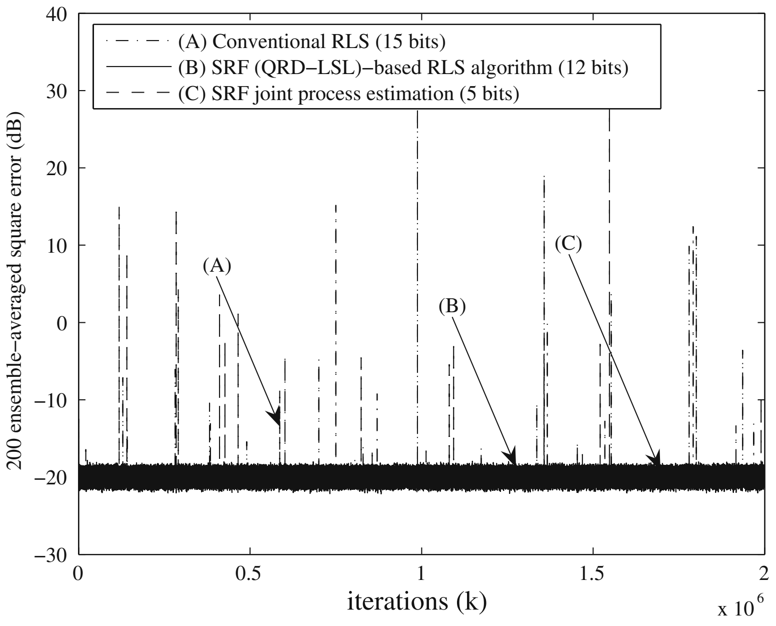

Fig. 5.2 Learning curves of three algorithms in a finite-precision environment.

curves are the results of using a conventional RLS algorithm of $\mathcal{O}[N^2]$ complexity in Table 2.5 (in Chapter 2), the SRF (QRD-LSL)-based RLS algorithm of $\mathcal{O}[N\log_2 N]$ complexity, and the joint process estimation of $\mathcal{O}[N]$ complexity in Table 5.1 without computing the optimal tap weight vector, all with a forgetting factor of $\lambda = 0.996$ and a regularization parameter $\delta = 0.004$. Our simulations demonstrated that when exact arithmetic is used, all three algorithms yielded exactly the same result, but they exhibited various unstable behaviors with finite-precision computation. While the conventional RLS algorithm became unacceptable for less than 16 mantissa bits, the SRF (QRD-LSL)-based RLS algorithm with 12 mantissa bits ran successfully for two million iterations (at which point the experiment was terminated). Although the mean square error produced by the joint process estimation based on the QRD-LSL prediction algorithm is a little higher than that of the SRF (QRD-LSL)-based RLS algorithm, it still produces useful results with only five mantissa bits. This is because the joint process section is subordinate to the prediction section, and the numerical stability of the joint process estimation thus depends on the SRF QRD-LSL prediction algorithm. It can be shown that the computations of the conventional forward and backward prediction errors in the SRF QRD-LSL prediction algorithm involve the error feedback mechanism described in Section 5.3, and therefore their error growth in a finite-precision environment is always bounded. For example, the computation of the forward prediction error summarized in the FP block in Table 5.1 is rewritten as

$$e_{f,m}(k) = e_{f,m-1}(k) - e_{b,m-1}(k-1)\overline{\pi}^{*}_{f,m}(k-1), \tag{5.77}$$

$$\overline{\pi}^{*}_{f,m}(k) = \overline{\pi}^{*}_{f,m}(k-1) + \overline{s}_{b,m-1}(k-1)e_{f,m}(k). \tag{5.78}$$

A unique feature of the error feedback mechanism is that the *a priori* prediction error $e_{f,m}(k)$ computed in (5.77) is fed back to time-update the forward reflection coefficient $\overline{\pi}^{*}_{f,m}(k)$ whose cumulative error is bounded for all time in a finite-precision environment. However, as mentioned in Section 5.4, the computations of the intermediate forward and backward prediction errors from the conventional forward and backward prediction errors in the two equation pairs using (5.49), (5.46) and (5.50), (5.48) in the QRD-LSL interpolation algorithm, correspond to the reversed formulations. These reversed formulations no longer involve an error feedback mechanism, potentially causing numerical instability in the QRD-LSL interpolation algorithm in a finite-precision environment. It has been shown by Skidmore and Proudler [23] that the error growth in this reversed formulation is *linear* with time in nature in a finite-precision environment (i.e., the errors accumulate but are not amplified by each iteration). This linear error growth may have resulted in the divergence of the SRF (QRD-LSL)-based RLS algorithm before two million iterations are completed when less than 12 mantissa bits is used. However, this linear error growth should be contrasted to the exponential error growth with time exhibited by the fast transversal filter (FTF) algorithm [32] that may much more rapidly diverge from the correct solution.

5.7 Conclusion

This chapter develops the SRF QRD-LSL interpolation algorithm, which is then reduced to the SRF QRD-LSL prediction algorithm, which is in turn extended to develop the SRF joint process estimation. As described in [32, 33], no exact-RLS algorithm, whether in lattice or transversal filter, can always be stable, because of limited-precision instabilities. The RLS algorithm based on the SRF QRD-LSL interpolation algorithm is no exception, and it may diverge in a finite-precision environment due to error accumulation. Simulations indicated that the SRF (QRD-LSL)-based RLS algorithm using only eight mantissa bits, started to diverge from the correct solution around $k = 2 \times 10^5$ in the computer experiment conducted in Section 5.6, because of the two equation pairs, (5.49), (5.46) and (5.50), (5.48), computed in the QRD-LSL interpolation algorithm. In contrast, the computations of the forward and backward prediction errors in the SRF QRD-LSL prediction algorithm involve an error feedback mechanism, and therefore the corresponding error growth in a finite-precision environment is always bounded. Consequently, the joint process estimation based on the SRF QRD-LSL prediction algorithm still exhibits well-conditioned behaviors with only five mantissa bits. However, the joint process estimation does not generate the filter weights, which are required in some applications.

Skidmore and Proudler [23] employed exactly the same concept as demonstrated by the two equation pairs, (5.49), (5.46) and (5.50), (5.48), to implement an SRF QR-RLS algorithm, referred to as the KaGE RLS algorithm. Although the KaGE RLS algorithm, like the SRF (QRD-LSL)-based RLS algorithm, may exhibit linear error growth with time in a finite-precision environment owing to the computation of the two equation pairs described above, simulation results presented in [23] reveal that the KaGE RLS algorithm can run reliably for many millions of iterations using single precision arithmetic and is inherently much more stable than the stabilized FTF algorithm proposed by Maouche and Slock [34]. Therefore, as suggested in [23], to generate the transversal filter weights, the KaGE algorithm as well as the SRF (QRD-LSL)-based RLS algorithm presented in this chapter, both employing interpolation residuals and both of $\mathscr{O}[N\log_2 N]$ complexity, may offer a favorable compromise between the computationally efficient FTF algorithms of $\mathscr{O}[N]$ complexity and the stable algorithms of $\mathscr{O}[N^2]$ complexity, such as the Inverse QRD-RLS algorithm proposed by Alexander and Ghirnikar [35]. The Inverse QRD-RLS algorithm may not be computationally feasible for some real-time implementations of long adaptive filters.

References

1. D. T. Lee, M. Morf, and B. Friedlander, Recursive least squares ladder estimation algorithms. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-29, no. 3, pp. 627–641 (June 1981)
2. L. J. Griffiths, An adaptive lattice structure for noise-cancelling applications. IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP'78, Tulsa, USA, pp. 873–90 (April 1978)
3. J. Makhoul, A class of all-zero lattice digital filters: Properties and applications. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-26, pp. 304–314 (August 1978)
4. P. Strobach, Linear Prediction Theory – A Mathematical Basis for Adaptive Systems. Springer-Verlag, Berlin, Germany (1990)
5. S. Haykin, Adaptive Filter Theory. 4th edition Prentice-Hall, Englewood Cliffs, NJ, USA (2002)
6. A. H. Sayed, Fundamentals of Adaptive Filtering. John Wiley, NJ, USA (2003)
7. F. Capman, J. Boudy, and P. Lockwood, Acoustic echo cancellation using a fast QR-RLS algorithm and multirate schemes. IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP'95, Detroit, USA, pp. 969–972 (May 1995)
8. J. M. Cioffi, The fast adaptive ROTOR's RLS algorithm. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-38, no. 4, pp. 631–653 (April 1990)
9. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, QRD-based lattice filter algorithms. SPIE Conference on Advanced Algorithms and Architectures for Signal Processing, San Diego, USA, vol. 1152, pp. 56–67 (August 1989)
10. F. Ling, Givens rotation based least squares lattice and related algorithms. IEEE Transactions on Signal Processing, vol. 39, no. 7, pp. 1541–1551 (July 1991)
11. P. A. Regalia and M. G. Bellanger, On the duality between fast QR methods and lattice methods in least squares adaptive filtering. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 879–891 (April 1991)

12. A. A. Rontogiannis and S. Theodoridis, New fast QR decomposition least squares adaptive algorithms. IEEE Transactions on Signal Processing, vol. 46, no. 8, pp. 2113–2121 (August 1998)
13. B. Yang and J. F. Böhme, Rotation-based RLS algorithms: unified derivations, numerical properties, and parallel implementations. IEEE Transactions on Signal Processing, vol. 40, no. 5, pp. 1151–1167 (May 1992)
14. A. K. Jain, Image coding via a nearest neighbors image model. IEEE Transactions on Communications, vol. COMM-23, no. 3, pp. 318–331 (March 1975)
15. E. A. Gifford, B. R. Hunt, and M. W. Marcellin, Image coding using adaptive recursive interpolative DPCM. IEEE Transactions on Image Processing, vol. 4, no. 8, pp. 1061–1069 (August 1995)
16. G. Yang, H. Leich, and R. Boite, Voiced speech coding at very low bit rates based on forward–backward waveform prediction. IEEE Transactions on Speech and Audio Processing, vol. 3, no. 1, pp. 40–47 (January 1995)
17. E. Masry, Closed-form analytical results for the rejection of narrow-band interference in PN spread spectrum systems-Part II: linear interpolation filters. IEEE Transactions on Communications, vol. COMM-33, no. 1, pp. 10–19 (January 1985)
18. J.-T. Yuan and J.-N. Lee, Narrowband interference rejection in DS/CDMA systems using adaptive (QRD-LSL)-based nonlinear ACM interpolators. IEEE Transactions on Vehicular Technology, vol. 52, no. 2, pp. 374–379 (March 2003)
19. S. Kay, Some results in linear interpolation theory. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-31, no. 3, pp. 746–749 (June 1983)
20. B. Picinobono and J. M. Kerilis, Some properties of prediction and interpolation errors. IEEE Transactions on Acoustic, Speech, and Signal Processing, vol. ASSP-36, no. 4, pp. 525–531 (April 1988)
21. C. K. Coursey and J. A. Stuller, Linear interpolation lattice. IEEE Transactions on Signal Processing, vol. 39, no. 4, pp. 965–967 (April 1991)
22. J.-T. Yuan, QR-decomposition-based least-squares lattice interpolators. IEEE Transactions on Signal Processing, vol. 48, no. 1, pp. 70–79 (January 2000)
23. I. D. Skidmore and I. K. Proudler, The KaGE RLS algorithm. IEEE Transactions on Signal Processing, vol. 51, no. 12, pp. 3094–3104 (December 2003)
24. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, The QRD-based least squares lattice algorithm: Some computer simulations using finite wordlengths. IEEE International Symposium on Circuits and Systems, ISCAS'90, New Orleans, USA, pp. 258–261 (May 1990)
25. S. F. Hsieh, K. J. R. Liu, and K. Yao, A unified square-root-free approach for QRD-based recursive least squares estimation. IEEE Transactions on Signal Processing, vol. 41, no. 3, pp. 1405–1409 (March 1993)
26. F. Ling, D. Manolakis, and J. G. Proakis, Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-34, no. 4, pp. 837–845 (August 1986)
27. I. K. Proudler, J. G. McWhirter, and T. J. Shepherd, Computationally efficient QR-decomposition approach to least squares adaptive filtering. IEE Proceedings-F, vol. 138, no. 4, pp. 341–353 (August 1991)
28. W. M. Gentleman, Least squares computations by Givens transformations without square roots. IMA Journal of Applied Mathematics, vol. 12. pp. 329–336 (December 1973)
29. J. G. McWhirter, Recursive least-squares minimization using a systolic array. SPIE Real-Time Signal Processing VI, vol. 431, pp. 105–112 (January 1983)
30. J.-T. Yuan, C.-A. Chiang, and C.-H. Wu, A square-root-free QRD-LSL interpolation algorithm. IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP'2008, Las Vegas, USA, pp. 3813–3816 (April 2008)
31. C.-A. Chiang, A recursive least-squares (RLS) algorithm based on interpolation lattice recursive structure. M.S. thesis, Adviser: J.-T. Yuan, Fu Jen Catholic University, Taipei, Taiwan, R.O.C. (April 2006)

32. J. M. Cioffi and T. Kailath, Fast, recursive-least-squares transversal filters for adaptive filtering. IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-32, no. 2, pp. 304–337 (April 1984)
33. S. Ljung, Fast algorithms for integral equations and least-squares identification problems. Ph.D. thesis, Linkoping University, Sweden (1983)
34. K. Maouche and D. T. M. Slock, Fast subsampled-updating stabilized fast transversal filter (FSU SFTF) RLS algorithm for adaptive filtering. IEEE Transactions on Signal Processing, vol. 48, no. 8, pp. 2248–2257 (August 2000)
35. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. IEEE Transactions on Signal Processing, vol. 41, no. 1, pp. 20–30 (January 1993)