Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control. James C. Spall
Copyright © 2003 John Wiley & Sons, Inc.,
ISBN: 0-471-33052-3

CHAPTER 3

RECURSIVE ESTIMATION FOR LINEAR MODELS

This chapter describes recursive optimization algorithms for linear models that have static or time-varying parameters $\boldsymbol{\theta}$. Consistent with the discussion of Section 1.1, the use of *recursive* here is meant to suggest that the data arrive over time, and that the algorithm processes the data approximately as they arrive. In fact, however, the algorithms can also be applied as general iterative methods for "off-line" analysis. Linear models are considered here for two reasons: (i) the models are popular and the associated recursive algorithms are widely used and (ii) the algorithms provide motivation for the more general nonlinear stochastic approximation framework to be introduced in Chapter 4. Broadly speaking, the linear models here lead to loss functions $L = L(\boldsymbol{\theta})$ that are quadratic in $\boldsymbol{\theta}$. So, the algorithms of this chapter are oriented to finding effective means for solving quadratic optimization problems with possibly noisy loss function or associated gradient information. This chapter marks the beginning of an emphasis on continuous-valued $\boldsymbol{\theta}$ problems that will last through Chapter 7.

Section 3.1 summarizes some essential properties of the linear model and main estimation criterion, introducing the estimation problem via consideration of a fixed (static) $\boldsymbol{\theta}$ vector. Section 3.2 describes some popular recursive methods for estimating a static $\boldsymbol{\theta}$. Section 3.3 extends the ideas in Sections 3.1 and 3.2 to the problem of estimating time-varying $\boldsymbol{\theta}$. This discussion includes the celebrated Kalman filter. Section 3.4 is a case study devoted to demonstrating the recursive methods on a set of data associated with a musical instrument—the oboe. Section 3.5 offers some concluding remarks. Chapter 13 considers some general issues associated with building and analyzing mathematical models, including models of the type here.

3.1 FORMULATION FOR ESTIMATION WITH LINEAR MODELS

This section focuses on the parameter estimation problem when it is assumed that there is no change over time in the underlying true values of the parameters $\boldsymbol{\theta}$. The first subsection in this section summarizes the principal linear model of this chapter and some of its applications. (Some extensions of this model are considered in Section 3.3.) The second subsection discusses the least-squares estimation criterion of interest, showing the connection of this criterion to an idealized (but infeasible) mean-squared error criterion. This subsection discusses classical least squares (essentially a deterministic optimization approach) and then makes connections to the stochastic optimization setting of prime interest in this book.

3.1.1 Linear Model

The model of interest here has the classical linear form

$$z_k = \boldsymbol{h}_k^T \boldsymbol{\theta} + v_k , \tag{3.1}$$

where z_k is the kth measured output (assumed for now to be a scalar), $\boldsymbol{h}_k$ is the corresponding *design vector*, v_k is the unknown noise value, and $\boldsymbol{\theta}$ represents the parameter vector to be estimated. Because we will be interested in squared-error-type loss functions, the linearity in $\boldsymbol{\theta}$ in (3.1) implies a loss function that is quadratic in $\boldsymbol{\theta}$. This form of criterion has a single global (= local) minimum, eliminating the concern about finding a global minimum among many local minima.

Of course, few *real* systems truly produce responses in the clean linear form of (3.1), but as a reasonable approximation of reality, model (3.1) has been used in countless applications. In fact, it is more general than it may first appear in that it also accommodates a certain type of nonlinearity—namely, the so-called curvilinear models where the model contribution $\boldsymbol{h}_k^T \boldsymbol{\theta}$ represents a sum of *nonlinear* functions of input variables (appearing within $\boldsymbol{h}_k$). The nonlinear summands are weighted by the elements of $\boldsymbol{\theta}$. Let us illustrate both pure linear and curvilinear models in the examples below. Another example of both a pure linear and curvilinear model is given in the case study of Section 3.4. The texts by Widrow and Stearns (1985), Haykin (1996), Neter et al. (1996), Ljung (1999), and Moon and Stirling (2000) are several of *many* books in a variety of fields that present detailed analysis and excellent examples of model (3.1). Chapters 13 and 17 here also consider aspects of model (3.1) as related to model selection and experimental design.

Example 3.1—Time series model as an example of (3.1). A large number of systems have been successfully analyzed with models described by a linear (or affine) relationship between input variables and output variables. Linear time series models describing the state of a system as a function of previous states are one important class of linear models. (Here, *state* refers to a vector summarizing the essential characteristics of the system *relative to the needs of the analysis.*) For example, for a macroeconomist, a state vector of interest might include—among other quantities—the gross national product and inflation rate, whereas to a physicist, a state vector might be the position, velocity, and acceleration of a moving object. More specifically, autoregressive (AR) models are an important subset of time series models. Scalar AR models have the form

$$x_{k+1} = \beta_0 x_k + \beta_1 x_{k-1} + \ldots + \beta_m x_{k-m} + w_k,$$

where x_k represents the (scalar) state of the system at time point k, β_i is the coefficient describing the effect of the state at an earlier time $k - i$ (i.e., x_{k-i}) on the next state x_{k+1}, and w_k is the input noise. This AR form fits into the framework of the linear model in (3.1) by letting $z_k = x_k$, $v_k = w_{k-1}$, $\boldsymbol{\theta} = [\beta_0, \beta_1, \ldots, \beta_m]^T$, and $\boldsymbol{h}_k = [x_{k-1}, x_{k-2}, \ldots, x_{k-m-1}]^T$. (Note that there is no dependence of $\boldsymbol{h}_k$ on the unknown variables $\boldsymbol{\theta}$, which would make model (3.1) *nonlinear* in $\boldsymbol{\theta}$. The lagged values x_{k-i} in $\boldsymbol{h}_k$ represent *actual* values of the physical system. We are only *modeling* those values as depending on $\boldsymbol{\theta}$. Even if the AR model form were precisely correct, the physical values x_{k-i} would be generated by some *fixed* value $\boldsymbol{\theta} = \boldsymbol{\theta}^*$.) ❑

Example 3.2—Nonlinear model transformed to linear model (3.1). Let us now present an example of a model that is nonlinear in both the parameters to be estimated and the input variables, but which can be converted to a linear model by a simple transformation. Suppose that the scalar output, say q_k, for the kth set of m inputs $x_{k1}, x_{k2}, \ldots, x_{km}$ is given by the multiplicative model

$$q_k = \beta_0 x_{k1}^{\beta_1} x_{k2}^{\beta_2} \cdots x_{km}^{\beta_m} \xi_k,$$

where the β_i, $i = 0, 1, \ldots, m$, represent the parameters of the model to be estimated and ξ_k is a multiplicative (positive) noise term. Taking logarithms of both sides of this expression yields

$$\log q_k = \log \beta_0 + \beta_1 \log x_{k1} + \beta_2 \log x_{k2} + \ldots + \beta_m \log x_{km} + \log \xi_k,$$

which is *linear* in a redefined set of parameters and input variables. In particular, setting $z_k = \log q_k$, $v_k = \log \xi_k$, $\boldsymbol{\theta} = [\log \beta_0, \beta_1, \beta_2, \ldots, \beta_m]^T$, and $\boldsymbol{h}_k = [1, \log x_{k1}, \log x_{k2}, \ldots, \log x_{km}]^T$ leads to a model of the form (3.1). There is no loss of information about the original parameters and variables in this logarithmic transformation.

The multiplicative model, for example, appears in microeconomics as the Cobb–Douglas production function (e.g., Section 4.2 here; Kmenta, 1997, pp. 252–253). This function measures the output (q_k) of a production facility as a function of relevant inputs where the parameters $\beta_1, \beta_2, \ldots, \beta_m$ reflect the relative importance of the various inputs. Here k is either an index across time for a given production facility or an index across multiple facilities of the same type. One special case is when $m = 2$ with x_{k1} representing a summary measure of labor input and x_{k2} representing a summary measure of capital equipment. In some applications of the Cobb–Douglas function, the effective dimension of $\boldsymbol{\theta}$ is reduced from $p = m+1$ to $p = m$ because there is a constraint $1 = \beta_1 + \beta_2 + \ldots + \beta_m$ (i.e., once an estimate for $m-1$ of the m parameters $\beta_1, \beta_2, \ldots, \beta_m$ is available, the remaining estimate is directly available from the constraint). ❑

Example 3.3—Polynomial class of curvilinear models for use in (3.1). Suppose that the observed output of a process is modeled as a sum of additive noise and an *m*th-order polynomial of a scalar input variable x. Let the *m*th-order polynomial be $\beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_m x^m$ where the β_i, $i = 0, 1, \ldots, m$, represent the parameters to be estimated. This fits into the framework of the linear model in (3.1), where repeated values of x, indexed as x_k, are used to generate outputs for estimating the β_i. In particular, the connection to (3.1) is apparent by letting $\boldsymbol{\theta} = [\beta_0, \beta_1, \beta_2, \ldots, \beta_m]^T$ and $\boldsymbol{h}_k = [1, x_k, x_k^2, \ldots, x_k^m]^T$. The additive noise for the *k*th measurement is represented by v_k.

A simple specific example of this curvilinear (polynomial) model is the motion of a particle along a line where time τ represents the fundamental input variable. From elementary physics, the position, say $\pi(\tau)$, of a particle undergoing constant acceleration is given by the formula $\pi(\tau) = \pi_0 + V_0\tau + A_0\tau^2/2$, where π_0 is the initial ($\tau = 0$) position, V_0 is the initial velocity, and A_0 is the constant acceleration. Suppose there is instrumentation that provides noisy measurements $z_1, z_2, \ldots$ of the position $\pi(\tau)$ at discrete times $\tau_1, \tau_2, \ldots$ (so $z_k = \pi(\tau_k) + v_k$). One may estimate the initial position and velocity and the constant acceleration from these noisy measurements by letting $\boldsymbol{\theta} = [\pi_0, V_0, A_0]^T$ and $\boldsymbol{h}_k = [1, \tau_k, \tau_k^2/2]^T$. ❑

3.1.2 Mean-Squared and Least-Squares Estimation

This subsection presents the mean-squared and least-squares estimation criteria and discusses the batch least-squares estimate for model (3.1). The aim is to estimate $\boldsymbol{\theta}$ from a sequence of input–output pairs $\{\boldsymbol{h}_k, z_k\}$. In deriving the mean-squared criterion, suppose that $z_k = \boldsymbol{h}_k^T\boldsymbol{\theta}^* + v_k$, where $\boldsymbol{\theta}^*$ is the unknown true value of $\boldsymbol{\theta}$ (so that the model and the true system have the same functional form). Under the assumption that the noises v_k have mean zero, a natural conceptual criterion for estimation is the mean-squared error

$$L(\boldsymbol{\theta}) = \frac{1}{n}E\left[\tfrac{1}{2}\sum_{k=1}^{n}(z_k - \boldsymbol{h}_k^T\boldsymbol{\theta})^2\right]$$

$$= \frac{1}{2n}\sum_{k=1}^{n}E\left[(\boldsymbol{h}_k^T\boldsymbol{\theta}^* + v_k - \boldsymbol{h}_k^T\boldsymbol{\theta})^2\right], \qquad (3.2)$$

where the immaterial factor 1/2 is included to facilitate later derivative calculations and the division by n is a normalization so that (3.2) reflects the *average* error over the n measurements. The expectations shown in (3.2) are with respect to randomness in v_k; the expectation may also be with respect to $\boldsymbol{h}_k$ if the inputs $\boldsymbol{h}_k$ are random. (Estimation based on the inclusion of weighted summands—i.e., the implied multiplier of unity on $(z_k - \boldsymbol{h}_k^T\boldsymbol{\theta})^2$ is replaced by a scale factor changing with k—is discussed in Subsection 3.2.6 and Section 3.3.)

Because $L(\boldsymbol{\theta})$ is a quadratic function of $\boldsymbol{\theta}$ (i.e., there is one unique minimum), the solution is available by setting the gradient $\boldsymbol{g}(\boldsymbol{\theta}) = \partial L/\partial\boldsymbol{\theta} = \boldsymbol{0}$ when the problem is unconstrained (i.e., $\Theta = \mathbb{R}^p$). If var(v_k) is independent of $\boldsymbol{\theta}$ and the $\boldsymbol{h}_k$ are deterministic, the derivative of the kth summand above (ignoring the immaterial division by n for the moment) is

$$\frac{\partial E\left[(\boldsymbol{h}_k^T\boldsymbol{\theta}^* + v_k - \boldsymbol{h}_k^T\boldsymbol{\theta})^2\right]/2}{\partial\boldsymbol{\theta}} = \frac{\partial(\boldsymbol{h}_k^T\boldsymbol{\theta}^* - \boldsymbol{h}_k^T\boldsymbol{\theta})^2/2}{\partial\boldsymbol{\theta}} = -\boldsymbol{h}_k(\boldsymbol{h}_k^T\boldsymbol{\theta}^* - \boldsymbol{h}_k^T\boldsymbol{\theta}).$$

Hence, an estimate, in principle, is available by solving for $\boldsymbol{\theta}$ such that

$$\sum_{k=1}^{n}\boldsymbol{h}_k(\boldsymbol{h}_k^T\boldsymbol{\theta}^* - \boldsymbol{h}_k^T\boldsymbol{\theta}) = \boldsymbol{0}.$$

This, however, is not useful for implementation since it depends on the very quantity that one is trying to find ($\boldsymbol{\theta}^*$). In other words, one has to know $\boldsymbol{\theta}^*$ to find $\boldsymbol{\theta}^*$! So, although the mean-squared error criterion is a desirable *conceptual* criterion, it does not lead to a usable solution strategy.

The method of least squares is a practical alternative to the infeasible method based on the mean-squared error criterion. This method works with a criterion, say $\hat{L}(\boldsymbol{\theta})$, that is an unbiased estimate of $L(\boldsymbol{\theta})$ in (3.2). The least-squares loss function is

$$\begin{aligned}\hat{L}(\boldsymbol{\theta}) &= \frac{1}{2n}\sum_{k=1}^{n}(z_k - \boldsymbol{h}_k^T\boldsymbol{\theta})^2 \\ &= \frac{1}{2n}(\boldsymbol{Z}_n - \boldsymbol{H}_n\boldsymbol{\theta})^T(\boldsymbol{Z}_n - \boldsymbol{H}_n\boldsymbol{\theta}),\end{aligned} \tag{3.3}$$

where $\boldsymbol{Z}_n = [z_1, z_2, \ldots, z_n]^T$ and $\boldsymbol{H}_n$ is the $n \times p$ concatenated matrix of $\boldsymbol{h}_k^T$ row vectors. This function is a relatively simple criterion that is quadratic in the elements of $\boldsymbol{\theta}$ but, unlike the mean-squared criterion (3.2), does not depend on the unknown $\boldsymbol{\theta}^*$. Further, the minimum of $\hat{L}(\boldsymbol{\theta})$ is not generally $\boldsymbol{\theta}^*$, but is a point tied to the specific observed data $\boldsymbol{Z}_n$. As n gets large, the minimum of $\hat{L}(\boldsymbol{\theta})$ approaches (in a stochastic sense) $\boldsymbol{\theta}^*$ under standard conditions.

Criterion (3.3) has some desirable connections to the mean-squared criterion. In particular, for any $\boldsymbol{\theta}$, the criterion and its gradient are unbiased estimators of the mean-squared criterion and its gradient. That is,

$$E[\hat{L}(\boldsymbol{\theta})] = L(\boldsymbol{\theta}), \tag{3.4a}$$

$$E\left[\frac{\partial \hat{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right] = \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{3.4b}$$

Result (3.4b) follows since $\frac{1}{2}E[\partial(z_k - \boldsymbol{h}_k^T\boldsymbol{\theta})^2/\partial\boldsymbol{\theta}] = -\boldsymbol{h}_k(\boldsymbol{h}_k^T\boldsymbol{\theta}^* - \boldsymbol{h}_k^T\boldsymbol{\theta})$, which corresponds to the derivatives of the summands in (3.2) that form $\partial L/\partial\boldsymbol{\theta}$. This is an example of an interchange of derivative and integral since $\partial L/\partial\boldsymbol{\theta}$ on the right-hand side of (3.4b) can be written as $\partial E[\hat{L}(\boldsymbol{\theta})]/\partial\boldsymbol{\theta}$. Therefore, unlike some general nonlinear problems considered in Chapter 5 and elsewhere, the relatively simple quadratic structure here does not require the general machinery given in Appendix A to justify the interchange.

By (3.4a, b), optimizing based on the least-squares criterion is equivalent to optimizing an unbiased estimate of the mean-squared criterion. Further, optimizing based on setting $\partial\hat{L}/\partial\boldsymbol{\theta} = \boldsymbol{0}$ is equivalent to root finding with an unbiased estimate of $\partial L/\partial\boldsymbol{\theta}$. These connections of least-squares estimation to minimum mean-squared estimation can be used to make formal statements about the convergence of the recursive estimates to $\boldsymbol{\theta}^*$ via stochastic approximation theory (see, especially, Sections 4.3 and 5.1).

The close connection of the desirable (but infeasible) minimum mean-squared error estimation to the feasible least-squares estimation provides the rationale for focusing on the least-squares criterion (3.3). A unique solution to the problem of estimating $\boldsymbol{\theta}$ is readily available if $n \geq p$ (so that there are at least as many data points as parameters to be estimated in $\boldsymbol{\theta}$). The form (3.3) is most appropriate when one assumes that $E(v_k) = 0$ and the var(v_k) terms are identical for all k (but not necessarily known). Criterion (3.3) leads to the ordinary least-squares solution. This solution follows by setting $\partial\hat{L}/\partial\boldsymbol{\theta} = \boldsymbol{0}$ and solving the resulting "normal equations" to yield the batch least-squares estimate for $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}}^{(n)} = (\boldsymbol{H}_n^T\boldsymbol{H}_n)^{-1}\boldsymbol{H}_n^T\boldsymbol{Z}_n, \tag{3.5}$$

assuming, of course, that the indicated matrix inverse exists. (The notation $\hat{\boldsymbol{\theta}}^{(n)}$ is used here and elsewhere to denote a batch estimate based on n data points. In contrast, the notation $\hat{\boldsymbol{\theta}}_n$ is generally used to denote a recursive estimate based on n iterations.) Suppose that var(v_k) = σ^2 for all k and that, as with the mean-squared criterion (3.2), the model form (3.1) is accurate in the sense that the physical data are generated by the "true process" $z_k = \boldsymbol{h}_k^T\boldsymbol{\theta}^* + v_k$. Then, two important properties of the batch solution are that it is unbiased (i.e., $E(\hat{\boldsymbol{\theta}}^{(n)}) = \boldsymbol{\theta}^*$) and has covariance matrix $\text{cov}(\hat{\boldsymbol{\theta}}^{(n)}) = (\boldsymbol{H}_n^T\boldsymbol{H}_n)^{-1}\sigma^2$ (Exercise 3.1).

If there is a symmetric weighting matrix W_n in the inner product of (3.3) (i.e., the second line of (3.3) becomes $(\boldsymbol{Z}_n - \boldsymbol{H}_n\boldsymbol{\theta})^T W_n(\boldsymbol{Z}_n - \boldsymbol{H}_n\boldsymbol{\theta})/(2n)$), then the solution in (3.5) for ordinary least squares is replaced by the weighted least-squares solution. Such a weighting matrix is typically used when the variances of the v_k terms are not identical for all k. In such a solution, the two $\boldsymbol{H}_n^T$ terms in (3.5) are replaced by $\boldsymbol{H}_n^T W_n$. We give a general form for the recursive implementation of least squares under a weighting matrix with multivariate z_k in Subsection 3.2.6.

Figure 3.1 provides a geometric interpretation of the least-squares estimate. In particular, the famous orthogonal projection principle (e.g., Sorenson, 1980, pp. 35–36; Jang et al., 1997, pp. 110–112; Moon and Stirling, 2000, pp. 114–115) states that the error in the predicted $\boldsymbol{Z}_n$, as given by the predictor $\boldsymbol{H}_n\hat{\boldsymbol{\theta}}^{(n)}$ (i.e., the error $\boldsymbol{Z}_n - \boldsymbol{H}_n\hat{\boldsymbol{\theta}}^{(n)}$), is orthogonal to each of the $n \times 1$ columns in $\boldsymbol{H}_n$. For convenience in Figure 3.1 and the following discussion, let us suppress the sub/superscript n. Formally, the orthogonality is then expressed as

$$\boldsymbol{H}^T(\boldsymbol{Z} - \boldsymbol{H}\hat{\boldsymbol{\theta}}) = \boldsymbol{0}.$$

One can see the origins of this relationship by observing that the prediction $\boldsymbol{H}\hat{\boldsymbol{\theta}}$ must lie in the hyperplane spanned by the p columns, say $\boldsymbol{h}_{\bullet j}$, in $\boldsymbol{H} \equiv [\boldsymbol{h}_{\bullet 1}, \boldsymbol{h}_{\bullet 2}, \ldots, \boldsymbol{h}_{\bullet p}]$ (the notation $\boldsymbol{h}_{\bullet j}$ is chosen to avoid confusion with the jth design vector $\boldsymbol{h}_j$ from (3.1)). In particular, with $\boldsymbol{\theta} = [t_1, t_2, \ldots, t_p]^T$:

$$\boldsymbol{H}\boldsymbol{\theta} = [\boldsymbol{h}_{\bullet 1}, \boldsymbol{h}_{\bullet 2}, \ldots, \boldsymbol{h}_{\bullet p}]\begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_p \end{bmatrix} = t_1[\boldsymbol{h}_{\bullet 1}] + t_2[\boldsymbol{h}_{\bullet 2}] + \ldots + t_p[\boldsymbol{h}_{\bullet p}],$$

indicating that the prediction $\boldsymbol{H}\hat{\boldsymbol{\theta}}$ is a linear combination of the basis vectors $[\boldsymbol{h}_{\bullet 1}, \boldsymbol{h}_{\bullet 2}, \ldots, \boldsymbol{h}_{\bullet p}]$.

Figure 3.1 depicts the case $p = 2$ and $n = 3$. The prediction $\boldsymbol{H}\hat{\boldsymbol{\theta}}$ is constrained to lie in the lower plane since that is the plane containing the two basis vectors. Clearly, the prediction $\boldsymbol{H}\hat{\boldsymbol{\theta}}$ is made as close as possible to the observed $\boldsymbol{Z}$ by picking $\hat{\boldsymbol{\theta}}$ such that $\boldsymbol{H}\hat{\boldsymbol{\theta}}$ represents the projection of $\boldsymbol{Z}$ into the plane containing the basis vectors. Geometrically, the error $\boldsymbol{Z} - \boldsymbol{H}\hat{\boldsymbol{\theta}}$ is orthogonal to the prediction $\boldsymbol{H}\hat{\boldsymbol{\theta}}$.

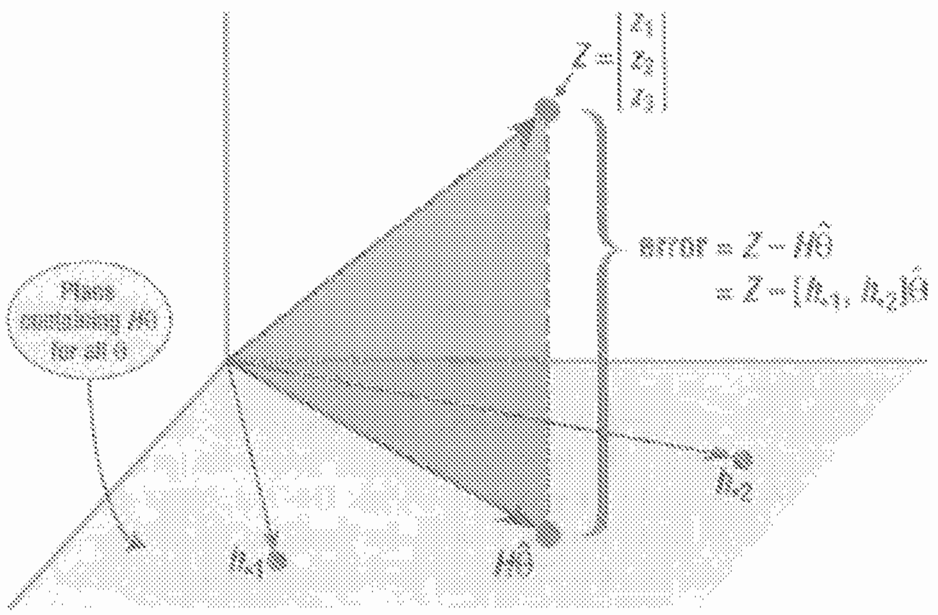
$Z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$
error $= Z - H\hat{\theta}$
$= Z - [h_1, h_2]\hat{\theta}$
Plane containing $H\theta$ for all θ
h_1
$H\hat{\theta}$
h_2

Figure 3.1. Geometric interpretation of least-squares estimate when $p = 2$ and $n = 3$. All possible predictions $H\hat{\theta}$ lie in indicated horizontal plane.

3.2 LEAST-MEAN-SQUARES AND RECURSIVE-LEAST-SQUARES FOR STATIC θ

3.2.1 Introduction

Suppose that $k = 1, 2, \ldots, n$ is a time index and suppose that the input–output data pairs $\{\boldsymbol{h}_k, z_k\}$ arrive sequentially: $\{\boldsymbol{h}_1, z_1\}$, $\{\boldsymbol{h}_2, z_2\}, \ldots, \{\boldsymbol{h}_n, z_n\}$. (Viewing k as a time index is for ease of discussion; it could as easily be viewed as another type of index—spatial, cross-sectional, and so on.) The focus in this and the next section is the development of *recursive* estimates $\hat{\theta}_k$ that are intended to approximate corresponding batch estimates $\hat{\theta}^{(k)}$. If $\hat{\theta}_k$ represents an estimate for θ based on k data pairs, we seek a way of computing $\hat{\theta}_{k+1}$ with a minimum of effort when the $(k+1)$st data pair arrives. Such recursive solutions are based on combining the currently available estimate with the $(k+1)$st data pair $\{\boldsymbol{h}_{k+1}, z_{k+1}\}$ in an efficient way to obtain the new estimate.

Aside from potential computational advantages, the recursive form clearly exhibits the value of a new data point. In fact, with advances in computing power, the computational advantages alone are becoming less important in many applications. However, it is likely that there will continue to be some real-time applications with massive data quantities where performing the required matrix inversion for a new batch solution at every time point may be difficult.

In the remaining five subsections, we present the least-mean-squares (LMS) and recursive-least-squares (RLS) solutions to this recursive estimation problem. We will see that these algorithms have, respectively, an interpretation as stochastic versions of the deterministic steepest descent and Newton–Raphson algorithms discussed in Section 1.4. The focus here is on cases where the optimal $\boldsymbol{\theta}$ is fixed over time; Section 3.3 considers the time-varying $\boldsymbol{\theta}$ setting.

3.2.2 Basic LMS Algorithm

A popular algorithm for parameter estimation in linear models is the LMS algorithm. LMS is widely used in signal processing and control applications, with one of the classic references being Widrow and Stearns (1985). The LMS algorithm has an interpretation as a stochastic version of the steepest descent algorithm of Section 1.4. LMS for linear models represents a special case of the nonlinear stochastic gradient algorithms to be considered in Chapter 5. (Additional detail on LMS is given in Sections 3.3 and 5.1.)

Before presenting the LMS algorithm, let us introduce the notion of the *instantaneous gradient*. When the least-squares criterion (3.3) is summed through the $(k+1)$st time point, the gradient of the latest summand at any $\boldsymbol{\theta}$ is

$$\frac{1}{2}\frac{\partial(z_{k+1}-\boldsymbol{h}_{k+1}^T\boldsymbol{\theta})^2}{\partial\boldsymbol{\theta}} = \boldsymbol{h}_{k+1}(\boldsymbol{h}_{k+1}^T\boldsymbol{\theta}-z_{k+1}). \tag{3.6}$$

The expression above is sometimes called the instantaneous gradient because it corresponds to the gradient of only the latest part of the least-squares loss function (3.3). Note that the expression in (3.6) is an unbiased, noisy measurement of the gradient of the $(k+1)$st summand in (3.2) (i.e., a noisy measurement of $\frac{1}{2}\partial E[(z_{k+1}-\boldsymbol{h}_{k+1}^T\boldsymbol{\theta})^2]/\partial\boldsymbol{\theta}$). As discussed in Section 3.3, the instantaneous gradient is especially important in time-varying systems where the aim is to estimate a dynamically changing value of $\boldsymbol{\theta}$. In such systems, it is important to put emphasis on the most recent information, corresponding to the last term in the sum of (3.2).

LMS is based on using the (noisy) instantaneous gradient expression in (3.6) as the gradient input in a steepest descent-type search. The qualifier "steepest descent-*type*" is used here because the gradient input is not deterministic, as pure steepest descent requires. In most practical applications, the general gain coefficient a_k in the steepest descent algorithm of Section 1.4 is set to a constant $a > 0$ that regulates the speed and stability of the algorithm. (A large a often helps the algorithm converge more quickly to the vicinity of $\boldsymbol{\theta}^*$, but a small a helps the algorithm avoid instability and divergence; what is meant by large and small in this context is highly problem dependent.) The iterative update for LMS is given below.

LMS Algorithm

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a\boldsymbol{h}_{k+1}(\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k - z_{k+1}). \tag{3.7}$$

Interestingly, the convergence theory for such a simple algorithm is not so simple! Informal arguments that $\hat{\boldsymbol{\theta}}_k$ is close to $\boldsymbol{\theta}^*$ in the sense that $E\left(\left\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*\right\|^2\right)$ is small are given in, for example, Widrow and Stearns (1985, pp. 101–103), Haykin (1996, pp. 392–399), and Moon and Stirling (2000, pp. 646–647). These arguments do not constitute a formal proof of convergence for LMS, but they are often used in guiding the implementation for many practical applications. Let us now outline the conditions given by Widrow and Stearns (1985, pp. 101–103).

Suppose that the $\boldsymbol{h}_k$ are random and mutually independent across k, having a common mean and second moment matrix $\boldsymbol{S}_h \equiv E(\boldsymbol{h}_k\boldsymbol{h}_k^T)$ (i.e., $E(\boldsymbol{h}_j\boldsymbol{h}_j^T) = E(\boldsymbol{h}_k\boldsymbol{h}_k^T)$ for all j, k). Further, suppose that $\boldsymbol{h}_{k+1}$ is independent of $\hat{\boldsymbol{\theta}}_k$ and

$$0 < a < \frac{2}{\lambda_{\max}(\boldsymbol{S}_h)},$$

where $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue of the argument matrix. Then, it is claimed that $E\left(\left\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*\right\|^2\right) \approx 0$ for large k. However, one should be careful in interpreting this result. Aside from some "casual" portions in the derivation, another potentially misleading aspect is that it ignores the magnitude of the noise (v_k). Noise typically has a dramatic affect on practical convergence.

A more rigorous analysis for fixed a (as in (3.7)) is given in Macchi and Eweda (1983) and Gerencsér (1995), where it is shown that $E\left(\left\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*\right\|^2\right) = O(a)$ for small a and $k \to \infty$. So, for small a and large k, the magnitude of the error in $\hat{\boldsymbol{\theta}}_k$ is proportional (in a stochastic sense) to $\sqrt{a}$, where the constant of proportionality depends on the noise level. Wang et al. (2000) consider a different setting where there is a fixed amount of data and LMS makes multiple passes through the fixed data. Although a type of convergence is possible in this batch setting for a constant a, the convergence is not to $\boldsymbol{\theta}^*$ but to a value that depends on the given data (similar ideas are considered in Sections 5.1, 5.2, and 15.4).

Conditions for formal convergence of recursive LMS to $\boldsymbol{\theta}^*$ are given, for example, in Ljung et al. (1992, Part III) and Guo and Ljung (1995). We also present a convergence result for LMS in Chapter 5 (Proposition 5.1). These analyses use stochastic approximation theory (Chapter 4) and require that the gain a be indexed by k and approach zero as $k \to \infty$. Under appropriate conditions, $\hat{\boldsymbol{\theta}}_k$ converges to $\boldsymbol{\theta}^*$ in the mean-squared or almost sure sense (as defined in Appendix C). Further, in a general setting that includes the linear model here as a special case, it is shown in Gerencsér (1993) that the difference between the LMS solution with decaying gain and the batch solution is proportional in a stochastic sense to $\log k/k$ for large k. That is, $\left\|\hat{\boldsymbol{\theta}}_k - \hat{\boldsymbol{\theta}}^{(k)}\right\|$ = $O(\log k/k)$, where the big-O term has an appropriate stochastic interpretation.

3.2.3 LMS Algorithm in Adaptive Signal Processing and Control

This subsection discusses the use of LMS in adaptive signal processing and control. We begin by introducing a slight generalization of the AR model of Section 3.1. We then discuss several ways that this model is used in adaptive control.

Suppose that a process is modeled as evolving according to

$$x_{k+1} = \beta_0 x_k + \beta_1 x_{k-1} + \ldots + \beta_m x_{k-m} + \gamma u_k + w_k, \tag{3.8}$$

where x_k represents the (scalar) state of the system at time point k, β_i is an unknown parameter describing the effect of the state at an earlier time $k - i$ (i.e., x_{k-i}) on the next state x_{k+1}, u_k is a deterministic or stochastic input, γ is an unknown scale factor that applies to the input, and w_k is the state noise accounting for random fluctuations in the process not captured via the other part of the linear model. The AR model of Example 3.1 is a special case of this model (corresponding to $\gamma = 0$). This form fits into the framework of the linear model in (3.1) by letting $\boldsymbol{\theta}$ = $[\beta_0, \beta_1, \ldots, \beta_m, \gamma]^T$ and $\boldsymbol{h}_k$ = $[x_{k-1}, x_{k-2}, \ldots, x_{k-m-1}, u_{k-1}]^T$ Hence, the LMS algorithm in (3.7) can be used to estimate $\boldsymbol{\theta}$ based on the noise-free measurements z_k = x_k. The model above is widely used in areas such as adaptive signal processing (e.g., noise cancellation) and adaptive control.

Note that this model violates one of the conditions discussed above from Widrow and Stearns (1985, pp. 101–103) and others. (Recall that if these conditions apply, there is informal evidence that $E\left(\left\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*\right\|^2\right) \approx 0$ for large k.) Namely, the $\boldsymbol{h}_k$ are *not* independent because they share common elements across k. Nevertheless, despite violating the independence condition, LMS is widely used in such models because the above conditions are merely *sufficient* conditions (and nonrigorous sufficient conditions at that!).

Two popular areas for such problems are noise cancellation and adaptive control. In noise cancellation, there exists an external sequence representing the sum of noise plus a useful signal. The aim is to try to recover the useful signal in this external sequence. Suppose that u_k represents a reference input signal that has frequency characteristics similar to the noise. LMS attempts to adapt $\boldsymbol{\theta}$ so that a predicted state value x_k can be differenced from the corresponding external value to recover the useful signal (e.g., Haykin, 1996, pp. 377–385).

For the remainder of this subsection, let us focus on the application of the above model in the area of adaptive control. There are many ways that adaptive controllers may be constructed. Let us outline one relatively simple (but useful!) *indirect* adaptive control method for solving a tracking problem. The method is indirect in that it relies on estimating the parameters of model (3.8) and then using the estimates to form the controller. (In contrast, a *direct* method bypasses the model estimation step while estimating unknown parameters in the control function u_k; see, e.g., Landau et al., 1998, pp. 12–20, for a more detailed discussion of direct versus indirect adaptive control.)

Consider a tracking problem, where the aim is to have the state x_k track a target ("desired") value d_k. This target value may be a deterministic sequence or a random sequence that is independent of the randomness in the system being controlled. Suppose further that there are noise-free measurements (i.e., $z_k = x_k$) and that the state noise w_k is independent with mean zero and variance q^2. (If only noisy measurements are available, the Kalman filter [see Section 3.3] can be used to estimate x_k.) Assuming for the moment that the model parameters $\boldsymbol{\theta}$ are known, then the control function u_k that forces x_{k+1} to be as close to d_{k+1} as possible is

$$u_k = \frac{d_{k+1} - \beta_0 x_k - \beta_1 x_{k-1} - \ldots - \beta_m x_{k-m}}{\gamma}, \quad k \geq m. \tag{3.9}$$

That is, substituting the above u_k into (3.8) produces a tracking error $x_{k+1} - d_{k+1} = w_k$ that has the smallest possible mean-squared magnitude given the presence of the state noise (i.e., $E[(x_{k+1} - d_{k+1})^2] = q^2$ for all $k \geq m$).

Of course, $\boldsymbol{\theta}$ in (3.9) is not known and must be estimated. There are both open- and closed-loop methods for estimating $\boldsymbol{\theta}$ using LMS. In the *open-loop* approach, $\boldsymbol{\theta}$ is estimated via some experimentation with predetermined inputs u_k *before* putting the controller "online" with u_k as given in (3.9). In such open-loop training, it is desirable to pick the inputs with the general goal of enhancing the estimation of $\boldsymbol{\theta}$. This ties directly to Chapter 17 on optimal input (experimental) design and to the related idea of *persistency of excitation* (e.g., Ljung, 1999, Chap. 13). A different suboptimal method that is sometimes used is to randomly generate a set of inputs via a pseudorandom generator as in Appendix D. After the open-loop training is completed, the terminal estimate, say $\hat{\boldsymbol{\theta}}_n$, is substituted for $\boldsymbol{\theta}$ in (3.9). The system can now be run in *closed-loop* mode based on the controller (3.9) (closed-loop because now the controller u_k depends on feedback via the previous state values as opposed to the predetermined values used in the open-loop phase).

Alternatively, the training can be done in closed-loop mode based on using (3.9) as input to the system, with the most recent $\boldsymbol{\theta}$ estimate used in (3.9) at each time point. That is, the system is started with the controller (3.9) relying on

$\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_0$. The state x_1 is observed, and then LMS is used to update $\boldsymbol{\theta}$ to $\hat{\boldsymbol{\theta}}_1$ based on the difference between x_1 and d_1. This procedure is repeated as long as desired. This has the advantage of not requiring a separate "off-line" open-loop set of experiments, but has the disadvantage that the inputs u_k are not chosen to enhance the estimation of $\boldsymbol{\theta}$. Among others, Ljung (1999, Sects. 13.4 and 13.5) discusses some of the issues associated with such closed-loop model estimation.

Sometimes open-loop training, as mentioned above, is combined with closed-loop training. The open-loop phase is used to get $\boldsymbol{\theta}$ estimates that are reasonable, with training in closed-loop being used to refine the values or adapt to changes in the optimal values.

The LMS recursion in (3.7) can be used to estimate $\boldsymbol{\theta}$ regardless of the means for determining the inputs u_k for the open-loop model estimation. The quality of estimate depends on the number of measurements used in the training and the way the inputs are chosen for the given number of measurements. In open- or closed-loop estimation, the justification for the substitution of an estimate for $\boldsymbol{\theta}$ in formula (3.9) rests on the *certainty equivalence principle* of adaptive control (e.g., Landau et al., 1998, p. 504). The example below considers an implementation of LMS in closed-loop estimation.

Example 3.4—LMS in closed-loop estimation. Suppose that the state evolution is modeled according to $x_{k+1} = \beta x_k + \gamma u_k + w_k$ with noise-free state measurements $z_k = x_k$. Then $\boldsymbol{\theta} = [\beta, \gamma]^T$ and $\boldsymbol{h}_k = [x_{k-1}, u_{k-1}]^T$. Given an initial state x_0, parameter vector $\hat{\boldsymbol{\theta}}_0$, and target value d_1, the initial control u_0 is constructed based on (3.9). The system is operated with this control, producing a new state value x_1. LMS is then used to update $\boldsymbol{\theta}$ according to

$$\hat{\boldsymbol{\theta}}_1 = \hat{\boldsymbol{\theta}}_0 - a\begin{bmatrix} x_0 \\ u_0 \end{bmatrix}\left([x_0 \;\; u_0]\hat{\boldsymbol{\theta}}_0 - x_1\right)$$

(note that $[x_0, u_0]\hat{\boldsymbol{\theta}}_0 = d_1$). This updating process continues for as long as desired. The general process is illustrated in the feedback diagram of Figure 3.2. (Exercise 3.4 considers a numerical implementation.) ❑

3.2.4 Basic RLS Algorithm

Another standard method for processing data in a sequential manner is RLS. While the LMS method is connected to stochastic steepest descent-type methods, RLS is a type of stochastic Newton–Raphson method, as shown below. Further, the RLS solution approaches the batch solution for large n. Large in this context may actually be quite small in practice, say $n = 5$ (see Exercise 3.8). That is, for the same set of n input–output pairs $\{\boldsymbol{h}_k, z_k\}$, $k = 1, 2, \ldots, n$, the RLS estimate of $\boldsymbol{\theta}$ is nearly the same as the batch estimate of $\boldsymbol{\theta}$ found by (3.5).

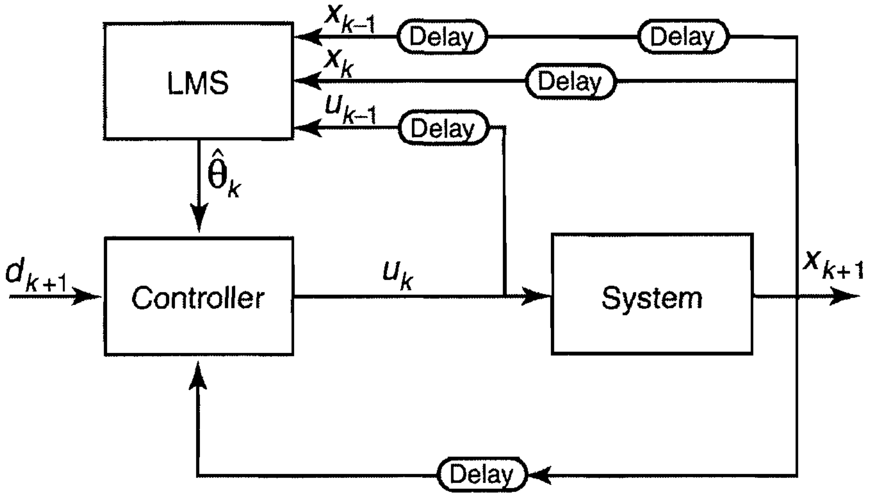

Figure 3.2. Estimation via LMS for first-order process with noise-free state measurements. Parameter values are updated while system operates in closed-loop mode.

Derivation of the RLS solution follows using the batch solution as a starting point. It is assumed that all indicated matrix inverses exist. In deriving the RLS algorithm, first note by (3.5) that the batch estimator satisfies

$$\hat{\theta}^{(k+1)} = \left(\begin{bmatrix} H_k \\ h_{k+1}^T \end{bmatrix}^T \begin{bmatrix} H_k \\ h_{k+1}^T \end{bmatrix} \right)^{-1} \begin{bmatrix} H_k \\ h_{k+1}^T \end{bmatrix}^T \begin{bmatrix} Z_k \\ z_{k+1} \end{bmatrix} \tag{3.10}$$

(recall that H_k is a $k \times p$ matrix). Because we have assumed that the necessary inverse exists (see (3.5)), let us define

$$P^{(k)} \equiv (H_k^T H_k)^{-1}.$$

From (3.10),

$$\hat{\theta}^{(k+1)} = P^{(k+1)}(H_k^T Z_k + h_{k+1} z_{k+1}). \tag{3.11}$$

Note that (3.5) implies that $[P^{(k)}]^{-1}\hat{\theta}^{(k)} = H_k^T Z_k$ for $k \geq p$.

We are now in a position to convert from the above batch forms for the θ estimate ((3.10) and (3.11)) and $P^{(k)}$ value to corresponding recursive forms. Consequently, we switch the notation from $\hat{\theta}^{(k)}$ to $\hat{\theta}_k$ and from $P^{(k)}$ to P_k. Although $\hat{\theta}_k$ and P_k are derived directly from the corresponding batch quantities, $\hat{\theta}^{(k)} \neq \hat{\theta}_k$ and $P^{(k)} \neq P_k$ in general. The discrepancies result from the need for user-specified initial conditions $\hat{\theta}_0$ and P_0 to initiate the recursive estimates. The effects of the initial conditions decay over time (see Exercise 3.8).

From the implied form for $\boldsymbol{P}^{(k+1)}$ in (3.10) (i.e., the $(\cdot)^{-1}$ part; see Exercise 3.5), the corresponding recursive form is

$$\boldsymbol{P}_{k+1}^{-1} = \boldsymbol{P}_k^{-1} + \boldsymbol{h}_{k+1}\boldsymbol{h}_{k+1}^T. \tag{3.12}$$

Likewise, (3.11) can be rewritten in recursive form as

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{k+1} &= \boldsymbol{P}_{k+1}(\boldsymbol{P}_k^{-1}\hat{\boldsymbol{\theta}}_k + \boldsymbol{h}_{k+1}z_{k+1}) \\ &= \boldsymbol{P}_{k+1}[(\boldsymbol{P}_{k+1}^{-1} - \boldsymbol{h}_{k+1}\boldsymbol{h}_{k+1}^T)\hat{\boldsymbol{\theta}}_k + \boldsymbol{h}_{k+1}z_{k+1}] \\ &= \hat{\boldsymbol{\theta}}_k - \boldsymbol{P}_{k+1}\boldsymbol{h}_{k+1}(\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k - z_{k+1}),\end{aligned} \tag{3.13}$$

where (3.12) is used in the second line of (3.13).

Note that (3.13) nicely exposes the relationship between the new and old estimates for $\boldsymbol{\theta}$. The new estimate is equal to the old estimate plus an adjustment due to the new data $\{\boldsymbol{h}_{k+1}, z_{k+1}\}$. The correction term is proportional to the prediction error $\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k - z_{k+1}$, where from (3.1) it is apparent that $\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k$ represents a prediction of z_{k+1}. In the idealized case of a perfect prediction, there would be no change in the subsequent $\boldsymbol{\theta}$ estimate.

Although the above recursive formulas for $\hat{\boldsymbol{\theta}}_k$ and $\boldsymbol{P}_k$ are sufficient to yield an RLS solution, the form in (3.12) for $\boldsymbol{P}_k$ is computationally undesirable since it involves the inverse of a $p\times p$ matrix when used in (3.13). A form based on the *matrix inversion lemma* (Appendix A, matrix relationship (xxii)) eliminates this requirement. Applying this lemma to (3.12) yields the following final recursions for the RLS algorithm based on model (3.1) (Exercise 3.6):

RLS Algorithm

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \frac{\boldsymbol{P}_k\boldsymbol{h}_{k+1}\boldsymbol{h}_{k+1}^T\boldsymbol{P}_k}{1 + \boldsymbol{h}_{k+1}^T\boldsymbol{P}_k\boldsymbol{h}_{k+1}}, \tag{3.14a}$$

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - \boldsymbol{P}_{k+1}\boldsymbol{h}_{k+1}(\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k - z_{k+1}). \tag{3.14b}$$

Initial conditions $\hat{\boldsymbol{\theta}}_0$ and $\boldsymbol{P}_0$ need to be specified to start the RLS recursion in (3.14a, b). A common means for initialization is to set $\hat{\boldsymbol{\theta}}_0 = \boldsymbol{0}$ or some other vector reflecting prior knowledge about $\boldsymbol{\theta}$. For $\boldsymbol{P}_0$, a common choice is $c\boldsymbol{I}_p$ for some $c > 0$. In particular, in the absence of prior knowledge suggesting another choice, $c >> 0$ is recommended in order to have the RLS solution and the batch solution be close to each other for small k (Young, 1984, p. 27; Jang et al., 1997, p. 115). Choosing some other form for $\boldsymbol{P}_0$ is associated with cases where one has knowledge about $\boldsymbol{\theta}$ that might suggest a value different from the "priorfree" batch solution. One alternative might arise if there is available a batch solution based on some data available *prior* to $\{\boldsymbol{h}_1, z_1\}$; $\hat{\boldsymbol{\theta}}_0$ and $\boldsymbol{P}_0$ may then be chosen based on this prior batch estimate. Another setting where an alternative prior might be useful is when the underlying true $\boldsymbol{\theta}$ varies in time; this is discussed in Section 3.3.

Expression (3.14b) has an intuitively appealing form. The recursion indicates that the new estimate $\hat{\boldsymbol{\theta}}_{k+1}$ is a linear combination of the old estimate $\hat{\boldsymbol{\theta}}_k$ and a correction term that is a weighting of the error in predicting the new measurement from the old $\boldsymbol{\theta}$ estimate (the error is $\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k - z_{k+1}$). This prediction error is sometimes called a *residual*.

In the case study of Section 3.4, which is based on real data for a complex process, we will see that the RLS and batch solutions also effectively reach the same points over the range of data available. In contrast, in some applications, there will be a slow drift in the data that produces a discrepancy between the true data-generating mechanism and the assumed model for batch estimation. In particular, the batch assumption of the noise being mean zero with a common variance may be violated. In cases with such a drift, it is likely that the RLS solution will tend to be better (for predictive purposes) since it puts more weight on recent measurements. Section 3.3 considers this setting in more detail.

It was mentioned in Subsection 3.1.2 that there is a close connection between the least-squares and minimum-mean-squared error approaches. Although RLS is usually presented (as above) as a recursive implementation of a least-squares algorithm (basically a deterministic algorithm), it also has an interpretation as *stochastic* optimization algorithms in the sense of using noisy input information to optimize the *mean-squared* error. The loss function can be changed to a mean-squared error criterion by taking the expectation of the right-hand side of (3.3). In some sense, this is what it *should* be since then we are trying to find the $\boldsymbol{\theta}$ that minimizes the cost over all possible outcomes (not just those that have actually been seen in the n observations). The input $\boldsymbol{h}_{k+1}(\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k - z_{k+1})$ on the right-hand side of (3.14b) represents a noisy measurement of the instantaneous gradient $\frac{1}{2}\partial E[(z_{k+1} - \boldsymbol{h}_{k+1}^T\boldsymbol{\theta})^2]/\partial\boldsymbol{\theta}$ (see (3.6)) weighted by the gain matrix $\boldsymbol{P}_{k+1}$.

Using stochastic approximation theory (see Chapter 4), conditions can be established such that the RLS algorithm converges to the minimum ($\boldsymbol{\theta}^*$) of the mean-squared error loss function (3.2) as the sample size $n \to \infty$. This may or may not change the actual mechanics of the algorithm, depending on whether the input–output pairs $\{\boldsymbol{h}_k, z_k\}$ satisfy the stochastic approximation conditions, but it does change the interpretation of the algorithm and provides for the invocation of asymptotic convergence and distribution theory available in stochastic approximation. Note also that this theory allows the design vectors (sometimes called *regressors*) $\boldsymbol{h}_k$ to be random. Macchi and Eweda (1983) and Ljung et al.

(1992, Part III) are two references that discuss the connection of RLS (and LMS) algorithms to stochastic approximation, establishing formal convergence in the process.

3.2.5 Connection of RLS to the Newton–Raphson Method

The RLS algorithm above has a close connection to the Newton–Raphson algorithm discussed in Section 1.4. As with LMS, this connection is via the instantaneous gradient (3.6). For convenience in the recursions of this subsection, let us index the least-squares criterion by k, writing $\hat{L}(\boldsymbol{\theta},k)$ for $\hat{L}(\boldsymbol{\theta}) = (2k)^{-1}\sum_{j=1}^{k}(z_j - \boldsymbol{h}_j^T\boldsymbol{\theta})^2$.

Suppose that the RLS estimate $\hat{\boldsymbol{\theta}}_k$ approximately minimizes $\hat{L}(\boldsymbol{\theta},k)$. (Because of the usually small discrepancy between the RLS solution and the batch solution, it cannot generally be assumed that $\hat{\boldsymbol{\theta}}_k$ exactly minimizes $\hat{L}(\boldsymbol{\theta},k)$.) Then

$$\left.\frac{\partial \hat{L}(\boldsymbol{\theta},k)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k} \approx \mathbf{0}.$$

Because $\hat{L}(\boldsymbol{\theta},k+1) = k\hat{L}(\boldsymbol{\theta},k)/(k+1) + (z_{k+1} - \boldsymbol{h}_{k+1}^T\boldsymbol{\theta})^2/[2(k+1)]$, this approximation implies from (3.6),

$$\begin{aligned}\left.\frac{\partial \hat{L}(\boldsymbol{\theta},k+1)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k} &= \frac{k}{k+1}\left.\frac{\partial \hat{L}(\boldsymbol{\theta},k)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_k} + \frac{\boldsymbol{h}_{k+1}(\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k - z_{k+1})}{k+1} \\ &\approx \frac{\boldsymbol{h}_{k+1}(\boldsymbol{h}_{k+1}^T\hat{\boldsymbol{\theta}}_k - z_{k+1})}{k+1}.\end{aligned}$$

Here, the gradient of the *instantaneous* part serves as a proxy for the gradient of the *overall* loss function in (3.3) (which is a sum over all measurements). Further, by (3.12), the Hessian of $\hat{L}(\boldsymbol{\theta},k+1)$ is given by

$$\begin{aligned}\text{Hessian}_{k+1} &= \frac{\partial^2 \hat{L}(\boldsymbol{\theta},k+1)}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^T} = \frac{1}{k+1}\sum_{j=1}^{k+1}\boldsymbol{h}_j\boldsymbol{h}_j^T \\ &= \frac{1}{k+1}(\boldsymbol{P}_{k+1}^{-1} - \boldsymbol{P}_0^{-1}),\end{aligned}$$

where the second line above follows by solving the difference equation in (3.12) in terms of $\boldsymbol{P}_0^{-1}$ and the sum of $\boldsymbol{h}_j\boldsymbol{h}_j^T$ over $j = 1$ to $k+1$. Hence, if we assume that P_0^{-1} is small relative to P_{k+1}^{-1} (usually reasonable given (3.12)), then, from the RLS recursion in (3.13),

$$\begin{aligned}\hat{\theta}_{k+1} &= \hat{\theta}_k - P_{k+1}h_{k+1}(h_{k+1}^T\hat{\theta}_k - z_{k+1})\\ &\approx \hat{\theta}_k - [\text{Hessian}_{k+1}]^{-1}\left.\frac{\partial \hat{L}(\theta, k+1)}{\partial \theta}\right|_{\theta=\hat{\theta}_k}.\end{aligned}$$

Hence, one step of RLS is closely related to one step of the Newton–Raphson algorithm. The discrepancy (hence the ≈) is due to the approximations involving the gradient of $\hat{L}(\theta, k)$ being nearly $\mathbf{0}$ and P_0^{-1} being relatively small. This close connection between RLS and Newton–Raphson helps explain the widespread observation of fast convergence in practice for RLS.

3.2.6 Extensions to Multivariate RLS and Weighted Summands in Least-Squares Criterion

The RLS form in (3.14a, b) readily extends to multivariate z_k (say, m-dimensional) with a least-squares criterion that is possibly weighted differently across k. In particular, suppose that the multivariate model for output z_k is

$$z_k = \bar{H}_k\theta + v_k, \tag{3.15}$$

where $\bar{H}_k$ is an $m \times p$ design matrix and v_k is a mean-zero noise vector with (possibly varying with k) full-rank covariance matrix Σ_k. Consider the weighted least-squares criterion

$$\hat{L}(\theta) = \frac{1}{2n}\sum_{k=1}^{n}(z_k - \bar{H}_k\theta)^T\Sigma_k^{-1}(z_k - \bar{H}_k\theta), \tag{3.16}$$

where the weighting Σ_k^{-1} makes the expected value of each summand in (3.16) identical (= 1) when the model (3.15) is an accurate description of the output z_k. Criterion (3.16) retains the $\hat{L}(\theta)$ notation of (3.3) (versus $L(\theta)$) to emphasize that this least-squares criterion is a noisy representation of the corresponding mean-squared criterion.

Multivariate analogues of the arguments yielding (3.14a, b) can be used to derive the RLS algorithm. In converting the batch solution to a recursive solution, the matrix inversion lemma (Appendix A, matrix relationship (xxii)) is used (e.g., Sorenson, 1980, pp. 56–57; Ljung, 1999, pp. 366–367). The resulting RLS algorithm is:

RLS Algorithm—Multivariate Output with Varying Noise Covariance Matrix

$$P_{k+1} = P_k - P_k \bar{H}_{k+1}^T (\Sigma_{k+1} + \bar{H}_{k+1} P_k \bar{H}_{k+1}^T)^{-1} \bar{H}_{k+1} P_k, \tag{3.17a}$$

$$\hat{\theta}_{k+1} = \hat{\theta}_k - P_{k+1} \bar{H}_{k+1}^T \Sigma_{k+1}^{-1} (\bar{H}_{k+1} \hat{\theta}_k - z_{k+1}). \tag{3.17b}$$

A comparison of the above with the scalar version in (3.14a, b) shows a strong resemblance. The primary differences are the substitution of the matrix $\bar{H}_k$ for the vector h_k^T and the inclusion of the covariance matrix Σ_k to account for the weighting in the criterion (3.16). If one has scalar measurements z_k, but time-varying noise variances, say σ_k^2, then, analogous to (3.16), the criterion in (3.3) may be changed to

$$\hat{L}(\theta) = \frac{1}{2n} \sum_{k=1}^{n} \sigma_k^{-2} (z_k - h_k^T \theta)^2$$

$$= \frac{1}{2n} (Z_n - H_n \theta)^T \begin{bmatrix} \sigma_1^{-2} & 0 & 0 & 0 \\ 0 & \sigma_2^{-2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_n^{-2} \end{bmatrix} (Z_n - H_n \theta).$$

Consequently, the recursions in (3.17a, b) apply directly with σ_k^2 replacing Σ_k and h_k^T replacing $\bar{H}_k$. As before, initial conditions $\hat{\theta}_0$ and P_0 need to be specified to start the RLS recursion in (3.17a, b). Also, as before, the recursion for θ has an intuitively appealing form by being a linear combination of the old estimate and the weighted residual error.

3.3 LMS, RLS, AND KALMAN FILTER FOR TIME-VARYING θ

3.3.1 Introduction

In many applications of linear models, the underlying θ evolves in time. Typical applications of time-varying θ include target tracking, adaptive control, time-series forecasting, sequential design and response surface methods (Chapter 17), and signal processing. This section discusses some popular approaches to this estimation problem.

The model considered here is based on scalar measurements $z_1, z_2, \ldots$ obtained according to the following form:

$$z_k = h_k^T \theta_k + v_k, \quad k = 1, 2, \ldots, n, \tag{3.18}$$

where $\boldsymbol{h}_k$ is the $p \times 1$ design vector, v_k is the random noise term, and $\boldsymbol{\theta}_k$ is the *time-varying* true (unknown) value of the parameter vector of interest. In general, the variation in $\boldsymbol{\theta}_k$ will be stochastic in nature, but this is not necessarily the case. Similar to Section 3.2, our interest is in taking the input–output pairs $\{\boldsymbol{h}_k, z_k\}$ and estimating $\boldsymbol{\theta}_k$.[1] Of course, the problem differs fundamentally from that in Sections 3.1 and 3.2 in that $\boldsymbol{\theta}_k$ may be constantly changing. Hence, in contrast to the discussion of Sections 3.1 and 3.2, the standard batch regression solution (eqn. (3.5)) for a fixed $\boldsymbol{\theta}$ is an inappropriate starting place for developing recursive forms.

Given the time variation in $\boldsymbol{\theta}_k$, the optimization problem is also time varying in that the aim is to minimize some time-varying loss L_k or associated least-squares estimate $\hat{L}_k$ that expresses, in some sense, the desire to pick an estimate $\hat{\boldsymbol{\theta}}_k$ that is close to the true value of $\boldsymbol{\theta}_k$. We discuss below three common approaches to such estimation: LMS, RLS, and the Kalman filter. Many books and articles consider one or more of these approaches in the context of time-varying estimation. Among such references are Anderson and Moore (1979), Sorenson (1980), Spall (1988a), Haykin (1996), Ljung (1999), and Moon and Stirling (2000).

The prototype recursive form for time-varying parameter estimation is

$$\hat{\boldsymbol{\theta}}_{k+1} = A_k\hat{\boldsymbol{\theta}}_k - \boldsymbol{\kappa}_{k+1}(\boldsymbol{h}_{k+1}^T A_k\hat{\boldsymbol{\theta}}_k - z_{k+1}), \qquad (3.19)$$

where A_k is a potentially time-varying $p\times p$ matrix and $\boldsymbol{\kappa}_k$ ("kappa") is some appropriately chosen $p\times 1$ gain vector. The specific forms for A_k and $\boldsymbol{\kappa}_k$ depend on the choice of LMS, RLS, or the Kalman Filter. Each of the three subsections below discusses the essential assumptions of one of these methods, the algorithmic form vis-à-vis (3.19), and the core theoretical results in support of the method.

Unlike the fixed parameter case emphasized in Sections 3.1 and 3.2, it is usually impossible for an estimation algorithm to formally converge to a time-varying solution in the sense that $\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k$ goes to zero in a probabilistic manner, as discussed in Appendix C.[2] In essence, the algorithm is not able to acquire enough information about any one $\boldsymbol{\theta}_k$. Hence, the theory discussed below pertains to *other* aspects of the algorithm performance, such as bounds on the mean-squared error of the parameter estimate.

[1]Following the precedent of using $\boldsymbol{\theta}^*$ for an optimal value, we could also write $\boldsymbol{\theta}_k^*$ for the true parameter $\boldsymbol{\theta}_k$ of interest. However, we write $\boldsymbol{\theta}_k$ for ease of notation and because the time-varying solution is often a random quantity, unlike the *deterministic* quantity $\boldsymbol{\theta}^*$.

[2]The qualifier "usually" is included because there are rare cases where extensive prior information is available on the evolution of the parameter, with the algorithm able to encompass this information in a manner that allows formal convergence.

3.3.2 LMS

Model Assumptions

There are minimal statistical assumptions made about the evolution of the process $\boldsymbol{\theta}_k$ (which may be random or deterministic). For example, a typical assumption is that the increments $\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$ are uncorrelated across k and have finite-magnitude covariance matrix. The aim with LMS is to minimize the instantaneous criterion $L_k(\boldsymbol{\theta}) = \frac{1}{2}E[(z_k - \boldsymbol{h}_k^T\boldsymbol{\theta})^2]$ as k evolves (minimizing L_k yields the ideal solution $\boldsymbol{\theta} = \boldsymbol{\theta}_k$). LMS allows for $\boldsymbol{h}_k$ to be random with the expectation in $L_k(\boldsymbol{\theta})$ being an average over the noise together with the randomness in $\boldsymbol{h}_k$.

Algorithm Terms for Use in (3.19)

In LMS, A_k in (3.19) is set to $\boldsymbol{I}_p$ for all k. The standard unnormalized LMS algorithm has a gain vector of

$$\boldsymbol{\kappa}_{k+1} = a\boldsymbol{h}_{k+1}, \tag{3.20a}$$

where $a > 0$ is a constant (nondecaying) scale factor useful in tracking problems. An often-used normalized variation on the basic form above is

$$\boldsymbol{\kappa}_{k+1} = \frac{a\boldsymbol{h}_{k+1}}{1 + a\|\boldsymbol{h}_{k+1}\|}, \tag{3.20b}$$

which sometimes helps to adapt to time variation in the system by scaling for different magnitudes in the design vector $\boldsymbol{h}_{k+1}$ (as usual, $\|\cdot\|$ denotes the standard Euclidean norm).

Discussion and Summary of Theory

Because LMS is an analogue of first-order steepest descent, it is generally easier to implement (in either the unnormalized or normalized form) than RLS described below (an analogue of Newton-Raphson). Further, because LMS only pertains to the instantaneous loss $L_k(\boldsymbol{\theta}) = \frac{1}{2}E[(z_k - \boldsymbol{h}_k^T\boldsymbol{\theta})^2]$, it tends to have less inertia than RLS. (As shown below, RLS works with a loss function representing a weighted sum over all time points up to the current time.) Some theory for this time-varying $\boldsymbol{\theta}_k$ case is presented in Guo and Ljung (1995) and Delyon and Juditsky (1995). These results pertain to the error $E\left(\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k\|^2\right)$ and to the approximate distribution of $\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k$ for large k and small a.

3.3.3 RLS

Model Assumptions

As with LMS, there are minimal statistical assumptions made about the evolution of the process $\boldsymbol{\theta}_k$. However, to reflect the greater importance of current measurements z_k relative to older measurements, the standard least-squares criterion in (3.3) is altered to assign greater weight to the new information. In particular, the RLS algorithm for time-varying $\boldsymbol{\theta}$ is tied to a criterion of the form

$$\hat{L}(\boldsymbol{\theta}, k) = \frac{1}{2k}\sum_{j=1}^{k} \gamma^{k-j}(z_j - \boldsymbol{h}_j^T\boldsymbol{\theta})^2 \qquad (3.21)$$

for $0 < \gamma \leq 1$. Note that (3.21) exponentially de-weights past measurements.

Algorithm Terms for Use in (3.19)

The contribution to (3.19) is similar to the constant-parameter solution in Subsection 3.2.4. In particular, A_k in (3.19) is set to I_p for all k and

$$\boldsymbol{\kappa}_{k+1} = \boldsymbol{P}_{k+1}\boldsymbol{h}_{k+1}, \qquad (3.22a)$$

$$\boldsymbol{P}_{k+1}^{-1} = \gamma \boldsymbol{P}_k^{-1} + \boldsymbol{h}_{k+1}\boldsymbol{h}_{k+1}^T, \qquad (3.22b)$$

where $\boldsymbol{P}_0 > 0$. The efficiency of the above form can be improved by avoiding the required matrix inverse to obtain the $\boldsymbol{P}_{k+1}$ appearing in $\boldsymbol{\kappa}_{k+1}$. A preferred (equivalent) form for implementation based on the matrix inversion lemma (Appendix A, matrix relationship (xxii)) is

$$\boldsymbol{P}_{k+1} = \frac{1}{\gamma}\left[\boldsymbol{P}_k - \frac{\boldsymbol{P}_k\boldsymbol{h}_{k+1}\boldsymbol{h}_{k+1}^T\boldsymbol{P}_k}{\gamma + \boldsymbol{h}_{k+1}^T\boldsymbol{P}_k\boldsymbol{h}_{k+1}}\right]. \qquad (3.22c)$$

Discussion and Summary of Theory

The weighting of past input information in the RLS loss function tends to create an "inertia" that will slow down the variations in the $\boldsymbol{\theta}_k$ estimate from time point to time point; as γ increases from 0 to 1, this weighting of earlier points increases. When $\gamma = 1$, the solution reduces to the "standard" (no damping) RLS solution in (3.14a, b) where all previous points are weighted equally (i.e., the greatest inertia). Note that no matrix inverse is required to compute $\boldsymbol{\kappa}_{k+1}$ when using the form in (3.22c). So, (3.22a) and (3.22c) together (versus (3.22a) and (3.22b)) constitute the generally preferred RLS algorithm to accommodate time-varying $\boldsymbol{\theta}_k$. While the standard RLS algorithm assumes deterministic $\boldsymbol{h}_k$, some theory exists for random $\boldsymbol{h}_k$ in this time-varying $\boldsymbol{\theta}_k$ setting. In particular, Ljung et al. (1992, pp. 95–113) and Guo and Ljung (1995) provide bounds on the mean-squared tracking error $E\left(\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k\|^2\right)$ when the inputs $\boldsymbol{h}_k$ are random and satisfy certain conditions, such as independence and bounded second moments across k.

3.3.4 Kalman filter

Model Assumptions

Unlike the LMS and RLS approaches above, the Kalman filter (Kalman, 1960) is based on a precise statistical representation for the evolution of $\boldsymbol{\theta}_k$. This representation, called a *state equation*, is given by the expression

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\Phi}_k\boldsymbol{\theta}_k + \boldsymbol{w}_k, \tag{3.23}$$

where $\boldsymbol{\Phi}_k$ is the transition matrix and $\boldsymbol{w}_k$ is a mean-zero noise vector having covariance matrix $\boldsymbol{Q}_k$. Hence, from (3.23), the evolution of $\boldsymbol{\theta}_k$ is via a first-order vector difference equation with random input. In the Kalman filter literature, (3.23) and (3.18) together constitute the *state-space model* for the process. In general, the state-space model and Kalman filter allows for multivariate measurements z_k (as in (3.15)), but for consistency with the discussion of RLS and LMS, we focus on the scalar measurement case.

Algorithm Terms for Use in (3.19)

The algorithm makes use of the state-space model terms, including σ_k^2 = var(v_k) and $\boldsymbol{Q}_k = \text{cov}(\boldsymbol{w}_k)$. In particular, $\boldsymbol{A}_k = \boldsymbol{\Phi}_k$ for all k and

$$\boldsymbol{\kappa}_{k+1} = \frac{(\boldsymbol{\Phi}_k\boldsymbol{P}_k\boldsymbol{\Phi}_k^T + \boldsymbol{Q}_k)\boldsymbol{h}_{k+1}}{\sigma_{k+1}^2 + \boldsymbol{h}_{k+1}^T(\boldsymbol{\Phi}_k\boldsymbol{P}_k\boldsymbol{\Phi}_k^T + \boldsymbol{Q}_k)\boldsymbol{h}_{k+1}}, \tag{3.24a}$$

$$\boldsymbol{P}_{k+1} = (\boldsymbol{I}_p - \boldsymbol{\kappa}_{k+1}\boldsymbol{h}_{k+1}^T)(\boldsymbol{\Phi}_k\boldsymbol{P}_k\boldsymbol{\Phi}_k^T + \boldsymbol{Q}_k), \tag{3.24b}$$

where $\sigma_k^2 > 0$, $\boldsymbol{P}_0 \geq \boldsymbol{0}$, and $\boldsymbol{Q}_k \geq \boldsymbol{0}$. (With multivariate measurements z_k, the gain $\boldsymbol{\kappa}_{k+1}$ becomes a matrix and the denominator in (3.24a) is replaced by an analogous matrix inverse; see any reference on Kalman filtering.)

Discussion and Summary of Theory

The Kalman filter form above is a special case of several more general forms. Aside from multivariate measurements, the more general forms encompass correlation between the process and measurement noises. The form above *does* allow for one significant extension, in that time-varying state-space parameters are allowed ($\boldsymbol{\Phi}_k$, $\boldsymbol{Q}_k$, σ_k^2). The Kalman filter also has important statistical properties. One is unbiasedness in the sense that $E(\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k) = \boldsymbol{0}$; another is that $\boldsymbol{P}_k$ in (3.24b) is equal to $E\left[(\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k)(\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k)^T\right]$ (hence $\boldsymbol{P}_k$ is called the error-covariance matrix). Note that $\boldsymbol{P}_k$ is an essential quantity in assessing the filter accuracy. Both the unbiasedness and error-covariance interpretation of $\boldsymbol{P}_k$ depend critically on the validity of the state-space model (including parameter values $\hat{\boldsymbol{\theta}}_0$, $\boldsymbol{P}_0$, $\boldsymbol{\Phi}_k$, $\boldsymbol{Q}_k$, σ_k^2).

As with RLS and LMS, the Kalman filter is a *linear* estimator in the sense that (3.19) yields a $\hat{\boldsymbol{\theta}}_k$ that is a linear combination of all previous measurements z_j, $j \leq k$. The Kalman filter minimizes the time-varying MSE $E\left(\left\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k\right\|^2\right)$ at each k among all possible linear estimators (i.e., among all estimators that are linear combinations of the measurements). It is *not* generally the minimum MSE estimator when one allows for estimators that are nonlinear functions of the measurements (see, e.g., Anderson and Moore, 1979, pp. 46–49; Spall, 1988a, pp. xx–xxi). However, under the *additional* assumption that the initial state, measurement noises, and process noises are independent normally distributed processes, the Kalman filter minimizes the time-varying MSE $E\left(\left\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}_k\right\|^2\right)$ among *all* estimators (linear or nonlinear).

Because RLS and LMS do not require the assumption of an underlying state equation for $\boldsymbol{\theta}_k$, it would not be expected that those methods would track as well as the Kalman filter *if* a state-space model were available. The LMS and RLS algorithms require a gain a or weight γ to be user specified, while the Kalman filter has all parameters other than $\hat{\boldsymbol{\theta}}_0$ and $\boldsymbol{P}_0$ uniquely specified by the assumptions of the state-space model. Hence, one cannot say that any one of these tracking algorithms is uniformly better than the others since the best choice depends on the assumptions governing the parameter variation (see, e.g., Ljung et al., 1992, pp. 96–101, and Guo and Ljung, 1995, for further comparative discussion).

3.4 CASE STUDY: ANALYSIS OF OBOE REED DATA

...an ill wind that nobody blows good.

—Comedian Danny Kaye, paraphrasing a proverb of English playwright John Heywood (approx. 1497–1580), in speaking of the oboe in "The Secret Life of Walter Mitty" (1947).

This section applies some of the recursive estimation algorithms above to the estimation of models associated with a musical instrument. The oboe is a medium-to-high-pitched wind instrument that has long played a central role in symphony orchestras. Its nasal tone and piercing quality make it one of the most distinctive of the orchestral sounds. Like the lower-pitched bassoon, the oboe is a double-reed instrument, so called because the musician blows into a reed separated into two parts that vibrate against one another to produce a sound. The quality of the reed is absolutely critical to the sound quality produced by the oboe. Figure 3.3 shows an oboe with an expanded display of the reed.

A fact unknown to many concertgoers is that almost all professional oboists make their own reeds. This can be a tedious and lengthy process of shaping a treated piece of cane (*Arundo donax*) into a reed that attaches to the oboe and meets stringent requirements for playing quality (e.g., easy articulation of notes and the ability to play in tune with other instruments). It is common for an oboist to spend several hours making a reed that lasts through only one concert or rehearsal, or, even worse, is ultimately unacceptable and must be discarded before any use in a concert or rehearsal.

Ceasar-Spall and Spall (1997) (CS&S) report on a study involving the application of statistical regression analysis to predicting the ultimate quality of a reed. The essential idea is to evaluate characteristics of a reed early in the reed-making process and make a prediction as to whether the reed will ultimately be successful. If the prediction indicates that the reed is likely to be unacceptable, the oboist may wish to stop working on that reed. Six input variables and one output variable are described in CS&S. Five of the input variables pertain to inherent qualities of the cane. These are termed "top close" (T), "appearance"

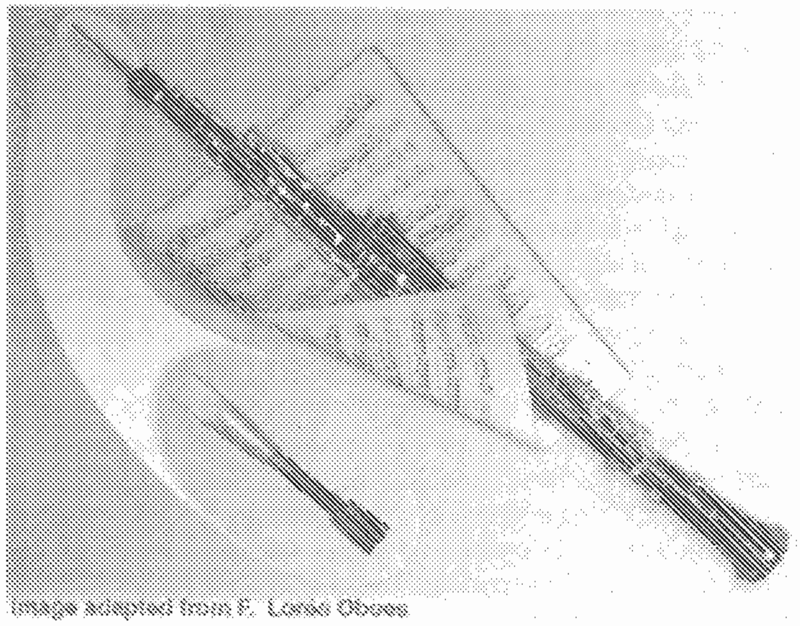

Figure 3.3. Oboe with magnification of the reed. (Reprinted with permission from the *Journal of Testing and Evaluation*, July 1997, copyright American Society for Testing and Materials.)

(A), "ease of gouge" (E), "vasculur bundles" (V), and "shininess" (S). The sixth input variable, "first blow" (F), pertains to the oboist's impression of the reed after an initial tryout about one-fourth of the way through the total reed-making process. The output variable (z_k) is a measure of quality of a finished reed.

Of course, the ratings assigned for each of the input and output variables represent a subjective impression of the oboist; this is unavoidable in such a process. The input and output variables are rated on a scale of 0 to 2, with 2 representing a greater degree of the attribute in question (so a rating of $z_k = 2$ indicates that the kth reed is of top quality). The physical interpretation of the variables and other aspects of the analysis are thoroughly described in CS&S, but the summary above is sufficient for our purposes here.

Let us focus on the use of RLS for two of the four models considered in CS&S (Exercises 3.16 and 3.17 consider LMS). The first is a standard linear model

$$z = \theta_{\text{const}} + \theta_T T + \theta_A A + \theta_E E + \theta_V V + \theta_S S + \theta_F F + v, \tag{3.25}$$

and the second is a curvilinear model

$$z = \theta_F F + \theta_{AE} AE + \theta_{TF} TF + \theta_{VS} VS + v, \tag{3.26}$$

where for convenience we have suppressed the subscript k on the input, output, and noise variables. Each of the $\boldsymbol{\theta}$ elements is labeled according to the input variable to which it is attached, with θ_{const} representing the additive constant. The corresponding parameter vector $\boldsymbol{\theta}$ for model (3.25) has seven elements (θ_{const}, θ_T, etc.), while for model (3.26) it has four elements (θ_F, θ_{AE}, etc.). For reasons of parsimony (fewer elements in $\boldsymbol{\theta}$), predictive ability, and connections to the underlying aspects of the physical process, CS&S recommend the model form in (3.26) as the preferred form from the four models considered.

Given the highly subjective nature of the process, it cannot be expected that any mathematical model will fully (or even largely) capture the input–output relationship. Using the traditional statistical measure of the amount of variability explained by a linear regression model (the coefficient of multiple determination, R^2 in the vernacular), CS&S find that about *one-third* of the variation in outputs can be explained by the models, leaving about two-thirds to unexplained phenomena or inherent randomness. (R^2 is the square of a measure of linear association between z and the input variables; see, e.g., Montgomery and Runger, 1999, pp. 521–522.) This is an example of a hard-to-model process that is typical of many in the humanities, business, and social sciences, where the human element is a significant part of the process. Tests are also performed in CS&S on an independent data set (different from that used to fit the models) and evidence is found of reasonable predictive ability for the models, but, of course, there are nontrivial uncertainties in the predictions.

Tables 3.1 and 3.2 present results comparing the batch and RLS solutions for 160 input–output points available from the fitting set, **`reeddata-fit`** (available at the book's Web site). These 160 data points are different from the fitting data used in CS&S. Initial values $\hat{\boldsymbol{\theta}}_0 = \mathbf{0}$ and $\boldsymbol{P}_0 = \boldsymbol{I}_4$ or $\boldsymbol{I}_7$ (as appropriate) are used for the RLS solutions in the tables.

Let us now compare the predictive ability of the four models in Tables 3.1 and 3.2. An independent data set of 80 input-output values, **`reeddata-test`**, is used for comparing the sample mean of the absolute prediction deviations (the sample mean of the 80 values $|\boldsymbol{h}_k^T\hat{\boldsymbol{\theta}}_{160} - z_k|$, where $\hat{\boldsymbol{\theta}}_{160}$ represents one of the four sets of parameter estimates from Tables 3.1 and 3.2; this is sometimes called the mean absolute deviation [MAD]). Table 3.3 is a summary of the mean prediction error for the four models.

A standard statistical test for comparing two means with matched samples (Appendix B, expression (B.3a)) can be used to determine whether the differences in the mean prediction errors in Table 3.3 are significant. This test is based on comparing the sample means of the absolute prediction errors for each of the 80 samples (the means of $|\boldsymbol{h}_k^T\hat{\boldsymbol{\theta}}_{160} - z_k|$). In each comparison of models

Table 3.1. Comparison of batch and RLS estimates for $\boldsymbol{\theta}$ parameters in basic linear model (3.25).

$\boldsymbol{\theta}$ parameters	Batch Model (3.25)	RLS Model (3.25)
θ_{const}	−0.156	−0.079
θ_T	0.102	0.101
θ_A	0.055	0.046
θ_E	0.175	0.171
θ_V	0.044	0.043
θ_S	0.056	0.056
θ_F	0.579	0.540

Table 3.2. Comparison of batch and RLS estimates for $\boldsymbol{\theta}$ parameters in curvilinear model (3.26).

$\boldsymbol{\theta}$ parameters	Batch Model (3.26)	RLS Model (3.26)
θ_F	0.584	0.557
θ_{AE}	0.101	0.106
θ_{TF}	0.078	0.086
θ_{VS}	0.034	0.036

Table 3.3. Mean and median absolute prediction errors for the four models from Tables 3.1 and 3.2.

	Batch Model (3.25)	RLS Model (3.25)	Batch Model (3.26)	RLS Model (3.26)
Mean	0.242	0.242	0.236	0.235
Median	0.243	0.250	0.227	0.224

from Table 3.3, it is enlightening to compute the P-value of the test statistic (the probability of seeing an outcome at least as extreme as that observed if there is, in fact, no underlying difference between the predictive abilities of the models). For example, the one-sided P-value in comparing the RLS/(3.26) model (fifth column in Table 3.3) with the RLS/(3.25) model (third column in Table 3.3) is 0.103. This value is moderately small, but is not typically considered small enough to conclude that there is a statistically significant difference between those two models (see Exercise 3.18).

3.5 CONCLUDING REMARKS

This has been the first of several chapters devoted to the study of search algorithms that are stochastic analogues of the deterministic steepest descent and Newton–Raphson algorithms. This chapter focused on estimation where the underlying system model has a linear form in $\boldsymbol{\theta}$, leading to loss functions that are quadratic in the parameters $\boldsymbol{\theta}$.

Methods of the type in this chapter are among the most widely used stochastic search and optimization techniques. As any quick Internet search will reveal, countless industrial and other systems rely on LMS, RLS, or the Kalman filter. These methods typically represent a reasonable balance between generality and ease of implementation. While the underlying linear models may be restrictive in some sense, they are general enough to apply to many practical systems, especially when care is used in defining the key variables (see Sections 13.1 and 13.2) and when the range of operation is restricted to values reasonably close to some nominal operating point. Further, the linearity provides significant advantages in implementation with respect to both derivations (e.g., easy gradient calculations) and numerical stability. Results in matrix theory (such as matrix factorizations) can be used to create numerically stable versions of methods such as the Kalman filter (e.g., Moon and Stirling, 2000, pp. 604–605).

Nonetheless, there are also a large number of problems where nonlinearity is significant to the point where the linear-based methods of this chapter do not apply. Chapters 4–10 discuss general stochastic search and optimization methods that apply in nonlinear systems.

EXERCISES

3.1 Given that the unknown true process has a form identical to the model in (3.1), show that the batch least-squares solution (3.5) is an unbiased estimator and derive its covariance matrix. Assume that the v_k are independent, identically distributed (i.i.d.) with mean 0 and variance σ^2.

3.2 Suppose that data are generated according to an AR process $x_{k+1} = -0.99x_k + w_k$, where w_k is i.i.d. $N(0, 1)$ and $x_0 = 0$. Based on a model of the same AR form as the true process with an unknown β replacing the constant -0.99, use 500 iterations of LMS to estimate β when $z_k = x_k$ for all k. Use $\hat{\boldsymbol{\theta}}_0 = 0$ and $a = 0.005$.

(a) Produce a plot showing one realization of estimates for β.

(b) Give the sample mean and approximate 95 percent confidence interval for the terminal estimate for β over 50 independent realizations (each of 500 iterations).

3.3 Suppose that random $\boldsymbol{h}_k$ are i.i.d. with mean $[2, 1]^T$ and covariance matrix $\begin{bmatrix} 3 & 1 \\ 1 & 1 \end{bmatrix}$. Based on the informal bound in Subsection 3.2.2, give an upper bound to the gain a to ensure that $E\left(\left\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*\right\|^2\right) \approx 0$ in LMS for sufficiently large k.

3.4 Consider the adaptive control problem of Example 3.4, where the model and true process have the same AR form, the unknown true values of β and γ are 0.9 and 0.5, respectively, and w_k is i.i.d. $N(0, 0.25^2)$. Given $x_0 = 0$, $\hat{\boldsymbol{\theta}}_0 = [1, 1]^T$, $d_k = \sin(\pi k/10)$, and a value of a of your choosing:

(a) Plot one realization of estimates for $\boldsymbol{\theta}$ as a function of $k = 0, 1, \ldots, 200$.

(b) Give the sample means for the terminal estimate for β and γ over 50 independent replications (each of 200 iterations).

3.5 Fill in the steps in going from (3.10) to (3.12) in the derivation of the basic RLS algorithm.

3.6 Use the matrix inversion lemma (Appendix A) to derive (3.14a) from (3.12).

3.7 Fill in the steps in deriving the multivariate form of the RLS algorithm in (3.17a, b).

3.8 Using (3.1) and assuming a true model having $p = 2$, $\boldsymbol{h}_k = [k, 1]^T$, $\boldsymbol{\theta}^* = [1, 2]^T$, and $v_k \sim N(0, 1)$, run some numerical cases comparing batch regression solutions to RLS solutions as a function of n. What is the sensitivity to $\hat{\boldsymbol{\theta}}_0$, $\boldsymbol{P}_0$, and n?

3.9 Suppose that $\boldsymbol{h}_k = [1, 1]^T$ for all k. Is RLS in (3.14a, b) implementable? If so, discuss reasons to doubt the validity of the solution. (Note that the batch least-squares solution does not exist.)

3.10 Assume the true model in Exercise 3.8, except that $\text{var}(v_k) = 1$ for k even and $\text{var}(v_k) = c$ for k odd, $c >> 1$. Compare the estimation accuracy of the unweighted RLS in (3.14a, b) with the weighted version of RLS in (3.17a, b). In particular, how do the algorithms compare in estimating $\boldsymbol{\theta}$? Try varying n, c, and the initial conditions (try $c = 100$ and 1000).

3.11 Derive the efficient matrix RLS form (3.22c) from the "basic" form (3.22b).

3.12 Explain in intuitive terms the types of problems for which each of RLS, LMS, and the Kalman filter are best suited. What types of applications are most appropriate for each of the approaches?

3.13 For a system with constant state vector and scalar measurements, comment on the connection between the RLS solution in (3.14a, b) and the Kalman filter. How do the terms in (3.14a, b) relate to the Kalman gain and error covariance matrix?

3.14 Consider a state-space model with $\boldsymbol{P}_0 = \boldsymbol{0}$, $\boldsymbol{Q}_0 = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$, and $\boldsymbol{h}_1 = [0, 1]^T$. Based on the diagonal elements of $\boldsymbol{P}_1$, comment in qualitative terms on the accuracy of the Kalman filter estimate for both elements of $\boldsymbol{\theta}_1$ as ρ approaches 0 or 1 and as σ_1^2 approaches 0 or ∞.

3.15 Reproduce the numbers in Table 3.1 using the data set **`reeddata-fit`** (from the book's Web site).

3.16 Estimate the four parameters in the curvilinear model (3.26) using basic LMS (eqn. (3.7)) with the data set **`reeddata-fit`**. Use $\hat{\boldsymbol{\theta}}_0 = \boldsymbol{0}$ as in the RLS studies and let $a = 0.005$, 0.05, and 0.10 (so you produce three $\boldsymbol{\theta}$ estimates, one for each a). Comment on the differences in the values produced by LMS and the values produced by RLS as given in Table 3.2.

3.17 In the setting of Exercise 3.16 (same $\hat{\boldsymbol{\theta}}_0$ and a), use the normalized LMS algorithm given in (3.19) and (3.20b). Comment on differences in the results from the basic LMS algorithm if you performed Exercise 3.16.

3.18 Calculate the one-sided P-value (using the matched-pairs t-test in Appendix B) for comparing the mean absolute prediction error of models batch/(3.25) and batch/(3.26) as given in the second and fourth columns of Table 3.3. Use the data set **`reeddata-test`** (from the book's Web site).