10

COMPUTING THE ESTIMATE

In Chapter 7 we gave three basic procedures for parameter estimation:

1. The prediction-error approach in which a certain function $V_N(\theta, Z^N)$ is minimized with respect to θ.
2. The correlation approach, in which a certain equation $f_N(\theta, Z^N) = 0$ is solved for θ.
3. The subspace approach to estimating state space models.

In this chapter we shall discuss how these problems are best solved numerically.

At time N, when the data set Z^N is known, the functions V_N and f_N are just ordinary functions of a finite-dimensional real parameter vector θ. Solving the problems therefore amounts to standard questions of nonlinear programming and numerical analysis. Nevertheless, it is worthwhile to consider the problems in our parameter estimation setting, since this adds a certain amount of structure to the functions involved.

The subspace methods (7.66) can be implemented in several different ways, and contain many options. In Section 10.6 we give the details of this, together with an independent derivation of the techniques.

10.1 LINEAR REGRESSIONS AND LEAST SQUARES

The Normal Equations

For linear regressions, the prediction is given as

$$\hat{y}(t|\theta) = \varphi^T(t)\theta \tag{10.1}$$

The prediction-error approach with a quadratic norm applied to (10.1) gives the least-squares method described in Section 7.3. The minimizing element $\hat{\theta}_N^{LS}$ can then be written as in (7.34):

$$\hat{\theta}_N^{LS} = R^{-1}(N)f(N) \tag{10.2}$$

with

$$R(N) = \frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t) \tag{10.3}$$

$$f(N) = \frac{1}{N}\sum_{t=1}^{N}\varphi(t)y(t) \tag{10.4}$$

An alternative is to view $\hat{\theta}_N^{LS}$ as the solution of

$$R(N)\hat{\theta}_N^{LS} = f(N) \tag{10.5}$$

These equations are known as the *normal equations*. Note that the basic equation (7.118) for the IV method is quite analogous to (10.5), and most of what is said in this section about the LS method also applies to the IV method with obvious adaptation.

The coefficient matrix $R(N)$ in (10.5) may be ill conditioned, in particular if its dimension is high. There exist methods to find $\hat{\theta}_N$ that are much better numerically behaved, which do not have the normal equations as a starting point. This has been discussed in an extensive literature on numerical studies of linear least-squares problems. Lawson and Hanson (1974)can be mentioned as a basic reference for the problems under consideration. The underlying idea in these methods is that the matrix $R(N)$ should not be formed, since it contains products of the original data. Instead, a matrix R is constructed with the property

$$RR^T = R(N)$$

Therefore, this class of methods is commonly known as "square-root algorithms" in the engineering literature. The term is not quite adequate, since no square roots are ever taken. It would be more appropriate to use the term "quadratic methods" when solving (10.5).

Solving for the LS Estimate by QR Factorization

There are some different approaches to the construction of R, such as Householder transformations, Householder (1964), the Gram-Schmidt procedure, Björck (1967), and Cholesky decomposition. We shall here describe an efficient way using QR-factorizations. See, e.g., Golub and van Loan (1996)for a thorough description of the method. The QR-factorization of an $n \times d$ matrix A is defined as

$$A = QR, \quad QQ^T = I, \qquad R \text{ upper triangular} \tag{10.6}$$

Here Q is $n \times n$ and R is $n \times d$.

To apply this to the LS parameter estimation case, we rewrite the general multivariable case (4.59) in matrix terms, by introducing

$$\begin{aligned} \mathbf{Y}^T &= \begin{bmatrix} y^T(1) & \dots & y^T(N) \end{bmatrix}, \quad \mathbf{Y} \text{ is } Np \times 1 \\ \Phi^T &= \begin{bmatrix} \varphi(1) & \dots & \varphi(N) \end{bmatrix}, \quad \Phi \text{ is } Np \times d \end{aligned} \tag{10.7}$$

(Here $p = \dim y$). Then the LS criterion can be written (cf. (II.13))

$$V_N(\theta, Z^N) = |\mathbf{Y} - \Phi\theta|^2 = \sum_{t=1}^{N} \left| y(t) - \varphi^T(t)\theta \right|^2 \tag{10.8}$$

The norm is obviously not affected by any orthonormal transformation applied to the vector $\mathbf{Y} - \Phi\theta$. Therefore if Q $(pN \times pN)$ is orthonormal, that is $QQ^T = I$, then

$$V_N(\theta, Z^N) = |Q(\mathbf{Y} - \Phi\theta)|^2$$

Now, introduce the QR-factorization

$$[\Phi \quad \mathbf{Y}] = QR, \quad R = \begin{bmatrix} R_0 \\ \dots \\ 0 \end{bmatrix} \tag{10.9}$$

Here R_0 is an upper triangular $(d+1) \times (d+1)$ matrix, which we decompose as

$$R_0 = \begin{bmatrix} R_1 & R_2 \\ 0 & R_3 \end{bmatrix}, \quad R_1 \text{ is } d \times d, \quad R_2 \text{ is } d \times 1, \quad R_3 \text{ is scalar} \tag{10.10}$$

This means that

$$V_N(\theta, Z^N) = \left| Q^T(\mathbf{Y} - \Phi\theta) \right|^2 = \left| \begin{bmatrix} R_2 \\ R_3 \end{bmatrix} - \begin{bmatrix} R_1\theta \\ 0 \end{bmatrix} \right|^2 = |R_2 - R_1\theta|^2 + |R_3|^2$$

which clearly is minimized for

$$R_1\hat{\theta}_N = R_2, \quad \text{giving} \quad V_N(\hat{\theta}_N, Z^N) = |R_3|^2 \tag{10.11}$$

There are three important advantages with this way of solving for the LS estimate:

1. With $R(N)$ as in (10.3), $R(N) = \Phi^T\Phi = R_1^T R_1$ so the conditioning number (the ratio between the largest and smallest singular value) of R_1 is the square root of that of $R(N)$. Therefore (10.11) is much better conditioned than its counterpart (10.5).
2. R_1 is a triangular matrix, so the equation is easy to solve.
3. If the QR-factorization is performed for a regressor size d^*, then the solutions and loss functions for all models with fewer parameters—obtained by setting trailing parameters in θ to zero—are easily obtained from R_0.

Note that the big matrix Q is never required to find $\hat{\theta}_N$ and the loss function. All information is contained in the "small" matrix R_0. In MATLAB considerable computational saving is obtained by taking `R=triu(qr(A))` to compute (10.6) if Q is of no interest, and A has many more rows than columns.

Initial Conditions: "Windowed" Data

A typical structure of the regression vector $\varphi(t)$ is that it consists of shifted data (possibly after some trivial reordering):

$$\varphi(t) = \begin{bmatrix} z(t-1) \\ \vdots \\ z(t-n) \end{bmatrix} \tag{10.12}$$

Here $z(t-1)$ is an r-dimensional vector. For example, the ARX model (4.11) with $n_a = n_b = n$ gives (10.12) with

$$z(t) = \begin{bmatrix} -y(t) \\ u(t) \end{bmatrix}$$

while an AR-model for a p-dimensional process $\{y(t)\}$ [cf. (4.57), $n_b = 0$] obeys (10.12) with $z(t) = -y(t)$.

With the structure (10.12), the matrix $R(N)$ in (10.3) will be an $n \times n$ block matrix, whose ij block is the $r \times r$ matrix

$$R_{ij}(N) = \frac{1}{N}\sum_{t=1}^{N} z(t-i)z^T(t-j) \tag{10.13}$$

If we have knowledge only of $z(t)$ for $1 \le t \le N$, the question arises of how to deal with the unknown initial conditions for $t \le 0$ in (10.13). Two approaches can be taken:

1. Start the summation in (10.3) and (10.4) at $t = n+1$ rather than at $t = 1$. Then all sums (10.13) will involve only known data. [After a suitable redefinition of N and the time origin, we can of course stick to our usual expressions, assuming $z(t)$ to be known for $t \ge -n$.]
2. Replace the unknown initial values by zeros ("prewindowing"). For symmetry, the trailing values $z(t)$, $t = N+1, \ldots, N+n$, could also be replaced by zeros ("postwindowing") and the summation in (10.3) is extended to $N+n$. In this case (10.5) are also known as the Yule-Walker equations. Often additional data windows ("tapering"; cf. Problem 6G.5) are applied to both ends of the data record to soften the effects of the appended zeros.

In speech processing these approaches are known as the *covariance method* and *autocorrelation method*, respectively (Makhoul and Wolf, 1972). No doubt, from a logical point of view, approach 1 seems the most natural. Approach 2, however, gives the special feature that the blocks (10.13) will depend only on the difference between the indexes:

$$R_{ij}(N) = R_\tau(N), \qquad \tau = i - j$$

$$R_\tau(N) = \frac{1}{N}\sum_{t=\tau}^{N} z(t-\tau)z^T(t) \qquad \tau \geq 0, \text{ analogously for } \tau < 0 \tag{10.14}$$

This makes $R(N)$ a *block Toeplitz matrix*, which gives distinct advantages when solving (10.5), as we shall demonstrate shortly.

Clearly, when $N \gg n$, the difference between the two approaches becomes insignificant.

Levinson Algorithm (∗)

The shift structure (10.12) gives a specific structure for the matrix $R(N)$. There is an extensive literature on *fast algorithms* that utilize such structures. The simplest, but generic, example of these methods is the *Levinson algorithm* (Levinson, 1947), which we shall now derive.

Consider the case of an AR model of a signal

$$\hat{y}^n(t|\theta) = -a_1^n y(t-1) - \cdots - a_n^n y(t-n) \tag{10.15}$$

(the upper index n indicates that we are fitting an nth-order model). This corresponds to a linear regression with $\varphi(t)$ subject to (10.12) with $z(t) = -y(t)$. If we apply the autocorrelation method, we should solve (10.5); that is,

$$\begin{bmatrix} R_0 & R_1 \dots & R_{n-1} \\ R_1 & R_0 \dots & R_{n-2} \\ \vdots & \vdots & \vdots \\ R_{n-1} & R_{n-2} \dots R_0 \end{bmatrix} \begin{bmatrix} a_1^n \\ a_2^n \\ \vdots \\ a_n^n \end{bmatrix} = \begin{bmatrix} -R_1 \\ -R_2 \\ \vdots \\ -R_n \end{bmatrix} \tag{10.16}$$

for a_i^n. Here

$$R_\tau = \hat{R}_y^N(\tau) = \frac{1}{N}\sum_{t=\tau}^{N} y(t-\tau)y(t), \qquad \tau \geq 0 \tag{10.17}$$

and we have dropped the argument N. Equation (10.16) can be rewritten as

$$\begin{bmatrix} R_0 & R_1 \dots & R_n \\ R_1 & R_0 \dots & R_{n-1} \\ R_2 & R_1 \dots & R_{n-2} \\ \vdots & \vdots & \vdots \\ R_n & R_{n-1} \dots & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1^n \\ a_2^n \\ \vdots \\ a_n^n \end{bmatrix} = \begin{bmatrix} V_n \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{10.18}$$

Here the n last rows are identical to (10.16), while the first row is a definition of V_n.

Suppose now that we have solved (10.18) for a_i^n and seek a solution for a higher-order model (10.15) with order $n+1$. The estimates a_i^{n+1} will then be defined analogously to (10.18). To find these, we first note that

$$\begin{bmatrix} R_0 & R_1 \dots & R_n & R_{n+1} \\ R_1 & R_0 \dots & R_{n-1} & R_n \\ \vdots & \vdots & \vdots & \vdots \\ R_n & R_{n-1} \dots & R_0 & R_1 \\ R_{n+1} & R_n \dots & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1^n \\ \vdots \\ a_n^n \\ 0 \end{bmatrix} = \begin{bmatrix} V_n \\ 0 \\ \vdots \\ 0 \\ \alpha_n \end{bmatrix} \tag{10.19}$$

Here the first $n+1$ rows are identical to (10.18), while the last row is a definition of α_n. The definition of $\hat{a}_i^{n+1}$ looks quite like (10.19), the only difference being that all but the first row of the right side should be zero. We thus seek to remove α_n. A moment's reflection on (10.19) shows that it can also be written as

$$\begin{bmatrix} R_0 & R_1 \dots & R_n & R_{n+1} \\ R_1 & R_0 \dots & R_{n-1} & R_n \\ \vdots & \vdots & \vdots & \vdots \\ R_n & R_{n-1} \dots & R_0 & R_1 \\ R_{n+1} & R_n \dots & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 0 \\ a_n^n \\ \vdots \\ a_1^n \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_n \\ 0 \\ \vdots \\ 0 \\ V_n \end{bmatrix} \tag{10.20}$$

since the coefficient matrix is a symmetric Toeplitz matrix. We can also view the last $n+1$ rows of (10.20) as the normal equations for the regression

$$\check{y}(t-n-1|\theta) = -a_1^n y(t-n) - a_2^n y(t-n+1) - \cdots - a_n^n y(t-1) \tag{10.21}$$

This is a reversed time model for the signal $y(t)$. Since the second order properties of a scalar stationary signal are symmetric with respect to the direction of time, the coefficients in (10.21) coincide with those of (10.15). This is a signal theoretic reason for the equality between (10.19) and (10.20). See also the Remark at the end of this subsection.

Now multiply (10.20) by $\rho_n = -\alpha_n / V_n$ and add it to (10.19). This gives

$$\begin{bmatrix} R_0 & R_1 \dots & R_n & R_{n+1} \\ R_1 & R_0 \dots & R_{n-1} & R_n \\ \vdots & \vdots & \vdots & \vdots \\ R_n & R_{n-1} \dots & R_0 & R_1 \\ R_{n+1} & R_n \dots & R_1 & R_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1^n + \rho_n a_n^n \\ \vdots \\ a_n^n + \rho_n a_1^1 \\ \rho_n \end{bmatrix} = \begin{bmatrix} V_n + \rho_n \cdot \alpha_n \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{10.22}$$

This is the defining relationship for $\hat{a}_i^{n+1}$. Hence

$$\begin{aligned} \hat{a}_k^{n+1} &= \hat{a}_k^n + \hat{\rho}_n \hat{a}_{n-k+1}^n, \quad k = 1, \dots, n \\ \hat{a}_{n+1}^{n+1} &= \hat{\rho}_n \\ V_{n+1} &= V_n + \hat{\rho}_n \alpha_n \\ \hat{\rho}_n &= \frac{-\alpha_n}{V_n} \\ \alpha_n &= R_{n+1} + \sum_{k=1}^{n} \hat{a}_k^n R_{n+1-k} \end{aligned} \tag{10.23}$$

(The hat here indicates the actual estimate, based on N data, as opposed to the general model parameters a_k^n.) This expression allows us to easily compute $\hat{a}_k^{n+1}$ from $\hat{a}_k^n$. With the initial conditions

$$\begin{aligned} V_1 &= R_0 - \frac{R_1^2}{R_0} \\ \hat{a}_1^1 &= \frac{-R_1}{R_0} \end{aligned} \tag{10.24}$$

we have a scheme for computing estimates of arbitrary orders. We note that going from $\hat{a}_k^n$ to $\hat{a}_k^{n+1}$ in (10.23) requires $4n + 2$ additions and multiplications and one division. The computation of $\hat{a}_k^n$ thus requires proportional to $2n^2$ operations, which is of an order of magnitude (in n) less than the general procedures (10.8) to (10.11). Hence the term "fast algorithms."

The Levinson algorithm (10.23) has been widely applied and extended to the case of vector-valued z, as well as to "the covariance method." See, for example, Whittle (1963), Wiggins and Robinson (1965), and Morf et.al. (1977).

Remark. The most important change when dealing with vector-valued z is that the corresponding reversed-time model (10.21)

$$\check{z}(t-n-1|\theta) = -b_1^n z(t-n) - \cdots - b_n^n z(t-1) \tag{10.25}$$

gives b_i estimates that differ from a_i. The scheme (10.23) must then be complemented with an analogous scheme for updating the b_i. See Problem 10G.1.

Lattice Filters (∗)

Consider the predictors (10.15) for orders n and $n+1$ evaluated at

$$\begin{aligned} \theta = \hat{\theta} &= \hat{\theta}_N \\ \hat{y}_n(t|\hat{\theta}^n) &= -\hat{a}_1^n y(t-1) - \cdots - \hat{a}_n^n y(t-n) \\ \hat{y}_{n+1}(t|\hat{\theta}^{n+1}) &= -\hat{a}_1^{n+1} y(t-1) - \cdots - \hat{a}_n^{n+1} y(t-n) \\ &\quad - \hat{a}_{n+1}^{n+1} y(t-n-1) \end{aligned} \tag{10.26}$$

Subtracting these expressions from each other gives, using (10.23),

$$\hat{y}_{n+1}(t|\hat{\theta}^{n+1}) = \hat{y}_n(t|\hat{\theta}^n) - \hat{\rho}_n \hat{r}_n(t-1) \tag{10.27}$$

where

$$\hat{r}_n(t-1) = y(t-n-1) + \hat{a}_1^n y(t-n) + \cdots + \hat{a}_n^n y(t-1) \tag{10.28}$$

We recognize in (10.28) the error in the reversed-time predictor (10.21). Let us, with the definition (10.28), consider

$$\hat{r}_{n+1}(t) = y(t-n-1) + \hat{a}_1^{n+1} y(t-n) + \cdots + \hat{a}_n^{n+1} y(t-1) + \hat{a}_{n+1}^{n+1} y(t)$$

Subtracting (10.28) from this gives, again using (10.23),

$$\hat{r}_{n+1}(t) - \hat{r}_n(t-1) = \hat{\rho}_n \left[y(t) + \hat{a}_1^n y(t-1) + \cdots + \hat{a}_n^n y(t-n) \right] \tag{10.29}$$

With the prediction error

$$\hat{e}_n(t) = y(t) - \hat{y}_n(t|\hat{\theta}^n)$$

the expressions (10.27) to (10.29) can be summarized as

$$\hat{e}_{n+1}(t) = \hat{e}_n(t) + \hat{\rho}_n \hat{r}_n(t-1) \tag{10.30a}$$

$$\hat{r}_{n+1}(t) = \hat{r}_n(t-1) + \hat{\rho}_n \hat{e}_n(t) \tag{10.30b}$$

$$\hat{e}_0(t) = \hat{r}_0(t) = y(t) \tag{10.30c}$$

This simple representation of the prediction errors [and the predictions $\hat{y}_n(t|\hat{\theta}^n) = y(t) - \hat{e}_n(t)$] can be graphically represented as in Figure 10.1. Because of the structure of this representation, (10.30) is usually called a *lattice filter* (sometimes a *ladder filter*).

An important feature of the representation (10.30) is that the variables $\hat{e}_n$ and $\hat{r}_n$ obey the following orthogonality relationships:

$$\begin{aligned}
\frac{1}{N}\sum_{t=1}^{N}\hat{e}_n(t)\hat{e}_{n-k}(t-k) &= \begin{cases} V_n, & \text{if } k = 0 \\ 0, & \text{if } k \neq 0 \end{cases} \\
\frac{1}{N}\sum_{t=1}^{N}\hat{r}_n(t)\hat{r}_k(t) &= \begin{cases} V_n, & \text{if } n = k \\ 0, & \text{if } n \neq k \end{cases} \\
\frac{1}{N}\sum_{t=1}^{N}\hat{e}_n(t)\hat{r}_k(t-1) &= \begin{cases} \alpha_n, & \text{if } n = k \\ 0, & \text{if } n > k \end{cases}
\end{aligned} \tag{10.31}$$

(see Problem 10D.1). Hence the *reflection coefficients* $\hat{\rho}_n$ can easily be computed as

$$\hat{\rho}_n = \frac{-\sum_{t=1}^{N}\hat{e}_n(t)\hat{r}_n(t-1)}{\sum_{t=1}^{N}\hat{e}_n^2(t)} \tag{10.32}$$

The scheme (10.30) together with (10.32) also forms an efficient way of estimating the reflection coefficients $\hat{\rho}_n$, as well as the predictions, as an alternative to the Levinson algorithm. An important aspect is that the scheme produces all lower-order predictors as a by-product. Lattice filters have been used extensively in signal-processing applications. See, for example, Makhoul (1977), Griffiths (1977), and Lee, Morf, and Friedlander (1981). See also Section 11.7 for recursive versions.

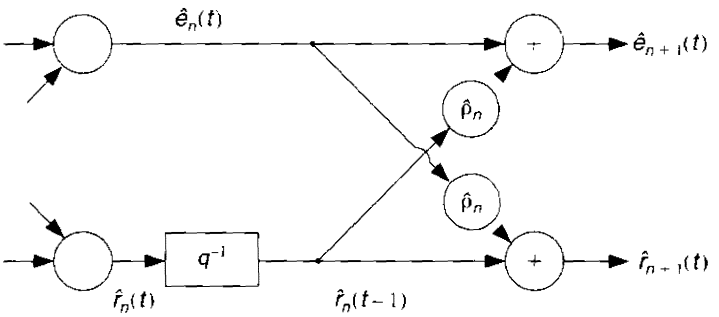

Figure 10.1 A lattice filter representation.

10.2 NUMERICAL SOLUTION BY ITERATIVE SEARCH METHODS

In general, the function

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N} \ell(\varepsilon(t,\theta),\theta) \tag{10.33}$$

cannot be minimized by analytical methods. Neither can the equation

$$0 = f_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N} \zeta(t,\theta)\alpha(\varepsilon(t,\theta)) \tag{10.34}$$

be solved by direct means in general. The solution then has to be found by iterative, numerical techniques. There is an extensive literature on such numerical problems. See, for example, Luenberger (1973), Bertsekas (1982), or Dennis and Schnabel (1983)for general treatments.

Numerical Minimization

Methods for numerical minimization of a function $V(\theta)$ update the estimate of the minimizing point iteratively. This is usually done according to

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} + \alpha f^{(i)} \tag{10.35}$$

where $f^{(i)}$ is a search direction based on information about $V(\theta)$ acquired at previous iterations, and α is a positive constant determined so that an appropriate decrease in the value of $V(\theta)$ is obtained. Depending on the information supplied by the user to determine $f^{(i)}$, numerical minimization methods can be divided into three groups:

1. Methods using function values only.
2. Methods using values of the function V as well as of its gradient.
3. Methods using values of the function, of its gradient, and of its Hessian (the second derivative matrix).

The typical member of group 3 corresponds to *Newton algorithms*, where the correction in (10.35) is chosen in the "Newton" direction:

$$f^{(i)} = -\left[V''(\hat{\theta}^{(i)})\right]^{-1} V'(\hat{\theta}^{(i)}) \tag{10.36}$$

The most important subclass of group 2 consists of *quasi-Newton methods*, which somehow form an estimate of the Hessian and then use (10.36). Algorithms of group 1 either form gradient estimates by difference approximations and proceed as quasi-Newton methods or have other specific search patterns. See Powell (1964).

Many standard programs implementing these ideas are available. The easiest way for the identification user could be to supply such a program with necessary information and leave the search for the minimum to the program. In any case, it will be necessary to compute the function values of (10.33) for any required value

of θ. The major burden for this lies in the calculation of the sequence of prediction errors $\varepsilon(t, \theta)$, $t = 1, \ldots, N$. This itself could be a simple or complicated task. Compare, for example, the model structures of Section 4.2 with the one in Example 4.1.

The gradient of (10.33) is

$$V_N'(\theta, Z^N) = -\frac{1}{N}\sum_{t=1}^{N}\left[\psi(t, \theta)\ell_\varepsilon'(\varepsilon(t, \theta), \theta) - \ell_\mu'(\varepsilon(t, \theta), \theta)\right] \tag{10.37}$$

Here, as usual, $\psi(t, \theta)$ is the $d \times p$ gradient matrix of $\hat{y}(t|\theta)$ ($p = \dim y$) with respect to θ. The major computational burden in (10.37) lies in the calculation of the sequence $\psi(t, \theta)$, $t = 1, 2, \ldots, N$. We discuss in Section 10.3 how this gradient is computed for some common model structures. However, for some models, direct calculation of ψ could be forbidding, and then one has to resort to the minimization methods of the group 1 or to form estimates of ψ by difference approximations.

Some Explicit Search Schemes

Consider the special case of scalar output and quadratic criterion

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\tfrac{1}{2}\varepsilon^2(t, \theta) \tag{10.38}$$

This problem is known as "the nonlinear least-squares problem" in numerical analysis. An excellent and authoritative account of this problem is given in Chapter 10 of Dennis and Schnabel (1983). The criterion (10.38) has the gradient

$$V_N'(\theta, Z^N) = -\frac{1}{N}\sum_{t=1}^{N}\psi(t, \theta)\varepsilon(t, \theta) \tag{10.39}$$

A general family of search routines is then given by

$$\hat{\theta}_N^{(i+1)} = \hat{\theta}_N^{(i)} - \mu_N^{(i)}[R_N^{(i)}]^{-1}V_N'(\hat{\theta}_N^{(i)}, Z^N) \tag{10.40}$$

where $\hat{\theta}_N^{(i)}$ denotes the ith iterate. $R_N^{(i)}$ is a $d \times d$ matrix that modifies the search direction (it will be discussed later), and the step size $\mu_N^{(i)}$ is chosen so that

$$V_N(\hat{\theta}_N^{(i+1)}, Z^N) < V_N(\hat{\theta}_N^{(i)}, Z^N) \tag{10.41}$$

We should also keep in mind that the minimization problem normally is a constrained one: $\theta \in D_{\mathcal{M}}$. Often, however, $\partial D_{\mathcal{M}}$, the boundary of $D_{\mathcal{M}}$, corresponds to the stability boundary of the predictor (cf. Definition 4.1) so that $V_N(\theta, Z^N)$ increases rapidly as θ approaches $\partial D_{\mathcal{M}}$. Then the constraint can easily be obeyed by proper selection of μ.

The simplest choice of $R_N^{(i)}$ is to take it as the identity matrix,

$$R_N^{(i)} = I \tag{10.42}$$

which makes (10.40) the *gradient* or steepest-descent method. This method is fairly inefficient close to the minimum. Newton methods typically perform much better there. For (10.38), the Hessian is

$$V_N''(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N} \psi(t,\theta)\psi^T(t,\theta) - \frac{1}{N}\sum_{t=1}^{N} \psi'(t,\theta)\varepsilon(t,\theta) \tag{10.43}$$

where $\psi'(t,\theta)$ is the $d \times d$ Hessian of $\varepsilon(t,\theta)$.

Choosing

$$R_N^{(i)} = V_N''(\hat{\theta}_N^{(i)}, Z^N) \tag{10.44}$$

makes (10.40) a Newton method. It may, however, be quite costly to compute all the terms of ψ'. Suppose now that there is a value θ_0 such that the prediction errors $\varepsilon(t,\theta_0) = e_0(t)$ are independent. Then this value yields the global minimum of $EV_N(\theta, Z^N)$. Close to θ_0 the second sum of (10.43) will then be close to zero since $E\psi'(t,\theta_0)e_0(t) = 0$. We thus have

$$V_N''(\theta, Z^N) \approx \frac{1}{N}\sum_{1}^{N} \psi(t,\theta)\psi^T(t,\theta) \stackrel{\Delta}{=} H_N(\theta) \tag{10.45}$$

If we apply a Newton method to the minimization problem, we need a good estimate of the Hessian only in the vicinity of the minimum. The reason for this is that Newton methods are designed to give one-step convergence for quadratic functions. When the function values between the current iterate and the minimum cannot be approximated very well by a quadratic function, the effect of the Hessian in (10.36) is not so important. Moreover, by omitting the last sum in (10.43) the estimate of the Hessian is always assured to be positive semidefinite. This makes the numerical procedure a descent algorithm and guarantees convergence to a stationary point. The conclusion is consequently that

$$R_N^{(i)} = H_N(\hat{\theta}_N^{(i)}) \tag{10.46}$$

is a quite suitable choice for our problem. This is also known as the *Gauss-Newton method*. In the statistical literature the technique is called "the method of scoring." (Rao, 1973). In the control literature the terms modified Newton-Raphson and quasi-linearization have also been used. Dennis and Schnabel (1983)reserve the term Gauss-Newton for (10.40) and (10.46) with the particular choice $\mu_N^{(i)} = 1$, and suggest the term *damped Gauss-Newton* when an adjusted step size μ is applied.

Even though the expression (10.45) is assured to be positive semidefinite, it may be singular or close to singular. This is the case, for example, if the model is over-parametrized or the data not informative enough. Then some numerical problems

arise in (10.40). Various ways to overcome this problem exist and are known as *regularization* techniques. One common way is the *Levenberg-Marquardt* procedure (Levenberg, 1944; Marquardt, 1963). Then an approximation

$$R_N^{(i)}(\lambda) = \frac{1}{N}\sum_{t=1}^{N} \psi(t, \hat{\theta}_N^{(i)})\psi^T(t, \hat{\theta}_N^{(i)}) + \lambda I \tag{10.47}$$

is used for the Hessian. Here λ is a positive scalar that is used to control the convergence in the iterative scheme rather than the step size parameter. We thus have $\mu_N^{(i)} = 1$ in (10.40). With $\lambda = 0$ we have the Gauss-Newton case. Increasing λ means that the step size is decreased and the search direction is turned towards the gradient. Several schemes for manipulating λ based on the test (10.41) have been suggested, see, e.g., Scales (1985).

If the minimum does not give independent prediction errors, the second sum of (10.43) then need not be negligible close to the minimum, and (10.45) need not be a good approximation of the Hessian. A typical method then is to make use of the known first sum of the Hessian and estimate the second sum with some secant technique (see Dennis, Gay, and Welsch, 1981).

Correlation Equation

Solving the equation (10.34) is quite analogous to the minimization of (10.33) (see, e.g., Dennis and Schnabel, 1983). Standard numerical procedures are the *substitution method* (corresponding to (10.40) and (10.42)):

$$\hat{\theta}_N^{(i)} = \hat{\theta}_N^{(i-1)} - \mu_N^{(i)} f_N(\hat{\theta}_N^{(i-1)}, Z^N) \tag{10.48}$$

and the *Newton-Raphson method* [corresponding to (10.40) and (10.44)]:

$$\hat{\theta}_N^{(i)} = \hat{\theta}_N^{(i-1)} - \mu_N^{(i)} \left[f_N'(\hat{\theta}_N^{(i-1)}, Z^N)\right]^{-1} f_N(\hat{\theta}_N^{(i-1)}, Z^N) \tag{10.49}$$

10.3 COMPUTING GRADIENTS

To use the formulas in the previous section, we need expressions for $\psi(t, \theta)$, the gradient of the prediction. The amount of work required to compute $\psi(t, \theta)$ is highly dependent on the model structure, and sometimes one may have to resort to numerical differentiation. In this section we shall provide expressions for some common model structures.

Example 10.1 The ARMAX Model Structure

Consider the ARMAX model (4.14). The predictor is given by (4.18):

$$C(q)\hat{y}(t|\theta) = B(q)u(t) + [C(q) - A(q)]y(t) \tag{10.50}$$

Differentiating this expression with respect to a_k gives

$$C(q)\frac{\partial}{\partial a_k}\hat{y}(t|\theta) = -q^{-k}y(t) \tag{10.51}$$

Similarly,

$$C(q)\frac{\partial}{\partial b_k}\hat{y}(t|\theta) = -q^{-k}u(t)$$

and

$$q^{-k}\hat{y}(t|\theta) + C(q)\frac{\partial}{\partial c_k}\hat{y}(t|\theta) = -q^{-k}y(t)$$

With the vector $\varphi(t, \theta)$ defined by (4.20), these expressions can be rewritten conveniently as

$$C(q)\psi(t, \theta) = \varphi(t, \theta) \tag{10.52}$$

The gradient is thus obtained by filtering the "regression vector" $\varphi(t, \theta)$ through the filter $1/C(q)$. This filter is stable for all θ for which the predictor (10.50) is stable.

SISO Black-box Model (∗)

Most formulas for SISO black-box models will be contained in a treatment of the general model (4.33). The predictor for this model is given by (4.35):

$$\hat{y}(t|\theta) = \frac{D(q)B(q)}{C(q)F(q)}u(t) + \left[1 - \frac{D(q)A(q)}{C(q)}\right]y(t) \tag{10.53}$$

from which we find, as in Example 10.1, that

$$\frac{\partial}{\partial a_k}\hat{y}(t|\theta) = -\frac{D(q)}{C(q)}y(t-k) \tag{10.54a}$$

$$\frac{\partial}{\partial b_k}\hat{y}(t|\theta) = -\frac{D(q)}{C(q)F(q)}u(t-k) \tag{10.54b}$$

$$\begin{aligned}\frac{\partial}{\partial c_k}\hat{y}(t|\theta) &= -\frac{D(q)B(q)}{C(q)C(q)F(q)}u(t-k) + \frac{D(q)A(q)}{C(q)C(q)}y(t-k) \\ &= \frac{1}{C(q)}\varepsilon(t-k, \theta)\end{aligned} \tag{10.54c}$$

$$\begin{aligned}\frac{\partial}{\partial d_k}\hat{y}(t|\theta) &= \frac{B(q)}{C(q)F(q)}u(t-k) - \frac{A(q)}{C(q)}y(t-k) \\ &= -\frac{1}{C(q)}v(t-k, \theta)\end{aligned} \tag{10.54d}$$

$$\begin{aligned}\frac{\partial}{\partial f_k}\hat{y}(t|\theta) &= -\frac{D(q)B(q)}{C(q)F(q)F(q)}u(t-k) \\ &= \frac{D(q)}{C(q)F(q)}w(t-k, \theta)\end{aligned} \tag{10.54e}$$

where we used ε, v, and w as defined by (4.37) to (4.39). The gradient $\psi(t, \theta)$ is thus also in this case obtained by filtering the regression vector $\varphi(t, \theta)$ [defined in (4.40)] through linear filters, although different parts of φ are associated with different filters in the general case. It is clear that the filters involved here are all stable for $\theta \in D_{\mathcal{M}}$, defined in Lemma 4.1, which are also the θ for which the predictors are stable.

In the special case of an output error model, $A(q) = C(q) = D(q) = 1$, [see also (4.25)], we obtain from (10.54b, e)

$$F(q)\psi(t, \theta) = \varphi(t, \theta) \tag{10.55}$$

General Finite-dimensional Linear Time-invariant Models (∗)

A linear time-invariant finite-dimensional model can always be represented as

$$\begin{aligned} \varphi(t + 1, \theta) &= \mathcal{F}(\theta)\varphi(t, \theta) + \mathcal{G}(\theta)\begin{bmatrix} y(t) \\ u(t) \end{bmatrix} \\ \hat{y}(t|\theta) &= \mathcal{H}(\theta)\varphi(t, \theta) \end{aligned} \tag{10.56}$$

with proper choices of the matrices $\mathcal{F}$, $\mathcal{G}$, and $\mathcal{H}$ and with dim $\varphi = n$. This is true for the general SISO model (4.35) for which $\varphi(t, \theta)$ can be chosen as (4.40), as well as for the general state-space model (4.86) for which

$$\mathcal{F}(\theta) = A(\theta) - K(\theta)C(\theta) \tag{10.57a}$$

$$\mathcal{G}(\theta) = \begin{bmatrix} K(\theta) & B(\theta) \end{bmatrix} \tag{10.57b}$$

$$\mathcal{H}(\theta) = C(\theta), \quad \hat{x}(t, \theta) = \varphi(t, \theta) \tag{10.57c}$$

Stability of the predictor (10.56) requires θ to belong to

$$D_{\mathcal{M}} = \{\theta | \mathcal{F}(\theta) \text{ has all eigenvalues inside the unit circle}\} \tag{10.58}$$

The equation (10.56) can now be differentiated with respect to θ. Introducing

$$\xi(t, \theta) = [\varphi^T(t, \theta) \quad \tfrac{\partial}{\partial \theta_1}\varphi^T(t, \theta) \quad \dots \quad \tfrac{\partial}{\partial \theta_d}\varphi^T(t, \theta)]^T \tag{10.59}$$

we may, for some matrices $\mathcal{A}(\theta)$, $\mathcal{B}(\theta)$, and $C(\theta)$, write

$$\begin{aligned} \xi(t + 1, \theta) &= \mathcal{A}(\theta)\xi(t, \theta) + \mathcal{B}(\theta)\begin{bmatrix} y(t) \\ u(t) \end{bmatrix} \\ \begin{bmatrix} \hat{y}(t|\theta) \\ \psi(t, \theta) \end{bmatrix} &= C(\theta)\xi(t, \theta) \end{aligned} \tag{10.60}$$

It can readily be verified that the $(d + 1)n \times (d + 1)n$ matrix $\mathcal{A}(\theta)$ will contain the matrix $\mathcal{F}(\theta)$ in each of its $d + 1$ block-diagonal entries and has all zeros above the block diagonal. Hence the stability properties of $\mathcal{A}(\theta)$ coincide with those of $\mathcal{F}(\theta)$.

We will find (10.60) useful for developing general algorithms. Clearly, however, these equations, as given, will not be suited for practical calculations of the gradient, since the filter is of order $d(n+1)$. It has been shown, though, by Gupta and Mehra (1974)that the dimension of the controllability subspace of (10.60) does not exceed $4n$ regardless of the parametrization and regardless of the value of θ. By proper transformations, the necessary calculations involved in (10.60) can thus be substantially reduced.

Finally, we note that when the methods of Section 10.2 are used the expressions of the present section have to be used between each iteration to compute $\psi(t, \hat{\theta}_N^{(i)})$. This means that we have to run the data from $t = 1$ to N through filters like (10.54) or (10.60) for each iteration in (10.40).

Nonlinear Black-Box Models and Back-Propagation

In connection with neural networks, the celebrated *Back-Propagation (BP) algorithm* is used to compute the gradients of the predictor. Back-propagation has been described in several contexts, see e.g., Werbos (1974), Rumelhart, Hinton, and Cybenko (1986). Sometimes in the neural network literature the entire search algorithm is called Back-Propagation. It is, however, more consistent to keep this notation just for the algorithm used to calculate the gradient.

Consider the general nonlinear black-box structure (5.43):

$$g(\varphi, \theta) = \sum_{k=1}^{n} \alpha_k \kappa(\beta_k(\varphi - \gamma_k)) \tag{10.61}$$

To find the gradient $\psi(t, \theta) = (d/d\theta)g(\varphi, \theta)$ for this one hidden layer network is quite simple. We just need to compute

$$\frac{d}{d\alpha}\alpha\kappa(\beta\varphi - \gamma) = \kappa(\beta\varphi - \gamma)$$

$$\frac{d}{d\gamma}\alpha\kappa(\beta\varphi - \gamma) = -\alpha\kappa'(\beta\varphi - \gamma)$$

$$\frac{d}{d\beta}\alpha\kappa(\beta\varphi - \gamma) = \alpha\kappa'(\beta\varphi - \gamma)\varphi$$

The BP algorithm in this case means that the factor $\alpha\kappa'(\beta\varphi - \gamma)$ from the derivative with respect to γ is re-used in the calculation of the derivative with respect to β.

The Back-Propagation algorithm is however very general and not limited to one-hidden-layer sigmoid neural network models. Instead, it applies to all network models and it can be described as the chain rule for differentiation applied to the expression (5.47) with a smart re-use of intermediate results which are needed at several places in the algorithm. For ridge construction models (5.42) where β_i is a parameter vector, the only complicated thing with the algorithm is actually to keep track of all indexes. When β_i is a parameter matrix, like in the radial approach (5.40), then the calculation becomes somewhat more complicated, but the basic procedure remains the same. See Saarinen, Bramley, and Cybenko (1993) for an illuminating description of these general aspects.

When shifting to multi-layer network models, the possibilities of re-using intermediate result increase and so does the importance of the BP algorithm.

For recurrent models the calculation of the gradient becomes more complicated. The gradient $\psi(t, \theta)$ at one time instant does not only depend on the regressor $\varphi(t, \theta)$ but also on the gradient at the previous time instant $\psi(t-1, \theta)$. See Nerrand et.al. (1993)for a discussion on this topic. The additional problem to calculate the gradient does, however, not change anything essential in the minimization algorithm. In the neural network literature this is often referred to as *Back-propagation through time.*

Recursive Techniques for Off-line Problems

An idea to save work in the minimization procedure could be to combine the methods of Sections 10.2 and 10.3 so as to modify the estimate $\hat{\theta}_N^{(i)}$ in (10.40) at the same time as the prediction error gradients are computed in (10.60) [i.e., to "link the index (i) to N"]. We shall develop such *recursive* algorithms in Chapter 11 for on-line applications. These are, however, quite useful also for off-line problems as alternatives to (10.40). Then typically the data record is run through the recursive algorithm a couple of times, and it can be shown that such a procedure will have the same convergence properties as (10.40). See Ljung and Söderström (1983), Section 7.2, and Solbrand, Ahlén, and Ljung (1985)for further details.

10.4 TWO-STAGE AND MULTISTAGE METHODS

The techniques described in Sections 10.2 and 10.3 should be regarded as the basic numerical methods for parameter estimation. They have the advantages of guaranteed convergence (to a local minimum), efficiency, and applicability to general model structures. Nevertheless, the literature is abundant with alternative techniques, mostly related to special cases of the general linear model structure (4.33):

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \tag{10.62}$$

(or multivariable counterparts), and to the general nonlinear black-box structure (10.61). A basic idea is to rephrase the problem as a linear regression problem or a sequence of such problems, so that the efficient methods of Section 10.1 can be applied. For (10.61) it may involve fixing the parameters that enter nonlinearly (i.e. β and γ) and estimate the α:s as a linear regression.

The algorithms typically involve two or several LS stages (or IV stages) applied to different substructures, and we therefore call them *two-stage* or *multi-stage* methods.

In this section we shall give a short description of the building blocks of such procedures. Mixing techniques (IV, LS, PEM, PLR) and models (FIR, ARX, ARMAX, etc.) into procedures involving several stages leads to a myriad of "identification methods." There will be no need to list all these. They can, however, be understood and analyzed by our techniques, applied to the different stages (see Problems 10G.2

and 10E.1). Our interest in this topic is twofold: it helps us to understand the identification literature, and the techniques may be useful for providing initial estimates for the basic schemes of Section 10.2.

The subspace method (7.66) can also be regarded as a two-stage method, being built up from two LS-steps. Due to the rather complex nature of this algorithm, we will treat it separately in Section 10.6. For the rich possibilities offered for nonlinear black-box models, we refer to Sjöberg et.al. (1995).

Bootstrap Methods

Consider the correlation formulation (7.110): Solve

$$f_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\zeta(t,\theta)\varepsilon(t,\theta) = 0 \tag{10.63a}$$

in the special case where the prediction error can be written

$$\varepsilon(t,\theta) = y(t) - \varphi^T(t,\theta)\theta \tag{10.63b}$$

This formulation contains a number of common situations:

- IV methods with $\zeta(t,\theta)$ as in (7.127) and $\varphi(t,\theta) = \varphi(t)$ as in (7.114).
- PLR methods with $\zeta(t,\theta) = \varphi(t,\theta)$ as in (7.113).
- Minimizing the quadratic criterion (10.38) for models that can be written as (7.112), taking $\zeta(t,\theta) = \psi(t,\theta)$.

With a nominal iterate $\hat{\theta}_N^{(i-1)}$ at hand, it is then natural to determine the next one by solving

$$\frac{1}{N}\sum_{t=1}^{N}\zeta(t,\hat{\theta}_N^{(i-1)})\left[y(t) - \varphi^T(t,\hat{\theta}_N^{(i-1)})\theta\right] = 0$$

for θ. This is a linear problem and can be solved as

$$\hat{\theta}_N^{(i)} = \left[\frac{1}{N}\sum_{t=1}^{N}\zeta(t,\hat{\theta}_N^{(i-1)})\varphi^T(t,\hat{\theta}_N^{(i-1)})\right]^{-1}\left[\frac{1}{N}\sum_{t=1}^{N}\zeta(t,\hat{\theta}_N^{(i-1)})y(t)\right] \tag{10.64}$$

Solving (10.64) is essentially a least-squares problem (10.2) with proper definitions of $R(N)$ and $f(N)$. The techniques described in Section 10.1 thus apply also to (10.64).

The algorithm (10.64) is known as a bootstrap method, since it alternates between computing θ and forming new vectors φ and ζ. It should be noted that it does not necessarily converge to a solution of (10.63). A convergence analysis is given by Stoica and Söderström (1981b), and Stoica et.al. (1985).

Bilinear Parametrizations

For some model structures, the predictor is bilinear in the parameters. That is, the parameter vector θ can be split up into two parts,

$$\theta = \begin{bmatrix} \rho \\ \eta \end{bmatrix}$$

such that

$$\hat{y}(t|\theta) = \hat{y}(t|\rho, \eta) \tag{10.65}$$

is linear in ρ for fixed η and linear in η for fixed ρ. A typical such situation is the ARARX structure (4.22):

$$\hat{y}(t|\theta) = B(q)D(q)u(t) + [1 - A(q)D(q)]y(t) \tag{10.66}$$

Clearly, by associating ρ with the A- and B-parameters and η with the D-parameters the preceding bilinear situation is at hand.

With this situation, a natural way of minimizing

$$V_N(\theta, Z^N) = V_N(\rho, \eta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\left(y(t) - \hat{y}(t|\rho, \eta)\right)^2 \tag{10.67}$$

would be to treat it as a sequence of least-squares problems. Let

$$\hat{\rho}_N^{(i)} = \arg\min_{\rho} V_N(\rho, \hat{\eta}_N^{(i-1)}, Z^N) \tag{10.68a}$$

$$\hat{\eta}_N^{(i)} = \arg\min_{\eta} V_N(\hat{\rho}_N^{(i)}, \eta, Z^N) \tag{10.68b}$$

Each of these problems is a pure least-squares problem and can be solved efficiently. Although this procedure bears some resemblance to the bootstrap methods, it is indeed a minimization method that will lead to a local minimum (cf. Problems 10T.3 and 10E.9).

Separable Least Squares

A more general situation than the bilinear case is when one set of parameters enter linearly and another set nonlinearly in the predictor:

$$\hat{y}(t|\theta, \eta) = \theta^T\varphi(t, \eta) \tag{10.69}$$

The identification criterion then becomes

$$V_N(\theta, \eta, Z^N) = \sum_{t=1}^{N}\left|y(t) - \theta^T\varphi(t, \eta)\right|^2 = |\mathbf{Y} - \Phi(\eta)\theta|^2 \tag{10.70}$$

where we introduced matrix notation, analogously to (10.7). For given η this criterion is an LS criterion and minimized w.r.t. θ by

$$\hat{\theta} = \left[\Phi^T(\eta)\Phi(\eta)\right]^{-1}\Phi^T(\eta)\mathbf{Y} \tag{10.71}$$

We can thus insert this into (10.70) and define the problem as

$$\min_{\eta} |\mathbf{Y} - P(\eta)\mathbf{Y}|^2 = |(I - P(\eta))\,\mathbf{Y}|^2,$$

$$P(\eta) = \Phi(\eta)\left[\Phi^T(\eta)\Phi(\eta)\right]^{-1}\Phi^T(\eta) \tag{10.72}$$

The estimate θ is then obtained by inserting the minimizing η into (10.71). Note that the matrix P is a projection matrix: $P^2 = P$. The method is called *separable least squares* since the LS-part has been separated out, and the problem reduced to a minimization problem of lower dimension. See Golub and Pereyra (1973)for a thorough treatment of this approach. It is known to give numerically well-conditioned calculations, but does not necessary give faster convergence than applying a damped Gauss-Newton method to (10.70) without utilizing the particular structure.

High-Order AR(X) Models

Suppose the true system is given as

$$y(t) = G_0(q)u(t) + H_0(q)e_0(t)$$

and an ARX structure

$$A^M(q)y(t) = B^M(q)u(t) + e(t)$$

of order M is used. Then it can be shown (e.g., Hannan and Kavalieris, 1984, and Ljung and Wahlberg, 1992) that as the number of data N tends to infinity, as well as the model M (N "faster than" M), the model $\hat{A}_N^M$, $\hat{B}_N^M$ will converge to the true system in the following sense:

$$\frac{\hat{B}_N^M(e^{i\omega})}{\hat{A}_N^M(e^{i\omega})} \to G_0(e^{i\omega}), \qquad \text{uniformly in } \omega \text{ as } N \gg M \to \infty$$

$$\frac{1}{\hat{A}_N^M(e^{i\omega})} \to H_0(e^{i\omega}), \qquad \text{uniformly in } \omega \text{ as } N \gg M \to \infty$$

This means that a high-order ARX model is capable of approximating any linear system arbitrarily well. It is of course desirable to reduce this high-order model to more tractable versions within the structure (10.62), and for that purpose a number of different possibilities are at hand:

1. Find $G = B/A$ as a rational structure by eliminating common factors in $\hat{A}_N^M$ and $\hat{B}_N^M$ (Söderström, 1975b).
2. Apply model reduction techniques based on balanced realizations to $\hat{B}_N^M/\hat{A}_N^M$ (Wahlberg, 1986, Zhu and Backx, 1993).
3. Let $z(t)$ be the output of the model $\hat{B}_N^M/\hat{A}_N^M$ driven by the actual input u and apply an ARX model to the input-output pair (z, u) (Pandaya, 1974; Hsia, 1977, Chapter 7).

4. Let $\hat{e}_M(t)$ be the residuals associated with the model $\hat{A}_N^M$, $\hat{B}_N^M$. Use a model structure

$$A(q)y(t) = B(q)u(t) + [C(q) - 1]\hat{e}_M(t) + e(t) \tag{10.73}$$

to estimate A, B, and C. Since $\hat{e}_M(t)$ is a known sequence, this structure is an ARX structure with two inputs, and the estimates are thus determined by the LS method (Mayne and Firoozan, 1982).

5. The subspace method (7.66) (which is described in detail in Section 10.6) should rightly be included in this family too: The k-step ahead predictors computed from (7.62) are the high order ARX-models, while (7.60) corresponds to the model reduction step, and (7.56) is the second LS-stage.

Separating Dynamics and Noise Models

In the general linear model (10.62), we can always determine the dynamic part from u to y using the IV method. Splitting the denominator estimate thus obtained into one factor $\hat{A}(q)$ that is supposed to be in common with the noise description and one factor $\hat{F}(q)$ that is particular to the dynamics (typically one would postulate one of A and F to be unity), we can then determine

$$\hat{v}(t) = \hat{A}(q)y(t) - \frac{\hat{B}(q)}{\hat{F}(q)}u(t) \tag{10.74}$$

as an estimate of the equation noise [cf. (4.38)]. This noise can then be regarded as a measured signal, and an ARMA model

$$\hat{v}(t) = \frac{C(q)}{D(q)}e(t) \tag{10.75}$$

can be constructed as a separate step. Young has developed this technique in a number of papers (see, e.g., Young and Jakeman, 1979).

Determining ARMA Models

The parameters of the ARMA model (10.75) can of course be estimated using the prediction-error approach. Two alternatives that avoid iterative search procedures are as follows:

1. Apply a high-order AR model to $\hat{v}(t)$ in (10.75) to form estimates of the innovations $\hat{e}$. Then form the ARX model

$$D(q)\hat{v}(t) = [C(q) - 1]\hat{e}(t) + e(t) \tag{10.76}$$

with $\hat{v}(t)$ as output and $\hat{e}(t)$ as input, and estimate D and C with the LS method [cf. (10.73)].

2. Estimate the AR parameters $D(q)$ using the IV method as explained in Problem 7E.1. Then model $\hat{w}(t) = D(q)\hat{v}(t)$ as an MA model. See Durbin (1959)and Walker (1961)for related techniques, and Broersen (1997)for a discussion of order selection.

10.5 LOCAL SOLUTIONS AND INITIAL VALUES

Local Minima

The general numerical schemes for minimization and equation solution that we discussed in Section 10.2 typically have the property that, with suitably chosen step length μ, they will converge to a solution of the posed problem. This means that (10.48) and (10.49) will converge to a point θ_N^* such that

$$f_N(\theta_N^*, Z^N) = 0 \tag{10.77}$$

while (10.40), with positive definite R, converges to a *local minimum* of $V_N(\theta, Z^N)$.

For the minimization problem, it is the *global minimum* that interests us. The theoretical results of Chapters 8 and 9 dealt with properties of the globally minimizing estimate $\hat{\theta}_N$. Similarly, the equation (10.77) may have several solutions. It is obviously an inherent feature of the iterative search routines of Section 10.2 that only convergence to a *local solution* of the problem can be guaranteed. To find the global solution, there is usually no other way than to start the iterative minimization routine at different feasible initial values and compare the results. An important possibility is to use some preliminary estimation procedure to produce a good initial value for the minimization. See the following discussion.

When validating the model as we shall discuss in Sections 16.5 and 16.6, the model is judged according to its performance, though. Therefore, local minima do not necessarily create problems in practice. If a model passes the validation tests, it should be an acceptable model, even if it does not give the global minimum of the criterion function.

The problem of "false" local solutions has two aspects. Let us concentrate on the problem of local minima. It may be that the limit of the criterion function as N tends to infinity, $\overline{V}(\theta)$, has such local minima. Then also $V_N(\theta, Z^N)$ will have such minima for large N, according to Lemma 8.2. The existence of local minima of $\overline{V}(\theta)$ can be analyzed, but only few results are available as yet. Some of them will be given later. The other aspect is that, even if $\overline{V}(\theta)$ has only one local minimum (= the global one), the function $V_N(\theta, Z^N)$ may have other local minima due to the randomness in data. This is a much harder problem to treat analytically. The one exception is the linear regression least-squares method, where by construction the criterion function has no nonglobal local minima regardless of the properties of the data.

Results for SISO Black-box Models

The only analytical results available on local solutions are for black-box models *under the assumption that the system can be described within the model set:* $S \in \mathcal{M}$. We list these results here with references for proofs. They all concern the general SISO model set (10.62), and refer to

$$\overline{V}(\theta) = \overline{E}\tfrac{1}{2}\varepsilon^2(t, \theta)$$

For simplicity we call a nonglobal, local minimum a "false minimum."

- For ARMA models ($B \equiv 0, D \equiv F \equiv 1$) all stationary points of $\overline{V}(\theta)$ are global minima (Åström and Söderström, 1974).
- For ARARX models ($C \equiv F \equiv 1$) there are no false local minima if the signal-to-noise ratio is large enough. If it is very small, false local minima do exist (Söderström, 1974).
- If $A \equiv 1$, there are no false local minima if $n_f = 1$ (Söderström, 1975c).
- If $A \equiv C \equiv D \equiv 1$, there are no false local minima if the input is white noise. For other inputs, however, false local minima can exist (Söderström, 1975c).

For the ARMAX model ($F \equiv D \equiv 1$), it is not known whether false local minima exist. For the pseudolinear regression approach (7.113), it can, however, be shown that

$$\overline{E}\varphi(t, \theta)\varepsilon(t, \theta) = 0 \Rightarrow \theta = \theta_0 \tag{10.78}$$

in the case of an ARMAX model, with θ_0 denoting the true parameters (Ljung, Söderström, and Gustavsson, 1975).

The practical experience with different model structures is that the global minimum is usually found without too much problem for ARMAX models. See, for example, Bohlin (1971)for a discussion of these points. For output error structures, on the other hand, convergence to false local minima is not uncommon.

Initial Parameter Values

Due to the possible occurrence of undesired local minima in the criterion function, it is worthwhile to spend some effort on producing good initial values for the iterative search procedures. Also, since the Newton-type methods described in Section 10.2 have good local convergence rates, but not necessarily fast convergence far from the minimum, these efforts usually pay off in fewer iterations and shorter total computing time.

For a *physically parametrized model structure*, it is most natural to use our physical insight to provide reasonable initial values. Also, it allows us to monitor and interact with the iterative search scheme.

For a *linear black-box model structure* several possibilities exist. It is our experience that the following is a good start-up procedure for the general model structure (10.62):

1. Apply the IV method to estimate the dynamic transfer function B/AF. Most often one of A and F is unity. For a system that has operated in open loop, first a LS estimate of an ARX model can be determined, to be used in the generation of instruments as in (7.123). (10.79a)
2. Determine an estimate of the equation noise as in (10.74). (10.79b)

3. Determine C and/or D in (10.75) by (10.76) after a first high-order AR step to find $\hat{e}$ (which is unnecessary if $C = 1$). The order of the AR model can be chosen as the sum of all model orders in (10.62) so as to balance the computational effort. (10.79c)

In case $S \in \mathcal{M}$ this will bring the initial parameter estimate arbitrarily close to the true values as N increases. From there, the methods of Section 10.2 will efficiently bring us to the global minimum of the criterion. In this case we thus have a procedure that is globally convergent to the global minimum for large enough N.

For a *nonlinear black-box structure* (10.61) several techniques exist. A simple one is to "seed" a large number of fixed values of the non-linear parameters β_k and γ_k, and estimate the corresponding α:s by linear least squares. The estimates α_k that are most significant (relative to their estimated standard deviations) are then selected and the corresponding γ_k and β_k are used as initial values for the ensuing Gauss-Newton iterative search.

Initial Filter Conditions

The filters (10.53)–(10.54) as well as (10.56) require initial values $\varphi(0, \theta)$ to be initialized. In case the filters have finite impulse response, which happens only for the ARX special case, we can wait to initialize the filters until enough past data are known. This is what we called approach 1 following (10.13) in Section 10.1. In the general case we need a strategy to deal with the unknown initial conditions. This has not been extensively discussed in the literature, but we can point to the following approaches:

1. Take $\varphi(0, \theta) = 0$.
2. Select $\varphi(0, \theta)$ so that the first $\hat{y}(t|\theta)$, $t = 1, \ldots, \dim\varphi$ match $y(t)$ exactly.
3. Introduce $\varphi(0, \theta) = \eta$ as a parameter, and estimate it along with θ.
4. Estimate or "backforecast" $\varphi(0, \theta)$ from the data by running suitable filters backwards in time, Knudsen (1994).

For a model where the predictor filter transient is short compared to the data record, it does not matter so much which approach is taken. However, with slowly decaying transients, the two first methods may have a very negative influence on the model quality. This is particularly pronounced for OE models, where no noise model will pick up the residuals from bad transient behaviour.

10.6 SUBSPACE METHODS FOR ESTIMATING STATE SPACE MODELS

Let us now consider how to estimate the system matrices A, B, C, and D in a state space model

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + w(t) \\ y(t) &= Cx(t) + Du(t) + v(t) \end{aligned} \tag{10.80}$$

We assume that the output $y(t)$ is a p-dimensional column vector, while the input $u(t)$ is an m-dimensional column vector. The order of the system, i.e., the dimension of $x(t)$, is n. We also assume that this state-space representation is a minimal realization. It is well known that the same input-output relationship can also be described by

$$\begin{aligned} \tilde{x}(t+1) &= T^{-1}AT\tilde{x}(t) + T^{-1}Bu(t) + \tilde{w}(t) \\ y(t) &= CT\tilde{x}(t) + Du(t) + v(t) \end{aligned} \tag{10.81}$$

for any invertible matrix T. This corresponds to the change of basis $\tilde{x}(t) = T^{-1}x(t)$ in the state space.

In Section 7.3, algorithm (7.66), we presented an archetypical algorithm for this. We shall here describe a family of related algorithms which all address this problem. The discussion will be quite technical and a reader who primarily is interested in the result could go directly to (10.125). In summary the algorithms are based on the following observations:

- If $\hat{A}$ and $\hat{C}$ are known, it is an easy linear least squares problem to estimate B and D from

$$y(t) = \hat{C}(qI - \hat{A})^{-1}Bu(t) + Du(t) + v(t) \tag{10.82}$$

  using the predictor

$$\hat{y}(t|B, D) = \hat{C}(qI - \hat{A})^{-1}Bu(t) + Du(t) \tag{10.83}$$

  (The initial state $x(0)$ can also be estimated; see (10.86) below.)

- If the (extended) observability matrix for the system

$$O_r = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{r-1} \end{bmatrix} \tag{10.84}$$

  is known, then it is easy to determine C and A. Use the first block row of O_r and the shift property, respectively. This is really the key step.

- The extended observability matrix can be consistently estimated from input-output data by direct least-squares like (projection) steps.

- Once the observability matrix has been estimated, the states $x(t)$ can be constructed and the statistical properties of the noise contributions $w(t)$ and $v(t)$ can be established.

We shall now deal with each of these steps in somewhat more detail.

Estimating B and D

For given and fixed $\hat{A}$ and $\hat{C}$ the model structure (10.83):

$$\hat{y}(t|B, D) = \hat{C}(qI - \hat{A})^{-1}Bu(t) + Du(t) \tag{10.85a}$$

is clearly linear in B and D. The predictor is also formed entirely from past inputs, so it is an *output error* model structure. If the system operates in open loop, we can thus consistently estimate B and D according to Theorem 8.4, even if the noise sequence

$$v(t) = C(qI - A)^{-1}w(t) + v(t)$$

in (10.82) is non-white.

Let us write the predictor (10.83) in the standard linear regression form

$$\hat{y}(t) = \varphi(t)\theta = \varphi(t)\begin{bmatrix} \text{Vec}(B) \\ \text{Vec}(D) \end{bmatrix} \tag{10.85b}$$

with a $p \times (mn + mp)$ matrix $\varphi(t)$. Here "Vec" is the operation that builds a vector from a matrix, by stacking its columns on top of each other. Let $r = (k-1)n + j$. To find the r:th ($r \leq mn$) column of $\varphi(t)$, which corresponds to the r:th element of θ, i.e., the element $B_{j,k}$, we differentiate (10.85b) w.r.t. this element and obtain

$$\varphi_r(t) = \hat{C}(qI - \hat{A})^{-1}E_j u_k(t)$$

where E_j is the column vector with the j:th element equal to 1 and the others equal to 0. The rows for $r > nm$ are handled in a similar way.

If desired, also the initial state $x_0 = x(0)$ can be estimated in an analogous way, since the predictor with initial values taken into account is

$$\hat{y}(t|B, D, x_0) = \hat{C}(qI - \hat{A})^{-1}x_0\delta(t) + \hat{C}(qI - \hat{A})^{-1}Bu(t) + Du(t) \tag{10.86}$$

which is linear also in x_0. Here $\delta(t)$ is the unit pulse at time 0. Moreover, the estimates can be improved by estimating the color of v in (10.82) and prefiltering the data accordingly.

Remark: If $\hat{A}$ and $\hat{C}$ are the correct values, the least squares estimates of B and D will also converge to their true values, according to Theorem 8.4. If consistent estimates $\hat{A}_N$ and $\hat{C}_N$ are used instead, convergence of $\hat{B}_N$ and $\hat{D}_N$ to their true values still holds. This follows by fairly straightforward calculations. See Vandersteen, Van hamme, and Pintelon (1996) for a general treatment of such issues.

Finding A and C from the Extended Observability Matrix

Suppose that a $pr \times n^*$ dimensional matrix G is given, that is related to the extended observability matrix of the system, (10.84). We have to determine A and C from G, and we shall here consider cases of increased complexity.

Known System Order. Suppose first we know that

$$G = O_r$$

so that $n^* = n$. To find C is then immediate:

$$\hat{C} = O_r(1 : p, 1 : n) \tag{10.87}$$

Here we used MATLAB notation, meaning that $M(s : \ell, j : k)$ is the matrix obtained by extracting the rows $s, s+1, \ldots, \ell$ and the columns $j, j+1, \ldots, k$ from the matrix M.

Similarly we can find $\hat{A}$ from the equation

$$O_r(p+1 : pr, 1 : n) = O_r(1 : p(r-1), 1 : n)\hat{A} \tag{10.88}$$

which is easily seen from the definition (10.84). Under the assumption of observability, O_{r-1} has rank n, so $\hat{A}$ can be determined uniquely. Normally (10.88) is an overdetermined set of equations (n^2 unknowns in A and $npr - np$ equations; recall that $r \geq n+1$). This is of no consequence if O_r is exactly of the form (10.84), since any full rank subset of equations will give the same $\hat{A}$.

Role of State-Space Basis. The extended observability matrix depends on the choice of basis in the state-space representation. For the representation (10.81) it is easy to verify that the observability matrix would be

$$\tilde{O}_r = O_r T \tag{10.89}$$

Applying (10.87) and (10.88) to $\tilde{O}_r$ would thus give the system matrices associated with (10.81). Consequently, *multiplying the extended observability matrix from the right by any invertible matrix before applying (10.87) and (10.88) will not change the system estimate—just the basis of representation.*

Unknown System Order. Suppose now that the true order of the system is unknown, and that n^*—the number of columns of G—is just an upper bound for the order. This means that we have

$$G = O_r \tilde{T} \tag{10.90}$$

for some unknown, but full rank, $n \times n^*$ matrix $\tilde{T}$, where also n is unknown to us. The rank of G is n. A straightforward way to deal with this would be to determine this rank, delete the last $n^* - n$ columns of G and then proceed as above. A more general and numerically sound way of reducing the column space is to use singular value decomposition (SVD):

$$G = USV^T = U \begin{bmatrix} \sigma_1 & 0 & 0 & \ldots & 0 \\ 0 & \sigma_2 & 0 & \ldots & 0 \\ 0 & 0 & \sigma_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \sigma_{n^*} \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix} V^T \tag{10.91}$$

Here U and V are orthonormal matrices ($U^TU = I$, $V^TV = I$) of dimensions $pr \times pr$ and $n^* \times n^*$, respectively. S is a $pr \times n^*$ matrix with the singular values of G along the diagonal and zeros elsewhere. If G has rank n, only the first n singular values σ_k will be non-zero. This means that we can rewrite

$$G = USV^T = U_1 S_1 V_1^T \tag{10.92}$$

where U_1 is a $pr \times n$ matrix containing the first n columns of U, while S_1 is the $n \times n$ upper left part of S, and V_1 consists of the first n columns of V. (We still have $V_1^T V_1 = I$.) From (10.90) we find $O_r \tilde{T} = U_1 S_1 V_1^T$. Multiplying this by V_1 from the right gives

$$O_r \tilde{T} V_1 = O_r T = U_1 S_1 \tag{10.93}$$

for some invertible matrix $T = \tilde{T} V_1$. We are now in the situation (10.89) that we know the observability matrix up to an invertible matrix T—or equivalently, we know the observability matrix in some state-space basis. Consequently we can use $\hat{O}_r = U_1 S_1$ or $\hat{O}_r = U_1$ or any matrix that can be written as

$$\hat{O}_r = U_1 R \tag{10.94}$$

for some invertible R in (10.87) and (10.88) to determine the $p \times n$ matrix $\hat{C}$ and the $n \times n$ matrix $\hat{A}$.

Using a Noisy Estimate of the Extended Observability Matrix. Let us now assume that the given $pr \times n^*$ matrix G is a noisy estimate of the true observability matrix

$$G = O_r \tilde{T} + E_N \tag{10.95}$$

where E_N is small and tends to zero as $N \to \infty$. The rank of O_r is not known, while the "noise matrix" E_N is likely to be of full rank. It is reasonable to proceed as above and perform an SVD on G:

$$G = USV^T \tag{10.96}$$

Due to the noise, S will typically have all singular values $\sigma_k; k = 1, \ldots, \min(n^*, pr)$ non-zero. The first n will be supported by O_r, while the remaining ones will stem from E_N. If the noise is small, one should expect that the latter are significantly smaller than the former. Therefore determine $\hat{n}$ as the number of singular values that are significantly larger than 0. Then keep those and replace the others in S by zeros, and proceed as in (10.92) to determine U_1 and S_1. Then use $\hat{O}_r$ in (10.94) to determine $\hat{A}$ and $\hat{C}$ as before. However, in this noisy case, $\hat{O}_r$ will not be exactly subject to the shift structure (10.88), so this system of equations should be solved in a least-squares sense.

The *consistency* of this process as $N \to \infty$ and $E_N \to 0$ is rather easy to establish by a continuity argument: As E_N tends to zero, the corresponding estimates of A and C will tend to the values that are obtained from (10.90) with $E_N = 0$.

It is more difficult to analyze how the variance of E_N will influence the variance of $\hat{A}$ and $\hat{C}$. Some results about this are given in Viberg et.al. (1993), based on work by T.W. Anderson.

Using Weighting Matrices in the SVD. For more flexibility we could pre- and post-multiply G as $\hat{G} = W_1 G W_2$ before performing the SVD

$$\hat{G} = W_1 G W_2 = USV^T \approx U_1 S_1 V_1^T \tag{10.97}$$

and then instead of (10.94) use

$$\hat{O}_r = W_1^{-1} U_1 R \tag{10.98}$$

to determine $\hat{C}$ and $\hat{A}$ in (10.87) and (10.88). Here R is an arbitrary matrix, that will determine the coordinate basis for the state representation. The post-multiplication by W_2 just corresponds to a change of basis in the state-space and the pre-multiplication by W_1 is eliminated in (10.98), so in the noiseless case $E = 0$, these weightings are without consequence. However, when noise is present, they have an important influence on the space spanned by U_1, and hence on the quality of the estimates $\hat{C}$ and $\hat{A}$. We may remark that post-multiplying W_2 by an orthonormal matrix does not effect the U_1-matrix in the decomposition. See Problem 10E.10. We shall return to these questions below.

Estimating the Extended Observability Matrix

The Basic Expression. From (10.80) we find that

$$\begin{aligned}
y(t+k) &= Cx(t+k) + Du(t+k) + v(t+k) \\
&= CAx(t+k-1) + CBu(t+k-1) + Cw(t+k-1) \\
&\quad + Du(t+k) + v(t+k) \\
&= \ldots \\
&= CA^k x(t) + CA^{k-1} Bu(t) + CA^{k-2} Bu(t+1) + \ldots \\
&\quad + CBu(t+k-1) + Du(t+k) \\
&\quad + CA^{k-1} w(t) + CA^{k-2} w(t+1) + \ldots \\
&\quad + Cw(t+k-1) + v(t+k)
\end{aligned} \tag{10.99}$$

Now, form the vectors

$$Y_r(t) = \begin{bmatrix} y(t) \\ y(t+1) \\ \vdots \\ y(t+r-1) \end{bmatrix}, \qquad U_r(t) = \begin{bmatrix} u(t) \\ u(t+1) \\ \vdots \\ u(t+r-1) \end{bmatrix} \tag{10.100}$$

and collect (10.99) as

$$Y_r(t) = O_r x(t) + S_r U_r(t) + V(t) \tag{10.101}$$

with

$$S_r = \begin{bmatrix} D & 0 & \cdots & 0 & 0 \\ CB & D & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ CA^{r-2}B & CA^{r-3}B & \cdots & CB & D \end{bmatrix}$$

and the k:th block component of $V(t)$

$$\begin{aligned} V_k(t) &= CA^{k-2}w(t) + CA^{k-3}w(t+1) + \dots \\ &\quad + Cw(t+k-2) + v(t+k-1) \end{aligned} \tag{10.102}$$

We shall use (10.101) to estimate O_r, or rather a matrix O_rT for some (unknown) T. The idea is to correlate both sides of (10.101) with quantities that eliminate the term with $U_r(t)$ and make the noise influence from V disappear asymptotically. For this, we have the measurements $y(t)$, $u(t)$, $t = 1, \dots, N + r - 1$ available. It will be easier to describe the correlation operations as matrix multiplications, and we therefore introduce

$$\begin{aligned} \mathbf{Y} &= \begin{bmatrix} Y_r(1) & Y_r(2) & \dots & Y_r(N) \end{bmatrix} \\ \mathbf{X} &= \begin{bmatrix} x(1) & x(2) & \dots & x(N) \end{bmatrix} \\ \mathbf{U} &= \begin{bmatrix} U_r(1) & U_r(2) & \dots & U_r(N) \end{bmatrix} \\ \mathbf{V} &= \begin{bmatrix} V(1) & V(2) & \dots & V(N) \end{bmatrix} \end{aligned} \tag{10.103}$$

These quantities depend on r and N, but this dependence is suppressed. We can now rewrite (10.101) as the basic expression

$$\mathbf{Y} = O_r\mathbf{X} + S_r\mathbf{U} + \mathbf{V} \tag{10.104}$$

Remark. Define as in (7.59) the vector $\hat{\mathbf{Y}}$ of true k-step ahead predictors. Then it follows from (10.104) that

$$\hat{\mathbf{Y}} = O_r\hat{\mathbf{X}} \tag{10.105}$$

where $\hat{\mathbf{X}}$ is made up from the predicted (Kalman-filter) states $\hat{x}(t|t-1)$, i.e. the best estimate of $x(t)$ based on past input-output data.

Removing the U-term. Form the $N \times N$ matrix

$$\Pi^{\perp}_{\mathbf{U}^T} = \mathbf{I} - \mathbf{U}^T(\mathbf{U}\mathbf{U}^T)^{-1}\mathbf{U} \tag{10.106}$$

(if $\mathbf{U}\mathbf{U}^T$ is singular, use pseudoinverse instead). This matrix performs projection orthogonal to the matrix $\mathbf{U}$, i.e.,

$$\mathbf{U}\Pi^{\perp}_{\mathbf{U}^T} = \mathbf{U} - \mathbf{U}\mathbf{U}^T(\mathbf{U}\mathbf{U}^T)^{-1}\mathbf{U} = \mathbf{0}$$

Multiplying (10.104) from the right by $\Pi^{\perp}_{\mathbf{U}^T}$ will thus eliminate the term with $\mathbf{U}$:

$$\mathbf{Y}\Pi^{\perp}_{\mathbf{U}^T} = O_r\mathbf{X}\Pi^{\perp}_{\mathbf{U}^T} + \mathbf{V}\Pi^{\perp}_{\mathbf{U}^T} \tag{10.107}$$

Removing the Noise Term. The next problem is to eliminate the last term. Since this term is made up of noise contributions, the idea is to correlate it away with a suitable matrix. Define the $s \times N$ matrix ($s \geq n$)

$$\Phi = [\varphi_s(1) \quad \varphi_s(2) \quad \dots \quad \varphi_s(N)] \tag{10.108}$$

where $\varphi_s(t)$ is a yet undefined vector. Multiply (10.107) from the right by Φ^T and normalize by N:

$$G = \frac{1}{N}\mathbf{Y}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T = O_r\frac{1}{N}\mathbf{X}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T + \frac{1}{N}\mathbf{V}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T \stackrel{\text{def}}{=} O_r\tilde{T}_N + V_N \tag{10.109}$$

Here $\tilde{T}_N$ is an $n \times s$ matrix. Suppose now that we can find $\varphi_s(t)$ so that

$$\lim_{N\to\infty} V_N = \lim_{N\to\infty} \frac{1}{N}\mathbf{V}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T = 0 \tag{10.110a}$$

$$\lim_{N\to\infty} \tilde{T}_N = \lim_{N\to\infty} \frac{1}{N}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T = \tilde{T} \quad \text{has full rank } n \tag{10.110b}$$

Then (10.109) would read

$$\begin{aligned} G &= \frac{1}{N}\mathbf{Y}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T = O_r\tilde{T} + E_N \\ E_N &= O_r(\tilde{T}_N - \tilde{T}) + V_N \to 0 \text{ as } N \to \infty \end{aligned} \tag{10.111}$$

The $pr \times s$ matrix G can thus be seen as a noisy estimate (10.95) and we can subject it to the treatment (10.96)–(10.98) to obtain estimates of A and C.

Finding Good Instruments. The only remaining question is how to achieve (10.110). Notice that these requirements are just like (7.119) and (7.120) for the *instrumental variable method*. Using the expression (10.106) for $\Pi_{\mathbf{U}^T}^{\perp}$ and writing out the matrix multiplications as sums gives

$$\begin{aligned} \frac{1}{N}\mathbf{V}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T = \frac{1}{N}\sum_{t=1}^{N} V(t)\varphi_s^T(t) - \frac{1}{N}\sum_{t=1}^{N} V(t)U_r^T(t) \\ \times \left[\frac{1}{N}\sum_{t=1}^{N} U_r(t)U_r^T(t)\right]^{-1} \frac{1}{N}\sum_{t=1}^{N} U_r(t)\varphi_s^T(t) \end{aligned} \tag{10.112}$$

Under mild conditions, the law of large numbers states that the sample sums converge to their respective expected values

$$\lim_{N\to\infty} \frac{1}{N}\mathbf{V}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T = \bar{E}V(t)\varphi_s^T(t) - \bar{E}V(t)U_r^T(t)R_u^{-1}\bar{E}U_r(t)\varphi_s^T(t) \tag{10.113}$$

where

$$R_u = \bar{E}U_r(t)U_r^T(t)$$

is the $r \times r$ covariance matrix of the input. Now, assume that the input u is generated in open loop, so that it is independent of the noise terms in V (see (10.102)). Then $EV(t)U_r^T(t) = 0$. Assume also that R_u is invertible. (This means that the input is *persistently exciting* of order r—see Section 13.2.) Then the second term of (10.113) will be zero. (If the pseudo-inverse is used in (10.106), this is still true, even if R_u is not invertible.) For the first term to be zero, we must require $V(t)$ and $\varphi_s(t)$ to be uncorrelated. Since $V(t)$ according to (10.102) is made up of white noise terms from time t and onwards, any choice $\varphi_s(t)$ built up from data prior to time t will satisfy (10.110a). A typical choice would be

$$\varphi_s(t) = \begin{bmatrix} y(t-1) \\ \vdots \\ y(t-s_1) \\ u(t-1) \\ \vdots \\ u(t-s_2) \end{bmatrix} \tag{10.114}$$

Now, turning to (10.110b) we find by a similar argument that

$$\tilde{T} = \overline{E}x(t)\varphi_s^T(t) - \overline{E}x(t)U_r^T(t)R_u^{-1}\overline{E}U_r(t)\varphi_s^T(t) \tag{10.115}$$

A formal proof that $\tilde{T}$ has full rank is not immediate and will involve properties of the input. See Problem 10G.6 and Van Overschee and DeMoor (1996).

Summing up, forming $G = \frac{1}{N}\mathbf{Y}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T$ with Φ defined by (10.114) and (10.108) gives the properties (10.111), which allows us to consistently determine A and C, via (10.97), (10.98), and (10.88), (10.87). We shall sum up all the steps in the algorithm later.

Finding the States and Estimating the Noise Statistics

In (10.104) we constructed a direct relationship between future outputs and the states. Let us now shift the perspective somewhat and return to the prediction approach of Section 7.3. This will give an expression which is closely related to (10.104) and shows the links between states and predictors.

Let us estimate the k-step ahead predictors. Recall (7.63).

$$Y_r(t) = \Theta\varphi_s(t) + \Gamma U_r(t) + E(t) \tag{10.116}$$

Collecting all t as in (10.104) gives

$$\mathbf{Y} = \Theta\Phi + \Gamma\mathbf{U} + \mathbf{E} \tag{10.117}$$

The Least Squares estimate of the parameters is

$$[\,\hat{\Theta} \quad \hat{\Gamma}\,] = [\,\mathbf{Y}\Phi^T \quad \mathbf{Y}\mathbf{U}^T\,]\begin{bmatrix} \Phi\Phi^T & \Phi\mathbf{U}^T \\ \mathbf{U}\Phi^T & \mathbf{U}\mathbf{U}^T \end{bmatrix}^{-1} \tag{10.118}$$

Using the expression for inverting block matrices gives (see Problem 10E.11)

$$\hat{\Theta} = \mathbf{Y}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T\left(\Phi\Pi_{\mathbf{U}^T}^{\perp}\Phi^T\right)^{-1} \tag{10.119}$$

This means that the matrix of predicted outputs can be written

$$\begin{aligned}\hat{\mathbf{Y}} &= \begin{bmatrix}\hat{Y}_r(1) & \dots & \hat{Y}_r(N)\end{bmatrix} \\ &= \mathbf{Y}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T(\Phi\Pi_{\mathbf{U}^T}^{\perp}\Phi^T)^{-1}\Phi\end{aligned} \tag{10.120}$$

which we recognize as $\hat{G}$ in (10.97), with G as in (10.109) and the weightings $W_1 = I$ and $W_2 = (\Phi\Pi_{\mathbf{U}^T}^{\perp}\Phi^T)^{-1}\Phi$. Performing the SVD and deleting small singular values thus gives

$$\hat{\mathbf{Y}} \approx U_1 S_1 V_1^T \tag{10.121}$$

Here V_1^T is an $n \times N$ matrix. We know from (10.98) that U_1 is related to the observability matrix by some invertible matrix R as $U_1 = O_r R^{-1}$. Introduce $\hat{\mathbf{X}} = R^{-1}S_1V_1$. Then (10.121) can be written as

$$\hat{\mathbf{Y}} \approx O_r R^{-1} S_1 V_1^T = O_r\hat{\mathbf{X}} \tag{10.122}$$

Comparing with (10.105) shows that $\hat{\mathbf{X}}$ must be the matrix of the correct state estimates if $\hat{\mathbf{Y}}$ is the matrix of true predicted outputs. The true predicted outputs are, however, normally obtained only when the orders s_i in (10.114) tend to infinity. In such a case we have then found the true state estimates—in the state-space basis in question—from the SVD. Alternatively we can write

$$\begin{aligned}\hat{\mathbf{X}} &= L\hat{\mathbf{Y}} = \begin{bmatrix}\hat{x}(1) & \dots & \hat{x}(N)\end{bmatrix} \\ L &= R^{-1}U_1^T\end{aligned} \tag{10.123}$$

since $U_1^T U_1 = I$ from the SVD-properties. With the states given, we can estimate the process and measurement noises as

$$\begin{aligned}w(t) &= \hat{x}(t+1) - \hat{A}\hat{x}(t) - \hat{B}u(t) \\ v(t) &= y(t) - \hat{C}\hat{x}(t) - \hat{D}u(t)\end{aligned} \tag{10.124}$$

and estimate their covariance matrices in a straightforward fashion. Here $\hat{A}, \hat{B}, \hat{C}, \hat{D}$ are the estimates of the system matrices obtained as described above. Alternatively, these could be directly estimated by the least squares procedure (7.66), once the states are known.

Putting It All Together

We have now described all the basic steps of the algorithm, although in somewhat reverse order. The complete subspace algorithm can be summarized as follows:

The Family of Subspace Algorithms (10.125)

1. From the input-output data, form

$$G = \frac{1}{N}\mathbf{Y}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T \tag{10.126}$$

with the involved matrices defined by (10.103), (10.100), (10.106), (10.114), and (10.108).

2. Select weighting matrices W_1 ($rp \times rp$ and invertible) and W_2 ($(ps_1+ms_2) \times \alpha$) and perform SVD

$$\hat{G} = W_1 G W_2 = USV^T \approx U_1 S_1 V_1^T \tag{10.127}$$

where the last approximation is obtained by keeping the n most significant values of the singular values in S and setting the remaining ones to zero. (U_1 is now $rp \times n$. S_1 is $n \times n$ and V_1^T is $n \times \alpha$.) As remarked above, post-multiplying W_2 by any $\alpha \times k$ orthonormal matrix (with $k \geq rp$) will not change U_1. (See Problem 10E.10.)

3. Select a full rank matrix R and define the $rp \times n$ matrix $\hat{O}_r = W_1^{-1}U_1 R$. Solve

$$\hat{C} = \hat{O}_r(1 : p, 1 : n) \tag{10.128a}$$

$$\hat{O}_r(p+1 : pr, 1 : n) = O_r(1 : p(r-1), 1 : n)\hat{A} \tag{10.128b}$$

for $\hat{C}$ and $\hat{A}$. The latter equation should be solved in a least squares sense.

4. Estimate $\hat{B}$, $\hat{D}$ and $\hat{x}_0$ from the linear regression problem:

$$\arg\min_{B,D,x_0} \frac{1}{N}\sum_{t=1}^{N} \Big\| y(t) - \hat{C}(qI - \hat{A})^{-1}Bu(t) - Du(t) - \hat{C}(qI - \hat{A})^{-1}x_0\delta(t) \Big\|^2 \tag{10.129}$$

5. If a noise model is sought, form $\hat{\mathbf{X}}$ as in (10.123) and estimate the noise contributions as in (10.124).

Numerical Implementation. It should be mentioned that the most efficient numerical implementation of the above steps is to apply QR-factorization of the data matrix $[\mathbf{U}^T \quad \Phi^T \quad \mathbf{Y}^T]^T = LQ$. ($L$ is here a lower triangular, $pr+mr+s$ square matrix, while Q is an orthonormal $(pr+mr+s) \times N$ matrix.) Then the crucial SVD-factor U_1 in (10.127) can be found entirely from the "L-part" of this factorization, i.e., the "small" matrix. See Problem 10G.5 as well as Van Overschee and DeMoor (1996) and Viberg, Wahlberg, and Ottersten (1997).

The family of methods contains a number of design variables. Different algorithms described in the literature correspond to different choices of these variables. It is at present not fully understood how to choose them optimally. The choices are:

- The choice of correlation vector $\varphi_s(t)$ in (10.114). The requirement is that (10.110) holds. Notice that this choice also determines the ARX-model that

is used to approximate the system when finding the k-step ahead predictors in (7.62). Most algorithms choose $\varphi_s(t)$ to consist of past inputs and outputs as in (10.114) with $s_1 = s_2$. The scalar s is then the only design variable.

If $s_1 = 0$ only past inputs are used, and an *output-error* variant of the algorithm is obtained. Then the noise is ignored when forming the predictors, which leads to models like (7.55) that do not attempt to describe the noise properties, but therefore also may describe the input-output properties with fewer states. The OE-MOESP algorithm of Verhaegen (1994)uses this choice of regressors.

- The scalar r, which is the maximal prediction horizon used. Many algorithms use $r = s$, but there is no particular reason for such a choice.
- The weighting matrices W_1 and W_2. This is the perhaps most important choice. Existing algorithms employ the following choices:
  - MOESP, Verhaegen (1994): $W_1 = I$, $W_2 = (\frac{1}{N}\Phi\Pi^{\perp}_{\mathbf{U}^T}\Phi^T)^{-1}\Phi\Pi^{\perp}_{\mathbf{U}^T}$
  - N4SID, Van Overschee and DeMoor (1994): $W_1 = I$, $W_2 = (\frac{1}{N}\Phi\Pi^{\perp}_{\mathbf{U}^T}\Phi^T)^{-1}\Phi$ (see also (10.120).)
  - IVM, Viberg (1995): $W_1 = (\frac{1}{N}\mathbf{Y}\Pi^{\perp}_{\mathbf{U}^T}\mathbf{Y})^{-1/2}$, $W_2 = (\frac{1}{N}\Phi\Phi^T)^{-1/2}$
  - CVA, Larimore (1990): $W_1 = (\frac{1}{N}\mathbf{Y}\Pi^{\perp}_{\mathbf{U}^T}\mathbf{Y})^{-1/2}$, $W_2 = (\frac{1}{N}\Phi\Pi^{\perp}_{\mathbf{U}^T}\Phi^T)^{-1/2}$

  (Note that W_2 can be expressed in several ways, since post-multiplying with any orthonormal matrix does not change the resulting estimates.) The effects of the weightings are discussed in several papers. See the bibliography.
- The matrix R in step 3. Typical choices are $R = I$, $R = S_1$ or $R = S_1^{1/2}$.

10.7 SUMMARY

Determining a parameter estimate from data has two aspects. First, one has to decide how to characterize the sought estimate: as the solution of a certain equation or as the minimizing argument of some function. Second, one has to devise a numerical method that calculates that estimate. It is important to keep these issues separate. The combination of several different approaches to characterize the desired estimate with many techniques to actually compute it has lead to a wide, and sometimes confusing, variety of identification methods. Our aim in this chapter, as well as in Chapter 7, has been to point to underlying basic ideas.

For linear regression problems (LS and IV methods), we have recommended QR factorization-type methods (10.11) and also pointed to the possibilities of using Levinson and/or lattice methods [(10.23) and (10.30), (10.32) respectively] for special structures.

For general PEM, we have recommended the damped Gauss-Newton iterative method (10.40), (10.41), and (10.46) as the basic choice, complemented with (10.79) to find initial values for linear black-box models.

For subspace methods the essential parts of the numerical calculations consist of a QR-factorization step and an SVD. This allows very robust numerical methods for the calculation of the estimates.

10.8 BIBLIOGRAPHY

The computation of estimates is of course a topic that is covered in many articles and books on system identification. The basic techniques are also the subject of many studies in numerical analysis.

For the linear least-squares problem of Section 10.1, an excellent overview is given in Lawson and Hanson (1974). An account of the Levinson algorithm and its ramifications is given, for example, in Kailath (1974). An early application of the Levinson algorithm for estimation problems was given by Durbin (1960). The algorithm with estimated $R(\tau)$ is therefore sometimes referred to as the Levinson-Durbin algorithm. The multivariable Levinson algorithm was given in Whittle (1963)and Wiggins and Robinson (1965). A Levinson algorithm for the "covariance method" [for which $R(N)$ is not a Toepliz matrix, but deviates "slightly" from this structure] was derived by Morf et.al. (1977). Levinson algorithm has been widely applied in geophysics (e.g., Robinson, 1967; Burg, 1967) and speech processing (e.g., Markel and Gray, 1976), while it has been less used in control applications. Its numerical properties are investigated in Cybenko (1980).

Lattice filters are used extensively in Markel and Gray (1976), Honig and Messerschmitt (1984), and Rabiner and Schafer (1978), in addition to the references mentioned in the text. The numerical stability of the calculation of reflection coefficients in (10.30), and (10.32) has been analyzed by Cybenko (1984). Considerable attention has been paid to the recursive updating of the reflection coefficients, which we shall return to in Chapter 11.

For the methods of Section 10.2, Dennis and Schnabel (1983)serves as a basic references. It contains many additional references and also pseudocode for typical algorithms. Variants of the Newton methods for system identification applications have been discussed in, for example, Åström and Bohlin (1965), Gupta and Mehra (1974), Kashyap and Nasburg (1974). The gradients $\psi_f = (\partial/\partial\theta)\hat{y}$, $(\partial/\partial\theta)\hat{x}$, and so on, are known as sensitivity functions or sensitivity derivatives. These have been studied also in connection with sensitivity analysis of control design. Simple methods for calculating these gradients in state-space models have been discussed in, for example, Denery (1971)and Neuman and Sood (1972), as well as in Gupta and Mehra (1974) and Hill (1985). The use of Lagrangian multipliers to reduce the computational burden is described in Kashyap (1970)and van Zee and Bosgra (1982). Another possibility is to apply Parseval's relationship to V_N' and H_N in (10.39) and (10.45) and evaluate these in the frequency-domain in terms of the Fourier transforms of the signals. See Hannan (1969)and Akaike (1973). The expressions follow easily from (7.25) and (9.53). A special technique for maximizing the likelihood functions, the EM algorithm, has been developed by Dempster, Laird, and Rubin (1977). See Problem 10G.3.

Further results on the uniqueness of solutions are given in Chapter 12 of Söderström and Stoica (1989).

Analysis of bootstrap methods is carried out in Stoica et.al. (1985). Two- and multistage methods have been discussed in many variants. In addition to those described in Section 10.4, there are, for example, the well-known methods of Durbin (1959)and Walker (1961). Both start with a high-order AR model. The coefficients

of these models (the corresponding covariance functions in the Walker case) lead to a system of equations for solving for MA parameters. See also Anderson (1971), Section 5.7.2. Other techniques to build models by reduction of high order ARX-models are described in Wahlberg (1989)and Zhu and Backx (1993).

The Subspace methods really originate from classical realization theory as formulated in Ho and Kalman (1966)and Kung (1978). These algorithms pointed to the essential relationships (10.88) and (10.87). The extended observability matrix can also be found by factorizing the Hankel matrix of impulse responses, and several identification methods based on this have been devised, like King, Desai, and Skelton (1988), Liu and Skelton (1992). Larimore developed his algorithms in Larimore (1983), Larimore (1990)inspired by Akaike's work on canonical correlation. Akaike (1974b)and Akaike (1976). Related algorithms were developed by Aoki in Aoki (1987)for the time-series case.

The family of methods developed with the related approaches by Moonen et.al. (1989), Verhaegen (1991). lead to the basic presentations Verhaegen (1994)and Van Overschee and DeMoor (1994). The relationships between the approaches have been pointed out by Viberg (1995)and Van Overschee and DeMoor (1996), which can be recommended as general overviews. Statistical analysis is presented in Peternell, Scherrer, and Deistler (1996). Special techniques to handle closed loop data are described in Chou and Verhaegen (1997). The presentation in Section 10.6 is largely based on Viberg, Wahlberg, and Ottersten (1997), with a similar perspective described in Jansson and Wahlberg (1996).

Subspace methods to fit frequency-domain data are treated in McKelvey, Akcay, and Ljung (1996).

10.9 PROBLEMS

10G.1 Let $z(t)$ be a p-dimensional signal, and let $\hat{a}_i^n$ and $\hat{b}_i^n$ ($p \times p$ matrices) be the least-squares estimates of the linear regression models

$$\begin{aligned} \hat{z}^n(t|\theta) &= -a_1^n z(t-1) - \cdots - a_n^n z(t-n) \\ \check{z}^n(t-n-1|\theta) &= -b_1^n z(t-n) - \cdots - b_n^n z(t-1) \end{aligned} \tag{10.130}$$

based on data $z(t)$, $1 \le t \le N$. Show, by arguments analogous to (10.15) to (10.24), that these estimates can be computed as

$$\hat{a}_k^{n+1} = \hat{a}_k^n + \rho_n^a \hat{b}_{n-k+1}^n, \quad \hat{a}_{n+1}^{n+1} = \rho_n^a$$

$$\hat{b}_k^{n+1} = \hat{b}_k^n + \rho_n^b \hat{a}_{n-k+1}^n, \quad \hat{b}_{n+1}^{n+1} = \rho_n^b$$

$$\alpha_n = R_{n+1} + \sum_{k=1}^{n} \hat{a}_k^n R_{n+1-k}, \quad \beta_n = R_{-n-1} + \sum_{k=1}^{n} \hat{b}_k^n R_{-n-1-k}. \ (= \alpha_n^T)$$

$$V_{n+1}^a = V_n^a - \alpha_n [V_n^b]^{-1} \beta_n, \quad V_{n+1}^b = V_n^b - \beta_n [V_n^a]^{-1} \alpha_n$$

$$\rho_n^a = -\alpha_n [V_n^b]^{-1}, \quad \rho_n^b = -\beta_n [V_n^a]^{-1}$$

Here

$$R_k = \frac{1}{N}\sum_{t=-n}^{N+n} z(t+k)z^T(t)$$

with $z(t) = 0$ outside the interval $1 \le t \le N$. (This is the multivariable Levinson algorithm, as derived by Whittle, 1963.)

10G.2 *Steiglitz-McBride method:* Steiglitz and McBride (1965)have suggested the following approach to identify a linear system subject to white measurement errors: Consider the OE model (4.25)

$$y(t) = \frac{B(q)}{F(q)}u(t) + e(t)$$

Step 1. Apply the LS method to the ARX model

$$F(q)y(t) = B(q)u(t) + e(t)$$

This gives $\hat{B}_N(q)$ and $\hat{F}_N(q)$.

Step 2. Filter the data through the prefilter

$$y_F(t) = \frac{1}{\hat{F}_N(q)}y(t), \qquad u_F(t) = \frac{1}{\hat{F}_N(q)}u(t)$$

Step 3. Apply the LS method to the ARX model

$$F(q)y_F(t) = B(q)u_F(t) + e(t)$$

Repeat from step 2 with the new $\hat{F}$ estimate. Stop when $\hat{F}_N$ and $\hat{B}_N$ have converged.

(a) This method can be interpreted as a way of solving for a correlation estimate as in (7.110) with a θ-dependent prefilter. What is the correlation vector $\zeta(t, \theta)$ and what is the prefilter $L(q, \theta)$?

(b) By what numerical technique (according to the classification of this chapter) is the estimate computed?

(c) Suppose the numerical scheme converges to a unique solution $\hat{B}_N$, $\hat{F}_N$ of the correlation equation. Use Theorem 8.6 to discuss whether these estimates will be consistent if the true system is described by

$$y(t) = \frac{B_0(q)}{F_0(q)}u(t) + v_0(t)$$

where $\{v_0(t)\}$ is white or colored, respectively, noise. In case $\{v_0(t)\}$ is white, what does Theorem 9.2 say about the asymptotic variance of the estimate? (See also, Stoica and Söderström, 1981a, for the analysis.)

10G.3 *The EM algorithm:* Consider Problem 7G.6 and the expression (7.161) for the negative log likelihood function:

$$V(\theta) = -\log p(Y|\theta) = -\log p(Y, X|\theta) + \log p(X|\theta, Y)$$

This expression holds for all X, and can thus be integrated over any measure for X, $f(X)$, without affecting the X-independent left side:

$$V(\theta) = -\int_{X\in\mathbf{R}^n} \log p(Y, X|\theta) f(X)\, dX + \int_{X\in\mathbf{R}^n} \log p(X|\theta, Y) f(X)\, dX$$

Let now in particular $f(X)$ be the conditional PDF for X, given Y and assuming $\theta = \alpha$:

$$f(X) = p(X|Y, \alpha)$$

Then

$$V(\theta) = H_1(Y, \theta, \alpha) + H_2(Y, \theta, \alpha)$$

$$H_1(Y, \theta, \alpha) = -\int_{X\in\mathbf{R}^n} \log p(Y, X|\theta) p(X|Y, \alpha)\, dX = E(-\log p(Y, X|\theta)|Y, \alpha)$$

$$H_2(Y, \theta, \alpha) = \int \log p(X|Y, \theta) \cdot p(X|Y, \alpha)\, dX$$

The EM-algorithm (Dempster, Laird, and Rubin, 1977) for minimizing $V(\theta)$ consists of the following steps:

1. Fix α_k and determine the conditional mean of $-\log p(Y, X|\theta)$ with respect to X, given Y under the assumption that the true value of θ is α_k. This gives $H_1(Y, \theta, \alpha_k)$. (Note that the θ in $p(Y, X|\theta)$ is left as a free variable.)
2. Minimize
$$H_1(Y, \theta, \alpha_k)$$
with respect to θ giving $\hat{\theta}_k$.
3. Set $\alpha_{k+1} = \hat{\theta}_k$ and repeat from 1.

(a) Now, show that

$$H_2(Y, \hat{\theta}_k, \alpha_k) \leq H_2(Y, \alpha_k, \alpha_k)$$

and that hence

$$V(\hat{\theta}_k) \leq V(\alpha_k)$$

The algorithm thus produces decreasing values of the negative log likelihood function.

(b) Write out the EM-algorithm applied to the case of Problem 7G.6.

[Step 1 is the Estimation and step 2 the Minimization step in the EM-algorithm. The algorithm is useful when the likelihood function given Y is complicated and the addition of some auxiliary measurements X would have given a much simpler likelihood function. We thus expand the problem with these fake measurements and average over them using their conditional density given the actual observations and that the system is described by the current θ-estimate. Note that $H_2(Y, \theta, \alpha)$ is never formed in the algorithm.]

10G.4 Let $\mathbf{Y}$ and $\mathbf{U}$ be defined by (10.103). Consider the following LQ-factorization

$$\begin{bmatrix} \mathbf{U} \\ \mathbf{Y} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix}$$

where

$$\begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} [\, Q_1 \quad Q_2 \,] = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

Show that

$$\mathbf{Y}\Pi_{\mathbf{U}^T}^{\perp} = \mathbf{Y}(\mathbf{I} - \mathbf{U}^T(\mathbf{U}\mathbf{U}^T)^{-1}\mathbf{U}) = L_{22}Q_2^T$$

Hint: Just plug in the factorized expressions for $\mathbf{U}$ and $\mathbf{Y}$.

10G.5 Let $\mathbf{Y}$, $\mathbf{U}$ and Φ be defined by (10.103) and (10.108). Consider the LQ-factorization

$$\begin{bmatrix} \mathbf{U} \\ \Phi \\ \mathbf{Y} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \\ Q_3^T \end{bmatrix}, \quad Q^T Q = I.$$

Show that

(a) $\mathbf{Y}/_{\mathbf{U}}\Phi = \mathbf{Y}\Pi_{\mathbf{U}^T}^{\perp}\Phi^T(\Phi\Pi_{\mathbf{U}^T}^{\perp}\Phi^T)^{-1}\Phi = L_{32}L_{22}^{-1}[\, L_{21} \quad L_{22} \,]\begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix}$

(b) $\mathbf{Y}/_{\mathbf{U}}\Phi\Pi_{U^T}^{-} = L_{32}Q_2^T$

Here the big 0-part of L and the corresponding rows of Q of the original LQ-factorization have been thrown away. You may assume that indicated inverses exist.

Hint: Compare with the previous exercise.

(The notation $\mathbf{Y}/_{\mathbf{U}}\Phi$ for (10.120) is due to Van Overschee and DeMoor (1994) and is to be read: "The oblique projection of $\mathbf{Y}$ onto the space Φ, along the row space of $\mathbf{U}$".)

Note that (a) corresponds to the N4SID choice of $\hat{G}$ according to (10.120). Moreover (b) corresponds to the matrix $\hat{G}$ used in MOESP. Finally, note that according to Problem 10E.10, you can always post-multiply the matrices with an orthonormal matrix, without affecting the factor U_1 in the SVD (10.127). Hence we can work entirely with the "L-parts" of the factorizations above, and throw away the (big) Q-parts, when performing the calculations in (10.125).

10G.6 Show that (10.115) has full rank n provided

1. $\mathrm{E}\varphi_s(t)\varphi_s^T(t)$ is positive definite.
2. $\mathrm{E}\begin{bmatrix} x(t) \\ U_r(t) \end{bmatrix}[\, x^T(t) \quad U_r^T(t) \,]$ is positive definite. This means that the r future inputs should not be linearly dependent on the current state.
3. s_1 and s_2 are sufficiently large so that $\hat{x}(t) \approx L_x\varphi_s(t)$ for some L_x. (See (10.123)).

Hint: Use that $\mathrm{E}x(t)\varphi_s^T(t) = \mathrm{E}\hat{x}(t)\varphi_s^T(t)$, where $\hat{x}(t)$ is that part of the state that can be reconstructed from past input-outputs. Similarly $\mathrm{E}x(t)U_r^T(t) = \mathrm{E}\hat{x}(t)U_r^T(t)$.

10E.1 In Hsia (1977), Section 6.7, the following identification procedure is described. Let

$$\varphi_1^T(t) = \left[-y(t-1)\ldots -y(t-n_a) \quad u(t-1)\ldots u(t-n_b)\right]$$

$$\rho^T = \left[a_1 \ldots a_{n_a}\, b_1 \ldots b_{n_b}\right]$$

The model is then written as

$$y(t) = \varphi_1^T(t)\rho + \varepsilon(t)$$

The estimate $\hat{\rho}_N$ is computed as a "bias correction"

$$\hat{\rho}_N = \hat{\rho}_N^{\mathrm{LS}} - \hat{\rho}_N^{\mathrm{BIAS}} \tag{10.131}$$

where $\hat{\rho}_N^{\mathrm{LS}}$ and $\hat{\rho}_N^{\mathrm{BIAS}}$ are computed iteratively as follows:

Step 1. Let

$$R_N^{(1)} = \frac{1}{N}\sum_{t=1}^{N} \varphi_1(t)\varphi_1^T(t)$$

and

$$\hat{\rho}_N^{\mathrm{LS}} = \left[R_N^{(1)}\right]^{-1} \frac{1}{N}\sum_{t=1}^{N} \varphi_1(t)y(t)$$

$$\hat{\rho}_N^{\mathrm{BIAS}} = 0$$

Step 2. Let

$$\hat{\rho}_N = \hat{\rho}_N^{\mathrm{LS}} - \hat{\rho}_N^{\mathrm{BIAS}}$$

Step 3. Let

$$\varepsilon(t) = y(t) - \varphi_1^T(t)\hat{\rho}_N$$

and define

$$\varphi_2^T(t) = [-\varepsilon(t), \ldots, -\varepsilon(t-n_d)]$$

Let

$$R_N^{(2)} = \frac{1}{N}\sum_{t=1}^{N} \varphi_2(t)\varphi_2^T(t)$$

$$R_N^{(12)} = \frac{1}{N}\sum_{t=1}^{N} \varphi_1(t)\varphi_2^T(t)$$

Compute

$$D_N = R_N^{(2)} - \left[R_N^{(12)}\right]^T \left[R_N^{(1)}\right]^{-1} R_N^{(12)}$$

$$\hat{\rho}_N^{\mathrm{BIAS}} = \left[R_N^{(1)}\right]^{-1} R_N^{(12)} D_N^{-1} \left[\frac{1}{N}\sum_{t=1}^{N} \varphi_2(t)y(t) - \left[R_N^{(12)}\right]^T \left[R_N^{(1)}\right]^{-1} \frac{1}{N}\sum_{t=1}^{N} \varphi_1(t)y(t)\right]$$

and repeat from step 2 until convergence.

Now show that this procedure is the bootstrap algorithm (10.64) for the PLR method (7.113) using the ARARX model (4.22).

10E.2 Consider the model structure of Problem 5E.1. Give an expression for how to compute the gradient

$$\psi(t, \theta) = \frac{d}{d\theta}\hat{y}(t|\theta)$$

10E.3 Apply the Gauss-Newton method (10.40) and (10.46) to the linear regression problem (10.1) with quadratic criterion. Take $\mu = 1$.

10E.4 Introduce the approximation

$$\varepsilon(t, \theta) \approx \varepsilon(t, \hat{\theta}^{(i-1)}) + \psi^T(t, \hat{\theta}^{(i-1)})(\theta - \hat{\theta}^{(i-1)})$$

Use this approximation in

$$V_N(\theta, Z^N) = \frac{1}{2}\sum_{t=1}^{N}\varepsilon^2(t, \theta) \tag{10.132}$$

and solve for the minimizing θ. Show that this gives the (undamped) Gauss-Newton method (10.40) and (10.46).

10E.5 Consider Problem 10E.4. Minimize (10.132) subject to the constraint

$$\left\|\theta - \hat{\theta}^{(i-1)}\right\| \leq C_\delta$$

Discuss the relationship to the Levenberg-Marquardt method (10.47).

10E.6 Let V_n be defined by (10.23) and (10.24). Show that

$$V_n = \frac{1}{N}\sum_{t=1}^{N}\left(y(t) - \hat{y}_n(t|\theta)\right)^2 = \frac{1}{N}\sum_{t=1}^{N}e_n^2(t)$$

with $\hat{y}^n$ given by (10.15).

10E.7 Consider the ARX model

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n) = b_1 u(t-1) + \cdots + b_n u(t-n) + e(t)$$

Introduce

$$z(t) = \begin{bmatrix} -y(t) \\ u(t) \end{bmatrix}$$

and show how the estimates of a_i and b_i can be computed using the multivariable Levinson algorithm of Problem 10G.1.

10E.8 Show that, in the lattice filter, we have

$$\hat{y}_n(t|\hat{\theta}^n) = -\hat{\rho}_1\hat{r}_1(t-1) - \hat{\rho}_2\hat{r}_2(t-1) - \cdots - \hat{\rho}_n\hat{r}_n(t-1)$$

Compute the covariance matrix of $\hat{\rho}_i$, $i = 1, \ldots, n$.

10E.9 Apply the method (10.68) to the ARARX model (10.66). Spell out the steps explicitly and show that they consist of a sequence of LS problems mixed with simple filtering operations. [This is the generalized least squares (GLS) method, developed by Clarke (1967). See also Söderström (1974).]

10E.10 Let the $p \times N$ matrix G be given, with $p < N$. Let its SVD be

$$G = USV^T$$

(U is $p \times p$ with $U^TU = I$, and V is $N \times N$ with $V^TV = I$, and S is a $p \times N$ matrix with the singular values along the diagonal, and zeros elsewhere.) Suppose $p \leq r < N$ and W is a $N \times r$ matrix such that $W^TW = I$. Let

$$GW = U_1S_1V_1^T$$

be the SVD of GW. Show that $U = U_1$. (Hint: Note that, with MATLAB notation, `S*V'=S(:,1:r)*V(:,1:r)'`. Then use that `W'*V(:,1:r)` will be orthogonal.)

10E.11 The block matrix inversion lemma says:

$$\begin{bmatrix} A & D \\ C & B \end{bmatrix}^{-1} = \begin{bmatrix} \Delta^{-1} & -\Delta^{-1}DB^{-1} \\ -B^{-1}C\Delta^{-1} & B^{-1} + B^{-1}C\Delta^{-1}DB^{-1} \end{bmatrix}$$

where

$$\Delta = A - DB^{-1}C$$

Apply this to the matrix in (10.118); show that Δ becomes $\Phi\Pi_{\overline{\mathrm{UT}}}^{+}\Phi^T$ and that (10.119) holds.

10T.1 *Householder transformations:* A Householder transformation is a matrix

$$Q = I - 2ww^T$$

where w is a column vector with norm 1. Show the following

(a) Q is symmetric and orthogonal.

(b) Let x be an arbitrary vector. Then there exists a Q as in part (a) such that

$$Qx = |x| \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

(c) Let A be an $n \times m$ matrix, $n \geq m$. Then there exists an orthogonal matrix

$$\overline{Q} = Q_mQ_{m-1} \cdots Q_1$$

being a product of Householder transformations such that

$$\overline{Q}A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where R is a square $(m \times m)$ upper triangular matrix (see Lawson and Hanson, 1974).

10T.2 Consider the system

$$A_0(q)y(t) = B_0(q)u(t) + C_0(q)e_0(t)$$

and an ARMAX model structure

$$\theta = \left[a_1 \dots a_{n_a}\, b_1 \dots b_{n_b}\, c_1 \dots c_{n_c}\right]^T$$

$$A(q)y(t) = B(q)u(t) + C(q)e(t)$$

with polynomial orders larger or equal to those of the true system. Let

$$\tilde{D}_{\mathcal{M}} = \left\{\theta \,\middle|\, \mathrm{Re}\frac{C(e^{i\omega})}{C_0(e^{i\omega})} > 0 \forall \omega\right\}$$

Show that the prediction-error criterion

$$\overline{V}(\theta) = \overline{E}\varepsilon^2(t,\theta)$$

has no false local minimum in $\theta \in \tilde{D}_{\mathcal{M}}$: that is,

$$\overline{V}'(\theta) = 0 \text{ and } \theta \in \tilde{D}_{\mathcal{M}} \Rightarrow \frac{B(q)}{A(q)} = \frac{B_0(q)}{A_0(q)}, \qquad \frac{C(q)}{A(q)} = \frac{C_0(q)}{A_0(q)}$$

10T.3 Consider the method (10.68) to minimize (10.67) for a bilinear parametrization. Write (10.68) as an update step (10.40) with a block-diagonal $R_N^{(i)}$ matrix. It is thus indeed a descent method that will converge to a local minimum.

10D.1 Verify the relationships (10.31). *Hint:* By definition,

$$\frac{1}{N}\sum_{t=1}^{N} \hat{e}_n(t)y(t-k) = 0 \qquad 1 \le k \le n$$

[the residuals are uncorrelated with the regressors: see Figure II.1].

10D.2 Use Problem 10G.1 to derive a lattice filter for a multivariable signal $z(t)$.