[2] J. S. Lim and A. V. Oppenheim. *Advanced Topics in Signal Processing*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[3] S. L. Marple. *Digital Spectral Analysis with Applications*. Prentice Hall, Englewood Cliffs, NJ, 1987.

[4] L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, Englewood Cliffs, NJ, 1978.

[5] D. D. Jackson. Interpretation of inaccurate, insufficient, and inconsistent data. *Geophysical Journal of the Royal Astronomical Society*, 28:97–109, 1972.

[6] C. Lanczos. *Linear Differential Operators*. Van Nostrand, New York, 1961.

[7] J. Makhoul. Linear Prediction: A Tutorial Review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[8] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Prentice Hall, Englewood Cliffs, NJ, 1974.

[9] J. R. Deller, Jr., J. G. Proakis, J. H. L. Hansen. *Discrete-Time Processing of Speech Signals*. Macmillan, New York, 1993.

[10] M. Golomb and H. F. Weinberger. Optimal approximation and error bounds. In R. E. Langer, editor, *On Numerical Approximation*, chapter 6, pages 117–190. The University of Wisconsin Press, Madison, WI, 1959.

[11] R. G. Shenoy and T. W. Parks. An optimal recovery approach to interpolation. *IEEE Transactions on Signal Processing*, ASSP-40(8):1987–1996, August 1992.

[12] D. G. Luenberger. *Optimization by Vector Space Methods*. John Wiley & Sons, New York, 1969.

[13] G. Oetken, T. W. Parks, and H. W. Schüßler. New results in the design of digital interpolators. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23(3):301–309, June 1975.

[14] K. Aki and P. G. Richards. *Quantitative Seismology Theory and Methods*, Volume 2. W. H. Freeman and Co., San Francisco, 1980.

LINEAR PREDICTION

OVERVIEW

The idea of linear prediction is a powerful one in signal modeling. It is also directly connected to the use of all-pole models in spectrum estimation. The tutorial paper by Makhoul [7] provides an excellent overview of the subject. Many textbooks also treat the topic (e.g., Rabiner and Schafer [4] for speech processing). The next section deals with this important application.

In the prediction problem, we are given a signal $x[n]$ and we want to build a system that will predict future values. A *linear predictor* (Fig. 11.1) does this with an FIR filter.[1]

$$\hat{x}[n] = \sum_{k=1}^{P} (-a_k) x[n-k] \tag{0-1}$$

The best linear predictor will be one that minimizes an error such as least squares. If we want $\hat{x}[n]$ to be a "prediction" of the future value, $x[n+r]$, we minimize

$$E = \sum_{n} \left| x[n+r] - \hat{x}[n] \right|^2 \tag{0-2}$$

by choosing the predictor coefficients $\{a_k\}$. The range of the sum, to be specified later, leads to two different methods.

[1] The minus sign with the predictor coefficients $\{a_k\}$ is awkward but necessary to match the sign convention in MATLAB's `filter` function, and at the same time, express the prediction error $e[n]$ as a difference.

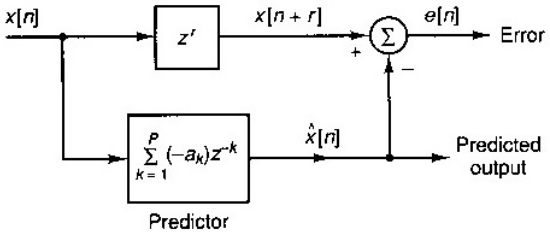

Figure 11.1
Block diagram for linear prediction. If $r = 0$, the predictor attempts to match the present value; if $r > 0$, it tries to predict a future value of $x[n]$.

After taking partials (or applying the orthogonality principle of least squares), the problem of minimizing E in (0-2) can be reduced to solving normal equations [8]. However, in MATLAB there is an easier way because the backslash operator (\) will solve a set of overdetermined linear equations in the least-squares sense. The predictor in (0-1) can be written out as a set of linear equations, with the minus sign moved to the left-hand side.

$$\begin{aligned}
-x[1+r] &\approx a_1x[0] + a_2x[-1] + \cdots + a_Px[1-P] && (n = 1)\\
&\vdots \quad\quad \vdots \quad\quad \vdots\\
-x[P+r] &\approx a_1x[P-1] + a_1x[P-2] + \cdots + a_Px[0] && (n = P)\\
&\vdots \quad\quad \vdots \quad\quad \vdots\\
-x[L-1] &\approx a_1x[L-2-r] + \cdots + a_Px[L-1-r-P] && (n = L-1-r)\\
&\vdots \quad\quad \vdots \quad\quad \vdots\\
-x[L-1+P+r] &\approx 0 + \cdots\cdots + 0 + a_Px[L-1] && (n = L-1+P)
\end{aligned} \tag{0-3}$$

This set of equations can be represented in matrix form as

$$-\mathbf{x} \approx \mathbf{X}\mathbf{a}$$

where the vector $\mathbf{x}$ and the matrix $\mathbf{X}$ contain known signal values. The squared error between the left- and right-hand sides will be minimized if the problem is solved in MATLAB via `a = -X \ x`. The resulting values for $\{a_k\}$ define the FIR linear predictor.

When $r = 0$ there are two methods of linear prediction, which are distinguished solely by which equations are included in the error sum (0-2).

1. *Autocorrelation method*: All possible equations from $n = 1$ to $n = L-1+P$ are included. Thus if the extent of the input data $x[n]$ is finite $0 \le n < L$, the prediction distance is r, and the length of the predictor is P, there will be $L-1+P$ equations. In some cases the predictor will be trying to match 0, because $x[L] = 0,\ x[L+1] = 0,\ \ldots,\ x[L-1+P+r] = 0$.
2. *Covariance method*: Only those equations for which all values of $x[n]$ needed on both sides are present in the data [i.e., equations $(n = P)$ to $(n = L-1-r)$ in (0-3)]. This method uses fewer equations, only $L-P-r$, but does not predict past the end of the data.

For long input sequences, however, there should be essentially no difference in the solution, which can be obtained with the backslash operator in either case.

There is often confusion over the notation used for the predictor coefficients $\{a_k\}$, because there is no standard convention used in textbooks and papers. In this section the sign of the predictor coefficients $\{a_k\}$ will be taken consistent with MATLAB, so that the "prediction error filter" $A(z)$ will have plus signs for the a_k's.

$$A(z) = z^r + \sum_{k=1}^{P} a_k z^{-k} \tag{0-4}$$

This is opposite from the convention found in [7]. The notable difference is that the error signal $e[n]$ must now be written with a plus sign:

$$e[n] = x[n+r] - \hat{x}[n]$$
$$= x[n+r] + \sum_{k=1}^{P} a_k x[n-k]$$

Thus the error signal $e[n]$ can be interpreted as the output of an LTI system with transfer function $A(z)$ and input $x[n]$.

PROJECT 1: LINEAR PREDICTION OF DATA

In this project, linear prediction is applied to synthetic signals and to real data. The real data are from the Dow-Jones Industrial Average sampled weekly for about 94 years. With such a long sequence, the linear prediction method can be designed over one section of the data and then tested over other sections to evaluate its effectiveness as a predictor. Performance of the method on real data should illustrate some of the limits of the method imposed by the inherent assumption that the data fit an all-pole model. The book by Marple [3] also contains an interesting data set—sunspot numbers for the years 1845–1978.

Hints

You may find the MATLAB function `convmtx` useful, along with the backslash (\) operator, which can solve simultaneous linear equations in the least-squares sense. For plotting poles and zeros, use the M-file `zplane` from Appendix A.

EXERCISE 1.1

Function for Linear Prediction

Write two MATLAB functions to compute the prediction error filter coefficients $\{a_k : k = 1, 2, \ldots, P\}$, one for the "autocorrelation" method and the other for the "covariance" method. Each function should accept three input arguments: a vector of data (`x`), the order of the predictor (`P`), and the prediction distance (`r`). The output arguments should include the filter coefficients (`a`), the error sequence (`e`), and an index variable containing the sample numbers at which the error signal was computed (`I`). An example function shell is as follows:

```
function [a,e,I] = covpred(x, p, r)
% COVPRED          Covariance Method to predict x[n+r]
%   Usage:   [a, e] = covpred(x, p, r)
%
%      x : input signal
%      p : order of predictor ( = number of poles)
%      r : (OPTIONAL) prediction distance, i.e., predict x[n+r]
%      a : prediction error filter coefficients
%      e : prediction error signal over I
%            total error is E = sum(abs(e).^2)
%      I : range of error signal
%            (e.g., for covariance method I=p:(Lx-1+r))
%
% Example: x = filter(1, [1 0.2 0.3], [1 zeros(1,100)]);
%          [a,e] = covpred(x, 2, 0)
% The returned vector a should be [1  0.2  0.3]
```