Fourier transforms based on Gaussian analysis windows," in *Rec. 1979 IEEE Int. Conf. Acoust., Speech, Signal Processing*, Washington, DC, Apr. 1979, pp. 186–189.

[22] T. G. Stockham, Jr., "High-speed convolution and correlation," in 1966 *Conf. Proc. AFIPS Spring Joint Comput. Conf.* Reprinted in *Digital Signal Processing*, L. R. Rabiner and C. M. Rader, Ed. New York: IEEE Press, 1972.

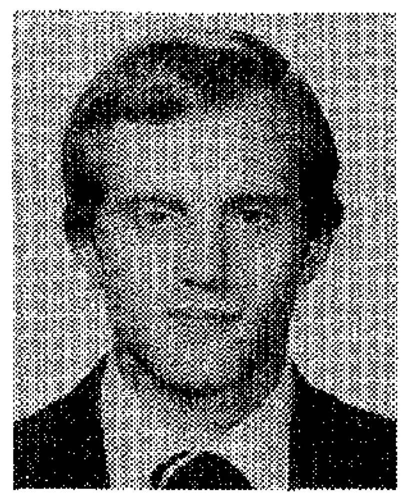
[23] C. R. Patisaul and J. C. Hammett, Jr., "Time-frequency resolution experiment in speech analysis and synthesis," *J. Acoust. Soc. Amer.*, vol. 58, pp. 1296–1307, Dec. 1975.

[24] R. J. Wang, "Optimum window length for the measurement of time-varying power spectra," *J. Acoust. Soc. Amer.*, vol. 52, part 1, pp. 33–38, Jan. 1972.

[25] G. Gambardella, "A contribution to the theory of short-time spectral analysis with nonuniform bandwidth filters," *IEEE Trans. Circuit Theory*, vol. CT-18, July 1971.

[26] J. E. Youngberg and S. F. Boll, "Constant-Q signal analysis and synthesis," in *Rec. 1978 IEEE Int. Conf. Acoust. Speech, Signal Processing*, Tulsa, OK, Apr. 1978, pp. 375–378.

[27] R. E. Crochiere, "A weighted overlap-add method of short-time Fourier analysis/synthesis," *IEEE Trans. Acoust., Speech, Signal Processing*, this issue, pp. 99–102.



Michael R. Portnoff (S'69–M'77) was born in Newark, NJ, on July 1, 1949. He was educated at the Massachusetts Institute of Technology, Cambridge, MA, receiving the S.B., S.M., and E.E. degrees in electrical science and engineering in 1973, and the Sc.D. degree in electrical engineering and computer science in 1978.

From 1969–1971, he was a cooperative student at Bell Telephone Laboratories, Inc. From 1971–1978, he was a Research Assistant in the M.I.T. Research Laboratory of Electronics, Digital Signal Processing Group, and a Teaching Assistant in the M.I.T. Department of Electrical Engineering and Computer Science. From 1978–1979, he was a Research Associate in the Digital Signal Processing Group at M.I.T. In 1979 he joined the University of California Lawrence Livermore Laboratory, where he is currently a Staff Member in the Engineering Research Division. His research interests are in the theory of digital signal processing and its application to speech, image, and seismic signal processing.

Dr. Portnoff received the 1977 IEEE Browder J. Thompson Memorial Prize Award and is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.

On the Implementation of a Short-Time Spectral Analysis Method for System Identification

LAWRENCE R. RABINER, FELLOW, IEEE, AND JONT B. ALLEN, MEMBER, IEEE

***Abstract*—Recent work has demonstrated the utility of a short-time spectral analysis approach to the problems of spectral estimation and system identification. In this paper several important aspects of the implementation are discussed. Included is a discussion of the computational effects (e.g., storage, running time) of the various analysis parameters. A computer program is included which illustrates one implementation of the method.**

I. Introduction

The problems of spectral estimation and system identification have been of great importance for a variety of applications. Although classical techniques have had various degrees of success, particular problems often require specialized techniques for the most efficient cost-effective solutions. Recently, a new method for spectral estimation and system identification was proposed based on the theory of short-time spectral analysis [1], [2]. This method was shown to be theoretically equivalent to the classical least squares method when the number of data points (N) was infinite [1]. For finite N the method has the property that the "misalignment" error (between the actual and computed system impulse responses) tends to zero as $1/N$, i.e., the solution rapidly approaches the least squares solution.

The purpose of this paper is to describe one implementation of the method described in [2]. Following a brief review of the basic method (Section II), we describe a DFT implementation in which the relevant quantities used in the analysis equation are computed entirely in the frequency domain (Section III). In Section IV we discuss the issues of computation speed, storage, and accuracy and show that tradeoffs between these factors can be made. Finally, in Section V we present a flowchart of one implementation of the method which is fairly general purpose.

II. Review of the Short-Time Spectral Analysis Approach to System Identification

Assume the input to the system to be identified is $x(n)$ and the output of the system [corrupted by additive noise $q(n)$] is $y(n)$, i.e.,

$$y(n) = x(n) * h(n) + q(n) \tag{1}$$

Manuscript received April 20, 1979; revised July 30, 1979.

The authors are with the Acoustics Research Department, Bell Laboratories, Murray Hill, NJ 07974.

0096-3518/80/0200-0069$00.75

where $h(n)$ is the (FIR) response of the linear system being identified, and $q(n)$ is an independent [of $x(n)$, $h(n)$] white noise with zero mean and variance σ_q^2. Assume we can observe $x(n)$ and $y(n)$ for $0 \leqslant n \leqslant N - 1$. The short-time spectral analysis approach to estimating $h(n)$ is to form overlap-add expansions of $x(n)$ and $y(n)$ [3]-[5], and then to approximate the classical least squares matrix equation solution for $h(n)$ by a simple Toeplitz matrix equation of the form

$$\hat{\boldsymbol{\Phi}}\hat{\boldsymbol{h}} = \hat{\boldsymbol{r}} \tag{2}$$

where $\hat{\boldsymbol{h}}$ is the $\hat{M}$ length vector

$$\hat{\boldsymbol{h}} = \begin{bmatrix} \hat{h}(0) \\ \hat{h}(1) \\ \vdots \\ \hat{h}(\hat{M} - 1) \end{bmatrix} \tag{3}$$

that approximates $\boldsymbol{h}$, the true impulse response, and $\hat{\boldsymbol{\Phi}}$ is an $\hat{M} \times \hat{M}$ symmetric Toeplitz matrix with the (l, m)th element

$$\hat{\phi}(l, m) = \hat{\phi}(l - m) = \sum_{p \in S} \sum_{k \in S} \phi_{p,k}(l, m) \tag{4}$$

where

$$\phi_{p,k}(l, m) = \frac{1}{D^2} \sum_{n=-\infty}^{\infty} x(n - l)\, x(n - m) \cdot w(pR + l - n)\, w(kR + m - n) \tag{5}$$

$$D = \frac{W(e^{j0})}{R}. \tag{6}$$

$w(n)$ is an L-point window used in the overlap-add expansion of $x(n)$, R is the shift (in samples) between adjacent windows, and $W(e^{j0})$ is the zero frequency value of the discrete Fourier transform of the window. Similarly, $\hat{\boldsymbol{r}}$ is the $\hat{M}$ length vector

$$\hat{\boldsymbol{r}} = \begin{bmatrix} \hat{r}(0) \\ \hat{r}(1) \\ \vdots \\ \hat{r}(\hat{M} - 1) \end{bmatrix} \tag{7}$$

with components

$$\hat{r}(l) = \sum_{p \in S} \sum_{k \in S} r_{pk}(l) \tag{8}$$

where

$$r_{pk}(l) = \frac{1}{D^2} \sum_{n=-\infty}^{\infty} y(n)\, x(n - l)\, w(pR - n)\, w(kR + l - n). \tag{9}$$

The set S in (4) and (8) are the integers p, k such that the pth and kth windows of the data are entirely in the range $0 \leqslant n \leqslant N - 1$, and such that the overlap between the windows is in the range [2]

$$\hat{M} - 1 \leqslant n \leqslant N - 1. \tag{10}$$

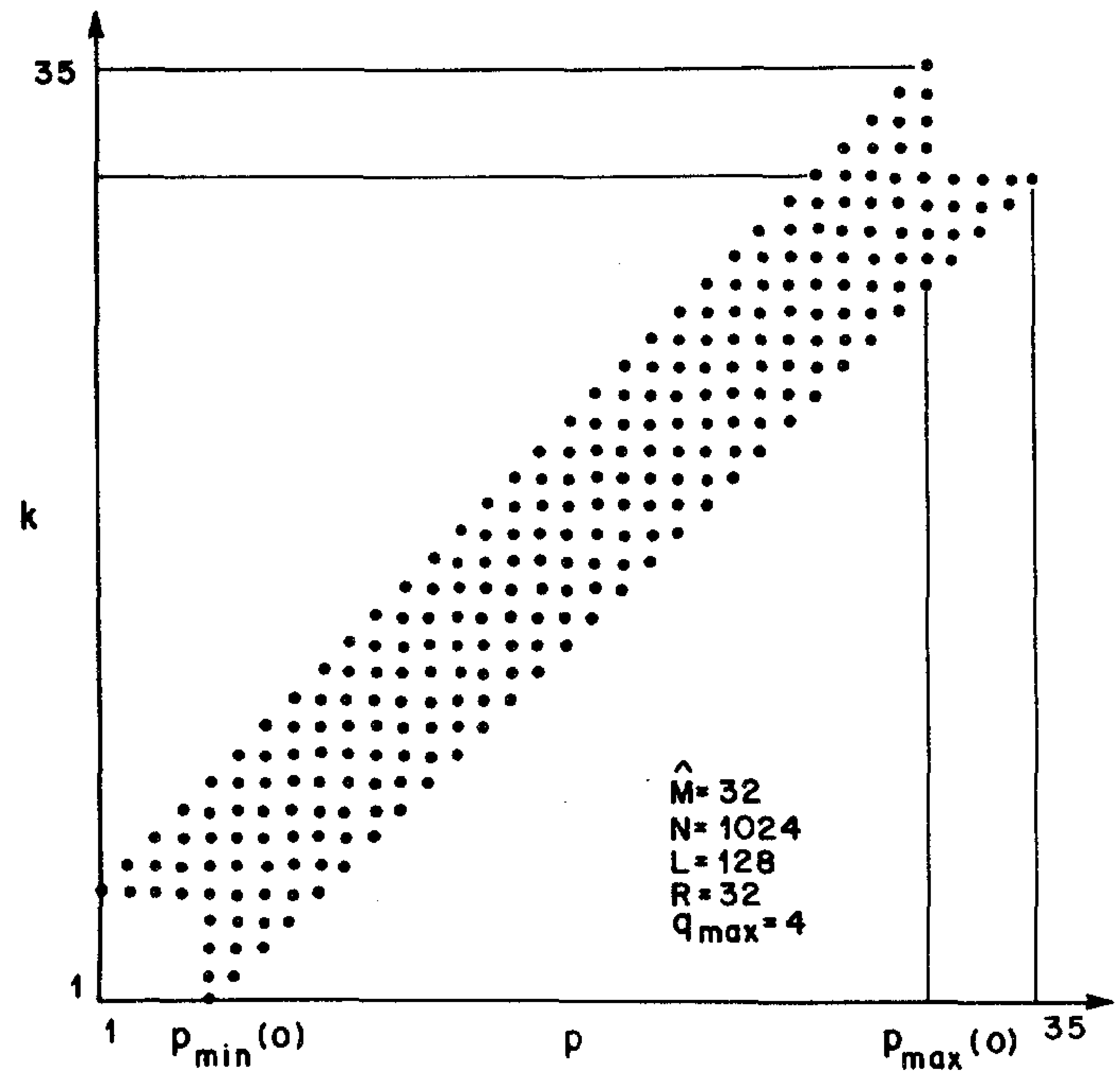


Fig. 1. Typical set of points (heavy dots) comprising the set S in the (p, k) plane which are used in computing $\hat{\phi}$ and $\hat{r}$.

As described in [2], the range of $p, k \in S$ is a strip in the (p, k) plane as illustrated in Fig. 1. By making the substitution

$$k = p + q, \tag{11}$$

(4) and (8) reduce to the forms

$$\hat{\phi}(l - m) = \sum_{q=-q_{\max}}^{q_{\max}} \sum_{p=p_{\min}(q)}^{p_{\max}(q)} \phi_{p,p+q}(l, m) \tag{12}$$

$$\hat{r}(l) = \sum_{q=-q_{\max}}^{q_{\max}} \sum_{p=p_{\min}(q)}^{p_{\max}(q)} r_{p,p+q}(l) \tag{13}$$

where

$$q_{\max} = \left\lfloor \frac{\hat{M} + L - 2}{R} \right\rfloor \tag{14}$$

where $\lfloor x \rfloor$ is the integer less than or equal to x, and

$$p_{\min}(q) = \left\lceil \frac{L + \hat{M} - 2}{R} \right\rceil - \max(0, q) \tag{15a}$$

$$p_{\max}(q) = \left\lfloor \frac{N - \hat{M}}{R} \right\rfloor - \min(0, q) \tag{15b}$$

where $\lceil x \rceil$ is the integer greater than or equal to x.

We now give a procedure for solving for $\hat{h}(n)$ from windowed sections of $x(n)$ and $y(n)$. The steps in the process are as follows.

1) Choose window $w(n)$, window length L, and window shift R. Compute D from (6).

2) Determine range on q [(14)], and p [(15)] for calculation of $\hat{\boldsymbol{\Phi}}$ and $\hat{\boldsymbol{r}}$.

3) For each pair of (p, q), determine $\phi_{p,p+q}(l, m)$ and $r_{p,p+q}(l)$ from (5) and (9). This computation is done for $0 \leqslant l \leqslant \hat{M} - 1$ and $0 \leqslant m \leqslant \hat{M} - 1$, and may be realized efficiently via fast correlation methods. (See Section III.)

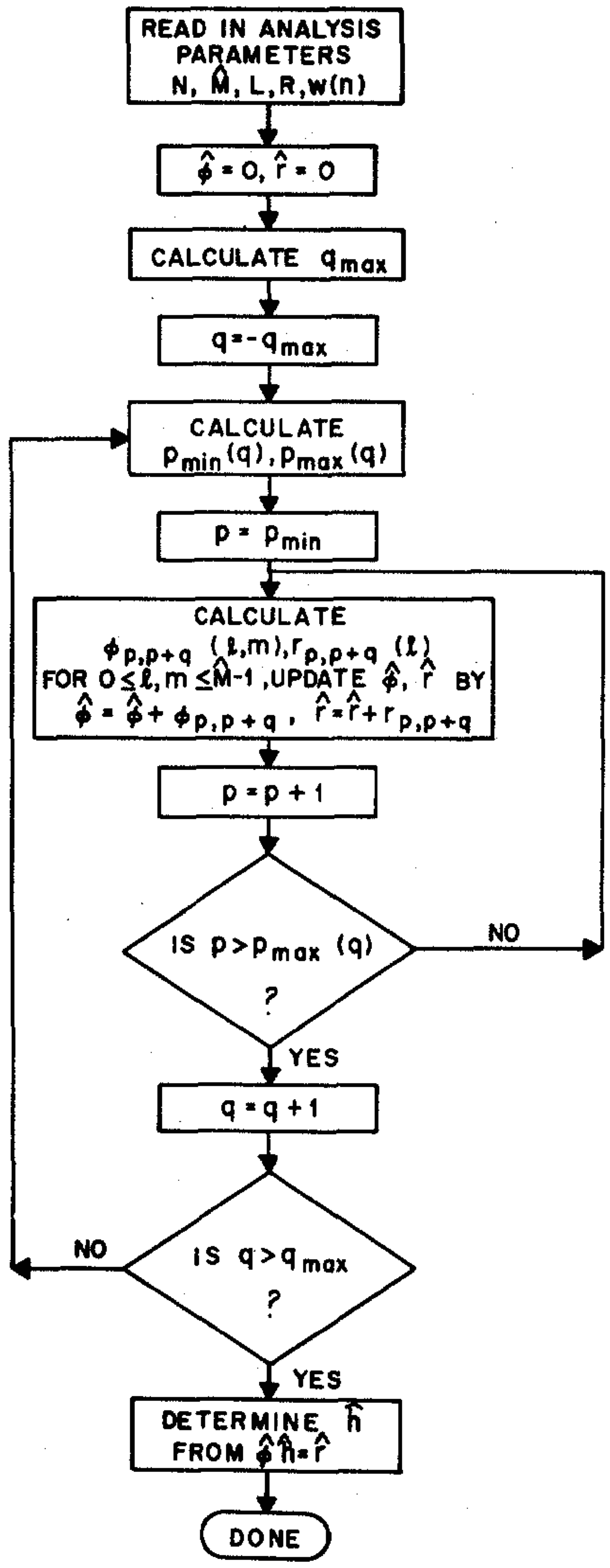


Fig. 2. Generalized flowchart of the short-time spectral analysis method.

4) Determine $\hat{\phi}(l-m)$ and $\hat{r}(l)$ by summing over the pairs of (p, q) indices of step 3.

5) Solve matrix (2) for $\hat{h}$ using a Toeplitz matrix solution method, e.g., the Trench method [6], or a Levinson algorithm [7].

Fig. 2 gives a flowchart corresponding to the above procedure. There are many ways in which the operations of the flowchart can be carried out. For example, we can consider several alternative methods of indexing p and q over all the grid points in the solution. Furthermore a variety of techniques can be used to calculate $\phi_{p,p+q}(l,m)$ and $r_{p,p+q}(l)$ for the complete range of l and m. In Section III we describe an FFT method which trades storage for computational speed. Finally, the Toeplitz matrix equation can be solved by any number of Toeplitz matrix solution methods. In Section III we discuss these alternative implementation techniques.

III. DFT Implementation of the System Identification Problem

We begin by considering the computation of the term $\phi_{p,p+q}(l,m)$ of (5) with $k = p+q$. We denote the pth window of x as $x_p(n)$. It is readily shown that (5) can be written as

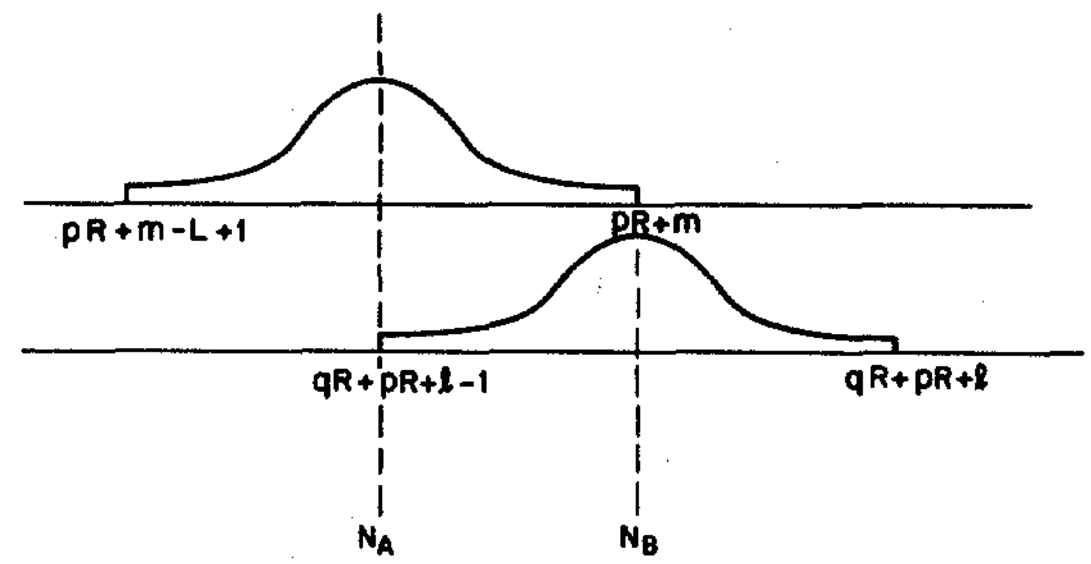


Fig. 3. Relative positions of the pth and $(p+q)$th windows for the matrix element $\phi_{p,p+q}$ (or $r_{p,p+q}$). The range $NA \leqslant n \leqslant NB$ is the overlap between the windows.

$$\phi_{p,p+q}(l,m) = \phi_{p,p+q}(l-m) = \frac{1}{D^2} \sum_{n=-\infty}^{\infty} \cdot x_p(n-l)\, x_{p+q}(n-m) \tag{16}$$

$$= \frac{1}{D^2}\, x_p(n) * x_{p+q}(-n), \tag{17}$$

i.e., as a correlation between $x_p(n)$ and $x_{p+q}(n)$, whenever the overlap between the pth and $(p+q)$th data windows are within the closed interval $[\hat{M}-1, N-1]$. Fig. 3 illustrates the placement of the pth and $(p+q)$th windows. If we define N_A as the lower limit on the overlap between windows, and N_B as the upper limit of the overlap, then (16) (with $s = l - m$) becomes the finite correlation

$$\phi_{p,p+q}(s) = \frac{1}{D^2} \sum_{n=N_A}^{N_B} x_p(n)\, x_{p+q}(n+s) \tag{18}$$

where

$$N_A = pR - L + 1 + \max(m, qR + l) \tag{19a}$$

$$N_B = pR + \min(m, qR + l). \tag{19b}$$

Equation (18) can be implemented using fast (FFT) correlation methods. However, we must carefully choose the FFT section size to guarantee no aliasing for the maximum q value for which (18) is valid, i.e., $q = q_{\max}$. It can readily be seen from (18) that the FFT section size NF has 3 components, namely the window length L, the maximum shift (in samples) between windows $q_{\max} \cdot R$, and the aliasing protection for $\hat{M}-1$ values of the correlation [i.e., for $r = 0, 1, \cdots, \hat{M}-1$ in (18)]. As such, we get

$$NF \geqslant L + q_{\max} \cdot R + (\hat{M}-1) \tag{20a}$$

$$= L + \left\lfloor \frac{\hat{M}-2+L}{R} \right\rfloor \cdot R + (\hat{M}-1). \tag{20b}$$

For our present FFT implementations, (i.e., radix 2), NF is chosen to be the power of 2 greater than or equal to NF of (20a). We will see in Section IV that (20a), along with some subsequent equations for the number of FFT's which must be performed, provides guidance on the choice of window length L, relative to $\hat{M}$, to minimize overall computation and storage.

In the implementation of the fast correlation computation of (18), it is assumed that the FFT size NF is an integer multiple of the shift between windows R. This assumption leads to a simple and efficient strategy for accounting for the real time placement of the pth window within the finite FFT frame. The idea is based on the well-known shifting property of Fourier transforms, namely if

$$x(n) \leftrightarrow X(e^{j\omega}) \tag{21a}$$

$$x(n - pR) \leftrightarrow X(e^{j\omega})\, e^{-j\omega pR}, \tag{21b}$$

or for NF point DFT's

$$x(n) \leftrightarrow X(k) \tag{22a}$$

$$x(n - pR) \leftrightarrow X(k)\, e^{-j(2\pi/NF)kpR}. \tag{22b}$$

If we define

$$K = NF/R \tag{23}$$

then (22b) shows that to compensate for the shift of pR samples we modulate $X(k)$ by the factor $e^{-j(2\pi/K)kp}$. The modulating function

$$G(k) = e^{-j(2\pi/K)k} \tag{24}$$

can be implemented as a K point complex table, and the modulation for a pR sample delay is implemented by accessing every pth point of the table, modulo K. Thus, to implement the FFT convolution we have to access the pth data window and store it in $x(n)$ for $n = 0, 1, \cdots, L - 1$, take its DFT, and modulate the DFT by the table $G(k)$ accessed every pth point modulo K, i.e.,

$$\tilde{X}(0) = X(0)\, G(0)$$
$$\tilde{X}(1) = X(1)\, G(p \oplus K)$$
$$\tilde{X}(2) = X(2)\, G(2p \oplus K)$$
$$\vdots$$

where $p \oplus K$ means p modulo K.

Similarly the windowed sequence $x_{p+q}(n)$ is accessed, transformed, and phase compensated. The desired correlation could be obtained as

$$\phi^{(s)}_{p,p+q} = \mathrm{DFT}^{-1}\, [\tilde{X}_p \tilde{X}^*_{p+q}] \tag{25}$$

and its results are valid for $0 \leqslant s \leqslant \hat{M} - 1$. The computation for $\hat{\boldsymbol{\phi}}$ (or $\hat{r}$), however, is clearly more efficiently done entirely in the frequency domain as

$$\hat{\phi}(s) = \mathrm{DFT}^{-1} \left[\sum_p \sum_q \tilde{X}_p \tilde{X}^*_{p+q} \right], \tag{26}$$

i.e., by accumulating the lagged products in the frequency domain and transforming back to the time domain only as a final step.

A. Summation Method in the (p, k) Plane

There are several alternative ways in which the quantities $\hat{\phi}$ and $\hat{r}$ of (12) and (13) can be calculated. The straightforward implementation of (12) is illustrated in Fig. 4(a). The computation along the path labeled 1 is for $q = -q_{\max}$ and all valid p. This is next followed by the path labeled 2 for $q = -q_{\max} + 1$ and all valid p. This is carried out until the $q = q_{\max}$ path is traced and the computation is finished. Although this summation method is valid, it suffers from (small) numerical problems of the following type. Each term $\phi_{p,p+q}$ entering into the computation of (12) decreases in magnitude as $|q|$ becomes large since the overlap between the pth and $(p + q)$th windows decreases. As such, the contributions of the $q_{\max}$ path [labeled 7 in Fig. 4(a)] to the total are numerically distorted because, by the time they are added, $\hat{\phi}$ is already large. As such, an alternate, numerically more accurate, method of computing $\hat{\phi}$ is illustrated in Fig. 4(b). Here the $q = -q_{\max}$ and $q = q_{\max}$ paths are computed first, followed by the $q = -q_{\max} + 1$ and $q = q_{\max} - 1$, etc. While the amount of computation remains the same, the accuracy greatly increases.

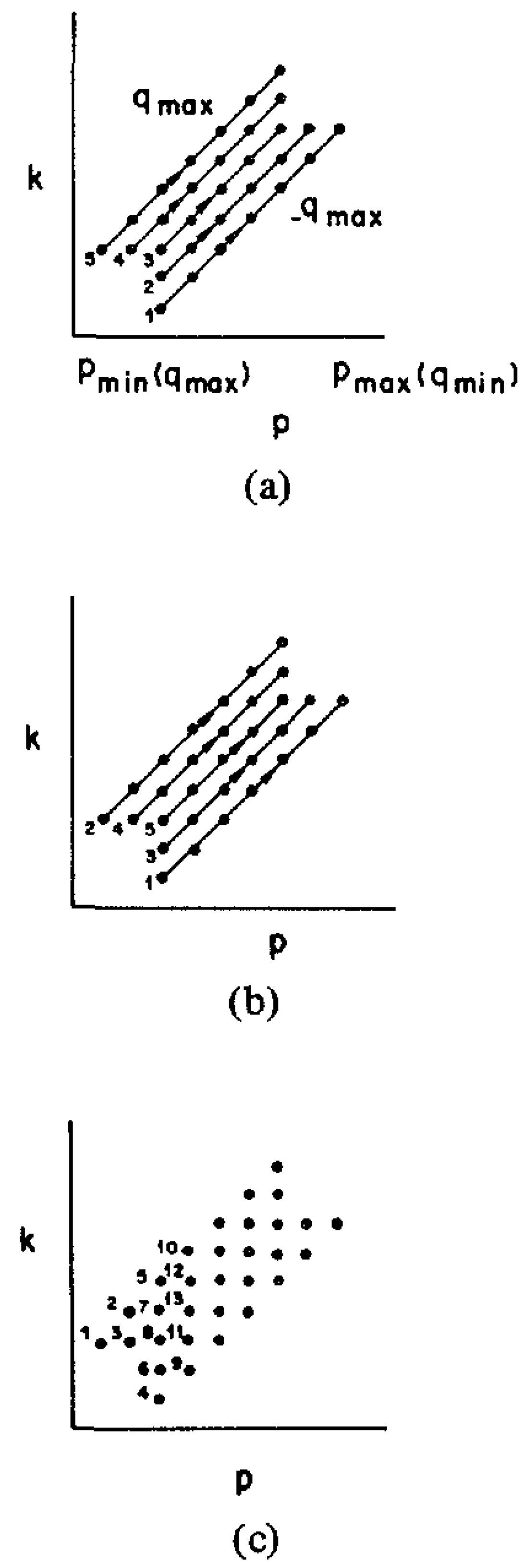


Fig. 4. Three possible ways of implementing the computation of ϕ_{pk} (or r_{pk}) for all valid sets of (p, k) in the plane.

The only problem with the computation of Fig. 4(b) is that a total of (approximately)

$$NC = 2(2q_{\max} + 1)(p_{\max}(q_{\min}) - p_{\min}(q_{\max})) \tag{27}$$

FFT's must be performed, i.e., 2 for each (p, q) pair. This strategy is clearly inefficient in that the total number of DFT's need be no more than the total number of rows $(p_{\max}(q_{\min}))$ and columns $(p_{\max}(q_{\min}))$. Thus, if we perform the summations of (12) in the manner shown in Fig. 4(c), namely by indexing p from $p_{\min}(q_{\max})$ to $p_{\max}(q_{\min})$, and then determining the range of q (or k) for each p, we can compute the DFT of the pth window just one time, store it, and use it for the computations of each of the q (or k) windows which are relevant. Similarly, if we have adequate storage (enough for $2q_{\max} + 1$ DFT's), we can store a vertical strip of DFT's and reduce computation of each column to a single column DFT (for the pth window) and a single row DFT [for the $(p + q_{\max})$th window]. Thus, with sufficient storage, the total number of DFT's is reduced to

$$NCP = 2(p_{\max}(q_{\min}) - p_{\min}(q_{\max})), \tag{28}$$

which can be considerably less than NC of (27). We can also employ our previous argument and along each column compute the DFT's so that the largest values of q are done first. When an entire column of computations is accumulated, it is then added to the previous computations, thus assuring maximum overall accuracy. Fig. 4(c) shows the order in which the computations would be done for one simple example.

B. *Final Solution of the Toeplitz Matrix Equation*

The final step in the system identification procedure is the solution of the Toeplitz matrix equation

$$\sum_{m=0}^{\hat{M}-1} \hat{\phi}(l-m)\,\hat{h}(m) = \hat{r}(l), \qquad l = 0, 1, \cdots, \hat{M}-1. \tag{29}$$

The matrix $\hat{\boldsymbol{\Phi}}$ is Toeplitz and symmetric. Two Toeplitz matrix solution methods were investigated, namely, the Trench method [6] and the Levinson method [7]. Both techniques require on the order of $\hat{M}^2$ multiplications and additions, and on the order of $\hat{M}$ storage locations. Informal experimentation with both methods indicated little or no difference in the solution for a number of examples. Hence either technique appears to be applicable to this problem. Since, in general, $N \gg \hat{M}$, the computations required in solving the Toeplitz matrix equation is generally negligible compared to those of computing $\hat{\boldsymbol{\Phi}}$ or $\hat{r}$.

IV. Computational Considerations

We have already discussed two major computational aspects of the method, namely the use of high-speed correlation to compute $\phi_{p,p+q}$ and $r_{p,p+q}$ terms, and a carefully chosen path in the (p, k) or (p, q) plane to minimize the number of FFT's required for the computation of $\hat{\boldsymbol{\Phi}}$ or $\hat{\boldsymbol{r}}$. There remains one additional computational consideration, namely, the choice of window length L. Theoretically, any value of L can be chosen. However, the amount of computation C in computing $\hat{\boldsymbol{\Phi}}$ or $\hat{r}$ is approximately

$$C = \text{Number of FFT's} \times \text{Computation per FFT} \tag{30}$$

$$= 2(p_{\max}(q_{\min}) - p_{\min}(q_{\max})) * NF \log_2 (NF) \tag{31}$$

where we have used (28) and (20a) to give the number of FFT's and the FFT size. From (15) and (14) we get

$$C(L) = 2\cdot\left[\left\lfloor\frac{N-\hat{M}}{R}\right\rfloor + \left\lfloor\frac{\hat{M}+L-1}{R}\right\rfloor - \left\lceil\frac{L+\hat{M}-2}{R}\right\rceil + \left\lfloor\frac{\hat{M}+L-2}{R}\right\rfloor\right]\cdot NF\log_2(NF). \tag{32}$$

We recall from our earlier discussion that, in general, NF is chosen as the power of 2 greater than or equal to the quantity NF of (20a). Fig. 5(a) shows a typical plot of computed values of NF and the nearest power of 2 as a function of the variable L for the case $\hat{M} = 16$, $N = 1000$.[1] We see the result

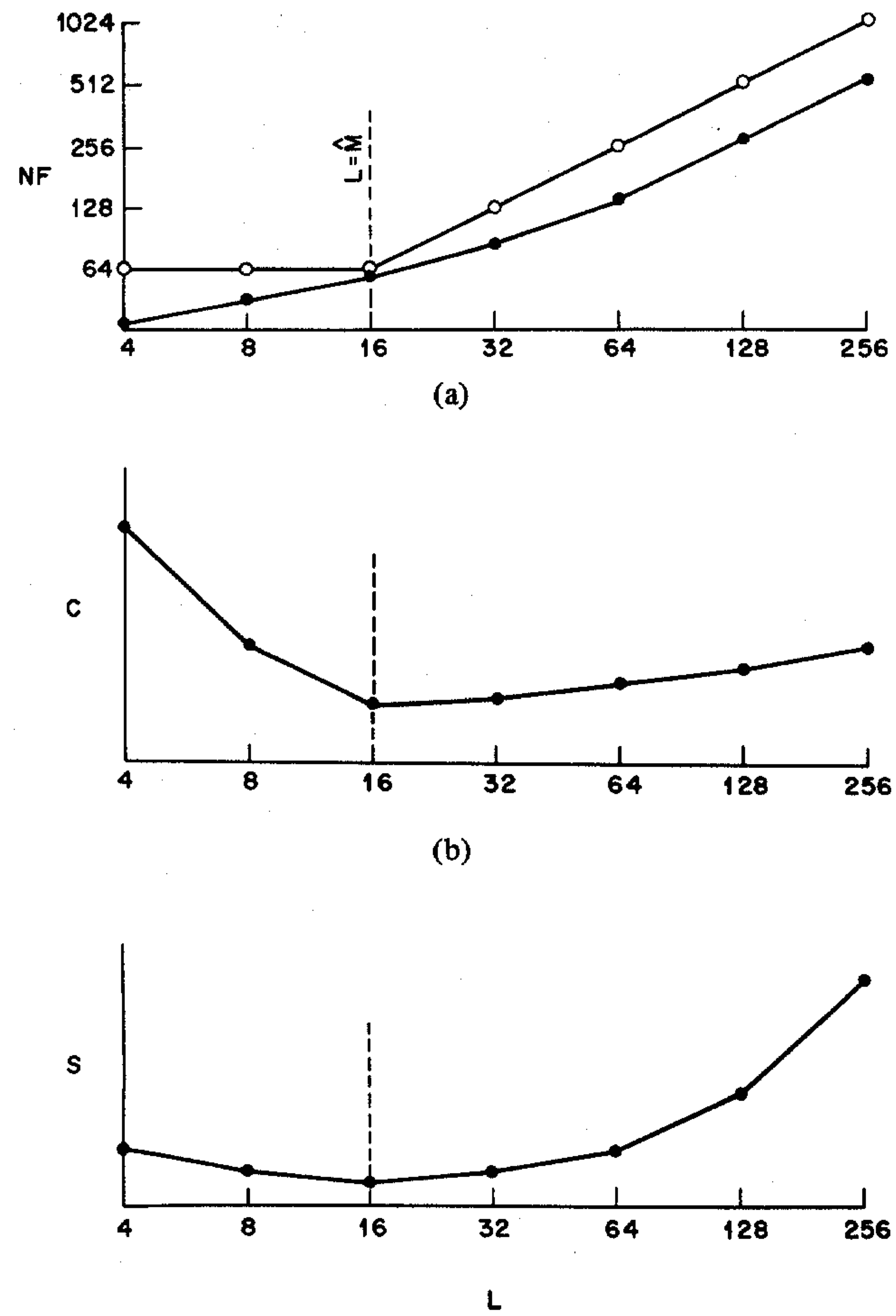


Fig. 5. Curves of FFT size (NF), computation (C), and storage (S) as a function of window size L for a given value for $\hat{M}$ and N, with $R = L/4$.

that for $L = \hat{M}$ the actual FFT size is closest to the computed value of NF. This result is valid when $\hat{M}$ is a power of 2 (or slightly less than a power of 2). For arbitrary $\hat{M}$, a slightly more complex picture emerges and we have to consider the total computation $C(L)$ of (32). This quantity is plotted in Fig. 5(b) for the parameters $\hat{M} = 16, N = 1000$. It can be seen that $C(L)$ decreases sharply until $L \approx \hat{M}$, at which point the curves rises only gradually. As such it can be argued that any reasonable value of $L \geqslant \hat{M}$ would serve to approximately minimize the total computation of $\hat{\boldsymbol{\Phi}}$ and $\hat{r}$.

If we now consider the storage required for the computation of $\hat{\boldsymbol{\Phi}}$ or $\hat{r}$, we see that we need to store a strip of width $(2q_{\max} + 1)$ DFT's. Thus the storage required is (approximately)

$$S(L) = (2q_{\max} + 1) \times \text{FFT (size)} \tag{33}$$

$$= \left(2\left\lfloor\frac{\hat{M}+L-2}{R}\right\rfloor + 1\right)\cdot \text{FFT (size)}. \tag{34}$$

Fig. 5(c) shows a plot of S as a function of L for the example of Fig. 5. It can be seen that the minimum value of S occurs at $L = \hat{M}$. The storage increases by 50 percent for $L = \hat{M}/2$, or $L = 2\hat{M}$, thus a fairly well-defined minimum of S occurs at $L = \hat{M}$.

Based on the above discussion, it is seen that the optimum computational strategy is to choose a value of L on the order

[1] For simplicity we assume $R = L/4$. For arbitrary R, less than this value, the results do not change significantly.

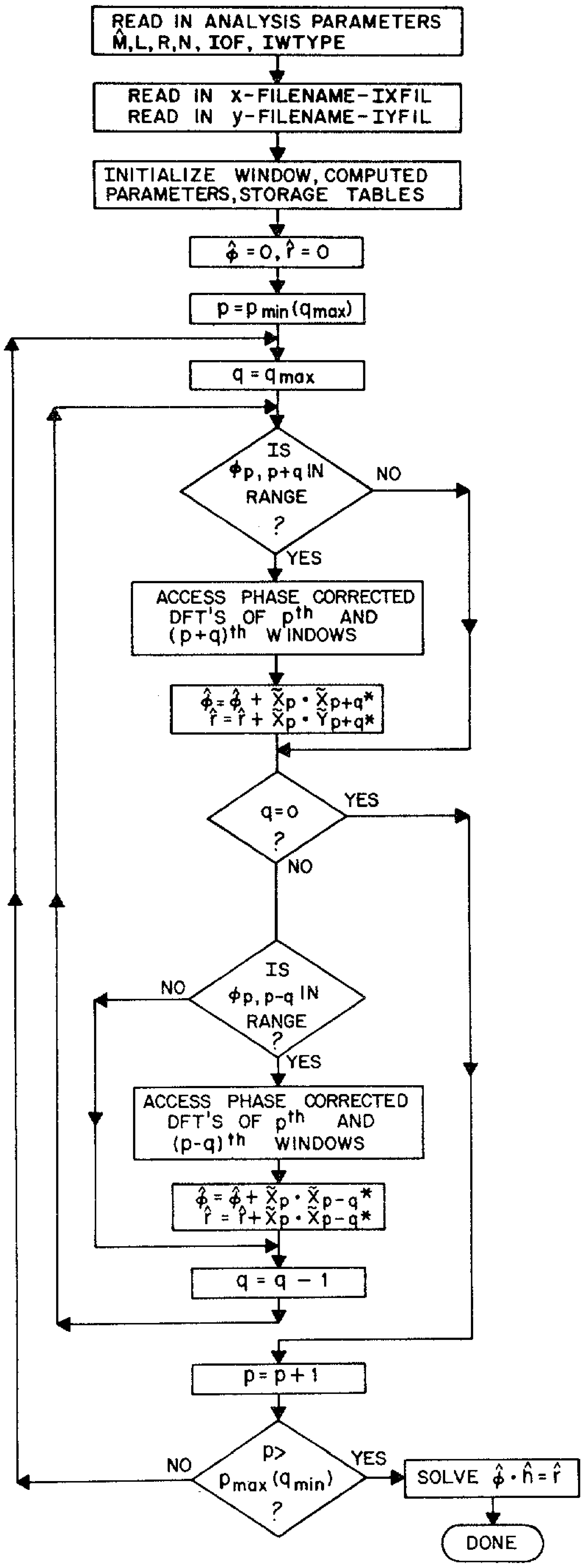


Fig. 6. Flowchart of the implementation described in this paper.

of $\hat{M}$ to simultaneously minimize total computation and total value of NF.

V. Flowchart, Computer Program, and Test Examples

A flowchart of the implementation used to realize the system identification methods described in Sections II and III is given in Fig. 6. A Fortran implementation of the flowchart is given as the test program TESTSTSPEST, the subroutine STSPEST, and its associated subroutines. The program assumes the sequences $x(n)$ and $y(n)$ are stored in disk files. Thus, it first reads in the disk file names for the input $(x(n))$ and output $(y(n))$ sequences. Channels are assigned to the disk files for reading values of $x(n)$ and $y(n)$. Next, the basic analysis parameters of the method are read in including $\hat{M}, L, R$, and N. Other parameters requested include an initial sample (IOF) in the files at which the sequences begin, i.e., the sample number corresponding to $n = 0$ in the equations, the window type, IWTYPE (1 for Hamming window, 0 for rectangular window), and the maximum value of q (IQCO) to be used in the analysis.

The subroutine computes $\hat{\phi}$ and $\hat{r}$ using the FFT fast convolution method of Section III on the path of Fig. 4(c). Then the Toeplitz matrix equation is solved using the Levinson method [7], and the resulting estimate of the system impulse response is returned to the main program. At this point the user can insert code to plot the impulse response estimate or the resulting frequency response estimate.

For maximum flexibility, all parameters and data arrays are passed in the calling statement to STSPEST. Although cumbersome, this ensures that the routine uses the minimum storage for implementation.

Two of the subroutines called within STSPEST are not provided in the Appendix. One is the machine dependent disk read routine RSECT, which reads in samples (in fixed point format) of $x(n)$ or $y(n)$ (depending on channel number) into a buffer beginning at a designated sample number on the file. The calling statement for the routine is

```
CALL RSECT (NCH, IBUF, NRD, XST, IER)
```

where

NCH = Channel number for reading, i.e., 0 for reading input samples, 1 for reading output samples.
IBUF = Buffer for storing integer input or output samples.
NRD = Number of samples of $x(n)$ or $y(n)$ to be read.
XST = Starting sample number in disk file.
IER = Error code.

The second set of missing routines are the FFT subroutines FAST and FSST, which are described in [8]. The calling sequences are

```
CALL FAST (X, N)

CALL FSST (X, N)
```

where FAST is used for a direct FFT of the real sequence $x(n)$ stored in array X of size N (where N must be a power of 2). The transform $X(k)$ is stored in the array X (i.e., the input data is overwritten) in the format

$$\text{Re}\,[X(0)] \to X(1)$$
$$\text{Im}\,[X(0)] \to X(2)$$
$$\text{Re}\,[X(1)] \to X(3)$$
$$\text{Im}\,[X(1)] \to X(4)$$
$$\vdots$$
$$\text{Re}\,[X(N/2)] \to X(N+1)$$
$$\text{Im}\,[X(N/2)] \to X(N+2).$$

A total of $N + 2$ locations are required for an N point FFT. The subroutine FSST does the inverse FFT and expects input

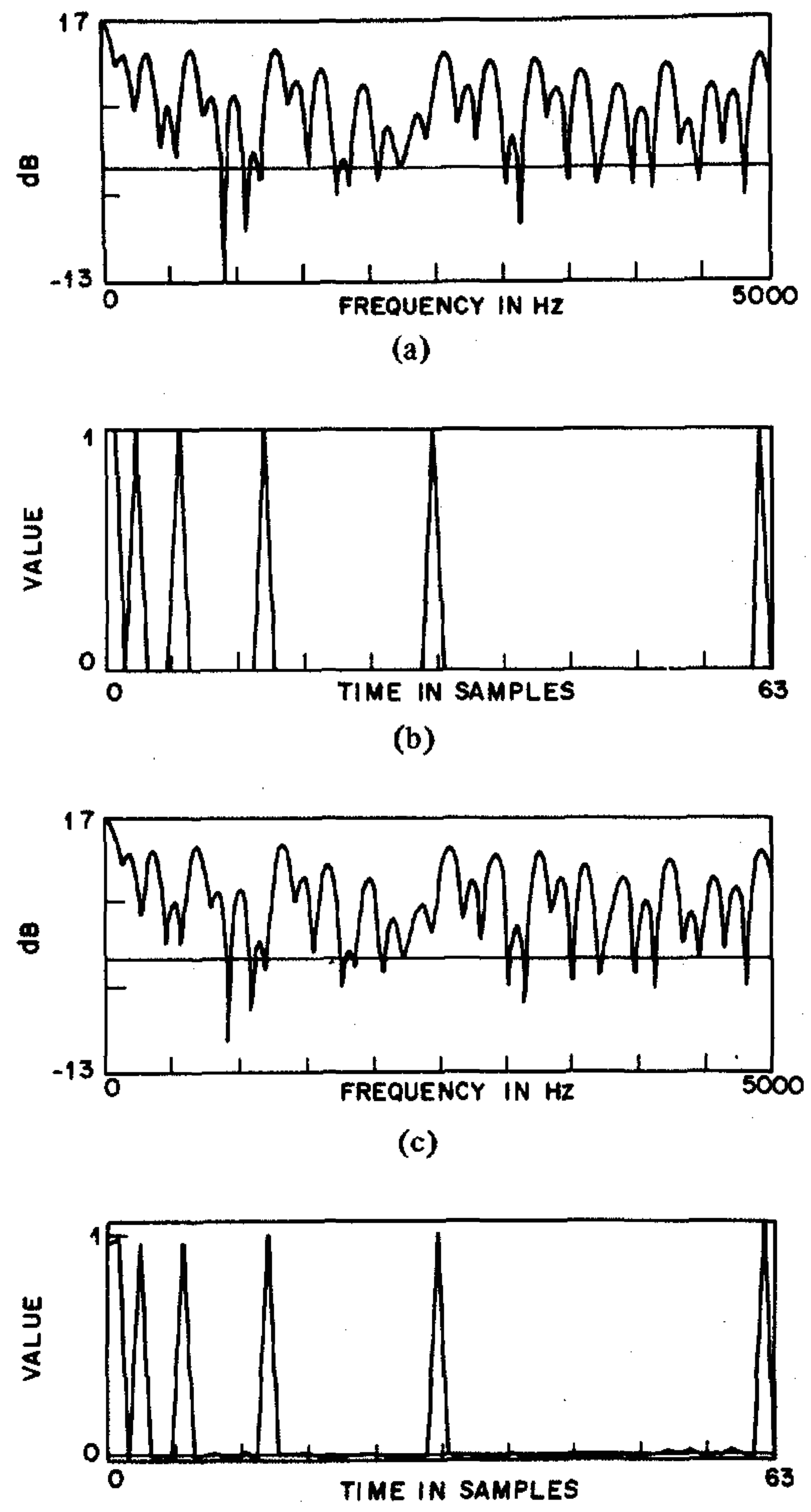


Fig. 7. Actual and estimated impulse responses [parts (b) and (d)], and log magnitude frequency responses [parts (a) and (c)] for a 64 point example.

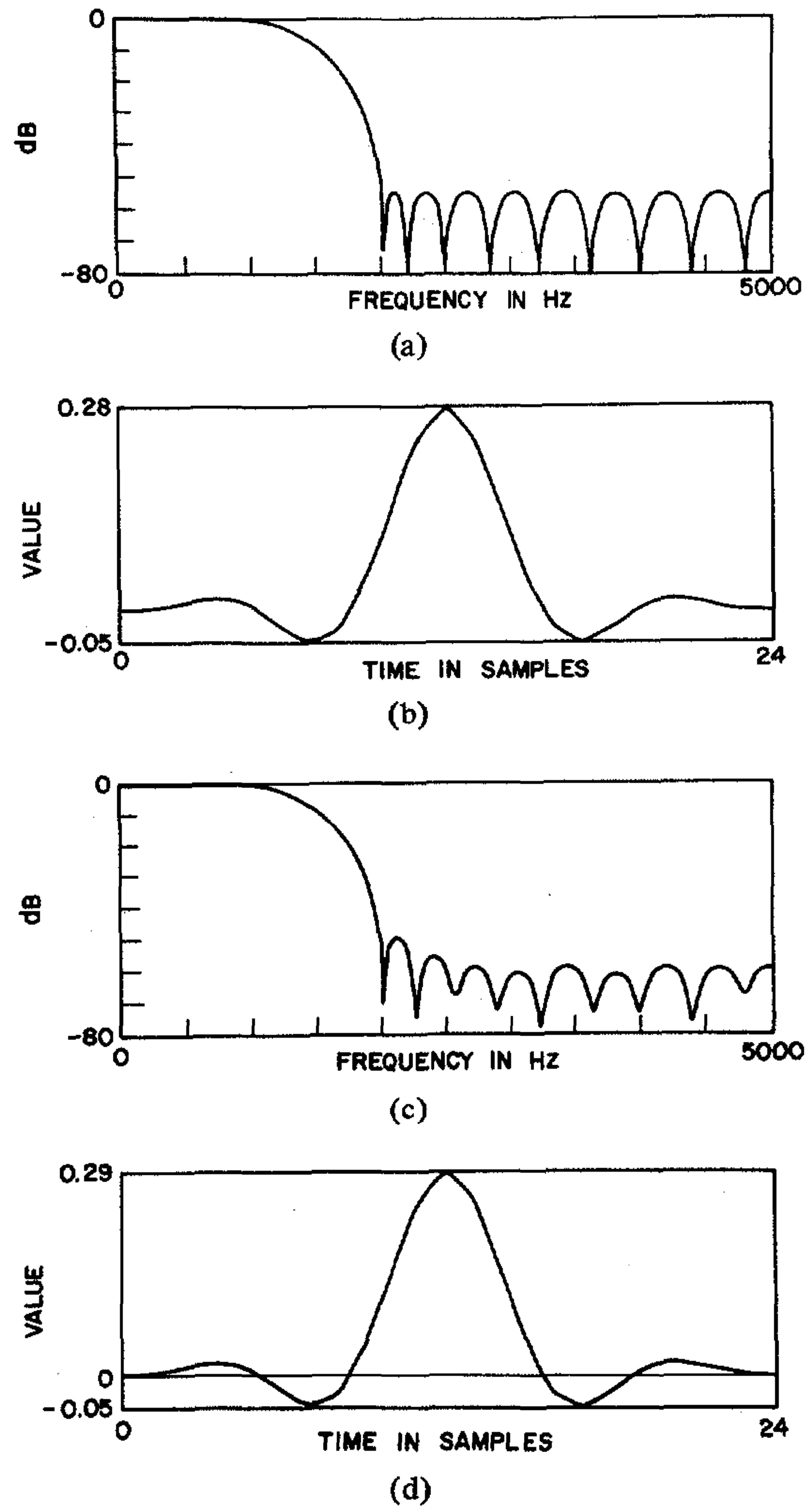


Fig. 8. Actual and estimated impluse responses [parts (b) and (d)] and log magnitude frequency responses [parts (a) and (c)] for a 25 point low-pass filter example.

data in the format obtained from FAST, and writes the real N point output over the first N input values.

Figs. 7 and 8 show examples of the use of the program. There are four parts to each of these figures. Parts (b) and (d) show $h(n)$, the true impulse response, and $\hat{h}(n)$, the estimate, whereas parts (a) and (c) show the true and estimated log magnitude responses. Fig. 7 is for a 64 point impulse response where

$$h(n) = 1 \quad n = 0, 1, 3, 7, 15, 31, 62$$
$$= 0 \quad \text{otherwise},$$

with analysis parameters $N = 1024$, $R = 16$, $L = \hat{M} = 64$, IOF = 500, and IWTYPE = 1 (Hamming window). The parameter IQCO specifies the largest value of $q_{\max}$ in the implementation. For full accuracy, IQCO is set to -1, or any large integer (e.g., 1000). The error in $\hat{h}(n)$ can be seen for values of n such that $h(n) = 0$ where $\hat{h}(n)$ is a small random value.

Fig. 8 is for an equiripple 25-point FIR linear-phase low-pass filter with a peak sidelobe ripple of -55 dB. The analysis parameters here were $N = 1024$, $R = 8$, $L = 32$, $\hat{M} = 25$, IOF = 100, IWTYPE = 1, and all q values retained. A peak log magnitude error of about 5 dB (relative to the maximum of the sidelobes) is seen in this figure.

V. Summary

In this paper we have described one implementation of the method described in [2]. We have attempted to make the implementation as efficient (in terms of speed and memory) and as accurate as possible, within the framework that was given. The implementation resides as a Fortran callable subroutine, and a simple main program was given which provides a first-level application of the routine.

Appendix

```
C
C-----------------------------------------------------------------------
C MAIN PROGRAM:      TEST OF STSPEST SUBROUTINE
C AUTHORS:           L. R. RABINER AND JONT B. ALLEN
C                    BELL LABORATORIES
C                    MURRAY HILL, NEW JERSEY, 07974
C
C INPUT:             MHAT=IMPULSE RESPONSE LENGTH IN SAMPLES
C                    L=WINDOW LENGTH IN SAMPLES
C                    N=NUMBER OF SAMPLES FOR LEAST SQUARES
C                      SOLUTION, NPRIME=N-MHAT+1
C                    IOF=STARTING SAMPLE IN DATA FILES FOR
C                      BOTH X AND Y DATA
C                    IWTYPE=WINDOW TYPE, 1 FOR HAMMING WINDOW
C                                        0 FOR RECTANGULAR WINDOW
C                    IQCO=MAXIMUM RANGE ON Q
C                    IFIL=INPUT FILENAME (X-DATA), OPENED ON
C                      CHANNEL 0
C                    JFIL=OUTPUT FILENAME (Y-DATA), OPENED ON
C                      CHANNEL 1
C-----------------------------------------------------------------------
C
```

```
      COMMON WIN(128),XW(NM),YTAB(NML),ZW(NM)
      COMPLEX XWC(NHF),YTABC(NHFL),ZWC(NHF)
      EQUIVALENCE (XW(1),XWC(1)),(YTAB(1),YTABC(1)),(ZW(1),ZWC(1))
      COMPLEX TMP(NHF)
      COMPLEX XM(64)
      DIMENSION IFIL(10),JFIL(10)
      DIMENSION PHIHAT(128),RHAT(128),H(128)
      INTEGER TTI,TTO
      INTEGER P,R
      PARAMETER NM=514, NHF=NM/2, NML=NM*9, NHFL=NHF*9
C
C  DEFINE TELETYPE INPUT AND TELETYPE OUTPUT DEVICES
C
      TTI=11
      TTO=10
C
C  DEFINE MAXIMUM ARRAY SIZES FOR COMPUTATION
C
C
C  READ IN X-DATA FILENAME AND Y-DATA FILENAME
C  SUBROUTINE GNAME READS IN AN ASCII FILENAME FROM TELETYPE
C
      WRITE(TTO,1)
 1    FORMAT("***X-DATA FILENAME***")
      CALL GNAME(IFIL)
      OPEN 0,IFIL
      WRITE(TTO,2)
 2    FORMAT("***Y-DATA FILENAME***")
      CALL GNAME(JFIL)
      OPEN 1,JFIL
C
C  READ IN ANALYSIS PARAMETERS, MHAT,R,L,N,IOF,IWTYPE,IQCO
C
 10   CONTINUE
      WRITE(TTO,3)
 3    FORMAT(" MHAT(I4)=")
      READ(TTI,4) MHAT
 4    FORMAT(I4)
      WRITE(TTO,5)
 5    FORMAT(" R(I4)=")
      READ(TTI,4) R
      WRITE(TTO,6)
 6    FORMAT(" L(I4)=")
      READ (TTI,4) L
      WRITE(TTO,7)
 7    FORMAT(" N(I6)=")
      READ(TTI,8) N
 8    FORMAT(I6)
      WRITE(TTO,9)
 9    FORMAT(" IOF(I6)=")
      READ(TTI,8) IOF
      WRITE(TTO,11)
 11   FORMAT(" WINDOW TYPE(1 FOR HW, 0 FOR RW)=")
      READ(TTI,12) IWTYPE
 12   FORMAT(I1)
      WRITE(TTO,13)
 13   FORMAT(" IQCO(I4)=")
      READ(TTI,4) IQCO
C
C  CALL SPECTRAL ANALYSIS ROUTINE
C
      CALL STSPEST(PHIHAT,RHAT,H,1,IERR,MHAT,R,L,N,IOF,
     1 IWTYPE,IQCO,NM,9,NHF,NML,NHFL,WIN,XW,YTAB,ZW,TMP,XM,
     2 XWC,YTABC,ZWC)
C
C  H(I) ARRAY CONTAINS THE ESTIMATE OF THE SYSTEM IMPULSE RESPONSE
C  USER CAN INSERT CODE FOR PLOTTING IMPULSE RESPONSE OR ITS
C  FREQUENCY RESPONSE HERE
C
      GO TO 10
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE: STSPEST
C SHORT TIME SPECTRAL ANALYSIS ROUTINE
C GENERALIZED SYSTEM IDENTIFICATION ANALYSIS
C-----------------------------------------------------------------------
C
      SUBROUTINE STSPEST(PHIHAT,RHAT,H,IPRT,IERR,MHAT,R,L,N,IOF,
     1 IWTYPE,IQCO,NM,MAXFFT,NHF,NML,NHFL,WIN,XW,YTAB,ZW,TMP,XM,
     2 XWC,YTABC,ZWC)
      DIMENSION PHIHAT(1),RHAT(1),H(1)
      DIMENSION WIN(1),XW(1),YTAB(1),ZW(1),TMP(1),XM(1),XWC(1)
      DIMENSION YTABC(1),ZWC(1)
      COMPLEX TMP,XM,XWC,YTABC,ZWC
      INTEGER P,R
C
C  PHIHAT=ARRAY TO HOLD PHIHAT(I),I=1,MHAT
C  RHAT=ARRAY TO HOLD RHAT(I),I=1,MHAT
C  H=ARRAY TO HOLD H(I),I=1,MHAT
C  IPRT=PRINTING PARAMETER--IPRT=1 TO PRINT, OTHERWISE NO PRINTING
C  IERR=ERROR FLAG
C      IERR=0 MEANS ALL IS OK WITHIN STSPEST
C      IERR=1 MEANS REQUIRED FFT SIZE IS TOO LARGE
C      IERR=2 MEANS MODULATION FACTOR (IMD) IS TOO LARGE
C      IERR=3 MEANS INSUFFICIENT STORAGE FOR YTAB
C
C  ******ANALYSIS PARAMETERS******
C  MHAT=IMPULSE RESPONSE LENGTH
C  R=NO OF SAMPLES BETWEEN WINDOWS
C  L=WINDOW LENGTH IN SAMPLES
C  N=NUMBER OF SAMPLES FOR LEAST SQUARES SOLUTION
C      I.E. N PRIME=N-MHAT+1
C  IOF=STARTING SAMPLES IN BOTH X-DATA AND Y-DATA FILES
C  IWTYPE=WINDOW TYPE--1 FOR HAMMING WINDOW,  0 FOR RECT WIND
C  IQCO=MAXIMUM RANGE ON Q CALCULATION--SET IQCO TO -1 FOR NO LIMIT
C  NM=MAXIMUM SIZE OF LOCAL ARRAYS FOR SHORT TIME SPECTRA
C  NHF=NM/2
C  NML=MAXIMUM STORAGE AVAILABLE FOR RECURSIVE ESTIMATION PART
C  NHFL=NML/2
C  MAXFFT=MAXIMUM POWER OF 2 FOR FFT
C  WIN=ARRAY TO HOLD WINDOW
C  XW=X STORAGE ARRAY--EQUIVALENCED TO XWC
C  YTAB=Y STORAGE TABLE--EQUIVALENCED TO YTABC
C  ZW=RESULTS STORAGE ARRAY--EQUIVALENCED TO ZWC
C  TMP=TEMPORARY STORAGE FOR ACCUMULATION OF RESULTS
C  XM=PHASE FACTOR TABLE--COMPLEX
C
C  CREATE APPROPRIATE (HAMMING OR RECTANGULAR) WINDOW OF LENGTH L
C    AND CALCULATE D=W(0)/R NORMALIZATION CONSTANT
C
C
C  DEFINE OUTPUT DEVICE FOR PRINTING (LPT)
C
      LPT=12
      IERR=0
      IF(IQCO.LT.0) IQCO=1000
      IF(IWTYPE.EQ.1) CALL CHAM(WIN,L)
      IF(IWTYPE.EQ.0) CALL CRECT(WIN,L)
      W0=0.
      DO 20 I=1,L
 20   W0=W0+WIN(I)
      D=W0/FLOAT(R)
C
C  CALCULATE FFT SIZE AND PHASE FACTOR TABLE
C
      XF=FLOAT(MHAT-2+L)/FLOAT(R)
      NFFT=L+ICEIL(XF)*R+(MHAT-1)
      DO 30 I=2,MAXFFT
      MTST=2**I
      IF(MTST.GE.NFFT) GO TO 40
 30   CONTINUE
      IERR=1
      RETURN
 40   CONTINUE
C
C  NFFT IS SIZE OF FFTS USED IN COMPUTATION
C  NF2 AND NFHF ARE EXTENDED AND HALF FFT SIZES FOR REAL
C    AND COMLEX ARRAYS
C  IMD IS MODULO PHASE FACTOR FOR TIME SHIFTING SEQUENCES
C
      NFFT=MTST
      NF2=NFFT+2
      NFHF=NF2/2
      IMD=NFFT/R
      IF(IMD.LE.64) GO TO 45
      IERR=2
      RETURN
 45   TWOPI=8.*ATAN(1.0)
C
C  CREATE PHASE FACTOR TABLE TO MODULATE EACH SHORT TIME TRANSFORM TO
C    ACCOUNT FOR PROPER TIME SEQUENCING
C
      DO 50 I=1,IMD
      T=TWOPI*FLOAT(I-1)/FLOAT(IMD)
 50   XM(I)=CMPLX(COS(T),-SIN(T))
C
C  DETERMINE QMIN, QMAX AND QRANGE=QMIN-QMAX+1
C
      XF=FLOAT(2-MHAT-L)/FLOAT(R)
      IQMIN=ICEIL(XF)
      IF(IQMIN.LT.(-IQCO)) IQMIN=-IQCO
      XF=FLOAT(MHAT-2+L)/FLOAT(R)
      IQMAX=IFLOR(XF)
      IF(IQMAX.GT.IQCO) IQMAX=IQCO
      IQR=IQMAX-IQMIN+1
C
C  NML IS MAXIMUM AVAILABLE STORAGE FOR RECURSIVE COMPUTATION OF PHIHAT
C    AND R
C
      IF(IQR*NF2.LE.NML) GO TO 55
      IERR=3
      RETURN
C
C  DETERMINE PA AND PB RANGE
C
 55   XF=FLOAT(L+MHAT-2)/FLOAT(R)
      IPA=ICEIL(XF)
      XF=FLOAT(N-MHAT)/FLOAT(R)
      IPB=IFLOR(XF)
C
C  LOOP FOR COMPUTING PHIHAT AND RHAT
C  JJ=1 FOR PHIHAT
C  JJ=2 FOR RHAT
C
      DO 220 JJ=1,2
      CALL ZERO(YTAB,NF2*IQR)
      CALL ZERO(ZW,NF2)
C
C  INITIALIZE YTAB FOR Y WINDOWS FROM 1 TO -IQMIN
C
      JJK=JJ-1
      JQMIN=-IQMIN
      DO 110 I=1,JQMIN
      I1=NF2*(I-1)+1
      I2=NFHF*(I-1)+1
 110  CALL GETSIG(YTAB(I1),XM,YTABC(I2),WIN,I,NFFT,
     1 L,R,N,IOF,IMD,JJK,1)
      IND=-IQMIN+1
C
C  LOOP ON P INDEX AND FIND ALL Q (OR K) VALUES
C
      IPA1=IPA+IQMIN
      IPA2=IPB+IQMAX
      DO 170 IP=IPA1,IPA2
C
C  READ IN X ARRAY DATA FOR IP-TH WINDOW
C
      CALL GETSIG(XW,XM,XWC,WIN,IP,NFFT,L,R,N,IOF,IMD,0,0)
C
C  READ IN Y ARRAY FOR (IP-IQMIN)-TH WINDOW
C
      INDY=IP-IQMIN
      IF(INDY.GT.(IPB+IQMAX)) GO TO 140
      I1=NF2*(IND-1)+1
      I2=NFHF*(IND-1)+1
      CALL GETSIG(YTAB(I1),XM,YTABC(I2),WIN,INDY,NFFT,L,R,N,
     1 IOF,IMD,JJK,1)
 140  CALL ZERO(TMP,NF2)
C
C  ACCUMULATE RESULTS FOR EACH VALUE OF P(IP) BY SUMMING ACROSS
C    VALUES OF Q(IQ)
C
      IQQ=-IQMIN+1
      DO 160 JQ=1,IQQ
      IQ=JQ-IQQ
      IQL=IQ
      DO 160 JCT=1,2
      ICT=JCT-1
      IF(ICT.EQ.1) IQL=-IQL
```

```
      IF(ICT.EQ.1.AND.IQL.EQ.0) GO TO 160
      IP1=IPA-MAX0(IQL,0)
      IP2=IPB-MIN0(IQL,0)
      IF(IP.LT.IP1.OR.IP.GT.IP2) GO TO 160
      INDP=MOD(IQL+IP-1,IQR)+1
      INDP1=NFHF*(INDP-1)
      DO 150 I=1,NFHF
      INDP1=INDP1+1
150   TMP(I)=TMP(I)+YTABC(INDP1)
160   CONTINUE
C
C  ACCUMULATE SUM OVER VALUES OF P(IP) ACROSS RANGE OF P
C
      DO 165 I=1,NFHF
165   ZWC(I)=ZWC(I)+XWC(I)*TMP(I)
      IND=IND+1
      IF(IND.GT.IQR) IND=1
170   CONTINUE
C
C  COMPLEX CONJUGATE RESULTS
C
      DO 175 I=1,NFHF
175   ZWC(I)=CONJG(ZWC(I))
C
C  PERFORM INVERSE FFT TO OBTAIN SEQUENCES PHIHAT AND RHAT
C
      CALL FSST(ZW,NFFT)
      DO 180 I=1,NFFT
 180  ZW(I)=ZW(I)/(D*D)
      DO 210 I=1,MHAT
      IF(JJ.EQ.1) PHIHAT(I)=ZW(I)
      IF(JJ.EQ.2) RHAT(I)=ZW(I)
210   CONTINUE
220   CONTINUE
C
C  SET UP LEVINSON SOLUTION OF TOEPLITZ MATRIX
C
      XC=PHIHAT(1)
      DO 230 I=1,MHAT
      PHIHAT(I)=PHIHAT(I)/XC
230   RHAT(I)=RHAT(I)/XC
C
C  PRINT OUT TO DEVICE LPT VALUES OF PHIHAT AND RHAT IF IPRT=1
C
      IF(IPRT.EQ.1) WRITE(LPT,3) (PHIHAT(I),I=1,MHAT)
 3    FORMAT(" PHIHAT=",4E13.5)
      IF(IPRT.EQ.1) WRITE(LPT,2)
 2    FORMAT(//)
      IF(IPRT.EQ.1) WRITE(LPT,4) (RHAT(I),I=1,MHAT)
 4    FORMAT(" RHAT=",4E13.5)
      IF(IPRT.EQ.1) WRITE(LPT,2)
C
C  SOLVE TOEPLITZ EQUATION FOR H

C
      CALL EUREKA(MHAT,PHIHAT,RHAT,H,XW)
C
C  PRINT OUT H OF DEVICE LPT IF IPRT=1
C
      IF(IPRT.EQ.1) WRITE(LPT,6) (H(I),I=1,MHAT)
 6    FORMAT(" H=",4E13.5)
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE: GETSIGD
C GET SIGNAL VALUES FOR SPECTRAL ESTIMATION
C READ VALUES FROM DISK FILE
C-----------------------------------------------------------------------
C
      SUBROUTINE GETSIG(XW,XM,XWC,WIN,IND,NFFT,L,R,N,IOF,IMD,IXY,ICJ)
      DIMENSION XW(1),WIN(1)
      COMPLEX XWC(1),XM(1)
      DIMENSION IBUF(128)
      INTEGER R
C
C  XW=ARRAY IN WHICH TO PUT SPECTRUM OF SIGNAL
C  XM=PHASE FACTOR ARRAY TO ACCOUNT FOR POSITION OF WINDOW
C  XWC=COMPLEX ARRAY EQUIVALENCED TO XW IN MAIN PROGRAM
C  WIN=WINDOW ARRAY--I.E. HAMMING WINDOW
C  IND=INDEX OF WINDOW TO BE ACCESSED
C  NFFT=SIZE OF FFT TO BE PERFORMED
C  L=WINDOW DURATION IN SAMPLES
C  R=SHIFT BETWEEN WINDOWS IN SAMPLES
C  N=TOTAL NUMBER OF SAMPLES FOR ANALYSIS
C  IOF=INITIAL SAMPLE IN FILE FOR READING
C  IMD=RATIO BETWEEN NFFT AND R--USED FOR PHASE FACTOR TABLE
C  IXY=VARIABLE INDICATING WHICH INPUT TO BE USED
C      IXY=0 USES X ARRAY
C      IXY=1 USES Y ARRAY
C  ICJ=VARIABLE TO CHOOSE WHETHER TO TAKE COMPLEX CONJUGATE OF SPECTRAL
C      ESTIMATE--ICJ=1 TAKES CONJUGATE--OTHERWISE NOT
C
      CALL IZERO(IBUF,L)
      CALL ZERO(XW,NFFT+2)
C
C  SCALE FACTOR IS MACHINE DEPENDENT
C  SCALE FACTOR USED HERE (FOR A 16-BIT MACHINE)
C  IS 32000.
C
      XSCAL=32000.
      I1=IND*R-L+1
      IST=1
      NRD=L
      IF(I1.GE.0) GO TO 5
      IST=L-IND*R
      I1=0
      NRD=L-IST+1
 5    XST=(IOF+I1)
      I1=IND*R
      IF(I1.LT.N) GO TO 8
      NRD=N-1+L-IND*R
C
C  RSECT IS A SUBROUTINE TO READ DATA FROM THE DISK FILE
C    FIRST ARGUMENT IS CHANNEL NUMBER (0 FOR INPUT, 1 FOR OUTPUT)
C    IBUF IS THE ARRAY WHICH HOLDS THE DATA READ FROM DISK
C    NRD IS THE NUMBER OF SAMPLES READ FROM THE DISK FILE
C    XST IS THE STARTING SAMPLE IN THE DISK FILE FOR READING
C    IEOF IS AND ERROR FLAG FOR READING
C
 8    IF(IXY.EQ.0) CALL RSECT(0,IBUF(IST),NRD,XST,IEOF)
      IF(IXY.EQ.1) CALL RSECT(1,IBUF(IST),NRD,XST,IEOF)
      DO 9 I=1,L
 9    XW(I)=FLOAT(IBUF(I))/XSCAL
      CALL WIND(XW,L,WIN,XW)
C
C  PERFORM FFT CALCULATION
C
      CALL FAST(XW,NFFT)
      JND=IND
 11   IF(JND.GE.1) GO TO 12
      JND=JND+IMD
      GO TO 11
 12   JDX=MOD(JND-1,IMD)
      JX=1
      JFFT=NFFT/2+1
C
C  PUT IN PHASE FACTOR FROM TABLE
C
      DO 10 I=1,JFFT
      XWC(I)=XWC(I)*XM(JX)
      IF(ICJ.EQ.1) XWC(I)=CONJG(XWC(I))
      JX=JX+JDX
      IF(JX.GT.IMD) JX=JX-IMD
 10   CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE: ICEIL
C EVALUATE CEILING FUNCTION
C-----------------------------------------------------------------------
C
      FUNCTION ICEIL(X)
      IS=0
C
C  INUM IS THE BIGGEST POSITIVE INTEGER IN MACHINE MINUS 1
C  FOR 16-BIT MACHINES, INUM IS 32767-1
C
      INUM=32766
      IF(X.GT.0.)IS=INUM
      ICEIL=IFIX(X-FLOAT(IS))+IS
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE: IFLOR
C EVALUATE FLOOR FUNCTION
C-----------------------------------------------------------------------
C
      FUNCTION IFLOR(X)
      IS=0
C
C  INUM IS THE BIGGEST POSITIVE INTEGER IN MACHINE MINUS 1
C  FOR 16-BIT MACHINES, INUM IS 32767-1
C
      INUM=32766
      IF(X.LT.0.)IS=INUM
      IFLOR=IFIX(X+FLOAT(IS))-IS
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE: EUREKA
C LEVINSON RECURSION SOLUTION OF TOEPLITZ EQUATION
C-----------------------------------------------------------------------
C
C  SOURCE OF CODE IS:
C  E. A. ROBINSON, MULTICHANNEL TIME SERIES ANALYSIS WITH
C    COMPUTER PROGRAMS, SECOND EDITION, P 44
C    HOLDEN-DAY, SAN FRANCISCO, CA. 1976
C
C.............................................
C  INPUTS:
C
C     LR=LENGTH OF FILTER=M
C     R=AUTOCORRELATION COEFS=(R0,R1,R2,...,RM)
C     G=RIGHT-HAND SIDE COEFS=(G0,G1,G2,...,GM)
C
C
C  OUTPUTS:
C
C     F=FILTER COEFS=(F0,F1,...,FM)
C     PREDICTION ERROR COEFS=(1,A1,A2...,AM)
C
C.............................................
C
C
C
      SUBROUTINE EUREKA(LR,R,G,F,A)
      DIMENSION R(1),G(1),F(1),A(1)
      V=R(1)
      D=R(2)
      A(1)=1.
      F(1)=G(1)/V
      Q=F(1)*R(2)
      IF(LR.EQ.1)RETURN
C
      DO 4 L=2,LR
      A(L)=-D/V
      IF(L.EQ.2)GO TO 2
      L1=(L-2)/2
      L2=L1+1
      IF(L2.LT.2)GO TO 5
      DO 1 J=2,L2
      HOLD=A(J)
      K=L-J+1
      A(J)=A(J)+A(L)*A(K)
      A(K)=A(K)+A(L)*HOLD
1     CONTINUE
5     CONTINUE
      IF(2*L1.EQ.L-2)GO TO 2
      A(L2+1)=A(L2+1)+A(L)*A(L2+1)
2     CONTINUE
      V=V+A(L)*D
      F(L)=(G(L)-Q)/V
      L3=L-1
      DO 3 J=1,L3
      K=L-J+1
      F(J)=F(J)+F(L)*A(K)
```

```
3       CONTINUE
        IF(L.EQ.LR)RETURN
        D=0
        Q=0
        DO 4 I=1,L
        K=L-I+2
        D=D+A(I)*R(K)
        Q=Q+F(I)*R(K)
4       CONTINUE
        STOP
        END
C
C---------------------------------------------------------------------
C SUBROUTINE: CRECT
C CREATE N POINT RECTANGULAR WINDOW
C---------------------------------------------------------------------
C
        SUBROUTINE CRECT(WIN,N)
        DIMENSION WIN(1)
C
C  WIN=ARRAY TO HOLD WINDOW COEFFICIENTS
C  N=NUMBER OF WINDOW COEFFICIENTS
C
        DO 10 I=1,N
 10     WIN(I)=1.0
        RETURN
        END
C
C---------------------------------------------------------------------
C SUBROUTINE: CHAM
C CREATE N POINT HAMMING WINDOW
C---------------------------------------------------------------------
C
        SUBROUTINE CHAM(WIN,N)
        DIMENSION WIN(1)
C
C  WIN=ARRAY TO HOLD WINDOW COEFFICIENTS
C  N=NUMBER OF WINDOW COEFFICIENTS
C
        PI=4.*ATAN(1.0)
        DO 10 I=1,N
 10     WIN(I)=0.54-0.46*COS((2.*PI*FLOAT(I-1))/FLOAT(N-1))
        RETURN
        END
C
C---------------------------------------------------------------------
C SUBROUTINE:WIND
C WINDOW DATA SEQUENCE
C---------------------------------------------------------------------
C
        SUBROUTINE WIND(X,N,WIN,Y)
        DIMENSION X(1),Y(1),WIN(1)
C
C  X=ARRAY WHICH HOLDS INPUT SEQUENCE
C  N=NUMBER OF POINTS IN ARRAY X
C  WIN=ARRAY WHICH HOLDS WINDOW COEFFICIENTS
C  Y=ARRAY WHICH HOLDS OUTPUT SEQUENCE
C
        DO 10 I=1,N
 10     Y(I)=X(I)*WIN(I)
        RETURN
        END
C
C---------------------------------------------------------------------
C SUBROUTINE: ZERO
C ZERO OUT A FLOATING POINT ARRAY
C---------------------------------------------------------------------
C
        SUBROUTINE ZERO(XAR,N)
        DIMENSION XAR(1)
C
C  XAR=ARRAY TO BE ZEROED OUT
C  N=NUMBER OF POINTS IN ARRAY XAR
C
        DO 10 I=1,N
 10     XAR(I)=0.
        RETURN
        END
C
C---------------------------------------------------------------------
C SUBROUTINE: IZERO
C ZERO OUT A FIXED POINT ARRAY
C---------------------------------------------------------------------
C
        SUBROUTINE IZERO(IAR,N)
        DIMENSION IAR(1)
C
C  IAR=ARRAY TO BE ZEROED OUT
C  N=NUMBER OF POINTS IN ARRAY IAR
C
        DO 10 I=1,N
 10     IAR(I)=0.
        RETURN
        END
C
C---------------------------------------------------------------------
C SUBROUTINE:GNAME
C ***
C *** THIS PROGRAM IS MACHINE DEPENDENT
C *** FORTRAN CODE HAS BEEN SUPPLIED FOR A DATA GENERAL COMPUTER
C *** WITH A FORTRAN 5 COMPILER
C ***
C THIS PROGRAM READS ASCII DATA INTO AN ARRAY "NAME(I)"
C IN A FORMAT THAT MAY BE USED BY : OPEN ICH, NAME
C WHICH OPENS DISK FILE "NAME" ON FORTRAN CHANNEL ICH
C---------------------------------------------------------------------
C
        SUBROUTINE GNAME(NAME)
        DIMENSION NAME(10)
        ITTI=11
C
C  READ UP TO 10 CHARACTERS FROM DEVICE ITTI IN S (STRING) FORMAT
C  THE CHARACTERS ARE PACKED 2 PER 16 BIT WORD AND ARE LEFT
C  JUSTIFIED IN THE ARRAY NAME
C
        READ(ITTI,9999) NAME(1)
 9999   FORMAT(S10)
        RETURN
        END
```

REFERENCES

[1] L. R. Rabiner and Jont B. Allen, "Short-time Fourier analysis techniques for FIR system identification and power spectrum estimation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 182–192, Apr. 1979.

[2] J. B. Allen and L. R. Rabiner, "Unbiased spectral estimation and system identification using short-time spectral analysis methods," *Bell System Tech. J.*, vol. 58, no. 8, pp. 1743–1763, Oct. 1979.

[3] Jont B. Allen and R. Yarlagadda, "The digital summation formula and one application," to be published.

[4] Jont B. Allen, "Short-term spectral analysis, synthesis, and modification by discrete Fourier transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 235–238, June 1977.

[5] Jont B. Allen and L. R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proc. IEEE*, vol. 65, pp. 1558–1564, Nov. 1977.

[6] S. Zohar, "Toeplitz matrix inversion: The algorithm of W. F. Trench," *J. Ass. Comput. Mach.*, vol. 16, pp. 592–601, 1967.

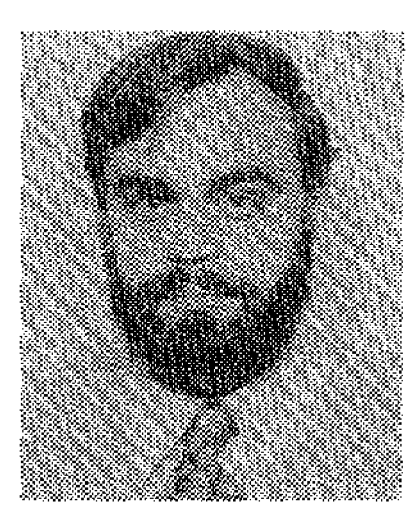
[7] E. A. Robinson, *Multichannel Time Series Analysis with Digital Computer Programs*, second edition only. San Francisco, CA: Holden-Day, 1976, p. 44.

[8] G. D. Bergland and M. T. Dolan, "Fast Fourier transform algorithms," *Programs for Digital Signal Processing*. New York: IEEE Press, 1979, sect. 1.2.

Lawrence R. Rabiner (S'62–M'67–SM'75–F'76) was born in Brooklyn, NY, on September 28, 1943. He received the S.B. and S.M. degrees simultaneously in 1964, and the Ph.D. degree in electrical engineering in 1967, all from the Massachusetts Institute of Technology, Cambridge.

From 1962 through 1964, he participated in the cooperative plan in electrical engineering at Bell Laboratories, Whippany and Murray Hill, NJ. He worked on digital circuitry, military communications problems, and problems in binaural hearing. Presently, he is engaged in research on speech communications and digital signal processing techniques at Bell Laboratories, Murray Hill, NJ. He is coauthor of the books *Theory and Application of Digital Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall, 1975) and *Digital Processing of Speech Signals* (Englewood Cliffs, NJ: Prentice-Hall, 1978).

Dr. Rabiner is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, and a Fellow of the Acoustical Society of America. He is past President of the IEEE Acoustics, Speech, and Signal Processing Society Ad Com, a member of the Acoustics, Speech, and Signal Processing Society Technical Committee on Digital Signal Processing, a member of the Acoustics, Speech, and Signal Processing Society Technical Committee on Speech Communications, a former Associate Editor of the IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, a member of the PROCEEDINGS OF THE IEEE Editorial Board, and a former member of the Technical Committee on Speech Communication of the Acoustical Society of America.



Jont B. Allen (M'76) was born in St. Charles, IL, on December 5, 1942. He received the B.S. degree in electrical engineering from the University of Illinois, Urbana-Champaign, in 1966, and the M.S. and Ph.D. degrees from the University of Pennsylvania, Philadelphia, in 1968 and 1970, respectively.

He joined Bell Laboratories, Holmdel, NJ, in 1970, and transferred to the Acoustics Research Department in Murray Hill, NJ, in 1974. He is presently working in the areas of small room acoustics, dereverberation of speech signals, cochlear modeling, and digital signal processing. His main efforts have been directed toward modeling the cochlea.