An Efficient Algorithm for Measuring the Impulse Response Using Pseudorandom Noise*

JEFFREY BORISH AND JAMES B. ANGELL

Stanford University, Center for Computer Research in Music and Acoustics, Department of Music, Stanford, CA 94305, USA

The impulse response of a linear system can be determined by exciting the system with white noise, and cross-correlating the input and output. As contrasted with the straightforward technique using an impulsive excitation, this approach is capable of providing vastly superior dynamic range. In order to minimize the amount of computation required by the cross-correlation step, the system can be excited by a binary maximal-length sequence, and the cross correlation performed using the fast Hadamard transform. By this means, only additions are required, and the number of additions is approximately $2.5n \log_2 n$, where n is the length of the sequence.

0 INTRODUCTION

In measuring the impulse response of a linear system, the most direct approach is to apply an impulsive excitation to the system and observe the response. There are two basic difficulties with this approach. The first is generating the impulsive excitation, and the second is obtaining adequate dynamic range. If one is dealing with an electronic system, generating an impulse is not a severe problem. But there are settings for which it is difficult, such as measuring the impulse response of a concert hall. Although techniques exist for producing an impulsive acoustical excitation—electronic spark gaps, pistol shots, or exploding balloons are often used—it is difficult to assure that the energy is equally distributed over all frequencies of interest. Because the duration of the impulse, by definition, is very short, it is difficult to deliver enough energy to the system to overcome the noise that is present. The amplitude of the impulse is limited by the range of linearity of the system and its duration by the range of frequencies of interest. This problem is exacerbated by a nonuniform distribution of energy in the impulsive excitation because the linearity limitation to the amplitude is imposed by the frequency range where most of the energy falls. In ranges that are shortchanged, it is not even possible to obtain the dynamic range that is theoretically possible.

A different approach is to excite the system with noise. Because the excitation is applied for a longer period of time, more energy is delivered to the system for a given amplitude of signal, circumventing the dynamic range problem. Further, it is easier to assure the uniformity of the energy distribution over frequency. The response is the convolution of the excitation with the impulse response. The impulse response can be extracted from the measurement by cross-correlating the noise input with the output. This paper will discuss a particularly efficient implementation of this approach on a digital computer.

The technique to be described is based upon a proposal by Schroeder [1], who observed that techniques developed in Hadamard spectroscopy by Nelson and Fredman [2] and Harwit and Sloane [3] could be applied in this problem. Schroeder originally described the use of pseudorandom noise for measuring the impulse response of a concert hall in [4]. In this work he performed the cross-correlation operation in the frequency domain. Because the length of the pseudorandom noise sequence was one less than a power of 2, it was necessary to interpolate a sample in order to exploit the fast Fourier transform. Cabot also reported using a cross-correlation technique based upon analog bucket-brigade delay lines [5]. Basing the measurement technique upon a digital computer offers the greatest potential for maximizing the dynamic range. Also, having the final result in digital form makes additional manipulations more convenient.

* Manuscript received 1982 August 3; revised 1983 March 21.

Because the techniques described in this paper are not widely known in the audio community, this paper will begin with a brief review of the relevant properties of maximal-length sequences which are used as the pseudonoise excitation. After reviewing the cross-correlation operation, we will show how the problem can be manipulated into a form that makes it possible to apply an efficient algorithm known as the fast Hadamard transform. Finally, we will illustrate the application of the technique and discuss some practical issues.

1 MAXIMAL-LENGTH SEQUENCE

The first step of the cross-correlation method is to generate a noiselike test signal to be applied to the linear system. Although several techniques exist for generating noise in a digital computer [2], [3], [6]–[9], binary maximal-length shift register sequences have a number of advantages. The most important is that except for a small dc error, their autocorrelation is a perfect impulse [9], [10] (see Fig. 1). In other words, the spectrum of the pseudo noise is flat everywhere except at direct current. Although a maximal-length sequence is actually a deterministic signal, it shares this desirable property with white noise. Another advantage of maximal-length sequences is that they are very easy to generate, requiring a minimum of execution time on a general-purpose computer, or of hardware in a specialized device. Also, because the sequences are binary, the cross-correlation operation is particularly simple. By assigning the values ± 1 to the two binary levels, it is clear that the cross correlation requires no multiplications, only additions and subtractions. (For simplicity, no distinction will be made in what follows between additions and subtractions in discussing computational requirements because the requirements as measured in execution time or in hardware are so similar.) Multiplication usually requires much more time than addition, so eliminating the need for multiplications greatly speeds the processing. But even more significantly, an efficient algorithm based upon the fast Hadamard transform exists for performing the additions. Like the more familiar fast Fourier transform, the fast Hadamard transform requires on the order of $n \log_2 n$ operations, which in this case are additions. So basing the technique upon excitation by a maximal-length sequence makes it possible to perform the cross correlation expeditiously. A final advantage is that although maximal-length sequences have statistical properties like true white noise, they are actually deterministic signals which can be repeated precisely. Differences in the response of the system to successive measurements can be unambiguously attributed to noise in the system.

Several of the references explain how to generate maximal-length sequences and provide a mathematical framework based upon primitive polynomials [2], [3], [7], [9]. The generation of maximal-length sequences is most easily described by considering a specific case, such as the three-stage shift register shown in Fig. 2. The boxes containing z^{-1} represent a unit-sample delay produced by memory elements or flip-flops. The operation designated by $\oplus$ is a modulo 2 sum, or exclusive-or, defined by

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 1 \oplus 1 &= 0 . \end{aligned} \tag{1}$$

A signal is fed back to the beginning of the shift register which is a modulo 2 sum of selected outputs. In other words, the shift register generates a sequence of 1's and 0's that satisfies the recursion relation

$$\tilde{n}(k+3) = \tilde{n}(k) \oplus \tilde{n}(k+2) . \tag{2}$$

Recursion relations can be specified for any shift register length. The sequences produced at each node of the three-stage shift register in Fig. 2 that are shown in the figure were produced by initializing the shift register to all 1's. Choosing different initial conditions will change the sequences that are produced in a way which corresponds to delaying the sequences by some amount. With m stages in the binary shift register it is theoretically possible to describe 2^m states, but if the content of the shift register is all 0's, it will be impossible for a 1 to occur, and the shift register will remain frozen

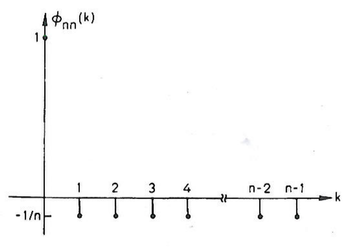


Fig. 1. Autocorrelation function of a maximal-length sequence with length n.

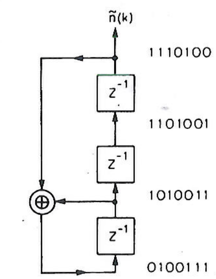


Fig. 2. Binary feedback shift register of length $m = 3$ for generating a maximal-length sequence of length $n = 7$.

in this state. In order to avoid this degenerate case, the longest sequence that can be generated using linear feedback has length $2^m - 1$. A binary sequence whose length is $2^m - 1$ is called a maximal-length sequence.

2 CROSS CORRELATION

In order to drive the linear system under test, the Boolean sequence produced by the shift register must be converted to a signal. It is customary in describing digital circuits to denote the two logic levels as 1 and 0. For signal processing, the sequence $\tilde{n}(k)$ produced by the shift register is mapped to $n(k)$ by changing 1 to -1 and 0 to $+1$. (Any vector or matrix augmented with the tilde refers to the 1, 0 code.) The sequence $n(k)$ is the one that is actually applied to the system. The cross correlation of the input and the output is related to the autocorrelation of the input by a convolution with the impulse response:

$$\phi_{ny}(k) = \phi_{nn}(k) * h(k) . \tag{3}$$

When the input autocorrelation $\phi_{nn}(k) = \delta(k)$, the Dirac delta function, the result of convolving $\phi_{nn}(k)$ with any function is the function itself, in this case, the impulse response. Thus the impulse response can be recovered by cross-correlating the noise input $n(k)$ with the output $y(k)$ [11]. The desirable impulsive autocorrelation of maximal-length sequences arises only under circular autocorrelation, so the indexing of the sequences in any correlation operation must be performed modulo n. Using the notation $((j))_n$ for the residue of j modulo n, or simply $((j))$, where the modulus can be inferred from the context, the cross-correlation operation is defined by

$$\phi_{ny}(k) = \frac{1}{n}\sum_{j=0}^{n-1} n(j)y((j + k)) . \tag{4}$$

By a change of indices, this expression is equivalent to

$$\phi_{ny}(k) = \frac{1}{n}\sum_{j=0}^{n-1} n((j - k))y(j) . \tag{5}$$

The circularity of the operation can also be achieved by performing linear cross correlations with periodic versions of the original sequences defined by

$$x_p(k) = x((k))_n . \tag{6}$$

In other words, each period of the periodic sequence is equal to the original sequence. Eq. (5) can also be described in terms of a matrix multiplication:

$$\Phi_{ny} = \frac{1}{n} N_n Y . \tag{7}$$

Φ_{ny} and Y are vectors whose elements are the $\phi_{ny}(\cdot)$ and $y(\cdot)$ of Eq. (5), and the matrix N_n contains the circularly delayed versions of the sequence $n(\cdot)$. The noise matrix N_n is a right circulant matrix [12] because successive rows are obtained from the previous one by rotating it one position to the right. For the specific case of $m = 3$, the noise matrix is

$$N_7 = \begin{bmatrix} - & - & - & + & - & + & + \\ + & - & - & - & + & - & + \\ + & + & - & - & - & + & - \\ - & + & + & - & - & - & + \\ + & - & + & + & - & - & - \\ - & + & - & + & + & - & - \\ - & - & + & - & + & + & - \end{bmatrix} \tag{8}$$

where the shorthand + and − has been substituted for +1 and −1.

Because the elements of N_n are all ±1, only additions and subtractions are required to perform the matrix multiplication. Finding each element of the correlation vector Φ requires $n - 1$ additions. There are n elements in the result vector, so the total number of additions required is $n(n - 1)$. When n is a large number, the number of operations can become prohibitively large. In architectural acoustics one is interested in measuring the impulse response of concert halls. The duration of the impulse response, as defined by the reverberation time, is often as long as 2–3 s. In order to deal with the entire audio bandwidth of 20 Hz–20 kHz, a sampling rate of at least 40 kHz is required. As a result, the number of samples in the maximal-length sequence could be on the order of 10^5, in which case the total number of operations would be on the order of 10^{10}. Assuming that a computer is capable of performing an addition in 1 μs, several hours of computer time would be required. Clearly, a more efficient algorithm must be found when one is interested in dealing with such long sequences.

3 FAST HADAMARD TRANSFORM

The efficient algorithm for performing the desired cross correlation is based upon the fast Hadamard transform. Like the discrete Fourier transform, the Hadamard transform can be described in terms of a matrix multiplication. The matrix that transforms the input vector is known as the Hadamard matrix H_n, where n gives the number of rows or columns. The elements of the Hadamard matrix are all ±1, and the matrix must satisfy the relation

$$H_n H_n^T = nI_n . \tag{9}$$

The efficient algorithm applies only to the specific class of Hadamard matrices known as Sylvester type. The Sylvester-type Hadamard matrix is defined recursively

by

$$H_1 = [1]$$

$$H_{2i} = \begin{bmatrix} H_i & H_i \\ H_i & -H_i \end{bmatrix} . \tag{10}$$

Only orders 2^k, where k is a nonnegative integer, exist. The Sylvester-type Hadamard matrix of order 8 is

$$H_8 = \begin{bmatrix} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & + & - \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & + & + & + & - & - & - & - \\ + & - & + & - & - & + & - & + \\ + & + & - & - & - & - & + & + \\ + & - & - & + & - & + & + & - \end{bmatrix} . \tag{11}$$

The partitioning emphasizes the structure that evolves from the recursive definition.

The flow graph for an 8-point fast Hadamard transform is shown in Fig. 3. It should be evident that the flow graph is identical to the flow diagram for the fast Fourier transform except that the twiddle factors [13] are all unity, reflecting the fact that no multiplies are required. It should also be noted that the bit-reversal shuffling of the input vector for the decimation in time algorithm or the output sequence for the decimation in frequency algorithm is not required in the fast Hadamard transform. When the bit reversal is performed, the transformation is often called the fast Walsh transform because the transform terms fall in order of increasing sequence [14].

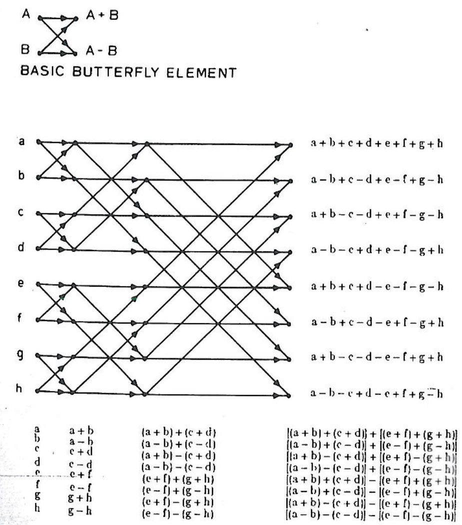


Fig. 3. Flow graph for an 8-point fast Hadamard transform.

4 PERMUTATIONS

4.1 Using the Fast Hadamard Transform to Perform the Cross Correlation

The cross-correlation operation is described in terms of a matrix multiplication in Eq. (7). Unfortunately, N_n is not a Hadamard matrix, so it is not possible to apply the fast Hadamard transform directly. But it is possible to manipulate the problem into the required form by a sequence of matrix multiplications:

$$N_n = P_2 S_2 H_{n+1} S_1 P_1 \tag{12}$$

P_1 and P_2 are permutation matrices whose purpose is to permute the rows and columns of H_{n+1}. Matrices S_1 and S_2 suppress the first row and column of H_{n+1} in order to reduce the $(n + 1) \times (n + 1)$ matrix into one that is only $n \times n$. The reader may confirm this equivalence for the specific case of $n = 7$ by using the matrices

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{13a}$$

$$S_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{13b}$$

$$S_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad (13b)$$

By substituting Eq. (12) into Eq. (7), we obtain the relation

$$\Phi_{ny} = \frac{1}{n} P_2(S_2(H_{n+1}(S_1(P_1 Y)))) \ . \qquad (14)$$

Although the operations can be performed in any order, the sequence indicated by the parentheses leads to the following interpretation. The measurement vector Y is permuted according to P_1 and a 0 element is affixed to the beginning of the vector. The resulting vector, which has the required 2^m terms, is transformed by the fast Hadamard transform algorithm. Then the first element is dropped, and the final result is obtained by permuting the vector according to P_2 and normalizing. This sequence of operations is illustrated for the specific case of a sequence with length 7 in Fig. 4.

4.2 Determining the Permutation Matrices

Once one knows the permutation matrices P_1 and P_2, their validity can always be checked by assuring that Eq. (12) holds. To find the permutations in the first place, we start with the N matrix and work backward

$$Y = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \end{bmatrix} \qquad P_1 Y = \begin{bmatrix} a \\ f \\ b \\ g \\ e \\ d \\ c \end{bmatrix} \qquad S_1 P_1 Y = \begin{bmatrix} 0 \\ a \\ f \\ b \\ g \\ e \\ d \\ c \end{bmatrix}$$

$$H_8 S_1 P_1 Y = \begin{bmatrix} 0+a+f+b+g+e+d+c \\ 0-a+f-b+g-e+d-c \\ 0+a-f-b+g+e-d-c \\ 0-a-f+b+g-e-d+c \\ 0+a+f+b-g-e-d-c \\ 0-a+f-b-g+e-d+c \\ 0+a-f-b-g-e+d+c \\ 0-a-f+b-g+e+d-c \end{bmatrix}$$

$$P_2 S_2 H_8 S_1 P_1 Y = \begin{bmatrix} -a-b-c+d-e+f+g \\ +a-b-c-d+e-f+g \\ +a+b-c-d-e+f-g \\ -a+b+c-d-e-f+g \\ +a-b+c+d-e-f-g \\ -a+b-c+d+e-f-g \\ -a-b+c-d+e+f-g \end{bmatrix} = N_7 Y$$

Fig. 4. Illustration of the efficient algorithm for the specific case of a sequence of length $n = 7$.

to the H matrix. The permutations to be described here are not essentially different from those first presented by Nelson and Fredman [2]. The relation we seek to establish is derived from Eq. (12):

$$G_n = S_2 H_{n+1} S_1 = P_2^{-1} N_n P_1^{-1} \ . \qquad (15)$$

The new matrix G_n is the original H_{n+1} matrix with its first row and column dropped. The procedure to be described will actually produce the inverse of the permutation matrices, but inverting a permutation matrix is accomplished simply by transposing it. The derivation of the permutations is facilitated by dealing with $\tilde{N}$ as well as N and seeking permutation matrices that transform it to $\tilde{G}$. A parallel sequence of operations applied to N will transform it to G. P_1^{-1} describes a permutation of the columns of the N matrix. The columns of $\tilde{N}$ are versions of the same maximal-length sequence obtained by progressively delaying each column. The different versions can also be considered to arise from different initial conditions in the shift register that generates the sequences. Once the initial conditions have been specified, all of the $2^m - 1 = n$ terms of the sequence are predestined. Therefore the identity of each column can be established by using any m consecutive terms of the column. By assigning binary weights to the terms, a number is produced that distinguishes the columns, and so serves as a tag. In computing the tag, the significance of the terms can be defined in either order. The decision about which m consecutive terms to consider and which order of significance to assign is based upon programming convenience. The matrix P_1^{-1} is defined to be the matrix that permutes the columns of N so that the keys are in ascending numerical order. Performing this operation produces the matrix $\tilde{N}' = \tilde{N}P_1^{-1}$. The operations performed so far are illustrated in Fig. 5.

Now we must determine the second permutation. The matrix P_2^{-1} is the matrix that permutes the rows of $\tilde{N}'$ to $\tilde{G}$. In order to derive this permutation, we start by examining H for a way to tag each row in a manner analogous to the procedure used before for rearranging the columns. The nature of the tag is suggested by the recursive definition for the Sylvester-type Hadamard matrix, Eq. (10). This definition indicates that the first entry in each row of H_{2i} is simply the Hadamard matrix of the next lower order, H_i. The second entry is also obtained from H_i either by direct replication or by inversion. We can use a single bit to record which op-

$$\tilde{N} = \begin{array}{c} \begin{array}{ccccccc} 1 & 3 & 7 & 6 & 5 & 2 & 4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \longrightarrow \tilde{N}' = \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Fig. 5. Permutation matrix P_1^{-1} rearranges the columns of $\tilde{N}$ so that the tags are in ascending numerical order.

eration was performed, with 0 indicating direct replication and 1 indicating inversion. Thus the first row of each H_{2i} has a 0 associated with it, and the second row has a 1. By placing the bit in the ith position of a binary word, a word is generated that tags each row. This procedure is illustrated in Fig. 6.

i: 3	2	1									tag: 2	1	0	dec
	0	0	+	+	+	+	+	+	+	+	0	0	0	0
0		1	+	−	+	−	+	−	+	−	0	0	1	1
	1		+	+	−	−	+	+	−	−	0	1	0	2
			+	−	−	+	+	−	−	+	0	1	1	3
			+	+	+	+	−	−	−	−	1	0	0	4
1			+	−	+	−	−	+	−	+	1	0	1	5
			+	+	−	−	−	−	+	+	1	1	0	6
			+	−	−	+	−	+	+	−	1	1	1	7

Fig. 6. Illustration of how the recursive definition of the Sylvester-type Hadamard matrix leads to the development of a tag for each row.

To determine P_2^{-1}, we examine the entries of $\tilde{N}'$ to deduce the corresponding tag. The desired permutation matrix is defined to be the matrix that arranges the tags in ascending numerical order. Examining $\tilde{N}'$ in Fig. 7 shows that the tag can be assembled by collecting the m terms from each row at the positions 2^k, where $k = 0, 1, \ldots, m - 1$. Performing this operation will produce the matrix $\tilde{G}_n$, as expected (Fig. 7). Applying the same permutations to N_n will result in G_n.

$$\tilde{N}' = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{array}{c} 1 \\ 2 \\ 4 \\ 3 \\ 6 \\ 7 \\ 5 \end{array} \longrightarrow \tilde{G}_7 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array}$$

Fig. 7. Permutation matrix P_2^{-1} rearranges the rows of $\tilde{N}'$ so that the tags are in ascending numerical order.

In a practical implementation, the permutations can be generated once in a separate operation from the cross correlation and stored on disk. By this means, repeated cross correlations for the same sequence length do not require that the permutations be generated repeatedly.

5 DEMONSTRATION

The algorithm can be demonstrated by measuring the impulse responses of known systems. Fig. 8 shows the measured impulse responses for a simple delay, and an 8th-order Butterworth low-pass filter. These measurements were obtained using a maximal-length sequence of length $n = 1023$. Aside from a small dc error (see Sec. 6.2) that is imperceptible on the scale of the plots, the impulse responses are exactly correct. One should be aware that if the measurements had been performed using random noise, there would have been some uncertainty in the amplitude of the results that were obtained. The actual impulse responses would only be approached asymptotically as the measurement time increased without bound. The near perfection using pseudorandom noise arises from the fact that the excitation is deterministic. It should perhaps be emphasized that in a practical application there will still be uncertainty in the amplitude due to noise in the system under test. But unlike measurements based upon truly random noise, the excitation itself does not contribute to this randomness.

Because the measurement technique is discrete time, the temporal resolution is limited by the sample period. Consequently there is always a small uncertainty in the group delay. In some cases there is also uncertainty in the amplitude due to the fact that the impulse response is only determined to the number of points in the excitation. However, as long as the system is strictly bandlimited to the Nyquist frequency, the intermediate values are available by interpolation.

6 PRACTICAL CONSIDERATIONS

6.1 Choosing the Length of the Maximal-Length Sequence

When the maximal-length sequence is injected into a linear system, the output of the system is the convolution of the input with the impulse response. The convolution is described mathematically by

$$y(k) = n(k) * h(k) = \sum_{j=0}^{n-1} h(j)n((k - j)) \ . \qquad (16)$$

As mentioned previously, the desirable autocorrelation property of maximal-length sequences arises only under

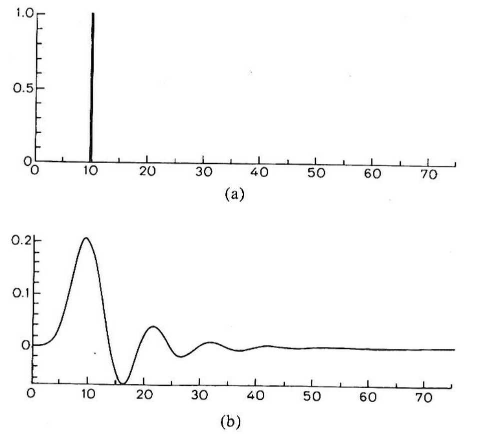


Fig. 8. Computed impulse responses. (a) Simple delay of 10 samples. (b) 8th-order Butterworth low-pass filter.

circular operations. But as a practical matter, it is usually more straightforward to deal with a linear convolution. Fig. 9 gives a pictorial representation of the convolution operation. The periodic pseudo noise excitation is delayed and then weighted by the corresponding term of the impulse response. The final output is obtained by summing all of the partial results. It can be seen in this figure that the effect of circular convolution can be obtained by retaining the data from any period of the output after the first. This illustration also clearly shows the effect of attempting to deconvolve an impulse response that is longer than the input maximal-length sequence. If there were an additional term in the impulse response, $h(7)$, its delayed sequence would overlap the repetition of the first undelayed term. There is no way of resolving the ambiguity, so an error will result. The error will be insignificant, though, if the amplitude of the impulse response has decayed to a small value. Clearly no error results if the length of the maximal-length sequence is longer than the impulse response, but there is a disadvantage in the additional computation that is required. For a finite impulse response (FIR) filter the overlap error can always be avoided by choosing the duration of the pseudo noise excitation to exceed that of the impulse response. But for an infinite impulse response (IIR) filter there will always be some overlap. The severity of the error that results can be reduced by choosing the duration of the pseudo noise sequence to be long enough that the amplitude of the impulse response has decayed to a small value.

6.2 Effect of dc Error in Autocorrelation

As discussed in Eq. (3), the convolution also relates the input autocorrelation to the cross correlation. This relationship is what makes it possible to recover the impulse response by way of the cross correlation when the input autocorrelation is a delta function. Unfortunately the autocorrelation of the maximal-length sequence is not a perfect impulse, as can be seen in Fig. 1. Rather, the autocorrelation is actually

$$\phi_{nn}(k) = \frac{n+1}{n}\,\delta(k) - \frac{1}{n} . \qquad (17)$$

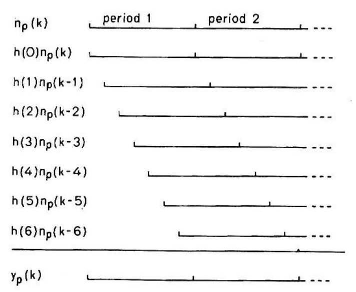


Fig. 9. Pictorial representation of the convolution operation showing the relative timing of the partial results.

Thus the cross correlation has an error that can be found as follows:

$$\phi_{ny}(k) = \sum_{j=0}^{n-1} h(j)\left[\frac{n+1}{n}\,\delta((j-k)) - \frac{1}{n}\right]$$

$$= \frac{n+1}{n}\sum_{j=0}^{n-1} h(j)\delta((j-k)) - \frac{1}{n}\sum_{j=0}^{n-1} h(j)$$

$$= \frac{n+1}{n}\,h(k) - \frac{1}{n}\sum_{j=0}^{n-1} h(j) . \qquad (18)$$

The summation term in Eq. (18), which is the dc component of the actual impulse response, is also the dc error in the computed impulse response. We can calculate the error in the computed impulse response by adding together the terms of the cross correlation:

$$\sum_{k=0}^{n-1} \phi_{ny}(k) = \frac{n+1}{n}\sum_{k=0}^{n-1} h(k) - \sum_{j=0}^{n-1} h(j)$$

$$= \frac{1}{n}\sum_{k=0}^{n-1} h(k) . \qquad (19)$$

The impulse response can be found exactly by correcting the cross correlation:

$$h(k) = \frac{n}{n+1}\left[\phi_{ny}(k) + \sum_{j=0}^{n-1} \phi_{ny}(j)\right] . \qquad (20)$$

In practice it is rarely necessary to perform this correction. Most practical systems in audio have no transmittance at direct current, so that the error term in Eq. (18) is nil. As long as the impulse response does not have a strong dc component, the error term will be negligible.

6.3 Additional Improvement of Dynamic Range

When measuring the impulse response of a linear system using an impulsive excitation, a technique that is often used to improve the signal-to-noise ratio is to average together the response to a number of impulses [15]. The desired response adds coherently, but the noise adds incoherently, providing an improvement in the signal-to-noise ratio of $\sqrt{N}$, where N is the number of periods being averaged together. This procedure has the disadvantage of consuming considerable time due to the fact that one must wait until the system has returned to its quiescent state before initiating the next measurement. If the impulse response has a duration of T seconds, a total of NT seconds will be required for the measurement. The cross-correlation method starts with a tremendous advantage in dynamic range. The advantage is exactly the same as that which would be obtained by averaging together n impulse responses, but the measurement only requires time T to perform. However, in extremely noisy environments it would be possible to combine the averaging technique with the cross-correlation method in order to obtain an additional improvement in dynamic range. Schroeder reports [4] that this technique has been used to measure the impulse response of a lecture hall while the hall was being used by exciting the hall with a quiet pseudo noise signal. It might be possible to apply the same concept to measuring the response of a concert hall during a performance while an audience is present. This approach would circumvent the difficult problem of relating the behavior of the empty hall to its response with an audience present.

6.4 Estimate of Computational Requirements

It is more difficult to develop a convenient rule of thumb for the number of computations required by the fast Hadamard transform than for the fast Fourier transform. In analyzing the fast Fourier transform it is customary to consider only the number of multiplications, because the multiplications usually dominate the computations. However, the elemental operation in the fast Hadamard transform is addition, which will not dominate the other computations that are required. Each stage of the fast Hadamard transform requires n additions for the butterfly operation. Since there are $\log_2 n$ stages, the total number of additions required by the butterfly to evaluate an n-point fast Hadamard transform is $n\log_2 n$. However, in addition to the butterfly, it is also necessary to compute the indices of the two terms being applied to the butterfly—two more additions—and to test for the completion of a loop. The compare in the loop completion test is basically a subtraction, so five additions are required for every two points, increasing the number of additions per stage from n to $2.5n$. Disregarding assignments, the best estimate for the number of operations required is $2.5n\log_2 n$. Note that in the example we considered previously of a 3-s sequence sampled at 40 kHz, the number of additions is reduced from approximately 10^{10} to about 5×10^6, a decrease of more than 10^3. Assuming the same computation rate as before of 10^6 additions per second, the time required to compute the impulse response will be reduced to the much more manageable figure of about 6 s.

6.5 Calculating the Frequency Response

One way to obtain the frequency response of the system under test is to apply the method described in this paper to determine the impulse response, and then to transform the impulse response to the frequency domain using a discrete Fourier transform. But if one is only interested in the frequency response, then another approach worth considering is to perform the computations in the frequency domain. It is well known that the correlation of two sequences can be related to a convolution:

$$\phi_{ny}(k) = n(-k) * y(k) . \qquad (21)$$

The z transform of the convolution of two sequences is the product of their z transforms:

$$n(k) * y(k) \leftrightarrow N(z)Y(z) . \qquad (22)$$

We also have the relation

$$x(-k) \leftrightarrow X\left(\frac{1}{z}\right) . \qquad (23)$$

Combining Eqs. (21)–(23) we obtain

$$\phi_{ny}(k) \leftrightarrow N\left(\frac{1}{z}\right)Y(z) . \qquad (24)$$

Because we are interested in the frequency response, we substitute $z = e^{j\omega}$ to obtain finally

$$\phi_{ny}(k) \leftrightarrow N(e^{-j\omega})Y(e^{j\omega}) = N^*(e^{j\omega})Y(e^{j\omega}) . \qquad (25)$$

In practice, the Fourier transform would be computed as a discrete Fourier transform. All of the terms of $N^*(e^{j\omega})$ can be calculated beforehand and stored. To find the frequency response one need only transform the measurement $y(k)$ and multiply its transform term by term with the stored values of N^*. Note that although the magnitude of $N^*(e^{j\omega})$ will be constant (except for the dc term), the phase will vary, so the multiplications will not be trivial.

As far as the amount of computation is concerned, the transformation of $y(k)$ required by the frequency domain approach is equivalent to the transformation of the impulse response in the time domain approach. In addition, the frequency domain approach requires n complex multiplications ($4n$ real multiplications), and the time domain approach requires $2.5n\log_2 n$ real additions. To compare the amount of time required for the two approaches, we can suppose that multiplication takes 10 times as long as addition, and find the value of n at which the computation times are equivalent:

$$2.5n\log_2 n = 10 \times 4 \times n$$

$$\log_2 n = 16 \rightarrow n = 2^{16} . \qquad (26)$$

For $n > 2^{16}$ the frequency domain approach will require less time, and for $n < 2^{16}$ the time domain approach is faster. If one is interested in the impulse response, it would also be possible to find the frequency response using the frequency domain approach and inverse Fourier transforming the result. Clearly, this approach will require a great deal more computation than the algorithm presented in this paper that finds the impulse response directly in the time domain.

7 SUGGESTIONS FOR ADDITIONAL WORK

The length of the impulse response that can be recovered by cross correlation is limited by the amount of available main memory. Two arrays must be maintained in core simultaneously, each as large as the maximal-length sequence. For acoustical measurements it is often of interest to measure impulse responses as long as 2–3 s. In order to deal with the entire audio spectrum up to 20 kHz, the sampling rate must be at least 40 kHz. Accordingly, each array could be as long as 120 000 samples, requiring that 240 000 samples be stored in core. In computer systems which do not have such prodigious memory capacity, or in cases when one is interested in measuring longer impulse responses, or when the sampling rate must be considerably higher, the resulting abundance of samples might overwhelm the resources available. In such cases it would be desirable to modify the algorithm to reduce the amount of core storage required. There is a fairly straightforward way to halve the memory requirements. The only reason that two arrays are required is to allow the permutation to transfer data from the unpermuted array to the permuted array. The permutation information is retrieved from disk in small segments so as not to deplete further the memory capacity. The memory requirements could be halved by performing the permutation in place. In-place permutation is possible when the permutation matrix is symmetrical, which unfortunately is not the case here. However, it is always possible to factor a permutation matrix into the product of two symmetric permutation matrices [16]. For example, the permutation matrices in Eq. (13) can be factored as follows:

$$P_1 = \begin{bmatrix} 1&0&0&0&0&0&0\\ 0&0&0&0&0&1&0\\ 0&1&0&0&0&0&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&1&0&0\\ 0&0&0&1&0&0&0\\ 0&0&1&0&0&0&0 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0 \end{bmatrix} \begin{bmatrix} 1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&1&0&0\\ 0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0 \end{bmatrix} \tag{27a}$$

$$P_2 = \begin{bmatrix} 1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&1&0&0 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&1&0&0&0&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0\\ 0&0&0&0&1&0&0 \end{bmatrix} \begin{bmatrix} 1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&1&0 \end{bmatrix} \tag{27b}$$

The penalty for halving the memory requirements by this technique is that the permutation must be performed in two passes.

Beyond this simple modification, it becomes more difficult to reduce the amount of memory required. Algorithms have been developed for performing a fast Fourier transform off disk [17], and it should be straightforward to adapt these to performing the fast Hadamard transform in the same manner. But it will also be necessary to devise a means of performing the permutation off the disk in a reasonably efficient manner. These modifications would make the technique completely general, being able to deal with any quantity of data.

It is also intriguing to consider hardware implementation of this cross-correlation technique. Because of the triviality of the arithmetic operations required, the hardware would be much more straightforward than for the fast Fourier transform. It is completely within reason to imagine real-time operation of the algorithm using the appropriate hardware.

8 CONCLUSIONS

Measuring the impulse response using a noiselike excitation is capable of providing much greater dynamic range than can be obtained using an impulsive excitation. When the noiselike excitation is chosen to be a binary maximal-length shift register sequence, several other advantages accrue. Foremost among these is that the cross correlation can be performed very efficiently. Because the excitation is binary, only additions are required. Furthermore, an efficient algorithm based upon the fast Hadamard transform exists for minimizing the number of additions. A straightforward cross-correlation technique would require the evaluation of approximately n^2 multiplications. But this efficient algorithm requires only about $2.5n\log_2 n$ additions, which represents a reduction in the execution time of several orders of magnitude for most values of n. There are also advantages to using maximal-length sequences due to the fact that they are deterministic signals. Measurements are exactly repeatable so that additional improvement in the dynamic range can be obtained by averaging together the responses to several measurements.

9 ACKNOWLEDGMENT

The authors wish to thank Dr. M. R. Schroeder for originally suggesting this technique. Dr. M. L. Fredman kindly provided copies of his computer programs. Thanks also to all of our colleagues at CCRMA for providing a stimulating research environment, especially J. Chowning, K. Sheeline, J. Gordon, and J. Smith; finally to Drs. E. D. Schubert and A. J. Rockmore for all of their support and advice.

10 REFERENCES

[1] M. R. Schroeder, private communication.

[2] E. D. Nelson and M. L. Fredman, "Hadamard Spectroscopy," *J. Opt. Soc. Am.*, vol. 60, pp. 1664–1669 (1970 Dec.).

[3] M. Harwit and N. J. A. Sloane, *Hadamard Transform Optics* (Academic Press, New York, 1979).

[4] M. R. Schroeder, "Integrated-Impulse Method Measuring Sound Decay Without Using Impulses," *J. Acoust. Soc. Am.*, vol. 66, pp. 497–500 (1979 Aug.).

[5] R. C. Cabot, "Acoustic Applications of Cross Correlation," presented at the 64th Convention of the Audio Engineering Society, *J. Audio Eng. Soc. (Abstracts)*, vol. 27, p. 1020 (1979 Dec.), preprint 1544.

[6] D. E. Knuth, *The Art of Computer Programming*, vol. 2 (Addison-Wesley, Reading, MA, 1981).

[7] W. D. T. Davies, "Generation and Properties of Maximum-Length Sequences," *Control*, pp. 302–304, 364–365, 431–433 (1966 June, July, Aug.).

[8] C. M. Rader, L. R. Rabiner, and R. W. Schafer, "A Fast Method of Generating Digital Random Numbers," *Bell Sys. Tech. J.*, vol. 49, pp. 2303–2310 (1970 Nov.).

[9] F. J. MacWilliams and N. J. A. Sloane, "Pseudo-Random Sequences and Arrays," *Proc. IEEE*, vol. 64, pp. 1715–1729 (1976 Dec.).

[10] D. V. Sarwate and M. B. Pursley, "Crosscorrelation Properties of Pseudorandom and Related Sequences," *Proc. IEEE*, vol. 68, pp. 593–619 (1980 May).

[11] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1975), p. 394.

[12] R. M. Gray, "Toeplitz and Circulant Matrices: II," Information Systems Laboratory, Stanford University, Tech. Rep. 6504-1, 1977 Apr.

[13] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1974), p. 363.

[14] H. F. Harmuth, *Transmission of Information by Orthogonal Functions* (Springer-Verlag, New York, 1969).

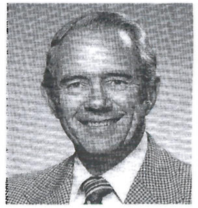
[15] J. M. Berman and L. R. Fincham, "The Application of Digital Techniques to the Measurement of Loudspeakers," *J. Audio Eng. Soc.*, vol. 25, pp. 370–384 (1977 June).

[16] D. Fraser, "Array Permutation by Index-Digit Permutation," *J. ACM*, vol. 23, pp. 298–309 (1976 Apr.).

[17] D. Fraser, "Optimized Mass Storage FFT Program," in *Programs for Digital Signal Processing* (IEEE Press, New York, 1979).

THE AUTHORS

J. Borish



J. Angell

Jeffrey Borish was born in New Jersey in 1952. He received an S.B. degree from M.I.T. in 1974 and an M.S. degree from Stanford University in 1975, both in electrical engineering. He joined Fairchild Semiconductor in 1975 where he worked in the consumer group on the design of audio integrated circuits and other products for the consumer electronics industry. In 1976 he went to Sound Technology where he was responsible for the development of several digital signal processors. He returned to Stanford University in 1980, and is now a candidate for a Ph.D. degree in electrical engineering.

Concurrent with his academic work, Mr. Borish is cofounder of a company that specializes in the application of digital technology to audio problems. He has also served as consultant to firms in the hi-fi industry.

An avid music buff, Mr. Borish is an amateur musician, piano player, and concertgoer.

Mr. Borish is a member of the Audio Engineering Society, the IEEE, the Acoustical Society of America, Tau Beta Pi, and Eta Kappa Nu.

James B. Angell was born in Staten Island, N.Y., in 1924. He received S.B. and S.M. degrees in 1946 and an Sc.D. in 1952 in electrical engineering, all from M.I.T. From 1946 to 1951 he worked at the Research Laboratory of Electronics at M.I.T., studying noise in tracking radars. From 1951 to 1960 he was with the research division of Philco Corporation where