Behavior Research Methods, Instruments, & Computers
1998, 30 (1), 54-67

Digital filtering in EEG/ERP analysis: Some technical and empirical comparisons

JACK B. NITSCHKE and GREGORY A. MILLER
University of Illinois at Urbana-Champaign, Urbana, Illinois

and

EDWIN W. COOK III
University of Alabama, Birmingham, Alabama

A general approach to time domain digital filtering is described, and examples of some filters used in EEG/ERP research are presented. Simulations are reported that evaluate the impact of the relative length of the filter weight series and the signal cycle to be filtered, the span and real-time density of the filter weights, and slow drift across the epoch being filtered. Results indicate that some filters commonly used in the EEG/ERP literature are inadequate. Frequency domain digital filtering is also briefly discussed. The fast Hartley transform, a fast but relatively unknown computational method for frequency domain filtering of ERP/EEG data, is introduced and compared with time domain filtering. Some practical recommendations are provided.

The analysis of encephalographic (EEG) data, either ongoing activity or event-related potentials (ERPs), generally requires the extraction of a signal of interest from a noisy background. This filtering process may be carried out in a variety of different ways. The term *digital filter* refers to a wide range of techniques that have in common the fact that they are mathematical procedures applied to discrete, numeric representations of continuous waveforms to emphasize or attenuate certain frequencies.

Digitally filtering an EEG waveform in the time domain[1] typically involves cross-multiplying each unfiltered data point and its neighbors with a set of weights. In effect, the weights represent a copy of the signal pattern of interest. The cross-multiplication process is repeated for each point to be filtered. The sums of these cross-products, arranged as a series, constitute the filtered waveform. An intuitive appreciation of how such a procedure can accomplish frequency-specific filtering can be gained by considering a set of weights with magnitudes forming a sine wave of a particular frequency. When data points are cross-multiplied with these weights, the sum of the cross-products will be largest when the data predominantly consist of a sine wave of the same frequency and are in phase[2] with the weights, such that the two sets of values rise and fall in synchrony. The sum of the cross-products will be most negative when the data have the same frequency but are inverted (180° out of phase). Any other pattern in the data will produce a sum closer to zero. In effect, the filter is "tuned" to detect a sinusoidal signal of a certain frequency, and signals that are nonsinusoidal or do not match the filter's frequency will not produce as much output.

Digital filters of this type are pervasive in psychophysiology. In designing an appropriate filter for a given EEG/ERP application, the investigator faces a choice among a variety of methods for generating the weights used in the cross-multiplication function and also a choice of the set of adjacent time points to which to apply the weights. These choices affect the computational speed of the filter, how much of the desired signal is preserved, and what kind of noise is attenuated. In aggregate, these choices involve a variety of subtle factors and tradeoffs. Unfortunately, it appears that digital filters are more widely used than understood, with the result that inferior and often inappropriate filters are used.

To facilitate appropriate use of digital filtering in the EEG/ERP literature, this paper will provide a general approach to conceptualizing time domain filter methods, followed by some empirical investigations of the limits of such methods. The presentation will emphasize accessibility rather than completeness (see Cook & Miller, 1992; Ruchkin, 1988; and especially Glaser & Ruchkin, 1976, for more technical discussions). Time domain filtering will be distinguished from frequency domain filtering. Basic concepts of frequency analysis will be presented, and a relatively new computational approach to frequency domain filtering that is largely unknown in the EEG/ERP literature will be introduced. Finally, some comparisons and recommendations involving these methods will be offered.

Portions of the introductory material in this paper are based on a more general tutorial on filtering (Cook & Miller, 1992). J.B.N. was supported in part by NIMH Grant T32 MH14257 to Lawrence Jones, and this project was supported by NIMH Grant R01 MH39628 to G.A.M. and by the Departments of Psychology and Psychiatry and the Beckman Institute of the University of Illinois. We gratefully acknowledge the contributions of Richard Davidson, Blair Hicks, Brandy Isaacks, James Long, Bruce Wheeler, and especially Joseph Senulis. Correspondence concerning this article should be addressed to G. A. Miller, Department of Psychology, University of Illinois, 603 E. Daniel Street, Champaign, IL 61820 (e-mail: gamiller@uiuc.edu).

DIGITAL FILTERING IN THE TIME DOMAIN

Concepts and Terminology

We will introduce a formal representation of a common filtering method and then discuss some examples. EEG voltage varies continuously over time and may be digitized at equal intervals, yielding a *time series* of n observations of the form

$$X_t, X_{t+p}, X_{t+2p}, X_{t+3p}, \cdots, X_{t+(n-1)p}.$$

The subscripts refer to the time at which the associated data point X is observed, such that t is the time at which recording began and p is the *sampling period* (the time between adjacent samples). Such a representation of the data is said to be "in the time domain." The present focus is on a very common type of digital filter, in which each filtered point is computed by using the corresponding unfiltered point and an equal number of unfiltered points before and after the point to be filtered. Specifically, each output value is computed as the sum of the cross-products of the weights and a symmetrical set of adjacent input data points, as follows:

$$Y_t = \sum_{i=-j}^{j} W_i * X_{t+i},$$

where W is the series of $2j+1$ weights (subscripted $-j$ to $+j$), X is the input time series, Y is the filtered time series, and the subscript t refers to each time point in the input series.

This cross-multiplication and summing procedure, when carried out across a range of values of t, is referred to as *convolution*. The $2j+1$ weights are symmetric (i.e., $W_i = W_{-i}$) about an unpaired center weight, W_0. Such filters have a *finite impulse response* (FIR), a term referring to a filter's response to a perturbation in an otherwise consistent input function. Filters that define output points solely on the basis of *input* points are said to have a finite impulse response, because the arithmetic effect of a single aberrant input point (an "impulse") is confined to the $2j$ points adjacent to it. As a result, the perturbation disappears after a finite amount of time (after the last filtered point that includes the aberrant unfiltered point in its computation—the jth point). In contrast, filters that define each filtered point in part on the basis of prior *filtered* points have an *infinite impulse response* (IIR), because the arithmetic effect of a single aberrant point will continue to some degree into all subsequent points. These two kinds of filters are also sometimes referred to as "nonrecursive" and "recursive," respectively (see Cook & Miller, 1992, for discussion of FIR and IIR filters).

A variety of FIR filters have been described in the EEG/ERP literature, with the most common use being to smooth a time series (i.e., remove high-frequency components). Smoothing is most often accomplished by averaging a symmetric set of points before and after each point being filtered. In an average of N points, each point is weighted by $1/N$. Because the N points generally cover only a small portion of the total time series, fast but not slow changes are removed by the filter, which thus serves a smoothing function. This filter is variously referred to as a *moving-average* filter, reflecting the fact that computation of the average is repeated to define each filtered point; as an *equal-weight* filter, reflecting the fact that the weights are identical; or as a *boxcar* filter, reflecting the shape of a plot of the weights. Boxcar filters vary only in the number of data points averaged together, since that dictates the value of the weights ($1/N$).

The *gain function* of a filter describes its gain (ratio of output to input) as a function of frequency. Thus, the gain functions illustrated in Figure 1 show the proportion of the input signal at a given frequency that is available in the output of the filter. A gain of 0.0 implies complete attenuation at a given frequency, whereas a gain of 1.0 implies no attenuation. In some circumstances, gain can exceed 1.0 (in effect, the filter functions as an amplifier at that frequency) or fall below 0.0 (an inverter). Gain functions for useful digital filters typically share several features: The *pass band* is the frequency range where the gain function is close to 1.0, the *stop band* is the frequency range where the gain function is close to 0.0, and the *transition band* is the frequency range where the gain is intermediate. The x-axis of a gain function for a digital filter ranges from 0 Hz (dc) to one half the sampling frequency (i.e., $f_s/2$, called the Nyquist frequency).

For a boxcar filter, the gain function is solely a function of the number of weights and the sample frequency. Gain functions for some boxcar filters illustrated in Fig-

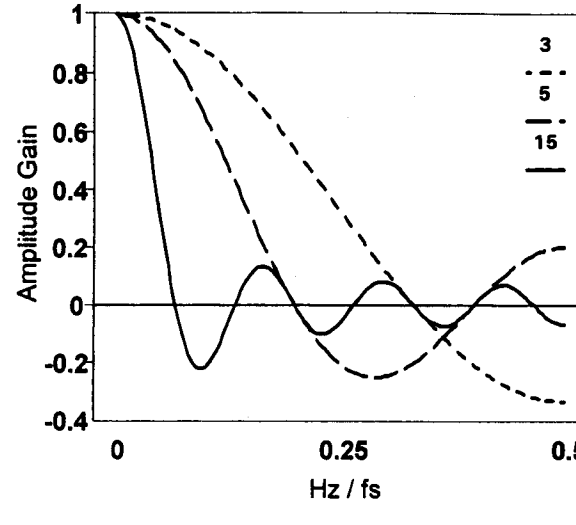

Figure 1. Amplitude gain functions for boxcar filters. f_s is the sampling frequency. The different line types illustrate filters differing in number of weights. The highest possible value for f_s and hence for the y-axis is the Nyquist frequency, which is half the sampling frequency. Note the considerable undershoot (negative gain) and ringing in the stop band.

ure 1 indicate several limitations of such filters. Unless a considerable number of weights is used (more than is typical in the ERP literature; see Farwell, Martinerie, Bashore, Rapp, & Goddard, 1993), the gain function shows very poor frequency precision. This is apparent in the breadth of the transition band and the ripple in the stop band. In fact, the signal can be inverted (gain < 0.0) by as much as 33% for a 3-weight filter and still as much as 22% even for a 15-weight filter. The half-amplitude cutoff frequency, f_c, for a boxcar filter of length $2j+1$ weights, applied to data digitized at a sampling frequency of f_s, is approximately $.61 * f_s/j$. This implies that j and f_c cannot be specified independently. Furthermore, because j must be an integer, available cutoff frequencies are quite limited for a given f_s.

Advantages of boxcar filters include intuitive simplicity and speed of computation. The speed consideration may be particularly important if the filter is to be implemented on line. These advantages must be weighed against the boxcar filter's limited range of parametric control (e.g., available cutoff frequencies), its very poor frequency precision, and its potential for introducing apparent phase shifts. Cook and Miller (1992), Farwell et al. (1993), Ruchkin (1988), and Ruchkin and Glaser (1978) have provided further discussion of boxcar filters. Farwell et al. make a particularly convincing case against the use of boxcar filters in ERP research.

In recent years, FIR digital filters have increasingly used sets of unequal weights. In exchange for the greater computational demands of these more complex FIR filters, they generally provide far superior frequency precision and greater control over other filter characteristics. Frequency precision is limited only by the number of weights (and thus time points) employed. Multiple pass or stop bands can be specified, which may be of use if several sources of narrow-band activity are of interest (e.g., different EEG frequency bands) or are to be removed as noise.

Any filtered time series can be subtracted from the original time series, yielding a second filtered time series. This implements the complement of the original filter. For example, a filter that *passes* frequencies only very close to 50 or 60 Hz can be used to *remove* 50 or 60 Hz via subtraction. This procedure can be applied with any low-pass filter (including a boxcar filter) in order to effectively convert it to a high-pass filter, and vice versa.

Given this broad view of FIR filters (see also Coles, Gratton, Kramer, & Miller, 1986; Cook & Miller, 1992; Donchin & Heffley, 1978), it becomes clear how pervasive they are in the EEG/ERP literature. Computation of the mean of a time series may be construed as the application of a boxcar filter, attenuating all frequencies except 0 Hz (dc). One can also conceptualize an area measure in the scoring of an ERP component as a boxcar filter, in which the number of weights is equal to the number of data points in the scoring window. Computation of the variance of a time series removes the dc component while retaining (and combining) all other (ac) frequencies. This follows from the fact that computation of the variance involves the subtraction of the mean. Similarly, the computation of the root mean square (RMS) value of an ac signal is equivalent to the standard deviation of the time series, since the mean (not explicitly subtracted in the RMS procedure) of an ac signal, in general, is 0. The most common scoring method in the ERP literature is the detection of a maximum or minimum value in a time window. This peak-picking can be understood as the application of a filter with weights (..., 0, 0, 0, 1, 0, 0, 0, ...). This filter is applied repeatedly, at each sample point within the scoring window, and the sample producing the largest convolution is selected as the score. As a high-frequency noise-reduction strategy, occasionally one chooses to remove an errant point and replace it with the average of its immediate neighbors. Such a procedure can be considered the application of a filter consisting of the weights (.5, 0, .5) applied around the errant point (although Glaser & Ruchkin [1976] warn against the unusual gain function of such a filter). Even the ubiquitous prestimulus baseline computation is, in effect, a boxcar filter.

Particularly interesting are FIR filtering methods used in ERP template-matching algorithms. The template can be seen as a set of weights that reflects a specific hypothesis about the shape of the signal being sought. For example, if the template (the set of weights) is simply the positive half cycle of a 2-Hz sine wave, then cross-multiplication of that template with raw EEG will constitute a means of filtering for the P300 component of the ERP on a single-trial basis (see, e.g., Ford, White, Lim, & Pfefferbaum, 1994). One might search EOG or EEG data for a blink by establishing a filter template whose weights approximate the shape of a blink (e.g., Gratton, Coles, & Donchin, 1983; Miller, Gratton, & Yee, 1988). The Woody (1967) filter technique used for latency correction of ERPs uses as its template a portion of the precorrection waveform for a given subject (i.e., the template weight function is customized for each subject). In all of these examples, one slides the template along the time series of data, cross-multiplies and sums, and notes the latency of maximum value as the likely latency of the signal for which one is filtering.

A somewhat different use of the weight series in time domain frequency filtering is seen in the autocorrelation function. In autocorrelation, a time series of N points is systematically correlated with a copy of the N points at various sample-point lags. At zero lag, the correlation is 1.0. The lag is systematically increased, with the correlation computed at each lag for up to $N-2$ lags. The resulting set of correlations can be plotted as a function of lag, as is shown in Figure 2.

The autocorrelation function filters by highlighting temporally recurrent information and deemphasizing other information in a time series. In the case of a pure sine wave signal (top panel of Figure 2), the autocorrelation function duplicates the original sine wave per-

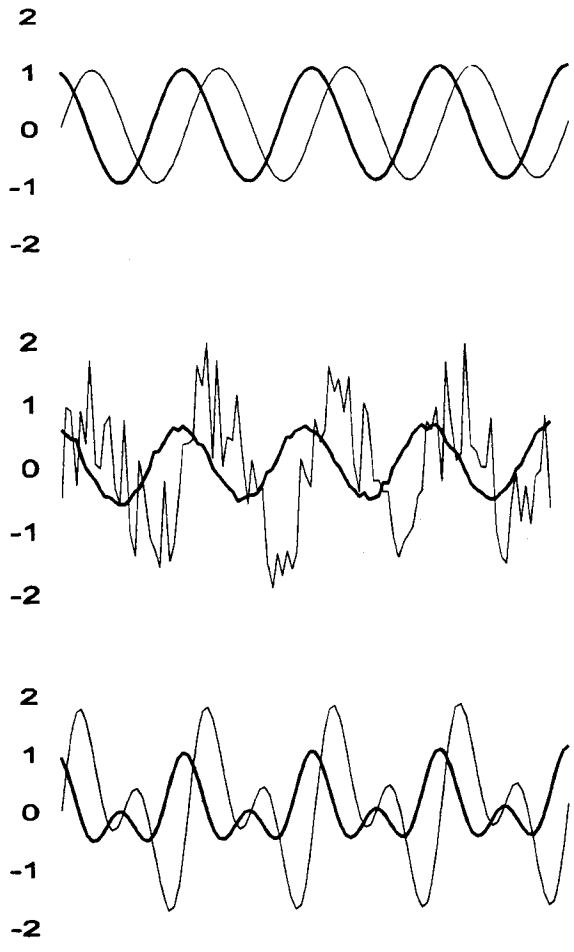

Figure 2. Examples of the autocorrelation function. The *x*-axis represents real time for the signal (thin lines) and real-time lag for the corresponding autocorrelation function (thick lines). The *y*-axis is in arbitrary units for the signal and correlation value for the autocorrelation function. Top panel: The signal is a noiseless, pure sine wave. The autocorrelation reproduces the original waveform exactly. There is an apparent difference between the signal and the autocorrelation only because the signal starts at zero amplitude. At zero lag the autocorrelation is +1.0. With growing nonzero lags, the correlation drops until a lag of 180°, when the correlation is −1.0. At a lag equal to one full cycle, the correlation is again +1.0. If the signal were a noiseless, pure cosine wave, there would be no lag between the raw data and the autocorrelation function, because both functions would start at +1.0. Middle panel: The signal is the same pure sine wave plus random noise. The autocorrelation function nevertheless again reproduces the dominant frequency and its phase quite well, although the maximum correlation is attenuated because of the noise added to the sine wave. Bottom panel: The signal is the same pure sine wave plus a second pure sine wave signal at twice the frequency of the first. Note the double peaks in the autocorrelation function, revealing the periods of the two signals. The range of correlation values is again attenuated relative to the first case, however. The maxima and minima will vary with the relative amplitude of, and the phase relationship between, the two sine waves, although the double peaks will always be apparent. In this example, the two signals began in phase, so they are in phase whenever the slower sine wave begins a new cycle. Thus, the autocorrelation reaches +1.0 at those times and never reaches −1.0.

fectly. When the signal-to-noise ratio is moderate, the autocorrelation function is fairly robust to random noise (middle panel). Finally, the autocorrelation can reveal multiple signal frequencies (bottom panel), although this is less useful as the number of frequencies increases. A cross-correlation function is generated similarly, except that distinct sets of points X and Y (e.g., from two channels) are correlated at various lags. This approach could be especially useful, for example, in determining latency differences between ERP component peaks at homologous sites over different hemispheres. Autocorrelation and cross-correlation procedures require no a priori model of a desired signal. They simply reveal whatever substantial periodic signals may be present.

Autocorrelation and cross-correlation procedures illustrate the breadth of means by which the investigator can obtain the set of weights for a digital filter. Although they obtain their weights rather differently from the other examples presented here, all of these methods have in common a simple summing-of-cross-products computation involving two time series. The general point is that such digital filters are pervasive in the EEG/ERP literature. Here are four examples that illustrate such filters used with EEG and ERP data.

In a standard ERP study, one often wants to identify components that are roughly half sinusoids and quantify their peak amplitude and the latency of that peak. The phase distortion that a conventional analog filter introduces might affect the latency measurement. Setting the analog low-pass filter too high is problematic, because searching for the maximal value in a latency window would then risk capitalizing on small, chance fluctuations (noise) in the channel. Hence, it is useful to smooth the data digitally before scoring. One must estimate the frequency characteristics of the component(s) of interest and select a filter that has either a narrow transition band or a cutoff frequency well above those frequencies. Giese-Davis, Miller, and Knight (1993) expected the main ERP components of interest to be below 5 Hz and wished to remove alpha-band information (around 10 Hz) prior to scoring. A low-pass filter with a half-amplitude cutoff of 5 Hz would require a moderately narrow transition band, in order to pass 0 Hz and still remove alpha. A non-boxcar 31-weight filter proved adequate for data digitized at 125 Hz, with amplitude gains of 96% at 0 Hz, 87% at 2 Hz, and just 2% at 10 Hz.

In contrast, in looking at alpha activity in baseline EEG digitized at 125 Hz, Etienne, Deldin, Giese-Davis, and Miller (1990) found that a non-boxcar 31-weight filter constructed to pass just 8–13 Hz (half-amplitude cutoffs) was much less effective. The filter passed only 61% of the desired activity at 10 Hz, yet it passed 25% of the undesired activity at 6 and 16 Hz. The large attenuation at 10 Hz and the considerable leakage outside the desired pass band were essentially due to 10 Hz being relatively close to both of the cutoff frequencies. Very narrow transition bands, requiring many weights, are necessary in

such a case. These first two examples are not as similar as they may seem. Although both involved 125-Hz sampling and 31-weight filters that could be described as having a pass band that is 5 Hz wide, the 8–13 Hz bandpass filter is much more exacting, because half of the 8-Hz amplitude must be removed. In contrast, for the 0–5 Hz low-pass filter, all of the 0-Hz activity is to be passed. For more success in separating 8–13 Hz activity in EEG digitized at 250 Hz, Heller, Nitschke, Etienne, and Miller (1997) employed a non-boxcar 501-weight filter that provided a virtually perfect gain function.

A quite different case is the measurement of very slow phenomena. To quantify the contingent negative variation (CNV), Yee and Miller (1988) employed, in effect, a boxcar filter to remove conventional (faster) EEG activity, averaging together the last 250 msec of EEG to score the CNV. Such a case in which signal and noise have very different frequencies permits a wide transition band, and one can benefit from the computational speed of a boxcar filter.

Practical Considerations in Time Domain Digital Filtering

The foregoing discussion has provided some definitions and some computational and intuitive perspectives on basic time domain digital filters (see also Cook & Miller, 1992, and Donchin & Heffley, 1978). Also worth consideration are some practical issues. For this paper, we undertook some empirical comparisons in order to address three issues: the signal cycle length to be filtered, the span and real-time density of the weights, and the impact of a linear trend (slow drift). For all of these comparisons, we digitally generated data with specified frequency and amplitude characteristics and then subjected them to digital filters varying along various dimensions of interest. The accuracy of the filter was judged on the basis of its ability to return a waveform with the same RMS amplitude as that of the original waveform.

Relationship of filter length to signal cycle length. Working with time domain digital filters in a variety of electrogastrogram, electrodermal, EEG, and ERP applications, we came to wonder whether the length of the filter (in number of weights or in real time) relative to the length of a given sine wave cycle could be a factor in the accuracy of the filter. There is some appeal to the notion that a filter might perform better to the degree that it spans a larger portion of a given sine wave cycle, or a larger number of cycles. Alternatively, it may be the span of real time that the filter covers, rather than the number of cycles, that matters. A third possibility is that neither the span of real time nor the number of cycles matters.

We explored this empirically by creating two simulated 60-sec samples of a 50-μV peak-to-peak (17.678 μV RMS) pure sine wave, one at 5 Hz and one at 10 Hz. We then sampled the continuous sine wave functions at either 125 Hz or 250 Hz. We constructed a number of filters, each with a half-amplitude bandpass of 4–6.5 Hz (applied to the 5-Hz signal) or 8–13 Hz (applied to the 10-Hz signal). The 10-Hz simulation was selected because of the importance of alpha band (8–13 Hz) activity in much EEG research and because of the threat that alpha band activity poses in much ERP work. The 5-Hz simulation allowed an unconfounding of simulated signal frequency and sample frequency. Thus, for example, a 5-Hz signal sampled at 125 Hz provides the same number of samples per sine wave cycle as does a 10-Hz signal sampled at 250 Hz.

Table 1 contains the RMS values obtained for 32 simulated cases in which the weights spanned varying amounts of real time and thus varying numbers of signal cycles, from less than a tenth of a second to over 8 sec and from less than half a cycle to over 80 cycles. Not surprisingly, as the number of weights increased, the filter improved. Examining each column in the table, the RMS values rise monotonically until they are within 1% of the 17.678-μV ideal. Importantly, in all four columns, RMS values do not reach asymptote until about 7 cycles were spanned. In contrast, the real-time span of the filter shows no consistent relationship to the number of weights at which asymptote was reached.

Certain regularities in Table 1 confirm the validity of the simulation. For a given row (i.e., a given number of weights), the first and last cases (columns) produced identical RMS values (within rounding error). For example, with 21 weights, a 5-Hz signal sampled at 125 Hz and a 10-Hz signal sampled at 250 Hz both produced 4.321 μV. In both cases, sampling provided 25 samples per cycle, and the 21-weight filter spanned .84 cycle.

Table 1 raises significant questions about some kinds of filters in widespread use. The relatively low number of weights often employed in time domain digital filters in the EEG/ERP literature (e.g., the 7-weight boxcar filters historically common in P300 studies or the 9- to 15-weight boxcar filters evaluated by Farwell et al., 1993)

Table 1
Effect of Filter Weight Cycle Span

	Simulated Signal Frequency (Hz)			
	5		10	
Sample frequency (Hz)	125	250	125	250
Samples per signal cycle	25	50	12.5	25
No. Weights				
21	4.321	3.079	7.865	4.321
31	6.052	3.587	10.861	6.051
51	9.390	4.721	14.933	9.390
125	16.335	10.532	**17.743**	16.335
175	**17.485**	13.404	**17.662**	**17.485**
251	**17.741**	16.048	**17.635**	**17.741**
501	**17.636**	**17.752**	**17.648**	**17.636**
1,023	**17.670**	**17.611**	**17.671**	**17.672**

Note—Table entries are postfiltering RMS signal voltage in μV for an input signal consisting of 60 sec of a 5- or 10-Hz pure sine wave with a peak-to-peak amplitude of 50 μV. The 5-Hz data were submitted to time domain 4–6.5 Hz bandpass filters with specified numbers of weights. The 10-Hz data were submitted to time domain 8–13 Hz bandpass filters with specified numbers of weights. For a perfect 4–6.5 or 8–13 Hz filter (passing all 5- or 10-Hz activity and passing nothing else), the entry would be 17.678 μV. Tables entries near that asymptote (< 2% error) are in boldface.

appears to be far short of asymptote. Even with 21 weights and the greater precision that an unequal-weight filter allows, the RMS value was only 17%–44% of the correct value. It must be noted, of course, that no simulation can be comprehensive. Whereas a span of half a dozen cycles or more seems necessary in the four cases evaluated here, that number could vary as a function of the width of the filter pass band (the frequencies that the filter should retain) relative to the bandwidth of the target signal and also as a function of the amplitude and frequency of noise to be filtered out in the stop band. If anything, however, the present noiseless simulations may underestimate the appropriate number of weights. Minimally, these empirical simulations suggest that the number of weights should be sufficient to span at least 5–10 cycles of prominent signals in the time series of data being filtered.

Real-time filter weight density. These same simulation data can address a second practical question. For the filtering of a given time point and a given number of weights, the number of signal cycles that the weight series spans is a function of the sampling frequency of the raw data. For a fixed number of weights, increasing the sample frequency reduces the real-time span of data involved in the filtering of a given point. Does increasing the sample frequency (and thus the density of the weights in real time) have an impact on the performance of the filter?

The results of the simulations presented in Table 1 can be rearranged to address this question. Table 2 illustrates five pairs of filters, each with a half-amplitude bandpass of 8–13 Hz, applied to a simulated 10-Hz signal. The first case compares the signal passed by a 51-weight filter applied to data sampled at 125 Hz to the signal passed by a 101-weight filter applied to the same data sampled at 250 Hz. The two filters span about the same amount of real time (about four cycles), thus differing only in the real-time density of the weights. The resulting RMS values differ by less than .5%. The other cases provide similar comparisons for filters that span more data, with the same result. The similarity of the two RMS values in each pair of rows does not vary as a function of how close the RMS values are to asymptote, even though, as in Table 1, the RMS values do approach asymptote as the number of weights grows. In sum, weight density does not affect the filtered representation of the signal.

Table 2
Effect of Filter Weight Density

No. Weights	Milliseconds Spanned	Cycles Spanned	Sampling Frequency (Hz) 125	250
51	408	4.08	14.933	
101	404	4.04		14.985
63	504	5.04	16.269	
125	500	5.00		16.335
125	1,000	10.00	17.743	
251	1,004	10.04		17.741
251	2,008	20.08	17.635	
501	2,004	20.04		17.636
501	4,008	40.08	17.648	
1,023	4,092	40.92		17.672

Note—Table entries are postfiltering RMS signal voltage in μV for an input signal consisting of a 10-Hz pure sine wave with a peak-to-peak amplitude of 50 μV. For a perfect 8–13 Hz filter (passing all 10-Hz activity and showing nothing else), the entry would be 17.678 μV.

The practical implication of Table 2 is that increasing the sampling frequency and correspondingly the number of weights per cycle does not improve filter accuracy. It simply increases filter computation time. If available, extra computation time should be spent on increasing the number of weights without changing the sampling frequency. That will mean that more signal cycles are spanned, which in Table 1 is established as beneficial. This conclusion is consistent with the dictum that frequency resolution and temporal resolution vary inversely—for better frequency resolution, sacrifice temporal resolution by including a greater span of real time (i.e., more cycles) in the filter (Cook & Miller, 1992).

Impact of slow drift. For reasons to be reviewed below in the discussion of frequency domain filtering, a standard assumption is that frequency domain methods are very vulnerable to nonsinusoidal changes during the filtered epoch. We investigated the vulnerability of the type of time domain filter emphasized here to a particularly extreme version of that problem. To our pure 10-Hz sine wave, we added a linear trend, simulating the slow drift sometimes seen in EEG/ERP data. We employed a number of different slopes, ranging from .5 to 250 μV/sec. The composite waveform was sampled at 250 Hz and submitted to 251- and 1,023-weight filters designed to pass 8–13 Hz. The output of the 251-weight filter varied less than 1% up to a trend of 125 μV/sec. At 250 μV/sec, an unusually large drift for real data, the error was 1.4%. The 1,023-weight filter showed no variability at all, perfectly matching the RMS alpha value obtained when no trend was added. It is clear that, given enough weights, low-frequency activity is fully attenuated, such that even large amounts of slow drift have little or no effect on the 8–13 Hz filter output. Filters using fewer weights, typical in EEG/ERP research, would be far more vulnerable, and this problem would be exacerbated in studies in which researchers are interested in lower pass bands. Simons, Miller, Weerts, and Lang (1982) developed a method for dealing with the latter problem in long-interval CNV studies that may be applicable to other low-frequency measures.

DIGITAL FILTERING IN THE FREQUENCY DOMAIN

Concepts and Terminology

Up to this point we have been concerned with data represented and filtered in the time domain. The frequency domain provides an alternative perspective on digital filtering and frequency analysis that warrants some discussion and comparison with the time domain perspective. This alternative is of interest both as a different means of implementing a digital filter (in the fre-

quency domain) and as a means of generating the weights to be used in a digital filter in the time domain. Crucial to understanding how data collected in the time domain might be filtered in the frequency domain is an understanding of how time series data can be converted to a frequency domain representation.

The fast Fourier transform. In the substantial literature on frequency domain methods of data analysis in general and digital filtering in particular, the community of EEG/ERP researchers is probably most familiar with the fast Fourier transform (FFT; Cooley & Tukey, 1965) as a means of estimating the frequency composition of a time series. Intensive treatments of the FFT are available in numerous sources (e.g., Brigham, 1974; Glaser & Ruchkin, 1976). Most relevant here is an intuitive understanding of what the FFT does, what some of its practical constraints are, how it can be used to accomplish frequency filtering in the frequency domain, and how it can be used to facilitate frequency filtering in the time domain.

Subject to certain constraints, Fourier's theorem states that any time series waveform may be modeled as the sum of a set of sinusoidal waveforms, each of a different frequency and having an associated amplitude and phase. Figure 3 provides an illustration of this summation approach. This principle is the basis of *Fourier analysis*, which yields an estimation of the amplitudes and phases of the constituent sinusoids as a function of frequency. This representation of a signal is said to be "in the frequency domain." A *direct* Fourier transform is an algorithm that converts a digitally represented signal from the time domain to the frequency domain; an *inverse* Fourier transform converts the signal from frequency domain to time domain.[3]

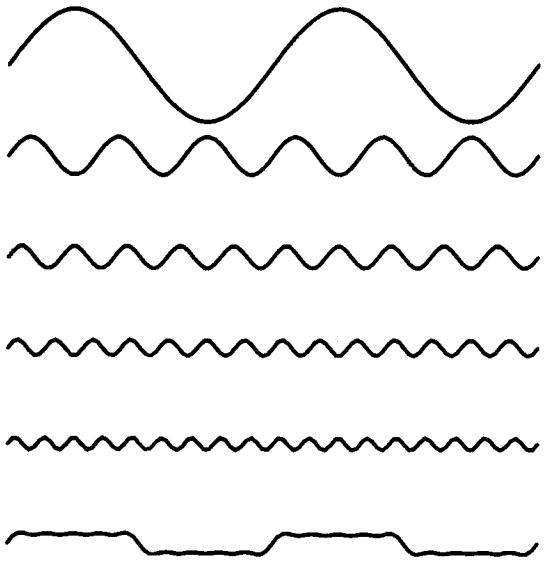

Figure 3. Illustration of the Fourier modeling of a square wave via summation of sine waves of appropriate frequency, amplitude, and phase. The square wave at the bottom is the average of the five sine waves above it. Note that the sine waves are in phase only at the half-cycle transitions of the square wave. Those synchronized transitions are thus reflected in the approximate square wave at the bottom. At all other times, the sine waves are out of phase with each other and thus cancel out, providing the plateaus of the square wave. Although this illustration employs five sine waves, according to the Fourier theorem a complete model would employ an infinite number of increasingly small-amplitude sine waves, further sharpening the half-cycle transitions and flattening the plateaus.

Frequency filtering with the FFT. Some simple arithmetic (see Cook & Miller, 1992) applied to a portion of the output of the Fourier transform produces a series of frequencies and amplitude (or, optionally, power) values at those frequencies. Those frequencies are often referred to as frequency *bins*.[4] The original time series of data (such as the square wave in Figure 3) is thus represented as a series of frequencies, each with a specific amplitude appropriate for modeling that time series. At this point, the investigator may pursue either of two goals. If the goal of the filtering is simply to quantify the amount of activity at certain frequencies, the investigator sums or averages the activity in the bins that come closest to spanning 8–13 Hz. The resulting score is analogous to the procedure, presented above, of leaving the data in the time domain, applying an FIR digital filter to the raw time series, and computing an RMS score on the filtered time series to quantify the amount of activity in a particular frequency band. If instead the goal of the filtering is to provide a filtered time series, the investigator can set the activity in all frequency bins outside the desired pass band to 0.0 and then perform an inverse Fourier transform to reproduce the (now filtered) time series.

In summary, the time domain and frequency domain approaches may be contrasted as follows. For time domain filtering, the investigator applies the filter to the raw data, directly producing the filtered time series. If frequency analysis is desired, an additional step is necessary, such as an RMS computation. For frequency domain filtering, the Fourier transform of the raw data is computed, the filter is applied (by zeroing unwanted frequency bins), and the inverse Fourier transform is applied. If quantification of pass band activity is all that is desired, the inverse Fourier transform step is omitted.

Some limitations of frequency domain filtering are well established. It requires that filtering be delayed until the full epoch to be filtered has been acquired. This may present difficulties if filtered data are required in real time. Another limitation is often imposed by the algorithm used to compute the transform. Most typically, some form of the FFT (for some variations, see Brigham, 1974) is used to provide increased computational speed in exchange for certain limitations (Dumermuth & Molinari, 1987). For example, the algorithm is usually applicable only to a time series with exactly 2^n members, where n is a positive integer. Despite these limitations, frequency domain filtering with the FFT is a well-established practice in the EEG/ERP literature. A third limitation is that, if the signal is not stationary (i.e., does not have consistent mean and frequency components) over

the epoch for which the Fourier transform is calculated, artifactual high-frequency noise and discontinuities between adjacent epochs can result (Attinger, Anne, & McDonald, 1966; Cook & Miller, 1992).

Stationarity. It is commonly noted that real-world psychophysiological data routinely violate the Fourier method's requirement of stationarity, but this issue has not received extensive treatment in the EEG/ERP literature. The property of *stationarity* refers to complete consistency over time of the constituent functions underlying an observed time series. Since the Fourier approach to a time series of length T (in seconds) starts with a sine wave of frequency $1/T$ (in cycles per second), the Fourier modeling of the time series will work properly only when the slowest frequency in the data is exactly $1/T$. Furthermore, all other (faster) frequencies in the data should be limited to the harmonics of $1/T$; that is, $2/T$, $3/T$, and so on. Brigham (1974, chap. 6) and Glaser and Ruchkin (1976, chap. 3) provided graphical illustrations of the misallocation of frequency information, called *leakage*, that occurs when nonharmonic frequencies are present. Were the FFT applied to a time series of infinite length, this leakage into inappropriate frequency bins would not occur. This point can readily be understood intuitively, in that as T approaches infinity, $1/T$ approaches 0.0. As a result, the frequency resolution becomes extremely high, so that virtually any activity is close to a harmonic. Very long analysis epochs are thus much less vulnerable to leakage of nonharmonic activity.

On the other hand, long analysis epochs are vulnerable to another violation of the stationarity assumption: changes in the constituent frequencies over time. The Fourier transform from the time to the frequency domain produces a set of amplitude and phase values, one amplitude and one phase value for each harmonic. Because the entire time series will be described by a (static) set of frequencies of specified amplitude and phase, this approach cannot deal correctly with any change in the amplitude or phase of a given frequency during the T epoch. In that sense, the data must be "stationary" during the epoch analyzed.

One way to deal with the stationarity assumption is to break up a long time series into shorter epochs, on the assumption that data will be more stable over shorter periods. Thus, for example, a 60-sec time series might be analyzed as sixty 1-sec epochs, rather than as a single 60-sec epoch. This is a computationally demanding strategy, for which the speed improvement of the FFT has proven very important. We wish to draw attention to a newer method, the fast Hartley transform (FHT; Bracewell, 1984; for a very accessible introduction, see O'Neill, 1988), which we believe would be superior in most EEG/ERP applications. Conceptually, the Hartley transform is closely analogous to the Fourier transform. However, the FHT algorithm relies solely on real arithmetic, unlike the traditional FFT, which involves complex arithmetic. O'Neill discussed why the FHT takes half the time and half the memory of the traditional FFT. Like the Cooley–Tukey FFT method discussed above, the FHT requires that the length of the time series be a power of 2.

Design of Time Domain Filters in the Frequency Domain

With the preceding brief discussion of Fourier analysis in mind, we can now examine one means of designing the time domain digital filters discussed earlier in this paper that will illustrate some aspects of the relationship between time domain and frequency domain data representations and filter methods. The FIR filters involve convolving a time series with a symmetric weight series (which may itself be considered a time series), yielding a filtered time series. Specific steps for constructing such filters (i.e., generating the weight series) have been described by Gold and Rader (1969; see also Ackroyd, 1973; Cook & Miller, 1992; Dumermuth & Molinari, 1987; Oppenheim & Schafer, 1975), and software implementing the weight-generation process is available (e.g., Cook, 1981). Briefly, the technique involves four steps, summarized in Figure 4. First, the filter's ideal, frequency domain gain function is specified as an array of 1s and 0s corresponding to the pass and stop bands, respectively. Second, the inverse Fourier transform is applied to this gain function to obtain its time domain equivalent, which serves as the initial set of filter weights. Following the principles of Fourier analysis described above, these weights are by definition a time series consisting of the sum of a set of sinusoids, each corresponding to a different frequency in the pass band of the ideal gain function. Although the process of designing an FIR filter can stop after these two initial steps, two further steps are often taken to improve and evaluate the filter. The third step involves refining the filter weight series by applying a *window* to it. Figure 4 illustrates three such windowing functions. Finally, the windowed filter is evaluated: the direct Fourier transform converts the windowed weight series back to the frequency domain, which yields the actual gain function, which can be evaluated in comparison with the ideal gain function specified in Step 1. Steps 3 and 4 may be repeated with different parameters until a weight series is obtained that has a gain function satisfactory for the user's application.

Although Cook and Miller (1992, Appendix B) described these steps in greater detail, the windowing process is so central to the filter generation process that it deserves further comment here. In general, windowing involves cross-multiplying a function with the original filter weights, with the result being a filter that is narrower (has fewer weights) and/or tapered at its ends. Windows vary in their shape and width, and the choice of a window involves balancing tradeoffs related to filter width, computation speed, transition bandwidth, and gain function ripple (i.e., variation in the actual gain function around 0.0 in the stop band and 1.0 in the pass band). The simplest window function is rectangular, and its effect is to truncate the weight series. Truncating the

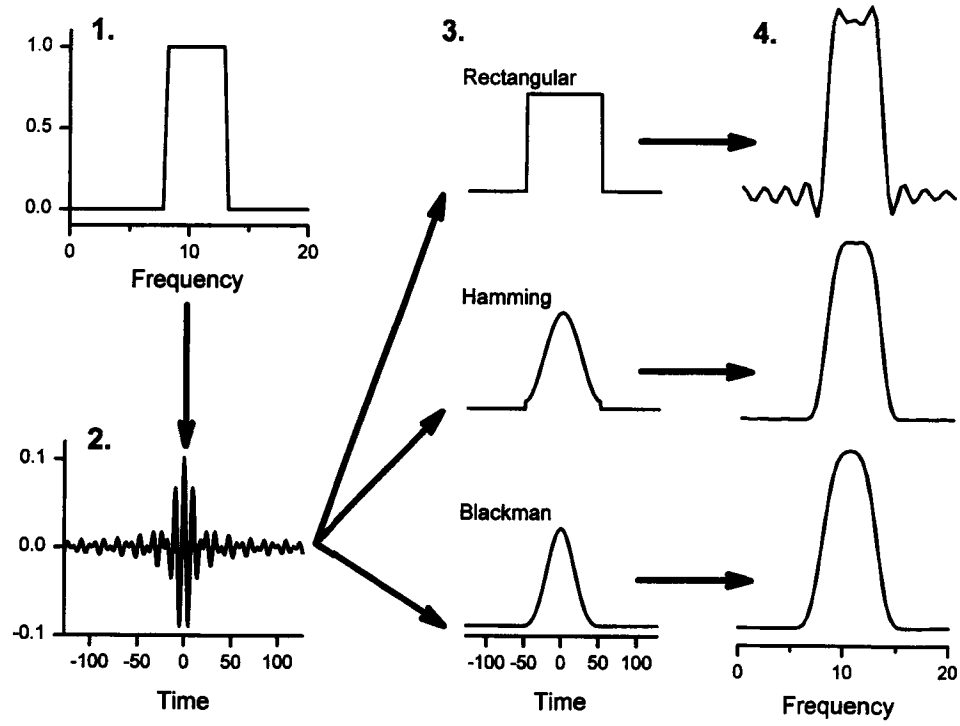

Figure 4. The four-step process of generating filter weights (see text). For illustration, an 8–13 Hz bandpass filter is the stipulated goal. The obtained gain functions resulting from each of three windowing functions are compared.

weight series directly reduces computation time. However, applying such a rectangular window also produces ripple in the gain function, as is apparent in Figure 4. Ripple can in turn be reduced by tapering the ends of the truncated weight series, using one of several tapering windows (Ackroyd, 1973). The Hamming window is relatively gradual among the tapering functions illustrated in Figure 4. For example, it reduces the weights by about 46% halfway from the center of the weight series to the outermost weight. In general, applying a Hamming window reduces but does not eliminate truncation-related ripple. However, it also widens the filter's transition band, making it less selective across the frequency spectrum. Compared with the Hamming window, the Blackman window is more severe (the corresponding reduction is 66% at the halfway point), and its effects are more extreme. That is, it nearly eliminates gain function ripple but further widens the gain function transition band.

Practical Considerations in Frequency Domain Digital Filtering

Slow drift. The requirement for stationarity, the requirement that all frequencies present in the time series be limited to the harmonics of $1/T$, and the typical availability of analysis only of time series the lengths of which are a power of 2 provide important practical constraints on the appropriateness of frequency domain analysis. A particularly underappreciated aspect of these constraints is the stationarity requirement in the face of slow drift in the data.

The wide range of possible violations of stationarity precludes a thorough empirical investigation of the conditions under which such violations seriously distort one's analysis. However, we used the simulated slow-drift data discussed earlier to investigate the impact on FHT output (see Glaser & Ruchkin, 1976, chap. 3, for an analytic discussion of this point). As was the case with the time domain filters with adequate numbers of weights, FHT output was not seriously affected. Even when we added a 250-μV/sec linear trend, FHT output in the alpha band changed by less than .07% from the ideal value. Of relevance in studies of P300, the 1–2 and 2–3 Hz bins showed almost no effect. Virtually all of the activity contributed by the linear trend was confined to the 0–1 Hz bin. Thus, slow drift is likely to be problematic only for investigators concerned about very slow activity.

Windowing. Because Fourier analysis in principle requires continuous data, but real-world data epochs are discontinuous, the ends of the epoch present a serious problem. Unless the only signals present in the data are sinusoidal waves with periods of exactly $1/T$ and its harmonics (T, again, being the length of the epoch), the data value at the end of the epoch will not generally be that of the start at the epoch. To minimize disruption from that discontinuity, it is common to window each epoch—to taper the ends toward zero by using a weighting function

that resembles the positive half cycle of a sine wave. Understanding the relationship of this windowing of data, in the context of frequency domain filtering, to the windowing of weights discussed earlier, in the context of time domain filtering, helps to clarify the relationship of time and frequency domain filtering more generally. In the case of time domain filtering, we window the *weights*, in the time domain, and then we cross-multiply the weights with the data. In the case of frequency domain filtering, we window the *data* in the time domain, and then we employ the FFT or FHT to convert the data to the frequency domain. In both cases, the windowing is applied in the time domain. In fact, the same choices of windowing functions are available for both uses. In both the time domain and the FHT computations used for the simulations that we report here, we employed the Hamming window.

The role of windowing in time domain and frequency domain approaches may be summarized and compared as follows. The time domain approach cross-multiplies three time series in the time domain: the windowing function, the (unwindowed) weights (the inverse transform of the desired gain function), and the original data. The frequency domain approach cross-multiplies two time series in the time domain: the windowing function and the original data. Then, after this product is transformed to the frequency domain, the frequency domain approach does a third cross-multiplication, using the desired gain function. The pragmatic point is that the investigator needs to choose a windowing function for use with FFT or FHT analysis, just as was needed for use with the time domain filtering discussed earlier.

Epoch overlap. Although advisable in principle, this tapering of data prior to FFT or FHT necessarily loses information (because some data points receive weights less than 1.0 in the tapering function). To compensate for this problem, overlapping segments are commonly analyzed when multiple contiguous epochs are available. For example, when one is processing a lengthy EEG epoch 1 sec at a time with 50% overlap, the first epoch to be analyzed covers 0–1,000 msec, and the second epoch covers 500–1,500 msec. The weighting function would have tapered (substantially underrepresented) the data from roughly 800 to 1,000 msec in the 0–1,000 msec epoch, but those data would pass through the tapering function almost intact during analysis of the 500–1,500 msec epoch.

There is no firm rule for how much overlap there should be, but common amounts are 25%, 50%, and 75%.[5] When one is using a pure sine wave as simulated data, the choice of overlap should make no difference, provided that the epoch is long relative to the sine wave frequency, because the same information is contained in any epoch. In the present case, we applied the FHT to 1.024-sec epochs (2^8 data points) of the 60-sec, 10-Hz sine wave. As expected, overlaps of 0%, 25%, 50%, and 75% made no difference, within rounding error, and they produced RMS values very close to those for the time domain filters with the most weights in Tables 1 and 2.

One consequence of the overlap strategy can be noted. The run time of a straightforward implementation of the Fourier transform is proportional to N^2, where N is the number of data points in the time series analyzed. The fast Fourier transform improves this to $N * \log_2 N$. This is an enormous improvement for large N. As noted earlier, one option to reduce violation of the stationarity assumption is to analyze long epochs as a series of short epochs, such as a 60-sec time series analyzed via 60 FFTs of 1-sec epochs. This approach has the added advantage of requiring less total run time—43% less for a 60-sec time series. However, using a 50% overlap, 119 rather than sixty 1-sec analyses must be run, taking 14% longer than a single 60-sec time series. Furthermore, because the frequency resolution of the FFT or FHT output is $1/T$, analyzing the longer epoch provides better frequency resolution. Glaser and Ruchkin (1976) showed that the frequency spectrum leakage discussed earlier affects less of the frequency spectrum for small values of $1/T$. Investigators must weigh these tradeoffs in selecting epoch length.

COMPARISON OF TIME AND FREQUENCY DOMAIN DIGITAL FILTERING

The interchangeability of time domain and frequency domain representations of a given waveform warrants emphasis. Either description completely specifies the raw phenomenon illustrated in the composite waveform. One description may be more tractable for a particular type of analysis or more intuitively appealing for a particular type of question, but exactly the same information is available in the two representations. Given the focus in this paper on filtering, the relevant implication of this fundamental point is that equivalent filtering can be accomplished in the time domain or in the frequency domain. For at least some kinds of methods in each domain, there is a direct analogue in the other domain. An example of this can be seen in the steps illustrated in Figure 4. We began with raw data collected in the time domain and with an ideal filter gain function specified in the frequency domain. To generate the weights for a time domain filter, we transformed the frequency domain gain function into the time domain. Alternatively and equivalently, however, we could transform the raw data into the frequency domain and conduct the filtering there, simply by cross-multiplying, frequency bin by frequency bin, the frequency spectrum of the raw data with the desired gain function. Various details, choices, and tradeoffs are left out of this comparison for clarity, but the point is that many time domain operations have frequency domain analogues and vice versa. The investigator should choose a method based on familiarity, availability, speed, and so forth.

Of practical interest is how the accuracy and speed of the time domain digital filtering method emphasized earlier in this paper compare with those of the FHT. We evaluated this empirically, finding both methods to be highly accurate (given sufficient weights for the time do-

Table 3
Time Domain Filtering of Real EEG

No. Weights	Filtered RMS Value
21	7.071
31	4.600
41	2.591
51	2.470
101	3.325
125	3.538
175	3.795
251	3.982
501	4.177
1,023	4.280

Note—Table entries are postfiltering RMS signal voltage in μV for an input signal consisting of real EEG sampled at 250 Hz.

main filter) and finding each method to be superior under some circumstances, depending on tradeoffs of accuracy and speed. We employed two 60-sec data sets, sampled at 250 Hz and filtered for an 8–13 Hz bandpass. The first was the same pure 10-Hz sine wave used in simulations reported above. The second was a sample of actual EEG, to provide a comparison using a complex, real-world signal containing noise.

Filter Accuracy

Choice of degree of overlap is potentially more of an issue with complex, nonstationary, real-world EEG/ERP data. We used 60 sec of real EEG data recorded from F3–M1 and M2–M1 derivations (International 10–20 System; Jasper, 1958) and converted off line to a linked mastoids reference, corrected for EOG artifact (Gratton et al., 1983; Miller et al., 1988), and analyzed in 1.024-sec epochs. Each epoch was Hamming windowed. With the FHT, overlaps of 0%, 25%, 50%, and 75% produced RMS values of 4.459, 4.025, 4.293, and 4.290 μV. These four values are quite consistent, although increased overlap appears to be associated with more stable values.

Table 3 illustrates the results of similar analyses but using a time domain filter with varying numbers of weights. As the number of weights increases from 51, the RMS values grow toward that for the 1,023-weight filter. At 1,023 weights, the filtered RMS value (4.280) is virtually identical to the values for the 50% (4.293) and 75% (4.290) overlaps in the FHT analysis.

As illustrated above in the Etienne et al. (1990) example and also in Figure 5, filters with fewer weights, in general, pass less of the activity in the pass band(s) and more activity in the stop band(s). If the epoch to be filtered contains considerable activity in frequencies adjacent to the desired pass band, the computed (filtered) RMS value may fluctuate substantially as a function of what transition band activity is passed. Thus, the RMS values could vary greatly as a function of activity in other parts of the frequency spectrum. This fluctuation is apparent for the filters in Table 3 with the fewest weights. The poor gain functions of those filters (see Figure 5) indicates that a significant portion of the low frequencies in the data would be passed. With RMS values at 0–1 Hz being substantially larger than those in the alpha band in the data sample, the RMS values obtained by the putative "alpha" filters in Table 3 that have relatively few weights fluctuated badly. It should be noted that these filters nevertheless have considerably more weights than do many used in the ERP literature, underscoring our earlier point about the potential inadequacy of common time domain digital filters.

The vulnerability of time domain filters with few weights is a function of the signal-to-noise ratio (pass-

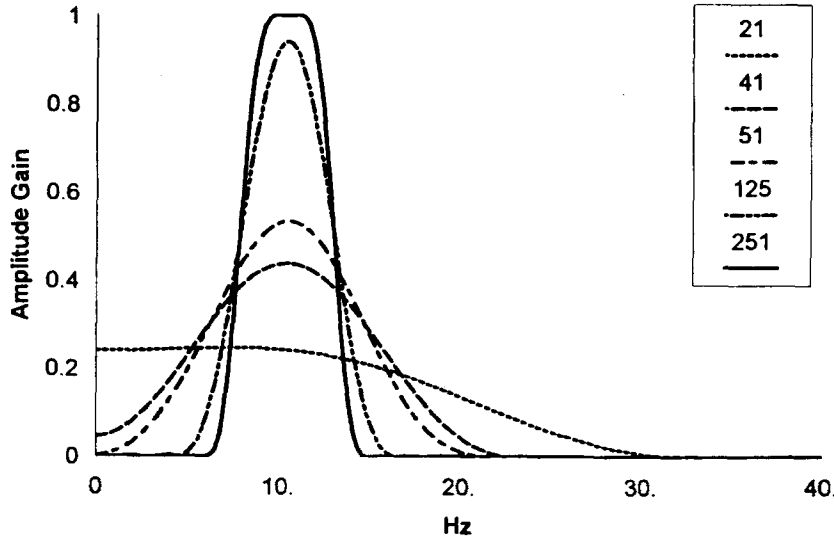

Figure 5. Amplitude gain functions for several of the filters presented in Table 3. The different line types illustrate filters differing in number of weights. The *x*-axis has been truncated at 40 Hz, to provide more visual resolution at lower frequencies. Note that the gain function approximates the ideal of a square wave better as the number of weights increases.

band activity/stop-band activity), the width of the pass band (because narrow pass bands are approximated especially badly with few weights), and how close the lower bound of the pass band is to 0 Hz (because a significant dc level may be present, and filters with few weights may pass much of that activity). In fact, no filter in Table 3 with fewer than 51 weights removed all of the 0-Hz activity nor passed even half of the activity in the pass band. The 51-weight filter eliminated over 99% of the 0-Hz activity but passed no more than 54% of the activity at any frequency within the 8–13 Hz pass band (see Figure 5). Not until the 175-weight filter was 100% of the 10–11 Hz activity passed (still, less than 100% of the 8–10 Hz and 11–13 Hz activity was passed). In summary, the FHT was clearly superior to the shorter time domain filters in representing the 10-Hz signal. With enough weights in the time domain filter, the two methods converge—as they should, given the equivalence of time domain and frequency domain filtering methods discussed earlier.

Filter Speed

Aside from accuracy, the two kinds of methods may differ in speed. An appeal of the FHT is its speed relative to that of typical FFT implementations, but it is not clear a priori how those frequency domain methods compare with the speed of the time domain digital filter method emphasized in this paper. For a given data sample, the run time of the time domain filter is directly proportional to the number of weights. Table 1 illustrated that that filter's accuracy asymptoted when the number of weights was sufficient to span several cycles of the target signal. In contrast, the run time of the FHT is directly proportional to the degree of overlap of the windowed epochs. Thus, because run times for the two methods are sensitive to different parameters, no single, general speed comparison is possible (see Ruchkin, 1988, for some general remarks on comparative speed). However, if on the basis of Tables 1 and 3 we assume that the 251-weight filter comes close enough to asymptotic accuracy (asymptote reached for three of the four columns in Table 1), and if we take 50% overlap of epochs to be a middle-ground case for the FHT approach (50% is commonly employed in EEG studies using the FFT), in our various simulations we found that the FHT computation took about 60% as long as that of the time domain filter.

Although such a difference in run time is not striking, the advantage of the FHT could become significant, depending on the size of the data set to be analyzed and the computational facilities available. On the other hand, typical applications of time domain filters often employ far fewer weights than do most of the cases investigated here. Filter speed would be substantially enhanced, although in many cases filter accuracy would be significantly reduced. In such cases, the time domain approach could be considerably faster than the FHT and, therefore, faster still than a standard FFT implementation. Furthermore, the time domain method is not confined to epochs consisting of 2^n samples.

It should be noted that the FFT and FHT algorithms were designed to minimize execution time. Conceivably, a variety of optimizations could be developed for time domain filters. The time domain method emphasized in this paper was not developed with speed in mind, nor is it thoroughly representative of the many other forms of time domain filter methods. In either domain, it is sometimes possible to optimize a given method for a highly constrained problem on a particular computing platform. For example, algorithms have been developed to generate efficient integer-oriented assembly language for a specific-length FFT (e.g., 256 points) for a specific computer architecture or to remove the power-of-2 constraint (Brigham, 1974). In summary, the implementation of a particular filter on a particular platform may have much more impact on computational speed than does the choice of generic type of filter.

SUMMARY AND RECOMMENDATIONS

This paper has emphasized, first, that time domain digital filters in various forms are pervasive in the EEG/ERP literature, although the conceptual continuity across specific examples is not always appreciated; second, that time domain filters are robust to variations in real-time filter weight density and slow drift but not to variations in number of cycles spanned; and, finally, that the FHT appears to be an underappreciated method which is faster not only than the commonly used FFT but also in some cases than time domain digital filtering. Such generalizations must be offered cautiously, for there are surely cases in which empirical conclusions might be different. Furthermore, the goals of digital signal processing vary greatly across studies. For example, the FFT and FHT have the advantage of providing a full-spectrum, frequency domain characterization of the data, which is more commonly of interest in EEG than in ERP studies. Time domain filtering can readily be applied in real time, and it preserves the data in their original form (more suitable for conventional ERP component scoring), although the output of frequency domain filtering can be transformed back to the time domain as well.

Our strongest recommendation is that investigators evaluate the filters they plan to use before becoming committed to them.[6] This can be done empirically, with simulations such as those undertaken for this paper, or analytically, relying on evaluation of the gain function of a given filter. As concluded above, the number of cycles spanned deserves particular attention.

Second, we recommend that gain functions be described more fully than is typical in published work. Although inclusion of a figure providing the full gain function is generally unnecessary, investigators should report more than the nominal cutoff frequency (which itself should be specified as the half-amplitude or the half-

power frequency; see Cook & Miller, 1992). For example, the gain function slope at the cutoff frequency and in some cases the degree of ripple in the pass band and stop band should be provided. Gains at specific frequencies are sometimes important, such as alpha band artifact in a P300 study involving single-trial scoring.

Third, we recommend that investigators pay as much attention to the low end of the frequency spectrum as to the high end. The need for a sampling frequency well above twice the highest frequency in the raw data is generally understood. Investigators often implement some sort of smoothing filter to minimize the resulting high-frequency activity that is not of interest in the time series to be filtered. However, present simulations suggest that time domain digital filters may need to span several cycles of the slowest frequencies present in order for artifactual results to be avoided. The number of cycles needed in order to reach asymptotic filter performance may be highly dependent on the parameters of a given case, but the general point is that frequency resolution will improve when more weights are used.

Fourth, we share Farwell et al.'s (1993) serious reservations about boxcar filters. We recommend that their use be confined to cases in which their appropriateness has been clearly established. Their computational speed advantage rarely provides a compelling reason for their use.

We hope that the present discussion encourages appreciation for and more systematic evaluation of time domain and frequency domain approaches to digital filtering.

REFERENCES

ACKROYD, M. H. (1973). *Digital filters*. London: Butterworth.

ATTINGER, E. O., ANNE, A., & MCDONALD, D. A. (1966). Use of Fourier series for the analysis of biological systems. *Biophysical Journal*, **6**, 291-304.

BRACEWELL, R. N. (1984). The fast Hartley transform. *Proceedings of the IEEE*, **72**, 1010-1018.

BLOOMFIELD, P. (1976). *Fourier analysis of time-series: An introduction*. New York: Wiley.

BRIGHAM, E. O. (1974). *The fast Fourier transform*. Englewood Cliffs, NJ: Prentice-Hall.

COLES, M. G. H., GRATTON, G., KRAMER, A., & MILLER, G. A. (1986). Principles of signal acquisition. In M. G. H. Coles, E. Donchin, & S. W. Porges (Eds.), *Psychophysiology: Systems, processes, and applications–A handbook* (pp. 183-221). New York: Guilford.

COOK, E. W. (1981). FWTGEN—An interactive FORTRAN II/IV program for calculating weights for a non-recursive digital filter. *Psychophysiology*, **18**, 489-490.

COOK, E. W., & MILLER, G. A. (1992). Digital filtering: Background and tutorial for psychophysiologists. *Psychophysiology*, **29**, 350-367.

COOLEY, W. J., & TUKEY, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematical Computing*, **19**, 297-301.

DONCHIN, E., & HEFFLEY, E. F., III (1978). Multivariate analysis of event-related potential data: A tutorial review. In D. A. Otto (Ed.), *Multidisciplinary perspectives in event-related brain potential (ERP) research* (Publ. No. EPA-600/9-77-043, pp. 555-572.). Washington, DC: U.S. Government Printing Office.

DUMERMUTH, G., & MOLINARI, L. (1987). Spectral analysis of the EEG: Some fundamentals revisited and some open problems. *Neuropsychobiology*, **17**, 85-99.

ETIENNE, M. A., DELDIN, P. J., GIESE-DAVIS, J., & MILLER, G. A. (1990). Differences in EEG distinguish populations at risk for psychopathology. *Psychophysiology*, **27** (Suppl. 4A), S28.

FARWELL, L. A., MARTINERIE, J. M., BASHORE, T. R., RAPP, P. E., & GODDARD, P. H. (1993). Optimal digital filters for long-latency components of the event-related brain potential. *Psychophysiology*, **30**, 306-315.

FORD, J. M., WHITE, P., LIM, K. O., & PFEFFERBAUM, A. (1994). Schizophrenics have fewer and smaller P300s: A single-trial analysis. *Biological Psychiatry*, **35**, 96-103.

GIESE-DAVIS, J. E., MILLER, G. A., & KNIGHT, R. A. (1993). Evidence for a memory deficit in subjects at risk for psychosis. *Psychophysiology*, **30**, 646-656.

GLASER, E. M., & RUCHKIN, D. S. (1976). *Principles of neurobiological signal analysis*. New York: Academic Press.

GOLD, B., & RADER, C. M. (1969). *Digital processing of signals*. New York: McGraw-Hill.

GRATTON, G., COLES, M. G. H., & DONCHIN, E. (1983). A new method for off-line removal of ocular artifact. *Electroencephalography & Clinical Neurophysiology*, **55**, 468-484.

HELLER, W., NITSCHKE, J. B., ETIENNE, M. A., & MILLER, G. A. (1997). Patterns of regional brain activity differentiate anxiety subtypes. *Journal of Abnormal Psychology*, **106**, 375-385.

JASPER, H. H. (1958). The ten–twenty electrode system of the International Federation. *Electroencephalography & Clinical Neurophysiology*, **10**, 371-375.

MAKEIG, S., & JUNG, T.-P. (1996, October). *Event-related spectral perturbations*. Paper presented at the annual meeting of the Society for Psychophysiological Research, Vancouver.

MILLER, G. A., GRATTON, G., & YEE, C. M. (1988). Generalized implementation of an eye movement correction procedure. *Psychophysiology*, **25**, 241-243.

O'NEILL, M. A. (1988, April). Faster than fast Fourier. *Byte*, pp. 293-300.

OPPENHEIM, A. V., & SCHAFER, R. W. (1975). *Digital signal processing*. Englewood Cliffs, NJ: Prentice-Hall.

RUCHKIN, D. S. (1988). Measurement of event-related potentials: Signal extraction. In T. W. Picton (Ed.), *Handbook of electroencephalography and clinical neurophysiology* (rev. ed., Vol. 3, pp. 7-43). Amsterdam: Elsevier.

RUCHKIN, D. S., & GLASER, E. M. (1978). Some simple digital filters for examination of CNV and P300 waveforms on a single trial basis. In D. A. Otto (Ed.), *Multidisciplinary perspectives in event-related brain potential (ERP) research* (Publ. No. EPA-600/9-77-043, pp. 579-581). Washington, DC: U.S. Government Printing Office.

SIMONS, R. F., MILLER, G. A., WEERTS, T. C., & LANG, P. J. (1982). Correcting baseline drift artifact in slow potential recording. *Psychophysiology*, **19**, 691-700.

SRINIVASAN, R., TUCKER, D. M., & MURIAS, M. (1998). Estimating the spatial Nyquist of the human EEG. *Behavior Research Methods, Instruments, & Computers*, **30**, 8-19.

WOODY, C. D. (1967). Characterization of an adaptive filter for the analysis of variable latency neuroelectric signals. *Medical & Biological Engineering*, **5**, 539-553.

YEE, C. M., & MILLER, G. A. (1988). Emotional information processing: Modulation of fear in normal and dysthymic subjects. *Journal of Abnormal Psychology*, **97**, 54-63.

NOTES

1. The methods discussed in this paper are applicable to data points arrayed in space as well as to those arrayed in time. Spatial frequency analysis becomes more feasible as the number and spatial density of electrode placements increases (see Srinivasan, Tucker, & Murias, 1998). The methods discussed in this article generally assume that the time series of data resulted from sampling at equal time intervals. Thus, for a straightforward application of these methods to spatial time-series data, electrodes would have to be equally spaced on the scalp. Furthermore, two- and three-dimensional extensions of such methods would be desirable. The methods discussed in this paper are also applicable beyond EEG/ERP data, encompassing other psychophysiological measures such as the electrogastrogram, the magnetoencephalogram, heart rate, and hemodynamic imaging data.

2. A sine wave signal can be completely characterized by three parameters: amplitude, frequency, and phase. The concept of *phase* can be readily understood if one conceives of the time course of a cosine wave as ro-

tation around a circle. Thus, phase refers to where the cosine function is in its cycle at a given moment. Phase is commonly quantified in terms of degrees (of the 360° in a circle) or in radians (of the 2π radians in a circle). A cosine wave starting at amplitude $y = 1$ (at $x = 0$) is said to be at a phase angle of 0° or 0 radians. One quarter cycle later, as it crosses 0 (at $x = 1$), the cosine wave is at 90° (360/4) or $\pi/2$ ($2\pi/4$) radians. One can follow this throughout the cycle or, equivalently, around the circle.

3. See Appendix A in Cook and Miller (1992) for discussion of the computational steps for the direct and inverse Fourier transforms—that is, for shuttling between time and frequency domain representations. See Makeig and Jung (1996) for a discussion of some nonsinusoidal approaches to frequency decomposition in psychophysiology.

4. Although *bin* is the common term, it can be misleading. It implies that the amplitude value for a given "bin" reflects all of the activity in a range of frequencies. For example, in an FFT of a 1-sec epoch, the frequency resolution will be $1/T = 1$ Hz. It is common to misunderstand, say, the 8-Hz bin as capturing all frequencies between 8 and 9 Hz or between 8.5 and 9.5 Hz. In fact, however, it correctly represents activity only at precisely 8 Hz. In this example, any activity that is not a harmonic of 1 Hz—that has a frequency that is not an integer—is not accurately represented by any "bin" and instead "leaks" into other "bins." This issue of harmonics and leakage is discussed later in this paper (see also Glaser & Ruchkin, 1976).

5. As this paper went to press, it was pointed out that if a Hanning window is used, then 50% epoch overlap provides uniform weighting over time (i.e., that the weights applied to each time point sum to 1.0), whereas there is no ideal overlap for a Hamming window (James Long, personal communication, July 25, 1997).

6. Software (based on Cook, 1981) available from E. W. Cook computes and plots the gain function for any arbitrary set of weights used in the kind of zero-phase-shift FIR filter emphasized in this paper, including a boxcar filter. Alternatively, it computes weights for such filters on the basis of user-specified cutoff frequencies, sampling frequency, and window type.