Online Chapter 12
Time and Frequency: A Closer Look at Filtering and Time-Frequency Analysis

A Note About This Chapter

This is a revised version of Chapter 5 from the first edition of An Introduction to the Event-Related Potential Technique (and is subject to the copyright of that edition). I wrote a completely new chapter on filtering for the second edition that left out the mathematical details of filtering, and the purpose of this online chapter is to provide those details. I've also added some new information about time-frequency analysis to this online chapter.

Overview

This chapter provides a more detailed discussion of filtering and time-frequency analyses. These topics have already been covered at a conceptual level in previous chapters; the goal of this chapter is to provide some of the mathematical details. However, the math will be very simple and will be limited to addition, subtraction, multiplication, and division, plus the concept of a sine wave and the mathematical operation of convolution (which was described in detail in Online Chapter 11). There will be no calculus!

Although I will provide several equations that describe how filtering works, it is still somewhat simplified. There are several books on filtering that the mathematically inclined reader may wish to read (e.g., Glaser & Ruchkin, 1976). There is also a very fine book by Mike Cohen on time-frequency analysis and related topics in EEG/ERP research (Cohen, 2014). It should also be noted that the term *filter* can refer to any of a large number of data manipulations, but this chapter will focus on the class of filters that ERP researchers typically use to attenuate specific ranges of frequencies, which are known as *finite impulse response filters*. A brief discussion of *infinite impulse response filters* will also be provided.

ERP waveforms are generally conceptualized and plotted as *time-domain* waveforms, with time on the X axis and amplitude on the Y axis. In contrast, filters are typically described in the *frequency-domain*, with frequency on the X axis and amplitude or power on the Y axis[1]. Because ERP researchers are typically more interested in temporal information rather than frequency information and because temporal information may be seriously distorted by filtering, it is important to understand filtering as a time-domain operation as well as a frequency-domain operation[2]. This chapter therefore describes how filters operate in both the time and frequency domains, as well as the relationship between these domains. I will focus primarily on the *digital* filters that are used for offline filtering, but the principles discussed here also apply to the *analog* filters that are found in amplifiers.

This chapter assumes that you've already read the basic chapter on filtering (Chapter 7), the chapter on averaging and time-frequency analysis (Chapter 8), and the chapter on convolution (Online Chapter 11). If you haven't read those chapters in a while, you might want to skim through them before diving into this chapter. In particular, you should be sure you understand the following fundamental principle, which was described near the beginning of Chapter 7.

Fundamental Principle of Frequency-Based Analyses: Power at a given frequency does not mean that the brain was oscillating at that frequency.

Filtering as a Frequency-Domain Procedure

Let's start with a reminder of how filtering is achieved in the frequency domain, which is the typical way of thinking about filtering. Figure 12.1 shows the process. First, the ERP waveform (or a segment of EEG data) is converted from the time domain into the frequency domain *spectrum* by means of the Fourier transform.

[1] In this context, power is simply amplitude squared.

[2] The discussion of filtering presented here applies to transient waveforms from virtually any source, including event-related magnetic fields. However, these principles are relevant primarily for the transient responses that are elicited by discrete stimuli and not to the steady-state responses that are elicited by fast, repetitive streams of stimuli presented at a constant rate.

This spectrum is then multiplied by the frequency response function of the filter, producing a filtered spectrum. In the example shown in Figure 12.1, the gain of the frequency response function falls from 1.0 at low frequencies to 0.0 at high frequencies, causing the low frequencies to be simply copied from the unfiltered spectrum to the filtered spectrum and causing the high frequencies to be attenuated. The gain of the filter is approximately 0.1 at 60 Hz, so 10% of the 60-Hz noise is passed from the unfiltered spectrum to the filtered spectrum. Finally, the filtered spectrum is converted back into the time domain by means of the inverse Fourier transform.

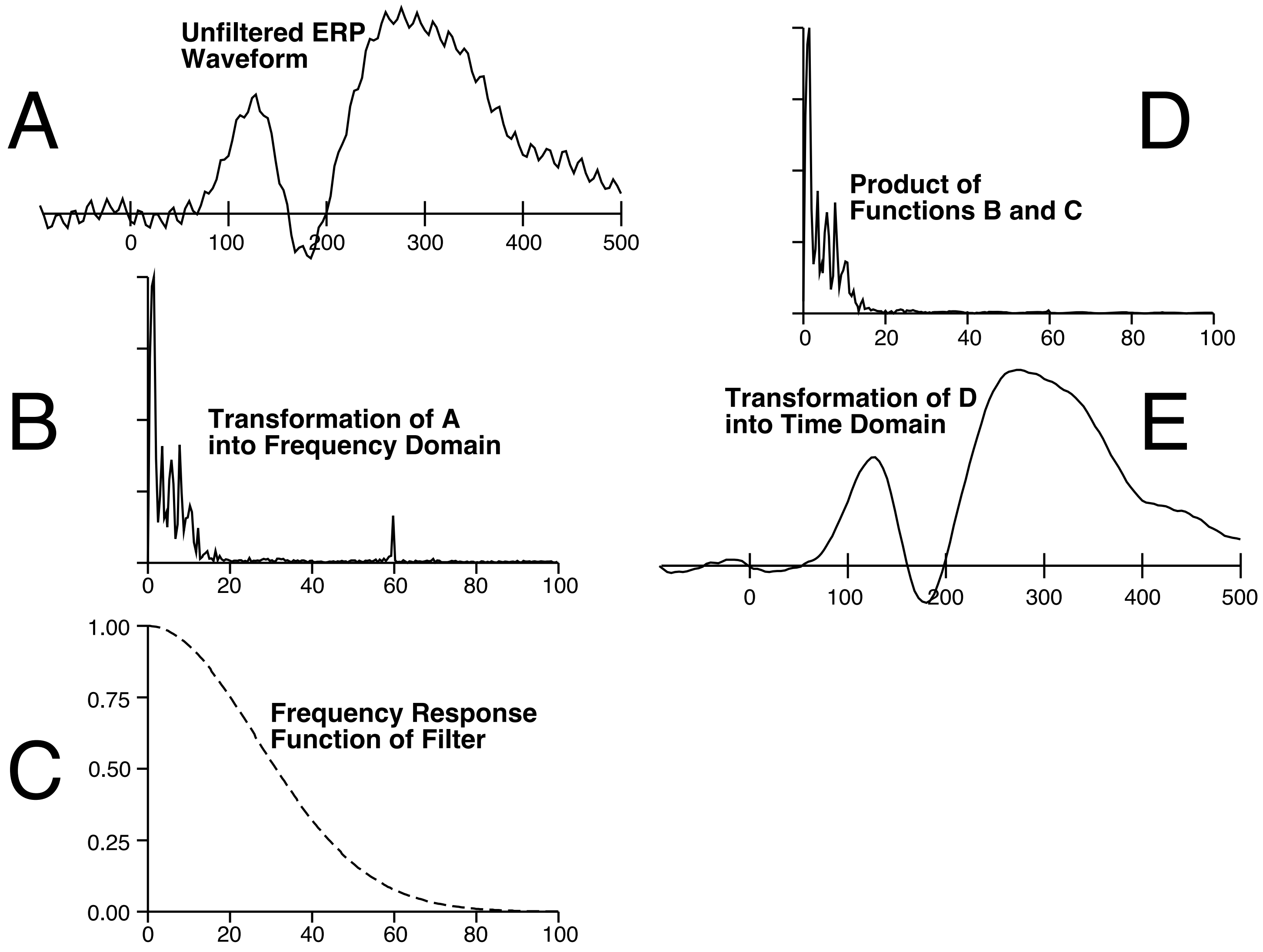

Figure 12.1. Frequency-based conceptualization of filtering. (A) Unfiltered ERP waveform, contaminated by substantial noise at 60 Hz. (B) Transformation of the waveform from Panel A into the frequency domain by means of the Fourier transform. Note the clear peak at 60 Hz. (C) Frequency response function of a low-pass filter that can be used to attenuate high frequencies, including the 60 Hz noise. (D) Product of the waveforms in Panels B and C; for each frequency point, the magnitude in Panel B is multiplied by the magnitude in Panel C. Note that Panel D is nearly identical to Panel B in the low frequency range, but it falls to zero at high frequencies. (E) Transformation of the waveform from Panel D back into the time domain by means of the inverse Fourier transform. The resulting waveform closely resembles the original ERP waveform in Panel A, except for the absence of the 60 Hz noise. Note that the phase portion of the frequency-domain plots has been omitted here for the sake of simplicity. Also, Y axis units have been omitted to avoid the mathematical details, except for Panel C, in which the Y axis represents the gain of the filter (the scaling factor for each frequency). Note that this is the same as Figure 7.1 in Chapter 7 and is subject to the same copyright.

Note that it is possible to use any function you want for the frequency response function. For example, you could set all of the odd-numbered frequencies to 1 and all of the even-numbered frequencies to 0, or you could use the profile of the Rocky Mountains near Banff to determine the scaling factors. In practice, however, frequency response functions are usually smoothly descending (low-pass filters), smoothly ascending (high-pass filters), flat at 1.0 except for a notch (notch filters), or flat at 0.0 at the lowest and highest frequencies with a single peak at intermediate frequencies (band-pass filters).

The discussion so far has focused on the amplitude of each frequency and has neglected phase. A frequency-domain representation actually has two parts, one representing the amplitude at each frequency and the other representing the phase at each frequency. Filters may shift the phases of the frequencies as well as modulating their amplitudes, and to fully characterize a filter in the frequency domain, it is necessary to specify both the frequency gain and the phase shift at each frequency. It is usually desirable for the phase portion of the transfer function to be zero for all frequencies, thereby leaving the phases from the ERP waveform unchanged. This is usually possible when digital filters are applied off-line, but is impossible when analog filters are applied on-line (e.g., when low and high frequencies are removed from the EEG during data acquisition). Because this chapter focuses on digital filters with no phase shift, I won't say much more about phase.

A Problem with Frequency-Domain Representations

Although it is mathematically convenient to conceptualize filtering as a frequency-domain procedure, this approach has an important shortcoming when applied to transient ERP waveforms. The difficulty stems from the fact that time-domain waveforms are represented in Fourier analysis as the sum of a set of infinite-duration sine waves, whereas transient ERP waveforms are finite in duration. Although the Fourier representation is accurate in the sense that ERPs can be transformed between the time and frequency domains with no loss of information, it is inaccurate in the sense that ERPs actually consist of finite-duration voltage deflections rather than sets of infinite-duration sine waves. The biologically unrealistic nature of the frequency-domain representation leads to a number of problems that will be discussed in this chapter.

At this point, let's consider a particularly extreme case in which activity at precisely 5 Hz is filtered from an ERP waveform. Completely suppressing the 5 Hz component of an ERP waveform would be equivalent to computing the amplitude and phase in the 5 Hz frequency band and then subtracting a sine wave with this amplitude and phase from the time-domain ERP waveform. After subtraction of this infinite-duration sine wave, the resulting waveform would contain a 5 Hz oscillation during intervals where the unfiltered ERP waveform was flat, such as the pre-stimulus interval. This is counterintuitive, because filtering out the activity at 5 Hz actually creates a 5-Hz oscillation in the prestimulus interval. Moreover, because the response to a stimulus obviously cannot precede the stimulus in time, this prestimulus oscillation reflects an impossible state of affairs. Thus, because Fourier analysis represents transient ERP waveforms as the sum of infinite-duration sine waves, which is a biologically incorrect representation, the use of analytical techniques based on frequency-domain representations may lead to serious distortions.

To understand how filters may distort your time-domain ERP waveform, you should remind yourself of the following principle, which was described previously in Chapter 7:

Fundamental Principle of Filtering: Precision in the time domain is inversely related to precision in the frequency domain.

This principle is illustrated in Figure 12.2. Panel A of this figure shows a frequency-domain waveform that has a value of 1.0 at 5 Hz and a value of 0 at all other frequencies, becomes an infinite-duration 5-Hz sine wave when it's converted into the time domain. The waveform is very precise (narrow) in the frequency domain, but the same data becomes infinitely imprecise (broad) when converted to the time domain. Panel B shows the converse pattern: A time-domain waveform with a value of 1 at time zero and 0 at every other time contains an equal amount of every frequency when transformed into the frequency domain. In this case, a very precise (narrow) waveform in the time domain is equivalent to an infinitely imprecise (broad) waveform in the frequency domain.

Because of this inverse relationship between the time and frequency domains, the use of filters to restrict the range of frequencies in a signal necessarily broadens the temporal extent of the signal. The more narrowly a filter is specified in the frequency domain, the greater the temporal spread. As a result, filtering out a single frequency, as in the 5-Hz filter example described earlier, leads to an infinite spread of the filtered waveform in time. You will see several examples later in this chapter showing how filtering a time-domain waveform causes the waveform to become spread out it time.

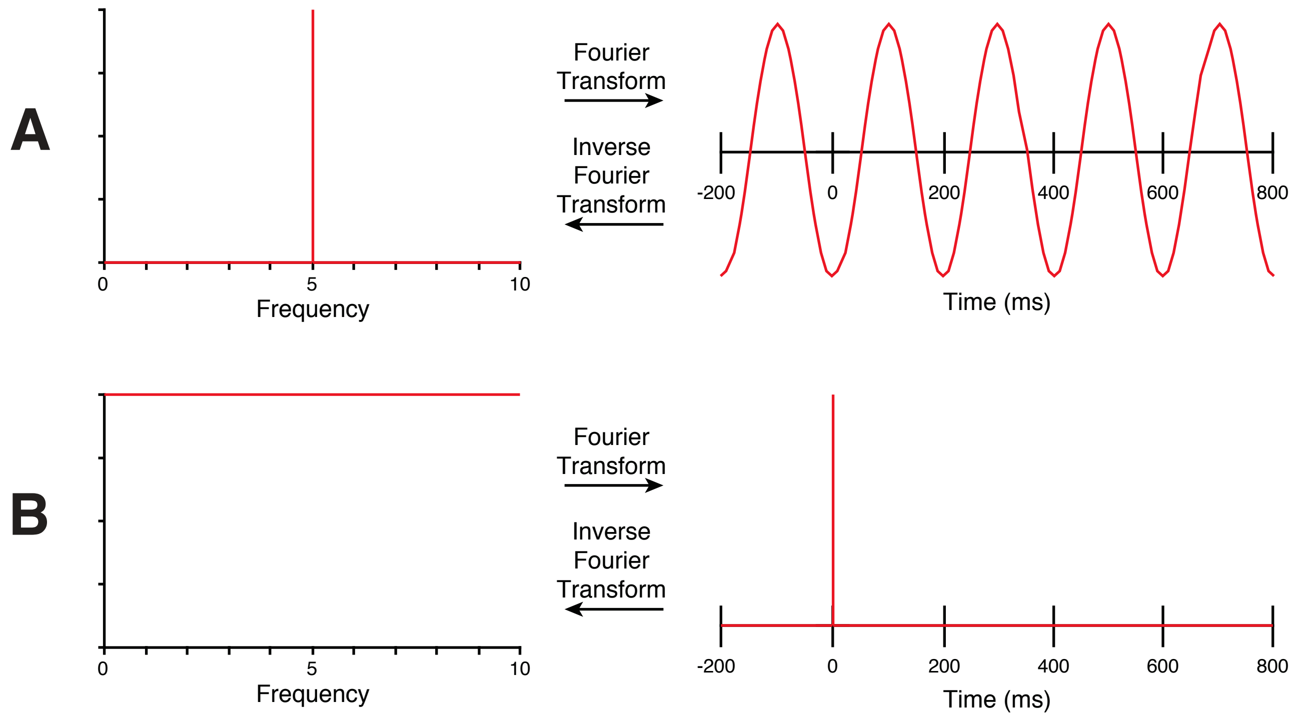

Figure 12.2. Examples of inverse relationship between precision in the time domain and precision in the frequency domain. (A) Frequency-domain waveform with a value of 1.0 at 5 Hz and a value of 0.0 at all other frequencies (left) and time-domain representation of the same data, which is simply a 5 Hz sine wave (right). (B) Time-domain waveform with a value of 1.0 at 0 ms (right) and frequency-domain representation of the same data, which is simply equal amounts of all frequencies (left).

Filtering as a Time-Domain Procedure

We will begin our description of time-domain filtering by considering a common-sense approach to suppressing high-frequency noise, which was previously described in Chapter 7. Consider, for example, the high-frequency noise shown in Figure 12.3A (which is identical to Figure 7.8 from Chapter 7). To attenuate this noise, one could simply average the voltage at each time point with the voltages present at adjacent time points. In other words, the filtered voltage at time point *n* would be computed as the average of the unfiltered voltages at time points *n*-1, *n*, and *n*+1. Figure 12.3B shows the results of applying such a filter to the ERP waveform in Figure 9.4A; this simple filter has clearly eliminated much of the high frequency noise from the waveform. To filter a broader range of frequencies, this filtering technique can be extended by simply averaging together a greater number of points. Figure 12.3C shows the results of averaging 7 adjacent points together instead of the 3 averaged together for Figure 12.3B, and this can be seen to further reduce the high frequency content of the waveform. This method of filtering can be formalized by the following simple formula:

$$fERP_i = \sum_{j=-n}^{n} wERP_{i+j} \qquad \text{[Equation 12.1]}$$

where:

$fERP_i$ is the filtered ERP waveform at time i

ERP_i is the unfiltered ERP waveform at time i

n is the number of points to be averaged together on each side of the current time point

$w = \frac{1}{2n+1}$ (which is the weighting value)

This equation states that the filtered voltage at a given time point is computed by multiplying each of the n voltages on either side of the current time point and the value at the current time point by the weighting value w and then adding together these weighted values. This is equivalent to averaging these $2n + 1$ points (n points before the current point + n points after the current point + the current point = $2n + 1$).

It is worth spending some time to make sure that you understand Equation 12.1. Once you understand it, you will have understood the essence of digital filtering.

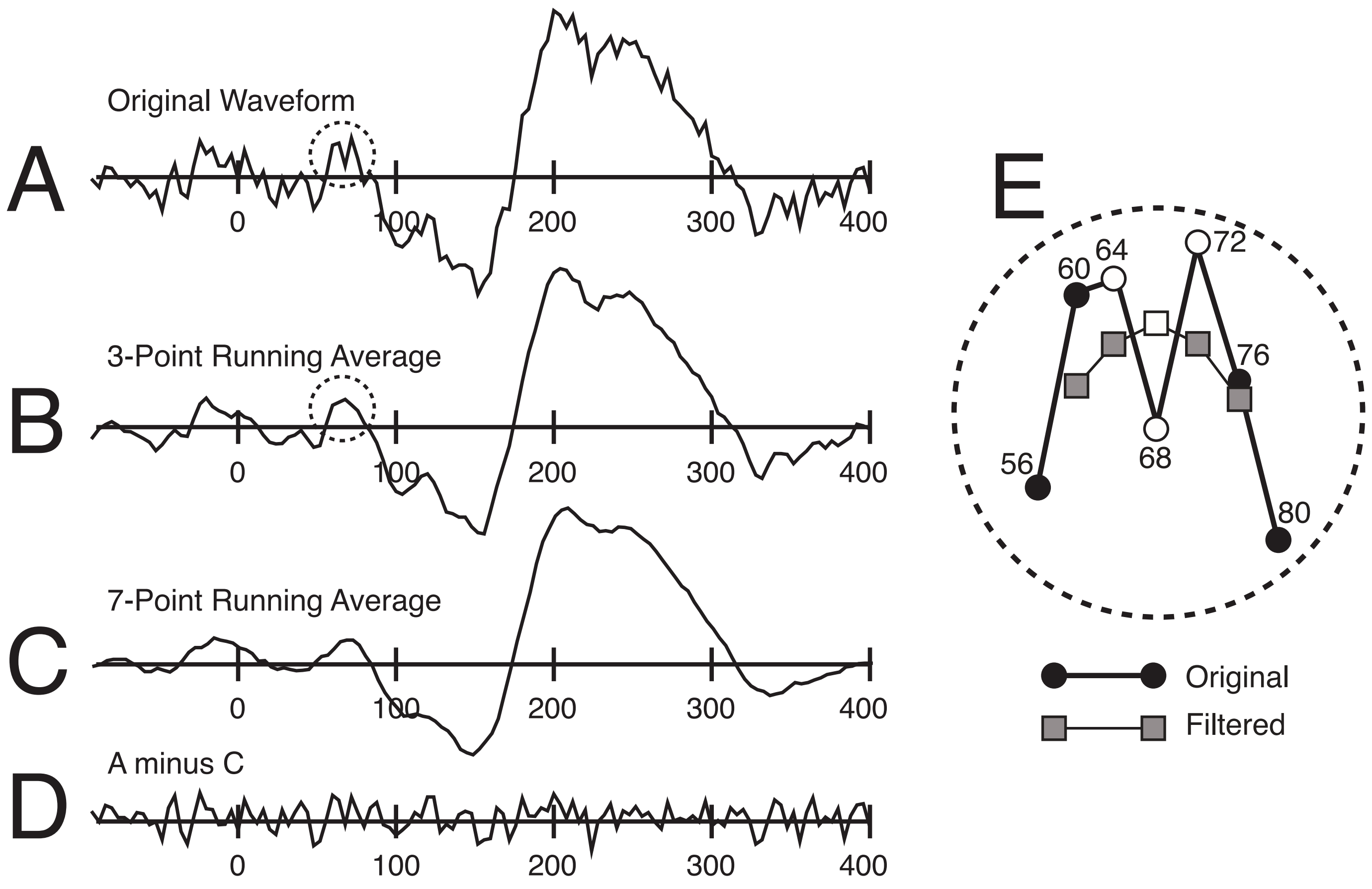

Figure 12.3. Example of filtering an ERP waveform with a running average filter, which works by averaging together the voltages surrounding each time point. (A) Unfiltered ERP waveform, contaminated by substantial high-frequency noise. (B) Result of filtering the waveform in Panel A by averaging the voltage at each time point with the voltages at the immediately adjacent time points (a 3-point running average filter). (C) Result of filtering the waveform in Panel A by averaging the voltage at each time point with the voltages at the 3 time points on either side (a 7-point running average filter). (D) High-pass-filtered waveform, constructed by subtracting the filtered waveform in Panel C from the unfiltered waveform Panel A. (E) Close-up of the original and filtered data between 56 and 80 ms. The original values are indicated by circles, and the filtered values are indicated by squares. Each number represents the time point of the value. The three white-filled circles were averaged together to create the white-filled square. Note that this is the same as Figure 7.8 in Chapter 7 and is subject to the same copyright.

This filtering technique can be extended in a straightforward manner to attenuate low frequencies instead high frequencies. The unfiltered waveform is equal to the sum of its high frequency components and its low frequency components; as a result, the low frequencies can be filtered out by simply subtracting the low-pass-filtered waveform from the unfiltered waveform. The result of this form of high-pass filtering is shown in Figure 12.3D, which is equal to the waveform in Figure 12.3A minus the waveform in Figure 12.3C.

When filtering is accomplished by simply averaging together the $2n + 1$ points surrounding each time point, all of the time points being averaged together contribute equally to the filtered value at the current time point, and this reduces the temporal precision of the filtered waveform. To mitigate this problem, one can use a weighted average that emphasizes nearby time points more than distant time points. For example, a 3-point filter might use weights of 0.25 for the two adjacent points and a weight of 0.50 for the current point (note that our original filter used equal weights of 0.33 for all three time points). In the general case, we can define an array W that contains $2n + 1$ weights, and recast our filtering formula as:

$$fERP_i = \sum_{j=-n}^{n} W_j ERP_{i+j} \qquad \text{[Equation 12.2]}$$

Our 3-point weighting function would then be defined as:

$$W_{-1} = .25, W_0 = .50, W_{+1} = .25$$

For an equal weighting over $2n + 1$ time points, as in our original formulation, the weighting function would be:

$$W_j = \frac{1}{2n+1}$$

Virtually any conceivable weighting function can be used for filtering, and the shape of this function will determine the properties of the filter. As will be described below, there is a straightforward relationship between the shape of the weighting function and the frequency response function of the filter.

In addition to computing the filtered value at a given time point, it is sometimes useful to consider how the unfiltered value at a given time point influences the filtered values at surrounding time points. If we reverse the weighting function in time, the resulting function represents the effect of the current point on the output of the filter. This reversed weighting function is equivalent to the waveform that would be produced by the filter in response to a momentary voltage spike or *impulse*, and it is therefore known as the *impulse response function* of the filter.

The relationship between the weighting function and the impulse response function is illustrated in Figure 12.4. Panel A of the figure shows a weighting function, and panel B shows the corresponding impulse response function. In addition, panel C shows the output of the filter if the input is a momentary impulse. Note that the output of the filter becomes non-zero before the impulse, which is possible only with an off-line, digital filter. Analog filters that operate in real time have impulse response functions that are zero before the time of the impulse.

Although it may seem more natural to think of filtering in terms of the original weighting function, most mathematical descriptions of filtering rely instead on the impulse response function. One reason for this is that it is possible to measure the impulse response function of an analog filter empirically by providing an impulse for an input and simply measuring the output of the filter. Of course, the impulse response function is identical to the weighting function if the function is symmetrical about time zero, which is usually the case with digital filters, making the two conceptualizations of filtering identical.

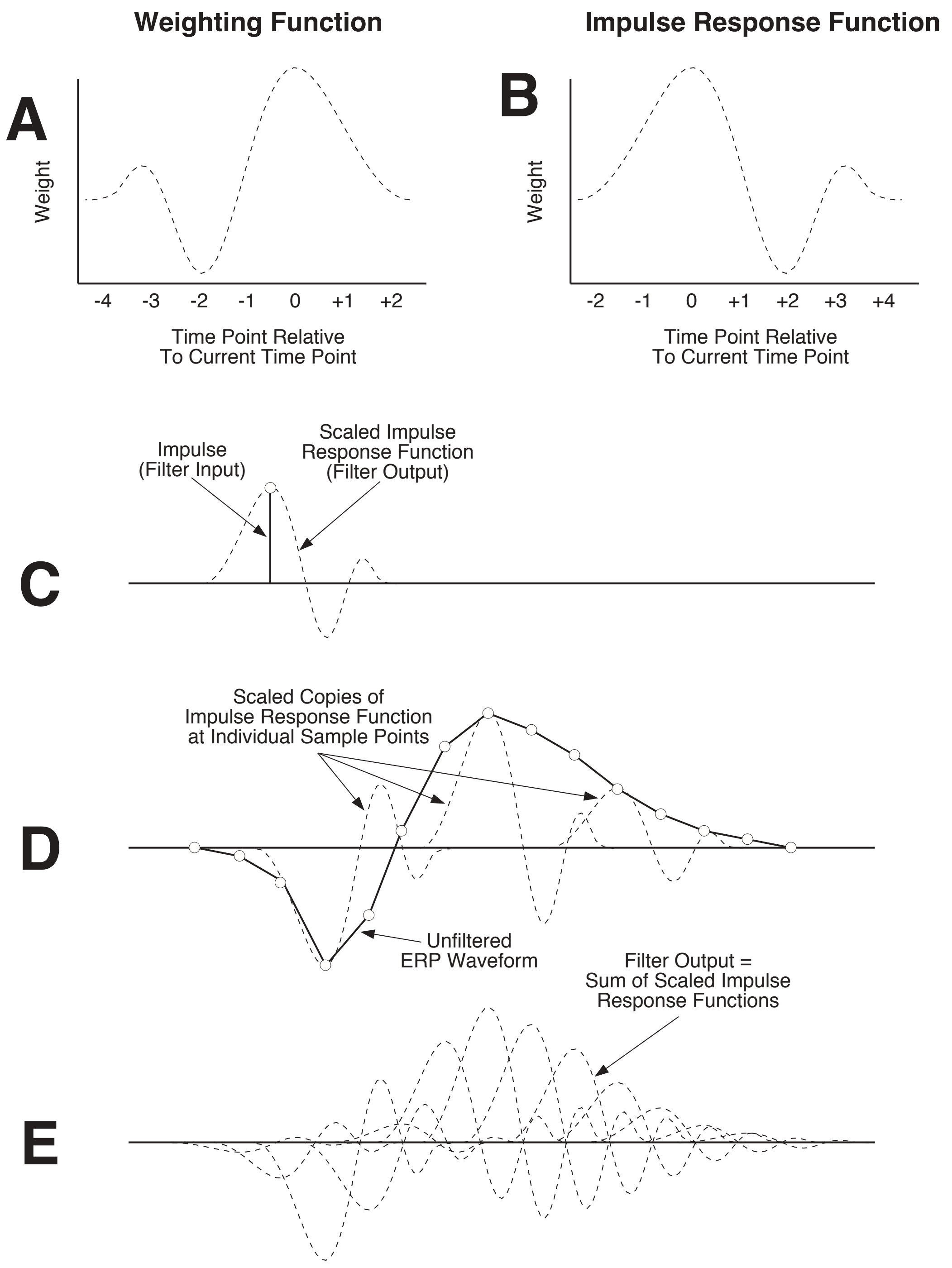
Weighting Function
Impulse Response Function
A
B
Weight
Weight
-4 -3 -2 -1 0 +1 +2
-2 -1 0 +1 +2 +3 +4
Time Point Relative To Current Time Point
Time Point Relative To Current Time Point
Impulse (Filter Input)
Scaled Impulse Response Function (Filter Output)
C
Scaled Copies of Impulse Response Function at Individual Sample Points
D
Unfiltered ERP Waveform
Filter Output = Sum of Scaled Impulse Response Functions
E

Figure 12.4. Filtering by convolving an ERP waveform with the filter's impulse response function. The weighting function of the filter (A) is reversed in time to produce the filter's impulse response function (B). As shown in (C), the impulse response function is the output of the filter in response to a brief impulse. ERP waveforms are analogously filtered by treating each sample point as an impulse and replacing each sample point with a scaled copy of the impulse response function, as shown in (D) and (E). In (D), the unfiltered ERP waveform is represented by the solid waveform, and each open circle represents a sample point; to illustrate the scaling process, the scaled impulse response functions corresponding to 3 of the sample points are also shown in this panel. Note that the function is inverted when the ERP voltage is negative. (E) shows the result of replacing every point in the ERP waveform with a scaled copy of the impulse response function; the filtered waveform is equal to the sum of these scaled impulse response functions.

To use the impulse response function instead of the original weighting function for computing the filtered value at each time point in the waveform, it is necessary to make a small change to the formula presented in Equation 5.2:

$$fERP_i = \sum_{j=-n}^{n} IRF_j ERP_{i-j} \qquad \text{[Equation 12.3]}$$

where IRF_j is the value of the impulse response function at time *j*, which is the same as the original weighting function at time *-j*. This equation essentially reverses the coefficients of the impulse response function—thus creating our original weighting function—and then performs the same filtering operation described in Equation 12.2.

When expressed in this manner, the combination of the impulse response function and the ERP waveform is termed a *convolution*, which is typically symbolized in mathematical equations by the * operator. Equation 12.3 can therefore be written as:

$$fERP = IRF * ERP \qquad \text{[Equation 12.4]}$$

This equation states that the filtered ERP waveform is equal to the convolution of the impulse response function and the unfiltered ERP waveform (note that the * symbol is often used to denote multiplication, especially in computer languages; we will use $\times$ to denote multiplication and * to denote convolution).

The filtering equations listed above are written in a manner that makes it easy to see how the filtered value at a given time point is computed from the unfiltered values at surrounding time points. It is also possible to take a complementary approach and compute the contribution of a given unfiltered time point to the filtered waveform. While less useful computationally, this approach more readily allows one to visualize the relationship between a filter's impulse response function and the filtered ERP waveform. In this approach, each point in the unfiltered waveform is replaced by a scaled copy of the impulse response function (scaled by the amplitude of the unfiltered waveform at the current time point). These scaled copies of the impulse response function are then simply added together to compute the filtered ERP waveform.

This approach to filtering is illustrated in Figure 12.4. This figure shows an arbitrary weighting function (panel A), which was designed solely for the purposes of illustration. The impulse response function of the filter (panel B) is equal to the weighting function reflected around time zero. If the input to the filter is a brief impulse, as shown in panel C, then the output of the filter is simply a copy of the impulse response function scaled by the size of the impulse (by definition). Panel D shows an unfiltered ERP waveform that was sampled at each of the points indicated by open circles (the lines connecting the circles are interpolations). This waveform is filtered by replacing each data point with a scaled copy of the impulse response function (for the sake of simplicity, this is shown for only 3 data points in panel D). Panel E shows all of the scaled copies of the impulse response function, which are added together to compute the filtered ERP waveform (not shown). By conceiving of filtering in this manner, it is possible to visualize how the output of a filter is related to its impulse response function, as discussed further below.

It may seem strange that filtering can be achieved by simply replacing each data point with the impulse response function and summing. In fact, this is true only for a subset of filters, called *finite impulse response* filters. For these filters, the filter's output does not feed back into its input, which leads to this simple pattern of behavior. It is possible to design more sophisticated filters that do not obey this rule (called *infinite impulse response* or *recursive* filters), but these filters are unnecessarily complex for the needs of most ERP experiments.

Relationship Between the Time and Frequency Domains

We have now seen how filtering can be accomplished in the frequency domain and in the time domain. These may seem like completely different procedures, but there is a fundamental relationship between the time-domain and frequency-domain approaches to filtering which allows a straightforward conversion between a filter's impulse response function and its transfer function. This relationship is based on an important mathematical principle: *Multiplication in the frequency domain is equivalent to convolution in the time domain.* As a result of this principle, convolving an ERP waveform with an impulse response function in the time domain is equivalent to multiplying the frequency-domain representation of the ERP waveform with the frequency-domain representation of the impulse response function.

This implies an important fact about filters, namely that the Fourier transform of a filter's impulse response function is equal to the filter's transfer function (remember that the transfer function is the combination of a frequency response function and a phase response function). It is therefore possible to determine a filter's transfer function by simply transforming its impulse response function into the frequency domain by means of the Fourier transform. Conversely, the inverse Fourier transform of a transfer function is equal to the impulse response function of the filter. As a result, one can compute the impulse response function that will yield a desired transfer function by simply transforming the transfer function into the time domain. This is illustrated in Figure 12.5.

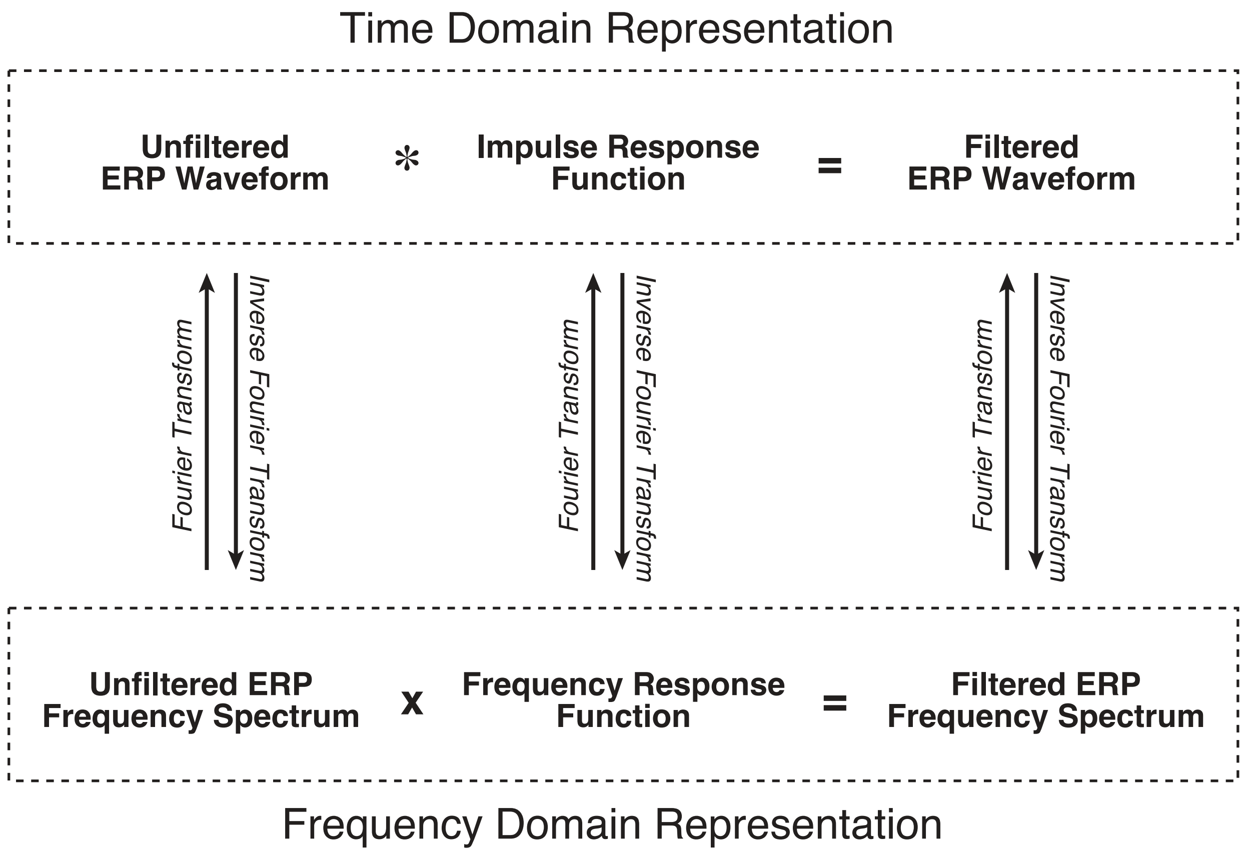
Time Domain Representation
Unfiltered ERP Waveform
*
Impulse Response Function
=
Filtered ERP Waveform
Fourier Transform
Inverse Fourier Transform
Fourier Transform
Inverse Fourier Transform
Fourier Transform
Inverse Fourier Transform
Unfiltered ERP Frequency Spectrum
x
Frequency Response Function
=
Filtered ERP Frequency Spectrum
Frequency Domain Representation

*Figure 12.5. Relationship between filtering in the time and frequency domains, showing: (1) that each term in the time domain can be converted to a corresponding term in the frequency domain by means of the Fourier transform; and (2) that convolution (represented by the * operator) in the time domain is equivalent to multiplication (represented by the X operator) in the frequency domain.*

It is often faster to filter via convolutions than via Fourier transforms, and because the two methods yield identical results, most digital filters are implemented by means of convolutions. The method usually recommended for constructing filters is therefore to create the desired transfer function for the filter and then transform this function into the time domain to create an impulse response function; this impulse response function can then be convolved with the ERP waveform to create the filtered ERP waveform[3]. However, although this approach yields very nice results in the frequency domain, it may lead to significant distortions in the time domain, as will be described below.

As an example of the relationship between the impulse response function and the corresponding frequency response function, consider how a filter would be constructed to pass the 60 Hz frequency band in an ERP

[3] The process is actually somewhat more complex than this due to the fact that both the ERP waveform and the impulse response function are finite in duration and digitally sampled.

waveform and eliminate all other frequencies. The frequency response function of such a filter would have a magnitude of 1.0 at 60 Hz and a magnitude of 0.0 at all other frequencies. Transferring this function into the time domain to derive the impulse response function of the filter would simply yield a sine wave at 60 Hz, and convolving a 60 Hz sine wave with an ERP waveform would therefore extract the 60 Hz component from the waveform.

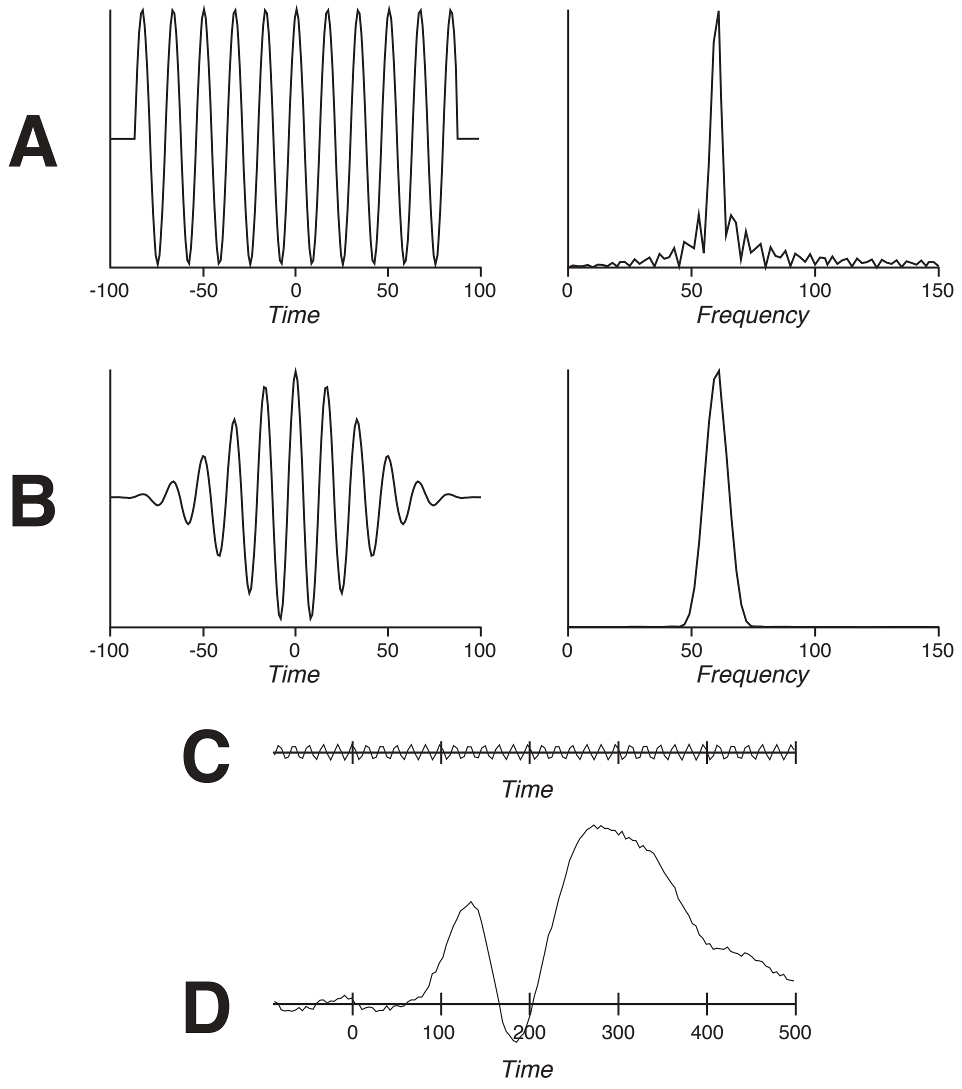

Figure 12.6. Relationship between the time and frequency domains for 60-Hz filtering. (A) Time- and frequency-domain representations of a finite-duration 60 Hz sine wave. (B) Time- and frequency-domain representations of a windowed sine wave (tapered with a Blackman window). (C) Result of convolving the 60-Hz sine wave in (A) with the ERP waveform shown in Figure 9.3A, which extracts the 60-Hz component of the waveform. (D) Effect of removing the 60-Hz component shown in (C) from the unfiltered ERP waveform, which effectively eliminates the 60-Hz artifact from the waveform without attenuating the higher or lower frequencies.

In practice, filtering everything except the 60-Hz activity would be complicated by the fact that the impulse response function must be finite in duration, as illustrated in Figure 12.6. Panel A of the figure shows how a 60 Hz sinusoidal impulse response function of finite length yields a frequency response function that contains a peak at 60 Hz but also contains power at a broad set of higher and lower frequencies. The power at these other frequencies is due to the sudden transitions at the beginning and end of the impulse response function, which is unavoidable with a sine wave of finite duration. As shown in panel B, this problem can be mitigated by multiplying the sinusoidal impulse response function by a windowing function to taper the ends of the function. This yields a frequency response function that, while somewhat broader around 60 Hz, is smoother and falls to zero sooner than the frequency response function of the pure sinusoid. Panel C shows the results of convolving this windowed sinusoid with the ERP waveform shown in Figure 12.3A, which effectively extracts the 60 Hz component from the ERP waveform. Panel D of Figure 12.6 shows how subtracting this 60 Hz component from the original waveform provides a means of eliminating the 60 Hz noise from the waveform. Unlike the filter used in Figure 12.3B to attenuate all high frequencies, this filter has removed only frequencies around 60 Hz, allowing information at both lower and higher frequencies to remain in the filtered waveform.

Time-Domain Distortions Produced By Notch Filters

In the above examples, we have seen how noise can be attenuated from ERP waveforms by means of filtering. However, transient ERP waveforms necessarily contain a broad range of frequencies, and filtering a restricted range of frequencies from a waveform consisting of both noise and an ERP response will attenuate those frequencies from both the noise and the ERP signal. This will almost always lead to some distortion of the time-domain ERP waveform, and the distortion may range from mild to severe depending on the nature of the impulse response function and the ERP waveform.

To illustrate the distortion produced by filtering, Figure 12.7 shows the effects of applying several types of notch filters to an artificial ERP waveform consisting of one cycle of a 10 Hz sine wave. Panel A of this figure shows the artificial ERP waveform and its Fourier transform. Note that although the ERP waveform consists of a portion of a 10 Hz sine wave, it contains a broad range of frequencies because it is restricted to a narrow time interval (remember, a narrow distribution over time corresponds to a broad distribution over frequencies, and vice versa). If we apply a filter to remove the 10 Hz component, therefore, we will not eliminate the entire ERP waveform, but only the 10-Hz portion of it. This is illustrated in Figure 12.7B, which shows the impulse response function for a 10 Hz notch filter, the filter's frequency response function, and the results of applying this filter to the waveform shown in Figure 12.7A.

The impulse response function was created by windowing a 10 Hz sine wave, as in Figure 12.6B, and then converting it into a filter that removes rather than passes power at 10 Hz (this will be described in greater detail shortly). The resulting impulse response function has a positive peak at time zero, surrounded on both sides by oscillations at ~10 Hz that are 180 degrees out of phase with the artificial ERP waveform; these opposite-phase oscillations cause power at ~10 Hz to be subtracted away from the original ERP waveform. When applied to the artificial ERP waveform, the peak amplitude of the waveform is therefore reduced, but the oscillation in the impulse response function is carried over into the filtered waveform, where it can be seen both in the prestimulus interval and in the time period following the end of the original response. The frequency spectrum of the filtered ERP waveform has a sharp drop to zero power at 10Hz, but the nearby frequencies still have significant power, and these nearby frequencies are the source of the oscillation that can be observed in the filtered waveform. Thus, a filter designed to eliminate the 10 Hz component from a waveform can actually produce an output containing artificial oscillations near 10 Hz that weren't present in the input. Although these oscillations are more extreme than those that would be produced by more typical combinations of filters and ERP waveforms, this example demonstrates that impulse response functions that contain oscillations can induce artificial oscillations in the filtered ERP waveform. This important fact is often overlooked when filtering is considered as a frequency-domain operation rather than a time-domain operation.

The presence of oscillations in the impulse response function is not alone sufficient to produce large oscillations in the output of the filter. As shown in Figures 12.7C and 12.7D, notch filters at 20 Hz and 60 Hz produce much smaller oscillations than the 10 Hz notch filter in the context of this particular artificial ERP waveform. This is due to the fact that there is much less power at these frequencies in the spectrum of the unfiltered ERP waveform (see Figure 12.7A). Thus, you must consider both the impulse response function and the nature of the unfiltered ERP waveform to determine the distortion that a filter will produce.

From these filter-induced distortions, it should be clear that you must know the shape of the impulse response function to assess the distortions that a filter might produce. For example, a filter that is simply labeled "60 Hz notch filter" on an EEG amplifier will have a very different impulse function from the filter shown in Figure 12.7D, and may lead to much greater distortion of the ERP waveform than the minimal distortion present in Figure 12.7D. Thus, the common practice of specifying only the half-amplitude or half-power cutoff of a filter is clearly insufficient; descriptions of filtering should specify the impulse response function in addition to the cutoff of the filter. For example, when I describe a filter in a journal article, I write something like this: "The ERP waveforms were low-pass filtered offline by convolving them with a Gaussian impulse response function with a standard deviation of 6 ms and a half-amplitude cutoff at ~30 Hz."

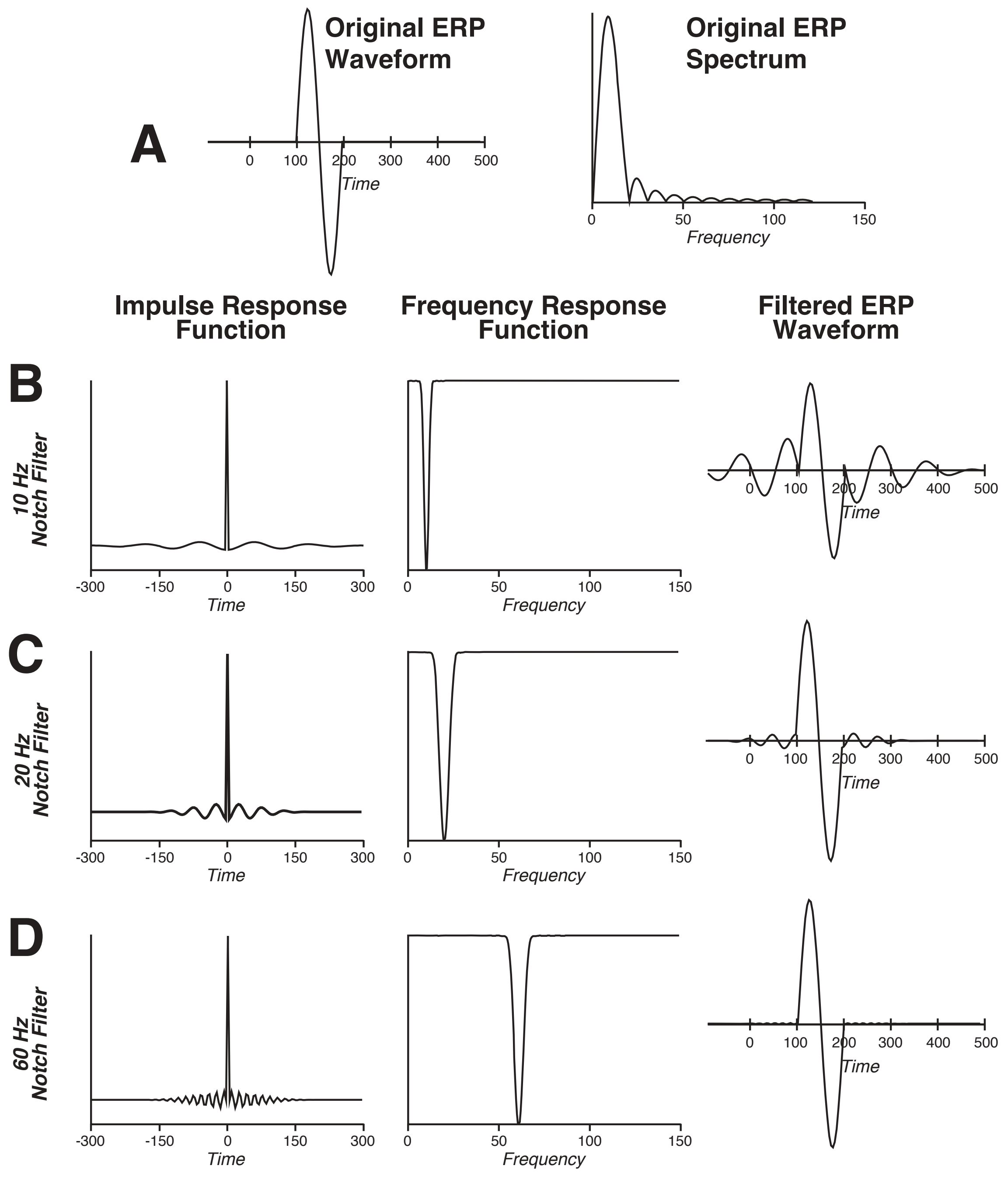

Figure 12.7. Examples of temporal distortions produced by notch filters. The original artificial ERP waveform (one cycle of a 10 Hz sine wave) and its Fourier transform are shown in (A). (B), (C), and (D) show the impulse response and frequency response functions for 10-, 20-, and 60-Hz notch filters and also display the result of applying these filters to the waveform shown in (A).

Distortions Produced By Low-Pass Filters

In many cases, it is useful to attenuate all frequencies above some specified point (low-pass filtering). Such filtering is necessary during digitization of the raw EEG data, because you must sample at a rate that is at least twice as high as the highest frequency in the incoming data. Low-pass filtering is also useful for attenuating the relatively high-frequency noise caused by muscle activity or by external electrical devices (e.g., line-frequency noise at 50 or 60 Hz). The cutoff frequency used for a given experiment will depend on the frequency content of the ERPs being recorded and the frequency content of the noise to be filtered. For example, brainstem evoked responses contain substantial power at very high frequencies, making a 5 KHz cutoff appropriate, but this makes it difficult to filter out muscle noise, which falls into the same frequency range. In contrast, the long-latency ERP components consist primarily of power under about 30 Hz, making a 30-100 Hz cutoff appropriate and allowing high-frequency muscle activity to be filtered without much distortion of the underlying ERP waveform. However, there is almost always some overlap between the frequencies in the ERP waveform and in the noise, and some filter-induced distortion is therefore inevitable.

This leads us once again to Hansen's Axiom: *There is no substitute for good data*. Filters necessarily cause distortion, and it is always better to reduce noise at the source (e.g., through shielding, low electrode impedance, high common-mode rejection, careful experimental design, etc.) than to attenuate it with filters. When filtering cannot be avoided, it is important to choose a filter with an optimal balance between suppression of noise and distortion of the ERP waveform. Depending on the nature of the experimental question being addressed, some types of distortion may be more problematic than others, and there are therefore no simple rules for designing an optimal filter.

Most discussions of filter design emphasize the frequency domain and therefore lead to filters such as the *windowed ideal* filter shown in Figure 12.8A. This particular windowed ideal filter has a half-amplitude cutoff at 12.5 Hz and is an "ideal" filter because it perfectly passes all frequencies below approximately 12 Hz and completely suppresses all frequencies above approximately 13 Hz, with only a narrow "transition band" in which the attenuation is incomplete. This filter also produces no phase distortion because its impulse response function is symmetric around time zero. Despite the usefulness of these attributes, the sharp transitions in the frequency response function of this filter require a broad, oscillating impulse response function, which leads to substantial distortion in the time domain. This can be seen when this filter is applied to the artificial ERP waveform from Figure 12.7A, yielding an output containing damped oscillations both before and after the time range of the original ERP waveform (see Figure 9.9A, right column). Importantly, the filtered waveform contains peaks at around 70 ms and 230 ms that are completely artifactual and are a consequence of the oscillations in the filter's impulse response function. Thus, filters with sharp cutoffs in their frequency response functions may lead to large distortions in the apparent onset and offset of an ERP waveform and to artifactual peaks that are not present in the original ERP waveform. In other words, an filter that is ideal in terms of its frequency-domain properties may be far from ideal in terms of its time-domain properties.

Let us consider briefly how this type of distortion might influence the interpretation of an ERP experiment. As an example, consider the ERP waveform elicited by a visual stimulus over frontal cortex, which typically contains an N1 component at about 130 ms but has no prior peaks (the earlier P1 component is typically absent at frontal sites). If this waveform were filtered with the low-pass filter shown in Figure 12.8A, the filtered waveform would contain an artifactual positive peak preceding the N1 component and peaking at about 70 ms post-stimulus, and this artifactual peak might be mistaken for the P1 component. Moreover, if two conditions were compared, one of which produced a larger N1 component than the other, these amplitude differences would also be observed in the artifactual pseudo-P1 component. On the basis of the filtered response, one might conclude that the experimental manipulation caused an increase in P1 amplitude at 70 ms even though there was no real P1 component and the experimental effect did not actually begin until 100 ms. This example underscores the necessity of knowing the impulse response function of a filter to avoid making incorrect conclusions about the time course of ERP activity.

The bottom three panels of Figure 12.8 show several alternative filters that have approximately the same 12.5-Hz half-amplitude cutoff frequency as the windowed ideal filter shown in the top panel, but produce substantially less waveform distortion. The first of these is a simple running average filter that is equivalent to averaging successive points together (as described in the initial discussion of low-pass filtering). The impulse response function of this filter extends over a very narrow time range compared to the windowed ideal filter, and therefore causes much less temporal spread in the filtered ERP waveform. As can be seen in the right column of Figure 12.8B, the output of the running average filter begins slightly before and ends slightly after the original waveform, but the filter causes relatively little change in the overall shape of the waveform. This filter has two shortcomings, however. First, there is substantial attenuation in the 10 Hz frequency band where most of the power of the original ERP waveform is located, so the filtered waveform is somewhat smaller than the original. Second, the frequency response function of the filter does not fall monotonically to zero. Instead, there are *side lobes* that allow substantial noise from some high frequencies to remain in the filtered waveform (this cannot be seen in the examples shown in Figure 12.8, which contain no high-frequency noise). These side lobes can be predicted from the square shape of the impulse response function: because the transfer function is the Fourier transform of the impulse response function, the high frequencies required for the sudden onset and offset of the impulse response function lead to the presence of substantial high frequency power in the frequency response function.

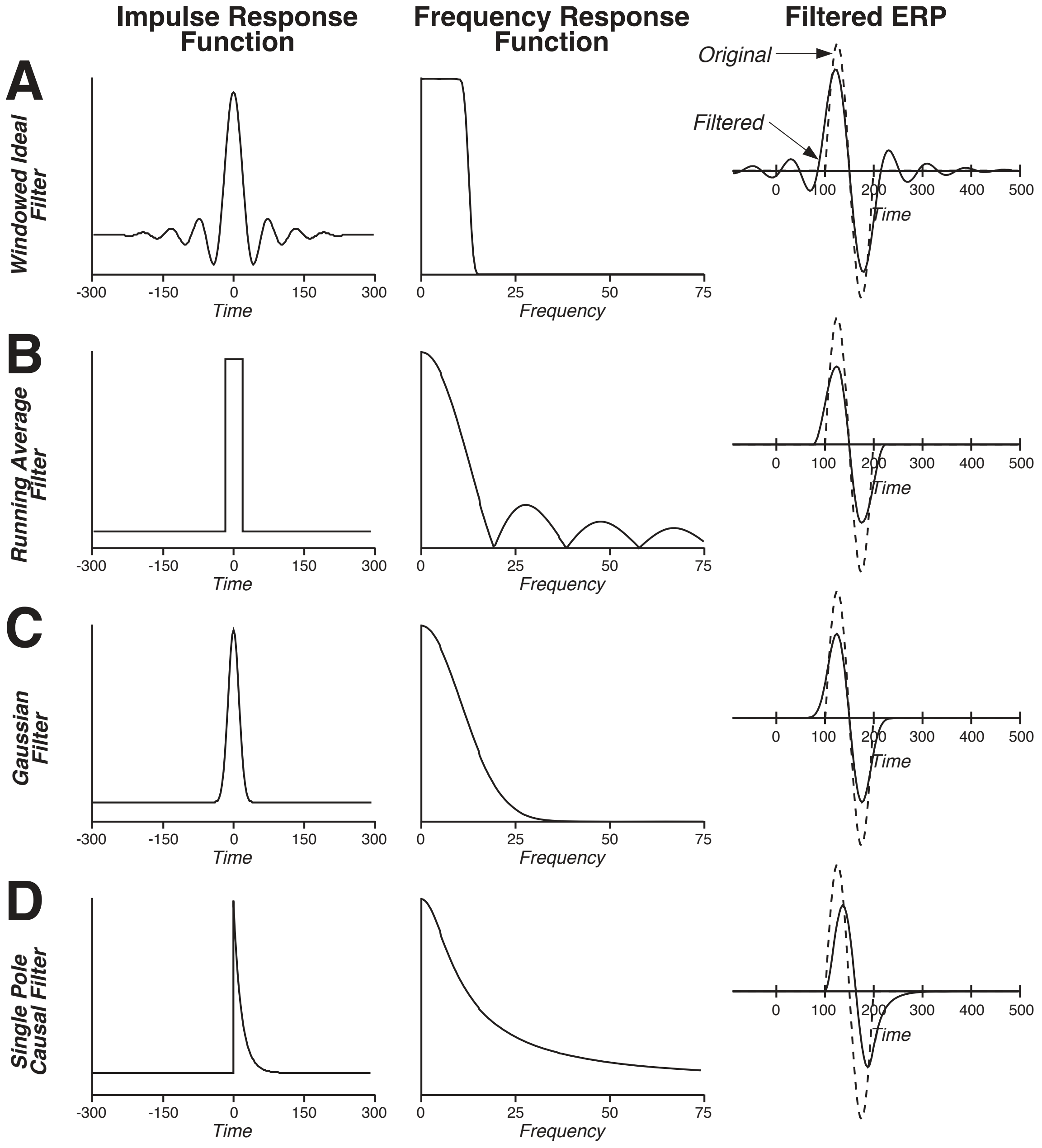

Figure 12.8. Examples of temporal distortions produced by several types of low-pass filters, each with a half-amplitude cutoff of approximately 12.5 Hz. The left and middle columns show the impulse and frequency response functions of the filters, respectively. The right column shows the result of applying each filter to the artificial ERP waveform shown in Figure 12.7A.

Figure 12.8C shows a filter with a Gaussian impulse response function that has approximately the same temporal spread as the running average impulse response function. Like the running average filter, the Gaussian filter produces some smearing in the onset and offset of the filtered ERP waveform and some attenuation of the overall waveform. However, the frequency response function of the Gaussian filter falls monotonically to zero, leading to virtually complete attenuation of frequencies greater than 30 Hz. In most cases, a Gaussian filter provides the best compromise between the time and frequency domains, with a monotonic and fairly rapid fall-off in the frequency response function combined with minimal temporal distortion of the ERP waveform. Typically, a Gaussian filter with a half-amplitude cutoff of 30 Hz will eliminate most line-frequency and muscle noise while producing very little attenuation of the long-latency ERP components and relatively little latency smearing. In the vast majority of cases, I would recommend using Gaussian impulse response functions for low-pass filtering.

The frequency-domain properties of filters are often described by a single number, the half-amplitude cutoff frequency. The time-domain properties of a Gaussian filter can also be described with a single number that reflects the width of the impulse response function. There are two common ways to do this. First, you can indicate the standard deviation of the Gaussian. Second, you can indicate how many milliseconds wide the Gaussian function is at half of its maximum value, which is called the *full width at half maximum* (FWHM). Spatial Gaussian filters are commonly used in fMRI experiments, and FWHM is the usual way of describing the impulse response function in that context. These values can be interconverted with the filter's half-amplitude cutoff. The half-amplitude cutoff in Hz of a Gaussian filter with a standard deviation of σ milliseconds is simply 185.5/σ (e.g., a Gaussian filter with a standard deviation of 4 ms would have a half amplitude cutoff of 185.5/4 Hz). The FWHM in milliseconds is equal to 2.355 times the standard deviation in milliseconds, and the half-amplitude cutoff in Hz can be computed as approximately 437/FWHM.

The windowed ideal, running average, and Gaussian filters described so far are among the most commonly used low-pass digital filters for ERP research, and each has certain advantages and disadvantages that I will now compare. Because of its sharp cutoff, the windowed ideal function may be appropriate when the ERP waveform and the to-be-filtered noise contain substantial power in nearby frequency ranges. However, this filter type leads to substantial distortion in the time domain, which may lead to incorrect conclusions about the timing of an ERP component and may even lead to spurious peaks before the onset or after the offset of the real ERP waveform. In recordings of long-latency ERP components, the noise is usually concentrated at substantially higher frequencies than the majority of the ERP power, making the sharp cutoff of the windowed ideal filter unnecessary, and a Gaussian or running average filter is therefore more appropriate. One exception to this occurs when an ERP waveform is contaminated by alpha-frequency (~10 Hz) EEG noise that interferes with late components such as the P300, which have their power concentrated at slightly lower frequencies. The best way to eliminate alpha noise is usually to maintain subject alertness and average together a large number of trials (consistent with Hansen's Axiom), but this is not always feasible. When necessary, it is possible to use a windowed ideal filter with a half-amplitude cutoff frequency of around 8 Hz to attenuate alpha activity, but the resulting data must be interpreted cautiously because of the time-domain distortions that such a filter will inevitably produce.

The running average and Gaussian filters produce similar patterns of temporal distortion, characterized primarily by the smearing of onset and offset times, but the Gaussian filter is usually preferable because its frequency response function falls to zero at high frequencies. The running average filter is somewhat easier to implement, however, which is important in some applications. In addition, it is sometimes possible to take advantage of the multiple zero points in the frequency response function of the running average filter. If the primary goal of filtering is to attenuate line-frequency noise, it may be possible to employ a running average filter where the zero points fall exactly at the line frequency and its harmonics, producing excellent attenuation of line frequency noise.

The last class of low-pass filters to be discussed here are called *causal* filters and are used in analog filtering devices such as EEG amplifiers (but can also be implemented digitally). These filters are labeled *causal* because they reflect the normal pattern of causation in which an event at a particular time can influence subsequent events but not previous events. More precisely, the impulse response functions of causal filters have values of zero for times preceding time zero. Viewed from the opposite temporal perspective, the output of the filter at a given time point reflects only the input values at previous time points and not the input values at subsequent time points. An example of such a filter is shown in Figure 12.8D, which displays the impulse response function of a very simple analog filter. Filters that do not obey the normal pattern of causation (i.e., because they nonzero values before time zero in their impulse response functions) are called *non-causal* filters.

Because their impulse response functions extend only one direction in time, causal filters produce shifts in peak latency and in offset time, which are typically considered negative traits. However, causal filters may produce relatively little distortion of onset latency, which may provide an advantage over non-causal filters when onset latency is an important variable (see Woldorff, 1993). This is not always true of causal filters, however; it depends on a relatively rapid, monotonically decreasing fall-off from time zero in the impulse response function, as in the filter shown in Figure 12.8D.

Some Important Properties of Convolution

Before going further, it is useful to discuss some important mathematical properties of the convolution operation. In particular, convolution has the same commutative, associative, and distributive properties as multiplication. The commutative property states that it doesn't matter which of two functions is placed first in a convolution formula. More precisely:

$$A * B = B * A$$

The associative property states that multiple consecutive convolutions can be performed in any order. More precisely:

$$A * (B * C) = (A * B) * C$$

The distributive property states that the convolution of some function A with the sum of two other functions B and C is equal to A convolved with B plus A convolved with C. More precisely:

$$A * (B + C) = (A * B) + (A * C)$$

These mathematical properties of convolution lead to several important properties of filters. For example, one common question about filtering is: what happens when a filtered waveform is filtered a second time? If we have two filters with impulse response functions denoted by IRF_1 and IRF_2 and an ERP waveform denoted by ERP, the process of filtering twice can be written as:

$$(ERP * IRF_1) * IRF_2$$

Because of the associative property, this is equal to convolving the two impulse response functions first and then convolving the result with the ERP waveform:

$$ERP * (IRF_1 * IRF_2)$$

To take a concrete example, the convolution of two Gaussian functions yields a somewhat wider Gaussian, and filtering an ERP twice with a Gaussian filter is therefore equivalent to filtering once with a wider Gaussian.

In addition, because convolution in the time domain is equivalent to multiplication in the frequency domain, the frequency response function of the double-filtering is equal to the product of the frequency response functions of the two individual filters. In the case of two Gaussian filters, this would lead to greater attenuation of high frequencies than would be produced by either filter alone. In the more common case of the application of both a low-pass filter and a high-pass filter, the result is a band-pass filter that simply cuts both the high and low frequencies. However, if both filters have relatively gradual frequency response functions, the multiplication of these functions may also lead to fairly substantial attenuation of the intermediate frequencies.

Time-Domain Implementation of High-Pass Filters

In the initial discussion of high-pass filters near the beginning of the chapter, I mentioned that high-pass filtering can be accomplished by creating a low-pass-filtered waveform and subtracting this from the unfiltered waveform, thus yielding the high frequencies that are present in the unfiltered waveform and absent in the low-pass-filtered waveform. This process can be expressed as:

$$ERP_H = ERP - ERP_L = ERP - (IRF_L * ERP)$$

where ERP_H and ERP_L are the high- and low-pass-filtered ERP waveforms and IRF_L is the impulse response function of the low-pass filter. This convolution-and-subtraction sequence can be replaced by a single convolution with a high-pass impulse response function that can be computed by some simple algebraic manipulations. First, it is necessary to define a "unity" impulse response function IRF_U that is 1.0 at time zero and 0.0 at all other points; the convolution of this function with an ERP waveform would be equal to the ERP waveform (analogous to the multiplication of a number by 1.0). By using this unity function, we can use the distributive property to create an impulse response function $\mathrm{IRF_H}$ that will accomplish high-pass filtering in a single step:

$$ERP_H = ERP - (IRF_L * ERP)$$

$$= (IRF_U * ERP) - (IRF_L * ERP) \quad \text{[because ERP = IRF}_\text{U} \text{ * ERP]}$$

$$= (IRF_U - IRF_L) * ERP \quad \text{[because of the distributive property]}$$

$$= IRF_H * ERP \text{, where } IRF_H = IRF_U - IRF_L$$

Thus, we can create a high-pass impulse response function by subtracting a low-pass impulse response function from the unity impulse response function $(IRF_U - IRF_L)$. The frequency response function of the resulting high-pass filter is simply the complement of the frequency response function of the low-pass filter (1.0 minus the response at each individual frequency point in the low-pass frequency response function).

Figure 12.9A diagrams the creation of a high-pass impulse response function from a unity impulse response function and a Gaussian low-pass impulse response function with a half-amplitude cutoff of 2.5 Hz. The unity impulse response function (Figure 12.9A, left) consists of a single spike of magnitude 1.0 at time zero, which is very large compared to the values at individual time points in the Gaussian low-pass impulse response function (Figure 12.9A, middle). The high-pass impulse response function is created by simply subtracting the low-pass impulse response function from the unity function, as shown at the right of Figure 12.9A.

A related attribute of the class of filters discussed here is that they are linear, and filtering can thus be combined with other linear operations in any order (see the Appendix for more information about linear operations). For example, signal averaging is also a linear process, and this means that filtering an averaged ERP waveform yields the same result as filtering the EEG data before averaging[4]. Averaged ERP data sets are typically considerably smaller than the EEG data from which they are derived, and filtering after averaging is therefore more efficient than filtering the raw EEG and then averaging.

[4] Artifact rejection is a nonlinear process, and filtering before artifact rejection and averaging will not yield the same result as filtering after artifact rejection and averaging (although the result will be very similar if the specific filter used doesn't influence the artifact rejection process very much).

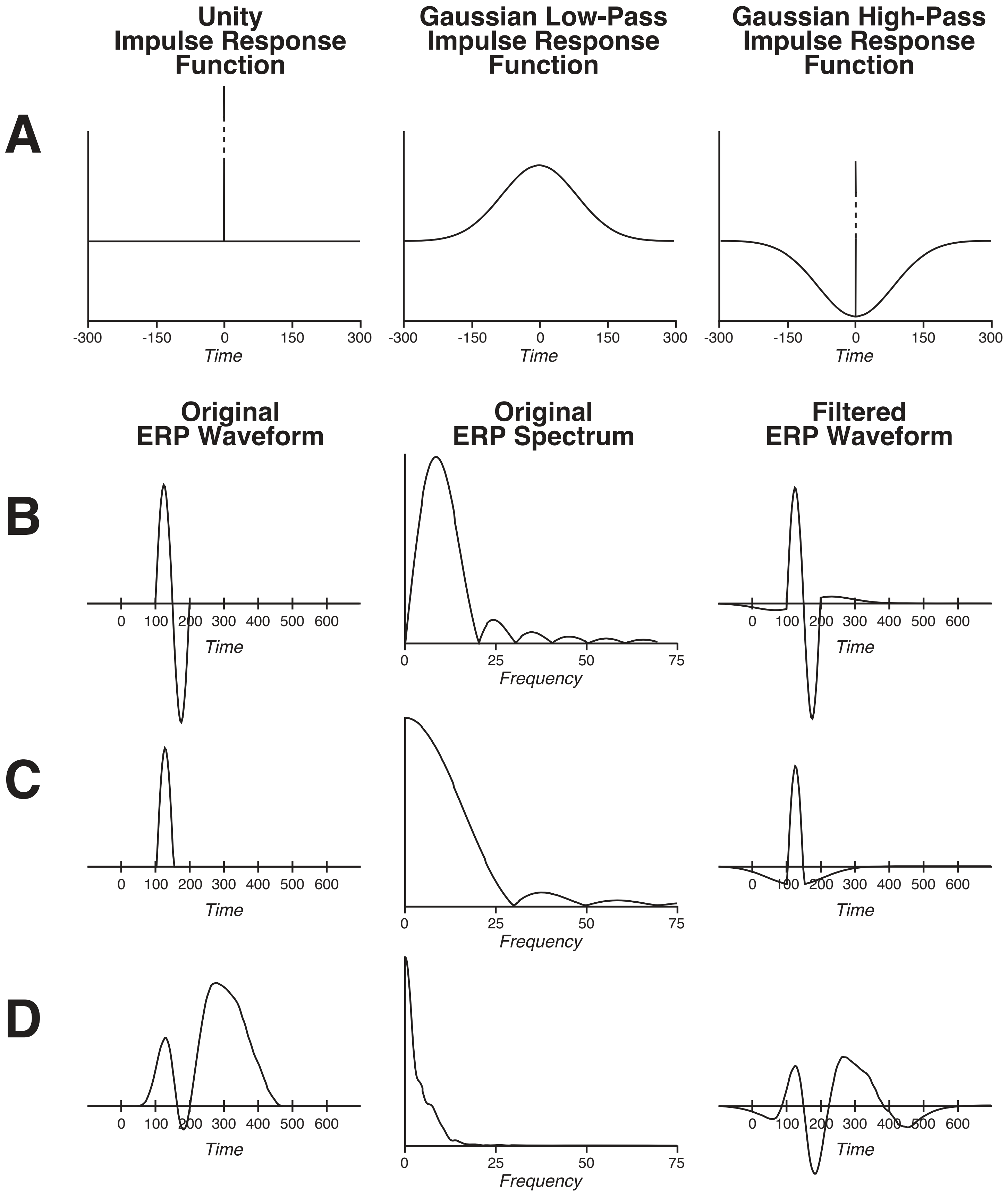

Figure 12.9. Construction and application of a gaussian high-pass filter. (A) A unity impulse response function minus a gaussian low-pass impulse response function yields a gaussian high-pass impulse response function. (B), (C), and (D) show the application of this filter to three different ERP waveforms.

Distortions Produced By High-Pass Filters

Figure 12.9 shows the application of a Gaussian high-pass filter to two artificial ERP waveforms and one realistic ERP waveform. The inverted Gaussian in the impulse response function is visible in the filtered ERP waveforms, where it produces overshoots that can be observed at the beginning and end of the waveforms. These distortions are particularly evident for ERP waveforms that consist primarily of one polarity, such as those shown in Figures 12.9C and 12.9D. This can be understood by viewing filtering as replacing each sample in the ERP waveform with a scaled copy of the impulse response function: when the unfiltered ERP waveform consists of roughly equivalent positive and negative subcomponents, as in Figure 12.9B, these subcomponents will lead to opposite-polarity copies of the impulse response function, which will cancel each other out to some extent. The amount of cancellation will depend on the exact shapes of the waveform and the impulse response function, but in most cases the overshoot will be greater when the ERP waveform is dominated by one polarity.

The distortions in Figure 12.9 are in some sense similar to the distortions produced by the Gaussian low-pass filter shown in Figure 12.8C, but reversed in polarity because the Gaussian in the high-pass impulse response function is inverted (because of the lower cutoff frequency of the high-pass filter, the distortion in Figure 9.10 is also somewhat broader than the distortion in Figure 12.8C). Thus, when an ERP waveform is low-pass filtered, the Gaussian impulse response function produces a spreading of the peaks that is the same polarity as the peaks, whereas opposite-polarity spreading is produced when a high-pass filter is used. Although these distortions are similar in many ways, the distortions produced by the high-pass filter may be more problematic because they may lead to the appearance of artifactual peaks, such as the peaks at approximately 75 ms and 450 ms in the filtered waveform shown in Figure 12.9D.

Figure 12.10 shows the effects of several different types of high-pass filters (all with half-amplitude cutoffs at approximately 2.5 Hz) on a realistic ERP waveform. Although the impulse response and frequency response functions of the windowed ideal and Gaussian filters (panels A and B) look very similar, the windowed ideal impulse response function contains damped oscillations just like those in the windowed ideal low-pass filter show in Figure 12.8A. The half-amplitude cutoff frequency of the high-pass filter shown here is so low, however, that these oscillations fall outside of the plotted time window; with a higher cutoff frequency, these oscillations would be more apparent. Thus, windowed ideal high-pass filters may induce artificial oscillations in the filtered ERP waveform, whereas Gaussian high-pass filters do not. As discussed above and shown in Figure 12.10B, however, Gaussian high-pass filters can produce individual artifactual peaks at the beginning and end of the filtered waveform, unlike Gaussian low-pass filters. For this reason, extreme caution must be taken in the interpretation of high-pass-filtered waveforms, even when a Gaussian impulse response function is used.

Figure 12.10C shows the impulse response and frequency response functions of the type of causal filter found in older EEG amplifiers. Because its impulse response function is zero for all points before time zero, this filter cannot produce any distortion before the onset of the unfiltered ERP waveform and thus does not produce an artifactual peak at the beginning of the filtered ERP waveform. However, this same attribute of causal filters leads to distorted peak latencies in the filtered waveform, whereas this form of distortion is absent for most non-causal filters. These different forms of distortion underscore the important principle that all filters produce distortions, and the choice of which filter to use will depend upon the goals of the experiment and the types of distortion that are acceptable. The filter shown in Figure 12.10C also has a relatively gradual roll-off, and this can be improved by using the filter shown in Figure 12.10D, which uses the right half of a Gaussian in its impulse response function. Like the filter in Figure 12.10C, this "half-Gaussian" filter produces minimal distortion at the beginning of the filter ERP waveform, but has a somewhat sharper roll-off and is therefore a better choice in many cases.

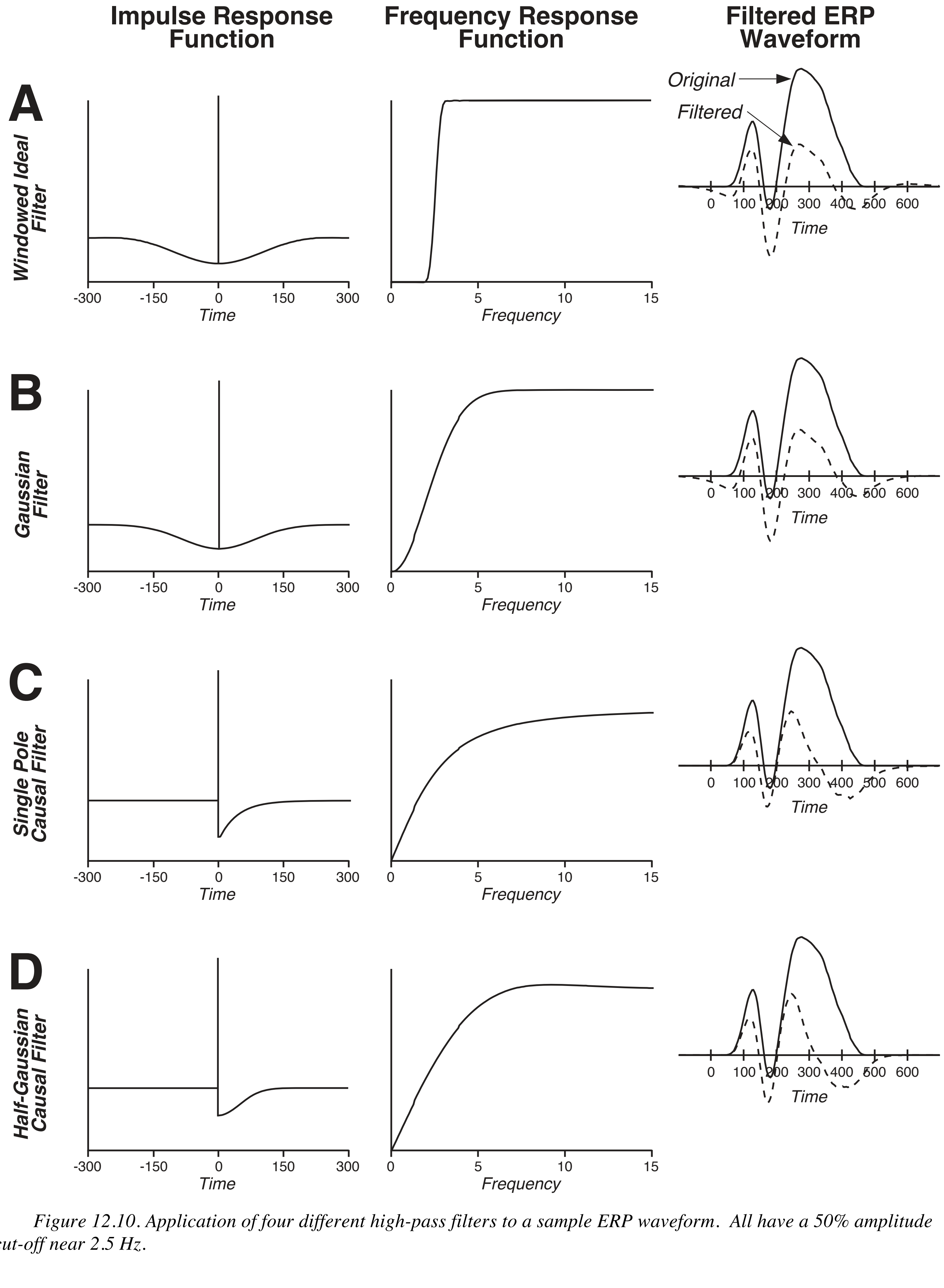

Figure 12.10. Application of four different high-pass filters to a sample ERP waveform. All have a 50% amplitude cut-off near 2.5 Hz.

In most cases, non-causal high-pass filters tend to take low-frequency information away from the time zone of the unfiltered ERP waveform and "push" it forward and backward in time in the filtered ERP waveform. In contrast, causal high-pass filters of the type shown in Figure 12.10 push the information exclusively to later time points. This is illustrated in Figure 12.11. ERP waveforms typically consist of small, relatively high-frequency early components followed by larger, relatively low-frequency late components, and this asymmetrical sequence makes the bi-directional distortions produced by non-causal high-pass filters particularly harmful. In particular, low frequency information from the large, low-frequency later components is pushed into the latency range of the smaller early components, causing substantial distortion of the early components even though they contain relatively little low-frequency information. By using a causal filter like those shown in panels C and D of Figure 12.10, however, distortion of the early components will be minimized because low-frequency information from the late components will not be pushed backward into the time range of the early components. It should be noted, however, that this is true only for filters that have monotonically increasing impulse response functions after time zero; filters that do not possess this property, such as Bessel filters, may cause distortions similar to those produced by non-causal filters. In addition, causal filters may produce substantial latency shifts that may be problematic in some cases.

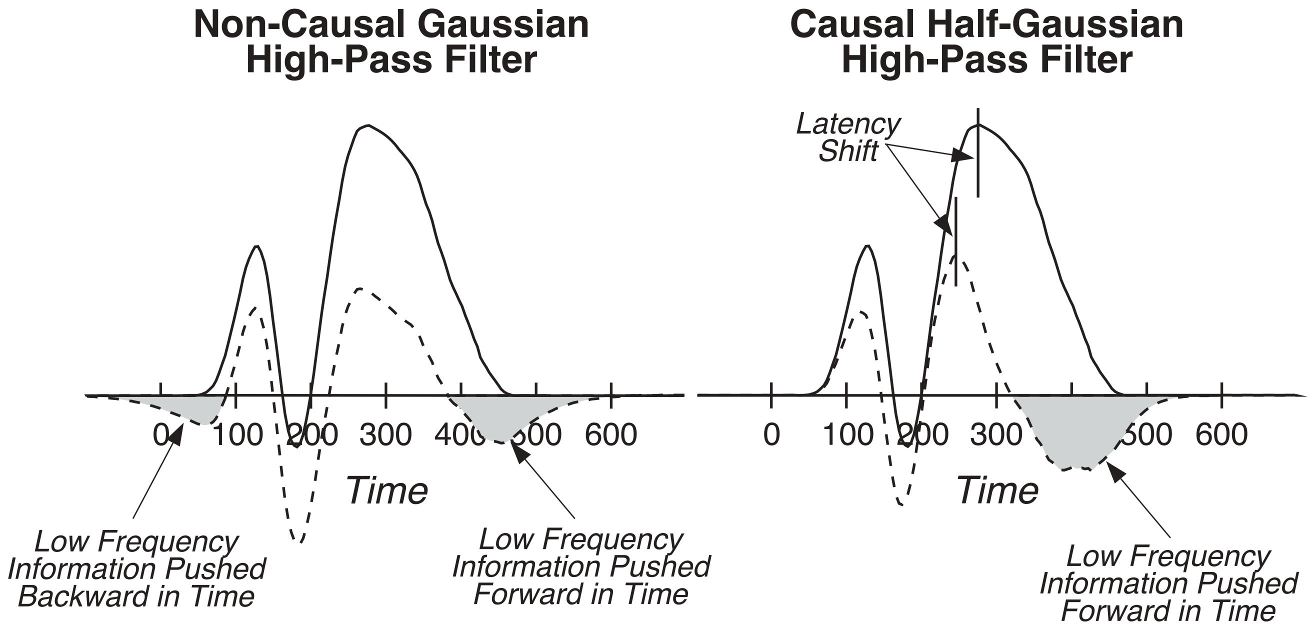

Figure 12.11. Comparison of a non-causal gaussian high-pass filter and a causal half-gaussian high-pass filter. Note that the non-causal filter produces obvious distortions at the beginning and end of the waveform, whereas the causal filter produces no distortion at the beginning of the waveform. Note also that the causal filter produces a large latency shift, but the non-causal filter does not.

Recommendations Revisited

I discussed some recommendations about filtering in Chapter 7. Now that you have a deeper understanding of how filters actually work, you might want to read through those recommendations again.

My two most important recommendations are as follows. First, filter the data as little as possible. As you’ve seen, filters always distort the time-domain properties of ERP waveforms, and you don’t want to distort your waveforms unnecessarily. And the more sharply you filter your data, the more you are making the data precise in the frequency domain but imprecise in the time domain. The time domain is usually more important than the frequency domain in conventional ERP studies, so you should try to optimize the time-domain properties of your filters rather than their frequency-domain properties. My second main recommendation is to think about exactly how your data might be distorted by a given filter before you use that filter. Often, the best way to see how a filter will distort your data is to create a simulated, noise-free waveform and apply the filter to it. This will allow you to see how the filter distorts component onsets and offsets and adds spurious components or oscillations to your data. If the effects of the filter on the simulated data aren't too bad, then you can use the filter on your real data with confidence that the filter will help you rather than hurt you. To help you with this, I have provided examples of various waveforms that you can import into your data analysis system to test your filters (available online at http://mitpress.mit.edu/luck2e).

A Closer Look at Time-Frequency Analysis

Now that you understand how filtering is achieved by convolving the filter's impulse response function with the ERP waveform, you're ready for a more detailed description of how time-frequency analysis works. I will just cover the basics of the math here; if you want more detail, you should take a like at Mike Cohen's book on data analysis (Cohen, 2014). Also, I assume you've already read the section on time-frequency analysis in Chapter 8 and the discussion of convolution in Online Chapter 11.

It's easiest to understand time-frequency analysis by thinking first about how it would be done for a single frequency (e.g., estimating the time course of 10 Hz activity on each trial and then averaging the trials together). Once you understand how this works for a single frequency, you can just repeat the process over and over again for every frequency of interest. Imagine, for example, that you wanted to estimate the time-course of the 10 Hz activity each of a set of EEG epochs, replacing the voltage value at each time point in the EEG waveforms with a measure of the 10 Hz amplitude at that time point.

This would be very similar to filtering the waveform with a narrow bandpass filter that passes voltages at 10 Hz and suppresses all other frequencies. The frequency response function of this filter would have a value of 1.0 at 10 Hz and a value of zero at every other frequency, as shown in Figure 12.12A. What would be the impulse response function this filter? As you may recall from earlier in this chapter, you can convert back and forth between a frequency response function and an impulse response function by means of the Fourier and inverse Fourier transforms (see Figure 12.5). To derive the impulse response function of the frequency response function shown in Figure 12.12A, you simply convert it from the frequency domain to the time domain with the inverse Fourier transform. Of course, given the simple frequency response function shown in Panel A of the figure, this would just be an infinite-duration sine wave at 10 Hz (as shown in Panel B). Thus, the impulse response function for a perfect 10 Hz bandpass filter is just an infinite-duration 10 Hz sine wave.

However, there is a problem with using this impulse response function: Because it is infinite in duration, convolving this function with an unfiltered EEG segment would create an infinite-duration 10 Hz sine wave (with an amplitude that corresponds to the amplitude of the 10 Hz component of the waveform being filtered). In other words, using this perfect 10 Hz bandpass filter will completely eliminate all the temporal information in the EEG waveform. This is a result of the inverse relationship between the time and frequency domains: Because this filter is infinitesimally narrow in the frequency domain, it discards all time-domain information and just gives us a time-independent measure of the amount of 10 Hz activity in the waveform.

To get an estimate of the 10 Hz activity at each individual time point in the waveform, it is necessary to limit the temporal extent of the impulse response function. As was discussed previously in this chapter (see Figure 12.6), a particularly good way to do this is to use a Gaussian function to gradually taper the amplitude of the sine wave, creating a Gabor function (which is just a Gaussian multiplied by a sine wave). Panels C and D of Figure 12.12 show what a 10 Hz Gabor function looks like, along with the corresponding frequency response function (which is computed by simply applying a Fourier transform to the Gabor function). You can see that this frequency response function has a gain of 1.0 at 10 Hz (i.e., it perfectly passes all activity at 10 Hz) and then falls to zero by approximately 5 Hz and 15 Hz. We could make this frequency response function narrower by using a wider Gaussian to create the Gabor function; in other words, we could get better precision in the frequency domain by having a wider spread in the time domain.

Panels E and F of Figure 12.12 show a single-trial EEG segment and the result of filtering this waveform by convolving it with the 10 Hz Gabor function from Panel D. Remember, this is equivalent to replacing each point in the original waveform with a scaled and shifted copy of the Gabor function and then summing together the results. Filtering leads to a dramatic reduction in the overall amplitude of the waveform, but this makes sense because we're removing everything from the waveform except the 10 Hz activity. And the remaining activity goes up and down at approximately 10 Hz.

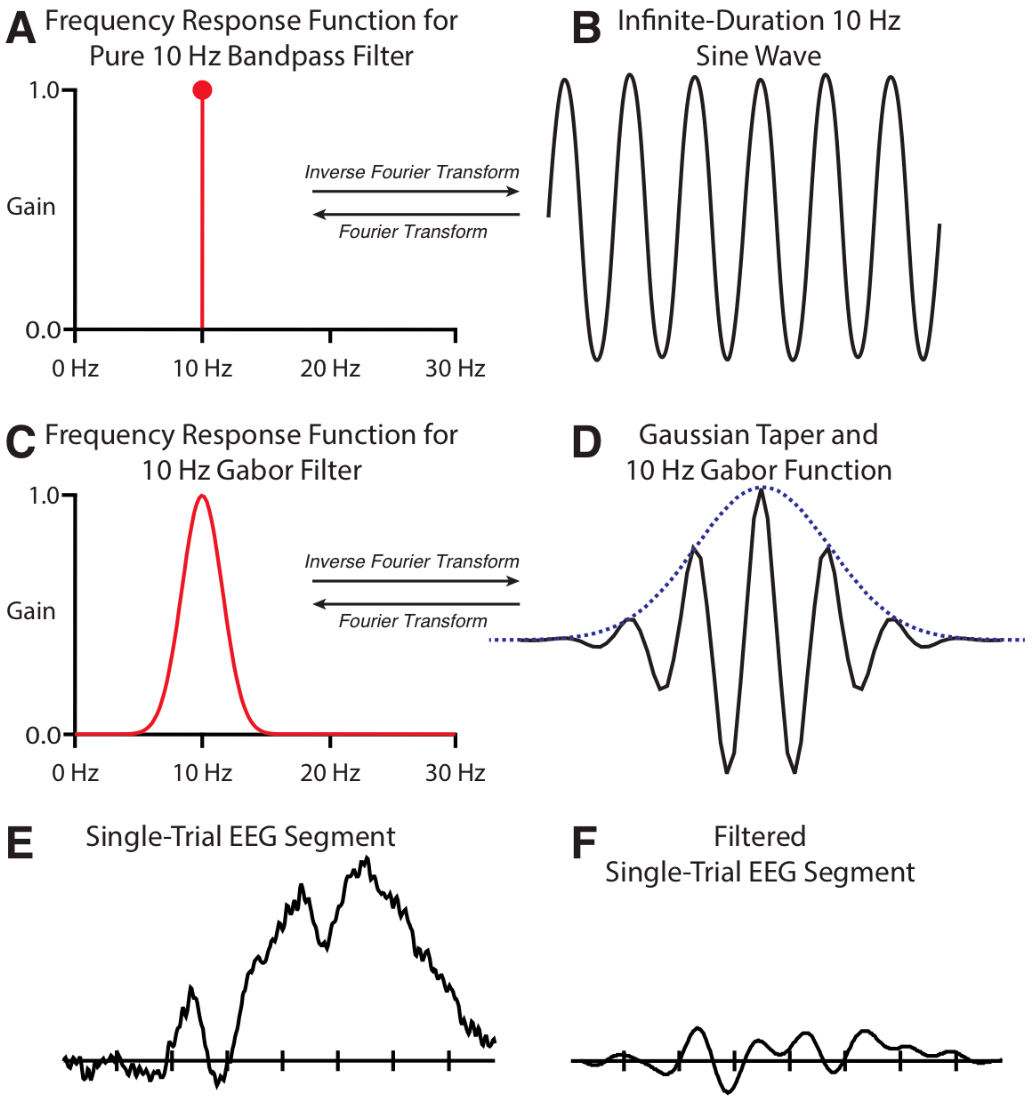

Figure 12.12. (A) Frequency response function for a filter that passes 10 Hz perfectly and completely suppresses every other frequency. (B) Infinite-duration sine wave, which is equivalent to transforming (A) into the time domain. (C) Frequency response function for a bandpass filter that perfectly passes 10 Hz and then falls off at higher and lower frequencies. (D) Gabor function created by multiplying a Gaussian-shaped taper function by a 10-Hz sine wave. This is the impulse response function for the frequency response function shown in (C). (E) Example of a 1-second EEG segment. (F) EEG segment from (E) after being filtered by convolving it with the impulse response function shown in (D).

This filter isn't by itself enough to give us the amplitude at 10 Hz at each time point in the waveform, because the filtered voltage at each time point combines phase information with amplitude information. This is illustrated in Figure 12.13, which illustrates the convolution process in detail. Panel A shows a single-trial epoch of simulated EEG data, consisting of a single cycle of a 10 Hz sine wave. The data are represented as bars at each time point, to help us keep in mind that convolution consists of replacing each time point in the EEG waveform with a scaled and shifted copy of the impulse response function. This sine wave is centered at a time point labeled *X*. If you think about the amplitude this sine wave in the frequency domain, ignoring phase, it extends from -50 ms to +50 ms relative to time X, with equal amplitude (in the frequency domain) throughout this 100-ms period. In other words, we've just chopped out a 100-ms period from an infinite-duration sine wave, and we haven't done anything to change the voltage of the sine wave over this 100-ms period. The phase of the sine wave causes the positive peak to be at -25 ms and the negative peak to be at +25 ms, but these peaks would be at different times if the phase of the sine wave were changed within this 100-ms window.

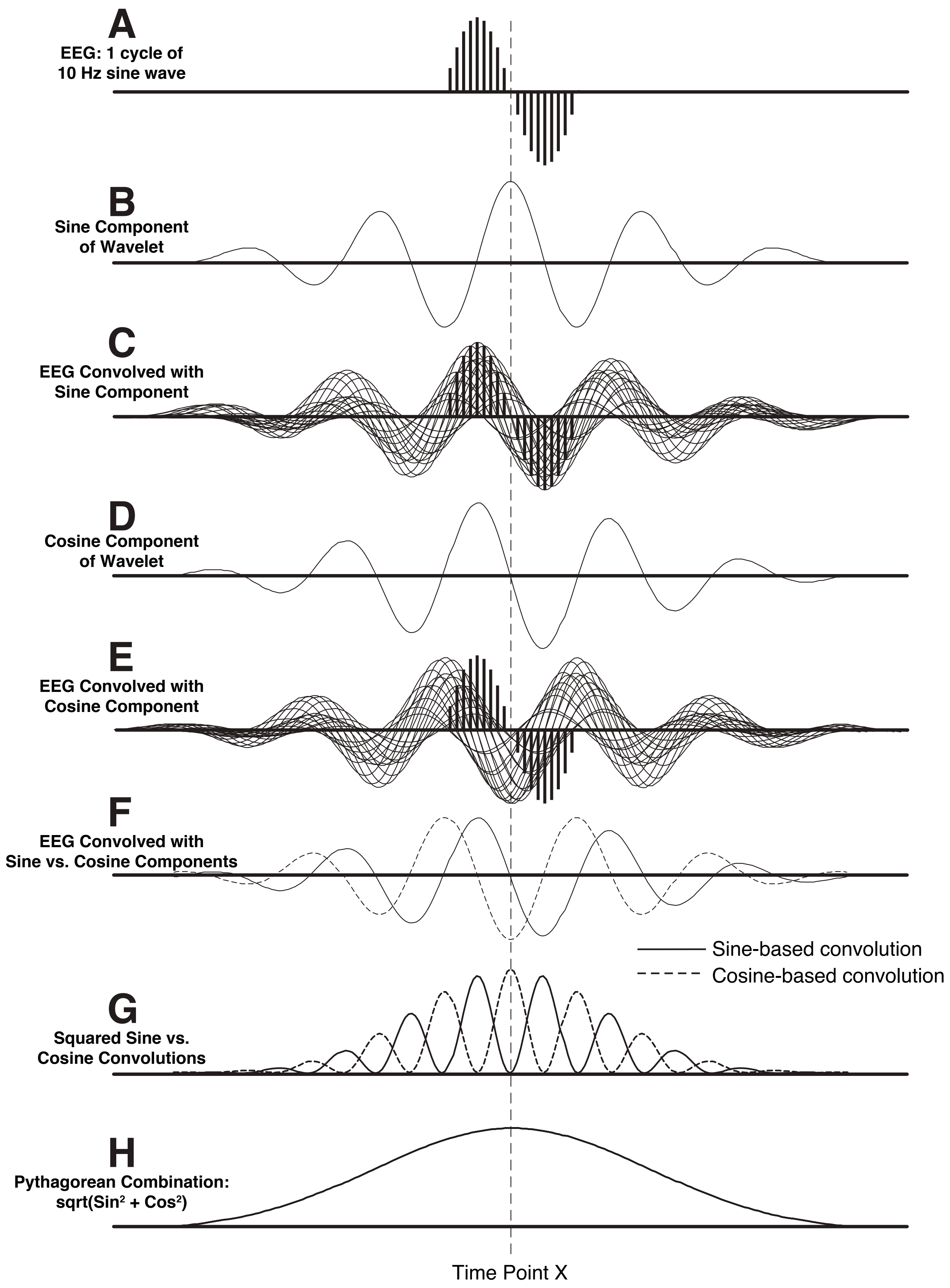

Figure 12.13. Simplified version of time-frequency analysis. See text for details.

Panel B of Figure 12.13 shows a 10 Hz Gabor function created by tapering a 10 Hz sine wave with a Gaussian (just as in Figure 12.12). Panel C shows what happens if we replace each point in our simulated EEG segment with a scaled and shifted copy of this Gabor function. If we then summed these scaled-and-shifted

Gabor functions together, we would get the filtered EEG segment. You can imagine that the filtered EEG segment would have a positive peak at -50 ms and a negative peak at +50 ms, reflecting the positive and negative peaks in the unfiltered EEG segment. This filtered waveform still reflects the phase of the original EEG waveform. If the phase of the sine wave in the original EEG waveform varied from trial to trial, and we averaged together multiple trials of filtered EEG with different phases, these oscillations would cancel each other out, so we wouldn't be able to see the oscillations in the average. A major goal of time-frequency analysis is to get ride of the phase information so that there is no cancellation of positive and negative when we average across trials.

To avoid this cancellation, we need a way to estimate the amplitude of the 10 Hz component of the EEG in a phase-independent manner. To do this, we need to use a simple little trick. Specifically, we also need to create a filtered waveform using a Gabor function in which the sinusoidal component has been shifted by 90°. In other words, instead of multiplying a *sine* wave by a Gaussian to get the impulse response function, we multiply a *cosine* wave by a Gaussian to create the impulse response function. This is shown in Panel D of Figure 12.13, and Panel E shows what happens if we replace each point in the unfiltered EEG with a scaled-and-shifted copy of this cosine-based Gabor function. The result is very similar to the sine-based convolution, but shifted in phase by 90°. Panel F shows the sine-filtered and cosine-filtered waveforms.

The next step is to combine the sine- and cosine-filtered waveforms in a manner that discards the phase information that is present in each of these waveforms. To get a single amplitude value at each time point, independent of phase, we combine the sine-based and cosine-based values at each time point using the Pythagorean theorem (note that the rationale for this is described in the online supplement to Chapter 7). In other words, we square the sine-filtered and cosine-filtered values for each time point, sum them together, and then take the square root of this sum. The square root gives us the phase-independent amplitude at each time point.

Panel G of Figure 12.13 shows what happens when we square the sine-based and cosine-based filtered waveforms. You can see that the squared sine-based waveform is large wherever the cosine-based waveform is small. Panel H shows what happens when we sum these two squared waveforms and then take the square root at each time point. The result is an entirely positive waveform that peaks exactly at time point X, which is the center of the original sine wave from the unfiltered EEG. If we applied this procedure to multiple EEG segments, each with a sinusoid extending from -50 to +50 ms but with a phase that varied across trials, we would get a waveform like that shown in Panel H for every trial. If we then averaged them together, they would not cancel out, because the phase information has been discarded.

Note that the original sine wave in Panel A lasted for only 100 ms, whereas the waveform in Panel H is much longer. This reflects the loss of temporal precision that is inevitable when we extract a relatively narrow band of frequencies from the data. In many cases, the fact that variable-phase EEG oscillations that are invisible in conventional ERP averages can be made visible using this procedure more than makes up for the loss of temporal precision. However, you should be aware that the onset time of a time-frequency effect will almost always be earlier than the true onset time of the effect, and the duration will almost always be longer.

In most cases, the width of the Gaussian that is used to create the Gabor functions is adjusted for each frequency so that the number of cycles of the sine wave is the same at each frequency. In other words, the Gabor function is very wide (possibly hundreds or thousands of milliseconds) for low frequencies and very narrow (possibly 100 ms or less) at high frequencies. This means that the precision in the frequency domain is equal across all frequencies, but the time-domain precision becomes progressively worse at lower frequencies. One implication of this is that it is very difficult to extract activity at frequencies of less than 2 Hz, because it requires several seconds of artifact-free data in each EEG epoch to get a reasonable estimate of activity at these low frequencies. Often, researchers don't bother with frequencies lower than approximately 5 Hz for this reason. However, as discussed in Chapter 8, it is difficult to tell the difference between a transient ERP and a true oscillation without seeing if the activity extends all the way down to very low frequencies. Thus, it is difficult to interpret time-frequency analyses unless the researchers show the results for reasonably low frequencies.

References

Glaser, E. M., & Ruchkin, D. S. (1976). Principles of Neurobiological Signal Analysis. New York: Academic Press.

Woldorff, M. (1993). Distortion of ERP averages due to overlap from temporally adjacent ERPs: Analysis and correction. Psychophysiology, 30, 98-119.

Cohen, M. X. (2014). Analyzing Neural Time Series Data: Theory and Practice. Cambridge, MA: MIT Press.