An Introduction to the Event-Related Potential Technique

Steven J. Luck

The MIT Press

From The MIT Press

MITCogNet

MIT Press books may be purchased at special quantity discounts for business or sales promotional use. For information, please email special_sales@mitpress.mit.edu or write to Special Sales Department, The MIT Press, 55 Hayward Street, Cambridge, MA 02142.

This book was set in Melior and Helvetica on 3B2 by Asco Typesetters, Hong Kong.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Luck, Stephen J.
An introduction to the event-related potential technique / Stephen J. Luck.
p. cm. — (Cognitive neuroscience)
Includes bibliographical references and index.
ISBN 0-262-12277-4 (alk. paper) — ISBN 0-262-62196-7 (pbk. : alk. paper)
1. Evoked potentials (Electrophysiology) I. Title. II. Series.
QP376.5.L83 2005
616.8′047547—dc22 2005042810

10 9 8 7 6 5 4 3 2 1

5 Filtering

This chapter discusses the application of filters to the EEG during data acquisition and to ERP waveforms before and after the averaging process. It is absolutely necessary to use filters during data acquisition, and it is very useful to apply filters offline as well, but filtering can severely distort ERPs in ways that ERP researchers frequently do not appreciate. For example, filters may change the onset and duration of an ERP component, may make monophasic waveforms appear multiphasic, may induce artificial oscillations, and may interfere with the localization of generator sources. This chapter will explain how these distortions arise and how they can be prevented. To avoid complex mathematics, I will simplify the treatment of filtering somewhat in this chapter, but there are several books on filtering that the mathematically inclined reader may wish to read (e.g., Glaser & Ruchkin, 1976). Note also that the term *filter* can refer to any of a large number of data manipulations, but this chapter will be limited to discussing the class of filters that ERP researchers typically use to attenuate specific ranges of frequencies, which are known as *finite impulse response filters.*

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. My goal in this chapter is to provide an intuitive understanding of how filters work, and I have therefore minimized the use of equations as much as possible and limited the equations to simple algebra. If you understand how summations work (i.e., the Σ symbol), then you know enough math to understand this chapter.

Even though an advanced math background is not necessary to understand this chapter, some of the concepts are pretty complicated. As a result, you may not want to spend the time required to understand exactly how filters work and exactly why they distort ERP waveforms in particular ways, especially on your first pass through the book. If so, you can just read the next two sections (''Why Are Filters Necessary'' and ''What Everyone Should Know About Filtering''). These sections will provide enough information to keep you from getting into too much trouble with filters, and you can read the rest of the chapter later.

Why Are Filters Necessary?

The most important message of this chapter is that filters can substantially distort ERP data. Given that this is true, it is useful to ask why filters are necessary in the first place. There are two answers to this question, the first of which is related to the *Nyquist Theorem*, discussed previously in chapter 3. This theorem states that it is possible to convert a continuous analog signal (such as the EEG) into a set of discrete samples without losing any information, as long as the rate of digitization is at least twice as high as the highest frequency in the signal being digitized. This is fundamentally important for the data acquisition process, because it means that we can legitimately store the EEG signal as a set of discrete samples on a computer. However, this theorem also states that if the original signal contains frequencies that are more than twice as high as the digitization rate, these very high frequencies will appear as artifactual *low* frequencies in the digitized data (this is called *aliasing*). Consequently, EEG amplifiers have filters that one can use to suppress high frequencies; these filters are generally set to attenuate frequencies that are higher than a half of the sampling rate. For example, in a typical cognitive ERP experiment, the digitization rate might be 250 Hz, and it would therefore be necessary to make sure that everything above 125 Hz is filtered. It would be tempting to choose a filter cutoff frequency of 125 Hz, but a cutoff frequency of 125 Hz just means that the power (or amplitude) has been reduced by 50 percent at 125 Hz, and there will be considerable remaining activity above 125 Hz. It is therefore necessary to select a substantially lower value such as 80 Hz (in practice, the digitization rate should be at least three times as high as the cutoff value of the filter).

The second main goal of filtering is the reduction of noise, and this is considerably more complicated. The basic idea is that the EEG consists of a signal plus some noise, and some of the noise is sufficiently different in frequency content from the signal that it can be suppressed simply by attenuating certain frequencies. For example, most of the relevant portion of the ERP waveform in a typical cognitive neuroscience experiment consists of frequencies between 0.01 Hz and 30 Hz, and contraction of the muscles leads to an EMG artifact that primarily consists of frequencies above 100 Hz; consequently, the EMG activity can be eliminated by suppressing frequencies above 100 Hz and this will cause very little change to the ERP waveform. However, as the frequency content of the signal and the noise become more and more similar, it becomes more and more difficult to suppress the noise without significantly distorting the signal. For example, alpha waves can provide a significant source of noise, but because they are around 10 Hz, it is difficult to filter them without significantly distorting the ERP waveform. Moreover, even when the frequency content of the noise is very different from that of the signal, the inappropriate application of a filter can still create significant distortions.

In addition to suppressing high frequencies, filters are also used in most experiments to attenuate very low frequencies. The most common use of such filters is to remove very slow voltage changes of non-neural origin during the data acquisition process. Specifically, factors such as sweating (which creates skin potentials) and drifts in electrode impedance can lead to slow, sustained changes in the baseline voltage of the EEG signal, and it is usually a good idea to remove these slow voltage shifts by filtering frequencies lower than approximately 0.01 Hz. This is especially important when obtaining recordings from patients or from children, because head and body movements are one common cause of these sustained shifts in voltage. If these shifts are not eliminated, they may cause the amplifier to saturate and data to be lost. Even if they do not cause amplifier saturation, they are very large voltages and may cause large distortions in the average ERP waveforms. Thus, it is almost always a good idea to filter the very low frequencies (< 0.01 Hz).

What Everyone Should Know about Filtering

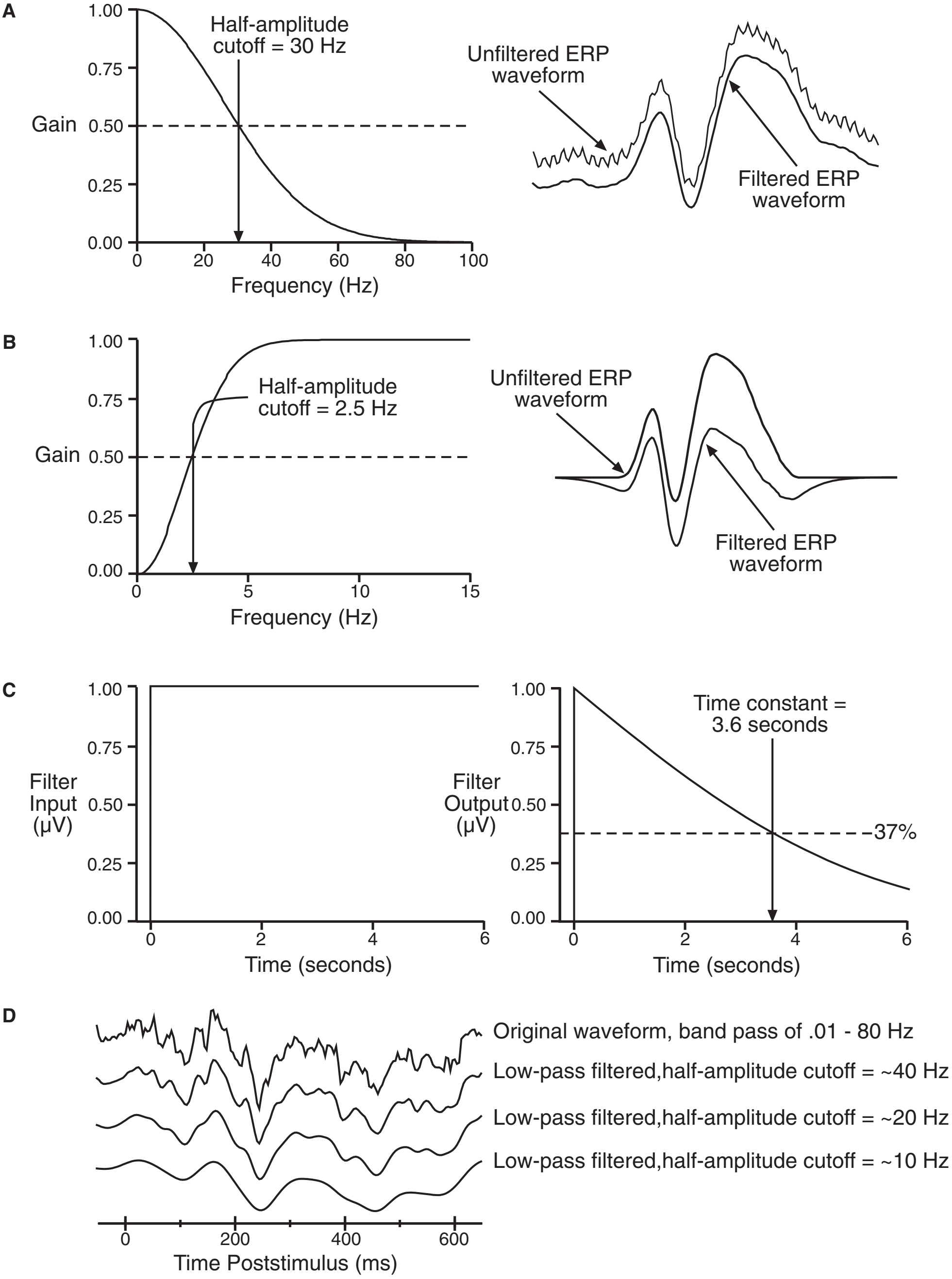
Any waveform can be decomposed into a set of sine waves of various frequencies and phases, and the waveform can be reconstructed by simply summing these sine waves together. Filters are usually described in terms of their ability to suppress or pass various different frequencies. The most common types of filters are: (1) *low-pass filters*, which attenuate high frequencies and pass low frequencies; (2) *high-pass filters*, which attenuate low frequencies and pass high frequencies; (3) *bandpass filters*, which attenuate both high and low frequencies, passing only an intermediate range of frequencies; and (4) *notch filters*, which attenuate some narrow band of frequencies and pass everything else.

Filters and Amplitude Attenuation

The properties of a filter are usually expressed by its *transfer function*, the function that determines how the input of the filter is ‘‘transferred’’ to the output of filter. The transfer function can be broken down into two components, a frequency response function that specifies how the filter changes the amplitude of each frequency, and a phase response function that specifies how the filter changes the phase of each frequency. Figure 5.1A shows an example of the frequency response function of a low-pass filter with a half-amplitude cutoff at 30 Hz, and it also shows how this filter suppresses 60-Hz noise in an actual ERP waveform. In ERP research, a filter is often described only in terms of its *half-amplitude cutoff*, which is the frequency at which the amplitude is cut by 50 percent (sometimes the cutoff frequency is specified in terms of power rather than amplitude; amplitude reaches 71 percent when power reaches 50 percent). As figure 5.1A illustrates, however, there is quite a bit of attenuation at frequencies below the half-amplitude cutoff, and the attenuation is far from complete for frequencies quite a bit higher than the half-amplitude cutoff. Nonetheless, this filter effectively suppresses the 60-Hz noise in the waveform while retaining the waveform's basic shape.

Figure 5.1B shows the frequency response function of a high-pass filter with a half-amplitude cutoff at approximately 2.5 Hz and the effect of this filter on an ERP waveform. As you can see, the large, broad upward deflection (similar to a P3 wave) is greatly reduced in amplitude by this filter, whereas the initial, higher frequency deflections (similar to P1 and N1 waves) are not reduced as much. However, the amplitudes of the earlier components are influenced somewhat, and small artifactual peaks have been created at the beginning and end of the waveform. This leads to the most important point of this section: *filters can significantly distort ERP waveforms, changing the amplitude and timing of the ERP components and adding artifactual peaks.*

High-pass filters are often described in terms of their *time constants* rather than their half-amplitude cutoffs. As figure 5.1C shows, if the input to a high-pass filter is a constant voltage, the output of the filter will start at this voltage and then gradually return toward zero, and the filter's time constant is a measure of the rate at which this occurs. The decline in output voltage over


A
Gain
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
Frequency (Hz)
Half-amplitude cutoff = 30 Hz
Unfiltered ERP waveform
Filtered ERP waveform
B
Gain
1.00
0.75
0.50
0.25
0.00
0
5
10
15
Frequency (Hz)
Half-amplitude cutoff = 2.5 Hz
Unfiltered ERP waveform
Filtered ERP waveform
C
Filter Input (μV)
1.00
0.75
0.50
0.25
0.00
0
2
4
6
Time (seconds)
Filter Output (μV)
1.00
0.75
0.50
0.25
0.00
0
2
4
6
Time (seconds)
Time constant = 3.6 seconds
37%
D
Original waveform, band pass of .01 - 80 Hz
Low-pass filtered,half-amplitude cutoff = ~40 Hz
Low-pass filtered,half-amplitude cutoff = ~20 Hz
Low-pass filtered,half-amplitude cutoff = ~10 Hz
0
200
400
600
Time Poststimulus (ms)

time is exponential, so the value never quite reaches zero. Consequently, the time constant is expressed as the time required for the filter's output to reach $1/e$ (37 percent) of the starting value. As the half-amplitude cutoff becomes higher, the time constant becomes shorter. If you know the half-power cutoff frequency of a high-pass filter (f_c, the frequency at which the filter's output is reduced by 3 dB), the time constant can be computed as $1/(2\pi f_c)$.

Filters and Latency Shift

So far, we have focused on the effects of filters on the amplitude at each frequency, but it also important to keep in mind that filters usually influence the phase (time shift) at each frequency as well. In particular, all analog filters (such as those in EEG amplifiers) shift the latency of the signal, and most analog filters shift the latency by different amounts for different frequencies. These shifts become more pronounced as the cutoff frequency of a low-pass filter is made lower and the cutoff frequency of a high-pass filter is made higher. In contrast, most digital filters (filters that are applied offline) do not cause a phase shift. Thus, it is usually best to do as little filtering as possible prior to digitizing the data and to do most of the filtering offline.

How Filters Are Used

Some filtering is essential in amplifying and digitizing the EEG signal. First, it is necessary to filter out high frequencies with a low-pass filter before digitizing the data to ensure that the

◄ **Figure 5.1** Basics of filtering. (A) Frequency response function of a low-pass filter with a half-amplitude cutoff at 30 Hz, and application of this filter to an ERP waveform. (B) Frequency response function of a high-pass filter with a half-amplitude cutoff at 2.5 Hz, and application of this filter to an ERP waveform. (C) Example of the time constant of a high-pass filter. A constant input to the filter is shown on the left, and the decaying output of the filter is shown on the right. (D) Noisy ERP waveform low-pass filtered with various half-amplitude cutoffs.

sampling rate is at least twice as high as the highest frequency in the signal. Second, it is almost always necessary to filter out very low frequencies (e.g., <0.01 Hz) so that slow, non-neural electrical potentials (e.g., skin potentials) do not bring the signal outside of the operating range of the amplifier and analog-to-digital converter. ERP researchers also frequently use a *line frequency filter*, a notch filter that filters out noise generated by AC electrical devices (usually 50 Hz or 60 Hz). As I will discuss later in this chapter, however, notch filters significantly distort the data and should be avoided if possible. Chapter 3 described some strategies for eliminating electrical noise at the source, but if it is impossible to reduce line-frequency noise sufficiently, a low-pass filter with a half-amplitude cutoff at 30 Hz can effectively filter out the noise without distorting the data as much as a notch filter.

ERP researchers sometimes use low-pass and high-pass filters to ''clean up'' noisy data. For example, figure 5.1D shows an ERP waveform that was mildly filtered (.01–80 Hz) prior to amplification and the same waveform after it was low-pass filtered offline with progressively lower cutoff frequencies. As the cutoff frequency decreases, the waveform becomes smoother and generally nicer looking. From this example, you might think that filtering is a good thing. However, the most important thing that I would like to communicate in this chapter is that filtering always distorts the ERP waveform, and the more heavily the data are filtered, the worse the distortion will be.

How Filters Can Distort Your Data

The distortions caused by filtering can be summarized by a key principle that you should commit to memory now and recall every time the topic of filtering comes up: *precision in the time domain is inversely related to precision in the frequency domain.* In other words, the more tightly you constrain the frequencies in an ERP waveform (i.e., by filtering out a broad range of frequencies), the more the ERP waveform will become spread out in time. Figure

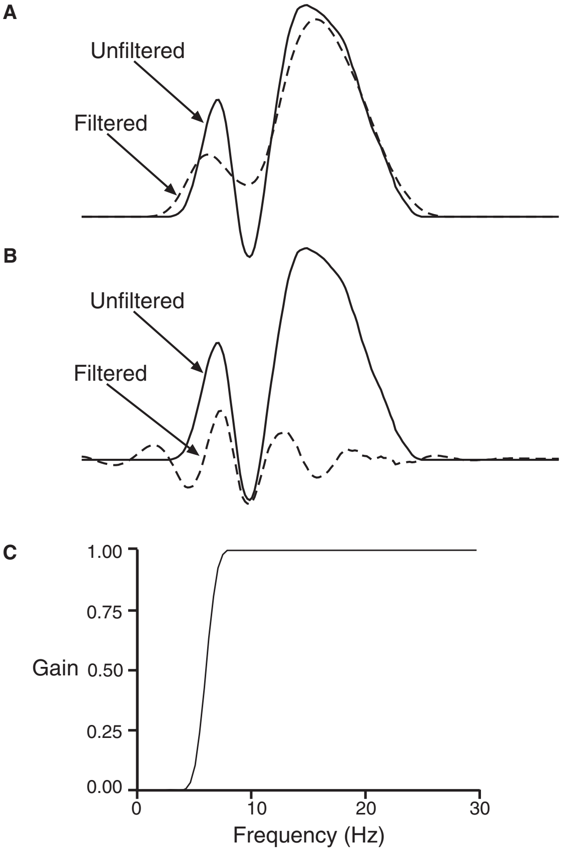


Figure 5.2 Examples of distortions caused by filtering. (A) Effects of low-pass filtering on onset and offset times of an ERP waveform. (B) Artifactual oscillations caused by a high-pass filter. (C) Frequency response function of the filter used in (B).

5.2A illustrates one type of spreading, showing how low-pass filtering an ERP waveform causes the filtered waveform to start earlier and end later than the unfiltered waveform. Low-pass filters almost always have this effect of ‘‘smearing out’’ the waveform and distorting the onset and offset times of the ERP components and experimental effects. Thus, if you see that an experimental effect starts at 120 ms in low-pass filtered waveforms, the effect may have actually started at 150 ms.

As figure 5.2B illustrates, high-pass filters also cause the ERP waveform to become spread out in time, but the distortion in this case consists of a series of up and down sinusoidal deflections. Thus, not only did this particular filter cause artifactual activity to begin well before the actual onset of activity, it created the appearance of oscillations in the waveform. This sort of effect could also cause an experimental effect in a component of one polarity to appear as an effect in an earlier component of opposite polarity (e.g., an increase in N1 amplitude in the original waveform might appear as an increase in P1 amplitude in the filtered waveform). As you can imagine, using this sort of filter might cause someone to completely misinterpret the results of an ERP experiment.

High-pass filters do not always create artifactual oscillations. For example, the high-pass filter shown in figure 5.1B produces a single opposite-polarity artifactual deflection at each end of the ERP waveform rather than a multi-peaked oscillation. The long-lasting and oscillating pattern of distortion the filter shown in figure 5.2B creates is a consequence of the filter's frequency response function, which is shown in figure 5.1C. Whereas the filter in figure 5.1B has a relatively gradual transition between suppressing the lowest frequencies and passing higher frequencies, the filter in figure 5.2B completely blocks a fairly broad set of low frequencies and then suddenly starts passing higher frequencies. That is, the filter in figure 5.2B has more precision in terms of its frequency properties than the filter in figure 5.1B, and this brings us back to the principle that precision in the time and frequency domains are inversely related. That is, more sudden changes in a filter's frequency response function lead to broader temporal distortions in the ERP waveform. Thus, a filter with a really sharp cutoff in its frequency response function might seem to be ideal, but a sharp cutoff usually means that the filter will cause more severe distortion of the ERP waveform than a filter with a gentler cutoff. Makers of commercial ERP systems often tout the sharp cutoffs of their filters, but such filters are usually a bad idea.

If you want to see how a filter might be distorting your ERP waveforms, the easiest thing to do is to pass some sort of known, artificial waveform through the filter and compare the original and filtered waveforms. For example, many amplifiers contain calibrators that produce a single pulse of a square wave, and you can record this and filter it with various types of filters.

Recommendations

Now that I've discussed why filters are used and how they can distort your data, I will make some specific recommendations. These recommendations might not be appropriate for every experiment, but they should be appropriate for the vast majority of ERP experiments in cognitive neuroscience.

First, keep Hansen's Axiom in mind: *There is no substitute for clean data* (see chapter 3). Some minor filtering is necessary when first collecting the data, and a modest amount of additional filtering can be helpful under some conditions. However, filters cannot help you if your data are noisy because of variability across subjects, variability across trials, a small number of trials in your averages, and so on. Filters may make the data *look* better under these conditions, but this will be an illusion that may lead you to draw incorrect conclusions.

Second, during the amplification and digitization process, you should do as little filtering as possible. It's always possible to filter the data more offline, but you can never really ''unfilter'' data that have already been filtered. The low-pass filter should be set at between one third and one fourth of the sampling rate (and I would recommend a sampling rate of between 200 and 500 Hz for most experiments). For experiments with highly cooperative subjects (e.g., college students), I would recommend using a high-pass filter of 0.01 Hz. Some amplifiers allow you to do no high-pass filtering at all (these are called DC recordings), but unless you are looking at very slow voltage shifts, this will lead to an unacceptable level of noise due to skin potentials and other slow artifacts.

If you are recording from less cooperative subjects (e.g., patients or children), you may need to use a higher high-pass cutoff, such as 0.05 Hz or even 0.1 Hz. The main problem with such subjects is that they may move around a lot, which causes changes in the baseline electrical potential that will slowly resolve over time. This creates a lot of noise in the data that averaging may not sufficiently attenuate, and it can also lead the amplifiers to saturate. But if you use a higher high-pass cutoff, be aware that this is distorting your data somewhat and reducing the amplitude of the lower-frequency components, such as the P3 and N400 waves. Indeed, Duncan-Johnson and Donchin (1979) showed many years ago that relatively high cutoff frequencies lead to a substantial reduction in apparent P3 amplitude.

During amplification, you should avoid using a notch filter (also called a line-frequency filter) to reduce line-frequency noise. These filters can cause substantial distortion of the ERP waveform. As chapter 3 described, there are various precautions you can take to eliminate line-frequency noise before it enters the EEG, and this is the best approach. However, you may sometimes be faced with such a huge level of line-frequency noise that the incoming EEG is completely obscured, and on such occasions you may need to use a notch filter to eliminate this noise.

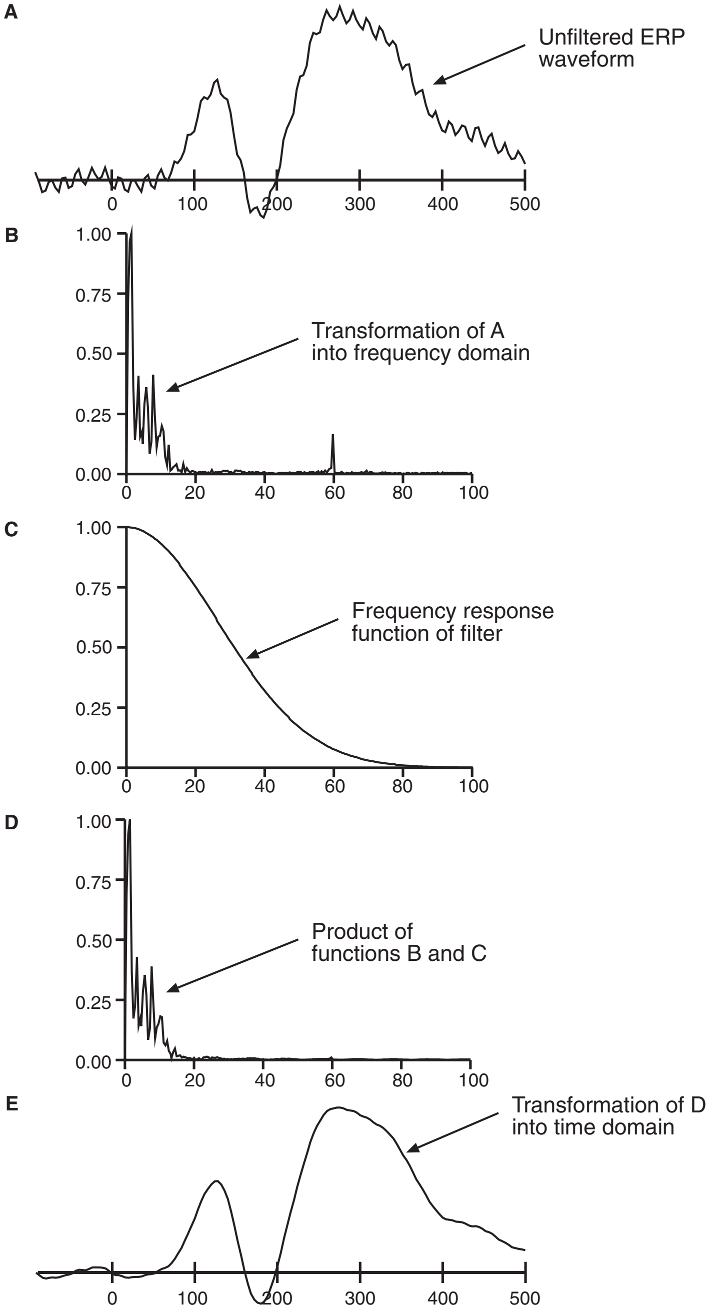
My third recommendation is to keep offline filtering to a minimum. If the averaged ERP waveforms are a little fuzzy looking, making it difficult to see the experimental effects, it can be helpful to apply a low-pass filter with a half-amplitude cutoff somewhere between 20 and 40 Hz. This can dramatically improve the appearance of the ERP waveforms when you plot them, and the temporal distortion should be minimal (especially if you use a filter with a relatively gentle cutoff). This sort of filtering will also be helpful if you are using peak amplitude or peak latency measures, but it is unnecessary if you are using mean amplitude or fractional area latency measures (see chapter 6 for more details). In fact, because filters spread out an ERP waveform, measuring the mean amplitude in a particular latency range from filtered waveforms (e.g., 200–250 ms poststimulus) is equivalent to measuring the mean amplitude from a broader range in the original unfiltered waveforms (e.g., 175–275 ms). My laboratory typically uses a low-pass filter with a half-amplitude cutoff at 30 Hz for plotting the data, but no offline filtering for measuring component amplitudes.

My fourth recommendation is to avoid using high-pass filters altogether (except during data acquisition, as described above). High-pass filters are much more likely than low-pass filters to cause major distortions of your ERP waveforms that might lead you to draw incorrect conclusions about your data. There are occasions when high-pass filters can be useful, such as when dealing with overlapping activity from preceding and subsequent stimuli. However, high-pass filters are sufficiently dangerous that you should use them only if you really understand exactly what they are doing. The remainder of this chapter will help provide you with this understanding.

Filtering as a Frequency-Domain Procedure

The key to understanding filters is to understand the relationship between the *time domain* and the *frequency domain*. A time-domain representation of an ERP is simply a plot of the voltage at each time point, as figure 5.3A illustrates. A frequency-domain representation of an ERP is a plot of the amplitude (and phase) at each frequency, as figure 5.3B illustrates. Time-domain and frequency-domain representations contain exactly the same information, viewed from different perspectives, and it is possible to transform one type of representation into the other via *Fourier analysis*, a technique developed in the 1800s by the mathematician Joseph Fourier.

The basic principle of Fourier analysis is that any time-domain waveform can be exactly represented by the sum of a set of sine waves of different frequencies and phases. In other words, it is possible to create any waveform (even a momentary spike or a square wave) by adding together a set of sine waves of varying frequencies


A
Unfiltered ERP waveform
0
100
200
300
400
500
B
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
Transformation of A into frequency domain
C
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
Frequency response function of filter
D
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
Product of functions B and C
E
Transformation of D into time domain
0
100
200
300
400
500

and phases. The *Fourier transform* is a mathematical procedure that takes a time-domain signal (such as an ERP waveform) as an input and computes the amplitudes and phases of the sine waves that would need to be added together to recreate the input waveform. As figure 5.3B illustrates, the output of the Fourier transform is usually shown as a plot of amplitude and phase[3] as a function of frequency. It is also possible to use the *inverse Fourier transform* to convert a frequency-domain representation back into the original time-domain representation, thus obtaining the original voltage × time ERP waveform.

In the context of Fourier analysis, one can conceptualize filtering as a series of three steps:

1. The time-domain ERP waveform is converted into a frequency-domain representation.
2. The to-be-filtered frequency range is set to zero in the frequency-domain representation.
3. The modified frequency-domain representation is converted back into the time domain.

This completely eliminates one set of frequencies from the ERP waveform without influencing the other frequencies. As I will discuss in detail below, a sudden drop-off in the frequency domain leads to some undesirable consequences, so it is necessary to add a slight complication to this three-step procedure. Specifically,

◀ **Figure 5.3** Example of the frequency-domain conceptualization of filtering. (A) Unfiltered ERP waveform, contaminated by substantial noise at 60 Hz. (B) Transformation of (A) into the frequency domain, with a clear peak at 60 Hz. (C) Frequency response function of a filter that can be used to remove the 60-Hz noise from (B) while retaining most of the signal, which is primarily confined to frequencies below 20 Hz. (D) Product of (B) and (C); for each frequency point, the magnitude in (B) is multiplied by the magnitude in (C). Note that (D) is nearly identical to (B) in the low frequency range, but falls to zero at high frequencies. (E) Transformation of (D) back into the time domain, where it closely resembles the original ERP waveform in (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, although the phase information is crucial for transforming between the time and frequency domains.

rather than setting some range of frequencies to zero and leaving the other frequencies untouched, it is useful to shift gradually from no attenuation to complete attenuation. The function that defines this gradual shift in attenuation as a function of frequency is called the *frequency response function* of a filter. More specifically, the frequency response function contains a scaling factor for each frequency that represents the extent to which that frequency will be passed by the filter, with a value of 1 for frequencies that will be unaffected by the filter, a value of zero for frequencies that will be completely suppressed, and an intermediate value for frequencies that will be partially attenuated. This function is used in step two of the filtering process: rather than setting some frequencies to zero, the frequency response function is multiplied by the frequency-domain representation of the ERP waveform to attenuate each frequency by a specific amount (i.e., each frequency in the ERP is multiplied by the corresponding scaling factor in the frequency response function).

Figure 5.3 illustrates this approach to filtering. Panel A shows a time-domain representation of an unfiltered ERP waveform, which contains an obvious 60-Hz noise oscillation. Panel B shows the Fourier transform of this waveform. Note that most of the power is at low frequencies (< 20 Hz), with a spike at 60 Hz corresponding to the noise oscillation in the time-domain waveform. Panel C shows the frequency response function of a filter that passes the very lowest frequencies and then gradually falls off at higher frequencies. Panel D shows the result of multiplying the frequency-domain representation of the ERP waveform (i.e., panel B) by the frequency response function of the filter (i.e., panel C). Note that the result is very similar to the waveform in panel B, but the spike at 60 Hz is missing. Panel E shows the result of applying the inverse Fourier transform to the waveform in panel D, which converts the frequency-domain representation back into the time domain. The result is an ERP waveform that is very similar to the original waveform in panel A, but without the 60-Hz noise oscillation.

Note that it is possible to use any set of scaling factors for the frequency response function shown in figure 5.3C. For example, you could set all of the odd-numbered frequencies to 1 and all of the even-numbered frequencies to zero, 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 for the lowest and highest frequencies with a single peak at intermediate frequencies (band-pass filters).

The above discussion has concentrated 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 its transfer function, which specifies 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 offline, but is impossible when analog filters are applied online (e.g., when low and high frequencies are removed from the EEG during data acquisition). However, certain analog filters have transfer functions in which the phase portion increases linearly as a function of frequency (e.g., Bessel filters), which means that all frequencies are shifted by the same amount of time; the output of such filters can be shifted back by this amount during the averaging process, thus eliminating the phase shift produced by the filter.

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 I will discuss 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 prestimulus 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.

As discussed earlier in this chapter, an important rule for understanding the distortion that filters may produce is that there is an inverse relationship between the spread of a signal in the time domain and the spread of the same signal in the frequency domain; a signal that is tightly localized in time will be broadly localized in frequency, and vice versa. For example, a signal with a single sharp spike in the frequency domain will translate into an infinite-duration sine wave in the time domain, and a single sharp spike in the time domain will translate into an even spread across all frequencies in the frequency domain. Because of this inverse relationship between the time and frequency domains, using 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 above, leads to an infinite spread of the filtered waveform in time.

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, such as that present in the ERP waveform shown in figure 5.4A. 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 5.4B shows the results of applying such a filter to the ERP waveform in figure 5.4A; this simple filter has clearly eliminated much of the high frequency noise from the waveform. To filter a broader range of frequencies, one can extend this filtering technique by simply averaging together a greater number of points. Figure 5.4C shows the results of averaging seven adjacent points together instead of the three averaged together for figure 5.4B, and this can be seen to further reduce the high frequency content of the waveform. The following simple formula formalizes this method of filtering:

$$fERP_i = \sum_{j=-n}^{n} wERP_{i+j} \tag{5.1}$$

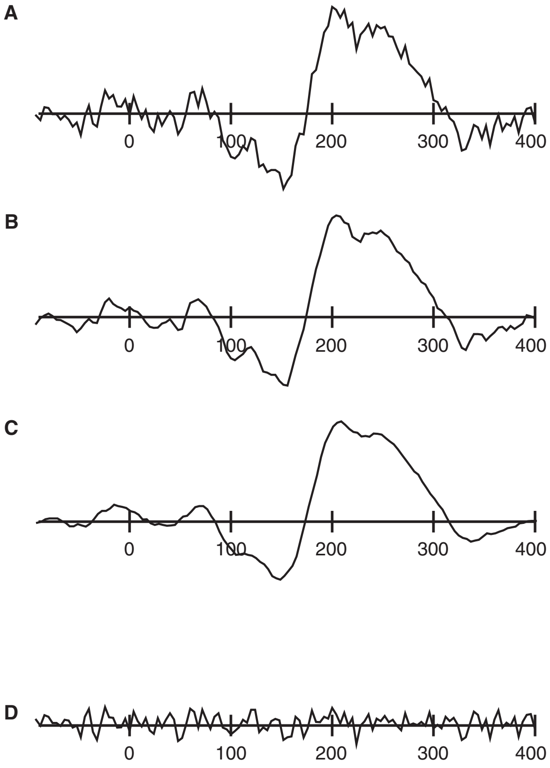


Figure 5.4 Example of filtering an ERP waveform by averaging together the voltages surrounding each time point. (A) Unfiltered ERP waveform, contaminated by substantial high-frequency noise. (B) Result of filtering the waveform in (A) by averaging the voltage at each time point with the voltages at the immediately adjacent time points. (C) Result of filtering the waveform in (A) by averaging the voltage at each time point with the voltages at the three time points on either side. (D) High-pass-filtered waveform, constructed by subtracting the filtered waveform in (C) from the unfiltered waveform (A).

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; and $w = 1/(2n + 1)$ (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 5.1. Once you understand it, you will have understood the essence of digital filtering.

One can extend this filtering technique 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, one can filter out the low frequencies by simply subtracting the low-pass-filtered waveform from the unfiltered waveform. Figure 5.4D shows the result of this form of high-pass filtering, which is equal to the waveform in figure 5.4A minus the waveform in figure 5.4C.

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 three-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:

$$fERP_i = \sum_{j=-n}^{n} W_j ERP_{i+j} \tag{5.2}$$

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

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

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

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

You can use virtually any conceivable weighting function for filtering, and the shape of this function will determine the properties of the filter. As I will describe 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 the filter would produce in response to a momentary voltage spike or *impulse*; it is therefore known as the *impulse response function* of the filter.

Figure 5.5 illustrates the relationship between the weighting function and the impulse response function. Panel A 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.

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:

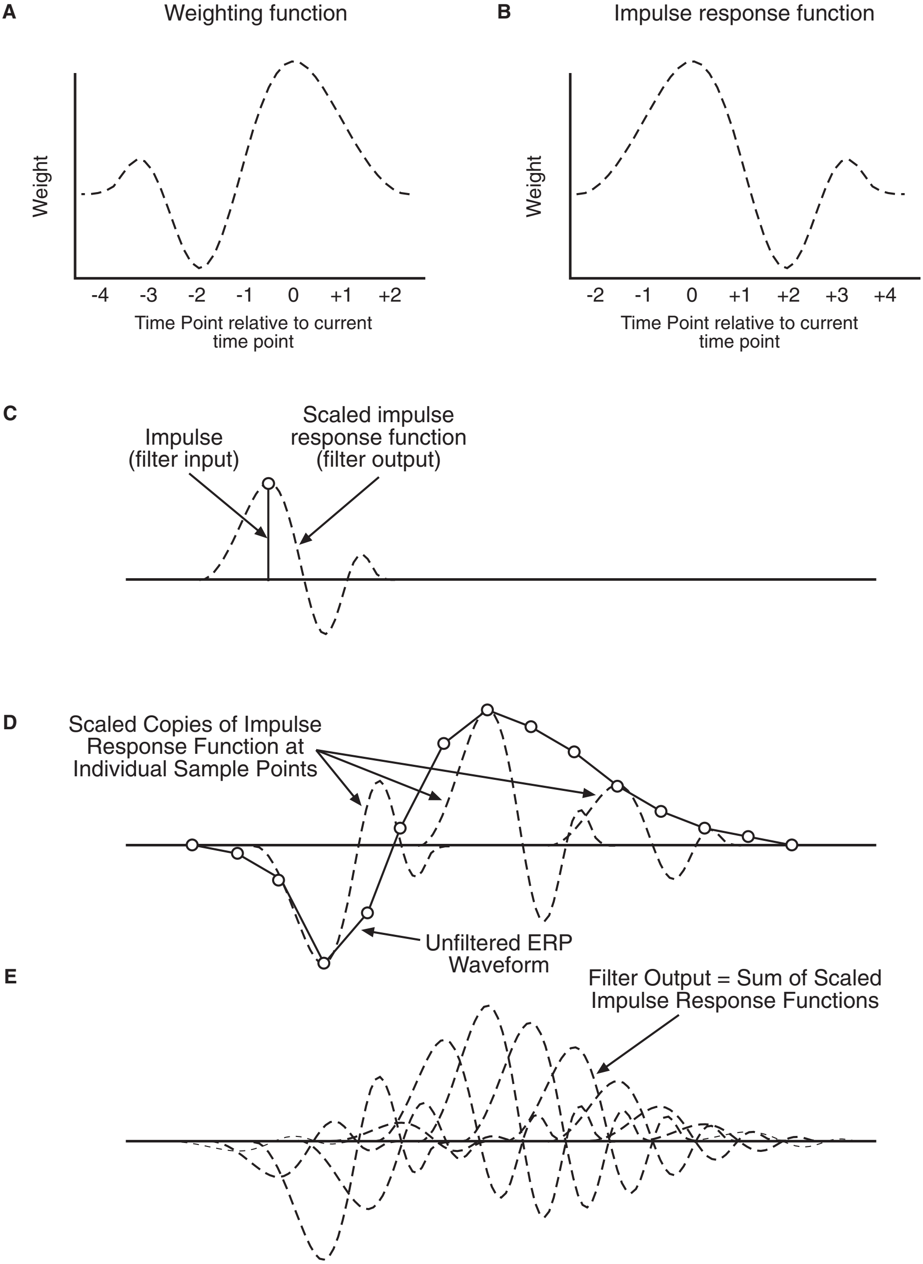
$$fERP_i = \sum_{j=-n}^{n} IRF_j ERP_{i-j} \tag{5.3}$$

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 5.2.

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. We can therefore write equation 5.3 as:

$$fERP = IRF * ERP \tag{5.4}$$

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 × to denote multiplication and $*$ to denote convolution).


A
Weighting function
Weight
-4
-3
-2
-1
0
+1
+2
Time Point relative to current time point
B
Impulse response function
Weight
-2
-1
0
+1
+2
+3
+4
Time Point relative to current time point
C
Impulse (filter input)
Scaled impulse response function (filter output)
D
Scaled Copies of Impulse Response Function at Individual Sample Points
Unfiltered ERP Waveform
E
Filter Output = Sum of Scaled Impulse Response Functions

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. Although 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.

Figure 5.5 illustrates this approach to filtering. 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

◄ **Figure 5.5** 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, this panel also shows the scaled impulse response functions corresponding to three of the sample points. 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.

waveform is filtered by replacing each data point with a scaled copy of the impulse response function (for the sake of simplicity, panel D shows this for only three data points). 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 a filter's output 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 that 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. Figure 5.6 illustrates this.

It is usually substantially 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

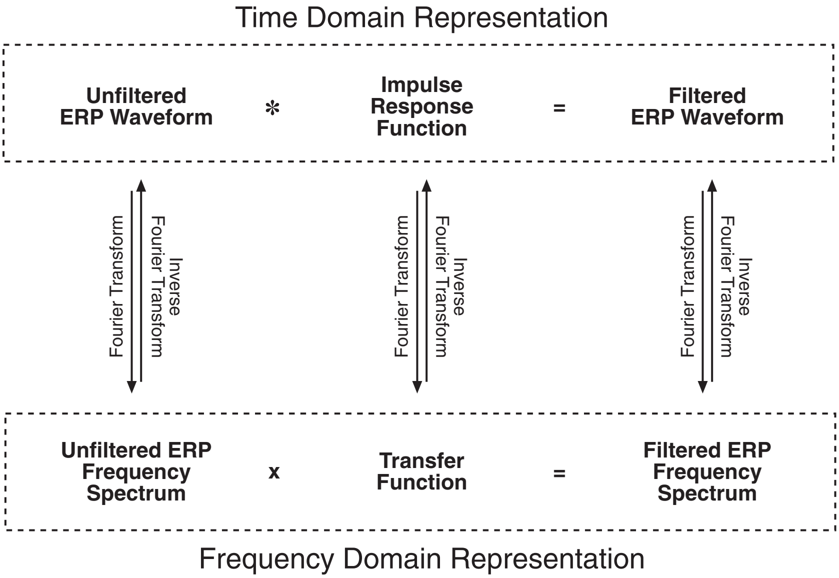

Time Domain Representation
Unfiltered ERP Waveform
*
Impulse Response Function
=
Filtered ERP Waveform
Fourier Transform
Inverse Fourier Transform
Fourier Transform
Inverse Fourier Transform
Fourier Transform
Inverse Fourier Transform
Unfiltered ERP Frequency Spectrum
x
Transfer Function
=
Filtered ERP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

Figure 5.6 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 $\times$ operator) in the frequency domain.

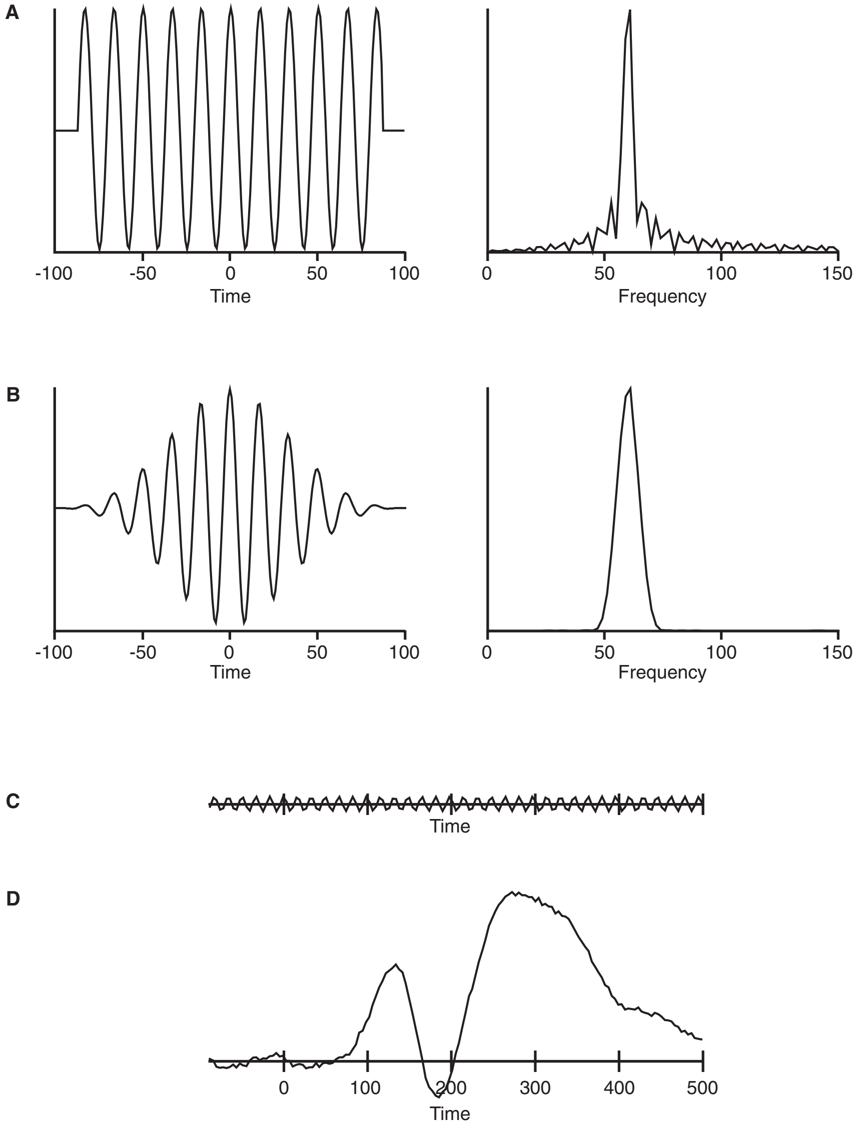


Figure 5.7 Relationship between the time and frequency domains for 60-Hz filtering. (A) Time- and frequency-domain representations of a finite-duration 60-Hz sine wave. (B) Time- and frequency-domain representations of a windowed sine wave (tapered with a Blackman window). (C) Result of convolving the 60-Hz sine wave in (A) with the ERP waveform shown in figure 5.1A, which extracts the 60-Hz component 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.[4] However, although this approach yields very nice results in the frequency domain, it may lead to significant distortions in the time domain, as I will describe 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 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.

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 figure 5.7 illustrates. 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 panel B shows, you can mitigate this problem by multiplying the sinusoidal impulse response function by a windowing

◄ **Figure 5.7** (continued)
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.

function to taper the ends of the function. This yields a frequency response function that, although 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 5.3A, which effectively extracts the 60-Hz component from the ERP waveform. Panel D of figure 5.7 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 5.3E 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.

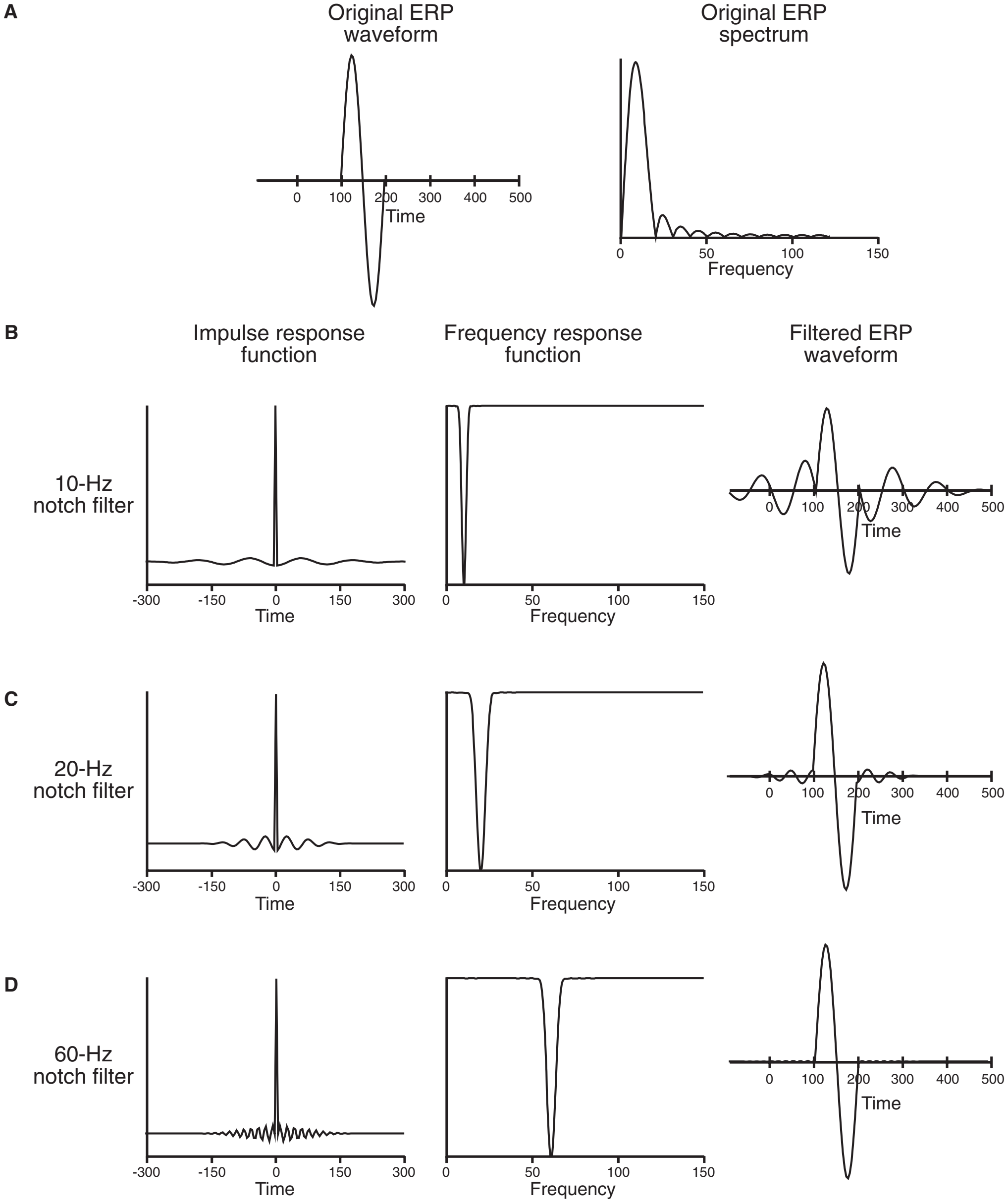
Time-Domain Distortions Produced by Filters

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.

Distortions Produced by Notch Filters

To illustrate the distortion produced by filtering, figure 5.8 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 5.8B, 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 5.8A.

The impulse response function was created by windowing a 10-Hz sine wave, as in figure 5.7B, and then converting it into a filter that removes rather than passes power at 10 Hz (see below). 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 10 Hz, 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 a more typical combinations of filters and ERP waveforms would produce, 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.


A
Original ERP waveform
0
100
200
300
400
500
Time
Original ERP spectrum
0
50
100
150
Frequency
B
Impulse response function
Frequency response function
Filtered ERP waveform
10-Hz notch filter
-300
-150
0
150
300
Time
0
50
100
150
Frequency
0
100
200
300
400
500
Time
C
20-Hz notch filter
-300
-150
0
150
300
Time
0
50
100
150
Frequency
0
100
200
300
400
500
Time
D
60-Hz notch filter
-300
-150
0
150
300
Time
0
50
100
150
Frequency
0
100
200
300
400
500
Time

The presence of oscillations in the impulse response function is not alone sufficient to produce large oscillations in the output of the filter. As figures 5.8C and 5.8D show, 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 5.8A). 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 5.8D, and may lead to much greater distortion of the ERP waveform than the minimal distortion present in figure 5.8D. 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.''

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

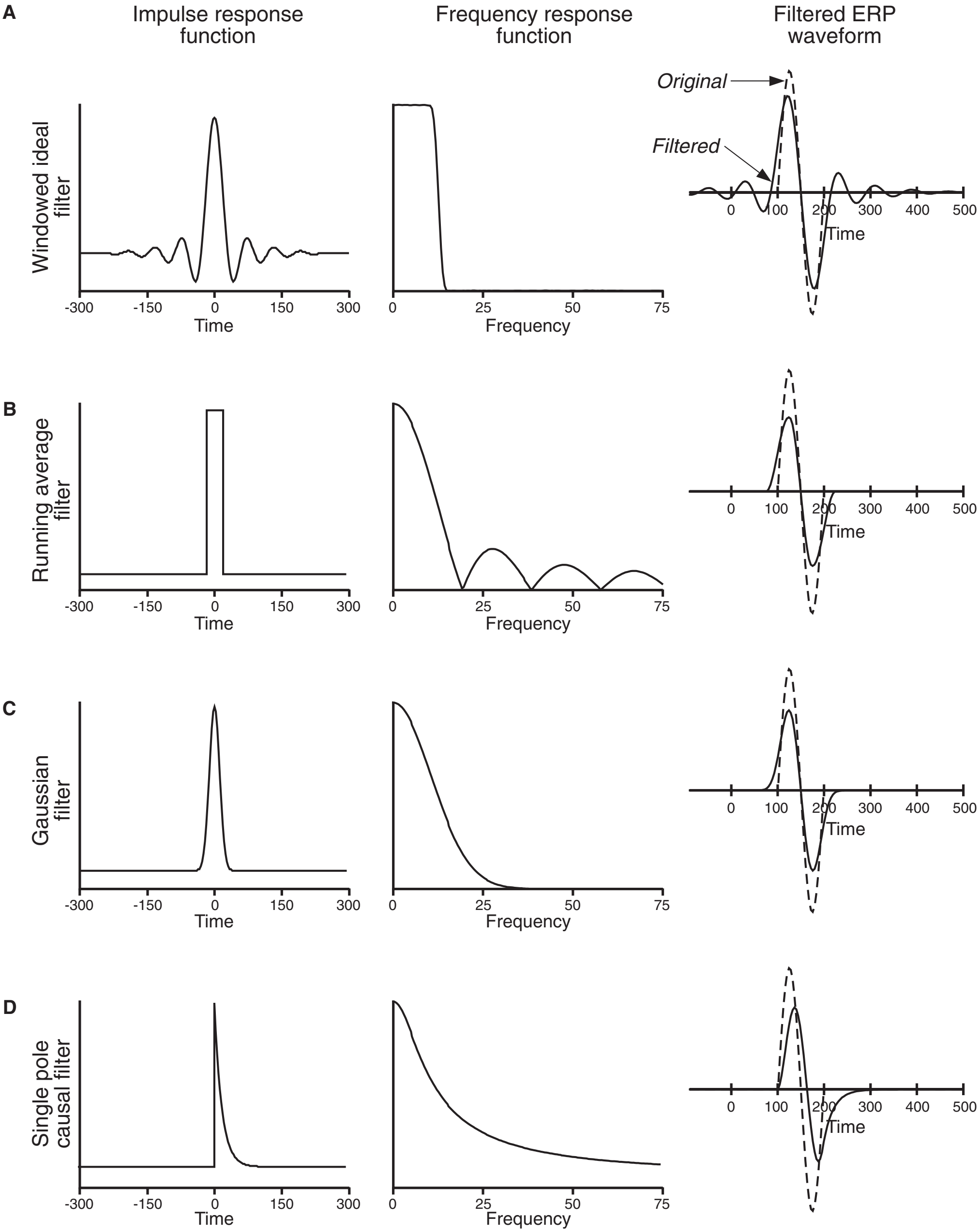
◀ **Figure 5.8** Examples of temporal distortions produced by notch filters. (A) Shows the original artificial ERP waveform (one cycle of a 10-Hz sine wave) and its Fourier transform. (B), (C), and (D) show the impulse response and frequency response functions for 10-, 20-, and 60-Hz notch filters and also display the result of applying these filters to the waveform shown in (A).

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 you cannot avoid filtering, 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 5.9A. 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 5.8A, yielding an output containing damped oscillations both before and after the time range of the original ERP waveform[5] (see figure 5.9A, 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, a 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 5.9A, 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


A
Impulse response function
Frequency response function
Filtered ERP waveform
Original
Filtered
Windowed ideal filter
B
Running average filter
C
Gaussian filter
D
Single pole causal filter
-300
-150
0
150
300
Time
0
25
50
75
Frequency
0
100
200
300
400
500
Time

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 5.9 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 the right column of figure 5.9B shows, 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 5.9, 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

◄ **Figure 5.9** 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 5.8A.

sudden onset and offset of the impulse response function lead to the presence of substantial high frequency power in the frequency response function.

Figure 5.9C 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). fMRI experiments commonly use spatial gaussian filters, 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 $185.5/\sigma$ (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 79.62/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. 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 I will discuss 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. Figure 5.9D shows an example of such a filter, displaying 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 *noncausal* 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 noncausal 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 5.9D.

The frequency response function of the causal filter shown in figure 5.9D is suboptimal for most purposes because it falls relatively slowly and never reaches zero. This is not an inherent property of causal filters, however. For example, if the gaussian impulse response function shown in figure 5.9C were simply shifted to the right so that all values were zero prior to time zero, the resulting causal filter would have the same frequency response function as the noncausal gaussian filter, but the output waveform would be shifted in time by the same amount as the shift in the impulse response function. Bessel filters, which can be implemented in analog circuitry, have an impulse response function that approximates a shifted gaussian and therefore make an excellent choice for on-line filtering (e.g., during EEG digitization). Bessel filters also have a linear phase response, which means that all frequencies in the output waveform are shifted by the same amount of time. This time shift can be computed and the filtered waveform can be shifted backwards by this amount off-line, thus eliminating the latency shifts inherent in analog filters. Unfortunately, Bessel filters are expensive to implement in analog circuitry, so they are relatively uncommon.

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 comes 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 you filter a filtered waveform 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, we can write the process of filtering twice 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 either filter would produce 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 one can accomplish high-pass filtering 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:

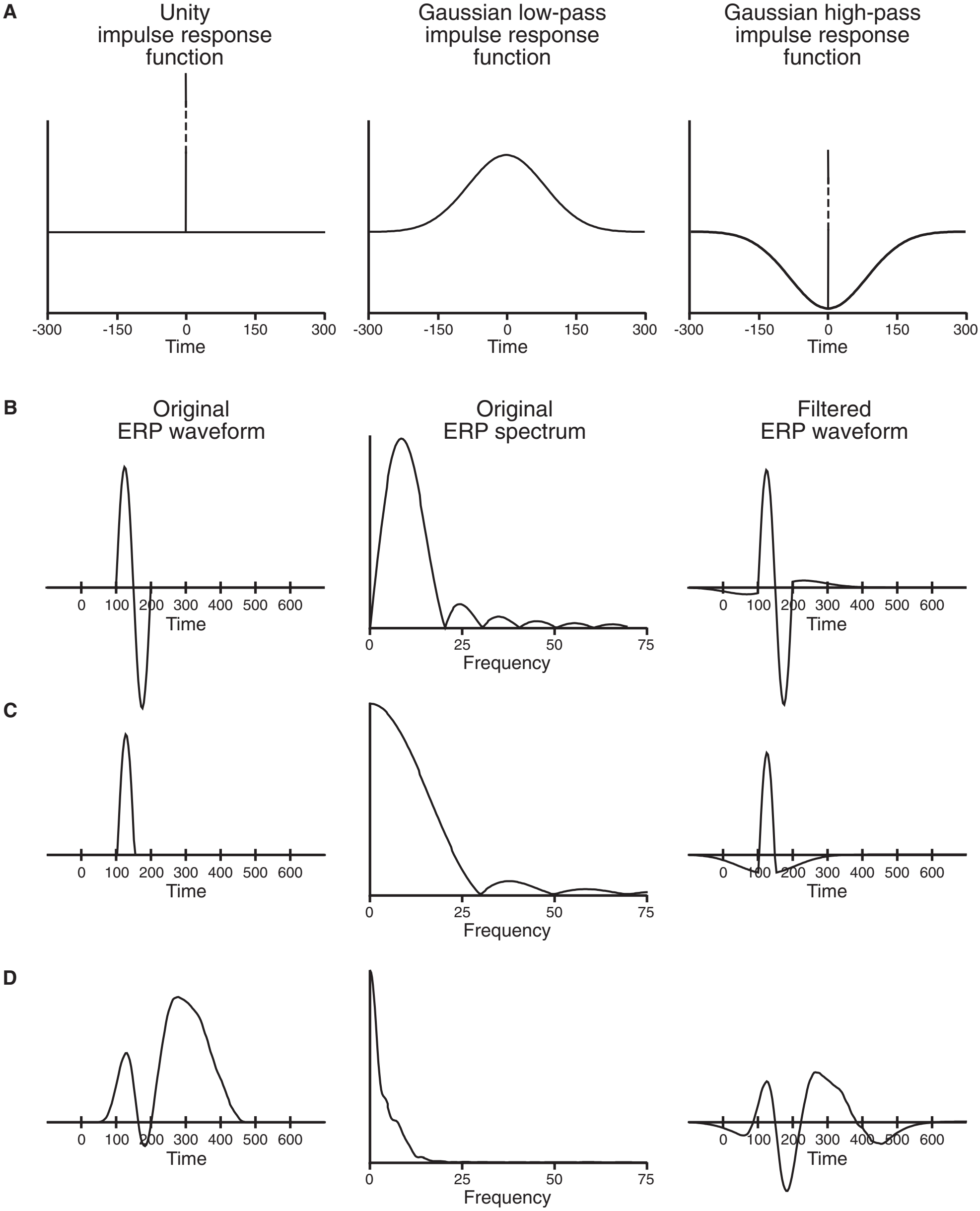
$$ERP_H = ERP - ERP_L = ERP - (IRF_L * ERP)$$

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 IRFH that will accomplish high-pass filtering in a single step:

$$\begin{aligned} ERP_H &= ERP - (IRF_L * ERP) \\ &= (IRF_U * ERP) - (IRF_L * ERP) \quad [\text{because ERP} = \text{IRF}_\text{U} * \text{ERP}] \\ &= (IRF_U - IRF_L) * ERP \quad [\text{because of the distributive property}] \\ &= IRF_H * ERP, \quad \text{where } IRF_H = IRF_U - IRF_L \end{aligned}$$

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 5.10A 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 5.10A, left) consists of a single spike of magnitude 1.0 at


A
Unity impulse response function
Gaussian low-pass impulse response function
Gaussian high-pass impulse response function
-300
-150
0
150
300
Time
B
Original ERP waveform
Original ERP spectrum
Filtered ERP waveform
0
100
200
300
400
500
600
Time
0
25
50
75
Frequency
C
D

time zero, which is very large compared to the values at individual time points in the gaussian low-pass impulse response function (figure 5.10A, 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 5.10A.

A related attribute of the class of filters discussed here is that they are linear, so you can combine filtering with other linear operations in any order. 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.[6] 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.

Distortions Produced by High-Pass Filters

Figure 5.10 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 one can observe 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 in figures 5.10C and 5.10D. 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 5.10B, these subcomponents will lead to opposite-polarity copies of the impulse response function, which will cancel each

◀ **Figure 5.10** Construction and application of a gaussian high-pass filter. (A) A unity impulse response function multiplied by 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.

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 one polarity dominates the ERP waveform.

The distortions in figure 5.10 are in some sense similar to the distortions the gaussian low-pass filter shown in figure 5.9C produced, 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 5.10 is also somewhat broader than the distortion in figure 5.9C). 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 using a high-pass filter produces opposite-polarity spreading. Although these distortions are similar in many ways, the distortions the high-pass filter produces 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 5.10D.

Figure 5.11 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 in figure 5.9A. 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 5.11B, 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, you must take extreme caution in interpreting high-pass-

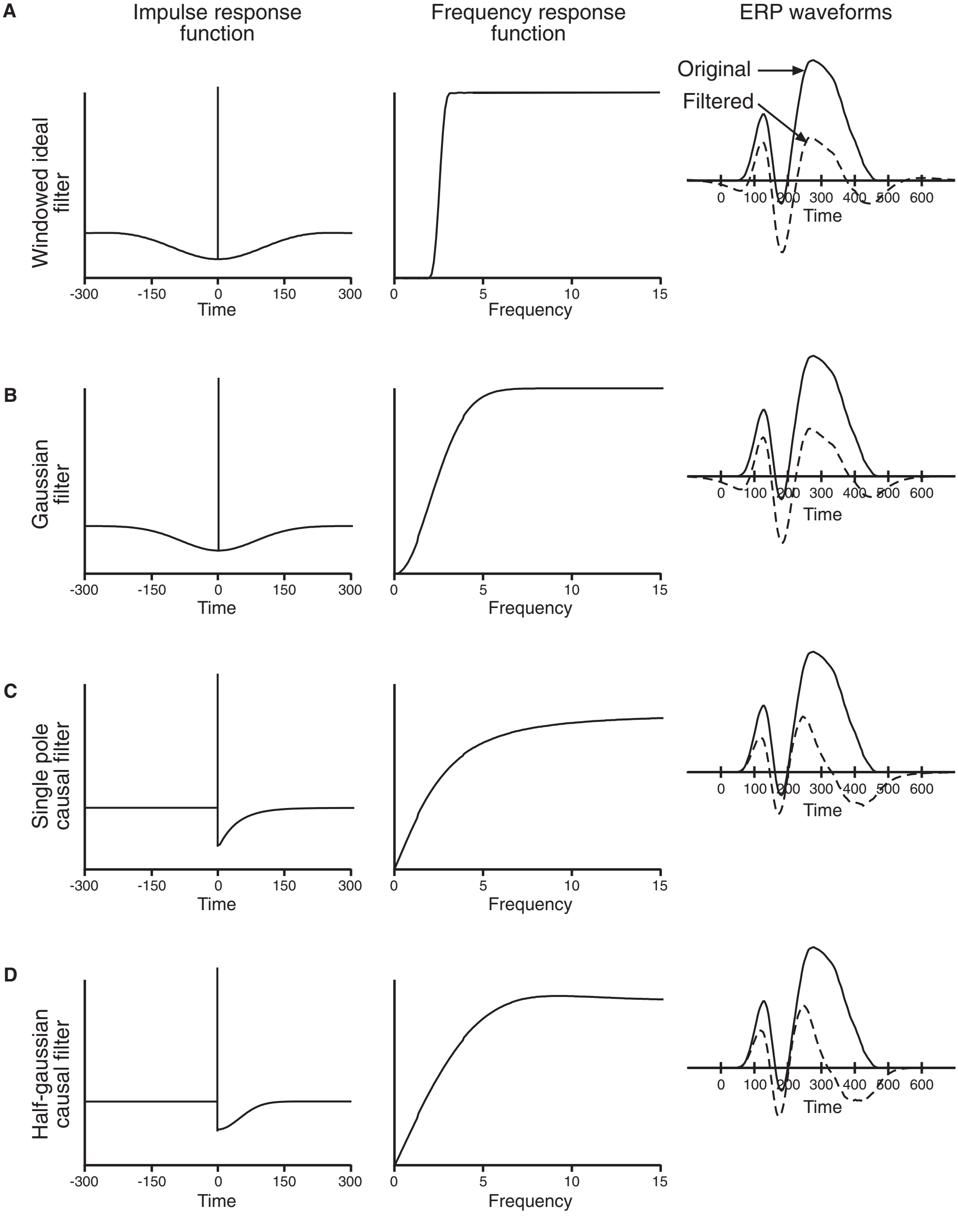


Figure 5.11 Application of four different high-pass filters to a sample ERP waveform. All have a 50 percent amplitude cutoff near 2.5 Hz.

filtered waveforms, even when using a gaussian impulse response function.

Figure 5.11C 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 noncausal 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 5.11C also has a relatively gradual roll-off; this can be improved by using the filter in figure 5.11D, which uses the right half of a gaussian in its impulse response function. Like the filter in figure 5.11C, 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.

In most cases, noncausal 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 5.11 push the information exclusively to later time points. Figure 5.12 illustrates this. 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 bidirectional distortions produced by noncausal 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-

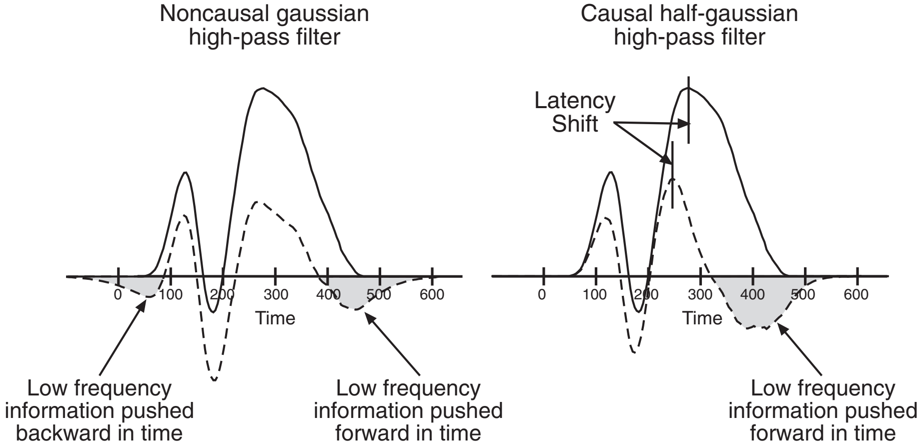


Figure 5.12 Comparison of a noncausal gaussian high-pass filter and a causal half-gaussian high-pass filter. Note that the noncausal filter produces obvious distortions at the beginning and end of the waveform, whereas the causal filter produces no distortion at the beginning of the waveform. Note also that the causal filter produces a large latency shift, but the noncausal filter does not.

frequency information. Using a causal filter such as those shown in panels C and D of figure 5.11, however, will minimize distortion of the early components because low-frequency information from the late components will not be pushed backward into the time range of the early components. Note, 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 noncausal filters. In addition, causal filters may produce substantial latency shifts that may be problematic in some cases.

Recommendations Revisited

I discussed some recommendations about filtering early in this chapter. 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. My second main recommendation is that before you use a filter, think about exactly how it might distort your data. 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.