Online Chapter 11
A Closer Look at Averaging: Convolution, Latency Variability, and Overlap

Overview

You might think that we discussed everything there is to know about averaging in the chapter on averaging (Chapter 8). Although we discussed the major issues, we did so in a relatively informal manner. It's time to get a little more formal by introducing the mathematical concept of convolution. This concept will provide you with a deeper understanding of many important things about ERPs that seem quite different on the surface, including: how latency variability in an ERP component influences the shape of the averaged ERP waveform; exactly how overlap from the previous trial distorts the ERP on the current trial (and how this overlap can be eliminated); how timing errors in your stimuli influence the averaged ERP waveform; how wavelets can be used to extract specific frequencies from the EEG in time-frequency analysis; and how filters work to attenuate specific frequency bands from your data (and can result in massive distortions and incorrect conclusions). The present chapter focuses on applying convolution to latency variability and overlap, and Online Chapter 12 will apply it to filtering and time-frequency analysis.

Convolution is not usually considered a fundamental concept in ERP research. For example, I read hundreds of ERP papers in my first few years doing ERP research, and I never encountered it. However, once I learned about convolution, it became clear to me that it is something that every ERP researcher should know about (Box 11.1 describes my first encounter with convolution).

You may have learned about convolution already in some other context. If so, this chapter should be very easy for you to read, although I will describe convolution in a way that is probably quite different from the way you learned it before.

If convolution is new to you, it may sound intimidating (especially if you're not mathematically oriented). Fortunately, convolution is quite simple, and I will break it down into a combination of addition, subtraction, multiplication, and division. If you can do simple arithmetic, you can understand convolution. My goal in this chapter is to provide an intuitive explanation, so there are no equations (if you like equations, you'll see plenty in the next chapter).

I have tried to explain convolution a dozen different ways over the years, but I have found that the best way is to start with the example of how trial-by-trial latency variability influences the averaged ERP waveforms. We discussed this issue informally in the chapter on averaging, so you should already have an intuitive understanding of how latency variability influences averaged ERP waveforms. I will build on this intuitive understanding in the present chapter.

I will then discuss how convolution helps us to understand exactly how overlapping activity from previous and subsequent stimuli distorts the ERP waveform on the current trial. Overlap is common in ERP research, because the brain's response to one stimulus is often still ongoing when the next stimulus occurs. In some cases, overlap is a major confound that makes experiments uninterpretable. In fact, overlap-related confounds are on my list of the Top Ten Reasons for Rejecting an ERP Manuscript (see Online Chapter 15). In other cases, however, overlap has absolutely no impact on the conclusions that you can draw from an experiment. I will therefore provide some guidelines about when overlap is likely to be a problem, along with some advice about how to minimize or eliminate overlap.

I will end the chapter by providing an intuitive explanation of how convolution is related to filtering.

Latency Variability and Convolution

When I introduce convolution in the context of latency variability during the ERP Boot Camp, I always get asked two questions. The first is: Isn't this just how averaging works? The answer is: Yes, this is just how averaging works. I am just going to take what you already know about averaging and express it in a

slightly different way. However, this different way of talking about averaging is very powerful, and it will help you to understand many things about ERPs. And by explaining convolution in terms of something you already understand (averaging), I can make everything very concrete. The second question people ask is: How can I use the concept of convolutions to analyze my data (e.g., to recover the waveshape of the single-trial ERP component)? To answer this second question would require discussing *deconvolution*, which is much more complicated and is often impossible to apply to ERPs. My goal in this section is to teach you a way of understanding ERPs more deeply, not a data analysis technique. This understanding will be very valuable as you look at ERP waveforms. Moreover, it will prepare you to start reading more advanced books and journal articles that explain when and how you may be able to deconvolve your waveforms. In addition, when I talk about overlap later in the chapter, I will describe how a deconvolution technique can be used to estimate and subtract away overlap.

Box 11.1- My Introduction to the Joys of Convolution

When I was in graduate school, we had a weekly 2-hour lab meeting that started promptly at 4:08 every Monday afternoon. Almost every session consisted of a presentation of recent research by a graduate student or postdoc. I learned more from those lab meetings than from any other experience in my entire life.

One week, Marty Woldorff gave a presentation on some analytic work he had been doing to understand how overlapping activity from previous trials distorted the current trials and how this overlap could be estimated and removed. The presentation ended up spilling over into a second week, and these were four of the most enlightening hours of my life (which shows you that I am truly an ERP geek).

Marty explained very clearly what a convolution was, how it was related to the overlap problem, and how it was related to filtering. He went on to describe a method he had developed for estimating and removing overlap (the *ADJAR filter* technique, which I will describe later in this chapter). I never looked at ERPs the same way again.

Marty wrote a paper describing his analysis of overlap and his new overlap removal technique (Woldorff, 1993), and it's on my list of the *Top Ten Papers Every ERP Researcher Must Read*. Ironically, I provided an incorrect citation to this paper in the first edition of this book. I will now make amends by giving the correct citation in this box as well as in the reference list. Here it is:

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

Frequency and Probability Distributions

The first step in understanding convolution in the context of ERP averaging is to understand *frequency distributions* and *probability distributions*, which were described in detail in Chapter 9 (see Figure 9.8). As a reminder, consider the reaction time (RT) distributions shown in Figure 11.1, which show how often responses occurred within specific time bins in an experiment with 100 trials. For example, Figure 11.1A uses 50-ms time bins and shows that 7 of the 100 RTs occurred in the time bin centered at 300 ms (i.e., from 275-325 ms), 17 occurred in the time bin centered at 350 ms, 25 occurred in the time bin centered at 400 ms, etc. Figure 11.1B shows the same data, but expressed as the probability instead of the frequency of a given RT bin.

Figure 11.1C shows actual RT probability distributions for a group of schizophrenia patients and a group of control subjects (from the study of Luck, Kappenman, Fuller, Robinson, Summerfelt, & Gold, 2009). The distributions are not symmetrical but are instead skewed to the right, which is almost always true of RT distributions. Moreover, differences between groups or conditions often consist of increases in skewness rather than a shift of the entire RT distribution. In the data shown in Figure 11.1C, for example, the fastest RTs were similar in the two groups, but patients had more long RTs than controls, leading to a more skewed distribution.

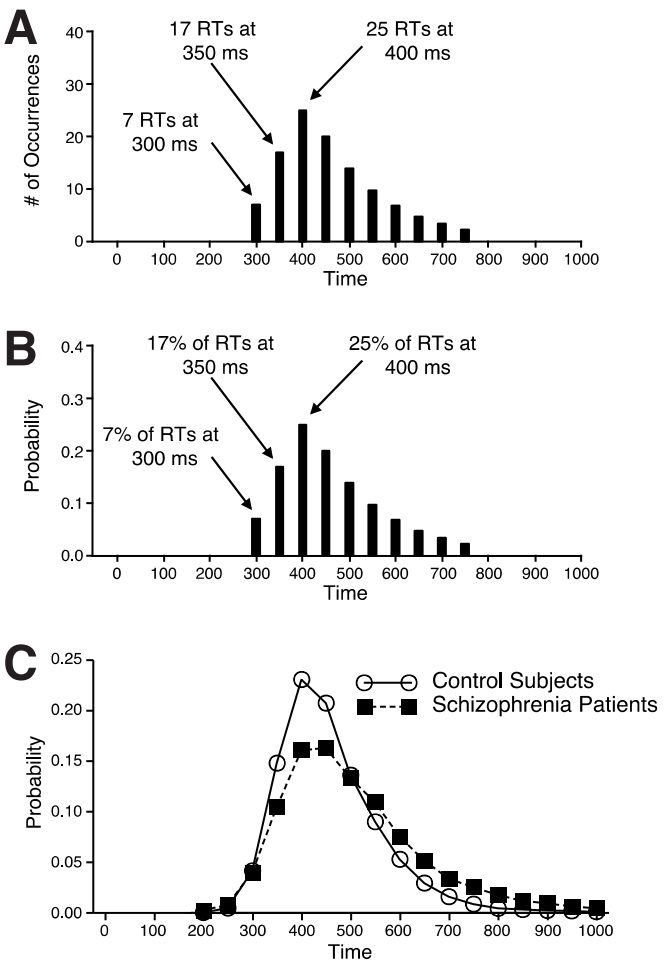

Figure 11.1
(A) Typical frequency distribution of reaction time (RT). The height of each bar represents the number of occurrences of an RT that occurred within ±25 ms of the latency of the bar. (B) Same as (A), but showing probability rather than frequency (by dividing each bar by the number of trials, which in this example is 100). (C) Actual RT probability distributions from a study of schizophrenia patients and control subjects (Luck et al., 2009). These distributions were aggregated across the subjects in each group. Note that this is identical to Figure 9.8 in Chapter 9.

To link these RT distributions with ERPs, let's consider an imaginary experiment in which the peak of the P3 component on a given trial always occurs at exactly the same time as the behavioral response, and the probability distribution for the single-trial P3 latency is therefore the same as the probability distribution for RT. (Although this will be a useful assumption for the initial explanation, it is not an intrinsic part of the logic behind using convolution to understand ERPs, so don't worry about whether it is a valid assumption in real experiments.) We'll also assume that the P3 waveshape is the same no matter what the P3 latency is on a given trial. This is actually a fairly reasonable assumption that is common in methods that attempt to characterize single-trial ERPs. Figure 11.2A shows the assumed shape of the P3 wave on two trials, one with a peak latency of 400 ms and another with a peak latency of 500 ms (corresponding to trials with RTs of 400 and 500 ms, respectively). To keep the math simple, we're assuming that the P3 has a peak amplitude of 1 μV on the single trials. We're also assuming that the distribution of P3 latencies in this experiment is the same as the RT distribution shown in Figure 11.1A.

Now let's think about how we would make an average of 100 trials, each with a P3 that peaks at the time of the behavioral response. To compute the mean of a set of values, you simply sum all the values together and then divide by the number of values. Consequently, the average of 100 single-trial waveforms can be computed by summing the waveforms and then dividing the summed value at each time point by 100. Normally you might do this by summing together the waveforms in the order that the trials occurred (i.e., waveform for trial 1 + waveform for trial 2 + …). However, you could also do this by adding together the waveforms for the 7 trials with a 300-ms latency, and then adding the 17 trials with a 350-ms latency, etc., until the waveforms for all 100 trials have been added together. These two approaches both consist of adding together the same 100 waveforms, but in different orders. However, the latter approach is more easily related to the frequency distribution.

Figure 11.2B shows how we can use the frequency distribution shown in Figure 11.1A in combination with the single-trial P3 waveform shown in Figure 11.2A to get the sum of all the waveforms. To sum together the 7 trials with a peak latency in the 300 ms bin, we can take the single-trial P3 waveform, center it at 300 ms, and multiply it by 7. We can do the same thing for the 17 trials with a peak P3 latency in the 350 ms bin; we take the single-trial P3 waveform, center it at 350 ms, and multiple it by 17. If we do this for each latency bin, we get a waveform that is centered at each bin and multiplied by the number of occurrences of that bin. If we then sum across these waveforms, we get the sum of all 100 trials (see the dashed waveform in Figure 11.2B). We could then get the average across all 100 trials by just dividing this summed waveform by 100. This introduces

a little bit of error, because we are assuming that all of the trials with a latency within a particular 50 ms bin have a peak in the center of the bin, but we could make that error infinitesimally small by using very small bins in our frequency distribution[i].

This may seem like a lot of work to get an average, but it makes it clear exactly how the shape of the averaged ERP waveform is related to a combination of the shape of the single-trial waveforms and the distribution of latencies. Note, for example, that the RT distribution is skewed to the right, and so is the averaged ERP waveform. This is the typical pattern for real ERP data in experiments with large P3 waves (see, e.g., Figure 1.4 in Chapter 1 and Figure 8.8B in Chapter 8). If you saw an averaged ERP waveform with this shape and didn't look at the corresponding RT distribution, you might think that the skewed shape of the waveform reflects the shape of the P3 on the individual trials. However, once you think about how the distribution of latencies affects the averaged ERP waveform, it's easy to understand why the waveform has a skewed shape.

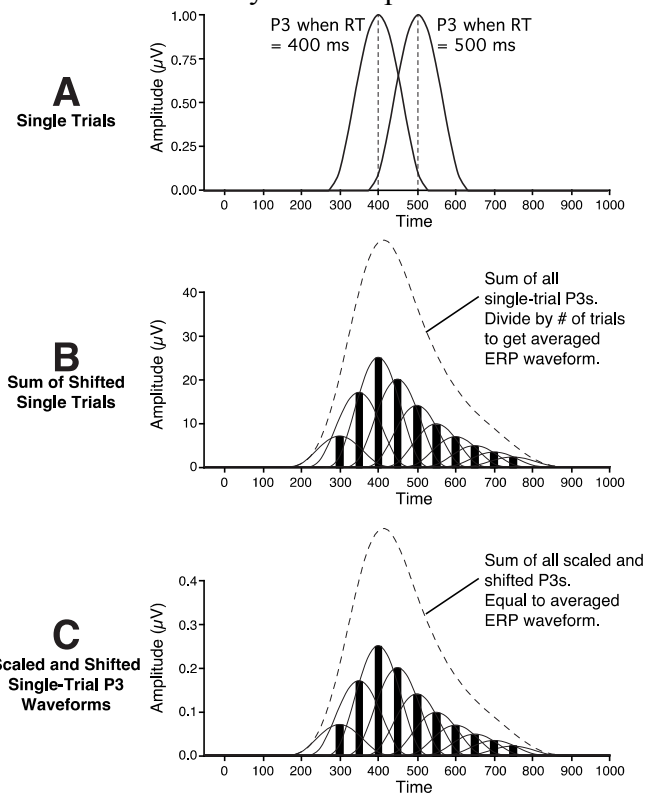

Figure 11.2
Example of the use of convolution to understand the effects of latency jitter on the averaged ERP waveform. (A) P3 waveform on two single trials, one when the reaction time (RT) was 400 ms and one when the RT was 500 ms. The peak of the P3 is assumed to occur at the same time as the behavioral response. (B) Use of the frequency distribution of single-trial P3 latencies to create an average across trials. The single-trial P3 waveform for a given time bin is moved to the time point of the bin and multiplied by the number of trials in the bin. After this is done for each bin, all the waveforms are summed together. The average would be created by dividing this sum by the number of trials. (C) Same as (B), but using the probability distribution rather than the frequency distribution so that it is not necessary to divide by the number of trials at the end (note the difference in amplitude scale between Panels B and C). This is equivalent to convolving the RT probability distribution with the single-trial P3 waveform.

Probability Distributions and Convolution

If we use a probability distribution rather than a frequency distribution to make our averaged waveform, we can finally get to the concept of a convolution. In Figure 11.2B, we created the averaged ERP waveform by multiplying the single trial P3 waveform by the frequency of each bin, summing the multiplied waveforms for the different bins, and then dividing the sum by the number of trials. We can combine the first step (multiplying by the frequency of a bin) and the last step (dividing by the total number of trials) by multiplying the single-trial waveform by the probabilities in the probability distribution instead of by the frequencies in the frequency distribution (because a probability distribution is just a frequency distribution in which each bin has been divided by the total number of trials). Figure 11.2C shows this slightly different approach. Each bar from the probability distribution shown in Figure 11.1B is replaced by a copy of the single-trial P3 waveform, scaled (multiplied) by the size of the bar and shifted so that it is centered on the bar. For example, the bar with a height of 0.07 at 300 ms in the probability distribution is replaced by a single-trial waveform that is multiplied by 0.07 and centered at 300 ms. This *scaling and shifting* is done for each bar in the probability distribution, and all of the scaled and shifted waveforms are then summed together. This

[i] The use of 50-ms bins in this example will introduce some timing error. A trial with a P3 latency of 322 ms, for example, will end up being treated the same as a trial with a P3 latency of 291 ms, and both will be represented by centering the single-trial P3 at 300 ms. If we were going to apply this approach to real data, we could use a much smaller bin size (e.g., 2 ms), which would virtually eliminate the timing errors. I'm using a fairly wide bin size in this example simply because it makes the figures look nicer. Note that serious "math people" would completely eliminate the whole binning process by using calculus. Personally, I'd rather have a timing error of ±1 ms than use calculus. I'd also rather have a tooth pulled than use calculus. But if you like math, you can use calculus to do convolution with continuous functions.

gives us the averaged ERP waveform, which is shown by the dashed line. If you look at the voltage scale in Figure 11.2C, you'll see that the peak amplitude of the averaged waveform is about what you'd expect when a 1 μV P3 wave is averaged over 100 trials with a considerable amount latency jitter.

You probably didn't realize it, but we just computed a convolution! When you take one waveform (or set of bars) and replace each point in the waveform with a scaled and shifted version of another waveform, this process is called convolution[ii]. In our example, we convolved the single-trial P3 waveform with the probability distribution of single-trial P3 latencies to see how these two waveforms combine to give us the averaged ERP waveform. In the context of averaged ERPs, convolution is just a fancy term for averaging across trials with different latencies (assuming that you know the distribution of latencies). However, this simple process can also be also used to explain filtering, overlap from previous trials, and the effects of stimulus timing errors on the ERP waveform. Here's a definition for you:

Convolution: To convolve two waveforms, you simply replacing each point in one waveform with a scaled-and-shifted copy of the other waveform and then sum together these scaled-and-shifted waveforms.

If you know that P3 latency is tightly coupled to RT, and the distribution of P3 latencies is therefore closely related to the RT distribution, a deconvolution method has been developed to compute an estimate of the single-trial P3 waveshape (Hansen, 1983)[iii]. I don't know of any studies that have actually used this method. However, even though you probably won't compute the single-trial P3 waveshape, you can look at your RT distributions and understand in general why your averaged ERPs look the way they do. And if you look at the differences in RT distributions across conditions or groups, you may have a better understanding of why the ERP waveforms look different across conditions or groups.

Some ERP components vary in latency a great deal from trial to trial (e.g., P3 and LRP). Other components are tightly time-locked to stimulus onset (e.g., the auditory brainstem responses and the visual P1). When both types of components are present in the same experiment, the averaged ERP waveforms will be approximately equal to the single-trial waveshape for the tightly-locked components plus the convolution of the variable-timing components with their probability distributions.

In many cases, you will not have any way to determine the distribution of single-trial ERP latencies. In these cases you will not be able to use your newfound knowledge of convolution to compute anything. However, when you look at an averaged ERP waveform, you will understand how it can misrepresent the single-trial waveforms, and this may lead you to some interesting hypotheses about the distribution of single-trial waveforms.

Understanding Overlap

Now that we've discussed convolution, we're ready to discuss the overlap problem. Overlapping ERPs from previous and subsequent stimuli will distort averaged ERP waveforms in ways that are sometimes subtle and sometimes obvious, and it is important to understand how this arises and when it might lead you to misinterpret your data (for a detailed treatment of this topic, see Woldorff, 1993). Overlap arises when the response to the previous stimulus has not ended before the baseline period prior to the current stimulus or when the subsequent stimulus is presented before the ERP response to the current stimulus has terminated. This problem is particularly acute when stimuli are presented rapidly (e.g., 1 second or less between stimulus onsets). However,

---

[ii] If you have learned about convolution previously, this might sound different. Convolution is typically described in terms of how the value at each time point is computed from the sum of a set of scaled values, but here we are doing the scaling and shifting first and then summing at the end. Both approaches lead to the same value, but with a different order of operations. I find that my way of explaining it (which I got from Marty Woldorff, my officemate in graduate school) makes it easier to understand the purpose of convolving two waveforms. The other way of thinking about it is more useful when you actually perform the computations. I will describe more formally how these two ways of thinking about convolution can be interconverted in Online Chapter 12.

[iii] More precisely, this technique allows you to estimate the waveform for the sum of all response-locked components, not just the P3 wave. However, the P3 wave is often much larger than the rest of the response-locked activity, and in these cases you will get an approximation of the P3 waveshape.

ERP waveforms can last for over 1000 ms, and overlap can significantly distort your data even at relatively long interstimulus intervals.

Overlap in a Gedankenexperiment

The overlap problem is illustrated in Figure 11.3 for a Gedankenexperiment (thought experiment) in which a stimulus is presented every 300-500 ms and elicits P2, N2, and P3 waves. The actual waveform elicited by a given stimulus—without any overlap—is shown in panel A. Note that the prestimulus period is flat and that the waveform falls to zero by 1000 ms poststimulus. This is the ERP waveform that we would have if we didn't have any overlap from the previous and subsequent stimuli (but which we cannot directly record in this experiment because of overlap from the previous and subsequent trials).

Panel B shows the waveforms that would be produced by the same stimulus if it appeared 300, 400, or 500 ms prior to the current stimulus; these are just copies of the original waveform shifted to the left by various amounts. Panel C shows the overlap that would be produced, on average, if the preceding stimulus could happen at any time between 300 and 500 ms prior to the current stimulus. This overlap waveform was created by averaging copies of the waveform from Panel A after shifting it to the left by 300, 304, 308, …, 500 ms (because the sampling rate was 250 Hz, with a sample every 4 ms).

Panels D and E show the same thing, but shifting the Panel A waveform to the right to simulate overlap from the subsequent stimulus.

Note that the jittered timing in this thought experiment leads to a "smearing" of the averaged waveform for the overlapping stimuli. That is, the relatively sharp positive and negative peaks at the beginning of the original waveform are mostly (but not entirely) eliminated in the overlapping waveform. If you look at Panels B and D, you can see why this happens: the P2 when the waveform is shifted by 300 ms is at the same time as the N2 when the waveform is shifted by 400 ms, and these positive and negative deflections cancel out when the waveforms are averaged together. In addition, the P3 ends at around 500 ms when the waveform is shifted 500 ms leftward, whereas it ends at around 700 ms when the waveform is shifted 300 ms leftward. When these are averaged together, the offset of the P3 is much more gradual. This is a lot like the "smearing" in the averaged ERP waveform that resulted from trial-to-trial latency jitter in P3 latency (see Figures 6.8, 6.9, and 7.2). This is because both latency jitter and overlap can be understood in terms of convolution. I will say more about this in a little bit.

Panel F shows the sum of the true ERP elicited by the current stimulus and the overlapping waveforms from the previous and subsequent stimuli. This sum is exactly what you would obtain if you simply averaged together all of the stimuli in this Gedankenexperiment, because the overlap is present on the single trials. The distortion due to overlap in this Gedankenexperiment is quite severe. First, the prestimulus baseline is completely distorted, and this leads the P2 to seem much larger than it really is (compare the P2 in Panels A and F). Second, the P2 appears to start before time zero (which is always a good indication that something is amiss). Third, the N2 component is much less negative than it should be. Finally, there is a late positive peak at 900 ms that is completely artifactual (because it reflects the response to the next stimulus rather than the current stimulus).

When Overlap is and isn't a Problem

Overlap is mainly problematic when it differs between experimental conditions. Recall the Gedankenexperiment that was described at the beginning of Chapter 4, in which two rare stimuli never occurred in succession. This meant that the frequent stimuli were sometimes preceded by a rare stimulus, but the rare stimuli were always preceded by a frequent stimulus. This led to less overlap from the previous trial's P3 wave for the rare stimuli than for the frequent stimuli, and this led to artifactual differences in amplitude (you might want to go back to Chapter 4 and take a look at Figure 4.3). This is on my list of the Top Ten Reasons for Rejecting an ERP Manuscript (see Chapter 16).

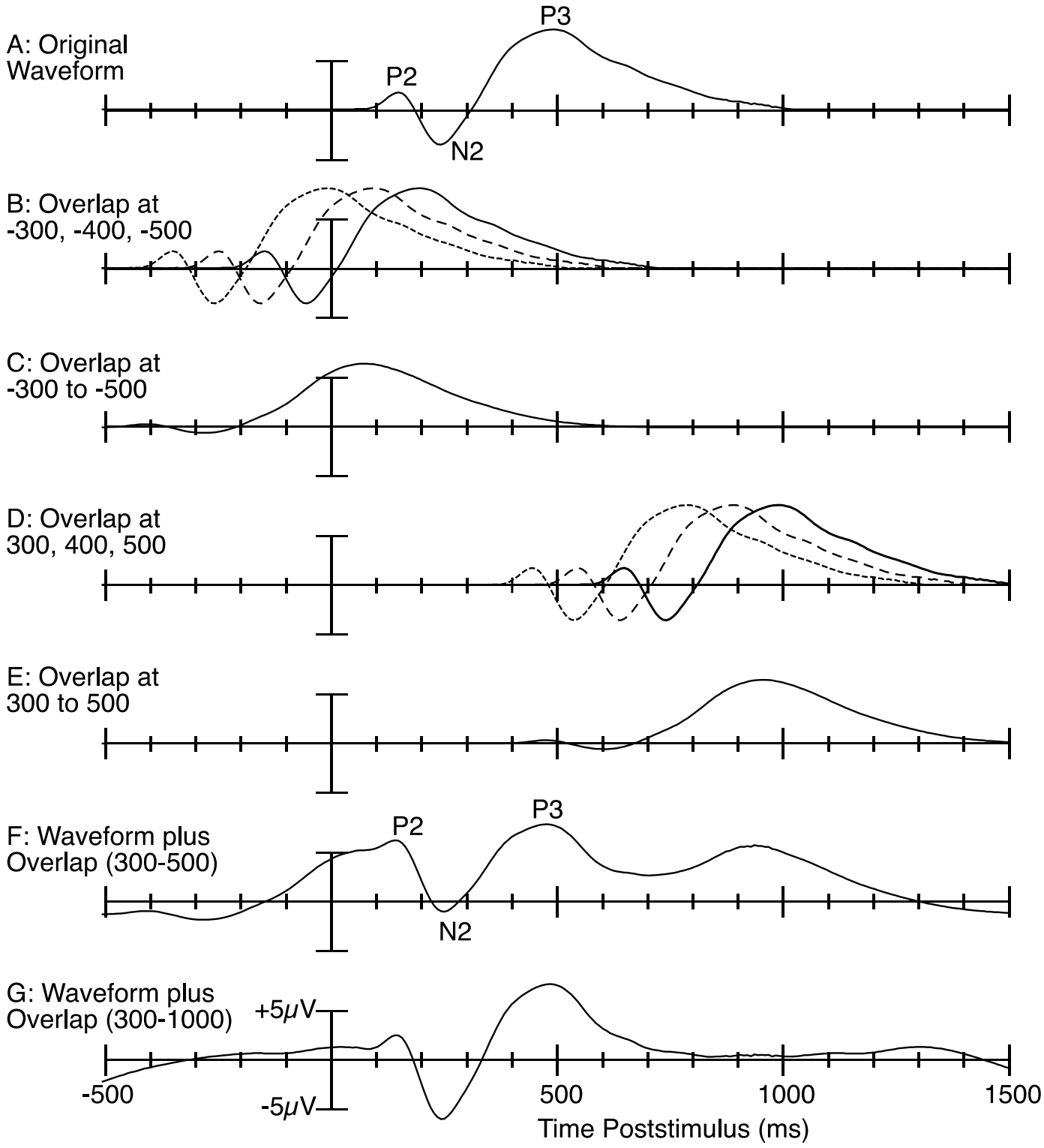

Figure 11.3
Example of the use of convolution to understand the effects of latency jitter on the averaged ERP waveform. (A) P3 waveform on two single Example of the problem of overlapping waveforms. (A) Hypothetical overlap-free ERP waveform. (B) Waveforms produced by the previous stimulus when it appears 300, 400, or 500 ms prior to the current stimulus. (C) Average waveform produced by the previous stimulus when it appears at random times between 300 and 500 ms prior to the current stimulus. (D) Waveforms produced by the subsequent stimulus when it appears 300, 400, or 500 ms after the current stimulus. (E) Average waveform produced by the subsequent stimulus when it appears at random times between 300 and 500 ms after the current stimulus. (F) Sum of the original waveform and the overlapping waveforms from (C) and (E). (G) Sum of the original waveform and the overlapping waveforms that would be produced if the interval between stimuli was increased to 300-1000 ms.

Overlap is not usually a problem when it is identical across conditions. My favorite example of this is the original N400 experiment (see Figure 3.13 in Chapter 3). In this experiment, sentences were presented one word at a time, and the last word in the sentence could either be semantically congruous ("She walked to school on a bright sunny day") or semantically incongruous ("She walked to school on a bright sunny pizza"). The main goal was to examine the ERP elicited by the last word in the sentence. However, the ERP to the previous word was still ongoing when the last word was presented, creating very substantial overlap. This is not a problem, however, because the ERP activity up to the point of the last word was equivalent for sentences with congruous and incongruous endings. Consequently, any differences in the ERPs to congruous and incongruous words could not have been a result of the overlapping activity from the preceding portion of the sentence.

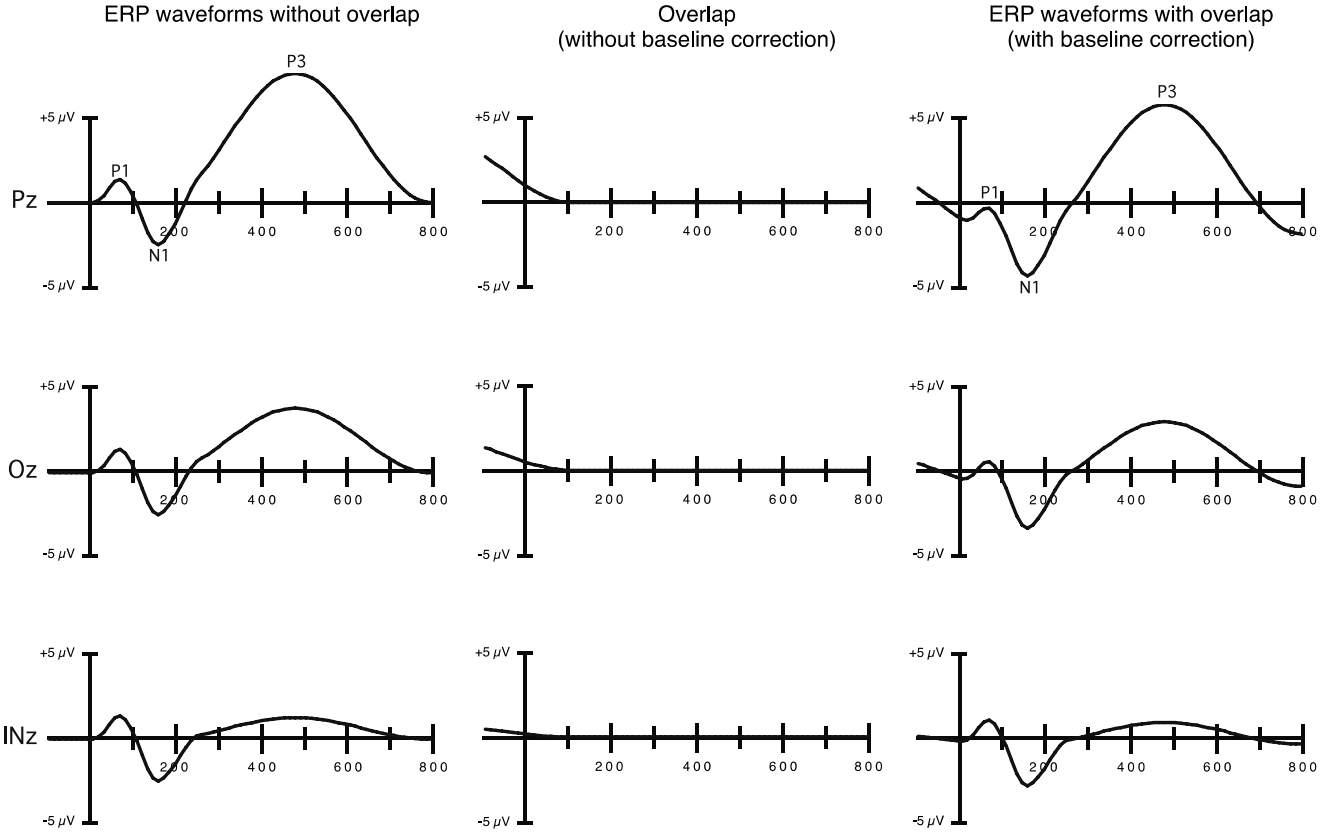

Figure 11.4
Example of the effects of overlap on scalp distributions. If the ERP to the current stimulus could be recorded without overlap, the P1 and N1 components would have equal amplitude at all three electrode sites. The overlapping activity from the previous trial mainly consists of the P3 wave, which is largest at Pz. This contaminates the baseline of the waveforms that are actually recorded (which contain the overlapping activity), dramatically distorting the scalp distributions of the P1 and N1 waves.

This kind of benign overlap is present in many experiments, and it is not usually a problem. Preparatory activity is also present prior to stimulus onset in many experiments, and this is also not usually a problem as long as it is equivalent across conditions.

However, non-differential overlap can be a problem if you are interested in scalp distributions (or source localization, which depends on scalp distributions). This is illustrated in Figure 11.4, which shows ERPs from three posterior scalp sites (Pz, Oz, and INz). In the absence of overlap (left column), the P1 and N1 waves have equal amplitudes across these sites, whereas the P3 is large at Pz, intermediate at Oz, and small at INz. The overlap from the previous trial mainly consists of the P3 (because the P3 is at the end of the waveform from the previous trial). The distortion of the baseline produced by the overlap is therefore largest at Pz, intermediate at Oz, and smallest at INz (middle column). When this overlap is combined with the ERP elicited by the current

trial, the baseline correction pushes the waveforms downward a lot at Pz, much less at Oz, and even less at INz (right column).

The overlap in this example has a very large effect on the scalp distributions of the P1 and N1 waves. The overlap essentially causes the scalp distribution of the P3 wave to be subtracted from the scalp distribution of the P1 and N1 waves (because the overlap in the baseline mainly consists of P3, and baseline correction involves subtracting the baseline). As a result, the P1 wave in the actual waveforms that you would record (right column of Figure 11.4) is largest at INz, intermediate at Oz, and slightly negative at Pz. Similarly, the N1 wave in the recorded waveforms is largest at Pz, intermediate at Oz, and smallest at INz. Without the overlap, both the P1 and N1 would have equivalent amplitudes across the three sites, so this is a large distortion. Thus, even non-differential overlap can be a problem when you are looking at scalp distributions or trying to localize the ERPs. The same is true of preparatory activity that contaminates the prestimulus baseline period. This issue is discussed at length by Urbach and Kutas (2006).

Keep in mind that you won't know what the overlap-free waveforms look like in an actual experiment, and there is no way to avoid doing some kind of baseline correction. Thus, the overlap will necessarily produce the kind of distortion of scalp distribution that is shown in the right column of Figure 11.4.

As was discussed in Chapter 4, this sort of problem can often be eliminated by means of difference waves. If the experiment shown in Figure 11.4 included two conditions, A and B, and the overlap was the same for conditions A and B, then we could subtract away the overlap by making an A-minus-B difference wave. We could then look at the scalp distribution of the difference wave without any contamination from the overlap.

Overlap and Convolution

When the SOA is variable (jittered), this causes the overlap to be a "smeared out" version of the ERP waveform. For example, the overlap arising from an SOA range of 300-500 ms in Figure 11.3C is a smeared out version of the waveform in Figure 11.3A. If we use a broader range of SOAs, the overlap will be even more smeared out. This is illustrated in Figure 11.3G, which shows what the data would look like with an SOA range of 300-1000. The overlap is much flatter in this case than it was with an SOA range of 300-500 ms, leading to less distortion of the ERP waveform. This is very much like the smearing that we saw with trial-to-trial jitter of P3 latency in Chapter 6. When the range of jitter was large (Figure 6.7A), this led to a broader averaged waveform with a lower peak than when the jitter was small (Figure 6.7B).

For both SOA jitter and latency jitter, this smearing can be understood in terms of convolution, which is mathematically equivalent to filtering out the high frequencies in the data. I will provide an intuitive explanation of how convolution is equivalent to filtering in the next section, and I will provide a more formal explanation in Online Chapter 12. But first I need to explain how convolution is involved in overlap.

Figure 11.5 shows how the overlap from the previous stimulus can be reconstructed by means of a convolution. Panel A shows the original (overlap-free) waveform. Panel B shows the probability distribution of the SOA, with a 25-ms bin width. This shows that 15% of the SOAs were between 475 and 500 ms, 10% were between 450 and 475 ms, etc. Note that the probabilities are not exactly the same for each bin; this is what will typically happen if you try to create a random set of SOAs with a finite number of trials.

Panel C of Figure 11.5 shows how we can compute the average overlap by scaling and shifting the original ERP waveform in accord with the probability distribution of SOAs. Each scaled-and-shifted waveform represents the subset of trials that occurred at a particular SOA and therefore had an ERP that started at that amount of time prior to the current trial. For example, the contribution of the 15% of trials with an SOA of 475-500 ms is represented by shifting the original waveform leftward so that it starts at the center of this time bin (at 487.5 ms), and multiplying it by 0.15 to reflect the proportion of trials that had this SOA. Similarly, the contribution of the 12.5% of trials with an SOA of 300-325 ms is represented by shifting the original waveform

leftward so that it starts at the center of this time bin (at 312.5 ms), and multiplying it by 0.125. This has been done for every time bin.

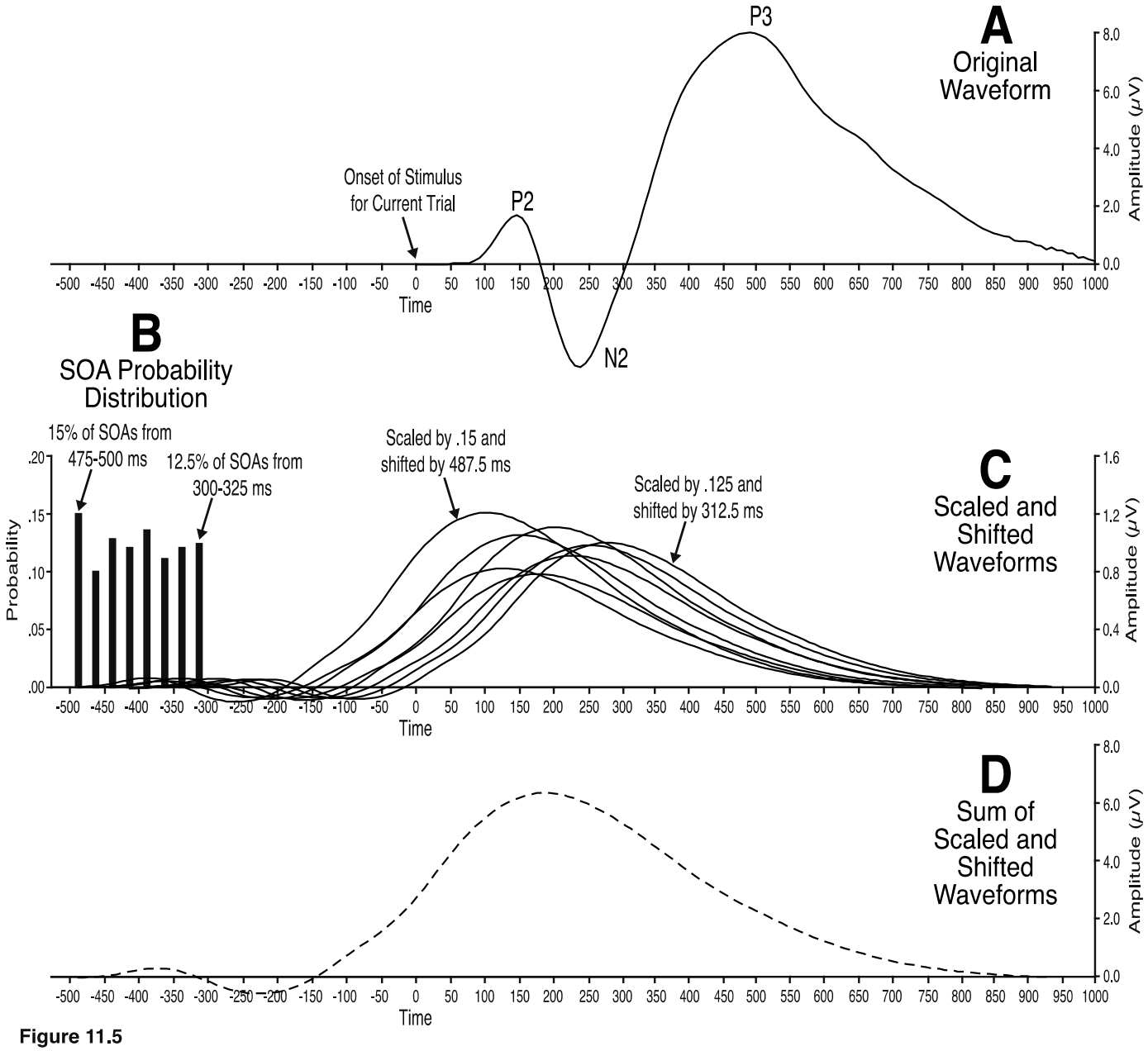

Figure 11.5

Relationship between overlap and convolution. (A) Hypothetical original waveform without overlap. (B) Probability distribution of the stimulus onset asynchrony (SOA). (C) Contribution of each SOA bin to the overlap. For each SOA bin, the original waveform is scaled by the height of the bin and shifted so that it begins at the time of the bin. Note that the amplitude scale is different from that in (A). (D) Sum of the scaled and shifted waveforms from (C). The combination of (C) and (D) is equivalent to convolving the SOA distribution with the original, overlap-free ERP waveform.

Recall that you can average a bunch of waveforms by summing all of them and then dividing by the number of trials, or by scaling (multiplying) each waveform by its probability of occurrence and then summing the scaled waveforms (cf. panels B and C of Figure 11.2). Here we have scaled the waveforms first, so now all we need to do to get the average is to add them together. Figure 11.5D shows the sum of the scaled-and-shifted waveforms. This is the overlap that we would expect given this distribution of SOAs.

To summarize, our first step was to replace each bar in the SOA probability distribution with an ERP waveform that was shifted so that it began at the center of the bar and was scaled by the height of the bar. Our

second step was to sum together these scaled-and-shifted waveforms. These two steps are the same thing as convolving the original ERP waveform with the SOA probability distribution (see the definition of convolution that I provided earlier in the chapter). As you will see in a minute, this gives us some interesting information about the nature of the overlap. First, however, we need to consider two assumptions that we've made.

The first assumption is that we can represent all of the trials that begin within a given time bin (e.g., 475-500 ms) as if they start at the middle of the time bin (e.g., 487.5 ms). This is a reasonable approximation, and we could be more precise simply by using very narrow time bins (e.g., 2 ms bins; see Box 8.2). The second assumption is that the overlapping ERP from the previous trial has the same shape no matter whether the SOA is short or long. This will generally be true as long as the SOA is not extremely short. The reason is that the current event cannot influence brain activity the occurs prior to the current event (because events at one time cannot influence events at previous times). If the SOA is short enough, then some of the overlap will happen after the brain has already realized that a new stimulus has been presented, and this could influence the overlap. In practice, however, this has relatively little effect in most cases. Thus, this is a very reasonable way to reconstruct the overlap.

The only problem is that we don't ordinarily know what the original, overlap-free waveform looks like, so we can't convolve it with the SOA distribution to compute the overlap. However, Marty Woldorff figured out a very clever way to do this called the ADJAR (adjacent response) filter (Woldorff, 1993). This technique essentially deconvolves the overlap-contaminated ERP and the distribution of SOAs to recover the original, overlap-free waveform. It's a fairly complicated technique, but it has been used effectively in several studies to eliminate overlap (e.g., Hopfinger & Mangun, 1998; Luck, Hillyard, Mouloua, Woldorff, Clark, & Hawkins, 1994; Woldorff & Hillyard, 1991).

The basic idea behind the ADJAR filter is that the overlap-contaminated ERP that we actually record can be used as an initial estimate of the overlap-free waveform (especially if we first do some tricks to reduce the overlap, like filtering out the low frequencies in the data). We then convolve this waveform with the distribution of SOAs to estimate the overlap. It's not a perfect estimate, because we are convolving a waveform that itself is contaminated by overlap. However, it gives us an initial estimate of the overlap. If we subtract this estimate of the overlap from the original ERP, we get a better estimate of what the overlap-free ERP should look like. It's not perfect, but it's better than the originally recorded waveform. We can then use this as a new estimate of the overlap-free waveform. This is then convolved with the distribution of SOAs to obtain a better estimate of the overlap. This better estimate is then subtracted from the original data to obtain an even better estimate of the overlap-free waveform. This process is iterated multiple times, and eventually it converges on a very good estimate of the overlap-free waveform. There are, of course, some complications, but in practice it can work very well, at least in experiments that start with a relatively broad range of SOAs.

An Intuitive Link Between Convolution and Filtering

As I mentioned before, convolution is mathematically equivalent to filtering. Online Chapter 12 will go into the details of the math, but here I'd like to give you an intuitive understanding that expands on the presentation provided by Woldorff (1993). After you read this section, you will understand why using a jittered SOA is equivalent to applying a low-pass filter to your data (i.e., filtering out the high frequencies).

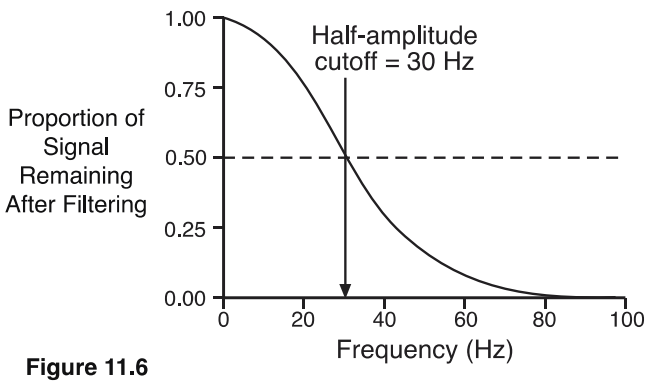

Figure 11.6
Frequency response function for a typical low-pass filter.

First, I want to remind you a little bit about how filters work. A conventional low-pass filter reduces the amplitudes of the highest frequencies by a lot, so that only a small proportion of the high frequencies in the original data are present in the filter's output. Low frequencies pass through the filter nearly perfectly, and intermediate frequencies are suppressed by an intermediate amount. As described in Chapter 7, this suppression is typically

quantified with a *frequency response function*. An example of the frequency response function for a typical low-pass filter is shown in Figure 11.6. At each frequency, this function tells you would proportion of the signal passes through the filter. This function is often summarized by a single number, the *half-amplitude cutoff frequency*. This number represents the frequency at which 50% of the signal passes through the filter and 50% is suppressed.

Imagine that you had a filter but didn't know its frequency response function or cutoff frequency. You could determine these empirically by taking sine waves of various frequencies, applying the filter to them, and seeing how much smaller they were in the filter's output than they were in the filter's input. If you plotted the amplitude of the filter's output for each frequency, this would give you the frequency response function of that filter (just as in Figure 11.6). I will use an analogous approach to show you how a jittered SOA serves as a low-pass filter.

The basic idea is illustrated in Panels A, B, and C of Figure 11.7, which show the effects of a jittered SOA on sine waves of three different frequencies. This allows us to see how much a given filter is attenuated by the jitter. If the waveform from the previous trial consists of a high frequency sine wave, as shown in Panel A, the jittering causes complete cancellation (because the uppie that happens when the sine wave is shifted so that it starts at one bar in the SOA distribution will be at the same time as the downie that happens when the sine wave is shifted so that it starts at a different bar in the SOA distribution). Consequently this frequency will be completely eliminated when we average the data with this jitter. If we have an intermediate frequency, as shown in Panel B, we get a moderate amount of cancellation, and this frequency will be reduced by a moderate amount by the jitter. If we have a low frequency, as shown in Panel C, there will be very little cancellation, and the signal is affected only a small amount by the jitter.

This is summarized by Panel E, which shows the percentage of the sine wave that will remain after averaging the jittered waveforms for each sine wave frequency (i.e., the frequency response function resulting from jittering the SOA). The lowest frequencies are not attenuated at all, and the amount of attenuation increases as the frequency of the sine wave increases. This is the frequency response function produced by the jittered SOA[iv]. It is exactly identical to what you would get by applying a very simple "smoothing" filter to the overlap. Note, however, that only the overlap is filtered by the jitter; the ERP elicited by the current trial is not filtered by the jitter. Thus, jitter selectively attenuates the high frequencies in the overlap.

Panel D of Figure 11.7 shows how a wider jitter range would influence the intermediate sine wave frequency from Panel B. With the wider jitter range, we now get complete cancellation of this intermediate frequency. The overall effect of this wider jitter range can be seen in the frequency response function in Panel E, which shows that we are now suppressing both high and intermediate frequencies, but still allowing the lowest frequencies to remain. In general, the wider the jitter range is, the lower the cutoff frequency of the filter is.

Figure 11.7 doesn't include any units on the X and Y axes, because I don't want you to worry about the mathematical details yet. These will be provided in Online Chapter 12, where you will learn how to calculate the frequency response function from the width of the SOA jitter range. This will be useful, because it allows you to use a simple trick to minimize the effects of overlap on your ERPs. Specifically, you can eliminate overlap by combining a jittered SOA (which filters out the high frequencies in the overlap) with a conventional high-pass filter (which removes the low frequencies from both the overlap and the ERP from the current trial). For example, imagine that you use an SOA jitter range that eliminates everything above 1 Hz from the overlap, and then you use a high-pass filter that eliminates everything below 1 Hz from your data. You would end up with no overlap remaining! It is difficult to use this approach to completely eliminate overlap in real experiments, because it requires an extremely wide SOA jitter range and because a high-pass cutoff at 1 Hz may

[iv] I've simplified this frequency response function slightly. The true frequency response function for this kind of filter is described in Online Chapter 12.

distort your data. However, you can use the combination of a modest jitter range and a modest high-pass filter to remove a great deal of the overlap from your data.

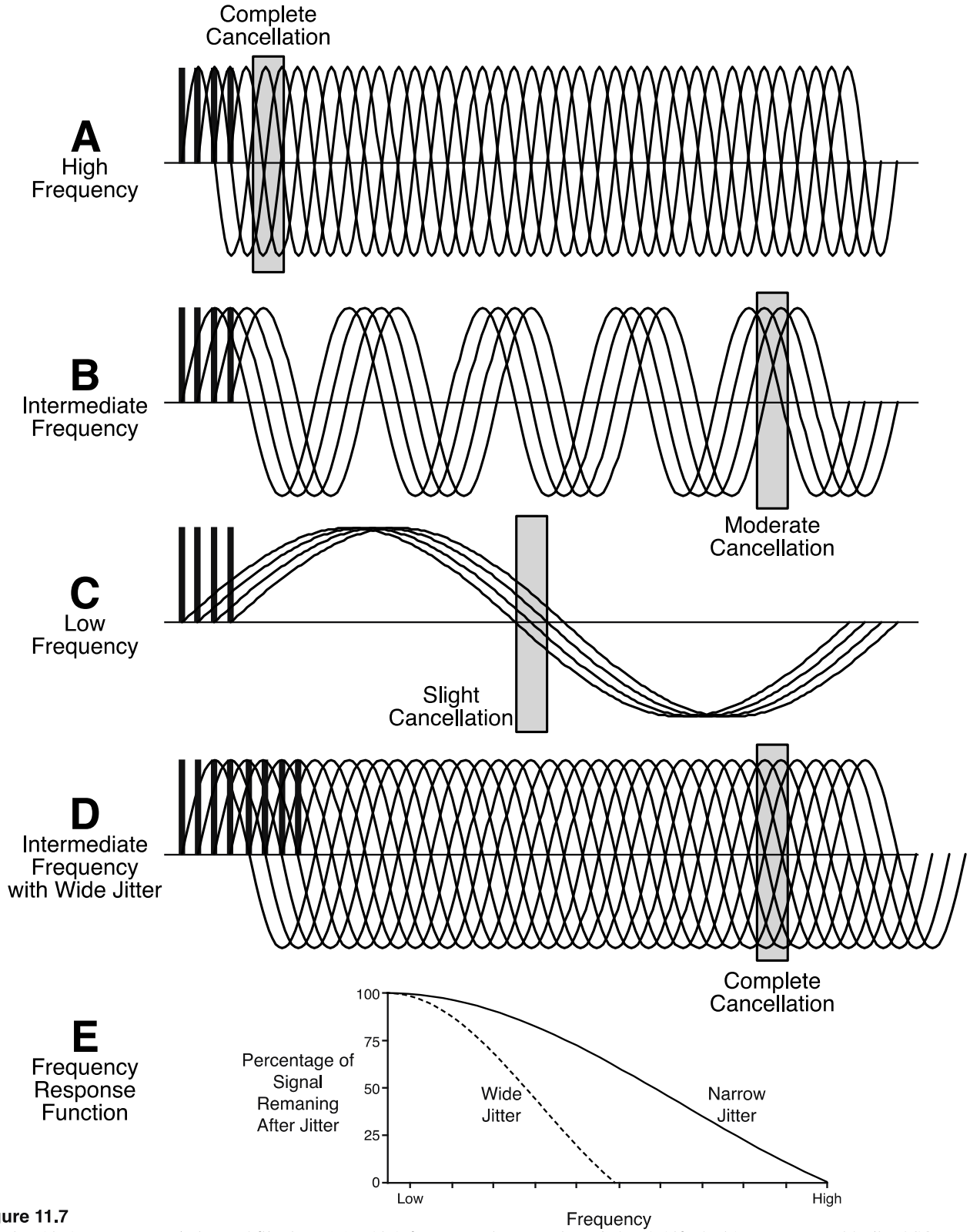

Figure 11.7
Relationship between convolution and filtering. (A) A high frequency sine wave is scaled and shifted with respect to an idealized SOA distribution. When the scaled-and-shifted sine waves are summed together to finalize the convolution process, the positive and negative portions of the waveforms cancel out completely. Thus, this frequency is completely suppressed by convolution with this distribution of SOAs. (B) Same as (A), but for an intermediate frequency. This frequency is only partially cancelled by the convolution, and the result would be a lower-amplitude sine wave of the same frequency. (C) Same as (B), but for an even lower frequency. Even less cancellation occurs, and the result would be a slightly smaller sine wave of this frequency. (D) Same intermediate-frequency sine wave as in (B), but convolved with a wider SOA jitter. Now the sine wave is completely cancelled by the convolution. (E) Frequency response functions for the narrow jitter in (A), (B), and (C) and for the wide jitter in (D). Note that these are simplifications of the actual frequency response functions (see Online Chapter 12 for the true frequency response functions).

Another Intuitive Way of Linking Convolution and Filtering

The previous section explained how convolution is the same as filtering by starting with the somewhat unusual example of overlap and jittered SOAs. In this section, I will explain this link in the other direction, taking an obvious kind of filtering and showing that it is equivalent to convolution. This will be an important step in helping you understand how convolution is related to the kinds of filters that are ordinarily used in ERP research and how time-frequency analysis is implemented by convolving a wavelet with the EEG prior to averaging.

First, however, I need to explain a small fact about convolution. In mathematics, the * symbol is used to denote convolution. That is, "A convolved with B" is written as "A * B" (which is a little confusing, because the * symbol is often used in computers to denote multiplication). Convolution obeys the "commutative property," which just means that the order of convolution doesn't matter: A * B = B * A. For example, to determine the effect of latency jitter on the averaged ERP waveform, we could replace each point in the ERP waveform with a scaled and shifted version of the SOA distribution and then sum them (rather than replacing each point in the SOA distribution with a scaled and shifted version of the ERP waveform and then summing them). This fact leads to another way of thinking about convolution and filtering.

Running Average Filters

Let's start by thinking about a really easy way that you could filter out noise in an ERP waveform, such as the high-frequency noise in Figure 11.8A (i.e., the quick uppies and downies that are superimposed on the slowly changing ERP waveform). The goal would be to smooth out the waveform, so that the fast changes are eliminated but the slower changes in the waveform remain (like the waveform in Figure 11.8B). This was already described in Chapter 7, but I will provide a reminder here.

Figure 11.8D shows a magnified view of the individual samples in a small segment of the ERP waveform (the region indicated by the circle in Figure 11.8A). Notice the rapid up-down-up sequence in the 3 samples indicated by white-filled circles in the middle of this segment. We could smooth out this rapid up-down-up sequence by replacing the middle sample with the average of the three white-filled samples. We could then do this for all the other samples, replacing each sample in the original waveform with the average of that sample and the immediately surrounding samples. This gives us a *3-point running average* of the data. The result of this running average is shown by the gray-filled circles in Figure 11.8D. Notice that the changes from sample to sample in the gray-filled circles are less extreme than the changes in the unfiltered data, giving us a smoother waveform. This can be seen even more clearly in Figure 11.8B, which shows what happens when we apply this 3-point running average to the whole waveform from Figure 11A. The result is much smoother, lacking the rapid uppies and downies while still containing the slower changes in the waveform. In other words, we have filtered out the high frequencies, which is equivalent to applying a low-pass filter to the data.

Figure 11.8C shows what happens if we use a 7-point running average instead of a 3-point running average. That is, each value in the original unfiltered waveform is replaced with the average of that point and the 3 points on either side. The result is an even smoother waveform. As you increase the number of points in a running average, the cutoff frequency of the filter decreases.

Using a Weighting Function

If we want to average the current point with the point on either side to create a 3-point running average filter, we can add the three values together and then divide by 3. Alternatively, we can multiply each value by 1/3 and then add them together. More generally, with an N-point running average filter, we can multiply each point by 1/N and then add them together. Because each point is multiplied by the same value (1/N), each point has the same *weight* in determining what the filtered waveform looks like.

If you use, for example, a 51-point running average, you are averaging the 25 points on either side of the current point to get the filtered value for the current point. Each of the 25 time points on each side contributes

equally to the filtered value at the current time point. This is a little weird, because the distant time points (e.g., 20-25 points away from the current point) have the same influence as the nearby time points. To solve this problem, you could use a weighted average instead of a conventional average. For a 3-point running average filter, for example, you could average the 3 points by multiplying the current point by .5 and each of the adjacent points by .25, and then summing these three values together. That would give the current point in the unfiltered data greater weight than the surround points in determining the filtered value.

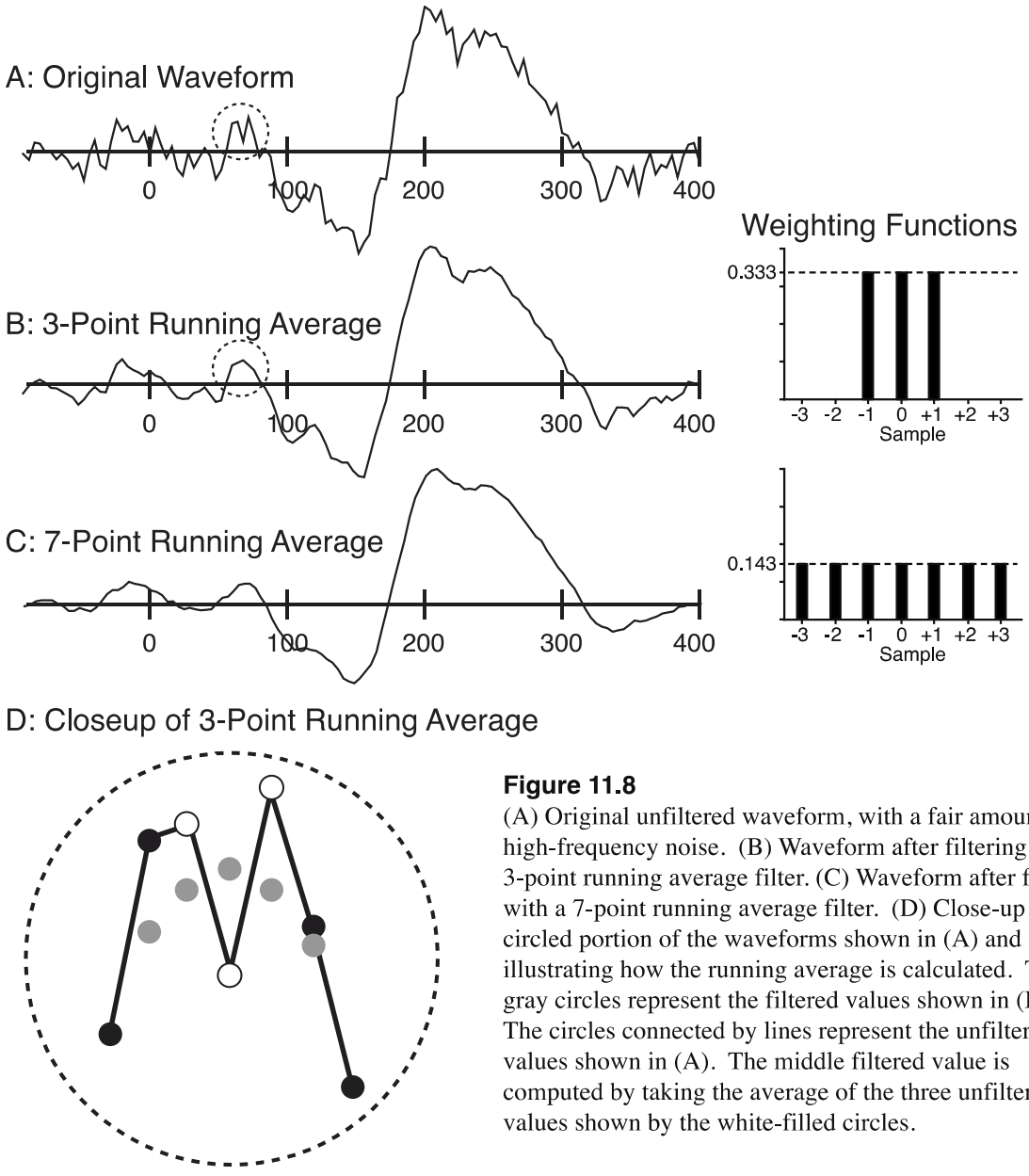

Figure 11.8
(A) Original unfiltered waveform, with a fair amount of high-frequency noise. (B) Waveform after filtering with a 3-point running average filter. (C) Waveform after filtering with a 7-point running average filter. (D) Close-up of the circled portion of the waveforms shown in (A) and (B), illustrating how the running average is calculated. The gray circles represent the filtered values shown in (B). The circles connected by lines represent the unfiltered values shown in (A). The middle filtered value is computed by taking the average of the three unfiltered values shown by the white-filled circles.

To describe the weighting values used for each point, we can define a *weighting function*. This function just indicates the value that will be multiplied by each of the surrounding points when computing the running average. Online Chapter 12 will discuss various different weighting functions that we might use. For now, we will stick with simple running average filters, in which each point has an equivalent weight, but I wanted you to understand why we have something called a *weighting function*.

The weighting functions for our 3-point and 7-point running average filters are shown in Figure 11.8. The value at each point is simply 1/N, where N is the number of points in the running average. For example, the

weighting function for the 3-point running average filter has a value of 1/3 at the current point, 1/3 at the point preceding the current point, and 1/3 at the point following the current point.

Using Convolution to Implement a Running Average Filter

Now we are ready to get back to convolution. It turns out that a running average filter is mathematically equivalent to convolving the ERP waveform with the weighting function used to compute the running average. Specifically, if you convolve the weighting function with the unfiltered ERP waveform, you will get exactly the same result that you would get by performing a running average with that weighting function. In other words, you can filter the ERP waveform by replacing each point in the original unfiltered ERP waveform with a scaled and shifted version of the weighting function and then summing these scaled-and-shifted weighting functions. This gives you exactly the same filtered waveform that you would get by replacing each point in the unfiltered waveform with the average of the current point and the surrounding points.

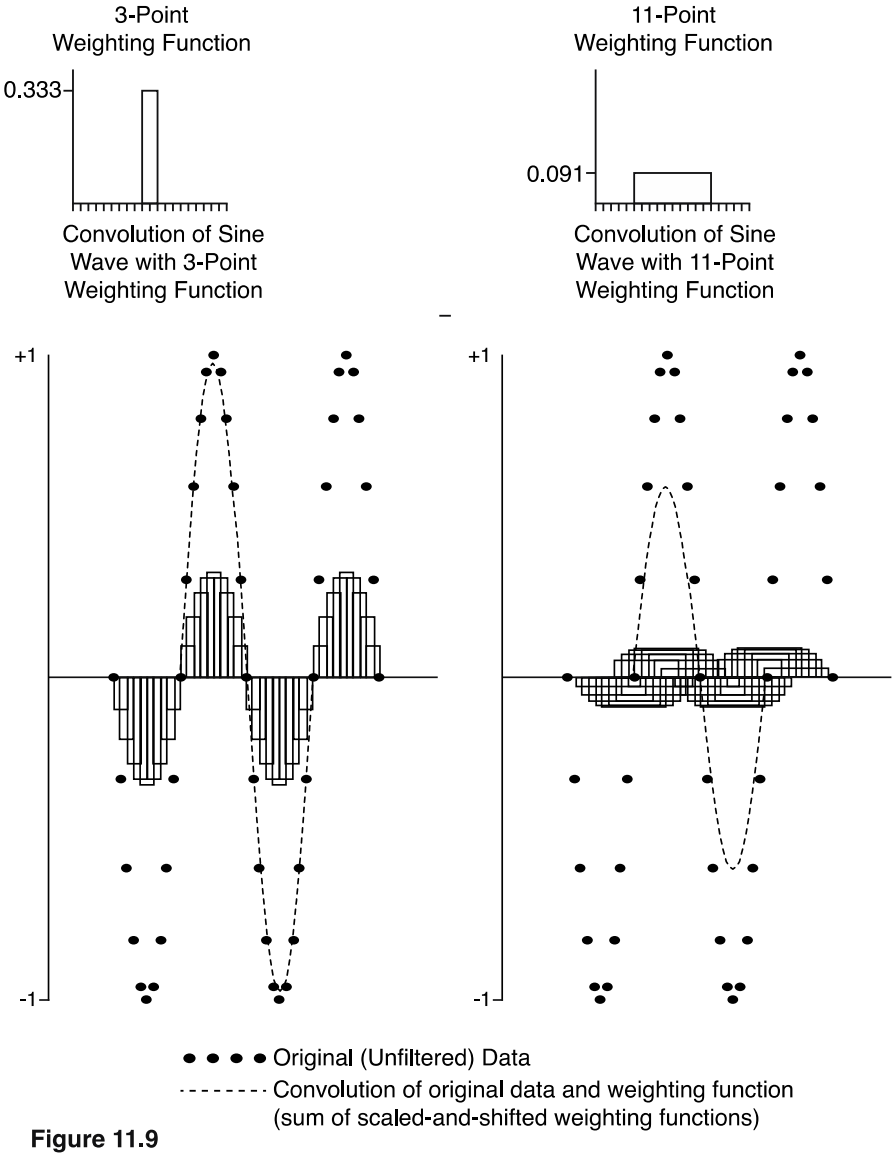

Figure 11.9
Convolution of a sampled sine wave with a 3-point weighting function (left) and an 11-point weighting function (right). Each point in the sine wave is replaced with a copy of the weighting function, scaled to reflect the height of the sine wave at that point and shifted to be centered at that point. These scaled-and-shifted weighting functions are then summed together (dashed line), giving us the convolution of the weighting function and the sine wave. Note that the sum is shown only for the middle set of points; things get complicated near the ends of the data (see Chapter Online Chapter 12 for details).

This is illustrated in Figure 11.9, which shows the convolution using a 3-point weighting function and an 11-point weighting function (which is the same as doing 3-point and 11-point running averages). Instead of a real ERP waveform, we are filtering a sine wave of an intermediate frequency that has been sampled at discrete time points, just as an ERP waveform is sampled. The rectangles in the figure show the 3-point and 11-point weighting functions, shifted and scaled for each time point. Each point in the sine wave is replaced with a scaled-and-shifted version of the weighting function. Note that the sine wave goes from values of -1 to +1, whereas each value in the 3-point weighting function is 1/3 and each value in the 11-point weighting function is 1/11. When a value of .33 in the weighting function is scaled by a value of 1 in the sine wave, the result is .33. Thus, each scaled-and shifted weighting function is much smaller than the sine wave. However, these scaled-and-shifted weighting functions are then

summed (indicated by the dashed lines), making the result more comparable to the original sine wave[v]. For example, the sum of the scaled-and-shifted values for the 3-point weighting function is almost identical to the original sine wave. In other words, this filter does not produce much attenuation of a sine wave of this frequency (although it would attenuate a higher-frequency sine wave). The sum of the scaled-and-shifted values for 11-point weighting function, in contrast, is reduced to 57% of the amplitude of the original sine wave. In other words, convolution with an 11-point weighting function passes only 57% of this frequency. The 11-point weighting function would cause less attenuation of a lower-frequency sine wave, and more attenuation of a higher-frequency sine wave. Online Chapter 12 will show you what the frequency response function looks like for this kind of filter.

I hope you now feel that you have an intuitive understanding of how convolution is related to filtering. And without a single equation! Online Chapter 12 provides a more detailed discussion, including equations, but the equations involve nothing more than addition, subtraction, multiplication, and division. And now you should be prepared for this little bit of math.

[v] Note that the filtered values become undefined at the left and right edges of the original sine wave, so the sum is shown only for the central set of points. I will say more about this complicating factor in Online Chapter 12.

References

Hansen, J. C. (1983). Separation of overlapping waveforms having known temporal distributions. Journal of Neuroscience Methods, 9, 127-139.

Woldorff, M., & Hillyard, S. A. (1991). Modulation of early auditory processing during selective listening to rapidly presented tones. Electroencephalography and Clinical Neurophysiology, 79, 170-191.

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

Luck, S. J., Hillyard, S. A., Mouloua, M., Woldorff, M. G., Clark, V. P., & Hawkins, H. L. (1994). Effects of spatial cuing on luminance detectability: Psychophysical and electrophysiological evidence for early selection. Journal of Experimental Psychology: Human Perception and Performance, 20, 887-904.

Hopfinger, J. B., & Mangun, G. R. (1998). Reflective attention modulates processing of visual stimuli in human extrastriate cortex. Psychological Science, 9, 441-447.

Urbach, T. P., & Kutas, M. (2006). Interpreting event-related brain potential (ERP) distributions: implications of baseline potentials and variability with application to amplitude normalization by vector scaling. Biological Psychology, 72, 333-343.

Luck, S. J., Kappenman, E. S., Fuller, R. L., Robinson, B., Summerfelt, A., & Gold, J. M. (2009). Impaired response selection in schizophrenia: Evidence from the P3 wave and the lateralized readiness potential. Psychophysiology, 46, 776-786.

Hansen, J. C. (1983). Separation of overlapping waveforms having known temporal distributions. Journal of Neuroscience Methods, 9, 127-139.

Hopfinger, J.B., & Mangun, G.R. (1998). Reflective attention modulates processing of visual stimuli in human extrastriate cortex. Psychological Science, 9, 441-447.

Luck, S.J., Hillyard, S.A., Mouloua, M., Woldorff, M.G., Clark, V.P., & Hawkins, H.L. (1994). Effects of spatial cuing on luminance detectability: Psychophysical and electrophysiological evidence for early selection. Journal of Experimental Psychology: Human Perception and Performance, 20, 887-904.

Luck, S.J., Kappenman, E.S., Fuller, R.L., Robinson, B., Summerfelt, A., & Gold, J.M. (2009). Impaired response selection in schizophrenia: Evidence from the P3 wave and the lateralized readiness potential. Psychophysiology, 46, 776-786.

Urbach, T. P., & Kutas, M. (2006). Interpreting event-related brain potential (ERP) distributions: implications of baseline potentials and variability with application to amplitude normalization by vector scaling. Biological Psycholology, 72, 333-343. doi: 10.1016/j.biopsycho.2005.11.012

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

Woldorff, M., & Hillyard, S.A. (1991). Modulation of early auditory processing during selective listening to rapidly presented tones. Electroencephalography and clinical neurophysiology, 79, 170-191.