Notes on Analyzing Neural Time Series Data Theory and Practice

EEG reflects mainly the summation of excitatory and inhibitory postsynaptic potentials at the dendrites of ensembles of neurons with parallel geometric orientation.

EEG cannot measure individual molecular or synaptic events,

If you would like to measure very slow activity, check whether your amplifier is well suited for this and check whether there are high-pass filters that are applied during the recording.

Very fast fluctuations (> 100 Hz) are also difficult but not impossible to measure. High-frequency activity generally has low power and thus is more difficult to distinguish from noise.

For EEG, activity above around 80 Hz should be carefully inspected to make sure it is not driven by noise spikes, EMG, eye movements, or other artifacts.

In the brain, the term oscillation refers to rhythmic fluctuations in the excitability of neurons or populations of neurons.

Ozkurt , T. E. , and A. Schnitzler . 2011 . A Critical Note on the Definition of Phase-Amplitude Cross-Frequency Coupling. Journal of Neuroscience Methods 201 (2): 438 – 443 .

Phase-locked activity (also sometimes called " evoked ") is phase-aligned with the time = 0 event and will therefore be observed both in time-domain averaging (the ERP) and in time-frequency-domain averaging.

Phase-locked and non-phase locked ??

There are recent simulations that suggest mechanisms of ERPs through complex additive and nonlinear effects (David, Kilner, and Friston 2006)

There are many software packages that deliver stimuli to the subject and event markers to the EEG acquisition system. Some software programs are commercial (e.g., Presentation or E-Prime), and others are toolboxes for Matlab (e.g., Psychophysics toolbox or Cogent). Make sure the software you use can integrate with the computer hardware to maximize timing precision and stimulus delivery.

There is no magic number of trials that will guarantee good results or sufficient signal-to-noise ratios.

If all you will do with the data is measure the P3 amplitude, you technically need only three electrodes (one placed over central parietal cortex to record the P3, one for a reference, and one for a ground)

There are also gel-free EEG caps that use sponges that are soaked in a salt-water solution; these sponge-based caps can decrease preparation time considerably.

Technically, you need to sample at least twice the highest frequency of interest.

In practice, sampling rates between 500 Hz and 2000 Hz are likely to be sufficient for all analyses.
This is better than recording the data at 128 Hz and then having a poor signal-to-noise ratio for power and phase estimates in the beta and gamma frequency ranges.

But given the choice between sampling rates of 1024 Hz and 1000 Hz, I would choose
the latter.

Eye trackers can also be used to measure changes in pupil dilation, which can provide insight into cognitive processes

If you plan on performing source localization analyses, and precise localization is important for the conclusions you hope to make from the data, using electrode localization equipment will be beneficial.
Standard computer keyboards are poor choices for response devices. Some keyboards that are specially designed for gaming have millisecond-precision responses

If you have epochs with overlapping data, and if you plan on using independent components analysis to analyze data (that is, analyzing the component time courses instead of the electrode time courses), be sure not to expose the independent components analysis to the same data more than once.

If you are reanalyzing a dataset that has already been epoched and cannot be reepoched from the continuous data, and if you are concerned that the epochs are too short and the time-frequency results might be contaminated by edge artifacts, you can use a " reflection " approach, whereby the EEG data from each trial and electrode are reversed and put in the beginning and end of the trial. Data reflection procedure. Data are reversed in time, concatenated to both ends of the real-data time series, analyses are performed, and then the reflected data are trimmed. This procedure attenuates edge artifacts and can be useful if the epochs are cut too short for planned analyses.

Notch filters at 50 Hz or 60 Hz help attenuate electrical line noise.

Most time-frequency decomposition methods (wavelet convolution, FFT, filter-Hilbert) involve applying a set of temporal filters to the data.

For example, there is no need to low-pass filter the time-domain data at 40 Hz if you will then perform time-frequency analyses to extract power from 2 to 20 Hz.

Applying a high-pass filter at 0.1 or 0.5 Hz to the continuous data is useful and recommended to minimize slow drifts.

High-pass filters should be applied only to continuous data and not to epoched data.

The problem with interpolated electrodes is that they do not provide unique data; they are a perfect weighted sum of the activity of other electrodes. This reduces the rank of the data matrix, which may lead to problems in analyses that require the matrix inverse (taking the pseudoinverse is usually an appropriate solution).

Removing Data Based on *Independent Components Analysis*
In this case the result of an *independent components analysis* is a set of weights for each microphone such that the weighted sum of all microphones best isolates the voice from one person.
In the context of EEG the independent components analysis provides a set of weights for all electrodes such that each component is a weighted sum of activity at all electrodes, and the weights are designed to isolate sources of brain electrical signals

The maximum number of components that can be isolated in the EEG data is the number of electrodes you have.

There are several methods that successfully attenuate the oculomotor artifacts while sparing brain activity. The two most commonly used methods are independent components analysis (Jung et al. 2000) and regression-based techniques (Gratton, Coles, and Donchin 1983).

Having an easy-to-see fixation spot on the monitor at all times will also help subjects prevent eye movements. Finally, instructing subjects that small eye movements cause artifacts in the data will help minimize the frequency of saccades.

Saccade: discretre conjugate eye movements from one object to another. Microsaccades can be minimized by having small stimuli such that subjects do not need to saccade to see the entire stimulus, and by presenting the stimulus on the screen for only a short period of time (this may not be feasible in all experiments). There are algorithmic approaches for detecting and removing microsaccade artifacts from EEG data (Hassler, Barreto, and Gruber 2011; Keren, Yuval-Greenberg, and Deouell 2010; Nottage 2010).
If you use a nose reference, eye movements might influence the data more than if you use an earlobe reference.

The *surface Laplacian*, for example, will help prevent the spread of oculomotor artifacts to activity at other electrodes. study on Surface Laplacian (a spatial filter)

EMG is noticeable as bursts of 20- to 40-Hz activity, often has relatively large amplitude, and is typically maximal in electrodes around the face, neck, and ears. EMG bursts are deleterious for EEG
data if you plan on analyzing activity above 15 Hz.

Sometimes the EMG activity is well localized to lateral temporal or anterior electrodes; if you plan on testing beta-band activity over motor electrodes that show little EMG artifact, you may be willing to keep that dataset in the group analyses. (beta 12-30 Hz)
Baseline normalization of time-frequency power is based on relative changes in power before versus after trial onset. This means that if the EMG activity has constant amplitude before and after trial onset, that EMG activity will be removed during baseline normalization

These artifacts can be called cognitive noise or cognitive artifacts , and you should consider removing these trials from your analyses. (engaging in task problem of your subjects)

Keep in mind that time-domain signal averaging over trials is itself a low-pass filter.

Figure 9.2 For ERPs

The wide-band 0 – 40 Hz filter had the least effect on the larger ERP fluctuations while removing the high-frequency fluctuations. This plot is an illustration of why you should carefully consider the frequency range of the filter used for interpreting ERPs, particularly if you use a narrow frequency range.

Butterfly plot (üst üste bir çok plot, renkli)

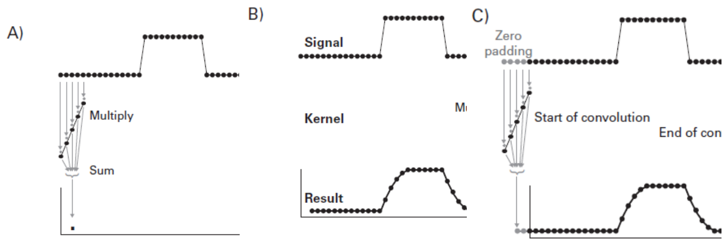
Many techniques for extracting frequency information from a time-domain signal rely on a mathematical procedure called *convolution* .

Dot product

cumulative interpretation statistical interpretation, geometric interpretation. The dot product may seem like a very simple and trivial expression, but do not **underestimate** its utility: the dot product is the basic building block of many data analysis techniques, including convolution and the Fourier transform.

Time Domain Convolution

The formula for how long the result of the convolution will be is length(signal) + length(kernel) – 1. Thus, after you run convolution, you need to trim the result by removing one-half of the length of the kernel from the beginning and one-half of the length of the kernel plus one from the end. This leaves you with a result of convolution equal to the length of the signal.



Convolution formula,

$$(a * b)_k = \sum_{i=1}^{n} a_i b_{k-i}$$

where a is orijinal dignal and b is kernel.

In EEG data analyses, convolution is used to isolate frequency-band-specific activity and to localize that frequency-band-specific activity in time. This is done by convolving wavelets — time-limited sine waves — with EEG data. **As the wavelet (the convolution kernel) is dragged along the EEG data (the convolution signal), it reveals when and to what extent the EEG data contain features that look like the wavelet. When convolution is repeated on the same EEG data using wavelets of different frequencies, a time-frequency representation can be formed.**

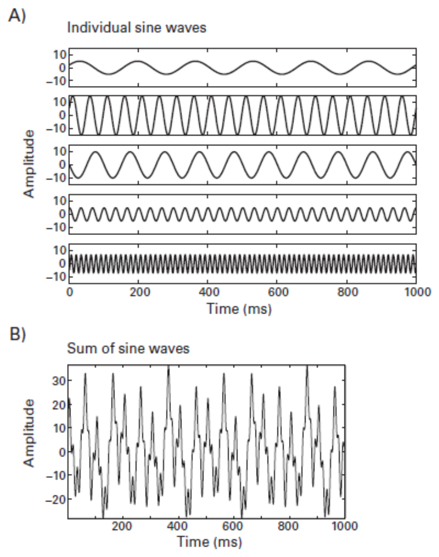
Excercises
1. Create two kernels for convolution: one that looks like an inverted U and one that looks like a decay function. There is no need to be too sophisticated in generating, for example, a Gaussian and an exponential; numerical approximations are fine.

2. Convolve these two kernels with 50 time points of EEG data from one electrode. Make a plot showing the kernels, the EEG data, and the result of the convolution between the data and each kernel. Use time-domain convolution as explained in this chapter and as illustrated in the online Matlab code. Based on visual inspection, what is the effect of convolving the EEG data with these two kernels?

The result of the Fourier transform is a three-dimensional (3-D) representation of the time series data in which the three dimensions are frequency, power, and phase.

To perform a Fourier transform, you need to know how to compute the dot product, and you need to know how to create a sine wave.

Asin(2π ft +θ)



in which A is the amplitude of the sine wave, π is pi, or 3.141 . . . , f is the frequency of the sine wave, t is time (this could also be space, but only time-varying sine waves are discussed here), and θ is the phase angle offset, which is related to the value of the sine wave at time = 0.

The better you understand how the Fourier transform works, the better you will understand the mechanics of the more advanced analyses.

Key point: The dot product computes the mapping between two vectors, which can be thought of as a way to measure the similarity between those two vectors.

Descrete Fourier Transform

The idea behind the discrete time Fourier transform is fairly simple: create a sine wave and compute the dot product between that sine wave and the time series data. Then create another sine wave with a different frequency and compute the dot product between that sine wave and the same time series data. The number of sine waves you create, and the frequency of each sine wave, is determined by the number of data points in the time series data (more on this in a few paragraphs). The following Matlab code will perform a discrete time Fourier transform of random data.
(Burada convolution yok dot product var. Kernel ile signal ın boyutu eşit çünkü)

```
N = 10; % length of sequence
data = rand(1,N); % random numbers  %kernel
% initialize Fourier coefficients
fourier = zeros(size(data));
time  =  (0:N-1)/N;  %  time  starts  at  0;  dividing
by N normalizes to 1
% Fourier transform
for fi=1:N
% create sine wave
sine_wave = exp(-1i*2*pi*(fi-1).*time);
% compute dot product between sine wave and data
fourier(fi) = sum(sine_wave.*data);
end
```

$$X_f = \sum_{k=1}^{n} x_k e^{-i2\pi f(k-1)n^{-1}}$$

To convert the frequencies to hertz, consider that the number of unique frequencies that can be extracted from a time series is exactly one-half of the number of data points in the time series, plus the zero frequency. This is due to the Nyquist theorem: you need at least two points per cycle to measure a sine wave, and thus half the number of points in the data corresponds to the fastest frequency that can be measured in those data.

The inverse Fourier Transform

The inverse Fourier transform reverses this procedure. In fact, figure 11.2 was conceptually similar to an inverse Fourier transform: you started with sine waves of different frequencies, amplitudes, and phases, and you summed them together to form a single time series. Thus, **to compute the inverse Fourier transform, you build sine waves of specific frequencies, multiply them by the respective Fourier coefficients at those frequencies, sum all of these sine waves together, and then divide by the number of sine waves (which is also the number of points in the time series).**

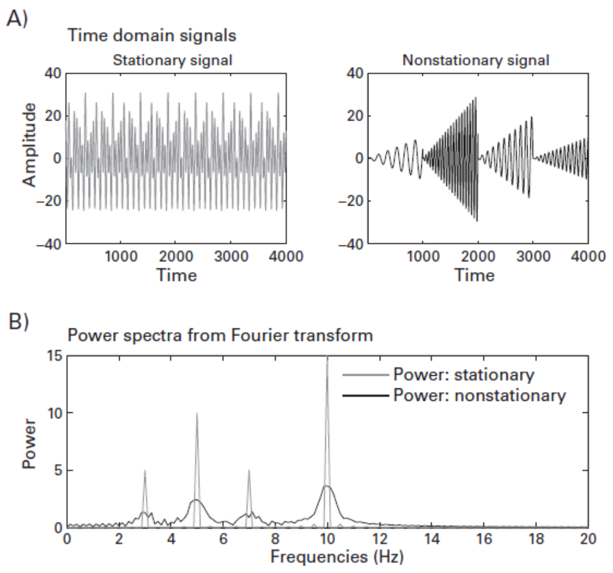
$$x_k = \sum_{k=1}^{n} X_k e^{i2\pi f(k-1)n^{-1}}$$

Fast fourier transform is ten times faster than descrete fourier transform.
Technically, the speed of a discrete time Fourier transform is related to the number of computations as N 2 , where N is the number of data points; the speed of the FFT is related to the number of computations as N log N .

This is clearly not the case for EEG data: the frequency structure of neurophysiological activity changes over time both because of task events and because of endogenous processes.

Statinonary and Non-Stationary Signals

Stationary means that the statistics of the data, including the mean, variance, and frequency structure, do not change over time (that is, the time series is “ well behaved ”).



Two time series were created from the same sine waves of frequencies 3, 5, 7, and 10 Hz (A).

One signal was stationary, and the other was nonstationary — variations were introduced by having the frequency structure change over time and also by having the amplitude and therefore also the variance increase over time within each sine wave. As seen in figure 11.9B , the power spectrum clearly shows the spectral peaks for both the stationary and nonstationary time series. However, the spectral peaks for the nonstationary data are less well defined, and there seems to be power at many other frequencies between the peaks, even though those frequencies were not explicitly defined in the simulated data. This is not an artifact but rather a feature: the nonstationary time series has a more complicated structure and therefore requires energy at a larger number of frequencies in order to represent the time series in the frequency domain.
This is one of two main reasons to perform temporally localized frequency decomposition methods such as *wavelet convolution*, *filter-Hilbert*, or *short-time FFT*.
Extracting More or Fewer Frequencies than Data Points
The frequency resolution of the Fourier analysis is specified by the number of time points in the data (N/2+1)
Zero padding increases the N of the Fourier transform without changing the data, thus allowing you to get more frequencies from the Fourier transform. However, because zero padding does not increase the amount of information in the data, it does not increase the frequency precision of the Fourier transform, only the frequency resolution.

Frequency Domain convolution
Because convolution in the time domain is the same as multiplication in the frequency domain, you can perform convolution in two ways.

(SLOWER) The first way is the time-domain versionof convolution shown in the previous chapter: flip the kernel backward, slide it along the signal, and compute the dot product at each time step.

(FASTER) The second way to perform convolutionis by taking the Fourier transforms of the signal and the kernel, multiplying the Fourier transforms together point-by-point (that is, frequency-by-frequency), and then taking the inverse Fourier transform (figure 11.10). Although this may not seem as if it should makea difference, it does: time-domain convolution is slow, and frequency-domain convolutionis fast.
(1) Power spectrum of convolution is equal to power spectrum of temporal signal.

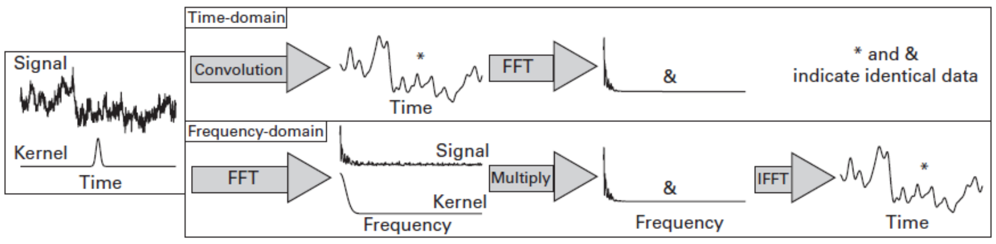
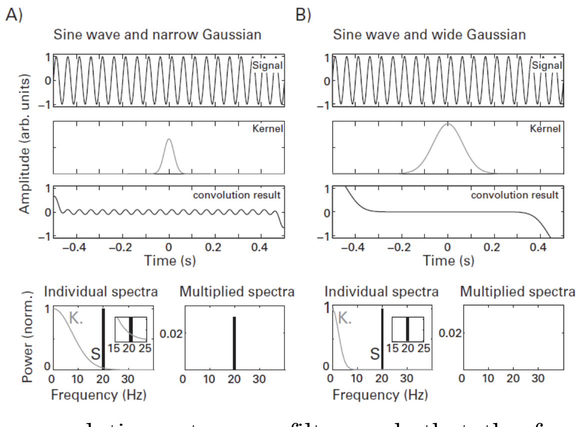


Figure 11.10
Illustration of the convolution theorem and the interchangeability of time-domain convolution and frequency-domain multiplication. The two time series with asterisks are identical, as are the two frequency spectra with ampersands.

Convolution as a Filter
A sine wave of 20 Hz was convolved with two Gaussians — one with a narrow width and one with a wide width (the frequencies of the Gaussian widths were 15 Hz and 5 Hz). The result of the convolution of the sine wave and the narrow Gaussian simply dampened the amplitude of the sine wave, whereas the result of the convolution with the wide Gaussian obliterated the sine wave.

To understand why this is the case, consider the frequency spectra of the sine wave and the two Gaussians, and consider the overlap of their spectra. The sine wave has a very narrow spectral peak at 20 Hz, and the Gaussians have a smoother slope-looking shape. What you should notice is that the power spectrum of the narrow Gaussian has nonzero values that overlap with the nonzero values of the power spectrum of the 20-Hz sine wave (see small inset plots in the bottom row of figure 11.11). In contrast, the power spectrum of the wide Gaussian is zero at frequencies where the power spectrum of the sine wave is nonzero. The resulting frequency spectrum multiplications will produce an amplitude-attenuated sine wave for the narrow Gaussian and no sine wave at all for the wide Gaussian.



Thus, convolution acts as a filter such that the frequency profile of the signal is passed through the frequency profile of the kernel. As you will see in the next several chapters, this is the basis of wavelet convolution: you pass the EEG data through a set of filters (wavelets) that are tuned for specific frequencies, and the result of convolution is the frequency-band intersection between the EEG data and the wavelet. (convolution with a Gaussian is not necessarily the best method for filtering EEG data)

```
result = conv(signal,kernel, ' same ' );
```

If you use the ‘ full ’ option instead of the ‘ same ’ option, the result will the length of the result of convolution, that is, the length of the signal plus the length of the kernel minus one.
Two limitations of the Fourier-transformbased frequency representation were also discussed: changes in frequency structure over time are difficult to visualize, and EEG data violate the stationarity assumption of Fourier analysis. These two limitations provide motivation to apply time-resolved frequency decomposition of EEG data. The next six chapters introduce you to several of the most commonly used methods for obtaining time-resolved frequency representations (or time-frequency representations) of EEG data.

Morlet wavelet
(Gabor wavelet, Gaussian tapering) convolution

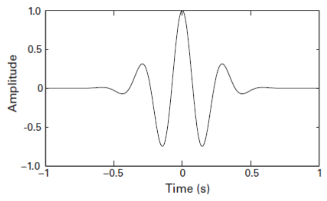


Figure 12.1
A Morlet wavelet, which is created by windowing a sine wave by a Gaussian.

The reason the Fourier transform does not show whether and how the frequency characteristics change over time is that the kernel used in the Fourier transform (a sine wave) has no temporal localization; the amplitude of the sine wave continues to fluctuate over its entire time series (theoretically, from negative). From inspecting figure 12.2 , it should be sensible that to obtain temporally localized frequency information, the dot product with the EEG signal should be computed with only part of a sine wave, specifically the part of a sine wave that occurs at a specific window of time.
Morlet wavelets are not the only kinds of wavelets. **There are many different kinds of wavelets, and you can even create your own wavelets**. **Wavelets must have values at or very close to zero at both ends, and they must have a mean value of zero.**
The stationarity assumption of wavelet convolution is that the signal is stationary only during the time period in which the wavelet looks like a sine wave. (This is a more reasonable assumption that is likely to be valid: EEG data often remain stationary for hundreds of milliseconds (Florian and Pfurtscheller 1995; Jeong, Gore, and Peterson 2002))
Each time point is an estimate of the instantaneous activity and is influenced by activity from neighboring time points.
A sine wave windowed with a Gaussian is called a Morlet wavelet.

Why Morlet ?

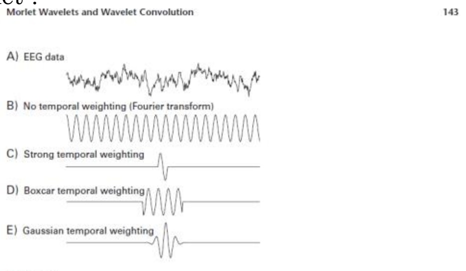


Figure 12.2
In order to extract time-varying frequency-specific information from EEG data (A), the data must be convolved with a sine wave. Without tapering of the sine wave (B), the result reflects frequency-specific information from the entire time series. Use of only one cycle (C) maximizes temporal precision but at the expense of frequency precision. A uniform boxcar tapering (D) is suboptimal because of decreased temporal specificity and potential artifacts from sharp edges. A Gaussian tapering (E) (also known as a Morlet wavelet) provides an adequate balance between temporal and frequency precision without introducing edge artifacts.

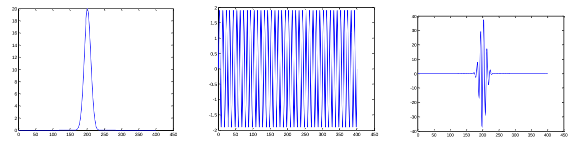
To make a Morlet wavelet, create a sine wave, create a Gaussian, and multiply them point by point (figure 12.3).

$$GaussWin = ae^{-(t-m)^2/(2s^2)}$$

The variable a refers to amplitude (the height of the Gaussian), t is time, m is an x –axis offset (this is not relevant for EEG analyses and can always be set to zero and thus left out of the equation), and s is the standard deviation or the width of the Gaussian. The standard deviation of the Gaussian is defined according to equation 12.2 :

$$s = \frac{n}{2\pi f}$$

The variable f is frequency (in hertz), and n refers to the number of wavelet cycles. The parameter n defines the trade-off between temporal precision and frequency precision. It is a nontrivial parameter that has implications for what kinds of results you can obtain from the data, and it should be carefully selected when you perform your analyses. For this chapter, a value of 6 is used.



The sine wave and the Gaussian must have the same number of time points and the same sampling rate (this also needs to be the same sampling rate as the EEG data). The frequency of the wavelet is the frequency of the sine wave. The frequency of a Morlet wavelet is called its *peak frequency*. Sometimes *center frequency* is also used, but the term *peak frequency* is preferred here because wavelets, unlike sine waves, do not contain energy in only one frequency band but, rather, in a range of frequency bands for which the frequency of the sine wave is the peak.
If you have 1 s of data, use wavelets that are 4 Hz and faster).

Similar to the way a Fourier transform uses many sine waves of different frequencies, time-frequency decomposition via wavelet convolution involves using many wavelets of different frequencies. **Unlike Fourier analysis, however, the frequencies of the wavelet can be specified by you rather than being specified by the number of data points in the time series.** Furthermore, the number of wavelets that can be used is not constrained — you can use as many wavelets as you want, and those wavelets can have almost any frequency you specify. A group of wavelets that share the same properties but differ in frequency is called a *family of wavelets*.

There are a few theoretical and practical limits to constructing a family of wavelets:

You cannot use frequencies that are slower than your epochs. That is, if you have 1 s of data, you cannot analyze activity lower than 1 Hz. In practice, you should have several cycles of activity (for example, if you have 1 s of data, use wavelets that are 4 Hz and faster).

The frequencies of the wavelets cannot be above the Nyquist frequency (one-half of the sampling rate).

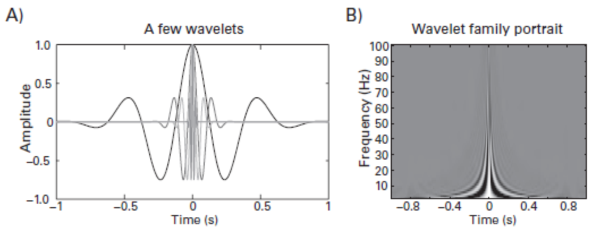


Figure 12.4
Different members of this wavelet family were created by changing the frequency of the sine wave while leaving other parameters unchanged.

Because of frequency smoothing from time-frequency precision trade-offs (this is discussed and demonstrated in the next chapter), frequencies that are very close to each other will likely provide similar or nearly identical results. For example, if you have a wavelet at 15.0 Hz, a wavelet at 14.9 Hz is unlikely to provide any unique information. More frequency bins may produce smoother-looking plots but will also increase computation time without increasing the information in the results. In general, between 15 and 30 frequencies (spanning, say, 3 Hz to 60 Hz) should be sufficient for most experiments.

Morlet Wavelet Convolution as a Bandpass Filter

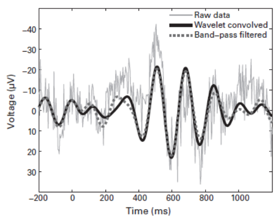


Figure 12.5 shows that convolving EEG data with a wavelet at a specific frequency is similar to bandpass filtering the data around that same frequency.

Figure 12.5 One trial of EEG data plotted before any filtering (gray line), after being bandpass filtered between 4 and 8 Hz (dashed line), and after being convolved with a 6-Hz Morlet wavelet (black line). This shows that wavelet convolution and bandpass filtering produce similar results.

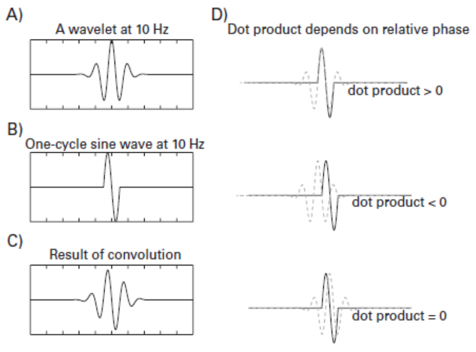
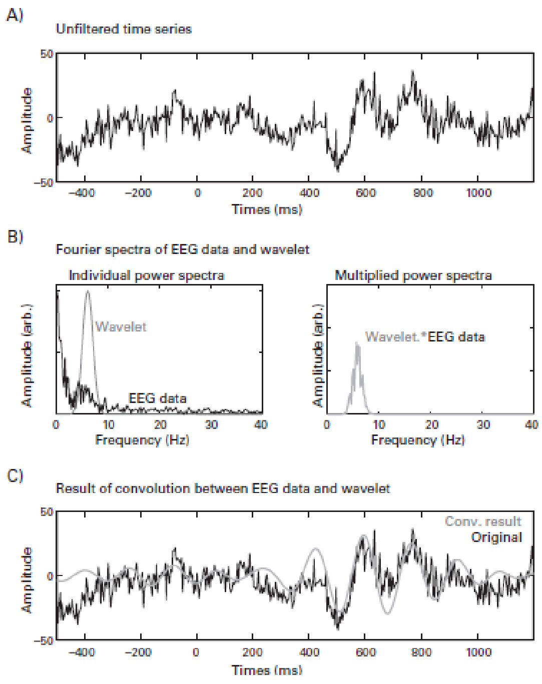


Figure 12.7
The result of each step of convolution (the dot product between the wavelet and the data) depends on the phase relationship between the kernel and the signal at that time point. This issue is resolved by using complex wavelets, as discussed in chapter 13.



To extract power and phase information from EEG data, *complex Morlet wavelets* are necessary. They are called complex because they have a real part and an imaginary part.

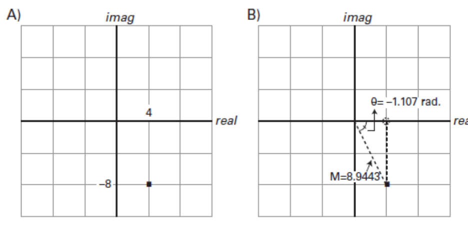


Figure 13.3
The same point in a complex space (with a real axis and an imaginary axis) can be represented using Cartesian (A) or polar (B) notations.

$$M = \sqrt{(real^2 + imag^2)} \qquad real = M\cos(\theta)$$

$$\theta = \arctan(imag / real) \qquad imag = M\sin(\theta)$$

$$real + imag = M\cos(\theta) + M\sin(\theta)$$

$$real + imag = M[\cos(\theta) + \sin(\theta)]$$

$$a + ib = M[\cos(\theta) + i\sin(\theta)]$$

Euler's Formula

$$Me^{i\theta} = M[\cos(\theta) + i\sin(\theta)]$$

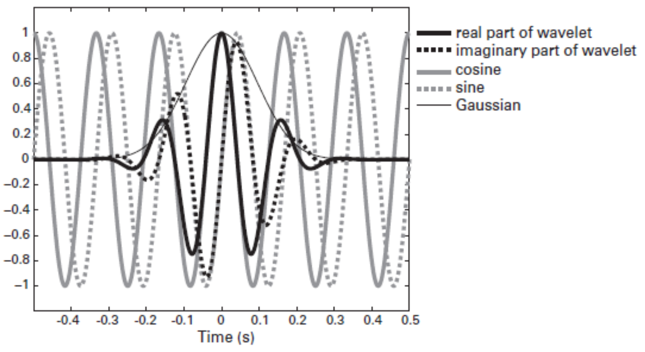


Figure 13.5
Overlaying the real and imaginary parts of a complex wavelet along with a cosine wave and a sine wave. This shows that the real part of the wavelet corresponds to a cosine wave, and the imaginary part of the wavelet corresponds to a sine wave.

Complex Wavelet
Remember from chapter 12 that a real-valued Morlet wavelet is created by multiplying a sine wave by a Gaussian; a complex Morlet wavelet (cmw) is created in the same way, except that the sine wave is a complex sine wave.

$$cmw = Ae^{-t^2/2s^2}e^{i2\pi ft}$$

$$A = \frac{1}{\left(s\sqrt{\pi}\right)^{1/2}}$$

The first part of equation 13.9 (the first exponential) is a Gaussian and should look familiar from equation 12.2 (some of the unused components of the Gaussian formula, for example the x -axis offset, are omitted here for simplicity). The second part of the equation (the second exponential) is a complex sine wave.

You should recognize that this complex sine wave is formed by combining Euler ' s formula and the 2 π ft part of a sine wave. That is, the θ in equation 13.8 is replaced with 2 π ft , in which t is time. The s in equations 13.9 and 13.10 is the standard deviation of the Gaussian (defined in equation 12.2), and f is the peak frequency of the wavelet. A is a frequency band-specific scaling factor.

$$X_f = \sum_{k=1}^{n} x_k e^{-i2\pi f(k-1)n^{-1}}$$

If you look back at equation 11.2, you can see that the formula for the discrete Fourier transform contains Euler ' s formula. Thus, to perform a Fourier transform, a dot product is computed between the time series data and a complex sine wave. It was mentioned several times in this chapter that a complex sine wave comprises both a cosine (real) and sine (imaginary) component. Thus, the formula for the Fourier transform can also be written and computed using two equations, one for sine and one for cosine.

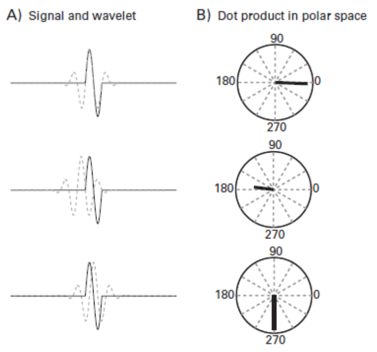


Figure 13.6
The dot product (one step of convolution) between a complex Morlet wavelet (real part shown here using a dotted line) and a one-cycle sine wave produces a complex number, here represented as a vector in a polar plot using the magnitude and angle. Note that the more the wavelet and the one-cycle sine wave overlap, the longer the vector is in complex space, regardless of the phase angle of that vector. This figure can be compared with figure 12.7.

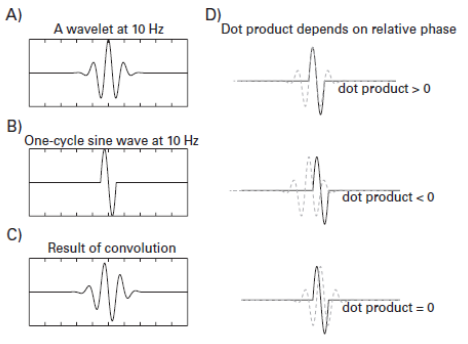


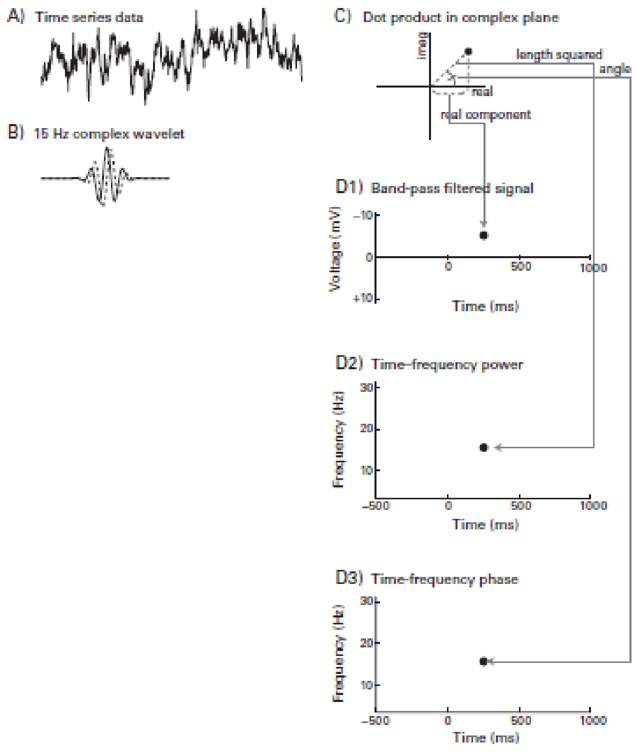
Figure 12.7
The result of each step of convolution (the dot product between the wavelet and the data) depends on the phase relationship between the kernel and the signal at that time point. This issue is resolved by using complex wavelets, as discussed in chapter 13.

There are three pieces of information that can be extracted from the complex dot product (illustrated in figure 13.7).

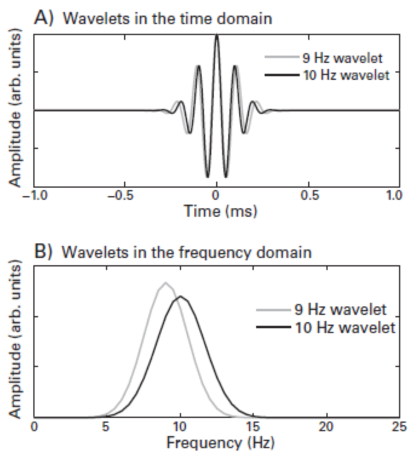
The first piece of information is the projection onto the real axis. This is the bandpass-filtered signal (figure 13.7D1) and was plotted in figures 12.5 and 12.6. This value can be positive or negative, depending on the phase relationship between the kernel and the signal (this was the limitation of real-valued Morlet wavelets shown in figure 12.7). You can also extract the projection onto the imaginary axis, but this is information not commonly used in EEG analyses.

The second piece of information that you can extract from the complex dot product is the magnitude of the vector from the origin to the point in complex space defined by the result of the dot product (equation 13.1 and figure 13.7D2). The length of this vector is related to the similarity or overlap between the kernel (in the case illustrated in figure 13.7 , a 15-Hz complex Morlet wavelet) and the signal (EEG data). Thus, when the EEG data contain a lot of energy at 15 Hz, the result of a dot product with a 15-Hz complex wavelet will produce a point in complex space that is far away from the origin (the magnitude of the line is large). The length (magnitude) of this vector is called the amplitude, and the length squared is called the power. This is the estimate of the instantaneous power at the point in time corresponding to the center of the wavelet with respect to the EEG data, and at the peak frequency of the wavelet. In Matlab, you can use the function abs to extract the magnitude of the complex number.

The third piece of information that can be extracted is the angle of that vector with respect to the positive real axis. This angle is an estimate of the phase angle at the point in time corresponding to the center of the wavelet and at the peak frequency of the wavelet. I Matlab you can use the function angle to extract the phase angle of the complex number. Note that the power and phase values are called estimates because they are influenced by activity at neighboring time points.



Wavelet convolusion as bandpass filter
Real kısm: band pass filtered
Magnitute: (abs(X).^2) veya (X.*conj(X), length squared.
Time frequency power
Angle: thime frequency phase



Lowest Frequency ? (it is a good idea to analyze activity a bit below and a bit above the alpha band to determine whether the effect is frequency band specific).

Should Frequencies Be Linearly or Logarithmically Spaced?

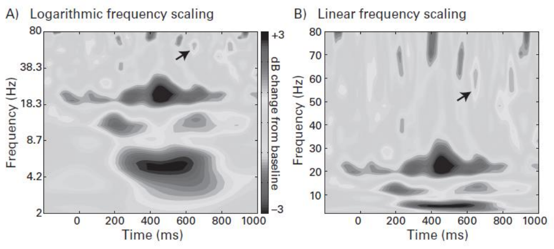


Figure 13.11 (plate 2)
Time-frequency results of the same analyses applied to the same data can look different depending on whether the frequencies in the y-axis are scaled logarithmically (panel A) or linearly (panel B). The black arrows, for example, show the same gamma power burst in both plots.

If your main results concern lower-frequency activity, it is advisable to use logarithmic scaling because this will highlight the lower-frequency end of the spectrum (figure 13.11A); on the other hand, if your main results concern higher-frequency activity, it is advisable to use linear scaling because this will highlight the higher frequencies and visually minimize the lower frequencies (figure 13.11B).

How Many Cycles Should Be Used for the Gaussian Taper?
The number of cycles of the Gaussian taper defines its width, which in turn defines the width of the wavelet.

As is shown below, a larger number of cycles gives you better frequency precision at the cost of worse temporal precision, and a smaller number of cycles gives you better temporal precision at the cost of worse frequency precision. This is the parameter that controls the Heisenberg uncertainty principle applied to time-frequency analysis: the more you know about when something happened, the less you know about where (i.e., at which frequency) it happened, and vice versa.

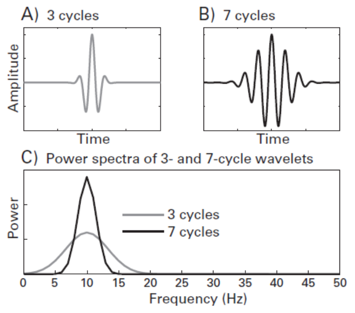


Figure 13.13
Two wavelets at the same frequency that differ only in number of cycles have different temporal and frequency characteristics and therefore different sensitivities to temporal and frequency features of the data. This has practical implications for the results, as shown in figure 13.14 (color plate 3).

Although wavelets at different frequencies may have different lengths, this will complicate your Matlab code and is not necessary. Unless you use wavelets with very low frequencies, defining wavelets using a time range of – 2 s to +2 s should be long enough.

A related and also important detail is that the wavelets should be centered in the time window. The easiest way to ensure that the wavelet is centered in time is to create the wavelet using a time vector from a negative to a positive number (e.g., -2 seconds to +2 seconds). The center of the wavelet will thus be located at time = 0. This procedure has the added benefit that it will give the wavelet an odd number of data points, which is convenient for convolution. Finally, make sure the wavelet is created using the same sampling rate as the EEG data.

How Long Should Wavelets Be?
This is an important issue, but it has an easy answer. Wavelets should be long enough such that the lowest-frequency wavelet tapers to zero (or extremely close to zero) at both the negative and positive ends of time.

How many wavelet cycles should you use with real data? As with many other parameters, there is no single correct answer; it depends on the goal of the analysis. If you are looking for transient changes in activity, a smaller number of cycles (around three or four) will be better. If you have a relatively long trial period in which you expect frequency-band-specific activity (**this could be the case during an extended visual presentation or a working memory delay),** a larger number of cycles (seven to ten) will facilitate identifying temporally sustained activity.

Determining the Frequency Smoothing of Wavelets
The extent to which neighboring frequencies contribute to the result of wavelet convolution can be reported in terms of full width at half-maximum (FWHM).
FWHM refers to the frequency width for which the power is at 50% on the left and right sides of the peak (that is, the lower and upper 50% attenuation frequencies). The formula to compute the FWHM follows.

$$\text{FWHM} = 2\sqrt{2\ln 2}\sigma$$

Tips
You can further decrease analysis time by using FFTs of an order that is a power of two. That is, the FFT of a 1024-point vector is faster to compute than the FFT of a 1023-point vector. Usually, data time series must be zero-padded to get to a length that is a power of two, so make sure you remove these extra points after convolution.

Finally, you do not need to perform convolution on each trial separately; it is faster and more elegant to concatenate all trials into one long time series, perform one convolution with all trials, and then reshape the result back to a time-by-trials matrix.

Hilbert Transform

The main advantage of the filter-Hilbert method over wavelet convolution is that the filter-Hilbert method allows more control over the frequency characteristics of the filter, whereas the frequency shape of a Morlet wavelet is always Gaussian.
There are two minor practical disadvantages to the filter-Hilbert method, which are that the Matlab filter kernel construction functions are in the Matlab signal-processing toolbox and that bandpass filtering is a bit slower than wavelet convolution.

The Hilbert transform is an alternative approach for extracting the imaginary part, iM sin(2 π ft), of a real-valued signal, M cos(2 π ft). This is done by creating and adding the phase quadrature component to the M cos(2 π ft) part. The phase quadrature (or one-quarter-cycle) component is created by rotating parts of the complex Fourier spectrum of a real-valued signal

The following procedure will accomplish the Hilbert transform, and the online Matlab code will show you this procedure step by step. First, compute the Fourier transform of a signal and create a copy of he Fourier coefficients that have been multiplied by the complex operator (i). This turns the M cos(2 π ft) into iM cos(2 π ft). Next, identify the positive and negative frequencies. The positive frequencies are those between but not including the zero and the Nyquist frequencies, and the negative frequencies are those above the Nyquist frequency (throughout the Hilbert transform, the zero and Nyquist frequencies are left untouched). The next step is to convert the iM cos(2 π ft) to iM sin(2 π ft). Remember that cosine and sine are related to each other by one-quarter cycle; thus, to convert a cosine to a sine, you rotate the positive-frequency coefficients one-quarter cycle counterclockwise in complex space (– 90 ° or – π /2) (think about the complex plane: rotating from the positive real axis to the positive imaginary axis turns a cosine into a sine). It was mentioned in section 11.5 that negative frequencies capture sine waves that travel in reverse order around the complex plane. Thus, to convert a cosine to a sine in negative frequencies, you rotate the negative-frequency coefficients one-quarter cycle clockwise (90 ° or π /2).

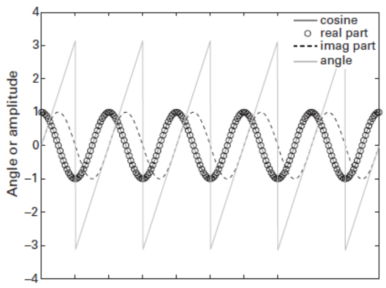


Figure 14.1
The Hilbert transform is used to extract complex information from a real-valued signal. Here, a cosine wave was created (solid gray line), and its Hilbert transform was taken. The real part of the result of the Hilbert transform is the original signal, while the imaginary part is a sine wave (which is what happens when a cosine is phase-shifted 90°).

Rotating the positive-frequency Fourier coefficients counterclockwise can be achieved by multiplication with – i . Notice that the iM sin(2 π ft) transformation for the positive frequencies involves multiplying i with – i , which is 1. This means that when the rotated positivefrequency Fourier coefficients are added to the original positive-frequency coefficients, the effect is to double the original positive-frequency Fourier coefficients.
Rotating the negative-frequency Fourier coefficients can be achieved by multiplication with i . Notice that this involves multiplying i with i , which is -1. Thus, when the rotated negative-frequency coefficients are added back to the original negative-frequency coefficients, the effect is a subtraction from themselves; that is, the negative-frequency coefficients become zero.
The final step is to take the inverse Fourier transform of the modulated Fourier coefficients.

Finite versus Infinite Impulse Response Filters
Filters can be classified as FIR or IIR (finite or infinite impulse response). These terms describe how a filter responds to an impulse — a single input. FIR filters have a response that ends at some point (that is, its response is finite), whereas IIR filters have a response that never end (its response is infinite).

Constructing a Filter
Recall that when you convolve a wavelet with EEG data, the real component of the result of the convolution is the EEG data filtered around the peak frequency of the wavelet. From the convolution theorem, you can say that the result of the convolution is a weighted combination of the frequency structure of the EEG data and the frequency structure of the wavelet (figures 11.10 and 11.11). Bandpass filtering works the same way: a kernel is constructed based on ideal frequency characteristics that you define, and when the kernel is convolved with EEG data, the requested frequencies are preserved while the undesired frequencies are attenuated. You can see in figure 14.3A that the filter kernel looks a bit like a wavelet in that it appears to be a tapered sine wave (or a tapered sum of sine waves), but the waveform is more complex than the Gaussian-tapered sine wave used for Morlet wavelets.

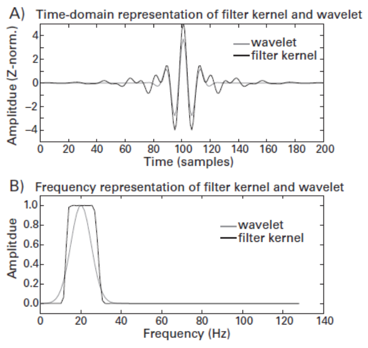


Figure 14.3
Comparison of the time-domain and frequency-domain representations of a Morlet wavelet and an FIR filter kernel. Both the wavelet and the filter kernel were designed to isolate 20-Hz activity. Data were amplitude-normalized to facilitate visual comparison. Panel B also illustrates why it is informative to refer to the "peak" frequency of a wavelet and the "center" frequency of a bandpass filter.

Matlab build-in
firls : create finite-impulse respons filters via *least square*
fir1 : create finite-impulse respons filters via a *windowed linear phase filter with tight transition zones*.

The power spectrum of a Morlet wavelet is Gaussian shaped. In contrast, the power spectrum of an FIR filter can be plateau shaped (figure 14.3B).

Other Matlab functions for creating filter kernels include fir2 (frequency-sampling-based filter construction), firrcos (raised cosine-shaped filter), gaussfir (Gaussian-shaped filter), and firpm (Parks-McClellan).

Short Time Fourier Transform
Fourier transform obscures time-varying changes in the frequency structure of the data, and the Fourier transform assumes that the data are stationary for the duration of the time series.

The short-time FFT method is straightforward: use the FFT to extract the frequency structure of brief segments of data (time windows) rather than the entire time series (Makeig 1993; Welch 1967).

The downside of tapering is that it attenuates valid EEG signal, but this effect can be mitigated by using temporally *overlapping* segments.

The Hann window is advantageous because it tapers the data fully to zero at the beginning and end of the time segment. This eliminates any possibility of even minor edge artifacts. The *Hann* window is advantageous because it tapers the data fully to zero at the beginning and end of the time segment. This eliminates any possibility of even minor edge artifacts. The *Gaussian* can be made to taper to zero, but the window becomes fairly narrow, which may excessively taper the data. The *Hamming* window does not fully taper the data to zero. A comparison of the Hann, Hamming, and Gaussian windows is shown in figure 15.3 .

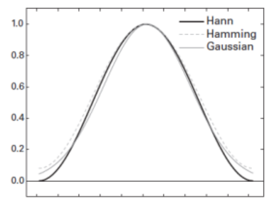
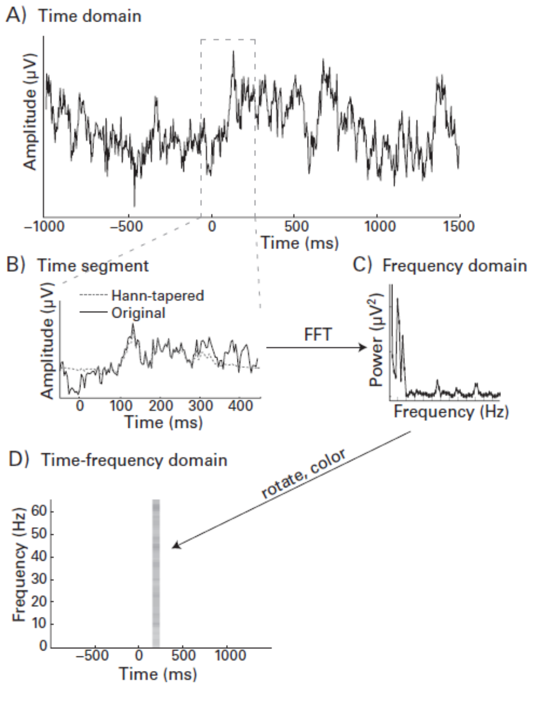


Figure 15.3
Different tapering functions often used for the short-time FFT in EEG data. The two features to look for in a taper are their width—narrow tapers attenuate more signal—and how close they are to zero at the beginning and at the end of the window.

Other tapers that could be used for the short-time FFT include *Kaiser*, *cosine*, and *Blackman*. You can test different windows exhaustively, but it is unlikely that they will provide significantly different results compared to the Hann window.

Overview of the short-time FFT method. From the time series data (panel A), a small segment of the data, comprising a few hundred milliseconds, is taken (panel B). A windowing taper is applied to that segment to minimize the possibility of edge artifacts, and then the Fourier transform of the tapered time series is taken (panel C). The power spectrum of that segment is then placed into a time-frequency space with the frequencies corresponding to that of the FFT and the time point corresponding to the center time point of the time segment from panel B. It is not necessary to perform these procedures separately for each trial because the Matlab function fft can accept a 2-D input (e.g., time by trials). Make sure you take the fft over time, not over trials.



The FFT returns as many frequencies between DC and the Nyquist frequency as there are time points in the segment, although you can keep only the frequencies of interest, such as 4 Hz to 60 Hz. For resting-state data, the same procedure could be applied, but the FFT results would be averaged over time in panel D (e.g., see figure 2 in Allen and Cohen 2010).
When you extract the requested frequencies, you can either extract the single closest frequency bin to each requested frequency or you can take the average of several frequency bins surrounding each requested frequency (figure 15.4). Taking an average is a good strategy because it increases the signal-to-noise ratio and minimizes the possibility that the frequency estimate is driven by outliers or nonrepresentative data.

Note that taking the *average power* from several neighboring frequencies makes the result of the short-time FFT method more similar to results from the *filter-Hilbert method.* To make the results from the short-time FFT method more similar to the results of *complex wavelet convolution*, you could average *Gaussian-weighted* neighboring frequencies.

Short-Time FFT 199

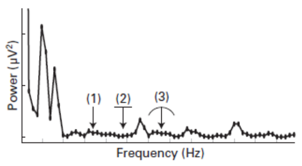


Figure 15.4
Illustration of different ways of extracting frequencies from the FFT of each time segment. Because the FFT returns more frequencies than you probably would want to include in the analyses, you can select "requested frequencies" from the full FFT result. To obtain the requested frequencies you can either select the single frequency bin closest to each requested frequency (arrow 1), average neighboring frequencies (arrow 2), or apply a distance weighting such as a Gaussian to neighboring frequencies to compute a weighted average (arrow 3). The latter two options increase the signal-to-noise ratio of the results of the short-time FFT. In practice any one of these methods can be used, but it should be consistently applied to all frequencies and all data within a study.

The length of the time segment offers the much-discussed trade-off between temporal and frequency precision and resolution: shorter time segments provide better temporal precision at the expense of frequency precision and resolution, whereas longer time segments provide better frequency precision and resolution at the expense of decreased temporal precision.

Frequency resolution of a Fourier transform is defined by the number of points in the time series. Therefore, the larger the time segment, the more frequencies can be extracted, and thus the greater the frequency resolution. The temporal resolution is always defined by the data sampling rate and thus does not depend on the length of the time segment.

The choice of time segment length is also related to the lowest frequency you want to analyze. The segment must be long enough to capture at least one cycle of the lowest frequency, and preferably more than one cycle to increase signal-to-noise ratio. Thus, if you want to analyze 3-Hz activity, the window must be at least 333 ms long, and preferably 667 ms or 999 ms to capture two or three cycles (or any length; time segment lengths need not be integer multiples of cycle durations). On the other hand, 999 ms might be so long as to average over and thus fail to capture transient higher-frequency activity.

To adapt the short-time FFT method such that the trade-off between temporal precision and frequency precision changes as a function of frequency, you can change the time segment length as a function of frequency: lower frequencies can have longer time segments while higher frequencies can have shorter time segments. This procedure will have a similar effect to wavelet convolution with a variable number of cycles (the width of the Gaussian) as a function of frequency. In this case the FFT will be performed several times on time segments around each time point. Each time the FFT is performed, a range of frequencies is extracted. For example, for a center time point of 300 ms, the FFT can be computed on the data from – 100 to +700 ms, and frequencies between 4 and 20 Hz are extracted. Next, the FFT is computed on the data from 0 to 600 ms, and frequencies between 20 Hz and 40 Hz are extracted. And so on. This is a simplified example; you could also change the length of the time segment for each requested frequency.

A second parameter of the short-time FFT is the amount of overlap between successive time segments. Having temporal overlap is useful for three reasons: it improves the temporal precision, it mitigates the loss of signal due to tapering, and it smoothes the time-frequency plots, thus facilitating visual inspection of the time courses of activities. Although there are no rules or strict guidelines for how much overlap to use, values between 50% and 90% of the length of the segment are acceptable. For example, steps of 75 ms for a 300-ms time segment would result in a 75% overlap.

If you use the short-time FFT approach to analyze your data, make sure the Methods section in your paper clearly states the relevant parameters so that someone else could replicate your analyses. The relevant parameters include the overlap between successive time segments, how many frequencies were extracted (that is, the requested frequencies), whether the time segment length changed as a function of requested frequency, whether neighboring frequencies were averaged together to increase signal-to-noise ratio, and finally, which function was used to taper the data. If you used a software package or toolbox such as eeglab, write which functions were used, and any parameters that were changed from their default settings.

Multitapers
The multitaper approach is an extension of the short-time FFT method that is designed to increase the signal-to-noise ratio of the frequency representation by applying several tapers that have slightly different temporal (and thus spectral) characteristics (Mitra and Pesaran 1999; Thomson 1982).

The multitaper method is particularly useful for situations of low signal-to-noise ratio, such as higher-frequency activity or single-trial estimates of power. For frequencies lower than around 30 Hz, the multitaper method may be less appropriate because the signal-tonoise ratio is already relatively high and because the spectral smoothing resulting from the multitaper procedure may impede frequency isolation, meaning that activities from multiple frequency bands become averaged together.

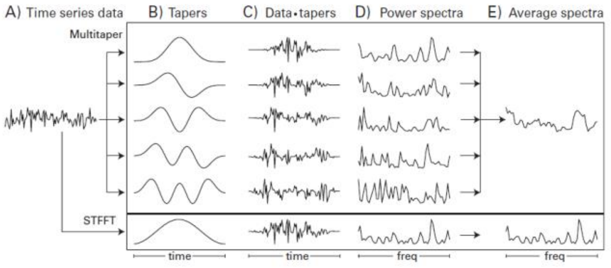


Figure 16.1
Overview of the multitaper method and comparison with the short-time FFT approach. The *y*-axes were arbitrary scaled to facilitate visual comparison. Note that the *x*-axes show time in columns A, B, and C and frequency in columns D and E. The dot between data and tapers in the label of panel C indicates the pointwise multiplication.

The tapers
The tapers used for the multitaper method are called discrete prolate spheroidal sequences
and can be obtained via the Matlab function `dpss`. (Also, `pmtm` for multi-taper analysis)
Slepian tapers are **orthogonal** to each other (that is, the dot product of any one taper with any other taper is 0), and they have slightly different frequency characteristics, thus focusing the spectra of the resulting tapered time series on different parts of the spectrum.
The multitaper method introduces some smoothing on the frequency axis. The amount of smoothing depends on the number of tapers, such that the more tapers that are used, the smoother the results.

When You Should and Should Not Use Multitapers
There are three situations in which the multitaper approach may be beneficial to your results. The first situation is if you have noisy data or a small number of trials and are concerned about the influence of noise on the results. The second, and related, situation is if you are performing single-trial analyses, particularly if you would like to analyze frequencies above around 30 Hz. The third situation is if you would like to focus on high-frequency power, particularly above around 60 Hz. The multitaper approach will help increase the signal-to-noise ratio of high-frequency activity, and the temporal and frequency smoothing may facilitate cross-subject averaging.

Several EEG analysis toolboxes will perform a multitaper-based timefrequency decomposition. However, the Matlab toolbox *fieldtrip* has the most active development of this method, including, for example, options to create frequency-band-selective tapers that further enhance their sensitivity beyond the basic multitaper application discussed in this chapter.

For multitaper analysis alseo read
Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques (Percival and Walden 1993).
Observed Brain Dynamics (Mitra and Bokil 2007).