CHAPTER

3

Signal Analysis in the Frequency Domain: The Fourier Series and the Fourier Transformation

3.1 GOALS OF THIS CHAPTER

In this chapter we determine how to find the sinusoidal components of a general signal. But this is something we have already done in Example 2.15 when we used correlation between an electroencephalography (EEG) signal and sinusoids to search for oscillatory behavior.[1] Here we develop a computationally more efficient approach to do the same thing. More importantly, we dig deeper into the greater significance of those sinusoidal components. We also address some issues we glossed over in Example 2.15. For example, in our search for oscillatory behavior, we compared the EEG signal with a group of sinusoids ranging in frequencies between 1 and 25 Hz in 0.5-Hz intervals. Could we have missed some oscillatory behavior with this arbitrary range of frequencies and frequency increments? Should we have used a broader range of frequencies and/or finer intervals, or perhaps used fewer sinusoids and larger intervals? Here we offer a definitive answer to these questions.

We also find that correlating a signal with a series of sinusoids gives us more than just a measure of the signal's oscillatory behavior. If we choose the right combination of sinusoids, and we use enough of them, the correlation coefficients become an equivalent, alternative representation of the signal. In other words, the correlation coefficients give a complete representation of the signal and it is possible to reconstruct the original signal from just these coefficients. This works only if we use enough sinusoids at the right frequencies, but the technique for finding the right correlation coefficients is straightforward.

The correlation coefficient signal representation can be very useful both in providing a new description of the signal and in certain operations applied to the signal. The correlation

[1]We actually used cross-correlation between a cosine and an EEG signal. But in cross-correlation, the waveforms are shifted with respect to one another, so it is as if we were performing simple correlations between the EEG and sinusoids that have a range of phase shifts. We then took the maximum cross-correlation as the best match between the EEG and a shifted sinewave.

Copyright © 2018 Elsevier Inc. All rights reserved.

coefficient representation of a signal is called the "frequency domain" representation since it is based on the correlations with sinusoids having a range of frequencies. Converting a signal from its time domain representation to its frequency domain representation is an example of a class of operations known as "signal transformations" or just "transformations." Again, the two representations are completely interchangeable. You can go from the time to the frequency domain and back again with no constraints; you just have to follow the mathematical rules.

In Example 2.15 we used digital signals in the digital domain, but in this chapter we also determine sinusoidal correlations analytically from continuous signals in the continuous domain. In fact, time–frequency domain transformations were first developed in the continuous domain and laboriously worked out by hand before the advent of the digital computer. We do a few easy examples on continuous domain operations because as engineers we should know the inner workings of this important transformation. Fortunately, all real-world time–frequency transformations are done in the digital domain on a computer.

To summarize, in this chapter we will:

- Show how to decompose any periodic waveform into a series of sinusoidal components and how to do the opposite, recombine the sinusoidal components into a waveform.
- Demonstrate how sinusoidal decomposition leads to the frequency characteristics of the spectrum of a waveform.
- Describe the Fourier transform using complex notation.
- Use the Fourier transform to find the frequency domain representation of a periodic waveform.
- Show one method for tracking the changes in waveform's spectrum over time.

3.2 TIME–FREQUENCY DOMAINS: GENERAL CONCEPTS

We use the Fourier series theorem, commonly known as the Fourier transform, to move between the time and frequency domains. The Fourier series theorem is one of the most important and far-reaching concepts presented in this book. Its several versions are explored using both analytical solutions in the continuous domain and computer algorithms in the digital domain. For a digital signal, both time and frequency representations consist of a sequence of discrete numbers. In the analog domain, the time domain representation is a continuous function of time, but it can also be described using a time plot. The frequency domain representation of an analog signal depends on the type of signal. If it is periodic, the frequency domain representation is a discrete series corresponding to specific frequencies, but if it is aperiodic, the frequency domain representation becomes a continuous function of frequency. Either way, the frequency plot describes the signal's spectrum, and the terms "spectrum" or "spectral" are inherently frequency domain terms. Bandwidth is also a frequency domain term, as it describes the characteristics of a signal's spectrum.

There are several motivations for transforming a signal into the frequency domain. Sometimes the spectral representation of a signal is more understandable than the time domain representation. The EEG time domain signal, such as shown in Figure 1.7, is quite complicated. The time domain signal may not be random, but whatever structure it contains is

not apparent in the time response. For example, the frequency domain representation determined in Example 2.15[2] revealed a structure in which certain frequencies contained more oscillatory behavior (i.e., more energy) than other frequencies. Electroneurophysiologists often use the spectral representation to study EEG waves and have identified some especially important frequencies. For example, an oscillation found between 8 and 12 Hz is called an "alpha wave" and is present when a subject is at rest with eyes closed, but not tired or asleep. Other oscillatory features revealed in the EEG frequency spectrum include the beta wave between 12 and 30 Hz associated with the muscle contractions or sensory feedback in static motor control, and the "delta wave" between 0 and 4 Hz, usually associated with deep sleep.

Systems can also be represented in the time and/or frequency domain. The spectral representation of a system provides a good description of how that system operates on signals that pass through it. For systems, the frequency domain is the more intuitive: it shows how the system alters each frequency component in the signal. Specifically, the system's spectrum shows if the signal energy at a given frequency is increased, decreased, or left the same and how it might be shifted in time. The same mathematical tools used to convert signals between the time and frequency domains are used to flip between the two system representations and, again, there are no constraints.

3.3 TIME–FREQUENCY TRANSFORMATION OF CONTINUOUS SIGNALS

Sinusoids are the most basic of waveforms and that makes them of particular value in signal analysis. Because of their simplicity, it is easy to represent them in either the time or frequency domain and we use them as a gateway between the two domains. Here we list four properties of sinusoids that we use for time–frequency transformations and two other properties that will come in handy in system analysis.

3.3.1 Sinusoidal Properties in the Time and Frequency Domains

1. Sinusoids have energy at only one frequency: the frequency of the sinusoid. This property is unique to sinusoids. Because of this property, sinusoids are sometimes referred to as "pure": the sound a sinusoidal wave produces is perceived as pure or basic. (Of the common orchestral instruments, the flute produces the most sinusoidal-like tone.)
2. Sinusoids are completely described by three parameters: magnitude (amplitude), phase, and frequency. If frequency is fixed, then only two variables are required, amplitude and phase. Moreover, if you use complex variables, these two parameters (amplitude and phase) can be rolled into a single complex variable.

We already know how to describe sinusoids in the time domain (e.g., Equation 1.14). Combining Property 1 with Property 2 gives us the frequency domain representation of a

[2]As we discover later in this chapter, Example 2.15 found only half of the frequency domain representation of the EEG signal. However, the half that is missing is not very informative and is often not displayed. But it is needed if we want to reconstruct the EEG signal from its frequency domain representation.

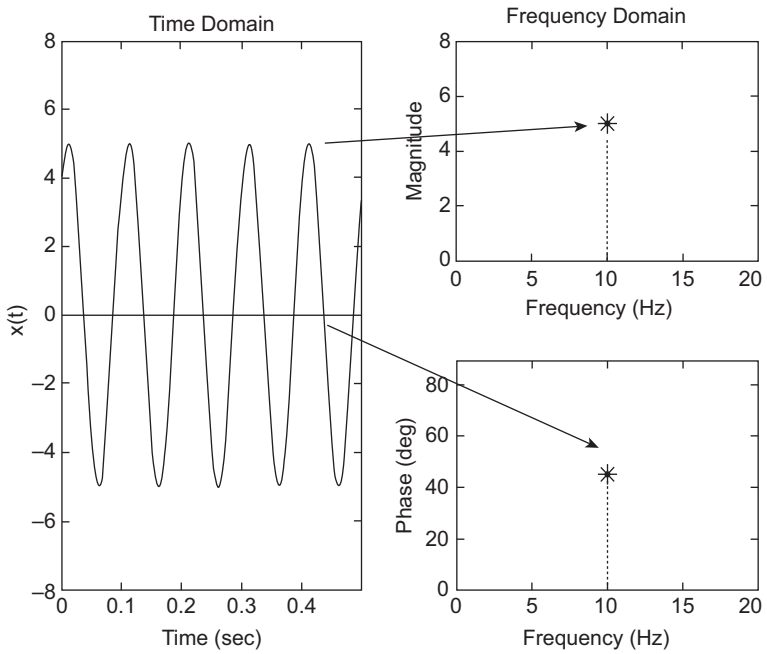


FIGURE 3.1 A sinusoid is completely represented by its magnitude and phase at a given frequency and its spectrum can be represented as single points on two plots: magnitude and phase against frequency.

sinusoid; i.e., its spectrum. It is uniquely defined by two points: one point on a plot of magnitude versus frequency and another on a plot of phase versus frequency, Figure 3.1. Both points are plotted against the frequency of the sinusoid. Moving sinusoids between the time and frequency domain is trivial. To go from the time to frequency domain: take the amplitude and phase of the sinusoid and plot them against its frequency in the magnitude and phase plots. To go from the frequency to time domain, find the sinusoidal amplitude from the magnitude plot, its phase from the phase plot, and the frequency from either plot and enter these into an equation such as Equation 1.14.

3.3.2 Sinusoidal Decomposition: The Fourier Series

If we could transform a general signal into sinusoids, we could use those sinusoidal components to determine the frequency domain representation of that signal. This brings us to the third important property of sinusoids: the "Fourier series."

3. All periodic signals[3] can be broken down into a series of sinusoids, plus a constant to account for any offset or DC component.

$$x(t)_{\text{periodic}} = C_0 + \text{sinusoid}_1 + \text{sinusiod}_2 + \text{sinusiod}_3 + \ldots \text{sinusoid}_M \tag{3.1}$$

[3]Recall that a periodic signal repeats itself exactly after some period of time, T: $x(t) = x(t + T)$.

where C_0 is the constant representing any offset or bias in the signal. The only constraint on the signal is that it must be periodic, or assumed to be periodic. Moreover, all the sinusoids in this series will be at the same or multiples of the frequency of the periodic signal. Hence, each sinusoidal component in the series will take the form:

$$C_m \cos(2\pi m f_1 t + \theta_m) \tag{3.2}$$

where f_1 is a constant equal to the frequency of the periodic signal ($f_1 \equiv f_p = 1/T$) and m is the reference number of the sinusoidal component and is an integer, $m = 1, 2, 3, \ldots$. The two variables are those that define the sinusoid (along with frequency): C_m, the amplitude of the mth sinusoid in the series, and θ_m, the phase angle of the mth sinusoid. A waveform at a multiple frequency of another waveform is termed a "harmonic" and a series containing sinusoids at multiple frequencies is called a harmonic series.[4] The sinusoidal series number m is also referred to as the harmonic number.

Substituting the format for a general sinusoid in Equation 3.2 into the series of Equation 3.1 leads to the formal Fourier series equation:

$$x(t) = \frac{C_0}{2} + \sum_{m=1}^{M} C_m \cos(2\pi m f_1 t + \theta_m) \tag{3.3}$$

Equation 3.3[5] works in both directions. You can represent any periodic signal by a sinusoidal series and you can construct any periodic signal by simply adding together harmonically related sinusoids. Of course, any sinusoidal component of the series must have a specific magnitude and phase (i.e., C_m and θ_m) to represent a given signal and you might need a large number of such sinusoids to accurately reproduce the signal, but it can always be done.[6] You can readily flip back and forth, convert the signal to a sinusoidal series, then sum the series to reconstruct the signal. This flipping between time and sinusoidal representation of a signal is illustrated in Figure 3.2. In this example the reconstructed signal (Figure 3.2 right side) is not perfect, but only three sinusoids were used to represent the signal.

3.3.3 Fourier Series Analysis and the Fourier Transform

Combining the sinusoidal decomposition properties of Property 3 with the frequency representation of sinusoids in Property 2 gives us a powerful and useful transformation: the ability to convert any periodic signal from the time domain into the frequency domain (and vice versa). Just decompose the periodic time domain signal into equivalent series of sinusoids,

[4]In music, an octave is the doubling in frequency of a musical tone. So in a harmonic series, each component is an octave apart from its higher and lower frequency neighbors.

[5]The first term of this equation is the mean value of $x(t)$. Different normalizations of the Fourier series can be found in the literature, but for the normalization used here C_0 is defined as twice the mean value (see Equation 3.10), hence it is divided by 2 in this equation.

[6]In a few cases, such as the square wave, you need an infinite number of sinusoidal waves. As shown in Figure 3.6, a finite series will exhibit an oscillatory artifact.

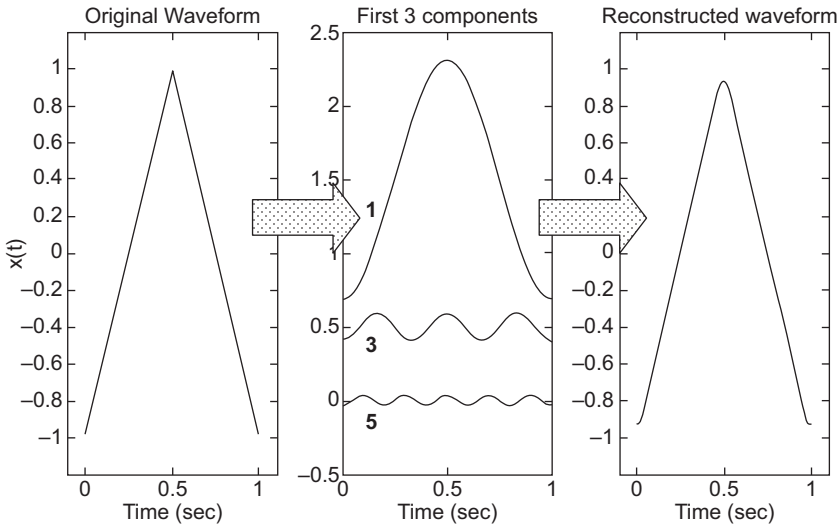


FIGURE 3.2 The periodic signal on the left (only one cycle shown) is decomposed into three harmonic sinusoids in the middle figure. (Recall that Figure 3.1 shows that these three signals can be represented by three points on a magnitude and phase frequency plots.) When the three sinusoids are added together, they produce a fair reconstruction of the original time domain signal (right side plot). More components would give a more accurate representation of the time single and a better reconstruction. (In Figure 3.5 we try reconstructing a periodic square wave using different numbers of sinusoids.)

and represent each sinusoid in the frequency domain as a magnitude and phase plot, Figure 3.3.

The only question is, how do we get the appropriate sinusoidal components? We already know the frequencies of the sinusoidal components; they are harmonics of the signal's frequency based on its period: $f_1 = 1/T$. (The base frequency, f_1, is also known as the "fundamental frequency.") The frequency associated with a given component is the fundamental frequency, f_1, times the harmonic number m:

$$f = \frac{m}{T} = mf_1 \tag{3.4}$$

For example, each C_m in Equation 3.3 would appear as a single point on the magnitude (Figure 3.3, upper right graph) plot, whereas each θ_m shows as a single point on the phase curve (Figure 3.3, lower right graph). These frequency plots provide the frequency domain representation, just as the original plot of $x(t)$ is the time domain representation.

We still need to find the magnitude and phases, the appropriate C_m's and θ_m's in Equation 3.3, but we already have an idea on how to find components in a time signal: use cross-correlation.[7] To summarize, sinusoids found through correlation provide a gateway between

[7]In practice we use alternative methods that are easier or faster. In the analog domain, we apply standard correlation (not cross-correlation) to a sine/cosine representation of Equation 3.3 and in the digital domain we apply correlation to a complex representation of the sinusoid (recall the Euler identity).

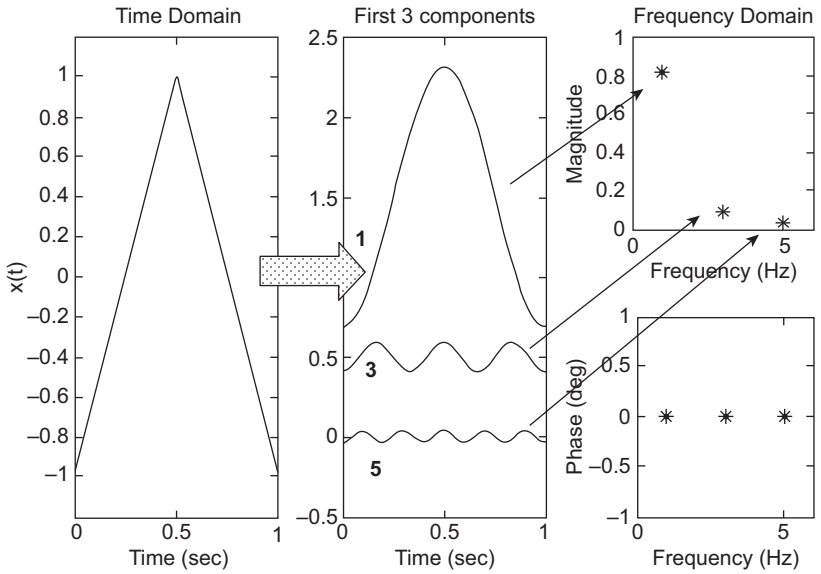


FIGURE 3.3 By combining the simplicity of a sinusoid's frequency representation (Property 2, a magnitude and a phase point on the frequency plot) with harmonic decomposition (Property 3) we can convert any periodic signal from a time domain plot to a frequency domain plot. If enough sinusoidal components are used, the frequency domain representation is equivalent to the time domain representation.

the time and frequency representations for any periodic function. We work out the details of calculating the correlation coefficients in the next section, but first we list some other useful sinusoidal properties.

4. As found in Chapter 2, harmonically related sinusoids are orthogonal (see Example 2.13). For Fourier series analysis this means that the values we calculate for a given Fourier series component (i.e., the C_m's and θ_m's) will be independent from the values of all other components. Practically, this means that if we decide to decompose a waveform into 10 harmonically related sinusoids, but later decide to decompose it into 12 sinusoids to attain more accuracy, the addition of more components will not change the C_m and θ_m values of the 10 components we already have.

The remaining two properties are not used in this chapter, but are useful in later discussions of systems analysis.

5. The calculus operations of differentiation and integration change only the magnitude and phase of a sinusoid. The result of either operation is still a sinusoid at the same frequency. For example, the derivative of a sine is a cosine, which is just a sine with a 90-degree phase shift; the integral of a sine is a negative cosine, the same as a sine with a -90-degree phase shift. In general, the derivative or integral of any sinusoid, $A \sin(2\pi ft + \theta)$, is unchanged except that A is scaled and θ is shifted by ± 90 degrees.

6. If the input to any linear system is a sinusoid, the output is another sinusoid at the same frequency irrespective of the complexity of the system. (Linear systems are defined

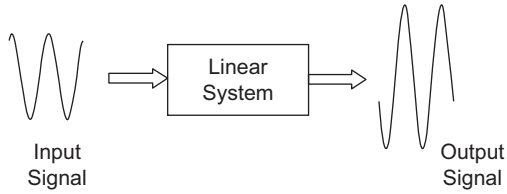


FIGURE 3.4 The output of any linear system driven by a sinusoid is a sinusoid at the same frequency. Only the magnitude and phase of the output differs from the input.

and studied in Chapters 5 and 6.) The only difference between the input and output is the magnitude, A, and phase, θ, of the sinusoid, Figure 3.4. This is because linear systems are composed of elements that can be completely defined by scaling, derivative, or integral operations and, as noted in Chapter 6, these operations modify only the magnitude and phase of a sinusoid.

3.3.4 Finding the Fourier Coefficients

The Fourier series equation, Equation 3.3, can give us the frequency domain representation of any signal, $x(t)$. For each sinusoidal term (i.e., each value of harmonic number m), the coefficient C_m contributes one magnitude frequency point, and the coefficient θ_m provides one magnitude phase point.[8] So all we have to do is find those coefficients. In principle we could use cross-correlation, but that is difficult to do analytically (and not efficient when we use the computer). Instead we split the sinusoidal term in Equation 3.3 into sine and cosine terms using Equations 2.23 and 2.24. The Fourier series equation of Equation 3.3 then becomes:

$$x(t) = \frac{a_0}{2} + \sum_{m=1}^{\infty} a_m \cos(2\pi m f_1 t) + \sum_{m=1}^{\infty} b_m \sin(2\pi m f_1 t) \tag{3.5}$$

where $a_0 \equiv C_0$ in Equation 3.3, and from Equation 2.24, noting that θ is defined as negative in those equations: $a_m = C_m \cos(-\theta) = C_m \cos(\theta)$ and $b_m = C_m \sin(-\theta) = -C_m \sin(\theta)$. In Equation 3.5 the a and b coefficients are the amplitudes of the cosine and sine components of $x(t)$. The Fourier series equation using a and b coefficients is referred to as the "rectangular representation," whereas the equation using C and θ is called the "polar representation." Rectangular to polar conversion is done using Equations 2.25 and 2.26 and are repeated here slightly modified:

$$C_m = \sqrt{a_m^2 + b_m^2} \tag{3.6}$$

[8]Note that the frequency domain representation of a continuous periodic signal is discrete: it is a series of discrete coefficients, C_m and θ_m. Later in this chapter we find that the frequency representation of a continuous aperiodic signal is continuous.

$$\theta_m = \tan^{-1}\left(\frac{-b_m}{a_m}\right) \tag{3.7}$$

Again, Equation 2.26 solves for negative θ, but in the Fourier series equations, Equations 3.2 and 3.3, θ is positive. By making b_m negative in Equation 3.7, the resulting θ becomes positive.

Using the rectangular representation of the Fourier series (Equation 3.5), we can find the a_m and b_m coefficients by simple correlation. We apply Equation 2.31, the basic correlation equation in the continuous domain, where $y(t)$ is the sine or cosine term and $x(t)$ is the signal (or vice versa):

$$a_m = \frac{2}{T}\int_0^T x(t)\cos(2\pi m f_1 t)dt \quad m = 1, 2, 3, \ldots \tag{3.8}$$

$$b_m = \frac{2}{T}\int_0^T x(t)\sin(2\pi m f_1 t)dt \quad m = 1, 2, 3, \ldots \tag{3.9}$$

These correlations are calculated for each harmonic number, m, to obtain the Fourier series coefficients, the a_m's and b_m's, representing the cosine and sine amplitudes at the associated frequencies, $2\pi m f_1$. (A formal derivation of Equations 3.8 and 3.9 are given in Appendix A-2.) Equations that calculate the Fourier series coefficients from $x(t)$ are termed "analysis equations." Equation 3.5 works in the other direction, generating $x(t)$ from the a's and b's, and is known as the "synthesis equation."

The factor of 2 is used because in the Fourier series, the coefficients, a_m and b_m, are defined in Equation 3.3 as amplitudes, not correlations. However, there is really no agreement on how to scale the Fourier equations, so you might find these equations with other scalings. MATLAB's approach is avoidance: its routine for determining these coefficients uses no scaling.[9] When using MATLAB to find a_m and b_m, it is up to you to scale the output as you wish.

Our usual strategy for transforming a continuous signal into the frequency domain is to first calculate the a and b coefficients, then convert them to C and θ coefficients using Equations 3.6 and 3.7. We use this approach because it is easier to work out the correlations using sines and cosines (i.e., Equations 3.8 and 3.9) than cross-correlating against a sinusoidal term. That said, for complicated signals, it can still be quite difficult to solve for the Fourier coefficients equations analytically. Fortunately, for all real-world signals, these equations are implemented on a computer.

The constant term in Equation 3.5, $a_0/2$, is the same as $C_0/2$ in Equation 3.3 and is also known as the "DC term." It accounts for the offset or bias in the signal. If the signal has zero mean, as is often the case, then $a_0 = C_0 = 0$. Otherwise, the value of the DC term is just twice the mean:

$$a_0 = \frac{2}{T}\int_0^T x(t)dt \tag{3.10}$$

[9]Although MATLAB determines the Fourier series coefficients using an algorithm that involves the complex representation of a sinusoid, the output of this algorithm is essentially the a and b coefficients which are the real and imaginary parts of the output variable.

The reason a_0 and C_0 are calculated as twice the average value is for them to be compatible with the Fourier analysis equations of Equations 3.8 and 3.9, which also involve a factor of 2. To offset this doubling, a_0 and/or C_0 are divided by 2 in the synthesis equations, Equations 3.3 and 3.5 (logical, if a bit confusing).

Sometimes, $2\pi mf_1$ is stated in terms of radians, where $2\pi mf_1 = m\omega_1$. Using frequency in radians makes the equations look cleaner, but in engineering practice frequency is measured in hertz. Both are used here. Another way of writing $2\pi mf_1$ is to combine the mf_1 into a single term, f_m, so the sinusoid is written as $C_m \cos(2\pi f_m t + \theta_m)$ or in terms of radians as $C_m \cos(\omega_m t + \theta_m)$. So an equivalent representation of Equation 3.3 is:

$$x(t) = \frac{C_0}{2} + \sum_{m=1}^{\infty} C_m \cos(\omega_m t + \theta_m) \tag{3.11}$$

Equations 3.8 and 3.9 are also sometimes written in terms of a period normalized to 2π. This can be useful when working with a generalized $x(t)$ without the requirement for defining a specific period.

$$a_m = \frac{2}{\pi}\int_{-\pi}^{\pi} x(t)\cos(mt)dt \quad m = 1, 2, 3, \ldots \tag{3.12}$$

$$b_m = \frac{2}{\pi}\int_{-\pi}^{\pi} x(t)\sin(mt)dt \quad m = 1, 2, 3, \ldots \tag{3.13}$$

Here we always work with time functions that have a specific period to make our analyses correspond more closely to real-world applications. So Equations 3.12 and 3.13 are for reference only.

To implement the integration and correlation in Equations 3.8 and 3.9, there are a few constraints on $x(t)$. First, $x(t)$ must be capable of being integrated over its period; specifically:

$$\int_0^T |x(t)|dt < \infty \tag{3.14}$$

Unfortunately this constraint rules out a large class of interesting signals: transient signals as described in Section 1.4.2 and shown in Figure 1.20. Recall, these are signals that change, rapidly or slowly, but do not repeat and do not return to a baseline in a finite amount of time. Because the change lasts forever, the integral in Equation 3.14 must be taken between 0 to ∞ and is not finite. In some cases, you might be able to recast a transient signal into a periodic signal and this is explored in one of the problems.

The second constraint is that, although $x(t)$ can have discontinuities, those discontinuities must be finite in number and have finite amplitudes. Finally, the number of maxima and minima must also be finite. These three criteria are sometimes referred to as the "Dirichlet conditions" and are met by many real-world signals.

A brief note about terminology: The analysis (Equations 3.8 and 3.9) and their discrete equivalents should be called "Fourier series analysis," but they are often called the "Fourier

transform," especially when implemented in the discrete domain on a computer. Technically, "Fourier transform" should be reserved for the analysis of continuous aperiodic signals. Likewise the synthesis equation, Equation 3.5, and its discrete equivalent are usually called the "inverse Fourier transform." This usage is so common that it is pointless to make distinctions between the Fourier transform and Fourier series analysis.

The more sinusoids included in the summations of Equations 3.3 or 3.5, the better the representation of the signal, $x(t)$. For an exact representation of a continuous signal, the summation should be infinite, but in practice the number of sine and cosine components that have meaningful amplitudes is limited. Often only a few sinusoids are required for a decent representation of the signal. Figure 3.5 shows the reconstruction of a square wave by a different number of sinusoids: 3, 9, 18, and 36. The square wave is one of the most difficult waveforms to represent using a sinusoidal series because of the sharp transitions. Figure 3.5 shows that the reconstruction is fairly accurate even when the summation contains only nine sinusoids.

The square wave reconstructions shown in Figure 3.5 become sharper as more sinusoids are added, but still contain oscillations. These oscillations, termed Gibbs artifacts, occur whenever a finite sinusoidal series is used to represent a signal with discontinuity. They

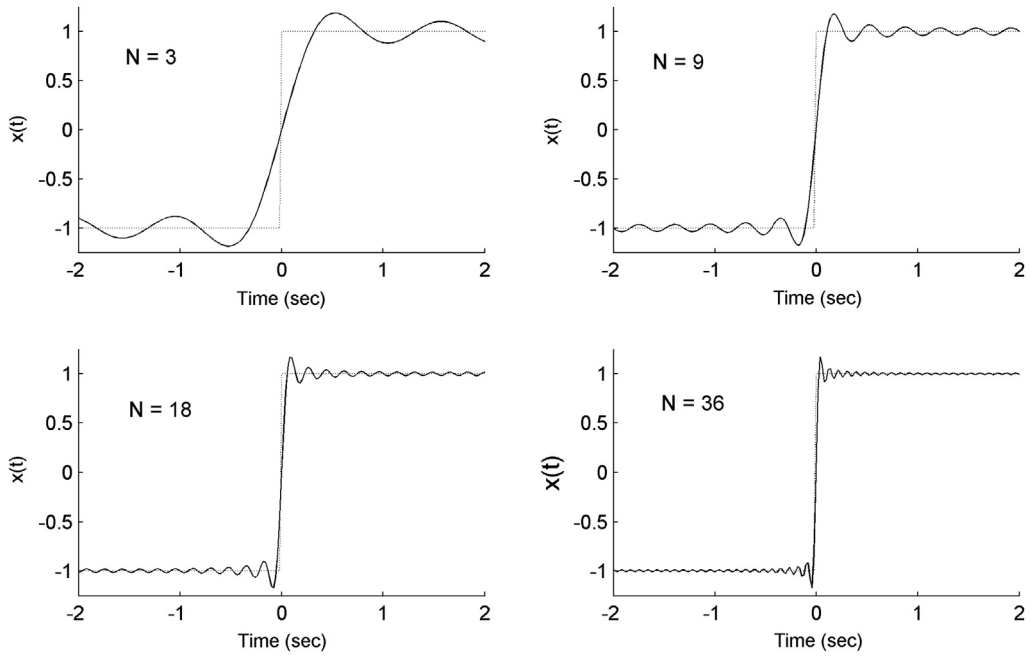


FIGURE 3.5 Reconstruction of a square wave using 3, 9, 18, and 36 sinusoids. The square wave is one of the most difficult signals to represent with a sinusoidal series. The oscillations seen in the sinusoidal approximations are known as "Gibbs artifacts." They increase in frequency, but do not diminish in amplitude, as more sinusoids are added to the summation.

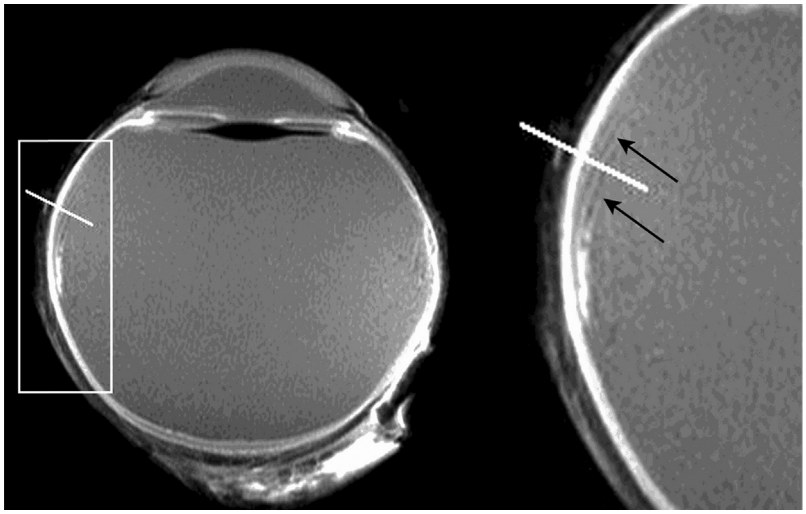


FIGURE 3.6 MR image of the eye showing Gibbs artifacts near the left boundary of the eye. This section is enlarged on the right side. *Image courtesy of Susan and Lawrence Strenk of MRI Research, Inc.*

increase in frequency when more sinusoidal terms are added so the largest overshoot moves closer to the discontinuity. Gibbs artifacts occur in a number of circumstances that involve truncation. They can be found in MR images when there is a sharp transition in the image and the resonance signal is truncated during data acquisition. They are seen as subtle dark ridges adjacent to high contrast boundaries as shown in the MR image of Figure 3.6. We encounter Gibbs artifacts again due to truncation of digital filter coefficients in Chapter 9.

3.3.5 Symmetry

Some waveforms are symmetrical or antisymmetrical about $t = 0$ so that one or the other of the Fourier series coefficients, either the a_m's or b_m's, will be zero. If the waveform has mirror symmetry about $t = 0$, that is, $x(t) = x(-t)$, Figure 3.7 (upper plot), then multiplications with all sine functions will be zero, so the b_m terms will be zero. Such mirror symmetry functions are termed "even functions." Functions with antisymmetry, $x(t) = -x(t)$, Figure 3.7 (middle plot), are "odd functions," and all multiplications with cosines will be zero so the a_m coefficients will be zero. Finally, functions that have half-wave symmetry will have no even coefficients, so both a_m and b_m will be zero for $m =$ even. These are functions where the second half of the period looks like the first half, but inverted, i.e., $x(t - T/2) = -x(t)$, Figure 3.7 (lower plot). Functions having half-wave symmetry are also even functions. These symmetries are useful not only for simplifying the task of solving for the coefficients manually, but also for checking solutions done on a computer. Table 3.1 and Figure 3.7 summarize these properties.

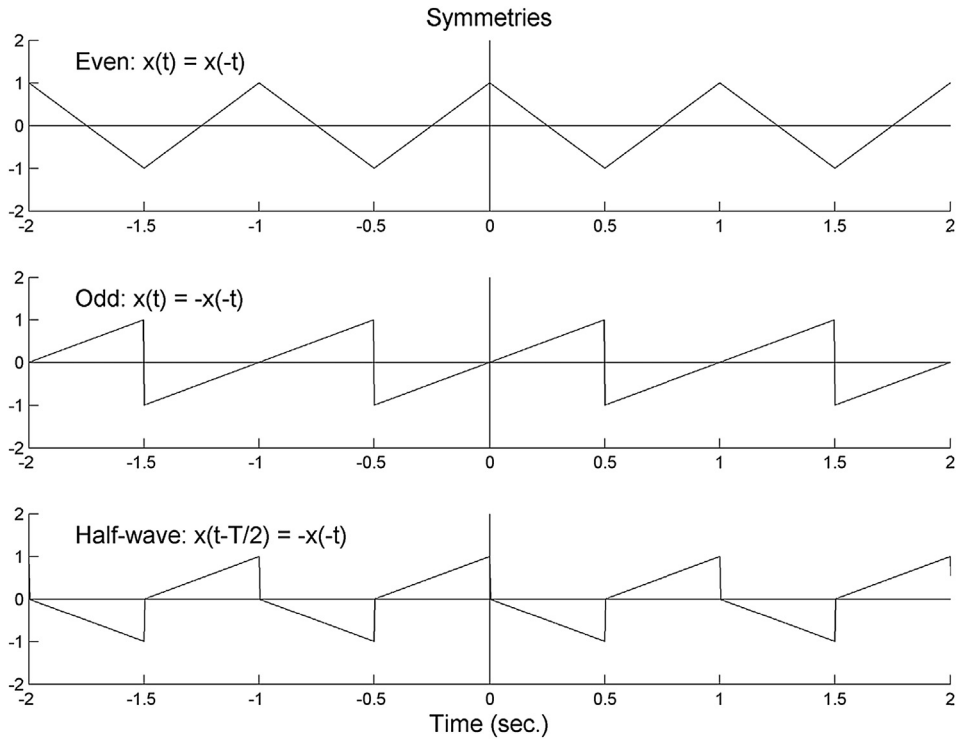


FIGURE 3.7 Waveform symmetries. Each waveform has a period of 1 s. Upper plot: even symmetry; middle plot: odd symmetry; lower plot: half-wave symmetry. Note that the lower waveform also has even symmetry.

TABLE 3.1 Function Symmetries

Function Name	Symmetry	Coefficient Values
Even	$x(t) = x(-t)$	$b_m = 0$
Odd	$x(t) = -x(-t)$	$a_m = 0$
Half-wave	$x(t) = x(t - t/2)$	$a_m = b_m = 0$; for m even

EXAMPLE 3.1

Find the Fourier series of the triangle waveform shown in Figure 3.8. The equation for this waveform is:

$$x(t) = \begin{cases} t & 0 < t \leq 0.5 \\ 0 & 0.5 < t \leq 1.0 \end{cases}$$

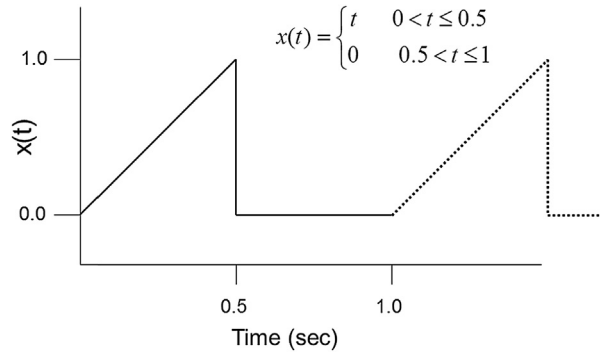


FIGURE 3.8 A half-triangle waveform used in Example 3.1.

Find the first four magnitude and phase components (i.e., $m = 1, 2, 3, 4$). Construct frequency plots of the magnitude components.

Solution: Use Equations 3.8 and 3.9 to find the a_m and b_m coefficients. Then convert to magnitude, C_m, and phase, θ_m, using Equations 3.6 and 3.7. Plot C_m and θ_m against the associated frequency, $f = mf_1$.

Start by evaluating either the sine or cosine coefficients; we begin with the sine coefficients using Equation 3.9:

$$b_m = \frac{2}{T}\int_0^T x(t)\sin(2\pi m f_1 t)dt = \frac{2}{1}\int_0^{0.5} t\sin(2\pi m t)dt$$

$$= \frac{2}{4\pi^2 m^2}[\sin(2\pi m t) - 2\pi m t\cos(2\pi m t)]_0^{0.5}$$

$$= \frac{2}{4\pi^2 m^2}[\sin(\pi m) - \pi m\cos(\pi m)] = \frac{-1}{2\pi^2 m^2}[\cos(\pi m)]$$

$$b_m = \frac{1}{2\pi};\ \frac{-1}{4\pi};\ \frac{1}{6\pi};\ \frac{-1}{8\pi};\ \ldots = 0.159; -0.080; 0.053; -0.040; \ldots$$

To find the cosine coefficients, use Equation 3.8:

$$a_m = \frac{2}{T}\int_0^T x(t)\cos(2\pi m f_1 t)dt = \frac{2}{1}\int_0^{0.5} t\cos(2\pi m t)dt$$

$$= \frac{2}{4\pi^2 m^2}[\cos(2\pi m t) - 2\pi m t\sin(2\pi m t)]_0^{0.5}$$

$$= \frac{2}{4\pi^2 m^2}[\cos(\pi m) - \pi m\sin(\pi m) - 1] = \frac{-1}{2\pi^2 m^2}[\cos(\pi m) - 1]$$

$$a_m = \frac{-1}{\pi^2};\ 0;\ \frac{-1}{9\pi^2};\ 0;\ \ldots = -0.101; 0; -0.0118; 0; \ldots$$

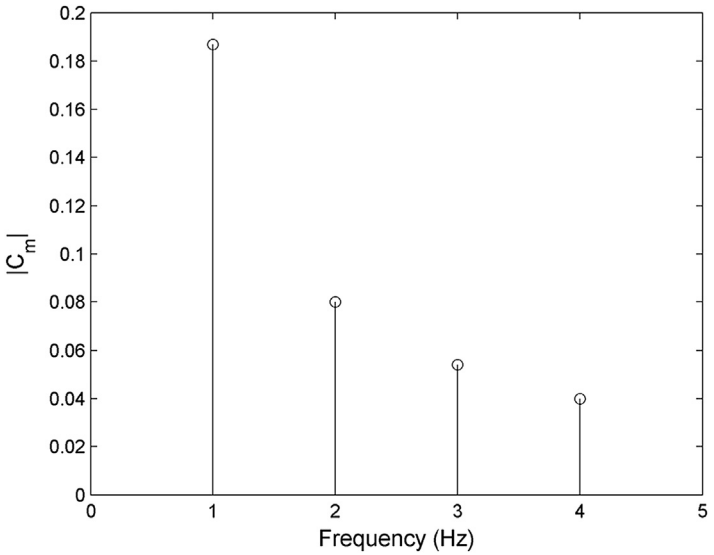


FIGURE 3.9 Magnitude plot (*circles*) of the frequency characteristics of the waveform given in Example 3.1. Only the first four components are shown because these are the only ones calculated in the example. These components were determined by taking the square root of the sum of squares of the first four cosine and sine terms (a's and b's).

To express the Fourier coefficients in terms of magnitude, C_m, and phase, θ_m, use Equations 3.6 and 3.7. Care must be taken in computing the phase angle to ensure that it represents the proper quadrant[10]:

$$C_m = \sqrt{a_m^2 + b_m^2} = 0.187, 0.080, 0.054, 0.040$$

$$\theta_m = \tan^{-1}\left(\frac{-b_m}{a_m}\right) = -122.42, 90.00, -101.72, 90.00 \text{ degrees}$$

For a complete description of $x(t)$ we need more components and also the a_0 term.
The a_0 term is twice the average value of $x(t)$, which can be obtained using Equation 3.10:

$$C_0 = a_0 = \frac{2}{T}\int_0^T x(t)dt = \frac{2}{1}\int_0^{0.5} tdt = \frac{2t^2}{2}\Big|_0^{0.5} = 0.25$$

$$\frac{a_0}{2} = \bar{x}(t) = \frac{1}{T}\int_0^T x(t)dt = \frac{1}{1}\int_0^5 tdt = \frac{t^2}{2}\Big|_0^5 = .125$$

To plot the spectrum of this signal, or rather a partial spectrum, we plot the sinusoid coefficients, C_m and θ_m, against frequency. Since the period of the signal is 1.0 second, the fundamental frequency, f_1, is 1 Hz. Therefore the first four values of m represent 1, 2, 3, and 4 Hz. The magnitude plot is shown in Figure 3.9.

[10]The arctangent function on some calculators does not take into account the signs of b_m an a_m in which case it is up to you to figure out the proper quadrant. This is also true of MATLAB's atan function. However, MATLAB does have a function, atan2(b,a), that takes the signs of b and a into account and produces an angle (in radians) in the proper quadrant.

The four components computed in Example 3.1 are far short of that necessary for a reasonable frequency representation, but when we implement Fourier series analysis on a computer we solve for hundreds, or even thousands, of components. We can check on how good the representation is by reconstructing the signal using only these four components. We do that in the next example with the help of MATLAB.

EXAMPLE 3.2

Reconstruct the waveform of Example 3.1 using the four components found in that example. Use the polar representation (i.e., magnitude and phase) of the Fourier series equation, Equation 3.3, to reconstruct the signal and plot the time domain reconstruction.

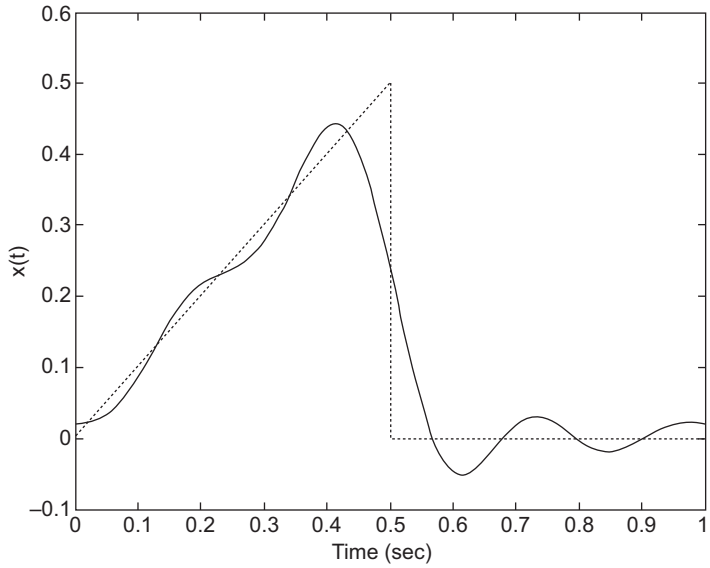


FIGURE 3.10 The waveform produced by applying the magnitude and phase version of the Fourier series equation (Equation 3.3) to the four magnitude and phase components found analytically in Example 3.1. The Fourier summation (*solid line*) produces a rough approximation of the original waveform (*dotted line*), even though only four components are used.

Solution: Apply Equation 3.3 directly using the four magnitude and phase components found in the last example. Remember to add the DC components, a_0. Plot the result of the summation.

```
% Example 3.1 Reconstruct the waveform of Example 3.1
fs = 500;                                   % Assumed sample frequency
N = 500;                                    % Number of points for 1 sec.
t = (1:N)/N;
C = [0.187 0.08 0.054 0.04];                % Component magnitudes
theta = [-122 90 -101 90]*2*pi/360;         % Component phase (in radians)
```

```
%
x = zeros(1,N);
for f = 1:4                                  % Add the 4 terms of Equation 3.3
  x = x + C(f)*cos(2*pi*f*t + theta(f));     %  using appropriate A and theta
end
x = x + 0.125;                               % Add the DC term
plot(t,x,'k');                               % Plot the result
    .......labels.......
```

The result produced by this program is shown in Figure 3.10, which shows it to be a rough approximation of the original signal.

Fourier series analysis is not the only way to transform a time signal into the frequency domain, but it is the most general approach and the most often used. Other approaches require you to make some assumptions about the signal. The digital version of Equations 3.8 and 3.9 can be calculated with great speed using an algorithm known as the "fast Fourier transform," abbreviated "FFT." There is also an "inverse fast Fourier transform," abbreviated "IFFT," which implements the synthesis equation (Equation 3.3 or 3.5).

The resolution of a spectrum can be loosely defined as the difference in frequencies that a spectrum can resolve: that is, how close two signal frequencies can get and still be identified as two frequencies in the frequency domain. This resolution clearly depends on the frequency spacing between harmonic numbers, which in turn is equal to $1/T$ (Equation 3.4). Hence the longer the signal period, the better the spectral resolution. Later we find this holds for digital data as well.

3.3.6 Complex Representation

It is possible to rewrite the Fourier analysis and synthesis equations using the complex representation of the sinusoid as given by Euler's identity (Equation 2.28). You might wonder why we would do this given that we have a perfectly good set of equations for moving between the time and frequency domains. There are actually four reasons: (1) the complex equations are more succinct and there is only one analysis equation; (2) in a few cases the complex equations are easier to solve analytically; (3) computer algorithms use the complex equations; and (4) this is the way you will see the Fourier transform equations written in research papers.

The complex representation of the Fourier series analysis can be derived directly from Equation 3.5 using only algebra and Euler's identity (Equation 2.28). We start with the exponential definitions of the sine and cosine functions which come directly from Euler's identity:

$$\begin{aligned} \cos(2\pi m f_1 t) &= \frac{1}{2}\left(e^{+j2\pi m f_1 t} + e^{-j2\pi m f_1 t}\right) \text{ and} \\ \sin(2\pi m f_1 t) &= \frac{1}{j2}\left(e^{+j2\pi m f_1 t} - e^{-j2\pi m f_1 t}\right) \end{aligned} \tag{3.15}$$

Recall the Fourier series synthesis equation (Equation 3.5) repeated here:

$$x(t) = \frac{a_0}{2} + \sum_{m=1}^{\infty} a_m \cos(2\pi m f_1 t) + \sum_{m=1}^{\infty} b_m \sin(2\pi m f_1 t)$$

Substituting the complex definitions for the sine and cosines into this equation and expanding:

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(\frac{a_m}{2}e^{j2\pi m f_1 t} + \frac{a_m}{2}e^{-j2\pi m f_1 t} + \frac{b_m}{2j}e^{j2\pi m f_1 t} - \frac{b_m}{2j}e^{-j2\pi m f_1 t}\right)$$

Moving the j term to the numerator using the fact that $1/j = -j$:

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(\frac{a_m}{2}e^{j2\pi m f_1 t} + \frac{a_m}{2}e^{-j2\pi m f_1 t} - \frac{jb_m}{2}e^{j2\pi m f_1 t} + \frac{jb_m}{2}e^{-j2\pi m f_1 t}\right)$$

Rearranging:

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(\left(\frac{a_m}{2} - \frac{jb_m}{2}\right)e^{j2\pi m f_1 t} + \left(\frac{a_m}{2} + \frac{jb_m}{2}\right)e^{-j2\pi m f_1 t}\right)$$

Collecting terms into positive and negative summations of m:

$$x(t) = \frac{a_0}{2} + \sum_{m=1}^{\infty}\left(\frac{a_m - jb_m}{2}\right)e^{j2\pi m f_1 t} + \sum_{m=-\infty}^{-1}\left(\frac{a_{-m} + jb_{-m}}{2}\right)e^{j2\pi m f_1} \tag{3.16}$$

This can be combined into a single equation going from negative to positive infinity:

$$x(t) = \sum_{m=-\infty}^{m=\infty} X_m e^{j2\pi m f_1 t} \tag{3.17}$$

where the new coefficient, X_m is defined as:

$$X_{+m} = \frac{a_m - jb_m}{2};\ X_{-m} = \frac{a_m + jb_m}{2};\ X_0 = a_0 \tag{3.18}$$

Equation 3.17 is the complex Fourier series synthesis equation. Note the DC term, a_0, is included in the complex number series X_m, which also incorporates both the a_m and b_m coefficients.

To find the Fourier series analysis equation, substitute the original correlation equations for a_m and b_m (Equations 3.8 and 3.9) into Equation 3.18:

$$X_m = \frac{2}{2T}\int_0^T x(t)\cos(2\pi m f_1 t)dt - \frac{j2}{2T}\int_0^T x(t)\sin(2\pi m f_1 t)dt$$

Combining:

$$X_m = \frac{1}{T}\int_0^T x(t)[\cos(2\pi m f_1 t) - j\sin(2\pi m f_1 t)]dt \tag{3.19}$$

The term in the brackets in Equation 3.19 has the form $\cos(x) - j\sin(x)$. This is equal to e^{-jx} as given by Euler's identity (Equation 2.28). Hence the term in the brackets can be replaced by a single exponential, which leads to the complex form of the Fourier series:

$$X_m = \frac{1}{T}\int_0^T x(t)e^{-j2\pi m f_1 t}\,dt \quad m = 0, \pm1, \pm2, \pm3, \ldots \tag{3.20}$$

Alternatively, since $f_1 = 1/T$

$$X_m = \frac{1}{T}\int_0^T x(t)e^{\frac{-j2\pi mt}{T}}dt \quad m = 0, \pm1, \pm2, \pm3, \ldots \tag{3.21}$$

This is the most common representation of Fourier series analysis commonly called the Fourier transform. It is also the equation you will see whenever this transform is used and referenced in a research paper. In the complex representation, the harmonic number m must range from minus to plus infinity as in Equation 3.20, essentially to account for the negative signs introduced by the complex definition of sines and cosines (Equation 3.15). However, the $a_0/2$ term (the average or DC value) does not require a separate equation, since when $m = 0$ the exponential becomes: $e^{-j0} = 1$, and the integral computes the average value. Finally, only one analysis equation is needed, as the a_m and b_m coefficients are embedded in X_m as detailed in the following discussion.

The negative frequencies implied by the negative m terms are often dismissed as mathematical contrivances since negative frequency has no meaning in the real world. However, when a signal is digitized these negative frequencies do have consequences as shown later. In fact, C_m is symmetrical about zero frequency so the negative frequency components are just mirror images of the positive components. Since they are the same components just in reverse order, they do not provide additional information, but they do double the summations, so the normalization is now $1/T$, not the $2/T$ used for the noncomplex analysis equations. (Recall that there are different normalization strategies and, although MATLAB uses the complex form to compute the Fourier transform, it does not normalize the summation. You are expected to apply whatever normalization you wish.)

The complex variable C_m combines the cosine and sine coefficients. To get the a_m's and b_m's from C_m just double the real and imaginary parts:

$$a_m = 2\mathrm{Re}(X_m); \quad b_m = 2\mathrm{Im}(X_m) \tag{3.22}$$

The magnitude C_m and phase θ_m representation of the Fourier series can be found from C_m using complex algebra:

$$|X_m| = \sqrt{\mathrm{Re}(X_m)^2 + \mathrm{Im}(X_m)^2} = \sqrt{\frac{a_m^2 + b_m^2}{2^2}} = \frac{1}{2}\sqrt{a_m^2 + b_m^2} = 0.5C_m \tag{3.23}$$

$$\text{Angle}(C_m) = \tan^{-1}\left(\frac{\text{Im}(X_m)}{\text{Re}(X_m)}\right) = \tan^{-1}\left(\frac{\frac{-b_m}{2}}{\frac{a_m}{2}}\right) = \tan^{-1}\left(\frac{-b_m}{a_m}\right) = \theta \tag{3.24}$$

The magnitude of X_m is equal to $1/2\ C_m$, the magnitude of the sinusoidal components, and the angle of C_m is equal to the phase, θ_m, of the sinusoidal components. Not only does X_m contain both sine and cosine coefficients, but these components can also readily be obtained in either the polar (X's and θ's, most useful for plotting) or the rectangular form (a's and b's). (Remember, the factors of 2 in Equation 3.22 and 0.5 in Equation 3.23 are based on the normalization strategy presented here.) Finally, X_0 is just the DC term (i.e., C_0 or a_0) so it must be real (the b_0 equivalent is zero).

EXAMPLE 3.3

(A) Find the Fourier series of the pulse waveform shown in Figure 3.11 with a period T, and amplitude V_p, and a pulse width of W. Use Equation 3.20, the complex equation. (B) Use MATLAB to plot the solution for $T = 1$ s; $V_p = 1.0$ and two values of pulse width: $W = 0.005$ and $W = 0.005$ s. Plot the first ± 100 harmonics.

(A) Solution: Apply Equation 3.20 directly, except since the signal is an even function ($x(t) = x(-t)$), it is easier to integrate from $t = -T/2$ to $+T/2$.

$$X_m = \frac{1}{T}\int_{-T/2}^{T/2} x(t)e^{-j2\pi m f_1 t}\,dt = \frac{1}{T}\int_{-W/2}^{W/2} V_p e^{-j2\pi m f_1 t}dt$$

$$= \frac{V_p}{T(-j2\pi m f_1)}\left[e^{-j2\pi m f_1 W/2} - e^{j2\pi m f_1 W/2}\right] = \left[\frac{V_p}{\pi m f_1 T}\right]\frac{\left[e^{j2\pi m f_1 W/2} - e^{-j2\pi m f_1 W/2}\right]}{2j}$$

where the second term in brackets is $\sin(2\pi m f_1 W/2) = \sin(\pi m f_1 W)$. So the expression for C_m becomes:

$$X_m = \frac{V_p}{\pi m f_1 T}\sin(\pi m f_1 W) \quad m = 0, \pm 1, \pm 2, \pm 3, \ldots$$

It is common to rearrange this equation in the form $\frac{\sin(x)}{x}$.

$$X_m = \left(\frac{V_p W}{T}\right)\frac{\sin(\pi m f_1 W)}{\pi m f_1 W} \quad m = 0, \pm 1, \pm 2, \pm 3, \ldots$$

(B) Solution MATLAB: Although this particular function of X_m is real and does not have an imaginary component, it still represents both magnitude and phase components. Plotting these components in MATLAB is straightforward. We enter the values given for T, V_p, and W. We set the harmonic number, m, to range between ± 100. Three MATLAB functions are then used to simplify the code: the `sinc` function to evaluate $\sin(\pi x)/\pi x$[11]; the `abs` (absolute value) function to get the

[11]The function $\sin(x)/x$ is also known as $\text{sinc}(x)$. The MATLAB function includes π in the numerator and dominator, i.e., in MATLAB $\text{sinc}(x) = \sin(\pi x)/\pi x$.

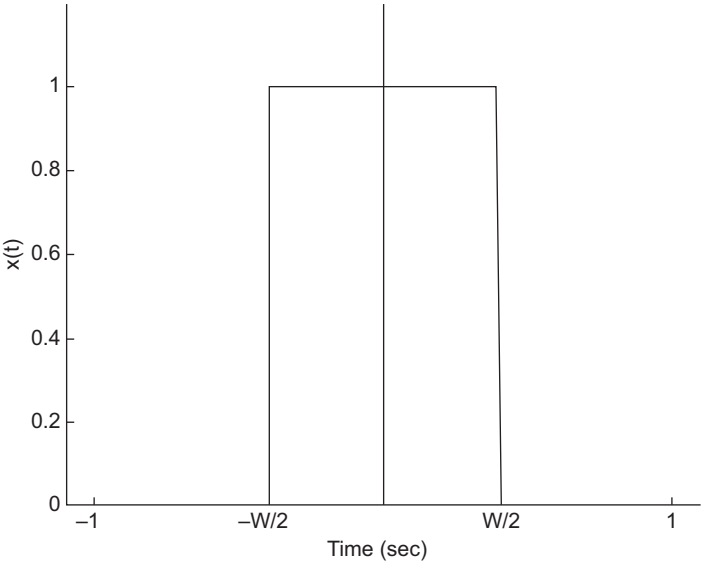


FIGURE 3.11 Pulse wave used in Example 3.3.

magnitude of C_m; and the `angle` function to get the phase of X_m. MATLAB always gives angles in radians so we convert to degrees by multiplying the angle by $360/2\pi$.

```
%  Example 3.3 Plot the magnitude spectrum of two pulse waveforms
%  found from the complex form of the Fourier series analysis in Example 3.3
%
T = 1.0;                % Period (sec)
f1 = 1/T;               % Pulse wave fundamental frequency
Vp = 1.0;               % Pulse amplitude, Vp
W = [0.01 0.05];        % Pulse widths
m = -100:100;           % Harmonic numbers
%
for k = 1:length(W);
  Xm = ((Vp*W(k))/T)*sinc(m*f1*W(k));      % Solution eq.
  Mag = abs(Xm);                    % Get magnitude spectrum
  Phase = angle(Xm)*360/(2*pi);     % Get phase spectrum (in deg)
  subplot(2,2,2*k-1);  hold on
  plot(m,Mag,'.k');                 % Plot magnitude spectrum
  .....labels, title, and zero frequency line.......
  subplot(2,2,2*k);  hold on;
  plot(m,Phase,'.k');               % Plot the phase spectrum
  .....labels, title,  and zero frequency line.......
end
```

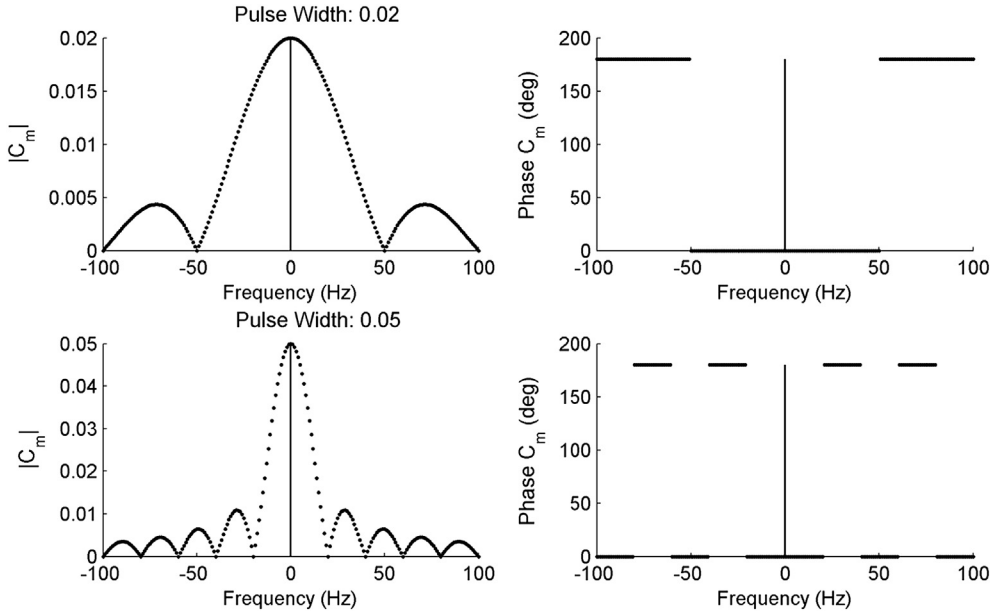


FIGURE 3.12 The frequency spectra of two pulse functions having different pulse widths found using the complex Fourier series analysis. The spectra consist of individual points spaced $f_1 = 1/T = 1.0$ Hz apart. The longer pulse produces a spectrum with a shorter peak (bottom figures). Both spectra have the shape of $|\sin(x)/x| \equiv |\text{sinc}(x)|$. When $\sin(x)$ switches between positive and negative, the phase spectra alternate between 0 and 180 degrees. Note the inverse relationship between pulse width and the width of the spectrum: the narrower the pulse the broader the spectrum and vice versa.

Result: The magnitude and phase spectra of the two pulses are shown Figure 3.12. These spectra are discrete consisting of individual points spaced $1/T$ or 1.0 Hz apart. Both curves take the shape of the magnitude of function $\sin(x)/x$, also known as $\text{sinc}(x)$. Note the inverse relationship between pulse width and spectrum width: a wider time domain pulse produces a narrower frequency domain spectrum. This inverse relationship is typical of time–frequency transformations. Later we will see that in the limit, as the pulse width $\rightarrow 0$, the spectrum becomes infinitely broad, i.e., a flat horizontal line.

EXAMPLE 3.4

Find the Fourier series of the waveform shown in Figure 3.13 using the complex form of the equation. The waveform is periodic and can be described over one period as: $x(t) = e^{-2.4t} - 1$.

Solution: Apply Equation 3.20 directly, noting that the equation for the waveform is:

$$x(t) = e^{2.4t} - 1 \quad \text{for } 0 < t \leq 1 \text{ and } f_1 = 1/T = 1.$$

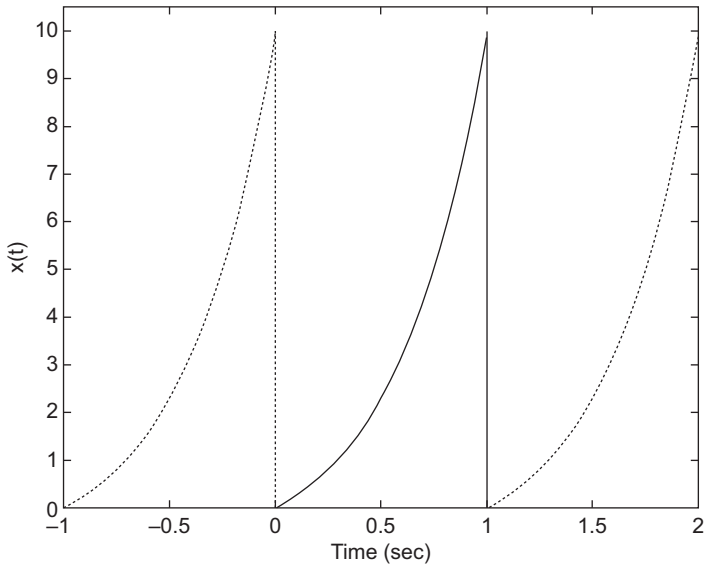


FIGURE 3.13 Periodic waveform used in Example 3.4.

$$X_m = \frac{1}{T}\int_0^T x(t)\mathrm{e}^{-j2\pi m f_1 t}dt = \frac{1}{T}\int_0^T \left(\mathrm{e}^{2.4t}-1\right)\mathrm{e}^{-j2\pi m f_1 t}dt$$

$$X_m = \int_0^1 \left(\mathrm{e}^{2.4t}\mathrm{e}^{-j2\pi mt}-\mathrm{e}^{-j2\pi mt}\right)dt = \int_0^1 \mathrm{e}^{t(2.4-j2\pi m)}dt - \int_0^1 \mathrm{e}^{-j2\pi mt}dt$$

$$X_m = \left.\frac{\mathrm{e}^{t(2.4-j2\pi m)}}{2.4-j2\pi m}\right|_0^1 - \left.\frac{\mathrm{e}^{j2\pi m}}{-j2\pi m}\right|_0^1$$

$$X_m = \frac{\mathrm{e}^{2.4-j2\pi m}-1}{2.4-j2\pi m} - \frac{\mathrm{e}^{-j2\pi m}-1}{-j2\pi m}$$

If we use Euler's identity to substitute in cosine and sine terms for $e^{-j2\pi m}$ ($e^{-j2\pi m} = \cos(2\pi m) + j\sin(2\pi m)$), we see that the sine term is always zero and the cosine term always 1 if m is an integer or zero. Hence, $e^{-j2\pi m} = 1$ for $m = 0, \pm1, \pm2, \pm3,\ldots$. This gives:

$$X_m = \frac{e^{2.4}e^{-j2\pi m}}{2.4-j2\pi m} - \frac{e^{-j2\pi m}-1}{-j2\pi m} = \frac{e^{2.4}-1}{2.4-j2\pi m} - \frac{1-1}{-j2\pi m} = \frac{e^{2.4}-1}{2.4-j2\pi m} \quad m = 0, \pm1, \pm2, \pm3, \ldots$$

The problem set contains additional applications of the complex Fourier series equation.

3.3.7 The Continuous Fourier Transform

The Fourier series analysis is a good approach to determining the frequency or spectral characteristics of a periodic waveform, but what if the signal is not periodic? Most real signals are not periodic, and for many physiological signals, such as the EEG signal introduced in the first

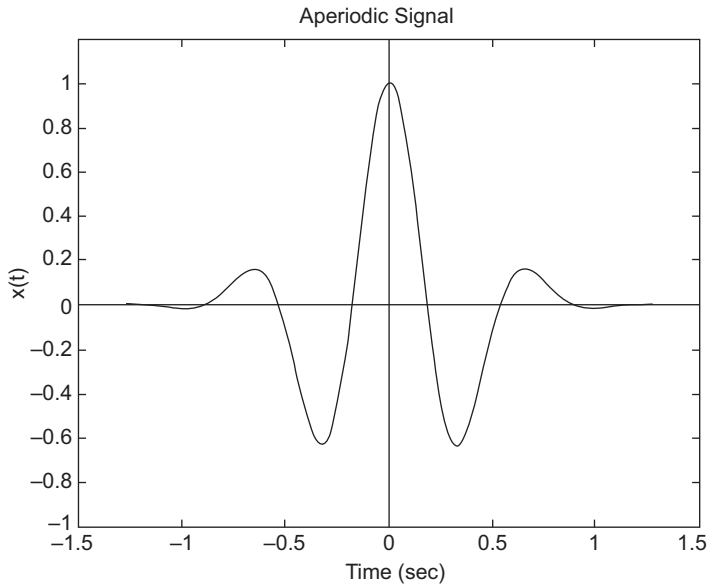


FIGURE 3.14 An aperiodic function exists for a finite length of time and is zero everywhere else. Unlike a periodic sine wave, you are seeing the complete signal here; this is all there is!

chapter, only a part of the signal is available. The segment of EEG signal shown previously is just a segment of the actual recording, but no matter how long the record, the EEG signal existed before the recording and will continue after the recording session ends (unless the EEG recording session was so traumatic as to cause untimely death!). Dealing with very long signals generally entails estimations or approximations, but if the signal is continuous and aperiodic, an extension of the Fourier series analysis can be used. An aperiodic signal is one that exists for a finite period of time and is zero at all other times, Figure 3.14.

To extend the Fourier series analysis to aperiodic signals, these signals are treated as periodic, but with a period that goes to infinity, (i.e., $T \rightarrow \infty$). If the period becomes infinite, then $f_1 = 1/T \rightarrow 0$; however, mf_1 does not go to zero since m goes to infinity. This is a case of limits: as T gets longer and longer, f_1 becomes smaller and smaller as does the increment between harmonics, Figure 3.15. In the limit, the frequency increment mf_1 becomes a continuous variable f. All the various Fourier series equations described above remain pretty much the same for aperiodic functions, only the $2\pi mf_1$ goes to $2\pi f$. If radians are used instead of Hz, then $m\omega_1$ goes to ω in these equations. The equation for the continuous Fourier transform in complex form is:

$$\lim_{\substack{T\rightarrow\infty \\ f\rightarrow 0}} X_m = \int_0^T x(t)e^{-j2\pi mf_1 t}dt = \int_{-\infty}^{\infty} x(t)e^{-2\pi ft}dt$$

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-2\pi ft}dt \text{ or } X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-\omega t}dt \tag{3.25}$$

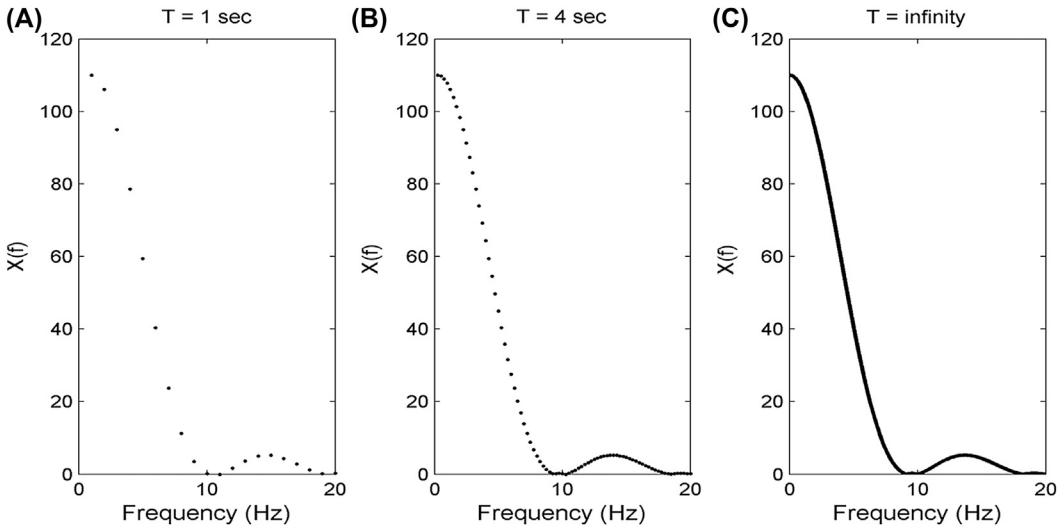


FIGURE 3.15 The effect of increasing the period, T, on the frequency spectrum of a signal. (A) When the period is relatively short (1 s), the spectral points are widely spaced at 1 Hz. (B) When the period becomes longer (4 s), the points are spaced closer together at intervals of 0.25 Hz. (C) When the period becomes infinite, the points become infinitely close together and what was a discrete set of points becomes a continuous curve.

Or for the noncomplex equations in terms of the sine and cosine:

$$a(f) = \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt \text{ or } a(\omega) = \int_{-\infty}^{\infty} x(t)\cos(\omega t)dt$$

$$b(f) = \int_{-\infty}^{\infty} x(t)\sin(2\pi ft)dt \text{ or } b(\omega) = \int_{-\infty}^{\infty} x(t)\sin(\omega t)dt \tag{3.26}$$

These transforms produce a continuous function as an output and it is common to denote these terms with capital letters. Also the transform equation is no longer normalized by the period since $1/T \to 0$. Although the transform equation is theoretically integrated over times between $\pm\infty$, the actual limits will only be over the nonzero values of $x(t)$.

Although it is rarely used in analytical computations, the inverse continuous Fourier Transform is given in the following complex format. This version is given in radians, ω, instead of $2\pi f$ (just for variety):

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega \tag{3.27}$$

where ω is frequency in radians.

EXAMPLE 3.5

Find the Fourier transform of the pulse in Example 3.3 assuming the period, T, goes to infinity.

Solution: We could apply Equation 3.20, but since $x(t)$ is an even function we could also use only the cosine equation of the Fourier transform in Equation 3.26 knowing that all of the sine terms would be zero for the reasons given in Section 3.3.5 on symmetry.

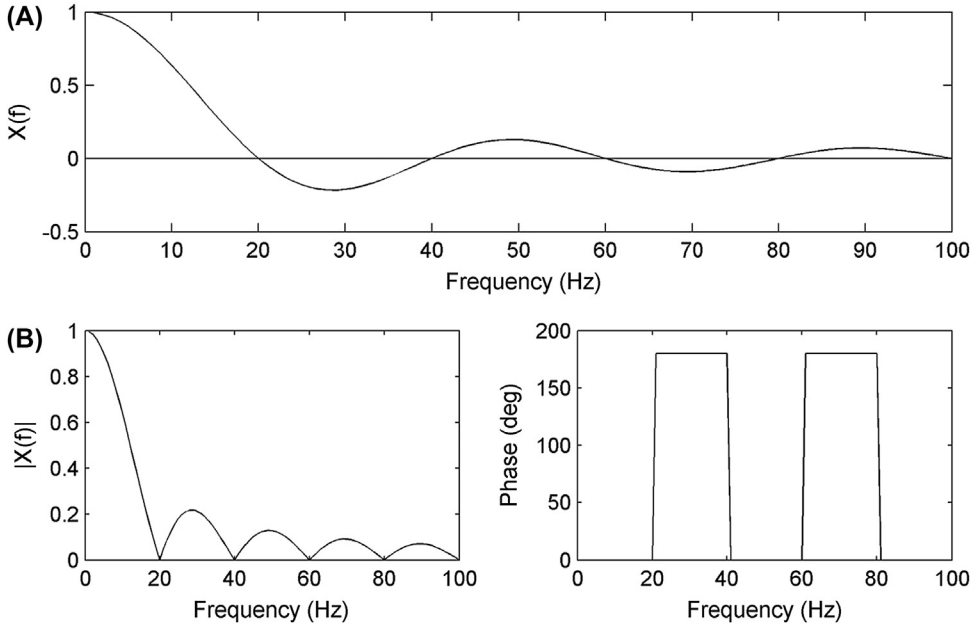


FIGURE 3.16 (A) The complex spectrum of an aperiodic pulse as determined in Example 3.5. $W = 0.05$ s. The complex function $X(f)$ is shown in the upper plot. (B) The magnitude and phase are shown in the lower plots. The vertical scale is not the same as in Figure 3.12 because the spectrum cannot be normalized by $1/T$ since $T \to \infty$.

$$X(f) = \int_{-\infty}^{\infty} x(t)\cos(2\pi ft)dt = \int_{-W/2}^{W/2} V_p \cos(2\pi ft)dt = \frac{V_p}{2\pi f}\sin(2\pi f)\Big|_{t=-W/2}^{t=W/2}$$

$$X(f) = \frac{V_p}{2\pi f}(\sin(2\pi f\, W/2) - \sin(-2\pi f\, W/2)) = \frac{V_p}{2\pi f}(\sin(\pi f\, W) + \sin(\pi f\, W)) \tag{3.28}$$

$$X(f) = \frac{V_p}{2\pi f}(2\sin(\pi f\, W)) = \frac{V_p \sin(\pi f\, W)}{\pi f}$$

Result: A plot of $X(f)$ is shown in Figure 3.16A. Note that the solution is similar to the solution found for a periodic pulse wave in Example 3.3. Again this solution is real, but it still represents both magnitude and phase components. The magnitude is the absolute value of $X(f)$, Figure 3.16B, whereas the phase alternates between 0 degree when the function is positive and 180 degrees when the function is negative.

A special type of pulse plays an important role in systems analysis: a pulse that is very short but maintains a pulse area of 1.0. For this special pulse, termed an "impulse function," the pulse width, W, theoretically goes to 0 but the amplitude of the pulse becomes infinite in such a way that the area of the pulse stays at 1.0. The spectrum of an impulse can be determined from Equation 3.28. For small angles $\sin(x) \approx x$, so in the solution above, as W gets small:

$$\sin(\pi f\, W) \rightarrow \pi f\, W$$

The frequency spectrum of this special pulse becomes:

$$X(f) = \frac{V_p}{\pi f}\pi f\, W = V_p W = 1 \tag{3.29}$$

since V_pW is the area under the pulse and by definition the area of an impulse function equals 1.0. So the frequency spectrum of this special pulse, the impulse function, is a constant value of 1.0 for all frequencies. You might imagine that a function with this spectrum would be particularly useful. You might also speculate that a pulse with these special properties, infinitely short, infinitely high, yet still with an area of 1.0, could not be realized, and you would be right. However, it is possible to generate a real-world pulse that is short enough and high enough to function as an impulse at least for all practical purposes. The approach for generating a real-world impulse function in particular situations is provided in Chapter 7.

3.4 TIME–FREQUENCY TRANSFORMATION IN THE DISCRETE DOMAIN

3.4.1 The Discrete Fourier Transform

Most Fourier analysis is done using a digital computer and is applied to discrete data. Although there are differences between the continuous and discrete Fourier transforms (DFTs), the basic concepts are the same: they both use correlation with sinusoids and their unique frequency properties to transfer signals from the discrete time to discrete frequency domain. In fact, if you applied the continuous analysis and synthesis blindly (replacing summation with integration of course), you would get similar results. However, there are a few important differences: the Fourier series is always finite, and there are some tricks to improve the efficiency and speed.

Discrete signals differ from continuous signals in two fundamental ways: they are time and amplitude sampled as discussed in Chapter 1, and they are always finite. The discrete version of the Fourier analysis equation is termed the "discrete time Fourier series" or, more commonly, the "DFT." The discrete Fourier synthesis equation is known as the "inverse discrete Fourier series" or the "inverse discrete Fourier transform" (IDFT).

The DFT analysis equation can be derived directly from Equation 3.21 noting that integration becomes summation and the time variables t and T become:

$$t = nf_s \text{ and } T = Nf_s$$

If we ignore the normalization by $1/T$, this gives the un-normalized analysis equation:

$$X[m] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi mnf_s}{Nf_s}} = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi mn}{N}} \quad m = 0, \pm 1, \pm 2, \pm 3 \ldots \pm M \tag{3.30}$$

where n is the index of the signal array, N is the array length, and m is the harmonic number. The number of harmonics evaluated, M, must be less than or equal to $N - 1$. In the most common implementation of this equation, $M = N - 1$. Although no normalization is used in Equation 3.30, it is common to normalize the summation by $1/N$, the equivalent of $1/T$. (Occasionally when we compare results between the complex (Equation 3.30) and noncomplex equations (Equations 3.8 and 3.9), we normalize by $2/N$ to match the noncomplex normalization strategy.) The equation for the IDFT is quite similar and is given in Table 3.3, Equation 3.37.

As with the continuous version of the Fourier series, Equation 3.30 produces a series of complex numbers that describe the amplitude and phase of a harmonic series of sinusoids. To relate the sinusoidal frequencies to the original analog signal, note that if a periodic signal of period T is digitized at a sample frequency of f_s into N samples, then T is equivalent to N/f_s. The fundamental frequency becomes:

$$f_1 = \frac{1}{T} = \frac{f_s}{N} \tag{3.31}$$

So the component frequencies when related to the original signal are:

$$f = mf_1 = \frac{mf_s}{N} = \frac{m}{T} \tag{3.32}$$

In the next example, we apply Equation 3.30 to a signal to obtain the magnitude spectra. Since we are using the computer to calculate the coefficients we can make the signal as complicated as we want, the computer does not care. Nevertheless, we fall back on the pulse wave used in Example 3.3. Since the phase spectrum is not very interesting we only plot the magnitude spectrum, again for ±100 components. In addition, we normalize the coefficients by $1/N$ to match that used in Equation 3.20.

EXAMPLE 3.6

Construct the pulse wave signal similar to that used in Example 3.3 with $T = 1$ s, $V_p = 1.0$, and $W = 0.05$. Assume $f_s = 1$ kHz and use $N = 1000$ points so the period of the waveform is 1.0 s. Since the transform is symmetrical about $f = 0$, calculate only positive coefficients using Equation 3.30 ($m = 0, 1, 2, 3 \ldots 100$). Plot only the first 100 components of the magnitude spectrum for comparison with the magnitude spectrum in Figure 3.12. Scale the coefficients by $1/N$ to match the analytical coefficients. Finally, ensure that the horizontal axis has the correct frequencies.

Solution: Generate the waveform by constructing an array of zeros ($N = 1000$ for a 1.0 s period) then make the first W^*f_s points equal to 1.0 to produce a pulse of width W. Although this pulse waveform is shifted with respect to the one in Figure 3.11, as shown later in this chapter, such a time

shift does not affect the magnitude spectrum, only the phase spectrum (and we are not asked to plot the phase spectrum). We normalize the coefficient by $1/N$ to match the scaling used in the analytical equation (Equation 3.20). To get the correct frequencies, we generate a frequency vector $f = mf_1 = mf_s/M$.

```
% Example 3.6 Find and plot the magnitude spectrum of a pulse waveform
% using the complex form of the Fourier series analysis (Equation 3.30)
%
fs = 1000;              % Sample frequency
N = 1000;               % Data array length for a 1.0 sec period.
W = 0.05;               % Pulse width
M = 100;                % Number of coefficients to calculate
x = zeros(1,N);         % Generate pulse waveform
PW = round(W*fs);       % Number of points in pulse
x(1:PW) = 1;            % Set pulse
%
% Apply complex Fourier series analysis
for m = 0:M-1
% Add 1 to m when used as an index
  Xf = sum(x.*exp(-j*2*pi*m(1:N)/N));  % Find complex coefficients, Equation 3.30
  Mag(m+1) = abs(Xf)/N;         % Find magnitude, Equation 3.23, and normalize
  f(m+1) = m*fs/N;              % Generate frequency vector, Equation 3.32
end
plot(f,Mag,'k.');               % Plot results
```

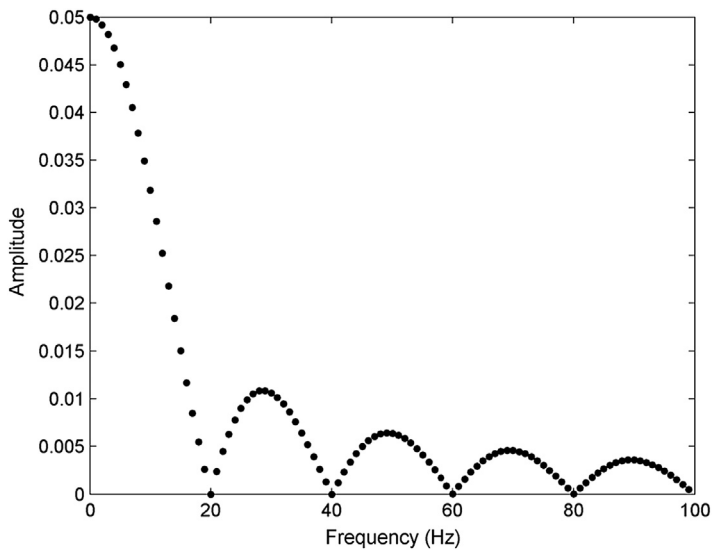


FIGURE 3.17 The spectrum generated by applying the discrete Fourier transform to a pulse waveform similar to that used in Example 3.3. The magnitude spectrum is identical to that found analytically and plotted in Figure 3.12.

Results: The magnitude spectrum generated by the program is shown in Figure 3.17 to match the one found analytically and shown in Figure 3.12. Note that our frequency vector correctly scales the frequency axis.

Example 3.6 shows how easy it is to determine a signal's spectrum using the discrete Fourier series equation, Equation 3.30. The same code could be used to find the magnitude spectrum of the most complex signal. However, in practice the DFT is normally implemented using the FFT.

Again, the underlying assumption of the discrete Fourier series is that the digitized signal represents one period of a periodic function. This is rarely the case in real situations and the assumption does produce some artifacts, particularly if the data segment is short. These artifacts and their potential remediation are discussed in the next chapter.

Application of Equation 3.30 produces a series of frequency components spaced $1/T$ apart, Equation 3.32. After the Fourier series is calculated, we sometimes pretend that the data set in the computer is actually an aperiodic function, that the original (undigitized) signal was zero for all time except for the segment captured in the computer. Under this aperiodic assumption, we can legitimately connect the points of the Fourier series together when plotting, since the difference between points would go to zero if the period was really infinite. This produces an apparently continuous curve and motivates the term digital Fourier transform (DFT) instead of digital Fourier series. Although the aperiodic assumption and associated term DFT are commonly used, we should understand that whenever we do a Fourier transform on a computer, we are implicitly assuming a periodic signal and what we get is a series of numbers not a continuous function.[12]

The number of different Fourier transforms has led to some confusion regarding terminology and there is a tendency to call all these operations "Fourier transforms" or, even more vaguely, "FFTs." Using appropriate terminology reduces confusion about what you are actually doing, and may even impress less informed bioengineers. To aid in being linguistically correct, Table 3.2 summarizes the terms and abbreviations used to describe the various aspects of Fourier analyses equations, and Table 3.3 does the same for the Fourier synthesis equation.

Tables 3.2 and 3.3 show a potentially confusing number of options for converting between the time and frequency domains. Of these options, there are only two real-world choices: the analysis and synthesis discrete Fourier transformations. In the interest of simplicity, the eight options are itemized in Table 3.4 (*sans* equations) with an emphasis on when they are applied.

Only two of these equations are used in real-world problems: the discrete Fourier series, usually just called the Fourier transform or DFT, and its inverse. These are the only equations that can be implemented on a computer. However, you need to know about all the versions, because they are sometimes referenced in engineering articles. If you do not understand these variations, you do not really understand the Fourier transform.

[12]In some theoretical analyses, infinite data are assumed, in which case the n/N in Equation 3.30 does become a continuous variable f as $N \rightarrow \infty$. In these theoretical analyses the data string, $x[n]$, really is aperiodic and the modified equation that results is the "discrete time Fourier transform" (the DTFT as opposed to just the DFT for periodic data). Since no computer can hold infinite data, the discrete time Fourier transform is of theoretical value only and is not used in calculating the spectrum.

TABLE 3.2 Analysis Equations (Forward Transform)[a]

$x(t)$ periodic and continuous:
Fourier Series:

$$X(m) = \frac{1}{T}\int_0^T x(t)e^{-j2\pi m f_1 t}dt \quad (3.33 \text{ and } 3.21)$$

$f_1 = 1/T$
$m = 0, \pm1, \pm2, \pm3,\ldots$

$x(t)$ aperiodic and continuous:
Fourier Transform or Continuous Time Fourier Transform:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt \quad (3.35 \text{ and } 3.25)$$

$x[n]$ periodic and discrete:
Discrete Fourier Transform (DFT) or Discrete Time Fourier Series[b,c]:

$$X[m] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi mn/N} \quad (3.34 \text{ and } 3.30)$$

$m = 0, \pm1, \pm2, \pm3,\ldots \pm M\ (M \le N)$

$x[n]$ aperiodic and discrete:
Discrete Time Fourier Transform (DTFT)[d]:

$$X[f] = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j2\pi nf} \quad (3.36)$$

[a]*Often ω is substituted for 2πf to make the equation shorter. In such cases the Fourier series integration is carried out between 0 and 2π or from −π to +π.*
[b]*Alternatively, 2πmnT$_s$/N is sometimes used instead of 2πmn/N where T$_s$ is the sampling interval.*
[c]*Sometimes this equation is normalized by 1/N and then no normalization term is needed in the inverse DFT.*
[d]*The DTFT cannot be calculated with a computer since the summations are infinite. It is used only in theoretical problems as an alternative to the DFT.*

TABLE 3.3 Synthesis Equations (Reverse Transform)[a]

Inverse Fourier Series:

$$x(t) = \sum_{n=-\infty}^{\infty} X(m)e^{j2\pi f_1 t} \quad m = 0, \pm1, \pm2, \pm3\ldots \quad (3.37)$$

Inverse Fourier Transform or Inverse Continuous Time Fourier Transform:

$$x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}df \quad (3.39 \text{ similar to } 3.27)$$

Inverse Discrete Fourier Transform (DFT) or Inverse Discrete Time Fourier Series[b,c]:

$$x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[m]e^{j2\pi mn/N} \quad (3.38)$$

Inverse Discrete Time Fourier Transform[d,e]:

$$x[n] = \frac{1}{2\pi}\int_0^{2\pi} X[f]e^{j2\pi nf}df \quad (3.40)$$

[a]*Often ω is substituted for 2πf to make the equation shorter. In such cases the Fourier series integration is carried out between 0 and 2π or from −π to +π.*
[b]*Alternatively, 2πmnT$_s$/N is sometimes used instead of 2πmn/N where T$_s$ is the sampling interval.*
[c]*Sometimes this equation is normalized by 1/N and then no normalization term is needed in the inverse DFT.*
[d]*The DTFT cannot be calculated with a computer since the summations are infinite. It is used only in theoretical problems as an alternative to the DFT.*
[e]*Evaluation of the integral requires an integral table because the summations that produce X[f] are infinite in number.*

TABLE 3.4 Real-World Application of Fourier Transform Equations

Analysis Type [Equation]	Application
Continuous Fourier series [Equation 3.31 and 3.33]	Analytical (i.e., textbook) problems involving periodic continuous signals
Discrete Fourier Series [Equation 3.30 and 3.34]	*All real-world problems*. (Implemented on a computer using the FFT algorithm)
Continuous Fourier Transform [Equation 3.25 and 3.35]	Analytical (i.e., textbook) problems involving aperiodic continuous signals
Discrete Time Fourier Transform [Equation 3.36]	None. Theoretical only
Inverse Fourier Series [Equation 3.37]	Analytical (i.e., textbook) problems to reconstruct a signal from the Fourier series. Not common even in textbooks. (Used in Example 3.3, but computer implemented.)
Inverse Discrete Fourier Series [Equation 3.38]	*All real-world problems*. (Implemented on a computer using the inverse FFT algorithm)
Inverse Fourier Transform [Equation 3.27 and 3.39]	Analytical (i.e., textbook) problems to reconstruct a signal from the continuous Fourier Transform. Not common even in textbooks. (No applications in this book)
Inverse Discrete Time Fourier Transform [Equation 3.40]	None. Theoretical only

Although the DFT is normally implemented on a computer, it is occasionally implemented manually as a textbook exercise. For some students, it is helpful to see a step-by-step implementation of the DFT, so the next example is dedicated to them.

EXAMPLE 3.7

Find the DFT of the discrete periodic signal shown in Figure 3.18 manually.

Solution: The signal has a period of seven points beginning at $n = -3$ and running to $n = +3$:

$$x[-3] = 0, x[-2] = 1, x[-1] = 0.5, x[0] = 0, x[1] = -0.5, x[2] = -1, x[3] = 0$$

The Fourier transform equation is given in Equation 3.30 and restated here with $N = 7$ so n ranging between ± 3. The coefficients will range between ± 6 (recall the maximum coefficient $M \leq N - 1$):

$$X[m] = \sum_{n=1}^{N} x[n] e^{\frac{-j2\pi mn}{N}} = \sum_{n=-3}^{3} x[n] e^{\frac{-j2\pi mn}{7}} \quad m = 0, \pm 1 \pm, 2, \pm 3, \ldots 6$$

Recall expanding the summation and substituting in the seven values of $x[n]$:

$$\begin{aligned} X[m] = \sum_{n=1}^{N} x[n] e^{\frac{j2\pi mn}{N}} &= x[-3]e^{j2\pi m3/7} + x[-2]e^{j2\pi m2/7} + x[-1]e^{j2\pi m1/7} + x[0]e^{j2\pi m0} \\ &\quad + x[1]e^{-j2\pi m1/7} + x[2]e^{-j2\pi m2/7} + x[3]e^{-j2\pi m3/7} \\ &= 0e^{j\pi m6/7} + 1e^{j\pi mn4/7} + 0.5e^{j\pi m2/7} + 0e^{j\pi m0} - 0.5e^{-j\pi m2/7} - 1e^{-j\pi m4/7} + 0e^{-j\pi m6/7} \end{aligned}$$

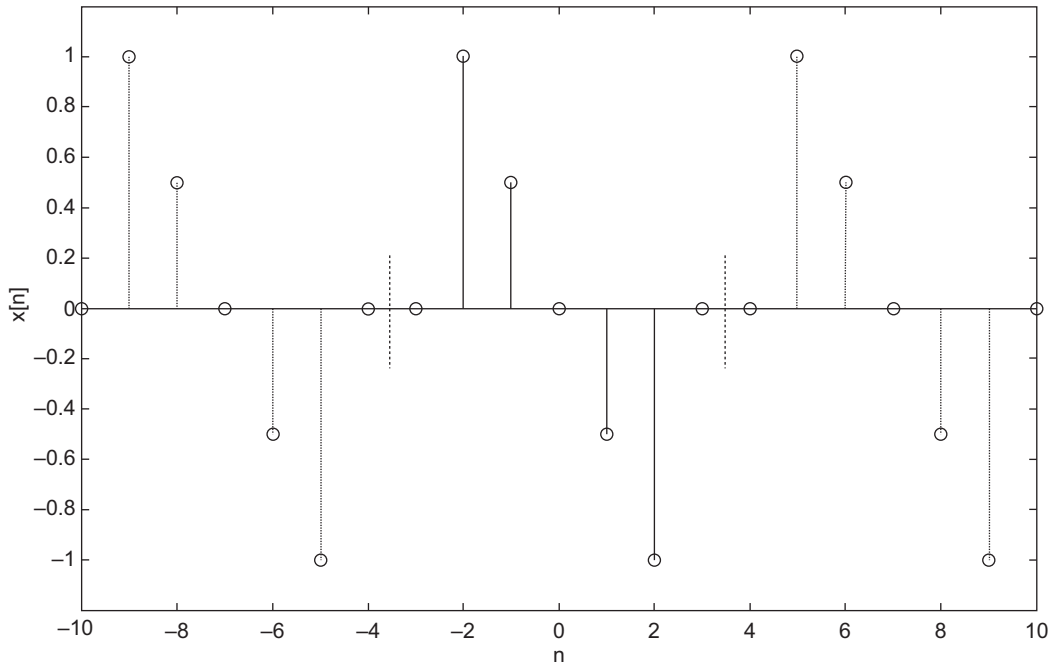


FIGURE 3.18 Discrete periodic waveform used in Example 3.7. The signal consists of discrete points indicated by the *circles*. The *lines* are simply to improve visibility. Discrete data are frequently drawn in this manner.

Collecting terms and rearranging:

$$X[m] = e^{j\pi m4/7} - e^{-j\pi m4/7} + 0.5\left(e^{j\pi m2/7} - e^{-j\pi m2/7}\right)$$

Noting that: $\sin x = \frac{e^{jx} - e^{-jx}}{2j}$

$$X[m] = 2j\sin(4/7\pi m) + j\sin(2/7\pi m) \quad m = 0, \pm 1, \pm 2, \ldots \pm 6$$

The problem set offers a couple of additional opportunities to solve for the DFT manually.

3.4.2 MATLAB Implementation of the Discrete Fourier Transform

MATLAB has a routine that uses the FFT algorithm to implement Equation 3.30 very quickly:

```
Xf = fft(x,n)        % Calculate the Fourier Transform
```

where `x` is the input waveform and `Xf` is a complex vector providing the sinusoidal coefficients. (Recall, it is common to use capital letters for the Fourier transform variable.) The first term of `Xf` is real and is the un-normalized DC component; you need to divide by the signal length, *N*, to get the actual DC component. The second term in `Xf` is the complex representation of the fundamental sinusoidal component; the third term represents the second harmonic; and

so on. The argument `n` is optional and is used to modify the length of data analyzed: if `n` is less than the length of `x`, then the analysis is performed over the first `n` points. If `n` is greater than the length of `x`, then the signal is padded with trailing zeros to equal `n`.

There is one downside to the `fft` routine. The FFT algorithm requires the data length to be a power of 2. Although the MATLAB routine will interpolate if need be, calculation time will go up and how much depends on data length. The algorithm is fastest if the data length is a power of 2, or if the length has many prime factors. For example, on a slow machine, a 4096-point FFT takes 2.1 s, but requires 7 s if the sequence is 4095 points long and 58 s if the sequence is 4097 points long. If at all possible, it is best to stick with data lengths that are powers of 2.

The magnitude and phase spectra are calculated from the complex output `Xf` using `abs(Xf)` and `angle(Xf)`, respectively (see Example 3.3). Again, the angle routine gives phase in radians so as to convert to the more commonly used degrees scale by $360/(2\pi)$.

EXAMPLE 3.8

Construct the waveform used in Example 3.2 (repeated below) and determine the Fourier transform using both the MATLAB `fft` routine and a direct implementation of the defining equations (Equations 3.8 and 3.9).

$$x(t) = \begin{cases} t & 0 < t \leq 0.5 \\ 0 & 0.5 < t \leq 1.0 \end{cases}$$

Solution: We need a total time of 1 s; let us assume a sample frequency of 256 Hz. This leads to a value of N that equals 256, which is a power of 2 as preferred by the `fft` routine. The MATLAB `fft` routine does no scaling. To compare the output of `fft` with the analytical results obtained from Equations 3.8 and 3.9, we must normalize by $2/N$.

The DC, or zero, frequency component is automatically calculated by MATLAB's `fft` routine, but it needs to be calculated separately when using Equation 3.10.

```
%  Example 3.8 Find the Fourier transform of half triangle waveform
%  used in Example 3.2.  Use both the MATLAB fft and a direct
%  implementation of Equations 3.8 and 3.9
%
T = 1;                          % Total time
F1 = 1/T;                       % Fundamental frequency
fs = 256;                       % Assumed sample frequency
N = fs*T;                       % Calculate number of points
M = 21;                         % Number of coefficients
t = (1:N)*T/N;                  % Generate time vector
f = (0:M)*fs/N;                 %  and frequency vector for plotting
x = [t(1:N/2) zeros(1,N/2)];    % Generate signal, ramp followed by zeros
%
Xf = fft(x);                    % Take Fourier transform, scale
Mag = abs(Xf)*2/N;              % Normalize
Phase = -angle(Xf)*360/(2*pi);
%
plot(f(1:M),Mag(1:M),'.'); hold on;  % Plot only first 20 coefficients plus DC
......labels....
```

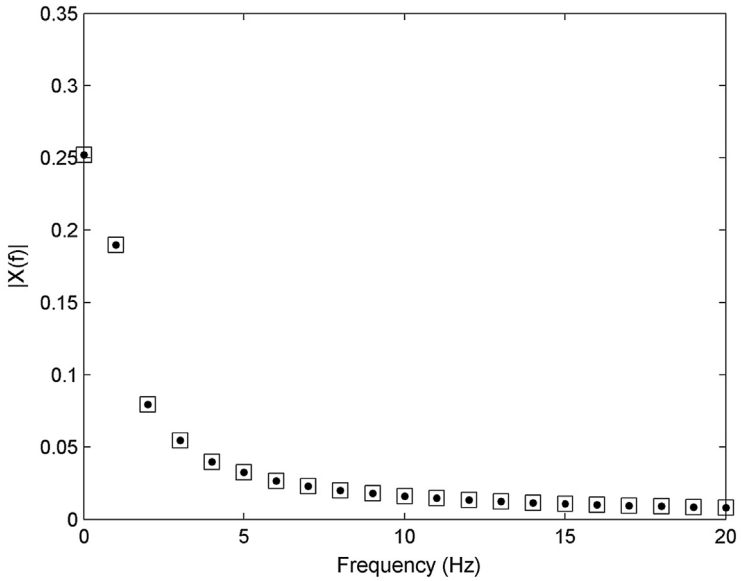


FIGURE 3.19 Magnitude frequency spectra produced by the MATLAB `fft` routine (*dots*) and a direct implementation of the Fourier transform equations, Equations 3.8 and 3.9 (*squares*).

```
%
% Calculate discrete Fourier Transform using basic equations
C(1) = (2/N)*sum(x);                              % DC Component, Equation 3.10
for m = 1:21
  a(m) = (2/N)*sum(x.*(cos(2*pi*m*f1*t)));        % Equation 3.8
  b(m) = (2/N)*sum(x.*(sin(2*pi*m*f1*t)));        % Equation 3.9
  C(m+1) = sqrt(a(m).^2 + b(m).^2);               % Equation 3.6
  theta(m) = atan2(b(m),a(m))* 360/(2*pi))        % Equation 3.7
end
disp([a(1:4)' b(1:4)' C(2:5)' Mag(2:5)' theta(2:5)' Phase(1:4)'])
plot(f(1:20),C(1:20),'sr');                       % Plot superimposed
```

Results: The spectrum produced by the two methods is identical as seen by the perfect overlap of points and squares in Figure 3.19.

The numerical values produced by this program are given in Table 3.5.

TABLE 3.5 Numerical Results From Example 3.8

m	a_m (Analytical)[a]	b_m (Analytical)	C_m (Analytical)	Mag(fft)	Phase (Analytical)	Phase (fft)
1	−0.1033 (−0.101)	0.1591 (0.159)	0.1897 (0.157)	0.1897	−122.98 (−122.42)	−121.57
2	0.0020 (0.0)	−0.0796 (−0.080)	0.0796 (0.080)	0.0796	88.59 (90.0)	91.40
3	−0.0132 (−0.011)	0.0530 (0.053)	0.0546 (0.054)	0.0546	−103.99 (−101.72)	−99.77
4	0.002 (0.0)	−0.0398 (−0.040)	0.0398 (0.040)	0.0398	87.18 (90.0)	92.81

[a]*Manual values are from Example 3.1.*

Analysis: Both methods produce identical magnitude spectra and similar phase spectra (compare C_m and *Phase* from the noncomplex analysis with the FFT results, Mag(fft) and Phase(fft)). Note that both the `atan2` and `angle` routines give the angle in radians, so in the program they are multiplied by $360/2\pi$ to convert to degrees.

The magnitudes and phases found by both methods closely match the values determined analytically (manually) in Example 3.1. However, the computer-determined phase angles for components 2 and 4 are not exactly zero as found analytically. This difference is due to small computational errors caused by rounding. This shows that analytical solutions done by hand can be more accurate than computer results!

How fast is the FFT? The next example compares the speed of direct implementation for the complex and noncomplex analysis equations against the FFT.

EXAMPLE 3.9

You are on a desert island with your laptop and your survival depends on finding the frequency spectra of a signal that is in your laptop. Unfortunately, you find your laptop does not have the FFT algorithm. "No problem," you say, "I will simply code the DFT algorithm given in Equation 3.30 and used in Example 3.6." (Of course you have copy of this textbook.) Then you find that for some mysterious reason your laptop cannot handle complex numbers. "No problem," you say, "I will simply code the discrete version of the real-valued Fourier series equations Equations 3.8 and 3.9." Naturally you expect the program will be considerably slower than the FFT algorithm, but you hope to find the solution before your battery dies.

To evaluate what you would be up against, compare the run times of the three possible approaches to evaluating the DFT. Your signal, `x` in file `desert_island.mat`, consists of approximately 8000 points sampled at 2 kHz. For the two non-fft approaches, calculate the magnitude and phase of the first 4000 components. (MATLAB's `fft` algorithm evaluates all N possible components.) Plot 4000 points of the magnitude spectra produced by the three methods superimposed. We use MATLAB's `tic` and `toc` to determine the run times.

Solution: Load the signal file. For the noncomplex DFT, reuse the code in Example 3.8 and for the complex FT that in Example 3.6. Apply the three methods in succession and time the executions. Plot the resulting magnitude spectra using different symbols. Again, since we want to compare results with Equations 3.8 and 3.9, we need to normalize by $2/N$.

```
% Example 3.9  Comparison of three different approaches to the DFT.
%
load desert_island.mat;  % Get signal
N = length(x);           % Number of samples
fs = 2000;               % Sample frequency (known)
f1 = fs/N;               % Fundamental frequency,Equation 3.31
M = 4000;                % Number of coefficients to evaluate
f = (0:M-1)*fs/N;        % Frequency vector for plotting
%
```

```
% Apply non-complex FT analysis
tstart = tic;            % Start timer
t = (1:N)/fs;            % Time vector
C(1) = mean(x);          % DC Component,Equation 2.10
for m = 1:M-1
  a(m+1) = sum(x.*(cos(2*pi*m*f1*t)));  % Equation 3.8
  b(m+1) = sum(x.*(sin(2*pi*m*f1*t)));  % Equation 3.9
end
C = sqrt(a.^2 + b.^2)/N;                % Find magnitude Equation 3.6
theta = (360/(2*pi)) * atan2(b,a);      % Find phase, Equation 3.7
disp(['Non-complex FT duration: ',num2str(toc(tstart))]);   % Display timing
plot(f,C,'ok'); hold on;         % Plot non-complex magnitude spectrum
clear C theta a b;               % To insure clean slate for next eval.
%
% Apply complex FT
tstart = tic;                    % Start timer
for m = 1:M
  Xf(m) = sum(x.*exp(-j*2*pi*m*(1:N)/N));   % Complex coefficients,Equation 3.30
end
Mag = abs(Xf)/N;                 % Magnitude,Equation 3.23
Theta = angle(Xf)*360/(2*pi);    % Phase
disp(['Complex FT duration: ',num2str(toc(tstart))]);  % Display timing
plot(f,Mag,'+k');                % Plot complex magnitude spectrum

clear Xf Mag Theta;              % To insure clean slate for next eval.
%
% FFT
tstart = tic;                    % Start timer
Xf = fft(x);
Mag = abs(Xf)/N;                 % Find Magnitude,Equation 3.23
Theta = angle(Xf)*360/(2*pi);    % Find phase
disp(['FFT duration: ',num2str(toc(tstart))]);   % Display timing
plot(f,Mag(1:M),'sqk');          % Plot FFT magnitude spectrum
```

Results: The magnitude spectra produced by the three methods are shown in Figure 3.20 to be identical (the three symbols overlap). The spectra look strange, with two plateaus, but the three methods agree so they must be correct. No doubt the spectra represent some sort of code that tells you something important to your survival. Good luck.

Although the results are the same, the computation times shown Table 3.6 are quite different. The complex analysis is a little faster than the noncomplex analysis, but the FFT is approximately 1000 times faster than the other approaches. The signal length, N, was actually 8192 points, which is a power of 2 (2^{13}), ideal for the FFT algorithm. Extending the data by only 1 point slowed the FFT algorithm to between 0.003 and 0.005 s and made the timing highly variable. Nonetheless, the FFT is

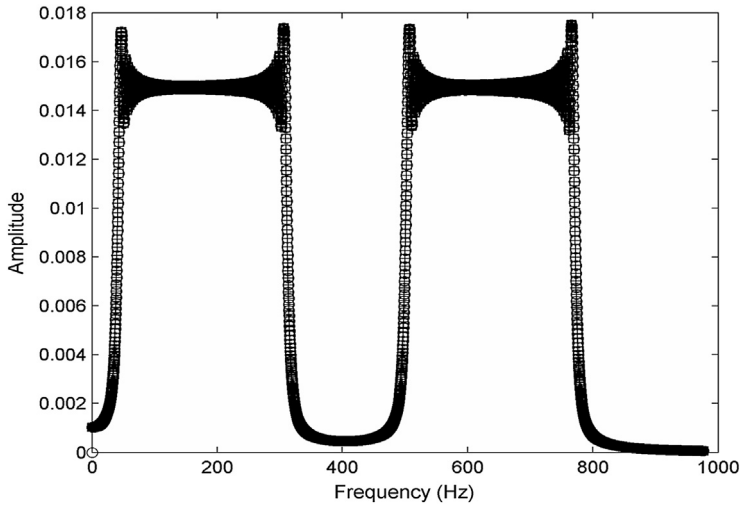


FIGURE 3.20 Magnitude spectra found for three algorithms that evaluate the Fourier transform analysis equations. The signal had $N = 8192$ and the first 4000 coefficients are shown (obviously not all 4000 points are actually plotted.) The symbols used for the different approaches are: "o" for noncomplex analysis; "+" for complex analysis, and a square for the FFT; however, the points overlap and it is difficult to see the individual symbols.

TABLE 3.6 Run-Time of Three Algorithms for the Fourier Transform Analysis Equations

Approach	Time for Execution (s)
Noncomplex analysis	3.55
Complex analysis	2.99
Fast Fourier transform	0.0027

really fast and the clear method of choice for converting from the time to frequency domain. Details of the FFT spectra are discussed in the next section.

3.4.3 Details of the DFT Spectrum

We have established that the three algorithms produce identical results, but the great speed of the FFT makes it the clear choice (moreover it is easier to code, just one line). We will take advantage of the FFT to examine some important properties of the DFT and the Fourier transform in general. To explore the properties of the Fourier transform, we sometimes use a test signal consisting of sinusoids and white noise. Here we present a MATLAB routine that generates multiple sinusoids accompanied by white noise. The routine, `sig_noise`, is found on the accompanying materials, and has a calling structure:

```
[x,t] = sig_noise([f],[SNR],N);     % Generate sinusoidal signals in noise
```

where `f` specifies the frequency of the sinusoid(s) in hertz, `SNR` specifies the desired noise associated with the sinusoid(s) in decibels, and `N` is the number of points in the signal. If `f` is a vector, then a number of sinusoids are generated, each with a signal to noise ratio (SNR) specified by `SNR` assuming it is a vector. If `SNR` is scalar, its value is used for the SNR of all the frequencies generated. The output waveform is in `x` and `t` is a time vector useful in plotting. The routine assumes a sample frequency of $f_s = 1$ kHz.

3.4.3.1 *DFT Spectral Redundancy*

The DFT produces twice as many points as contained in the input signal since both the magnitude and phase have the same number of points as the signal. Since the information content of the signal and its Fourier transform are the same, some on these points must be redundant. We have yet to look at a plot of the full spectrum generated by the FFT; that is, one with the same number of components as the signal (i.e., $M = N - 1$, in Equation 3.30). In the next example, we demonstrate that the upper half of the magnitude and phase spectra are mirror images of the lower half and therefore redundant.

EXAMPLE 3.10

Use `fft` to find the magnitude spectrum of a signal consisting of a single 250 Hz sine wave and white noise with an SNR of −14 dB. Plot the full magnitude spectrum (i.e., $M = N - 1$).

Solution: Use `sig_noise` to generate the waveform and use `fft to` obtain the complex Fourier transform. Use `abs` to find the magnitude spectrum and plot the full spectrum. Since we are not comparing our results with other methods, we use no normalization (the MATLAB default).

```
% Example 3.10  Determine the magnitude spectrum of a noisy waveform.
% N = 1024;                    % Number of data points
fs = 1000;                     % 1 kHz fs assumed by sig_noise.
f = (0:N-1)*fs/N;              % Frequency vector for plotting
% Generate signal using 'sig_noise'
%   250 Hz sin plus white noise; N data points; SNR = -14 dB
[x,t] = sig_noise (250,-14,N);  % Generate signal and noise
%
Xf = fft(x);                   % Calculate FFT
Mf = abs(Xf);                  % Calculate the magnitude
plot(f,Mf);                    % Plot the magnitude spectrum
   .......label and title.......
```

Analysis: The program is straightforward. After constructing the signal using the routine `sig_noise`, the program takes the Fourier transform with `fft` and then plots the magnitude (using `abs`) versus frequency using a frequency vector to correctly scale the frequency axis. The frequency vector ranges from 0 to `N-1` to include the DC term (which happens to be zero in this example.)

Results: The spectrum is shown in Figure 3.21 and the peak related to the 250 Hz sine wave is clearly seen. The spectrum above $f_s/2$ (i.e., 500 Hz, dashed line) is a mirror image of the lower half of the spectrum (as explained in Section 4.2.1). Theoretically the background spectrum should be a constant value since it is due to white noise, which has equal energy at all frequencies. As is typical

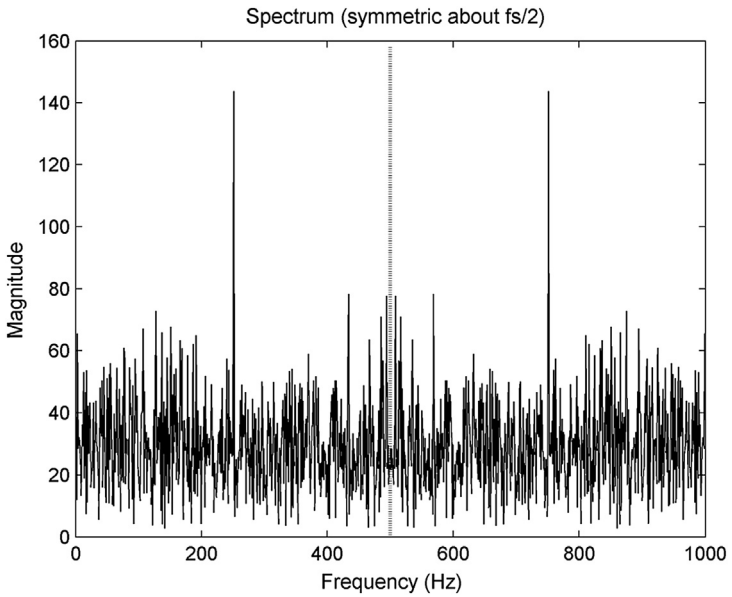


FIGURE 3.21 Plot produced by the previous MATLAB program. The peak at 250 Hz is apparent. The sampling frequency of these data is 1 kHz and the spectrum is symmetric about $f_s/2$ (*dashed line* at 500 Hz) for reasons described later. Normally only the first half of this spectrum would be plotted. (Signal characteristics: $f_{sine} = 250$ Hz; SNR $= -14$ dB; $N = 1024$.)

of noise spectra, the background is highly variable with occasional peaks that could be mistaken for signals. A better way to determine the spectrum of white noise is to use an averaging strategy described in the next chapter.

3.4.3.2 *Phase Wrapping*

The `fft` routine produces a complex spectrum that includes both the magnitude and phase information. Calculating and plotting both the magnitude and phase spectra seems straightforward, but the latter can be problematic. The output of the `angle` routine is limited to $\pm 2\pi$; larger values "wrap around" to fall within that limit. Thus a phase shift of $3\pi/2$ would be rendered as $\pi/2$ by the `angle` routine, as would a phase shift of $5\pi/2$. So when a signal's spectrum results in large phase angles, this wraparound characteristic produces jumps on the phase curve. An example of this is seen in Figure 3.22, which shows the magnitude and phase spectra of a pulse signal. The phase plot, this time plotted in radians, shows a sharp transition of approximately 2π whenever the phase reaches $-\pi$ (lower *dashed line*, Figure 3.22B).

The correct phase spectrum could be recovered by subtracting 2π from the curve every time there is a sharp transition. This is sometimes referred to as unwrapping the phase spectrum. It would not be difficult to write a routine to check for large jumps in the phase spectrum and subtract 2π from the rest of the curve whenever they are found. But whenever there is a common problem, you can bet that MATLAB has a routine to address it. The MATLAB

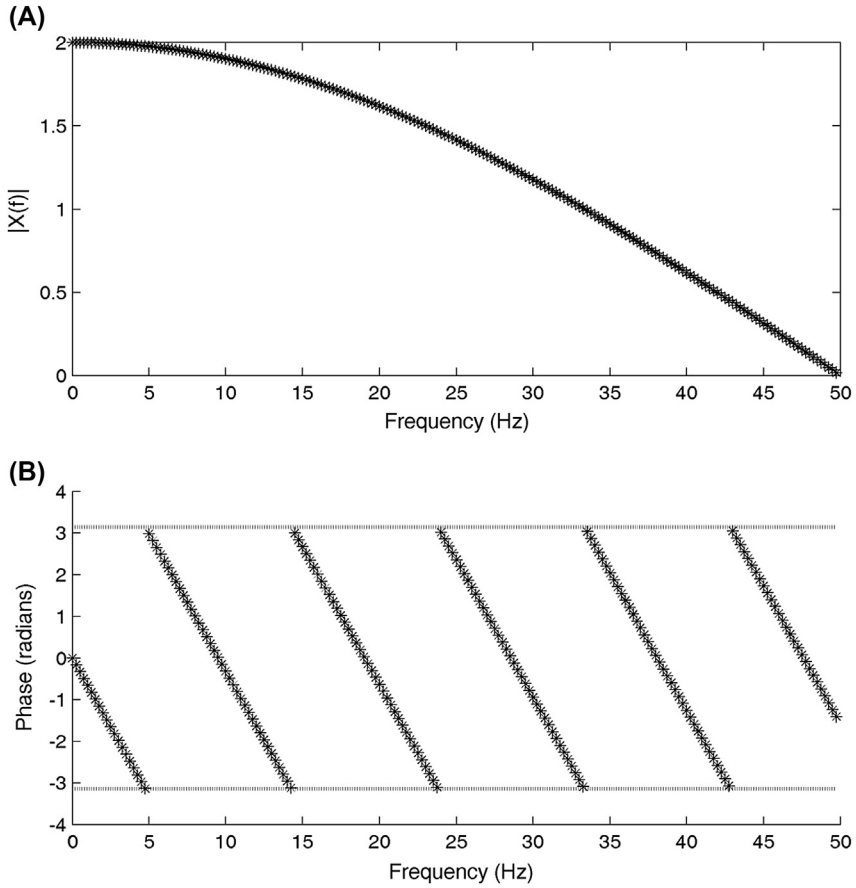


FIGURE 3.22 (A) Magnitude spectrum of a pulse waveform. (B) Phase spectrum in radians of the pulse waveform showing upward jumps of approximately 2π whenever the phase approaches $-\pi$. The *dashed horizontal lines* represent $\pm\pi$.

routine `unwrap` checks for jumps greater than π and does the subtraction whenever they occur. (The threshold for detecting a discontinuity can be modified as described in the `unwrap` help file.) The next example illustrates the use of `unwrap` for unwrapping the phase spectrum.

3.4.3.3 *The Effect of Time Shifts on the Fourier Transform*

Shifting a signal in time has a well-defined effect on the Fourier transform. This can be described by comparing the Fourier transform of two signals: an unshifted signal $x(t)$ and signal time shifted by an amount τ, $x(t-\tau)$. Using the Fourier transform for a continuous aperiodic function, Equations 3.25 and 3.35 (again we use ω instead of $2\pi f$ for variety):

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$$

For signal time shifted τ sec:

$$X(f) = \int_{-\infty}^{\infty} x(t-\tau)e^{-j\omega(t)}dt$$

Substituting $u = t - \tau$ and $t = u + \tau$:

$$X(f) = \int_{-\infty}^{\infty} x(u)e^{-j\omega(u+\tau)}du = e^{-j\omega\tau}\int_{-\infty}^{\infty} x(u)e^{-j\omega(u)}du = X(f)_{unshifted}e^{-j\omega\tau} \tag{3.41}$$

So the effect of a time shift, τ, in the frequency domain is to multiply the Fourier transform of the unshifted signal by $e^{-j\omega\tau}$. The magnitude of this term is 1.0 and the phase is $-\omega\tau$ or $-2\pi f\tau$. So shifting a signal in time has no effect on the magnitude spectrum, but does decrease the phase spectrum: the phase decreases linearly with frequency and in proportion to the time shift. The time-shift influence on the DFT is illustrated empirically in the next example.

EXAMPLE 3.11

Evaluate and plot the magnitude and phase of the waveform shown in Figure 3.14, but for two different time shifts. One signal should have the waveform symmetrical about $t = 0$ and the other centered in the signal array. Plot the two signals and their spectra. Plot the phase shift in degrees with and without unwrapping. Use the MATLAB `unwrap` routine.

Solution: The hardest part of this example is constructing the two signals. The waveform in Figure 3.14 is called a "wavelet" and is symmetrical. The equation for the right-hand side is:

$$x(t) = e^{-t^2}\cos\left(2\pi\sqrt{\frac{2}{\ln 2}}t\right) \tag{3.42}$$

There are several ways to construct the two waveforms. For a signal that is symmetrical about $t = 0$, we recall that all DFT signals are periodic. To make the signal symmetrical about $t = 0$, we can use Equation 3.42 to build the positive half of the wavelet waveform at the beginning of the array, then construct its reversed version at the end of the array, with some zero points in between. We can use MATLAB's `fliplr` routine to reverse the waveform. This can be implemented using the MATLAB code:

```
morlet = (exp(-t1.^2).* cos(wo1*t1));              % Generate Morlet wavelet
x1 = [morlet, zeros(1,2*N1), fliplr(morlet)];    % Construct using t=0 symmetry
```

where `N1` is 512 points and is the same length as the wavelet, `morlet`. When plotted this may not look like a waveform that is symmetrical, but it is if you consider that it represents a periodic signal. For the centered waveform, we put the flipped and unflipped version of Equation 3.42 in sequence with zeros placed symmetrically on either side.

```
x1 = [zeros(1, N1), fliplr(morlet), morlet, zeros(1,N1)];     % Construct centered
```

Once we have constructed the two signals, the rest is straightforward following methods used in previous examples. We find the length of the signals, N, construct frequency, and time vectors for plotting, find the complex spectra, and extract the magnitude and phase spectra. We find and plot both the wrapped and unwrapped phase spectra. As with most signals, only the lower frequencies are of interest, so we plot only the first 30 frequency components (found by trial and error to be significantly greater than zero). We arbitrarily assume a sample frequency, f_s, of 150 Hz.

```
% Example 3.11 Evaluate and plot the magnitude and phase of signal based on
% the waveform shown in Figure 3.14.  Construct two signals with two different time
shifts.
% One waveform should be symmetrical about t = 0 and the other centered in the
signal array.
%
fs =150;                          % Assumed sample frequency
%
% Generate waveforms based on Morlet wavelet
N1 = 512;                         % Wavelet number of points
t1 = ((0:(N1/2)-1)/fs)*2;         % Show =/- 40 sec of the wavelet
wo1 = pi*sqrt(2/log2(2));         % Wavelet constant
mor = exp(-t1.^2).* cos(wo1*t1);       % Generate Morlet wavelet
x1 = [mor, zeros(1,2*N1), fliplr(mor)];     % Symmetrical about t = 0
x2 = [zeros(1,N1), fliplr(mor), mor, zeros(1,N1)];  % Centered
%
N = length(x1);                   % Signal number of points
t = (1:N)/fs;                     % Time vector for plotting
f = (0:N-1)*fs/(N-1);             % Frequency vector for plotting
%
subplot(1,2,1);                   % Plot the two signals
plot(t,x1)                        % Signal 1 time domain
  .......labels .........
subplot(1,2,2);
plot(t,x2)                        % Signal 2 time domain
  .......labels and new figure.........
%
Xf = fft(x1);                     % Signal 1 complex spectrum
Mag = abs(Xf);                    % Get magnitude
Phase = angle(Xf)*360/(2*pi);     % Get phase in deg not unwrapped
Phase_unwrap = unwrap(angle(Xf))*360/(2*pi);  % Unwrapped phase in deg.

subplot(2,2,1);                   % Plot Signal 1 spectrum
plot(f(1:30),Mag(1:30),'.');      % Magnitude spectrum. Only first 30 components
  .......labels and title.........
subplot(2,2,2); hold on;
  plot(f(1:30),Phase(1:30),'+'); % Plot wrapped phase. Use + points.
  plot(f(1:30),Phase_unwrap(1:30),'.');  % Plot unwrapped phase
  .......labels .........
```

```
%
Xf1 = fft(x2);                    % Signal 2complex spectrum
Mag = abs(Xf1);                   % Get magnitude
Phase = angle(Xf1)*360/(2*pi);    % Get phase in deg not unwrapped
Phase_unwrap = unwrap(angle(Xf1))*360/(2*pi);    % Unwrapped phase in deg.
subplot(2,2,3);           % Plot Signal 2 spectrum
plot(f(1:30),Mag(1:30),'.');      % Magnitude spectrum
  .......labels .........
subplot(2,2,4); hold on;
  plot(f(1:30),Phase(1:30),'+'); % Plot wrapped phase. Use + points.
  plot(f(1:30),Phase_unwrap(1:30),'.');  % Plot unwrapped phase
  .......labels .........
```

Results: The time domain representation of the two waveforms is Figure 3.23. The waveforms have the expected shape and position.

The spectra of the two signals are shown in Figure 3.24. The magnitude spectra of the two signals are identical (Figure 3.24, left-hand plots). Shifting the position of the waveform does not affect the magnitude spectrum: it is not affected by time shifts of the waveform. However, the two phase spectra are quite different. The phase of the signal that is symmetrical about $t = 0$ is constant and essentially zero (Figure 3.24 upper right) for all frequencies so the wrapped and unwrapped phase points are the same. The signal with the centered waveform produces a phase shift that increases linearly with frequency (Figure 3.24 lower right). Because the time shift is so large, the decrease in phase shift is enormous (note the scale of the vertical axis in Figure 3.24 lower right). If the phase is not unwrapped (+ points), it warps around and looks like a set of alternating points. The

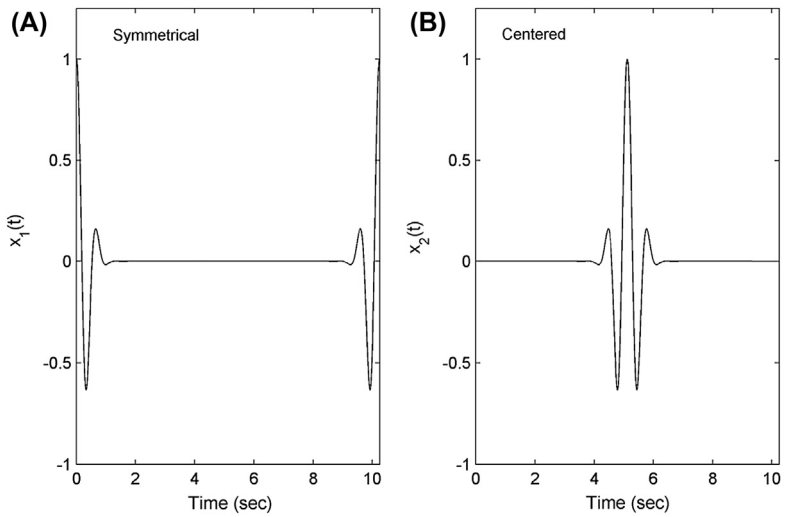


FIGURE 3.23 The two signals used in Example 3.11. Both are based on the Morlet wavelet. (A) The waveform is symmetrical about $t = 0$ (recall these are periodic signals). (B) The waveform is centered in the signal array.

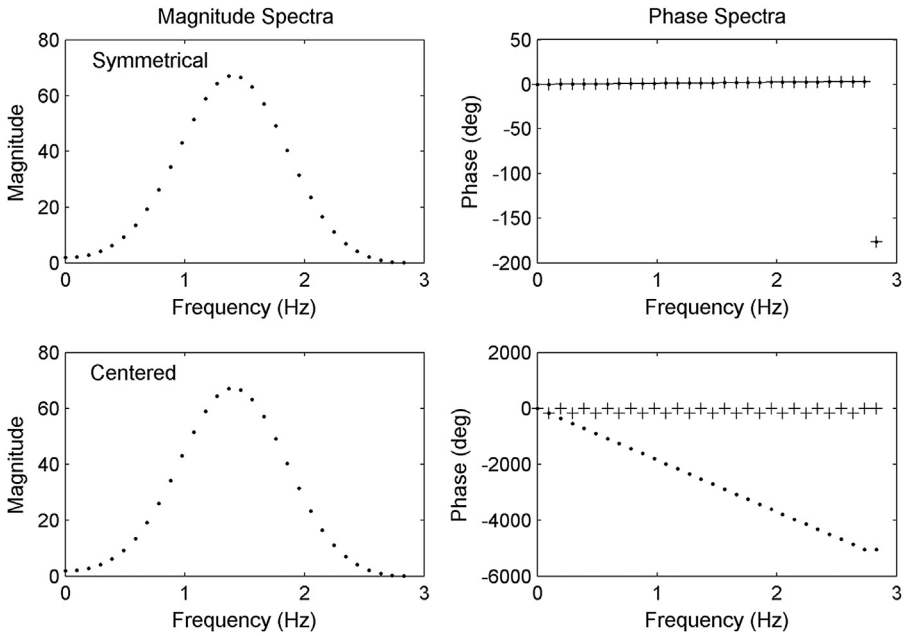


FIGURE 3.24 The magnitude and phase spectra of the signals shown in Figure 3.23. Left-hand plots: The magnitude spectra of the two signals are identical. The magnitude spectrum is not influenced by time shifts. Right-hand plots: The phase spectra of the signal symmetrically placed about $t = 0$ is constant at zero for all frequencies (upper right plot) and hence the same for wrapped and unwrapped data. The centered signal, when it is unwrapped, shows a dramatic linear decrease in phase with increasing frequency (dots, lower right), but this behavior is not apparent in the unwrapped data (+ points, lower right).

unwrapped version (dots) shows the dramatic decrease in phase with increasing frequency (Figure 3.24, lower right). The take-home message is always unwrap the phase data unless you are absolutely sure it will never be greater or less than 2π.

3.4.3.4 *Respiratory Example*

The next example applies the DFT to a signal related to respiration and uses the spectrum to find the breathing rate.

EXAMPLE 3.12

Find the magnitude and phase spectrum of the respiratory signal found in variable `resp` of file `Resp.mat` (data from PhysioNet, Golberger et al., 2000). The data have been sampled at 125 Hz. Plot only frequencies between the fundamental and 2 Hz. The magnitude spectrum should have a

maximum energy peak corresponding to the frequency of the average respiration rate. Find this peak and use it to estimate the breaths per minute of this subject.

Solution: Loading the data and taking the DFT are straightforward, as is finding the spectral peak. The trick in this example is to find the series number, m, that corresponds to 2 Hz. From Equation 3.32 , $f = mf_s/N$. So 2 Hz corresponds to series number $m_{2\text{Hz}} = f\,N/f_s = 2N/f_s$.

To find the peak frequency, use the second output of MATLAB's `max` routine to find the series number m_{max} that corresponds to the maximum spectral energy. The second output argument of this routine gives the index of the maximum value, in this case the index of m_{max}. The frequency corresponding index m_{max} is $f_{max} = m_{max}f_s/N$ (Equation 3.32, but you can also get this from your frequency vector). The average interval between breaths is $t_{breath} = 1/f_{max}$. This represents the average time between each breath so this time divided into 60 gives breaths per min: $\text{BPM} = 60/t_{max}$.

In all of this remember that in the `fft` output, `X(m)`, the first frequency component starts at `X(2)` as `X(1)` holds the DC component.

```
% Example 3.12  Example applying the FT to a respiration signal.
%
load Resp                    % Get respiratory signal
fs = 125;                    % Sampling frequency in Hz
max_freq = 2;                % Max desired plotting frequency in Hz
N = length(resp);            % Length of respiratory signal
f = (1:N)*fs/N;              % Construct frequency vector
t = (1:N)/fs                 % Construct the time vector
plot(t,resp)                 % Plot the time signal
   .......labels.......
Xf = fft(resp);              % Calculate the spectrum
m_2Hz = round(2N/fs);        % Find m for 2 Hz
subplot(2,1,1);
plot(f(1:m_2Hz-1),abs(Xf(2:m_2Hz)));     % Plot magnitude spectrum
.......labels.......
phase = unwrap(angle(Xf))*360/(2*pi);    % Calculate phase spectrum in deg
%
subplot(2,1,2);
plot(f(1:m_2Hz-1),phase(2:m_2Hz));       % Plot phase spectrum
   .......labels.......
[peak m_max] = max(abs(Xf(2:m_Hz)));     % Find m at max magnitude peak
f_max = f(m_max)                         % Calculate and display frequency
time_max= 1/f_max                        % Calculate and display max time
breath_min = 60/ t_max                   % Calculate and display breath/min
```

Results: The time data are shown in Figure 3.25 to be a series of peaks corresponding to maximum inspiration. The spectrum plot produced by the program is shown in Figure 3.26. A peak is shown around 0.3 Hz, with a second peak around twice that frequency, a harmonic of the

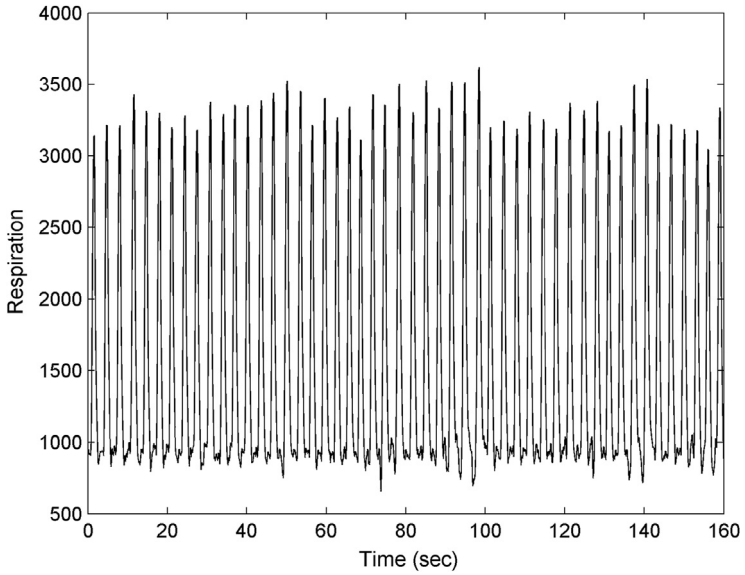


FIGURE 3.25 Respiratory signal used in Example 3.12. The peaks correspond to maximum inspiration.

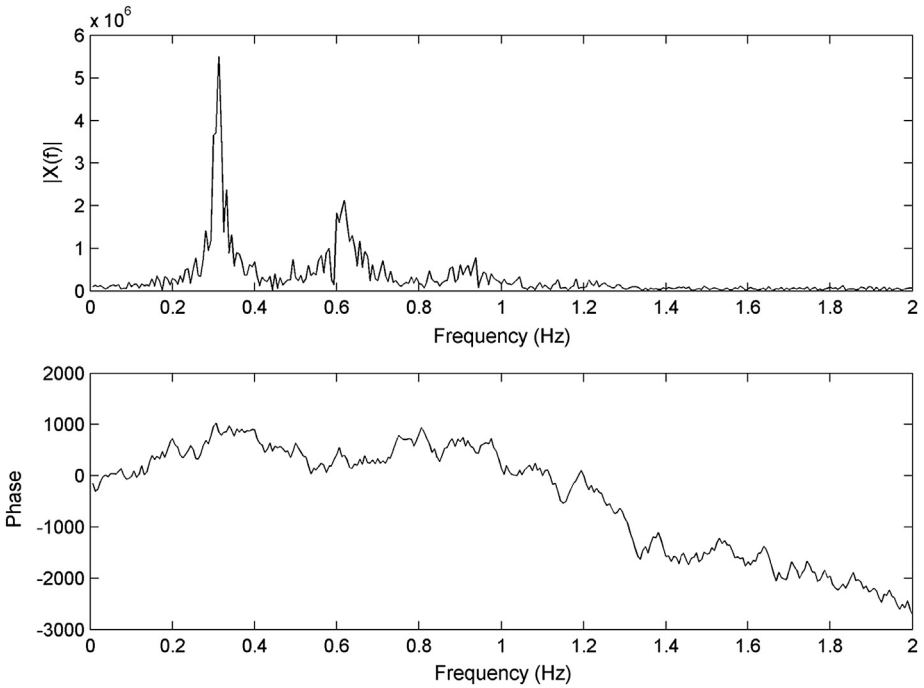


FIGURE 3.26 The magnitude and phase spectrum of the respiratory signal seen in Figure 3.25. The peak around 0.3 Hz corresponds to the interval between breaths. This is equivalent to 18.75 breaths per min.

first peak. The frequency of the peak was found to be 0.3125 Hz corresponding to an interbreath interval of 3.2 s giving rise to a respiratory rate of 18.75 breaths/min. Of course the respiratory rate could have been found from the time domain data by looking at intervals between the peaks in Figure 3.26, and then taking averages, but this would have been more difficult to code.

3.4.4 The Inverse Discrete Fourier Transform

The inverse Fourier transform performs the reverse Fourier transform and constructs a waveform from its Fourier coefficients. It implements the discrete version of the Fourier synthesis equation, Equation 3.38. It would not be difficult to code Equation 3.38 using a slightly modified version of Example 3.6. But, as usual, it is easier to use MATLAB's inverse Fourier transform routine, `ifft`. An opportunity to code a direct implementation of Equation 3.38 and show you are as good as MATLAB is provided in one of the problems.

The format of MATLAB's `ifft` routine is:

```
x = ifft(Xf,N);     % Inverse Fourier transform
```

where `Xf` is the Fourier coefficients and `N` is an optional argument limiting the number of coefficients to use. A few other less common options are described in the associated help file.

EXAMPLE 3.13

The final example uses the Morlet waveform used in Example 3.11, with the waveform initially on the left side of the signal array. We take the Fourier transform, and then modify the resulting spectrum before reconstructing a time domain signal using the inverse Fourier transform. Specifically, we multiply the phase spectrum by 4. In Example 3.11, we saw that a time shift alters only the phase; specifically, a shift that increases time increases the downward slope of the phase spectrum. Since the phase curve in Example 3.11 is linear, multiplying the phase curve by 4 increases the downward slope of the phase curve by 2 and should shift the Morlet waveform to the right, but not change its shape.

Solution: The first part of the program is identical to Example 3.11 with a slight change in the construction of the signal: the Morlet waveform is placed at the beginning of the signal array. This is done just for variety and the visual effect when the waveform is time shifted. The DFT is taken and the magnitude and phase are determined. The phase is left in radians and not unwrapped since we want to convert back using the inverse DFT after modification. The phase spectrum is multiplied by 4 and the complex spectrum is constructed from the combination of modified phase spectrum and unmodified magnitude spectrum.

MATLAB's inverse Fourier transform routine, iff(Y), expects the input, Y, to be in the complex form:

$$X_m = a_m + jb_m;$$

where: $a_m = C_m \cos(\theta_m)$ and $b_m = -C_m \sin(\theta_m)$ where C_m is the magnitude and θ_m is the modified phase (see Equation E-1, Appendix E).

```
%  Example 3.13 Take the Fourier transform of the Morlet waveform used in Example 3.11.
%  Modify the phase spectrum then reconstruct the time signal using the inverse
%  Fourier transform.
%
fs =150;              % Assumed sample frequency. Same as Example 3.11.
N1 = 512;             % Next 4 lines, same as Example 3.11
wo1 = pi*sqrt(2/log2(2));                % Const. for wavelet time scale
t1 = ((0:(N1/2)-1)/fs)*2;                % Time vector for Morlet wavelet
mor = (exp(-t1.^2).* cos(wo1*t1));       % Generate Morlet wavelet
x = [fliplr(mor), mor, zeros(1,2*N1)];   % Construct signal
%
N = length(x);                % Signal number of points
t = (1:N)/fs;                 % Time vector for plotting
f = (0:N-1)*fs/N;             % Frequency vector for plotting
%
subplot(1,2,1);
plot(t,x);                    % Plot original signal
  ........labels and axis.......
%
X = fft(x);                   % Signal complex spectrum
x1 = ifft(X);                 % Unmodified inverse Fourier transform
Mag = abs(X);                 % Get magnitude
Phase = angle(X);             % Get phase in radians not unwrapped
Phase1 = 4*Phase;             % Modify phase spectrum
%
am = Mag.*cos(Phase1);        % Calculate new real coefficients
bm = Mag.*sin(Phase1);        % Calculate new imaginary coefficients
Y = am + j*bm;                % New complex spectrum
y = ifft(Y);                  % Take modified inverse Fourier transform.
%
subplot(1,2,2); hold on;      % Plot modified and original signal
plot(t,x1);                   % Signal from unmodified spectrum
plot(t,y,':');                % Signal from modified spectrum
  ........labels and axis.......
```

Results: The original signal, Figure 3.27A, is the same as that used in Example 3.11 except that the Morlet wavelet occurs at the beginning of the signal. The signal reconstructed from the unmodified complex signal is identical to the original, Figure 3.27B (solid line). The signal constructed from the modified phase spectrum has the same shape as the original, Figure 3.27B (dotted line), but has been shifted in time approximately 5 s. This is expected because increasing the slope of the phase trajectory is equivalent to a time shift.

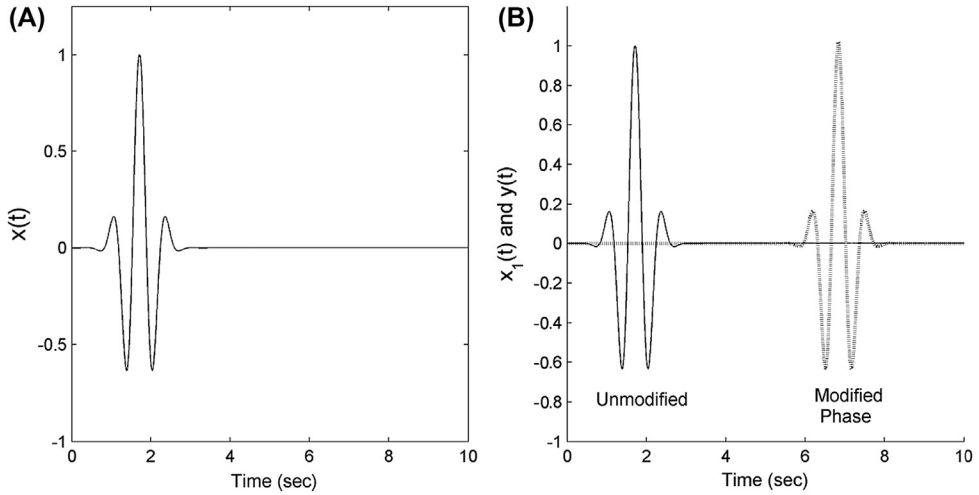


FIGURE 3.27 (A) The original signal used in Example 3.13 is similar to that used in Example 3.11 except for the position of the wavelet waveform. (B) The signal reconstructed from the unmodified Fourier transform of the signal in A is the same. The signal reconstructed from the Fourier transform after the phase shift has been modified by increasing its slope is the same but shifted in time.

There are many signal processing algorithms that follow the strategy used in this example; they move signals to the frequency domain, process the resulting spectral curves, and then move the signals back into the time domain. Although this may seem convoluted, the speed of the FFT and inverse Fourier transform makes this approach attractive and often faster than operations in the time domain.

3.5 SUMMARY

The sinusoid (i.e., $A \cos(\omega t + \theta)$) is a unique signal with a number of special properties. A sinusoid can be completely defined by three values: its amplitude A, its phase θ, and its frequency ω (or $2\pi f$). Any periodic signal can be broken down into a series of harmonically related sinusoids, although that series might have to be infinite in length. Reversing that logic, a periodic signal can be reconstructed from a series of sinusoids so a periodic signal can be equivalently represented by a sinusoidal series. A sinusoid is also a pure signal in that it has energy at only one frequency, the only waveform to have this property. Since they have such a simple frequency domain representation, sinusoids are useful intermediaries between the time and frequency domain representation of signals.

The technique for determining the sinusoidal series representation of a periodic signal is known as Fourier series analysis. To determine the Fourier series, the signal is represented by an equivalent series of sinusoids that have frequencies harmonically related to the signal. The amplitude and phase of this series is found by correlating the signal with each sinusoid in the series. The resulting amplitude and phase of these sinusoids can be plotted against the associated frequency to construct the frequency domain representation of the signal. As harmonically related sinusoids are orthogonal, the components of the Fourier series have no influence on one another. Fourier series analysis is often described and implemented using the complex representation of a sinusoid.

If the signal is not periodic, but exists for a finite time period, Fourier decomposition is still possible by assuming that this aperiodic signal is in fact periodic, but that the period is infinite. This approach leads to the true Fourier transform where the correlation is now between the signal and infinite number of sinusoids having continuously varying frequencies. The frequency plots then become continuous curves. The inverse Fourier transform also applies to continuous frequency.

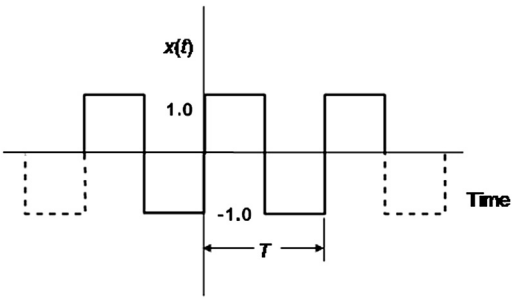
Fourier decomposition applied to digitized data is known as the discrete Fourier series or the DFT. In the real world (as opposed to the textbook world), Fourier analysis is always done on a computer using a high-speed algorithm known as the fast Fourier transform or FFT. The discrete equations follow the same pattern as those developed for the continuous time signal except integration becomes summation and both the sinusoidal and signal variables are discrete. The digitized signal is assumed to be periodic even if it is not. Cases where this assumption causes problems, and possible solutions, are described in the next chapter. There is a discrete version of the continuous Fourier transform known as the discrete time Fourier transform, but this equation and its inverse are of theoretical interest only.

The FFT works best on signals that have a length that is a power of 2. They produce a complex spectrum that is the same length as the signal, N, and ranges in frequencies between 0 and f_s Hz. But the second half of the spectrum is redundant: it is the mirror image of the first half so only the components between and 0 and $f_s/2$ Hz are of interest. Taking the absolute value and angle of the complex spectrum produces the magnitude and phase spectra (Equations 3.23 and 3.24). Each of these spectra is N points long, but only $N/2$ points are unique, so the total number of unique points in the two spectra is equal to the number of points in the signal.

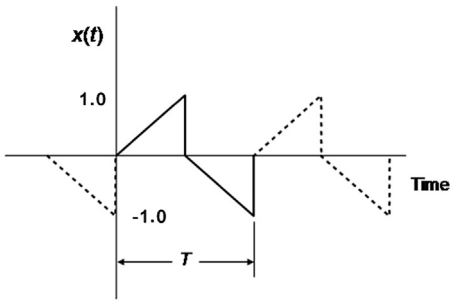
The FFT implements the Fourier analysis equations while the IFFT implements the Fourier synthesis equations. A number of signal processing operations involve converting a signal to the frequency domain using the FFT, operating on the signal while in the frequency domain, then converting back to a time domain signal using the IFFT. The great speed of the FFT and IFFT not only makes such involved operations practical, but also greatly enhances the value and practicality of time–frequency conversion.

PROBLEMS

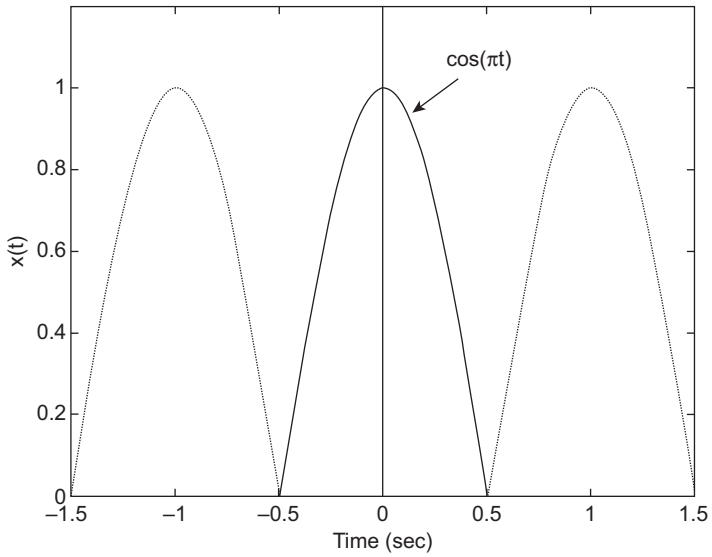
1. Find the Fourier series of the square wave below using analytical methods (i.e., Equations 3.8 and 3.9). (Hint: Take advantage of symmetries to simplify the calculation.)



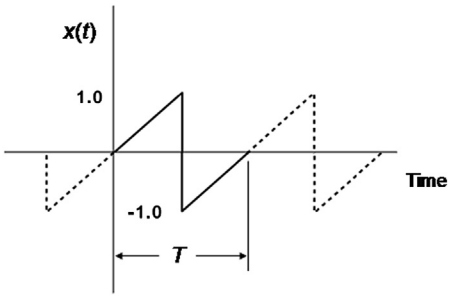
2. Find the Fourier series of the waveform below using analytical methods. The period, T, is 1 s.



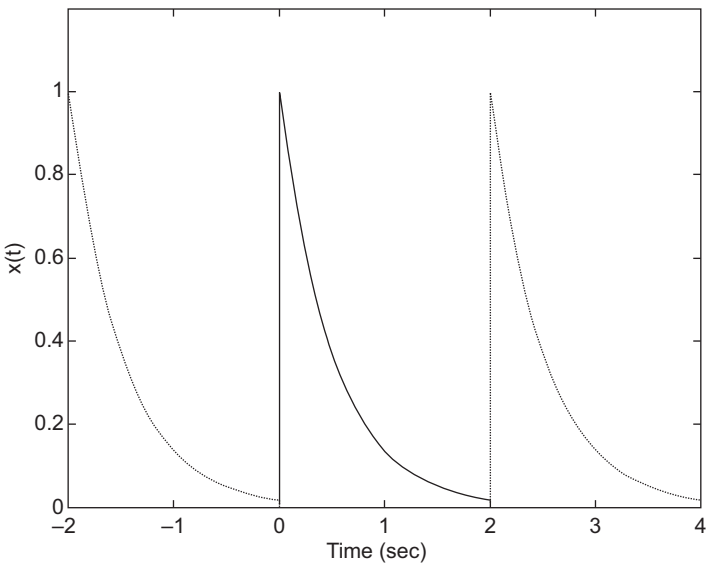
3. Find the Fourier series of the half-wave rectified sinusoidal waveform below using analytical methods. Take advantage of symmetry properties.



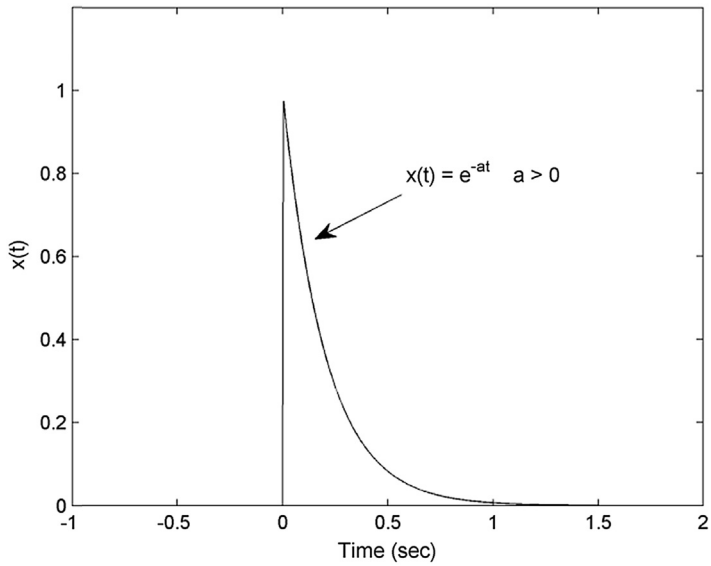
4. Find the Fourier series of the "sawtooth waveform" below using analytical methods.



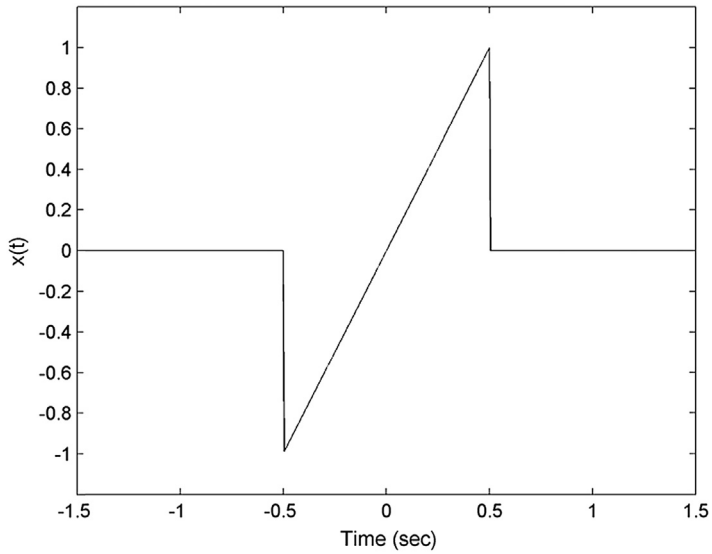
5. Find the Fourier series of the periodic exponential waveform shown below where $x(t) = e^{-2t}$ for $0 < t \leq 2$ (i.e., $T = 2$ s). Use the complex form of the Fourier series in Equation 3.20.



6. Find the continuous Fourier transform of an aperiodic pulse signal given in Example 3.5 using the complex equation, Equation 3.25 (or Equation 3.35).
7. Find the continuous Fourier transform of the aperiodic signal shown below using the complex equation, Equation 3.25.



8. Find the continuous Fourier transform of the aperiodic signal shown below. This is easier to solve using the noncomplex form, Equation 3.26, in conjunction with symmetry considerations.



9. Use the MATLAB Fourier transform routine to find the spectrum of a waveform consisting of two sinusoids with frequencies of 200 and 400 Hz. Make $N = 512$ and $f_s = 1$ kHz. Take the Fourier transform of the waveform and plot the magnitude of the full spectrum (i.e., 512 points) as in Example 3.10. Generate a frequency vector so the spectrum plot has a properly scaled horizontal axis. (Hint: To generate the waveform, first construct a time vector `t`, then generate the signal using the code: `x = sin(2*pi*f1*t) + sin (2*pi*f2*t)`where `f1` = 200 and `f2` = 400.)

10. Use the routine `sig_noise` to generate a waveform containing 200 and 400 Hz sine waves as in Problem 9, but add noise so that the signal to noise ratio (SNR) is −8 dB, i.e., `x = sig_noise([200 400], -8,N)` where `N` = 512. Recall `sig_noise` assumes $f_s = 1$ kHz. Plot the magnitude spectrum, but only plot the nonredundant points (2 to $N/2$) and do not plot the DC term, which again is zero. Repeat for an SNR of −16 dB. Use `subplot` to plot the two spectra side by side for easy comparison and scale the frequency axis correctly. Note that the two sinusoids are hard to distinguish at the higher (−16 dB) noise level.

11. Use the routine `sig_noise` to generate a waveform containing 200 and 400 Hz sine waves as in Problems 9 and 10 with an SNR of −12 dB with 1000 points. Plot only the nonredundant points (no DC term) in the magnitude spectrum. Repeat for the same SNR, but for a signal with only 200 points. Use subplot to plot the two spectra side by side for easy comparison and scale the frequency axis correctly.

Note that the two sinusoids are hard to distinguish with the smaller data sample. Taken together, Problems 10 and 11 indicate that both data length and noise level are important when detecting sinusoids (also known as "narrowband signals") in noise.

12. Generate the signal shown in Problem 2 and use MATLAB to find both the magnitude and phase spectrum of this signal. Assume that the period, T, is 1 sec and $f_s = 500$ Hz (hence, $N = 500$). Plot the time domain signal, $x(t)$, then calculate and plot the spectrum. The spectrum of this curve falls off rapidly, so plot only the first 20 points plus the DC term and plot them as discrete points, not as a line plot. As always scale the frequency axis correctly. Note that every other frequency point is zero as predicted by the symmetry of this signal.

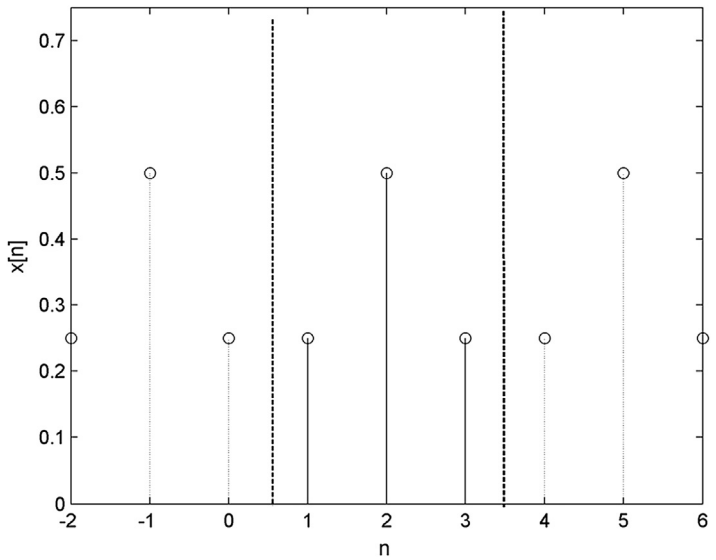
13. Generate the signal used in Problem 4 with two different time shifts. One version should be as shown and the other shifted by $T/2$. For both signals, take the Fourier transform using `fft` and plot only the first 20 values (plus the DC term) as discrete points and use `subplot` to place the magnitude and phase plots side by side (you could put all four plots together). Unwrap the phase plot and scale by $360/(2\pi)$ so it is in degrees. Note the similarities and differences between the two frequency domain representations. (Hint: The hardest part of this problem is constructing the signals, particularly the first signal. To make the first signal, the one shown in Problem 4, merge two linear segments using MATLAB brackets. Care must be taken

when defining the intercepts of the second segment. The second signal is easier, as it is just a straight line over the entire period with a slope of 2 and an intercept of −1.)

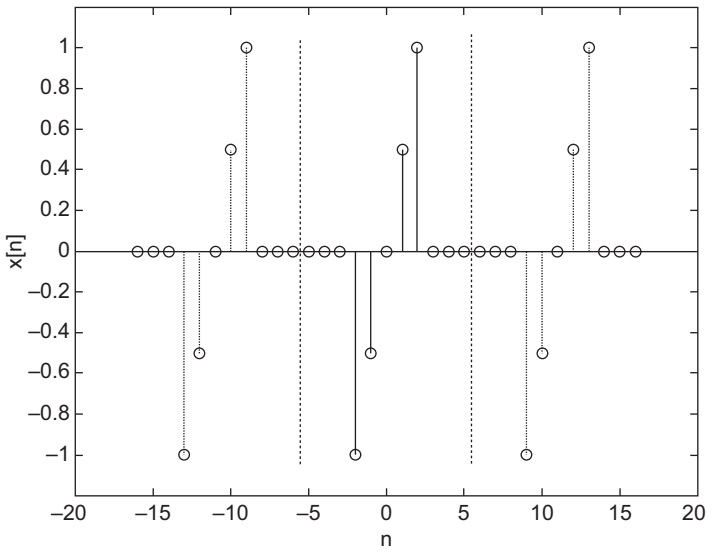
14. Generate two versions of the signal used in Problems 4 and 13. One version should be as shown and the other its inverse. For both signals, take the Fourier transform using `fft` and plot only the first 20 values (plus the DC term) as discrete points. Use `subplot` to place the magnitude and phase plots side by side (you could put all four plots together). Scale the phase plot by $360/(2\pi)$ so it is in degrees. The two signals have very similar frequency domain characteristics, but there is a difference. Can you find the difference?

15. Use the `fft` routine to find the spectrum of a waveform consisting of two sinusoids at 100 and 200 Hz similar to Problem 9. Again generate the signal with $N = 512$ and $f_s = 1$ kHz. Plot the full magnitude spectrum (all N points). Repeat using sinusoidal frequencies of 200 and 900 Hz and plot. Use `subplot` to plot the two spectra side by side for easy comparison. Can you explain the results? If not, no problem, as we discuss this phenomenon in the next chapter.

16. Find the DFT for the signal shown below using the manual approach of Example 3.7. The dashed lines indicate one period.



17. Find the DFT for the signal shown below using the manual approach of Example 3.7. The dashed lines indicate one period. (Hint: You can just ignore the zeros on either side except for the calculation of N.)



18. Plot the magnitude and phase components of the ECG signal found as variable `ecg` in file `ECG.mat`. Plot only the nonredundant points and do not plot the DC component (i.e., the first point in the Fourier series.) Also plot the time function and correctly label and scale the time and frequency axes. The sampling frequency was 125 Hz. Based on the magnitude plot, what is the bandwidth of the ECG signal, i.e., the range of frequencies that have some energy?

19. The data file `pulses.mat` contains three signals: `x1`, `x2`, and `x3`. These signals are all 1 s in length and were sampled at 500 Hz. Plot the three signals and show that each contains a single 40-ms pulse, but at three different delays: 0, 100, and 200 ms. Calculate and plot the spectra for the three signals superimposed on a single magnitude and single phase plot. Plot only the first 20 points as discrete points plus the DC term using a different color for each signal's spectra. Apply the `unwrap` routine to the phase data and plot in degrees. Note that the three magnitude plots are identical, whereas the phase plots are all straight lines but with radically different slopes.

20. Load the file `chirp.mat`, which contains a sinusoidal signal, `x`, that increases its frequency linearly over time. The sampling frequency of this signal is 5000 Hz. This type of signal is called a "chirp signal" because of the sound it makes when played through an audio system. If you have an audio system, you can listen to this signal after loading the file using the MATLAB command: `sound(x,5000);`. Take the Fourier transform of this signal and plot magnitude and phase (no DC term). Note that the magnitude spectrum shows the range of frequencies that are present but there is no information on the timing of those frequencies. Actually, information on signal timing

is contained in the phase plot but, as you can see, this plot is not easy to interpret. Advanced signal processing methods known as "time–frequency methods" are necessary to recover the timing information. These methods are briefly covered in the next chapter.

21. Load the file `ECG_1min.mat` that contains 1 min of ECG data in variable `ecg`. Take the Fourier transform. Plot both the magnitude and phase (unwrapped) spectrum up to 20 Hz and do not include the DC term. Find the average heart rate using the strategy found in Example 3.12. The sample frequency is 250 Hz. [Note: The spectrum will have a number of peaks; the largest low-frequency peak (which is also the largest overall) corresponds to the cardiac cycle.]

22. Construct two waveforms: the waveform in Problem 7 (e^{-at}) and a short pulse. For the first waveform make $a = 2$ and the period equal 2 s. Make the pulse as short as possible, i.e., one point. Make both waveforms 1024 points long and assume $f_s = 400$ Hz. Take the Fourier transform of both waveforms. Use `ifft` to reconstruct the Problem 7 waveform. Then multiply the two complex spectra together (point by point) and use `ifft` to get the time domain response. The two waveforms should look the same. Can you explain why the multiplication of the two complex spectra had no effect? If not, no worries, as we revisit the issue in Chapter 5.

23. Construct two waveforms. For the first, use `sig_noise` to produce a 2 Hz sinusoid, a signal to noise ratio of −3 dB, and $N = 2048$ (`sig_noise(2,-3,N);`). Recall `sig_noise` assumes an f_s of 1 kHz. The other signal should be a pulse having the same length as the noisy signal and a width of 10 samples. Take the Fourier transform of both waveforms. As in Problem 22, multiply the two complex spectra together (point by point) and use `ifft` to get the time domain response of the result. Plot the time domain of the original noisy signal and the signal produced by the frequency domain multiplication. Note that the latter has less noise. Again we revisit the concepts behind the results in Chapter 5. In the meantime you know one way of reducing noise in a signal.