CHAPTER 9

Applications of the DFT

The Discrete Fourier Transform (DFT) is one of the most important tools in Digital Signal Processing. This chapter discusses three common ways it is used. First, the DFT can calculate a signal's *frequency spectrum*. This is a direct examination of information encoded in the frequency, phase, and amplitude of the component sinusoids. For example, human speech and hearing use signals with this type of encoding. Second, the DFT can find a system's frequency response from the system's impulse response, and vice versa. This allows systems to be analyzed in the *frequency domain*, just as convolution allows systems to be analyzed in the *time domain*. Third, the DFT can be used as an intermediate step in more elaborate signal processing techniques. The classic example of this is *FFT convolution*, an algorithm for convolving signals that is hundreds of times faster than conventional methods.

Spectral Analysis of Signals

It is very common for information to be encoded in the sinusoids that form a signal. This is true of naturally occurring signals, as well as those that have been created by humans. Many things oscillate in our universe. For example, speech is a result of vibration of the human vocal cords; stars and planets change their brightness as they rotate on their axes and revolve around each other; ship's propellers generate periodic displacement of the water, and so on. The *shape* of the time domain waveform is not important in these signals; the key information is in the *frequency*, *phase* and *amplitude* of the component sinusoids. The DFT is used to extract this information.

An example will show how this works. Suppose we want to investigate the sounds that travel through the ocean. To begin, a microphone is placed in the water and the resulting electronic signal amplified to a reasonable level, say a few volts. An analog low-pass filter is then used to remove all frequencies above 80 hertz, so that the signal can be digitized at 160 samples per second. After acquiring and storing several thousand samples, what next?

章

9

DFT的应用

离散傅里叶变换（DFT）是数字信号处理领域最重要的工具之一。本章将探讨其三种常见应用方式。首先，DFT能够计算信号的*频率谱*，这相当于直接解析信号中正弦分量的频率、相位和振幅所编码的信息。例如，人类语音和听觉系统正是通过这种编码方式传递信息。其次，DFT可从系统的脉冲响应推导出其频率响应，反之亦然。这使得系统分析可以像通过卷积运算在*时间域*分析系统那样，在*频域*进行分析。第三，DFT可作为更复杂信号处理技术的中间步骤。经典案例是*FFT 卷积*算法，其信号卷积速度比传统方法快数百倍。

信号的频谱分析

信息通常以正弦波形式编码，这种编码方式在自然界和人工信号中都十分常见。宇宙中处处可见振荡现象：人类的声带振动产生语音，恒星行星自转公转时改变亮度，船舶螺旋桨周期性推动海水位移，诸如此类。这些信号的关键信息并不在于时域波形的*形状*，而在于正弦波分量的*频率*、*相位*和*幅值*。离散傅里叶变换（DFT）正是用于提取这些关键信息的技术。

我们通过一个实例来说明这一过程。假设要研究海洋中的声音传播特性，首先将麦克风置于水中，将产生的电子信号放大至合理水平（例如几伏特）。随后使用模拟低通滤波器去除所有高于80赫兹的频率，以便以每秒160个采样点的速率进行数字化处理。在采集并存储数千个样本后，接下来该做什么？

The first thing is to simply *look* at the data. Figure 9-1a shows 256 samples from our imaginary experiment. All that can be seen is a noisy waveform that conveys little information to the human eye. For reasons explained shortly, the next step is to multiply this signal by a smooth curve called a **Hamming window**, shown in (b). (Chapter 16 provides the equations for the Hamming and other windows; see Eqs. 16-1 and 16-2, and Fig. 16-2a). This results in a 256 point signal where the samples near the ends have been reduced in amplitude, as shown in (c).

Taking the DFT, and converting to polar notation, results in the 129 point frequency spectrum in (d). Unfortunately, this also looks like a noisy mess. This is because there is not enough information in the original 256 points to obtain a well behaved curve. Using a longer DFT does nothing to help this problem. For example, if a 2048 point DFT is used, the frequency spectrum becomes 1025 samples long. Even though the original 2048 points contain more information, the greater number of samples in the spectrum dilutes the information by the same factor. Longer DFTs provide better frequency resolution, but the same noise level.

The answer is to use more of the original signal in a way that doesn't increase the number of points in the frequency spectrum. This can be done by breaking the input signal into many 256 point *segments*. Each of these segments is multiplied by the Hamming window, run through a 256 point DFT, and converted to polar notation. The resulting frequency spectra are then *averaged* to form a single 129 point frequency spectrum. Figure (e) shows an example of averaging 100 of the frequency spectra typified by (d). The improvement is obvious; the noise has been reduced to a level that allows interesting features of the signal to be observed. Only the *magnitude* of the frequency domain is averaged in this manner; the *phase* is usually discarded because it doesn't contain useful information. The random noise reduces in proportion to the *square-root* of the number of segments. While 100 segments is typical, some applications might average *millions* of segments to bring out weak features.

There is also a second method for reducing spectral noise. Start by taking a very long DFT, say 16,384 points. The resulting frequency spectrum is high resolution (8193 samples), but very noisy. A low-pass digital filter is then used to *smooth* the spectrum, reducing the noise at the expense of the resolution. For example, the simplest digital filter might average 64 adjacent samples in the original spectrum to produce each sample in the filtered spectrum. Going through the calculations, this provides about the same noise and resolution as the first method, where the 16,384 points would be broken into 64 segments of 256 points each.

Which method should you use? The first method is easier, because the digital filter isn't needed. The second method has the *potential* of better performance, because the digital filter can be tailored to optimize the trade-off between noise and resolution. However, this improved performance is seldom worth the trouble. This is because both noise and resolution can be improved by using *more data* from the input signal. For example,

首先需要*看*数据。图9-1a展示了我们假设实验中的256个样本。目前只能看到一个对人眼信息量有限的噪声波形。基于后文将解释的原因，下一步需要将该信号与称为**哈明窗**的平滑曲线相乘（如图(b)所示）。（第16章提供了哈明窗及其他窗口的计算公式，详见公式16-1、16-2及图16-2a）。经过处理后得到的256点信号中，末端附近的样本振幅有所减弱，如图(c)所示。

采用DFT并转换为极坐标表示后，得到图(d)中的129点频谱。但令人遗憾的是，这个频谱看起来也像是一团乱麻。这是因为原始的256个数据点信息量不足，无法获得一条平滑的曲线。延长DFT的长度并不能解决这个问题。例如，若使用2048点DFT，频谱长度会变成1025个采样点。虽然原始的2048个数据点包含更多信息，但频谱中采样点数量的增加反而使信息量按相同比例稀释。更长的DFT虽然能提供更好的频率分辨率，但噪声水平却依然如故。

解决方法是通过不增加频谱点数的方式充分利用原始信号。具体实现方式是将输入信号分割成多个256点的*分段*，每个分段先经过汉明窗处理，再进行256点离散傅里叶变换，最后转换为极坐标表示。将这些频谱进行*平均处理*后，最终得到一个129点的综合频谱。图(e)展示了对(d)中典型频谱进行100次平均处理的示例。改进效果显著：噪声被有效抑制到足以清晰呈现信号特征的水平。在此过程中，仅对频域的*幅值*进行平均处理，而通常会舍弃包含无用信息的*相位*。随机噪声的衰减程度与分段数量的*平方根*成正比。虽然100个分段是常规处理量，但某些应用场景可能需要平均处理*数百万*个分段才能有效提取微弱特征。

降低频谱噪声还有第二种方法。首先进行超长时域傅里叶变换（DFT），比如采集16,384个数据点。虽然生成的频谱分辨率高达8193个采样点，但噪声水平极高。此时可采用低通数字滤波器对频谱进行*平滑处理*，虽然会牺牲部分分辨率，但能有效降低噪声。例如，最基础的数字滤波器会取原始频谱中相邻64个采样的平均值，生成滤波后的每个采样点。经过计算验证，这种方法在噪声和分辨率方面与第一种方法效果相当——即把16,384个数据点拆分成64段，每段256个点。

您该选择哪种方法？第一种方法更简单，因为无需使用数字滤波器。第二种方法*有潜力*提升性能，因为数字滤波器可以针对噪声与分辨率之间的平衡进行优化。不过这种性能提升往往不值得费这个劲，因为通过增加输入信号*的数据量*，既能降低噪声又能提高分辨率。例如，

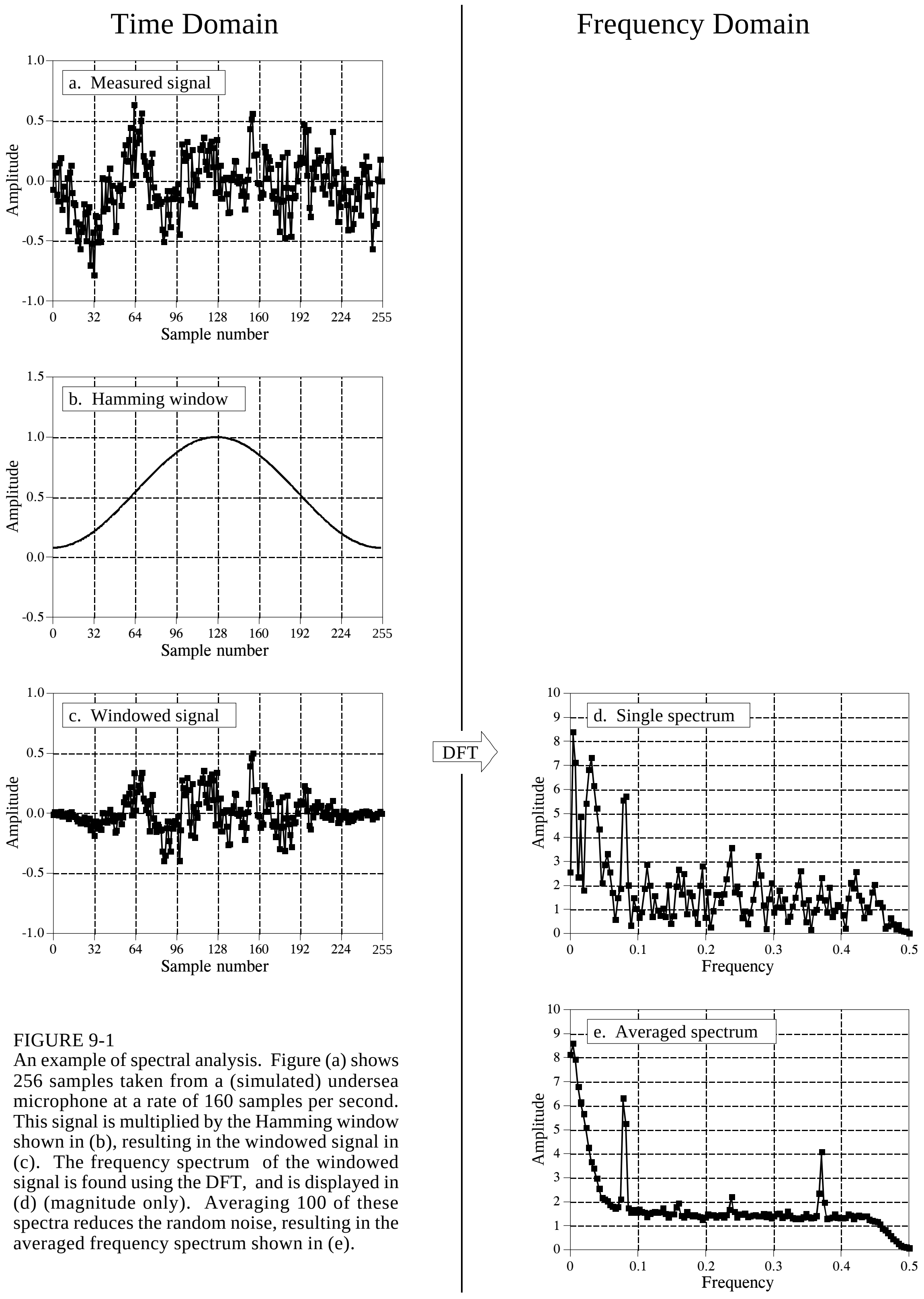
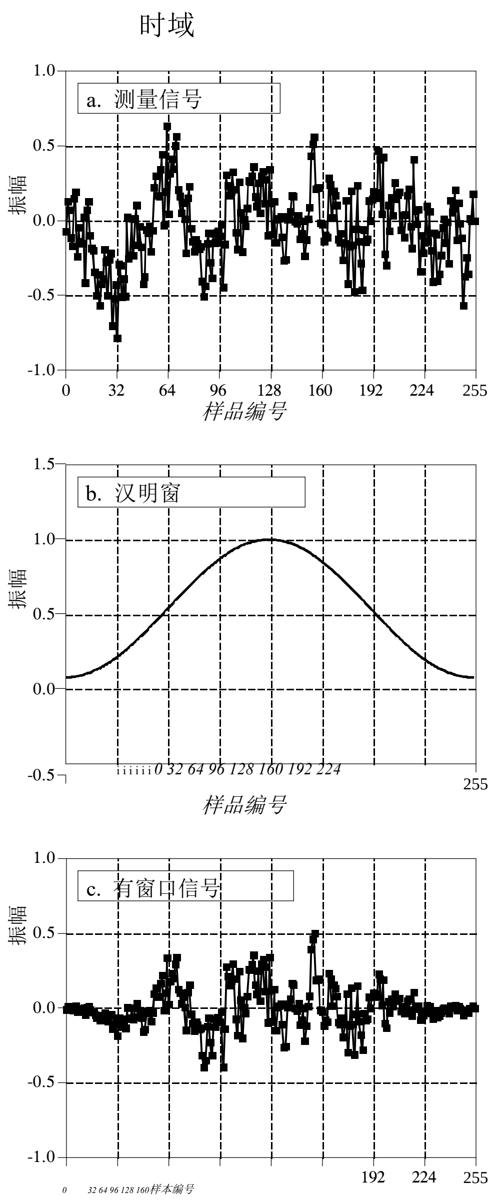


FIGURE 9-1
An example of spectral analysis. Figure (a) shows 256 samples taken from a (simulated) undersea microphone at a rate of 160 samples per second. This signal is multiplied by the Hamming window shown in (b), resulting in the windowed signal in (c). The frequency spectrum of the windowed signal is found using the DFT, and is displayed in (d) (magnitude only). Averaging 100 of these spectra reduces the random noise, resulting in the averaged frequency spectrum shown in (e).



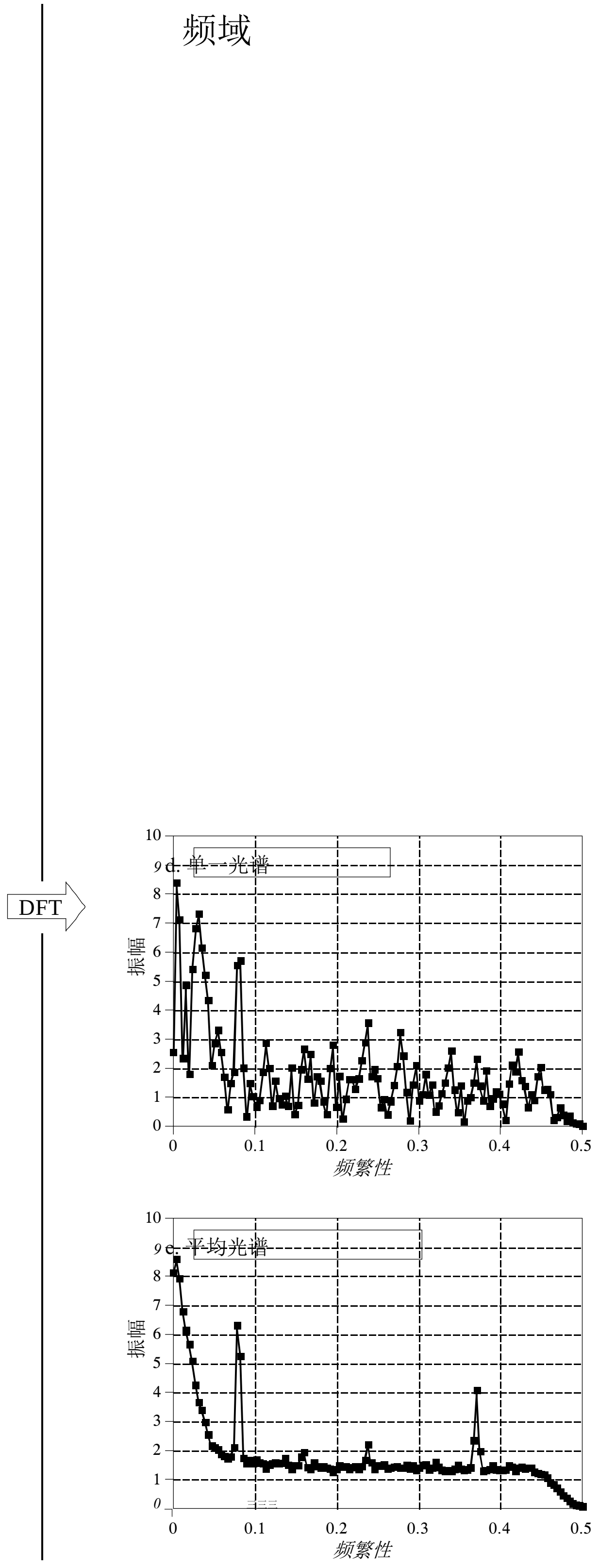


图9-1

频谱分析示例图解。图(a)展示了从海底麦克风（模拟信号）采集的256个样本，采样率为每秒160个。该信号经(b)所示汉明窗处理后，得到(c)中的窗化信号。通过离散傅里叶变换（DFT）计算窗化信号的频谱，其幅值如(d)所示。对100个频谱进行平均处理可有效降低随机噪声，最终得到(e)所示的平均频谱。

imagine breaking the acquired data into 10,000 segments of 16,384 samples each. This resulting frequency spectrum is high resolution (8193 points) *and* low noise (10,000 averages). Problem solved! For this reason, we will only look at the averaged segment method in this discussion.

Figure 9-2 shows an example spectrum from our undersea microphone, illustrating the features that commonly appear in the frequency spectra of acquired signals. Ignore the sharp peaks for a moment. Between 10 and 70 hertz, the signal consists of a relatively flat region. This is called **white noise** because it contains an equal amount of all frequencies, the same as white light. It results from the noise on the time domain waveform being *uncorrelated* from sample-to-sample. That is, knowing the noise value present on any one sample provides no information on the noise value present on any other sample. For example, the random motion of electrons in electronic circuits produces white noise. As a more familiar example, the sound of the water spray hitting the shower floor is white noise. The white noise shown in Fig. 9-2 could be originating from any of several sources, including the analog electronics, or the ocean itself.

Above 70 hertz, the white noise rapidly decreases in amplitude. This is a result of the roll-off of the antialias filter. An ideal filter would pass all frequencies below 80 hertz, and block all frequencies above. In practice, a perfectly sharp cutoff isn't possible, and you should expect to see this gradual drop. If you don't, suspect that an aliasing problem is present.

Below about 10 hertz, the noise rapidly increases due to a curiosity called **1/f noise** (one-over-f noise). 1/f noise is a mystery. It has been measured in very diverse systems, such as traffic density on freeways and electronic noise in transistors. It probably could be measured in all systems, if you look low enough in frequency. In spite of its wide occurrence, a general theory and understanding of 1/f noise has eluded researchers. The cause of this noise can be identified in some specific systems; however, this doesn't answer the question of why 1/f noise is everywhere. For common analog electronics and most physical systems, the transition between white noise and 1/f noise occurs between about 1 and 100 hertz.

Now we come to the sharp peaks in Fig. 9-2. The easiest to explain is at 60 hertz, a result of electromagnetic interference from commercial electrical power. Also expect to see smaller peaks at multiples of this frequency (120, 180, 240 hertz, etc.) since the power line waveform is not a *perfect* sinusoid. It is also common to find interfering peaks between 25-40 kHz, a favorite for designers of switching power supplies. Nearby radio and television stations produce interfering peaks in the megahertz range. Low frequency peaks can be caused by components in the system vibrating when shaken. This is called *microphonics*, and typically creates peaks at 10 to 100 hertz.

Now we come to the actual signals. There is a strong peak at 13 hertz, with weaker peaks at 26 and 39 hertz. As discussed in the next chapter, this is the frequency spectrum of a nonsinusoidal periodic waveform. The peak at 13 hertz is called the fundamental frequency, while the peaks at 26 and 39

想象一下将获得的数据分成10000个片段，每个片段有16384个样本。由此产生的频谱是高分辨率的（8193个点）*和*低噪声的（10000个平均值）。问题解决了！因此，在此讨论中，我们只考虑平均片段方法。

图9-2展示了我们海底麦克风采集的典型频谱，清晰呈现了信号频谱中常见的特征。暂且忽略那些尖锐的峰值，在10到70赫兹的频段内，信号呈现出相对平坦的区域。这种现象被称为**白噪声**，因为它包含所有频率的等量成分，就像白光一样。其成因在于时域波形上的噪声在采样之间呈现*不相关性*——也就是说，知道某个采样点的噪声值并不能推断其他采样点的噪声值。例如，电子电路中电子的随机运动会产生白噪声。更贴近生活的例子是，水花溅到淋浴底盘时发出的声响就是白噪声。图9-2所示的白噪声可能源自多种来源，包括模拟电子设备或海洋本身。

当频率超过70赫兹时，白噪声的振幅会迅速衰减。这是抗混叠滤波器滚降效应的结果。理想滤波器应能通过80赫兹以下的所有频率，并阻断80赫兹以上的频率。实际应用中，完全陡峭的截止频率难以实现，因此应预期出现这种渐进式衰减。若未观察到此现象，则需怀疑存在混叠问题。

当频率低于约10赫兹时，由于一种被称为**1/f噪声**（即1/f噪声）的特殊现象，噪声水平会急剧上升。这种1/f噪声至今仍是个未解之谜。研究者已在多种系统中观测到它，例如高速公路的车流密度和晶体管中的电子噪声。只要频率足够低，理论上所有系统都可能产生这种噪声。尽管1/f噪声普遍存在，但学界始终未能建立统一的理论解释。虽然在某些特定系统中可以识别出其成因，但这并不能解释为何这种噪声无处不在。对于常见的模拟电子设备和大多数物理系统而言，白噪声与1/f噪声之间的过渡频率通常在1到100赫兹之间。

现在我们来看图9-2中的尖锐峰值。最容易解释的是60赫兹处的峰值，这是商用电力电磁干扰造成的。由于电网波形并非*完美*正弦波，因此在该频率的倍数处（如120、180、240赫兹等）也会出现较小的峰值。在25-40千赫兹频段，开关电源设计者常会遇到干扰峰值，这正是他们钟爱的干扰频段。附近的广播电视台会在兆赫兹范围内产生干扰峰值。低频峰值可能由系统组件在受到震动时产生振动引起，这种现象称为*微动效应*，通常会在10至100赫兹处形成峰值。

现在我们来看实际信号。在13赫兹处存在一个显著峰值，26赫兹和39赫兹处则有较弱的峰值。如下一章所述，这是非正弦周期波形的频谱。13赫兹处的峰值称为基频，而26赫兹和39赫兹处的峰值则为次谐波。

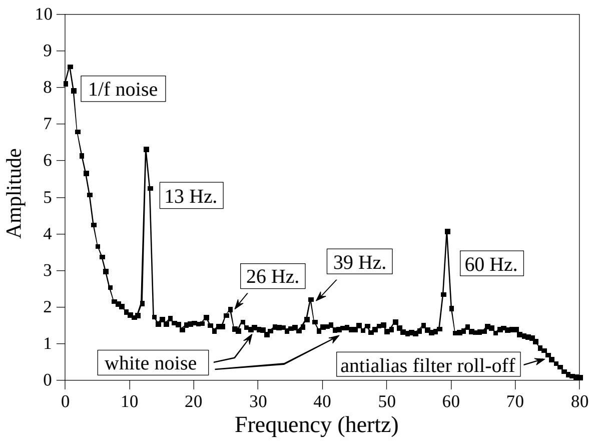


FIGURE 9-2
Example frequency spectrum. Three types of features appear in the spectra of acquired signals: (1) random noise, such as white noise and 1/f noise, (2) interfering signals from power lines, switching power supplies, radio and TV stations, microphonics, etc., and (3) real signals, usually appearing as a fundamental plus harmonics. This example spectrum (magnitude only) shows several of these features.

hertz are referred to as the second and third harmonic respectively. You would also expect to find peaks at other multiples of 13 hertz, such as 52, 65, 78 hertz, etc. You don't see these in Fig. 9-2 because they are buried in the white noise. This 13 hertz signal might be generated, for example, by a submarines's three bladed propeller turning at 4.33 revolutions per second. This is the basis of *passive* sonar, identifying undersea sounds by their frequency and harmonic content.

Suppose there are peaks very close together, such as shown in Fig. 9-3. There are two factors that limit the frequency resolution that can be obtained, that is, how close the peaks can be without merging into a single entity. The first factor is the length of the DFT. The frequency spectrum produced by an N point DFT consists of $N/2+1$ samples equally spaced between zero and one-half of the sampling frequency. To separate two closely spaced frequencies, the sample spacing must be *smaller* than the distance between the two peaks. For example, a 512 point DFT is sufficient to separate the peaks in Fig. 9-3, while a 128 point DFT is not.

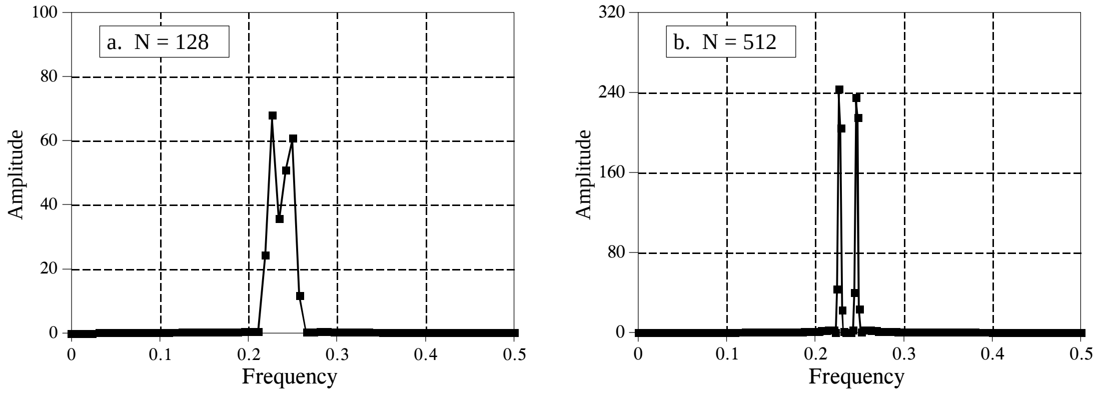


FIGURE 9-3
Frequency spectrum resolution. The longer the DFT, the better the ability to separate closely spaced features. In these example magnitudes, a 128 point DFT cannot resolve the two peaks, while a 512 point DFT can.

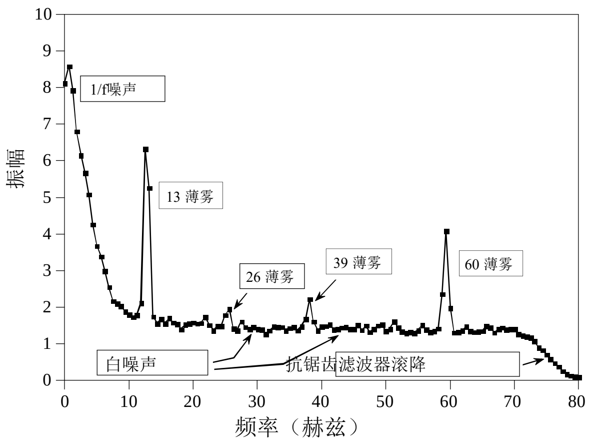


图9-2
示例频谱。采集信号的频谱中呈现三种特征：(1)随机噪声，如白噪声和1/f噪声；(2)来自电力线、开关电源、广播电视台、麦克风等的干扰信号；(3)真实信号，通常表现为基波及其谐波。该示例频谱（仅显示幅值）展示了其中几种特征。

赫兹信号分别对应第二和第三谐波。理论上还应存在其他13赫兹倍数的峰值，例如52、65、78赫兹等。但图9-2中未显示这些峰值，因为它们被白噪声掩盖。例如，潜艇三叶螺旋桨以每秒4.33转的转速运转时，可能产生这种13赫兹信号。这正是被动声呐技术的原理基础——通过分析水下声音的频率及其谐波成分来识别其来源。

假设存在非常接近的峰值，如图9-3所示。有两个因素限制了可获得的频率分辨率，即峰值在不合并为单一实体时能有多接近。第一个因素是DFT的长度。由*N*点DFT产生的频谱由*N*/2+1个等距采样点组成，这些采样点分布在采样频率的零点到一半之间。要分离两个紧密相邻的频率，采样间隔必须*小于*两个峰值之间的距离。例如，512点DFT足以分离图9-3中的峰值，而128点DFT则无法做到。

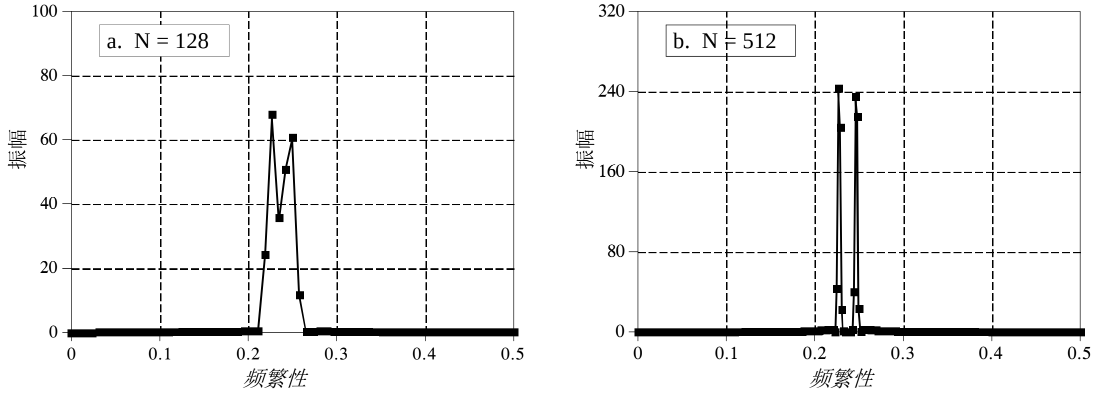


图9-3
频谱分辨率。DFT越长，分离紧密相邻特征的能力越好。在这些示例幅度中，128点DFT无法分辨两个峰值，而512点DFT则可以。

The second factor limiting resolution is more subtle. Imagine a signal created by adding two sine waves with only a slight difference in their frequencies. Over a short segment of this signal, say a few periods, the waveform will look like a *single* sine wave. The closer the frequencies, the longer the segment must be to conclude that more than one frequency is present. In other words, the *length* of the signal limits the frequency resolution. This is distinct from the first factor, because the *length of the input signal* does not have to be the same as the *length of the DFT*. For example, a 256 point signal could be padded with zeros to make it 2048 points long. Taking a 2048 point DFT produces a frequency spectrum with 1025 samples. The added zeros don't change the shape of the spectrum, they only provide more samples in the frequency domain. In spite of this very close sampling, the ability to separate closely spaced peaks would be only slightly better than using a 256 point DFT. When the DFT is the same length as the input signal, the resolution is limited about equally by these two factors. We will come back to this issue shortly.

Next question: What happens if the input signal contains a sinusoid with a frequency *between* two of the basis functions? Figure 9-4a shows the answer. This is the frequency spectrum of a signal composed of two sine waves, one having a frequency *matching* a basis function, and the other with a frequency *between* two of the basis functions. As you should expect, the first sine wave is represented as a single point. The other peak is more difficult to understand. Since it cannot be represented by a single sample, it becomes a peak with **tails** that extend a significant distance away.

The solution? Multiply the signal by a Hamming window before taking the DFT, as was previously discussed. Figure (b) shows that the spectrum is changed in three ways by using the window. First, the two peaks are made to look more alike. This is good. Second, the tails are greatly reduced.

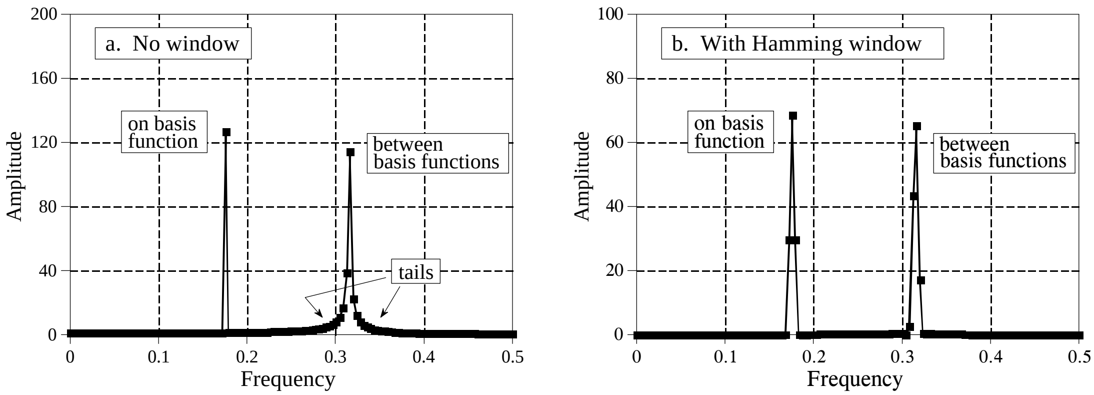


FIGURE 9-4
Example of using a window in spectral analysis. Figure (a) shows the frequency spectrum (magnitude only) of a signal consisting of two sine waves. One sine wave has a frequency exactly equal to a basis function, allowing it to be represented by a single sample. The other sine wave has a frequency *between* two of the basis functions, resulting in *tails* on the peak. Figure (b) shows the frequency spectrum of the same signal, but with a Blackman window applied before taking the DFT. The window makes the peaks look the same and reduces the tails, but broadens the peaks.

第二个影响分辨率的因素更为微妙。想象一个由两个频率仅有微小差异的正弦波叠加产生的信号。在该信号的短片段（比如几个周期）内，波形看起来就像*单一*正弦波。频率越接近，需要的片段就越长才能确认存在多个频率。换句话说，信号的*长度*限制了频率分辨率。这与第一个因素不同，因为*输入信号的长度*不必与*DFT的长度*相同。例如，一个256点的信号可以通过添加零点填充到2048点。进行2048点DFT后得到的频谱包含1025个样本。这些添加的零点不会改变频谱形状，只是在频域中提供了更多样本。尽管采样间隔非常接近，但分离紧密相邻峰值的能力仅比使用256点DFT稍好。当DFT与输入信号长度相同时，分辨率会受到这两个因素的同等程度限制。我们很快将回到这个问题上来。

接下来的问题是：如果输入信号包含一个频率*介于*两个基函数之间的正弦波会发生什么？图9-4a给出了答案。这是由两个正弦波组成的信号频谱，其中一个频率*与*一个基函数匹配，另一个频率*介于*两个基函数之间。正如预期的那样，第一个正弦波表现为一个单一的峰值。而另一个峰值则更难理解——由于无法用单个采样点表示，它形成了具有明显**尾部**的峰值，这些尾部延伸了相当长的距离。

解决方案？如先前讨论，先将信号乘以汉明窗再进行DFT。图(b)显示使用该窗后频谱发生三方面改变：首先，两个峰值呈现更接近的形态（此为优势）；其次，频谱尾部显著缩短。

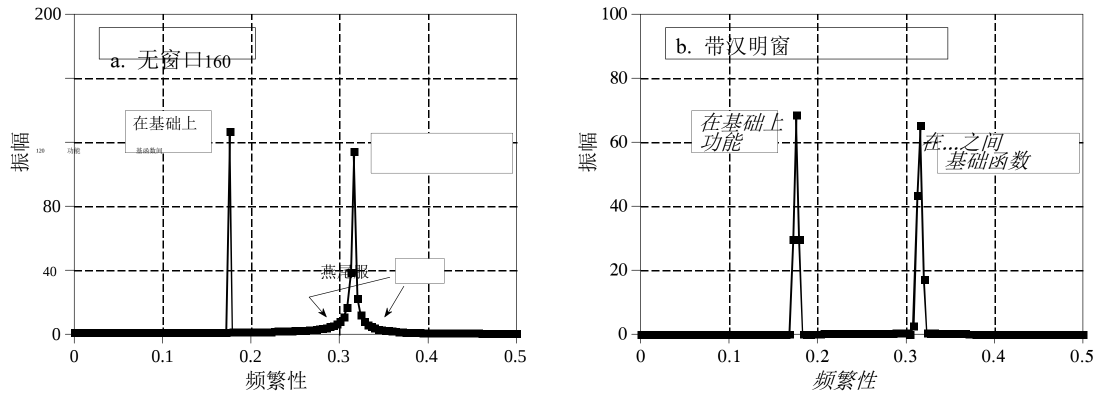


图9-4
频谱分析中使用窗函数的示例。图(a)展示了由两个正弦波组成的信号频谱（仅显示幅值）。其中一个正弦波的频率恰好等于某个基函数，因此可以用单个采样点表示；另一个正弦波的频率*介于*两个基函数之间，导致其峰值处出现*尾部*。图(b)展示了同一信号的频谱，但在进行离散傅里叶变换前应用了布莱克曼窗。该窗函数使峰值看起来更加均匀，同时减弱了尾部信号，但会略微展宽峰值。

This is also good. Third, the window reduces the resolution in the spectrum by making the peaks wider. This is bad. In DSP jargon, windows provide a tradeoff between *resolution* (the width of the peak) and *spectral leakage* (the amplitude of the tails).

To explore the theoretical aspects of this in more detail, imagine an infinitely long discrete sine wave at a frequency of 0.1 the sampling rate. The frequency spectrum of this signal is an infinitesimally narrow peak, with all other frequencies being zero. Of course, neither this signal nor its frequency spectrum can be brought into a digital computer, because of their infinite and infinitesimal nature. To get around this, we change the signal in two ways, both of which distort the true frequency spectrum.

First, we *truncate* the information in the signal, by multiplying it by a window. For example, a 256 point *rectangular window* would allow 256 points to retain their correct value, while all the other samples in the infinitely long signal would be set to a value of zero. Likewise, the Hamming window would *shape* the retained samples, besides setting all points outside the window to zero. The signal is still infinitely long, but only a finite number of the samples have a nonzero value.

How does this windowing affect the frequency domain? As discussed in Chapter 10, when two time domain signals are *multiplied*, the corresponding frequency domains are *convolved*. Since the original spectrum is an infinitesimally narrow peak (i.e., a delta function), the spectrum of the windowed signal is the spectrum of the window shifted to the location of the peak. Figure 9-5 shows how the spectral peak would appear using four different window options (If you need a refresher on dB, look ahead to Chapter 14). Figure 9-5a results from a rectangular window. Figures (b) and (c) result from using two popular windows, the Hamming and the Blackman (as previously mentioned, see Eqs. 16-1 and 16-2, and Fig. 16-2a for information on these windows).

As shown in Fig. 9-5, all these windows have degraded the original spectrum by broadening the peak and adding tails composed of numerous side lobes. This is an unavoidable result of using only a portion of the original time domain signal. Here we can see the tradeoff between the three windows. The Blackman has the widest main lobe (bad), but the lowest amplitude tails (good). The rectangular window has the narrowest main lobe (good) but the largest tails (bad). The Hamming window sits between these two.

Notice in Fig. 9-5 that the frequency spectra are continuous curves, not discrete samples. After windowing, the time domain signal is still infinitely long, even though most of the samples are zero. This means that the frequency spectrum consists of $\infty/2+1$ samples between 0 and 0.5, the same as a continuous line.

This brings in the second way we need to modify the time domain signal to allow it to be represented in a computer: *select N points from the signal*. These *N* points must contain all the nonzero points identified by the window, but may also include any number of the zeros. This has the effect

这同样具有优势。第三，该窗口会降低光谱中的分辨率。
使峰值变宽。这是不好的。在DSP术语中，窗口在*分辨率*（峰值宽度）和*频谱泄漏*（尾部幅度）之间提供了一种权衡。

为了更深入地探讨这一理论问题，我们可以设想一个无限长的离散正弦波，其频率为采样率的0.1倍。该信号的频谱呈现一个无限小的窄峰，其他频率均为零。当然，由于信号本身具有无限性和无限小的特性，无论是信号还是其频谱都无法被数字计算机处理。为解决这一问题，我们通过两种方式对信号进行处理，但都会导致真实频谱发生畸变。

首先，我们通过乘以窗函数对信号信息进行*截断*处理。例如，使用256点*矩形窗*时，前256个采样点会保留原始值，而无限长信号中的其余采样点则被置零。同样地，汉明窗除了将窗口外的所有点置零外，还会对保留的采样点进行*形态化*处理。虽然信号长度依然无限，但仅有有限数量的采样点会保持非零值。

这种窗函数处理会对频域产生怎样的影响？正如第十章所述，当两个时域信号进行*相乘*时，对应的频域信号会进行*卷积*运算。由于原始频谱是无限窄的尖峰（即δ函数），经过窗函数处理的信号频谱就相当于将窗函数平移到了该尖峰的位置。图9-5展示了四种不同窗函数选项下频谱峰值的呈现效果（若需要复习分贝单位，可参考第14章）。图9-5a展示了矩形窗函数的效果，图(b)和(c)则分别对应汉明窗与布莱克曼窗的处理结果（关于这两种窗函数的详细信息，可参考前文公式16-1、16-2及图16-2a）。

如图9-5所示，所有这些窗函数都通过展宽峰值并添加由大量旁瓣构成的尾部，导致原始频谱质量下降。这是仅使用原始时域信号部分不可避免的结果。通过对比三种窗函数的特性，我们可以发现它们之间的权衡关系：布莱克曼窗的主瓣最宽（缺点），但尾部振幅最低（优点）；矩形窗的主瓣最窄（优点），但尾部振幅最大（缺点）；而汉明窗则介于两者之间。

注意图9-5中，频谱是连续曲线，而非离散样本。经过窗函数处理后，时域信号仍然无限长，尽管大部分样本为零。这意味着频谱由0到0.5区间内的∞/2+1个样本组成，与连续线相同。

这就引出了我们需要修改时域信号的第二种方法，以便在计算机中表示它：*从信号中选择N个点*。这些*N*个点必须包含窗口识别的所有非零点，但也可能包含任意数量的零点。这会产生这样的效果

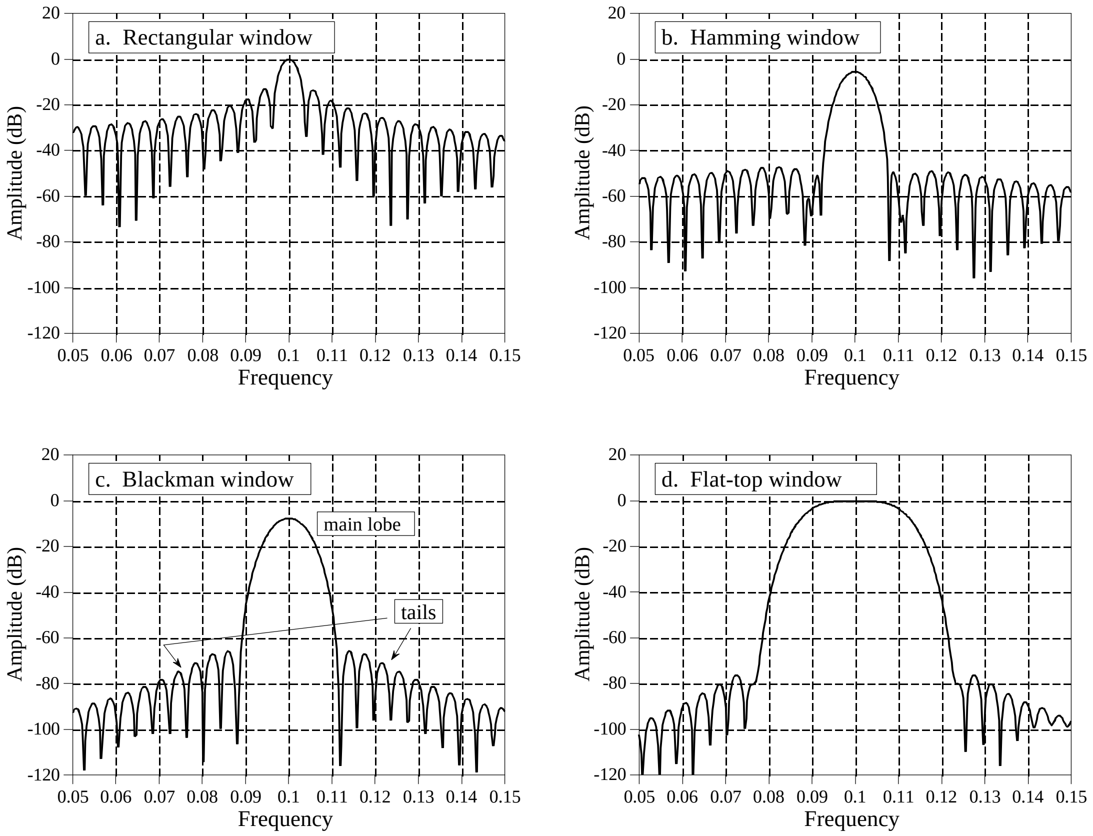


FIGURE 9-5
Detailed view of a spectral peak using various windows. Each peak in the frequency spectrum is a central lobe surrounded by tails formed from side lobes. By changing the window shape, the amplitude of the side lobes can be reduced at the expense of making the main lobe wider. The rectangular window, (a), has the narrowest main lobe but the largest amplitude side lobes. The Hamming window, (b), and the Blackman window, (c), have lower amplitude side lobes at the expense of a wider main lobe. The flat-top window, (d), is used when the amplitude of a peak must be accurately measured. These curves are for 255 point windows; longer windows produce proportionately narrower peaks.

of *sampling* the frequency spectrum's continuous curve. For example, if *N* is chosen to be 1024, the spectrum's continuous curve will be sampled 513 times between 0 and 0.5. If *N* is chosen to be much larger than the window length, the samples in the frequency domain will be close enough that the peaks and valleys of the continuous curve will be preserved in the new spectrum. If *N* is made the same as the window length, the fewer number of samples in the spectrum results in the regular pattern of peaks and valleys turning into irregular tails, depending on where the samples happen to fall. This explains why the two peaks in Fig. 9-4a do not look alike. Each peak in Fig 9-4a is a *sampling* of the underlying curve in Fig. 9-5a. The presence or absence of the tails depends on where the samples are taken in relation to the peaks and valleys. If the sine wave exactly matches a basis function, the samples occur exactly at the valleys, eliminating the tails. If the sine wave is between two basis functions, the samples occur somewhere along the peaks and valleys, resulting in various patterns of tails.

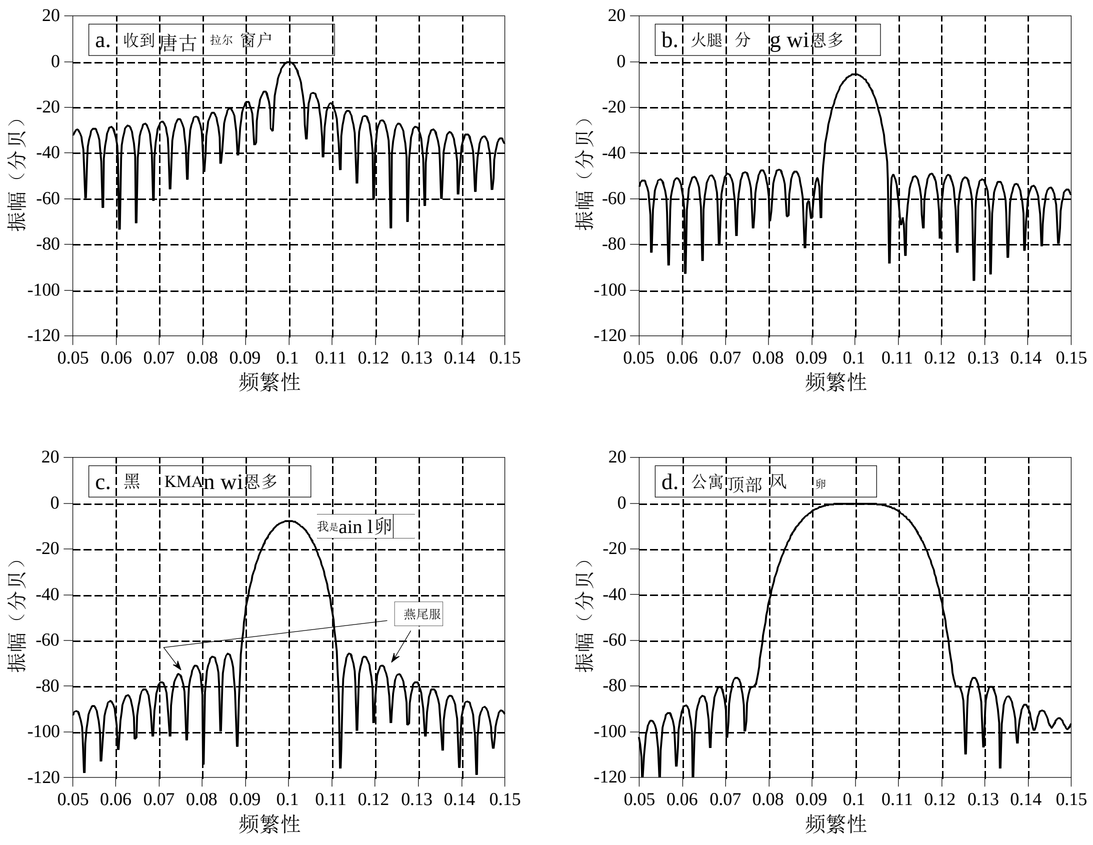


图9-5
通过不同窗口形状对频谱峰值进行详细观察。频谱中的每个峰值都由主瓣和侧瓣构成，主瓣位于中央，两侧延伸出尾瓣。改变窗口形状时，可以通过加宽主瓣来降低侧瓣的振幅。矩形窗口(a)的主瓣最窄，但侧瓣振幅最大；汉明窗口(b)和布莱克曼窗口(c)的主瓣较宽，但侧瓣振幅较低；平顶窗口(d)则用于需要精确测量峰值振幅的情况。这些曲线均基于255点窗口绘制，窗口越长，峰值会相应变窄。

关于*采样* 频率谱的连续曲线。例如，若选择*N*为1024，频率谱的连续曲线将在0到0.5区间内被采样513次。当N远大于窗长时，频域中的采样点会足够密集，使得连续曲线的峰谷特征能在新频谱中得以保留。若N与窗长相等，频谱中采样点数量减少会导致原本规律的峰谷模式因采样位置不同而形成不规则的尾部。这解释了图9-4a中两个峰值为何形态迥异——图9-4a中的每个峰值都是图9-5a中基础曲线的*采样* 结果。尾部的有无取决于采样点相对于峰谷的位置：若正弦波恰好匹配基函数，采样点会精确落在谷值处，从而消除尾部；若正弦波位于两个基函数之间，采样点会分布在峰谷之间，从而形成各种不同的尾部形态。

This leads us to the **flat-top window**, shown in Fig. 9-5d. In some applications the *amplitude* of a spectral peak must be measured very accurately. Since the DFT's frequency spectrum is formed from samples, there is nothing to guarantee that a sample will occur exactly at the top of a peak. More than likely, the nearest sample will be slightly off-center, giving a value lower than the true amplitude. The solution is to use a window that produces a spectral peak with a *flat top*, insuring that one or more of the samples will always have the correct peak value. As shown in Fig. 9-5d, the penalty for this is a very broad main lobe, resulting in poor frequency resolution.

As it turns out, the shape we want for a flat-top window is exactly the same shape as the filter kernel of a low-pass filter. We will discuss the theoretical reasons for this in later chapters; for now, here is a cookbook description of how the technique is used. Chapter 16 discusses a low-pass filter called the *windowed-sinc*. Equation 16-4 describes how to generate the filter kernel (which we want to use as a window), and Fig. 16-4a illustrates the typical shape of the curve. To use this equation, you will need to know the value of two parameters: M and f_c. These are found from the relations: $M = N-2$, and $f_c = s/N$, where N is the length of the DFT being used, and s is the number of samples you want on the flat portion of the peak (usually between 3 and 5). Table 16-1 shows a program for calculating the filter kernel (our window), including two subtle features: the normalization constant, K, and how to avoid a divide-by-zero error on the center sample. When using this method, remember that a DC value of *one* in the time domain will produce a peak of amplitude *one* in the frequency domain. However, a sinusoid of amplitude *one* in the time domain will only produce a spectral peak of amplitude *one-half*. (This is discussed in the last chapter: *Synthesis, Calculating the Inverse DFT*).

Frequency Response of Systems

Systems are analyzed in the *time domain* by using convolution. A similar analysis can be done in the *frequency domain*. Using the Fourier transform, every input signal can be represented as a group of cosine waves, each with a specified amplitude and phase shift. Likewise, the DFT can be used to represent every output signal in a similar form. This means that any linear system can be *completely* described by how it changes the amplitude and phase of cosine waves passing through it. This information is called the system's **frequency response.** Since both the impulse response and the frequency response contain complete information about the system, there must be a one-to-one correspondence between the two. Given one, you can calculate the other. The relationship between the impulse response and the frequency response is one of the foundations of signal processing: *A system's frequency response is the Fourier Transform of its impulse response*. Figure 9-6 illustrates these relationships.

Keeping with standard DSP notation, impulse responses use lower case variables, while the corresponding frequency responses are upper case. Since $h[\ ]$ is the common symbol for the impulse response, $H[\ ]$ is used for the frequency response. Systems are described in the time domain by convolution, that is:

这使我们得出如图9-5d所示的**平顶窗**。在某些应用中
频谱峰值的振幅 必须进行高精度测量。由于DFT的频谱由采样点构成，无法保证采样点会精确落在峰值顶部。实际情况中，最近的采样点往往会出现轻微偏移，导致测得值低于真实振幅。解决方法是采用具有平坦*顶部*的窗函数，确保至少有一个采样点能准确捕捉峰值。如图9-5d所示，这种处理方式的代价是主瓣过于宽泛，从而导致频率分辨率降低。

事实证明，我们想要的平顶窗形状与低通滤波器的滤波核形状完全相同。我们将在后续章节讨论这一现象的理论原因；目前，这里提供一个关于该技术应用的实用指南。第16章讨论了一种称为*窗函数sinc的*低通滤波器。公式16-4描述了如何生成滤波核（我们将其用作窗函数），图16-4a展示了该曲线的典型形状。要使用此公式，您需要知道两个参数的值：M和 f_c。这些参数通过以下关系式求得：M=N-2，以及 f_c=s/N，其中N是所用DFT的长度，s是峰值平缓部分所需的采样点数（通常为3到5个）。表16-1展示了一个计算滤波核（即我们的窗函数）的程序，其中包含两个微妙特征：归一化常数K，以及如何避免中心采样点的除零错误。使用此方法时，请记住时域中 一个 的直流值将在频域中产生 一个 的振幅峰值。然而，时域中振幅为 一个 的正弦波仅会产生振幅为 *一半* 的频谱峰值（这在最后一章*合成，计算逆DFT* 中讨论过）。

系统频率响应

系统分析通常通过卷积运算在*时间域* 进行，*频域分析也采用类似方法*。利用傅里叶变换，每个输入信号都可以表示为一组具有特定振幅和相位偏移的余弦波。同理，离散傅里叶变换（DFT）也能将输出信号以类似形式表示。这意味着任何线性系统都可以*完全* 通过其对通过系统的余弦波振幅和相位变化的描述来表征。这些信息被称为系统的**频率响应。**由于脉冲响应和频率响应都包含系统的完整信息，二者之间必然存在一一对应关系。已知其中一个，即可计算出另一个。脉冲响应与频率响应之间的关系是信号处理的基础之一：*系统的频率响应是其脉冲响应的傅里叶变换* 。图9-6展示了这些关系。

按照标准的DSP符号表示法，脉冲响应使用小写字母变量，而相应的频率响应则使用大写字母。由于h[ ]是脉冲响应的常用符号，H[ ]则用于频率响应。系统在时域中通过卷积来描述，即：

$x[n] * h[n] = y[n]$. In the frequency domain, the input spectrum is *multiplied* by the frequency response, resulting in the output spectrum. As an equation: $X[f] \times H[f] = Y[f]$. That is, *convolution* in the time domain corresponds to *multiplication* in the frequency domain.

Figure 9-7 shows an example of using the DFT to convert a system's impulse response into its frequency response. Figure (a) is the impulse response of the system. Looking at this curve isn't going to give you the slightest idea what the system does. Taking a 64 point DFT of this impulse response produces the frequency response of the system, shown in (b). Now the function of this system becomes obvious, it passes frequencies between 0.2 and 0.3, and rejects all others. It is a band-pass filter. The *phase* of the frequency response could also be examined; however, it is *more* difficult to interpret and *less* interesting. It will be discussed in upcoming chapters.

Figure (b) is very jagged due to the low number of samples defining the curve. This situation can be improved by **padding** the impulse response with zeros before taking the DFT. For example, adding zeros to make the impulse response 512 samples long, as shown in (c), results in the higher resolution frequency response shown in (d).

How much resolution can you obtain in the frequency response? The answer is: *infinitely* high, if you are willing to pad the impulse response with an *infinite* number of zeros. In other words, there is nothing limiting the frequency resolution except the length of the DFT. This leads to a very important concept. Even though the impulse response is a *discrete* signal, the corresponding frequency response is *continuous*. An N point DFT of the impulse response provides $N/2 + 1$ *samples* of this continuous curve. If you make the DFT longer, the resolution improves, and you obtain a better idea of

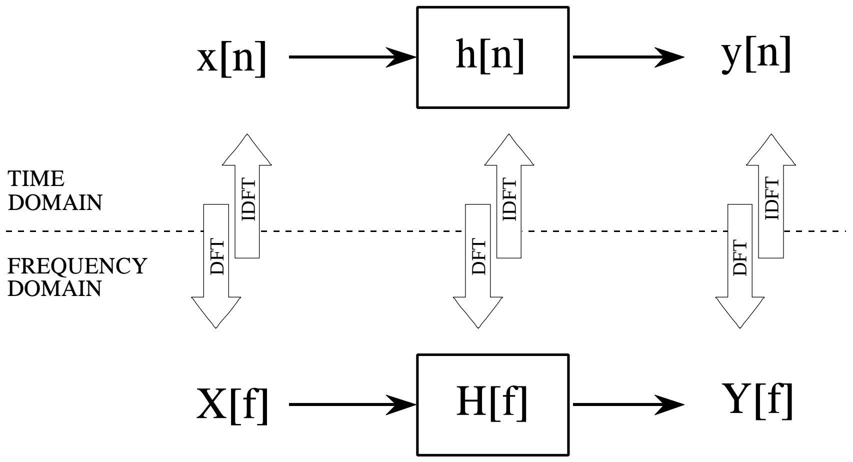


FIGURE 9-6
Comparing system operation in the time and frequency domains. In the time domain, an input signal is *convolved* with an impulse response, resulting in the output signal, that is, $x[n] * h[n] = y[n]$. In the frequency domain, an input spectrum is *multiplied* by a frequency response, resulting in the output spectrum, that is, $X[f] \times H[f] = Y[f]$. The DFT and the Inverse DFT relate the signals in the two domain.

x[*n*]t *h*[*n*] =*y*[*n*] 。在频域中，输入频谱被*乘以*频率响应，从而得到输出频谱。其方程为：*X*[*f*]× *H*[*f*] =*Y*[*f*] 。也就是说，时域中的*卷积*对应于频域中的*乘法*。

图9-7展示了使用DFT将系统脉冲响应转换为频率响应的示例。图(a)是系统的脉冲响应曲线，但仅凭观察这条曲线，你根本无法理解系统的实际工作原理。通过对该脉冲响应进行64点DFT变换，即可得到系统的频率响应曲线(b)。此时系统的功能特性变得清晰可见：它会通过0.2到0.3之间的频率范围，而完全阻断其他频率成分，这正是带通滤波器的典型特征。虽然也可以分析频率响应的*相位*特性，但其解读难度*更高*，且*趣味性*也相对较低。这部分内容将在后续章节中详细探讨。

图(b)的曲线非常锯齿状，这是由于定义曲线的样本数量较少。这种情况可以通过在进行DFT之前用零值**填充**脉冲响应来改善。例如，如图(c)所示，通过添加零值使脉冲响应长度达到512个样本，从而得到图(d)所示的高分辨率频率响应。

在频率响应中你能获得多少分辨率？答案是：*无限*高，只要你愿意用*无限*个零来填充冲激响应。换句话说，除了DFT的长度外，没有什么能限制频率分辨率。这引出了一个非常重要的概念。尽管冲激响应是一个*离散*信号，但对应的频率响应是*连续*的。对冲激响应进行*N*点DFT会提供这条连续曲线的*N*/2+1 *个样本*。如果你让DFT更长，分辨率就会提高，你就能获得更好的理解

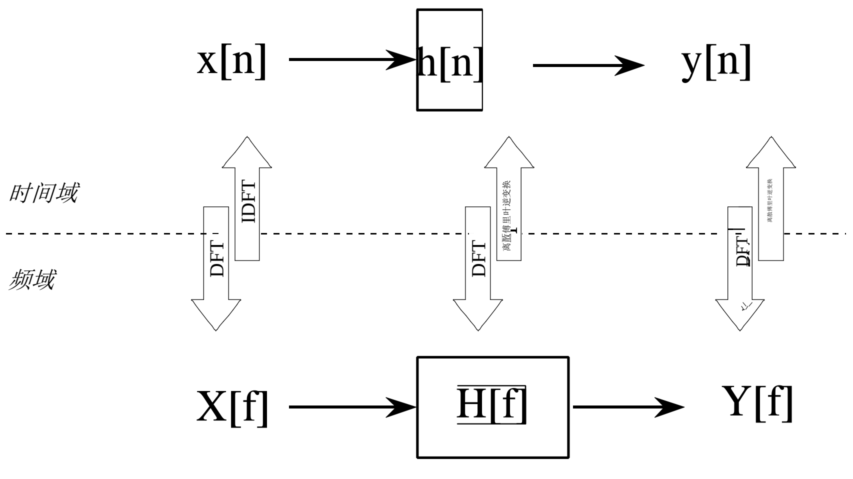


图9-6
比较时域与频域中的系统运行。在时域中，输入信号与冲激响应*卷积*，得到输出信号，即*x*[*n*]t *h*[*n*] =*y*[*n*]。在频域中，输入频谱与频率响应*相乘*，得到输出频谱，即*X*[*f*]× *H*[*f*] =*Y*[*f*]。DFT与逆DFT将两个域中的信号联系起来。

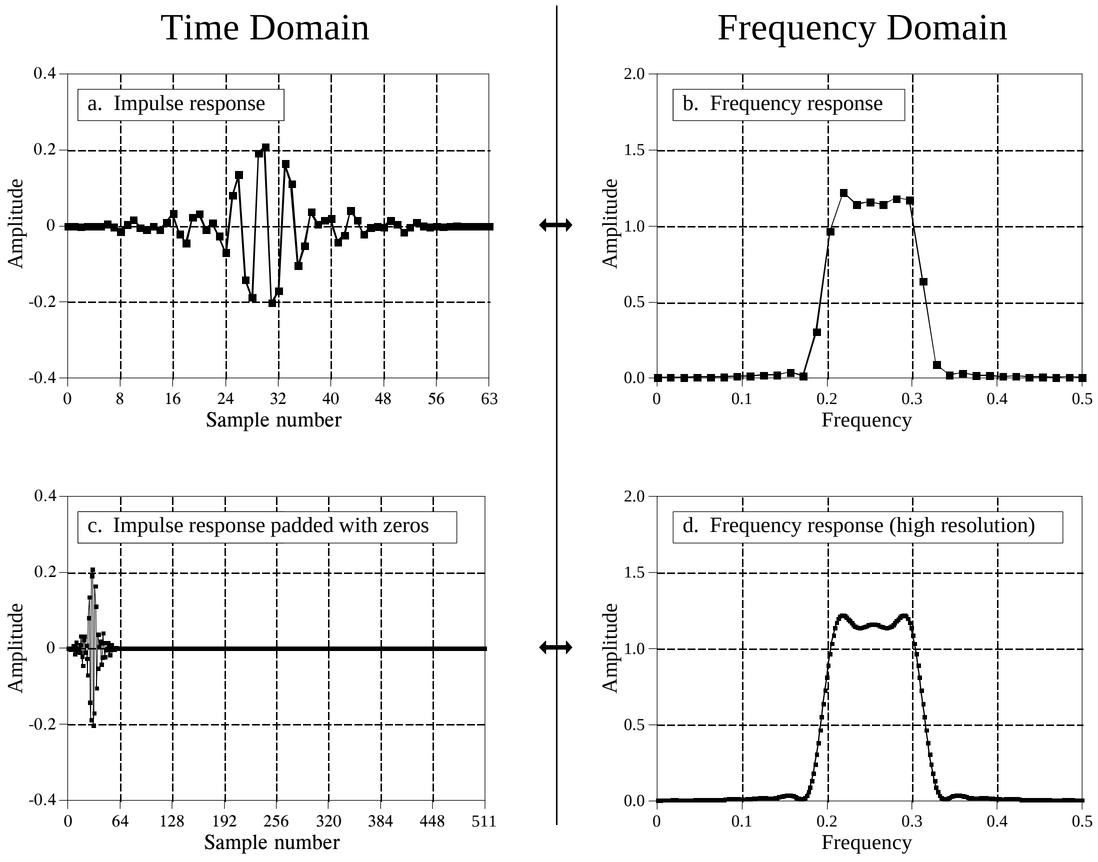


FIGURE 9-7
Finding the frequency response from the impulse response. By using the DFT, a system's impulse response, (a), can be transformed into the system's frequency response, (b). By padding the impulse response with zeros (c), higher resolution can be obtained in the frequency response, (d). Only the magnitude of the frequency response is shown in this example; discussion of the phase is postponed until the next chapter.

what the continuous curve looks like. Remember what the frequency response represents: amplitude and phase changes experienced by cosine waves as they pass through the system. Since the input signal can contain *any* frequency between 0 and 0.5, the system's frequency response *must* be a continuous curve over this range.

This can be better understood by bringing in another member of the Fourier transform family, the **Discrete Time Fourier Transform (DTFT)**. Consider an N sample signal being run through an N point DFT, producing an $N/2+1$ sample frequency domain. Remember from the last chapter that the DFT considers the time domain signal to be *infinitely long* and *periodic*. That is, the N points are repeated over and over from negative to positive infinity. Now consider what happens when we start to pad the time domain signal with an ever increasing number of zeros, to obtain a finer and finer sampling in the frequency domain. Adding zeros makes the period of the time domain *longer*, while simultaneously making the frequency domain samples *closer together*.

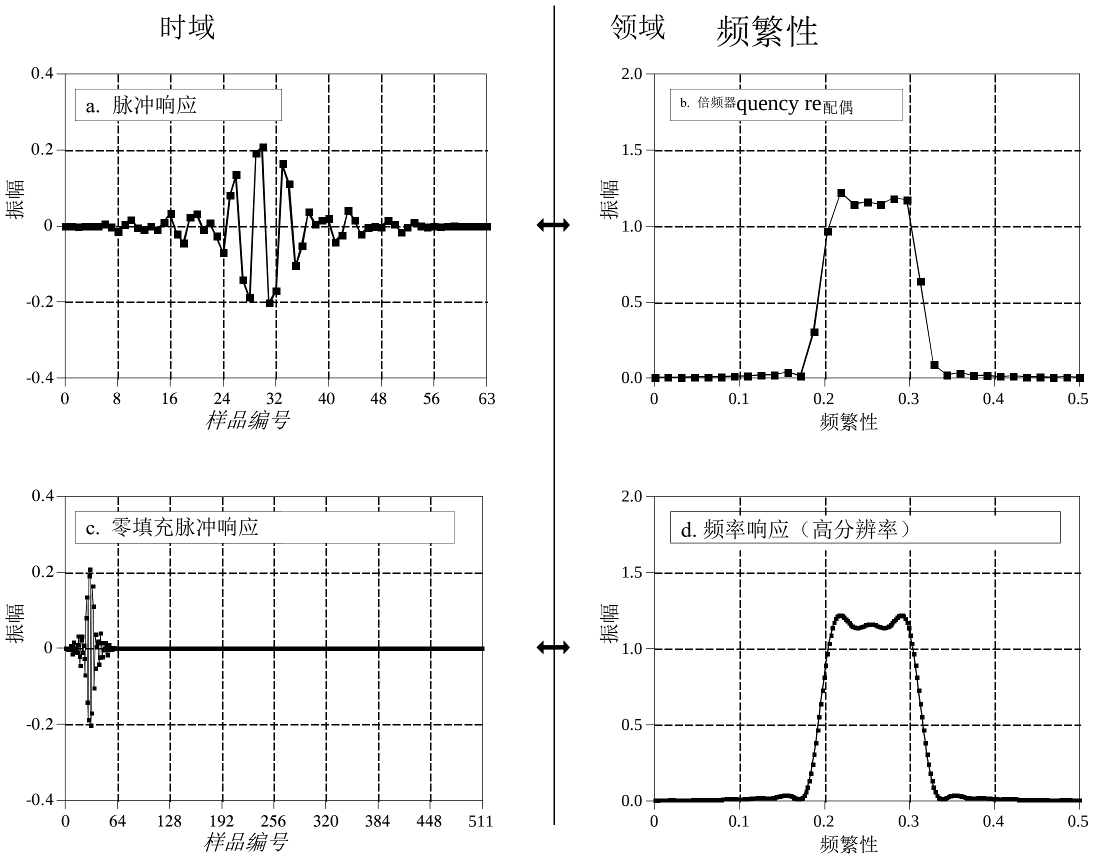


图9-7
从脉冲响应推导频率响应。通过使用离散傅里叶变换（DFT），系统的脉冲响应(a)可转换为频率响应(b)。通过对脉冲响应进行零值填充(c)，可获得更高分辨率的频率响应(d)。本例仅展示频率响应的幅值；相位的讨论将推迟至下一章。

连续曲线的形状。记住频率响应代表什么:余弦波通过系统时经历的幅度和相位变化。由于输入信号可以包含0到0.5之间的*任何*频率，系统的频率响应*必须*在这个范围内是一个连续的曲线。

这可以通过引入傅里叶变换家族的另一个成员**——离散时间傅里叶变换（DTFT）**来更好地理解。考虑一个*N*个采样点的信号经过*N*点DFT处理，生成一个*N*/2+1个采样点的频域。记得上一章提到，DFT将时域信号视为*无限长*且*周期性*的，即*N*个采样点会从负无穷到正无穷不断重复。现在考虑当我们开始用越来越*多*的零来填充时域信号，以在频域获得更精细采样时会发生什么。添加零点会使时域的周期*变长*，同时使频域采样点*更靠近*。

Now we will take this to the extreme, by adding an *infinite* number of zeros to the time domain signal. This produces a different situation in two respects. First, the time domain signal now has an infinitely long period. In other words, it has turned into an *aperiodic* signal. Second, the frequency domain has achieved an infinitesimally small spacing between samples. That is, it has become a *continuous signal*. This is the DTFT, the procedure that changes a discrete aperiodic signal in the time domain into a frequency domain that is a continuous curve. In mathematical terms, a system's frequency response is found by taking the DTFT of its impulse response. Since this cannot be done in a computer, the DFT is used to calculate a *sampling* of the true frequency response. This is the difference between what you do in a computer (the DFT) and what you do with mathematical equations (the DTFT).

Convolution via the Frequency Domain

Suppose that you despise convolution. What are you going to do if given an input signal and impulse response, and need to find the resulting output signal? Figure 9-8 provides an answer: transform the two signals into the frequency domain, multiply them, and then transform the result back into the time domain. This replaces one convolution with two DFTs, a multiplication, and an Inverse DFT. Even though the intermediate steps are very different, the output is *identical* to the standard convolution algorithm.

Does anyone hate convolution enough to go to this trouble? The answer is yes. Convolution is avoided for two reasons. First, convolution is *mathematically* difficult to deal with. For instance, suppose you are given a system's impulse response, and its output signal. How do you calculate what the input signal is? This is called **deconvolution**, and is virtually impossible to understand in the time domain. However, deconvolution can be carried out in the frequency domain as a simple *division*, the inverse operation of multiplication. The frequency domain becomes attractive whenever the complexity of the Fourier Transform is less than the complexity of the convolution. This isn't a matter of which you like better; it is a matter of which you hate less.

The second reason for avoiding convolution is *computation speed*. For example, suppose you design a digital filter with a kernel (impulse response) containing 512 samples. Using a 200 MHz personal computer with floating point numbers, each sample in the output signal requires about one millisecond to calculate, using the standard convolution algorithm. In other words, the throughput of the system is only about 1,000 samples per second. This is 40 times too slow for high-fidelity audio, and 10,000 times too slow for television quality video!

The standard convolution algorithm is slow because of the large number of multiplications and additions that must be calculated. Unfortunately, simply bringing the problem into the frequency domain via the DFT doesn't help at all. Just as many calculations are required to calculate the DFTs, as are required to directly calculate the convolution. A breakthrough was made in the problem in the early 1960s when the *Fast Fourier Transform* (FFT) was developed.

现在我们将把这一过程推向极致，通过在时域信号中添加*无限*个零点。这会产生两个方面的不同情况：首先，时域信号现在具有无限长的周期，换句话说，它变成了一个*非周期*信号；其次，频域实现了采样点之间无限小的间隔，即它变成了一个*连续信号*。这就是 DTFT ，即把时域中的离散非周期信号转换为频域中连续曲线的过程。用数学术语来说，系统的频率响应是通过对其脉冲响应取 DTFT 得到的。由于计算机无法实现这一过程，因此使用DFT来计算真实频率响应的*采样*。这就是你在计算机中操作（DFT）与通过数学方程操作（DTFT）之间的区别。

频域卷积

假设你对卷积运算感到头疼。当给定输入信号和冲激响应时，如何求得输出信号？图9-8给出了答案：将两个信号转换到频域进行相乘运算，再将结果转换回时域。这种方法用两个离散傅里叶变换（DFT）、一次乘法运算和一个逆DFT运算，替换了传统的单次卷积操作。虽然中间步骤大不相同，但最终输出结果与标准卷积算法完全*一致*。

难道真有人讨厌卷积运算到这种程度？答案是肯定的。人们回避卷积主要有两个原因。首先，卷积在*数学上*处理起来相当困难。举个例子，假设你得到一个系统的脉冲响应和输出信号，要如何反推输入信号？这叫**去卷积**，在时域几乎无法理解。不过在频域中，去卷积就像简单的*除法*——乘法的逆运算，操作起来就简单多了。当傅里叶变换的复杂度低于卷积时，频域就变得很有吸引力。这并不是说你更喜欢哪一种，而是说你讨厌哪一种的程度不同罢了。

避免卷积运算的第二个原因在于*计算速度*。举个例子，假设你设计的数字滤波器核函数（冲激响应）包含512个采样点。使用配备浮点运算的200 MHz个人电脑时，按照标准卷积算法计算每个输出信号采样点大约需要1毫秒。换言之，系统吞吐量仅有每秒约1000个采样点。这对于高保真音频来说慢了40倍，而电视级视频更是慢了1万倍！

标准卷积算法运算速度慢，因为需要计算大量的乘法和加法。遗憾的是，仅仅通过DFT将问题转换到频域并不能解决问题。计算DFT所需的运算量与直接计算卷积所需的运算量相当。直到20世纪60年代初，随着*快速傅里叶变换*（FFT）的出现，这个问题才有了突破。

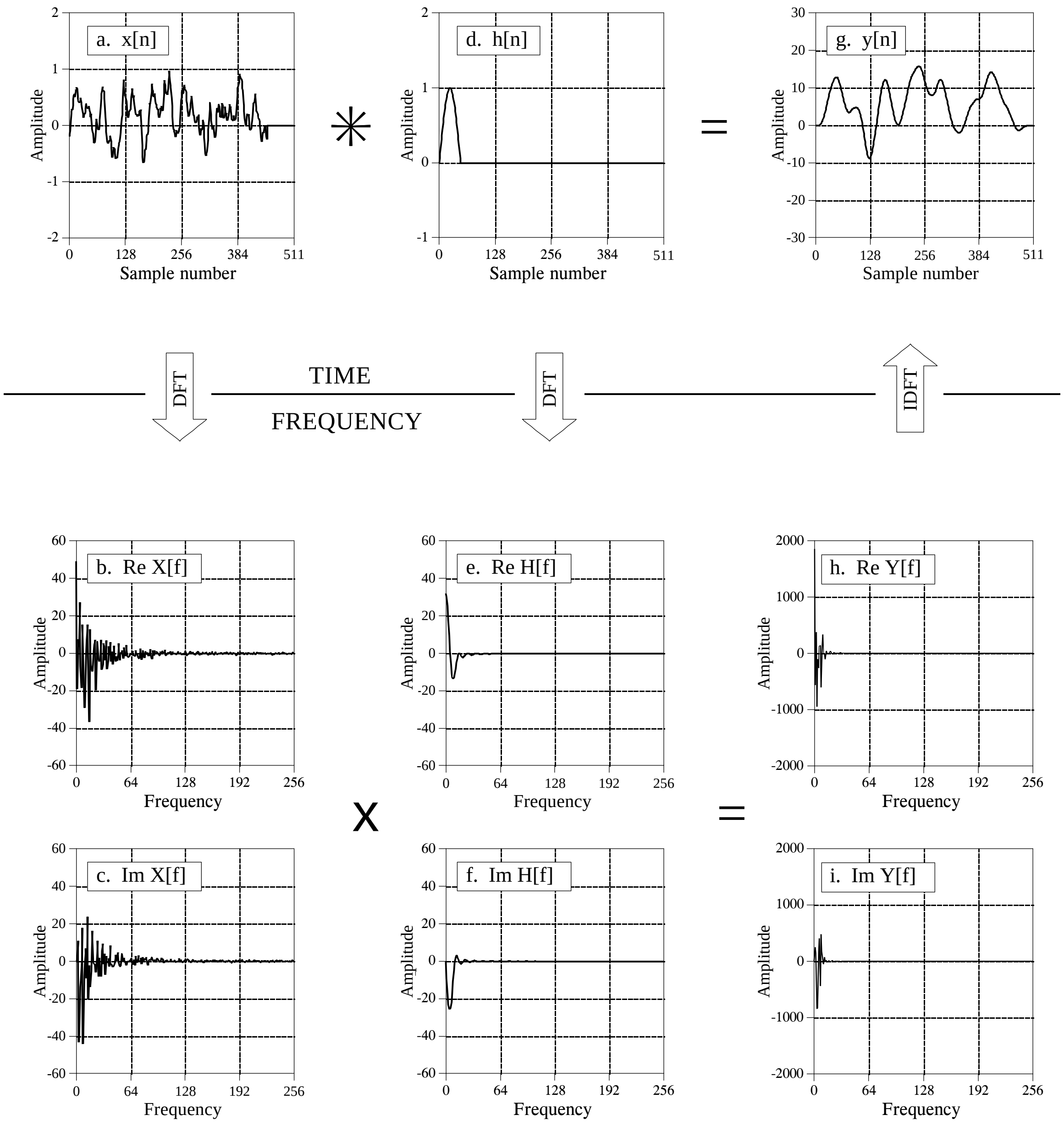


FIGURE 9-8
Frequency domain convolution. In the *time domain*, $x[n]$ is convolved with $h[n]$ resulting in $y[n]$, as is shown in Figs. (a), (d), and (g). This same procedure to be accomplished in the *frequency domain*. The DFT is used to find the frequency spectrum of the input signal, (b) & (c), and the system's frequency response, (e) & (f). Multiplying these two frequency domain signals results in the frequency spectrum of the output signal, (h) & (i). The Inverse DFT is then used to find the output signal, (g).

The FFT is a clever algorithm for rapidly calculating the DFT. Using the FFT, convolution by multiplication in the frequency domain can be hundreds of times faster than conventional convolution. Problems that take hours of calculation time are reduced to only minutes. This is why people get excited about the FFT, and processing signals in the frequency domain. The FFT will be presented in

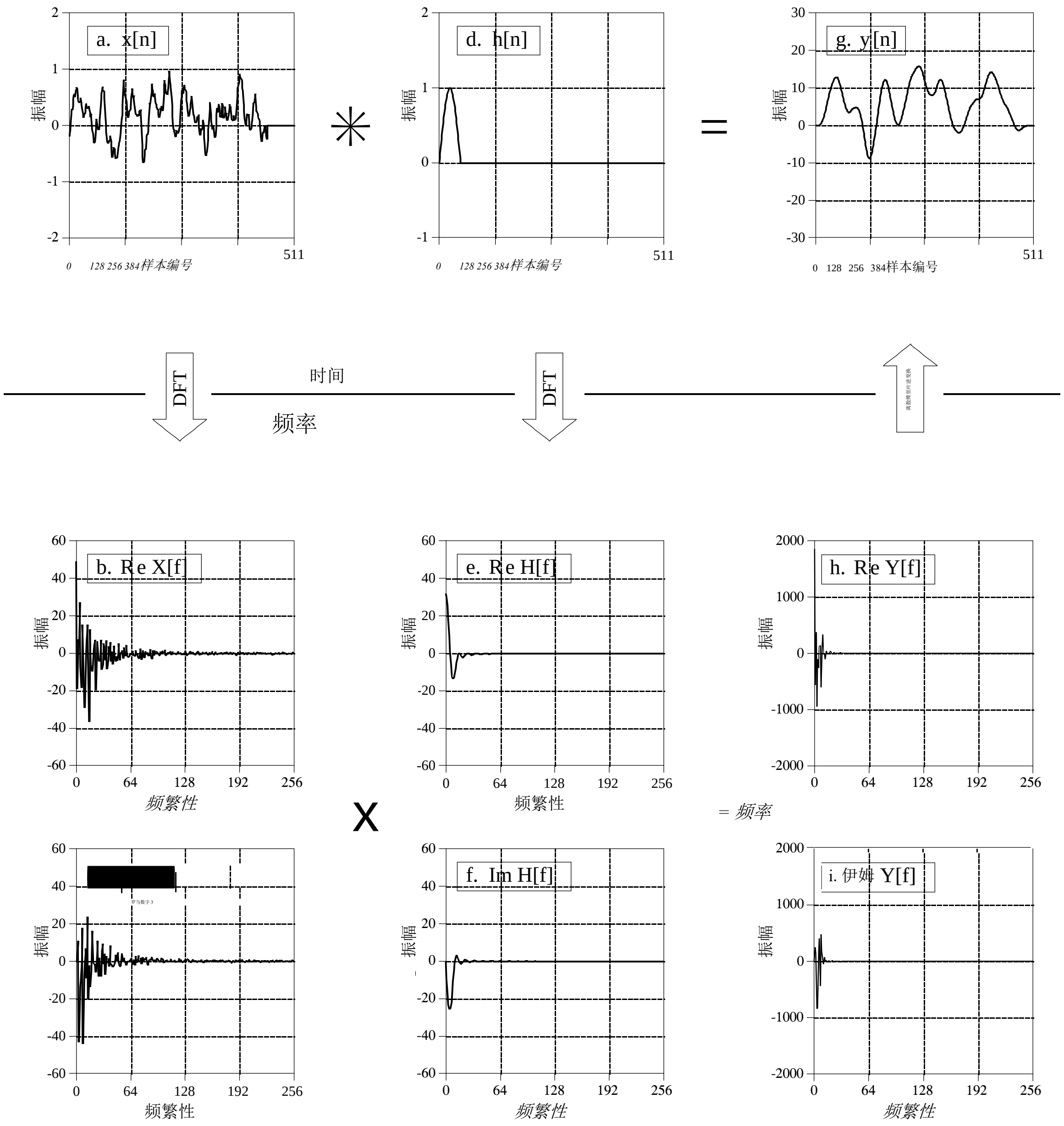


图9-8
频域卷积。在*时域*中，$x[n]$与$h[n]$进行卷积，得到$y[n]$，如图(a)、(d)和(g)所示。该相同过程需在*频域*中完成。使用DFT获取输入信号(b)和(c)的频谱以及系统的频响(e)和(f)。将这两个频域信号相乘可得到输出信号(h)和(i)的频谱。随后使用逆DFT求得输出信号(g)。

FFT 是一种用于快速计算离散傅里叶变换（DFT）的智能算法。借助该 FFT，频域乘法卷积运算速度可达传统卷积的数百倍。原本需要数小时计算的问题，现在仅需几分钟即可完成。这正是 FFT 及其频域信号处理技术备受关注的原因。 FFT 将在

Chapter 12, and the method of FFT convolution in Chapter 18. For now, focus on how signals are convolved by frequency domain multiplication.

To start, we need to define how to multiply one frequency domain signal by another, i.e., what it means to write: $X[f] \times H[f] = Y[f]$. In polar form, the magnitudes are multiplied: $Mag\,Y[f] = Mag\,X[f] \times Mag\,H[f]$, and the phases are added: $Phase\,Y[f] = Phase\,X[f] + Phase\,H[f]$. To understand this, imagine a cosine wave entering a system with some amplitude and phase. Likewise, the output signal is also a cosine wave with some amplitude and phase. The polar form of the frequency response directly describes how the two amplitudes are related and how the two phases are related.

When frequency domain multiplication is carried out in *rectangular form* there are cross terms between the real and imaginary parts. For example, a sine wave entering the system can produce both cosine and sine waves in the output. To multiply frequency domain signals in rectangular notation:

EQUATION 9-1
Multiplication of frequency domain signals in rectangular form: $Y[f] = X[f] \times H[f]$.

$$Re\,Y[f] = Re\,X[f]\,Re\,H[f] - Im\,X[f]\,Im\,H[f]$$

$$Im\,Y[f] = Im\,X[f]\,Re\,H[f] + Re\,X[f]\,Im\,H[f]$$

Focus on understanding multiplication using *polar notation*, and the idea of cosine waves passing through the system. Then simply accept that these more elaborate equations result when the same operations are carried out in rectangular form. For instance, let's look at the *division* of one frequency domain signal by another. In polar form, the division of frequency domain signals is achieved by the inverse operations we used for multiplication. To calculate: $H[f] = Y[f] / X[f]$, divide the magnitudes and subtract the phases, i.e., $Mag\,H[f] = Mag\,Y[f] / Mag\,X[f]$, $Phase\,H[f] = Phase\,Y[f] - Phase\,X[f]$. In rectangular form this becomes:

EQUATION 9-2
Division of frequency domain signals in rectangular form, where: $H[f] = Y[f] / X[f]$.

$$Re\,H[f] = \frac{Re\,Y[f]\,Re\,X[f] + Im\,Y[f]\,Im\,X[f]}{Re\,X[f]^2 + Im\,X[f]^2}$$

$$Im\,H[f] = \frac{Im\,Y[f]\,Re\,X[f] - Re\,Y[f]\,Im\,X[f]}{Re\,X[f]^2 + Im\,X[f]^2}$$

Now back to frequency domain convolution. You may have noticed that we cheated slightly in Fig. 9-8. Remember, the convolution of an N point signal with an M point impulse response results in an $N+M-1$ point output signal. We cheated by making the last part of the input signal all *zeros* to allow this expansion to occur. Specifically, (a) contains 453 nonzero samples, and (b) contains 60 nonzero samples. This means the convolution of the two, shown in (c), can fit comfortably in the 512 points provided.

第12章和第18章的 FFT 卷积方法。目前重点讲解信号如何通过频域乘法进行卷积。

首先，我们需要定义如何将一个频域信号与另一个相乘，即如何表示：*X*[*f*]× *H*[*f*] =*Y*[*f*]。在极坐标形式中，幅值相乘：*MagY*[*f*] =*MagX*[*f*]× *MagH*[*f*]，相位相加：*PhaseY*[*f*] =*PhaseX*[*f*] +*PhaseH*[*f*]。为理解这一点，想象一个具有特定幅值和相位的余弦波进入系统。同样，输出信号也是一具有特定幅值和相位的余弦波。频响的极坐标形式直接描述了两个幅值之间的关系以及两个相位之间的关系。

当进行*矩形表示法*的频域乘法时，实部和虚部之间会产生交叉项。例如，输入系统的正弦波会在输出中同时产生余弦波和正弦波。要以矩形表示法进行频域信号乘法：

方程9-1
矩形频域信号的乘积：*Y*[*f*] =*X*[*f*]× *H*[*f*]。

$$ReY[f] = ReX[f]\,ReH[f] - ImX[f]\,ImH[f]$$

$$ImY[f] = ImX[f]\,ReH[f] + ReX[f]\,ImH[f]$$

专注于通过*极坐标表示法*理解乘法，以及余弦波通过系统的概念。然后简单接受这些更复杂的方程是通过以矩形形式执行相同运算而产生的。例如，让我们看看一个频域信号被另一个频域信号*除法*的情况。在极坐标形式中，频域信号的除法通过我们用于乘法的逆运算实现。计算公式为：*H*[*f*] =*Y*[*f*] /*X*[*f*]，即对幅值进行除法并相位相减，即*MagH*[*f*] =*MagY*[*f*] / *MagX*[*f*]，*Phase H*[*f*] =*Phase Y*[*f*] &*Phase X*[*f*]。在矩形形式中这变为：

方程9-2
将频域信号划分为矩形形式，其中：*H*[*f*] =*Y*[*f*] /*X*[*f*]。

$$ReH[f] = \frac{ReY[f]\,ReX[f] + ImY[f]\,ImX[f]}{ReX[f]^2 + ImX[f]^2}$$

$$ImH[f] = \frac{ImY[f]\,ReX[f] - ReY[f]\,ImX[f]}{ReX[f]^2 + ImX[f]^2}$$

现在回到频域卷积。你可能已经注意到我们在图9-8中稍微作弊了。记住，一个*N*点信号与一个*M*点冲激响应的卷积会产生一个*N*+*M*&1点的输出信号。我们通过将输入信号的最后一部分全部设为*零*来实现这种扩展，这算是作弊了。具体来说，(a)包含453个非零样本，(b)包含60个非零样本。这意味着这两个信号的卷积结果（如(c)所示）可以轻松容纳在提供的512个点中。

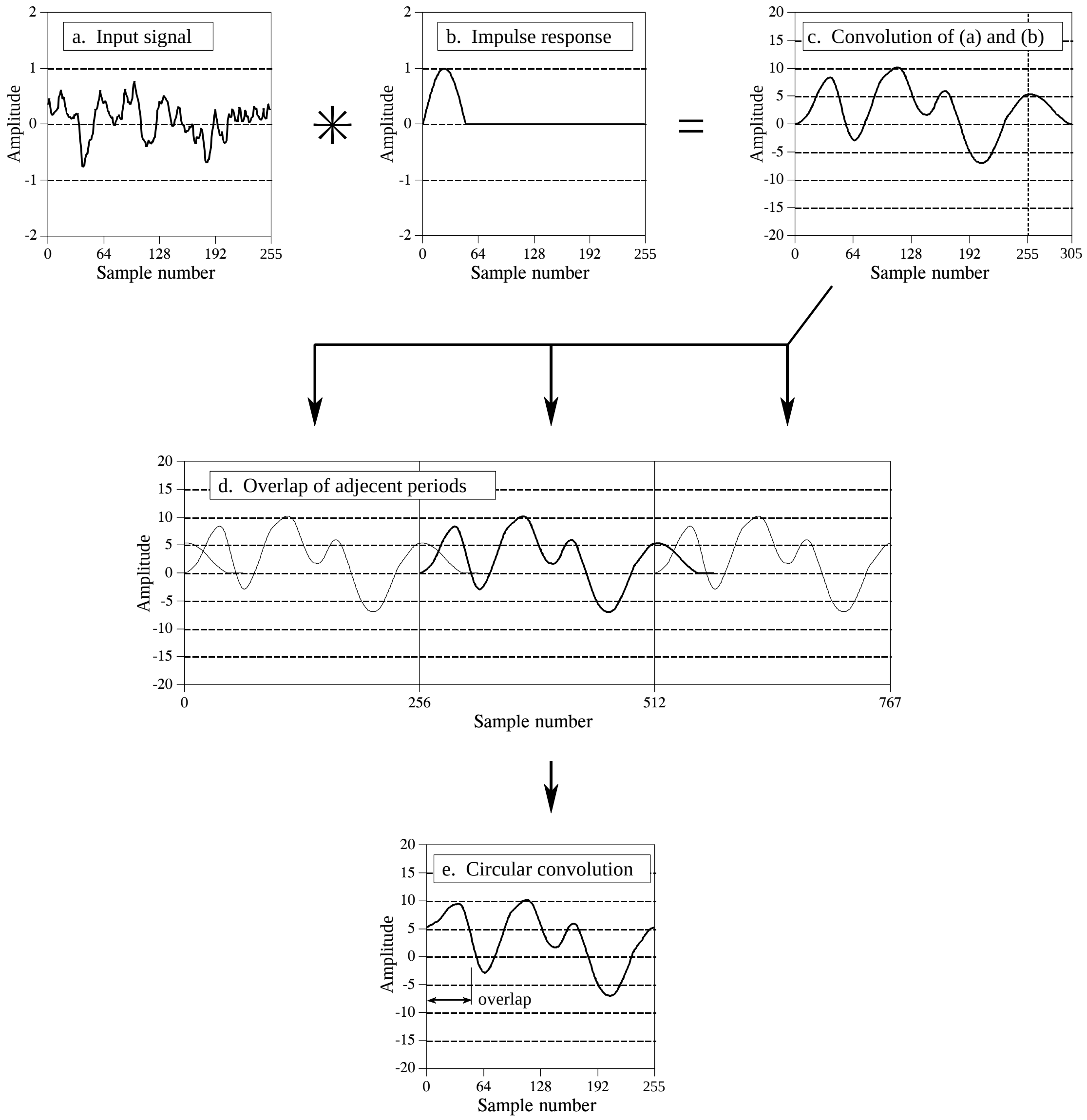


FIGURE 9-9
Circular convolution. A 256 sample signal, (a), convolved with a 51 sample impulse response, (b), results in a 306 sample signal, (c). If this convolution is performed in the frequency domain using 256 point DFTs, the 306 points in the correct convolution cannot fit into the 256 samples provided. As shown in (d), samples 256 through 305 of the output signal are pushed into the next period to the right, where they *add* to the beginning of the next period's signal. Figure (e) is a single period of the resulting signal.

Now consider the more general case in Fig. 9-9. The input signal, (a), is 256 points long, while the impulse response, (b), contains 51 nonzero points. This makes the convolution of the two signals 306 samples long, as shown in (c). The problem is, if we use frequency domain multiplication to perform the convolution, there are only 256 samples *allowed* in the output signal. In other words, 256 point DFTs are used to move (a) and (b) into the frequency

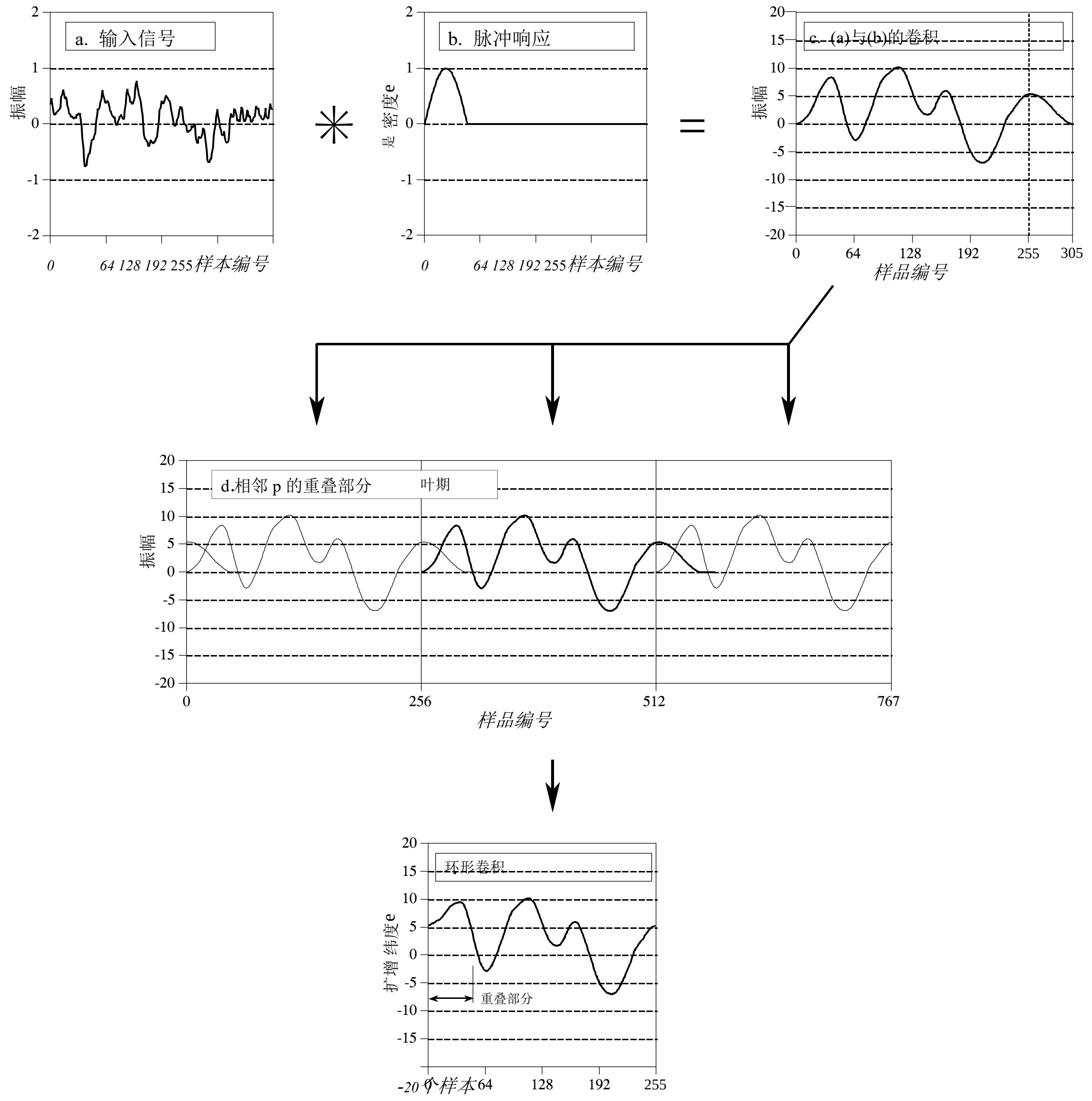


图9-9
循环卷积。将256个采样点的信号(a)与51个采样点的冲激响应(b)进行卷积后，得到306个采样点的信号(c)。若在频域中使用256点离散傅里叶变换（DFT）执行此卷积操作，
正确的卷积306点无法适应256个样本。如(d)所示，输出信号的256到305个样本被推入下一个周期的右侧，它们*加到*下一个周期信号的开头。图(e)是结果信号的一个周期。

现在考虑图9-9中更普遍的情况。输入信号(a)包含256个采样点，而冲激响应(b)则有51个非零点。如(c)所示，这两个信号的卷积结果将产生306个采样点。问题在于，若采用频域乘法进行卷积运算，输出信号中只能保留256个采样点*（允许）*。换句话说，需要使用256点离散傅里叶变换（DFT）将(a)和(b)转换到频域。

domain. After the multiplication, a 256 point Inverse DFT is used to find the output signal. How do you squeeze 306 values of the correct signal into the 256 points provided by the frequency domain algorithm? The answer is, you can't! The 256 points end up being a distorted version of the correct signal. This process is called **circular convolution**. It is important because you want to *avoid* it.

To understand circular convolution, remember that an N point DFT views the time domain as being an infinitely long periodic signal, with N samples per period. Figure (d) shows three periods of how the DFT views the output signal in this example. Since $N = 256$, each period consists of 256 points: 0-255, 256-511, and 512-767. Frequency domain convolution tries to place the 306 point *correct output signal*, shown in (c), into each of these 256 point periods. This results in 49 of the samples being pushed into the neighboring period to the right, where they overlap with the samples that are legitimately there. These overlapping sections add, resulting in each of the periods appearing as shown in (e), the *circular convolution*.

Once the nature of circular convolution is understood, it is quite easy to avoid. Simply pad each of the signals being convolved with enough zerosto allow the output signal room to handle the $N+M-1$ points in the correct convolution. For example, the signals in (a) and (b) could be padded with zeros to make them 512 points long, allowing the use of 512 point DFTs. After the frequency domain convolution, the output signal would consist of 306 nonzero samples, plus 206 samples with a value of zero. Chapter 18 explains this procedure in detail.

Why is it called *circular* convolution? Look back at Fig. 9-9d and examine the center period, samples 256 to 511. Since all of the periods are the same, the portion of the signal that flows out of this period to the *right*, is the same that flows into this period from the *left*. If you only consider a single period, such as in (e), it *appears* that the right side of the signal is somehow *connected* to the left side. Imagine a snake biting its own tail; sample 255 is located next to sample 0, just as sample 100 is located next to sample 101. When a portion of the signal exits to the right, it magically reappears on the left. In other words, the N point time domain behaves as if it were *circular*.

In the last chapter we posed the question: does it really matter if the DFT's time domain is viewed as being N points, rather than an infinitely long periodic signal of period N? Circular convolution is an example where it *does* matter. If the time domain signal is understood to be *periodic*, the distortion encountered in circular convolution can be simply explained as the signal expanding from one period to the next. In comparison, a rather bizarre conclusion is reached if only N points of the time domain are considered. That is, frequency domain convolution acts as if the time domain is somehow wrapping into a circular ring with sample 0 being positioned next to sample N-1.

在完成乘法运算后，系统会使用256点逆离散傅里叶变换（Inverse DFT）来获取输出信号。但问题来了：如何将306个正确信号值压缩到频域算法提供的256个采样点中？答案是——根本不可能！最终得到的256点数据会扭曲原始信号。这个过程被称为**循环卷积**，关键在于要*避免*这种失真现象。

要理解循环卷积，需要记住N点DFT将时域视为无限长的周期信号，每个周期包含N个采样点。图(d)展示了该示例中DFT如何呈现输出信号的三个周期。由于N=256，每个周期包含256个点：0-255、256-511和512-767。频域卷积试图将306点的*正确输出信号*（如图(c)所示）分别放置到这256个点的周期中。这导致49个采样点被推入右侧相邻周期，与本应存在的采样点发生重叠。这些重叠部分相加后，每个周期呈现为图(e)所示的*循环卷积*。

一旦理解了循环卷积的本质，就很容易避免其影响。只需用足够数量的零值对每个待卷积信号进行填充，就能让输出信号有足够的空间来正确处理N+M&1个点。例如，(a)和(b)中的信号可以通过填充零值使其长度达到512点，从而可以使用512点的DFT。经过频域卷积后，输出信号将包含306个非零样本，外加206个零值样本。第18章将详细解释这一过程。

为什么它被称为*循环*卷积？回顾图9-9d并观察中心周期（样本256至511）。由于所有周期都相同，从这个周期*右侧*流出的信号部分，与从*左侧*流入这个周期的部分完全一致。如果只考虑单个周期（如(e)所示），信号的右侧*看起来*像是以某种方式*连接*到了左侧。想象一条蛇咬住自己的尾巴：样本255紧挨着样本0，就像样本100紧挨着样本101一样。当信号的一部分向右流出时，它就像变魔术般重新出现在左侧。换句话说，N点时域的表现就像*循环*的。

在上一章中，我们提出了一个问题：将DFT的时间域视为N个采样点，而非周期为N的无限长周期信号，这是否真的重要？循环卷积就是一个*确实*重要的例子。如果将时间域信号理解为*周期性*的，那么循环卷积中出现的失真现象可以简单解释为信号从一个周期扩展到下一个周期。相比之下，如果仅考虑时间域的N个采样点，就会得出一个相当奇怪的结论。也就是说，频域卷积的作用就像时间域以某种方式卷成一个环形结构，使得采样点0紧邻采样点N-1。