

Spectral Analysis

It is easy enough to measure the frequency of a clean sinusoid, assuming that we have seen enough of the signal for its frequency to be determinable. For more complex signals the whole concept of frequency becomes more complex. We previously saw two distinct meanings, the spectrum and the instantaneous frequency. The concept of spectrum extends the single frequency of the sinusoid to a simultaneous combination of many frequencies for a general signal; as we saw in Section 4.5 the power spectral density (PSD) defines how much each frequency contributes to the overall signal. Instantaneous frequency takes the alternative approach of assuming only one frequency at any one time, but allowing this frequency to vary rapidly. The tools that enable us to numerically determine the instantaneous frequency are the Hilbert transform and the differentiation filter.

There is yet a third definition about which we have not spoken until now. Model based spectral estimation methods assume a particular mathematical expression for the signal and estimate the parameters of this expression. This technique extends the idea of estimating the frequency of a signal assumed to be a perfect sinusoid. The difference here is that the assumed functional form is more complex. One popular model is to assume the signal to be one or more sinusoids in additive noise, while another takes it to be the output of a filter. This approach is truly novel, and the uncertainty theorem does not directly apply to its frequency measurements.

This chapter deals with the practical problem of numerically estimating the frequency domain description of a signal. We begin with simple methods and cover the popular FFT-based methods. We describe various window functions and how these affect the spectral estimation. We then present Pisarenko's Harmonic Decomposition and several related super-resolution methods. We comment on how it is possible to break the uncertainty barrier. We then briefly discuss ARMA (maximum entropy) models and how they are fundamentally different from periodogram methods. We finish off with a brief introduction to wavelets.

13.1 Zero Crossings

Sophisticated methods of spectral estimation are not always necessary. Perhaps the signal to noise ratio is high, or we don't need very high accuracy. Perhaps we know that the signal consists of a single sinusoid, or are only interested in the most important frequency component. Even more frequently we don't have the real-time to spare for computationally intensive algorithms. In such cases we can sometimes get away with very simple methods.

The quintessence of simplicity is the zero crossing detector. The frequency of a clean analog sinusoid can be measured by looking for times when it crosses the t axis (zero signal value). The interval between two successive zero crossings represents a half cycle, and hence the frequency is half the reciprocal of this interval. Alternatively, we can look for zero crossings of the same type (i.e., both 'rising' or both 'falling'). The reciprocal of the time interval between two rising (or falling) zero crossings is precisely the frequency. Zero crossings can be employed to determine the basic frequency even if the signal's amplitude varies.

In practice there are two distinct problems with the simple implementation of zero crossing detection. First, observing the signal at discrete times reduces the precision of the observed zero crossing times; second, any amount of noise makes it hard to accurately pin down the exact moment of zero crossing. Let's deal with the precision problem first. Using only the sign of the signal (we assume any DC has been removed), the best we can do is to say the zero is somewhere between time n and time $n + 1$. However, by exploiting the signal values we can obtain a more precise estimate. The simplest approach assumes that the signal traces a straight line between the two values straddling the zero. Although in general sinusoids do not look very much like straight lines, the approximation is not unreasonable near the zero crossings for a sufficiently high sampling rate (see Figure 13.1). It is easy to derive an expression for the fractional correction under this assumption, and expressions based on polynomial interpolation can be derived as well.

Returning to the noise problem, were the signal more 'observable' the Robins-Munro algorithm would be helpful. For the more usual case we need to rely on stationarity and ergodicity and remove the noise through a suitable averaging process. The simplest approach is to average interpolated time intervals between zero crossings.

The time duration between zero crossings predicts the basic frequency, only assuming this basic frequency is constant. If it does vary, but sufficiently slowly, it makes sense to monitor the so-called 'zero crossing derivative', the sequence of time differences between successive zero crossing intervals.

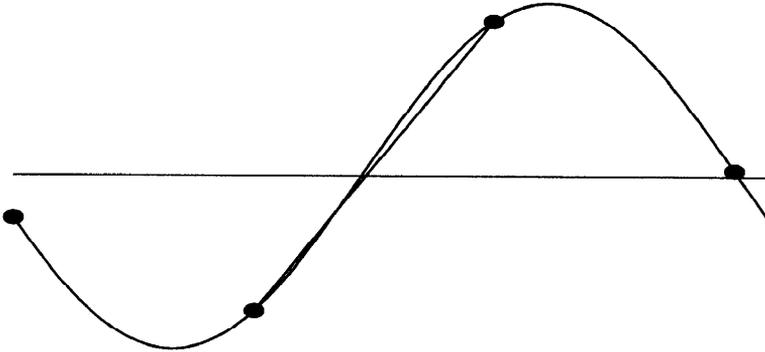


Figure 13.1: Zero crossing detector for clean sinusoid with no DC offset. The sampling rate is about double Nyquist (four samples per period). Note that the linear approximation is reasonable but not perfect.

Given the signal we first compute the sequence of interpolated zero crossing instants $t_0, t_1, t_2, t_3 \dots$ and then compute the zero crossing intervals by subtraction of successive times (the finite difference sequence) $\Delta_1 = t_1 - t_0$, $\Delta_2 = t_2 - t_1$, $\Delta_3 = t_3 - t_2$ and so on. Next we find the zero crossing derivative as the second finite difference $\Delta_2^{[2]} = \Delta_2 - \Delta_1$, $\Delta_3^{[2]} = \Delta_3 - \Delta_2$, $\Delta_4^{[2]} = \Delta_4 - \Delta_3$ etc. If the underlying frequency is truly constant the Δ sequence averages to the true frequency reciprocal and the $\Delta^{[2]}$ sequence is close to zero. Frequency variations show up in the derivative sequence.

This is about as far as it is worth going in this direction. If the zero crossing derivatives are not sufficient then we probably have to do something completely different. Actually zero crossings and their derivatives are frequently used to derive features for pattern recognition purposes but almost never used as frequency estimators. As feature extractors they are relatively robust, fast to calculate, and contain a lot of information about the signal. As frequency estimators they are not reliable in noise, not particularly computationally efficient, and cannot compete with the optimal methods we will present later on in this chapter.

EXERCISES

- 13.1.1 What is the condition for two signal values s_n and s_{n+1} to straddle a rising zero crossing? A falling zero crossing? Any zero crossing?
- 13.1.2 Assume that we have located a rising zero crossing between times n and $n+1$. Derive an expression for δt , the fractional correction to be added to $t = n$, assuming that the signal traces a straight line between s_n and s_{n+1} . Extend to an arbitrary (rising or falling) zero crossing.

13.2 Bank of Filters

The zero crossing approach is based on the premise of well-defined instantaneous frequency, what we once called the ‘other meaning’ of frequency. Shifting tactics we return to the idea of a well-defined spectrum and seek an algorithm that measures the distribution of energy as a function of frequency. The simplest approach here is the ‘bank of filters’, inspired by the analog spectrum analyzer of that name. Think of the frequency band of interest, let’s say from 0 to F Hz, as being composed of N equal-size nonoverlapping frequency subbands. Employing N band-pass filters we extract the signal components in these subbands, which we denote \tilde{s}^0 through \tilde{s}^{N-1} . We have thus reduced a single signal of bandwidth F into N signals each of bandwidth $\frac{F}{N}$; see Figure 13.2.

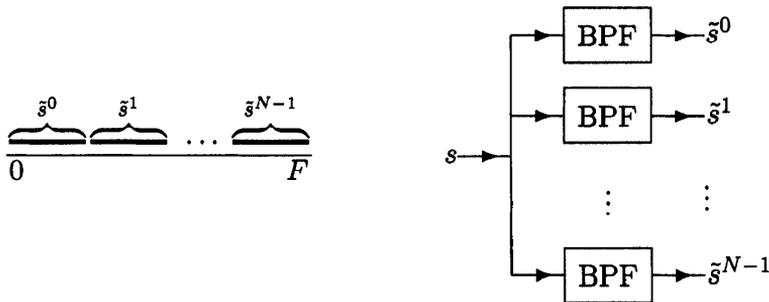


Figure 13.2: A bank of filters dividing the frequency band from 0 to F into N subbands, each containing a band-pass signal. On the left the spectrum is depicted, while the right shows the bank of filters that accomplishes this division.

At this point we could simply add the filter outputs \tilde{s}^k together and reconstruct the original signal s ; thus the set of signals \tilde{s}^k contains all the information contained in the original signal. Such an equivalent way of encoding the information in a signal is called a *representation*. The original signal s is the time domain representation, the spectrum is the frequency domain representation, and this new set of signals is the *subband representation*. Subband representations are useful in many contexts, but for now we will only compute the energies of all the subband signals \tilde{s}^k , obtaining an estimate of the power spectrum. The precision of this estimate is improved when using a larger number of subbands, but the computational burden goes up as well.

The bank of filters approach to the PSD does not differentiate between a clean sinusoid and narrow-band noise, as long as both are contained in the

same subband. Even if the signal is a clean sinusoid this approach cannot provide an estimate of its frequency more precise than the bandwidth of the subband.

We have taken the subbands to be equal in size (i.e., we have divided the total spectral domain into N equal parts), but this need not be the case. For instance, speech spectra are often divided equally on a logarithmic scale, such that lower frequencies are determined more precisely than higher ones. This is no more difficult to do, since it only requires proper design of the filters. In fact it is computationally lighter if we build up the representation recursively. First we divide the entire domain in two using one low-pass and one high-pass filter. The energy at the output of the high-pass filter is measured, while the signal at the output of the low-pass filter is decimated by two and then input to a low-pass and a high-pass filter. This process is repeated until the desired precision of the lowest-frequency bin is attained.

Returning to the case of equal size subbands, we note that although all the signals \tilde{s}^0 through \tilde{s}^{N-1} have equal bandwidth, there is nonetheless a striking lack of equality. The lowest subband s^0 is a low-pass signal, existing in the range from 0 to $\frac{F}{N}$. It can be easily sampled and stored using the low-pass sampling theorem. All the other \tilde{s}^k are band-pass signals and hence require special treatment. For example, were we required to store the signal in the subband representation rather than merely compute its power spectrum, it would be worthwhile to downmix all the band-pass signals to the frequency range of 0 to $\frac{F}{N}$. Doing this we obtain a new set of signals we now call simply s^k ; s^0 is exactly \tilde{s}^0 , while all the other s^k are obtained from the respective \tilde{s}^k by mixing down by $\frac{kF}{N}$. This new set of signals also contains all the information of the original signal, and is thus a representation as well. We can call it the *low-pass subband representation* to be contrasted with the previous *band-pass subband representation*. The original signal s is reconstructed by mixing up each subband to its proper position and then summing as before. The power spectrum is computed exactly as before since the operation of mixing does not affect the energy of the subband signals.

The low-pass subband representation of a signal can be found without designing and running N different band-pass filters. Rather than filtering with band-pass filters and then downmixing, one can downmix first and then low-pass filter the resulting signals (see Figure 13.3). In sequential computation this reduces to a single mixer-filter routine called N times on the same input with different downmix frequencies. This is the digital counterpart of the *swept-frequency spectral analyzer* that continuously sweeps in sawtooth fashion the local oscillator of a mixer, plotting the energy at the output of a low-pass filter as a function of this frequency.

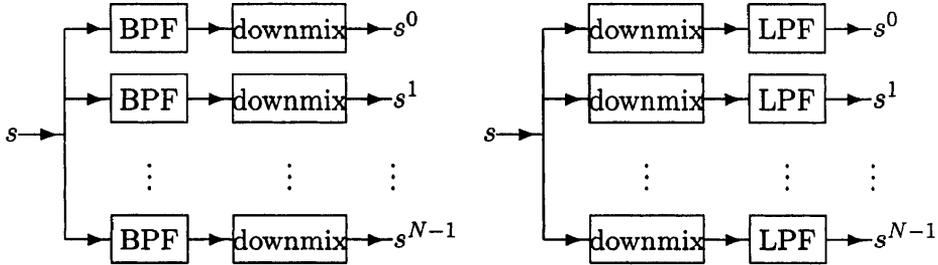


Figure 13.3: Two equivalent implementations of a bank of filters dividing the frequency range into N low-pass signals. In (A) the band-pass signals are band-pass filtered and then downmixed using a real mixer, while in (B) the input signal is downmixed by a complex mixer and then low-pass filtered.

Although the two methods of computing the band-pass representation provide exactly the same signals s^k , there is an implementational difference between them. While the former method employed band-pass filters with real-valued multiplications and a real mixer (multiplication by a sine function), the latter requires a complex mixer (multiplication by a complex exponential) and then complex multiplications. The complex mixer is required in order to shift the entire frequency range without spectral aliasing (see Section 8.5). Once such a complex mixer is employed the signal becomes complex-valued, and thus even if the filter coefficients are real two real multiplications are needed.

Since all our computation is complex, we can just as easily input complex-valued signals, as long as we cover the frequency range up to the sampling frequency, rather than half f_s . For N subbands, the analog downmix frequencies for such a complex input are $0, \frac{f_s}{N}, \frac{2f_s}{N}, \dots, \frac{(N-1)f_s}{N}$, and therefore the digital complex downmixed signals are

$$s_n e^{-i\frac{2\pi}{N}kn} = s_n W_N^{nk} \quad \text{for } k = 0 \dots N-1$$

where W_N is the N^{th} root of unity (see equation (4.30)). These products need to be low-pass filtered in order to build the s^k . If we choose to implement the low-pass filter as a causal FIR filter, what should its length be? From an information theoretic point of view it is most satisfying to choose length N , since then N input samples are used to determine N subband representation values. Thus we find that the k^{th} low-pass signal is given by

$$s_n^k = \sum_{n=0}^{N-1} h_n s_n e^{-i\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} h_n s_n W_N^{nk} \quad (13.1)$$

which looks somewhat familiar. In fact we can decide to use as our low-pass filter a simple moving average with all coefficients equal to one (see Section 6.6). Recall that this *is* a low-pass filter; perhaps not a very good one (its frequency response is a sinc), but a low-pass filter all the same. Now we can write

$$s_n^k = \sum_{n=0}^{N-1} s_n e^{-i\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} s_n W_N^{nk} = S_k \quad (13.2)$$

which is, of course, precisely the DFT. However, instead of thinking of the DFT as providing the frequency domain representation of a signal, here we consider it as calculating the low-pass subband representation. In this fashion the DFT becomes a tool for efficiently simultaneously downmixing and filtering the signal. The mixers are easily seen in the definition of the DFT; the filtering is implicit in the sum over N input values.

We have to acclimate ourselves to this new interpretation of the DFT. Rather than understanding S_k to be a frequency component, we interpret s_n^k as a time domain sample of a subband signal. For instance, an input signal consisting of a few sinusoids corresponds to a spectrum with a few discrete lines. All subband signals corresponding to empty DFT bins are correctly zero, while sinusoids at bin centers lead to constant (DC) subband signals. So the interpretation is consistent for this case, and we may readily convince ourselves that it is consistent in general.

We have seen that in our bank of filters approach to computing the power spectrum we actually indirectly compute the DFT. In the next section we take up using the DFT to directly estimate the power spectrum.

EXERCISES

- 13.2.1 The low-pass subband representation can be useful in other contexts as well. Can you think of any? (Hint: FDM.)
- 13.2.2 Why does the bank of filters approach become unattractive when a large number of filters must be used?
- 13.2.3 Compare the following three similar spectral analysis systems: (1) a bank of $N + 1$ very steep skirted analog band-pass filters spaced at Δf from 0 to $F = N\Delta f$; (2) a similar bank of $N + 1$ digital filters; (3) a single DFT with bin size Δf . We inject a single sinusoid of arbitrary frequency into each of the three systems and observe the output signal (note that we do not observe only the energy). Do the three give identical results? If not, why not?
- 13.2.4 Compare the computational complexity of the recursive method of finding the logarithmic spectrum with the straightforward method.

- 13.2.5 Prove that the energy of a band-pass signal is unchanged when it is mixed to a new frequency range.
- 13.2.6 We saw that the DFT downmixes the subbands before filtering, and we know that a mixer is not a filter. In what sense is the DFT equivalent to a bank of filters? How can we empirically measure the frequency response of these filters?
- 13.2.7 Build a bank of filters spectrum analyzer using available filter design or FFT software. Inject static combinations of a small number of sinusoids. Can you always determine the correct number of signals? Plot the outputs of the filters (before taking the energy). Do you get what you expect? Experiment with different numbers of bins. Inject a sinusoid of slowly varying frequency. Can you reconstruct the frequency response of the filters? What happens when the frequency is close to the border between two subbands?

13.3 The Periodogram

In 1898, Sir Arthur Schuster published his investigations regarding the existence of a particular periodic meteorological phenomenon. It is of little interest today whether the phenomenon in question was found to be of consequence; what *is* significant is the technique used to make that decision. Schuster introduced the use of an empirical STFT in order to discover hidden periodicities, and hence called this tool the *periodogram*. Simply put, given N equally spaced data points $s_0 \dots s_{N-1}$, Schuster recommended computing (using our notation)

$$P(\omega) = \frac{1}{N} \left| \sum_{n=0}^{N-1} s_n e^{-i\omega n} \right|^2 \quad (13.3)$$

for a range of frequencies ω and looking for peaks—peaks that represent hidden periodicities. We recognize this as the DFT power spectrum evaluated for the available data.

Many of today's DSP practitioners consider the FFT-based periodogram to be the most natural power spectral estimator. Commercially available hardware and software digital spectrum analyzers are almost exclusively based on the FFT. Indeed DFT-based spectral estimation is a powerful and well-developed technique that should probably be the first you explore when a new problem presents itself; but as we shall see in later sections it is certainly not the only, and often not even the best technique.

What is the precise meaning of the periodogram's $P(\omega)$? We would like for it to be an estimate of the true power spectral density, the PSD that would be calculated were an infinite amount of data (and computer time) to be available. Of course we realize that the fact that our data only covers a finite time duration implies that the measurement cannot refer to an infinitesimal frequency resolution. So the periodogram must be some sort of average PSD, where the power is averaged over the bandwidth allowed by the uncertainty theorem.

What is the weighting of this average? The signal we are analyzing is the true signal, which exists from the beginning of time until its end, multiplied by a rectangular window that is unity over the observed time interval. Accordingly, the FT in the periodogram is the convolution of the true FT with the FT of this window. The FT of a rectangular window is given by equation (4.22), and is sinc shaped. This is a major disappointment! Not only do frequencies far from the minimum uncertainty bandwidth 'leak' into the periodogram PSD estimate, the strength of these distant components does not even monotonically decrease.

Is the situation really as bad as it seems? To find out let's take 64 samples of a sinusoid with digital frequency $15/64$, compute the FFT, take the absolute square for the positive frequencies, and convert to dB. The analog signal, the samples, and the PSD are shown in Figure 13.4.A. All looks fine; there is only a single spectral line and no leakage is observed. However, if we look carefully at the sinc function weighting we will see that it has a zero at the center of all bins other than the one upon which it is centered. Hence there is never leakage from a sinusoid that is exactly centered in some neighboring bin (i.e., when its frequency is an integer divided by the number of samples). So let's observe what happens when the digital frequency is slightly higher (e.g., $f_d = 15.04/64$) as depicted in Figure 13.4.B. Although this frequency deviation is barely noticeable in the time domain, there is quite significant leakage into neighboring bins. Finally, the worst-case is when the frequency is exactly on the border between two bins, for example, $f_d = 15.5/64$ as in Figure 13.4.C. Here the leakage is already intolerable.

Why is the periodogram so bad? The uncertainty theorem tells us that short time implies limited frequency resolution but DSP experience tells us that small buffers imply bothersome edge effects. A moment's reflection is enough to convince you that only when the sinusoid is precisely centered in the bin are there an integer number of cycles in the DFT buffer. Now recall that the DFT forces the signal to be periodic outside the duration of the buffer that it sees; so when there are a noninteger number of cycles the signal

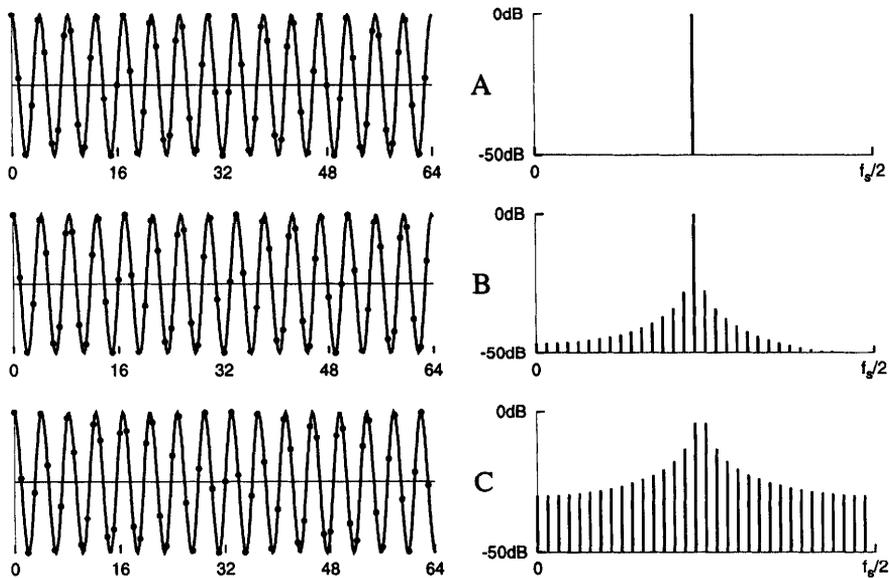


Figure 13.4: Leakage in the spectrum of a single sinusoid. In (A) precisely 15 cycles of the sinusoid fit into the buffer of length 64 samples and thus its periodogram contains a single line. In (B) 15.04 cycles fit into the buffer and thus there is a small discontinuity at the edge. The periodogram displays leakage into neighboring bins. In (C) $14\frac{1}{2}$ cycles fit and thus the discontinuity and leakage are maximal. Note also that the two equal bins are almost 4 dB lower than the single maximal bin in the first case, since the Parseval energy is distributed among many bins.

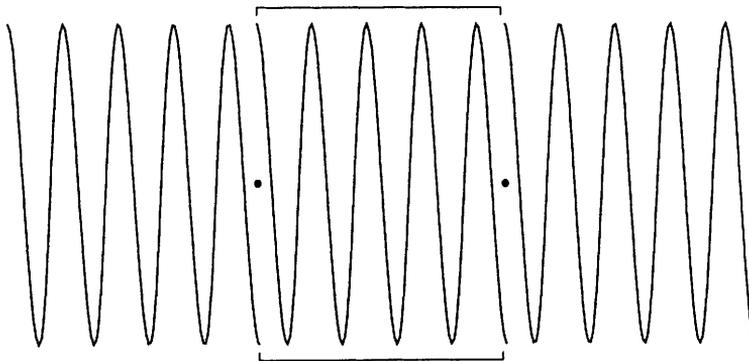


Figure 13.5: The effect of windowing with a noninteger number of cycles in the DFT buffer. Here we see a signal with $4\frac{1}{2}$ cycles in the buffer. After replication to the left and right the signal has the maximum possible discontinuity.

effectively becomes discontinuous. For example, a signal that has $4\frac{1}{2}$ cycles in the DFT buffer really looks like Figure 13.5 as far as the DFT is concerned. The discontinuities evident in the signal, like all discontinuities, require a wide range of frequencies to create; and the more marked the discontinuity the more frequencies required. Alternatively, we can explain the effect in terms of the Gibbs phenomenon of Section 3.5; the discontinuity generated by the forced periodicity causes ripples in the spectrum that don't go away.

Many ways have been found to fix this problem, but none of them are perfect. The most popular approaches compel continuity of the replicated signal by multiplying the signal in the buffer by some *window function* w_n . A plethora of different functions w_n have been proposed, but all are basically positive valued functions defined over the buffer interval $0 \dots N - 1$ that are zero (or close to zero) near the edges $w_0 \approx 0, w_N \approx 0$, but unity (or close to unity) near the middle $w_{N/2} \approx 1$. Most window functions (as will be discussed in more detail in Section 13.4) smoothly increase from zero to unity and then decrease back in a symmetric fashion. The exception to this smoothness criterion is the *rectangular window* (i.e., the default practice of not using a window at all, multiplying all signal values outside the buffer by zero, and all those inside by unity). For nondefault window functions, the new product signal $s'_n = w_n s_n$ for which we compute the DFT is essentially zero at both ends of the buffer, and thus its replication contains no discontinuities. Of course it is no longer the same as the original signal s_n , but for *good* window functions the effect on the power spectrum is tolerable.

Why does multiplication by a *good* window function not completely distort the power spectrum? The effect can be best understood by considering the half-sine window $w_n = \sin(\pi \frac{n}{N})$ (which, incidentally, is the one window function that no one actually uses). Multiplying the signal by this window is tantamount to convolving the signal spectrum with the window's spectrum. Since the latter is highly concentrated about zero frequency, the total effect is only a slight blurring. Sharp spectral lines are widened, sharp spectral changes are smoothed, but the overall picture is relatively undamaged.

Now that we know how to correctly calculate the periodogram we can use it as a *moving* power spectrum estimator for signals that vary over time. We simply compute the DFT of a windowed buffer, shift the buffer forward in time, and compute again. In this way we can display a sonogram (Section 4.6) or average the periodograms in order to reduce the variance of the spectral estimate (Section 5.7). The larger the buffer the better the frequency resolution, and when computing the DFT using the FFT we almost always want the buffer length to be a power of two. When the convenient buffer length doesn't match the natural data buffer, we can zero-pad the buffer.

Although this zero-padding seems to increase the frequency resolution it obviously doesn't really add new information. We often allow the buffers to overlap (half-buffer overlap being the most prevalent choice). The reason is that the windowing reduces the signal amplitude over a significant fraction of the time, and we may thus miss important phenomena. In addition, the spectral estimate variance is reduced even by averaging overlapped buffers.

EXERCISES

- 13.3.1 Show directly, by expressing the sample s_{lN+m} outside the buffer in terms of the complex DFT coefficients s_k , that computing the N -point DFT corresponds to replicating the signal in the time domain.
- 13.3.2 Plot the energy in a far bin as a function of the size of the discontinuity. (It's enough to use a cosine of digital frequency $\frac{1}{4}$ and observe the DC.) Why isn't it practical to use a variable-length rectangular window to reduce leakage?
- 13.3.3 Is signal discontinuity really a necessary condition for leakage? If not, what is the exact requirement? (Hint: Try the sinusoid $\sin(2\pi(k + \frac{1}{2})/N)$.)
- 13.3.4 As the length of the buffer grows the number of discontinuities per time decreases, and thus we expect the spectral SNR to improve. Is this the case?
- 13.3.5 In the text we discussed the half-sine window function. Trying it for a frequency right on a bin boundary (i.e., maximal discontinuity) we find that it works like a charm, but not for other frequencies. Can you explain why?

13.4 Windows

In Sections 4.6 and 13.3 we saw the general requirements for window functions, but the only explicit examples given were the rectangular window and the somewhat unusual half-sine window. In this section we will become acquainted with many more window functions and learn how to 'window shop', that is, to choose the window function appropriate to the task at hand.

Windows are needed for periodograms, but not only for periodograms. Windows are needed any time we chop up a signal into buffers and the signal is taken to be periodic (rather than zero) outside the observation buffer. This is a very frequent occurrence in DSP! When calculating autocorrelations (see Chapter 9) the use of windows is almost universal; a popular technique of designing FIR filters is based on truncating the desired impulse response

by a window (see Section 7.8); sample buffers are windowed before LPC analysis (see Section 9.9); and the list goes on and on. Yet windowing as a preprocessing stage for the periodogram is probably the best known use, and we will concentrate on it here. Recalling the interpretation of the FT as a bank of FIR band-pass filters, we will see that the frequency response of these filters is directly determined by the window function used.

We must, once again, return to the issue of buffer indexing. The computer programming convention that the buffer index runs from 0 to $N - 1$ is usually used with a window that obeys $w_0 = 0$ and $w_N = 0$. In this fashion the first point in the output buffer is set to zero but the last point is not (the N^{th} point, which *is* zero, belongs to the next buffer). Some people cannot tolerate such asymmetry and make either both $w_0 = 0, w_{N-1} = 0$ or $w_{-1} = 0, w_N = 0$. These conventions should be avoided! The former implies two zeroed samples in the replicated signal, the latter none. In theoretical treatments the symmetric buffer indexation $-M \dots M$ with $M \equiv \frac{N}{2}$ is common, and here only one of the endpoints is to be considered as belonging to the present buffer. To make things worse the buffer length may be even or odd, although FFT buffers will usually be of even length. As a consequence you should always check your window carefully before looking through it. We will present expressions in two formats, the practical $0 \dots N - 1$ with even N and $w_0 = 0, w_N = 0$ and the symmetric odd length $-M \dots M$ with $w_{\pm m} = 0$ and thus $N = 2M + 1$. To differentiate we will use an index n for the former case and m for the latter.

The rectangular window is really not a window function at all, but we consider it first for reference. Measuring analog time in units of our sampling interval, we can define an analog window function $w(t)$ that is one between $t = -M$ and $t = +M$ and zero elsewhere. We know that its FT is

$$W(\omega) = M \operatorname{sinc}(M\omega)$$

and its main lobe (defined between the first zeros) is of width $\frac{2\pi}{M}$. As M increases the main lobe becomes narrower and taller, but if we increase the frequency resolution, as allowed by the uncertainty theorem, we find that the number of frequency bins remains the same. In fact in the digital domain the $N = 2M$ point FFT has a frequency resolution of $\frac{1}{N}$ (the sampling frequency is one), and thus the main lobe is two frequency bins in width for all M . It isn't hard to do all the mathematics in the digital domain either. The digital window is $w_m = 1$ for $-M \leq m \leq +M$ and $w_m = 0$ elsewhere.

The DFT is given by

$$W_k = \sum_{m=-M}^M e^{-ikm} = e^{-ikM} \frac{1 - e^{-ik} e^{-2ikM}}{1 - e^{-ik}} = \frac{\sin(\frac{1}{2}Nk)}{\sin(\frac{1}{2}k)}$$

where we have used formula (A.46) for the sum of a finite geometric series, and substituted $N = 2M + 1$.

From this expression we can derive everything there is to know about the rectangular window. Its main lobe is two bins in width, and it has an infinite number of sidelobes, each one bin in width. Its highest sidelobe is attenuated 13.3 dB with respect to the main lobe, and the sidelobes decay by 6 dB per octave, as expected of a window with a discontinuity (see Section 4.2).

Before we continue we need some consistent quantities with which to compare windows. One commonly used measure is the *noise bandwidth* defined as the bandwidth of an ideal filter with the same maximum gain that would pass the same amount of power from a white noise source. The noise bandwidth of the rectangular window is precisely one, but is larger than one for all other windows. Larger main lobes imply larger noise bandwidths. Another important parameter is the ripple of the frequency response in the pass-band. The rectangular window has almost 4 dB pass-band ripple, while many other windows have much smaller ripple. We are now ready to see some nontrivial windows.

Perhaps the simplest function that is zero at the buffer ends and rises smoothly to one in the middle is the triangular window

$$w_n = 1 - \left| \frac{n - M}{M} \right| \quad \left| \quad \right. \quad w_m = 1 - \frac{|m|}{M + 1} \quad (13.4)$$

which is also variously known as the Bartlett window, the Fejer window, the Parzen window, and probably a few dozen more names. This window rises linearly from zero to unity and then falls linearly back to zero. If the buffer is of odd length there is a point in the middle for which the window function is precisely unity, for even length buffers all values are less than one. The highest sidelobe of the triangular window is 26 dB below the main lobe, and the sidelobes decay by 12 dB per octave, as expected of a window with a first derivative discontinuity. However, the noise bandwidth is 1.33, because the main lobe has increased in width.

The Hanning window is named after the meteorologist Julius von Hann.

$$w_n = \frac{1}{2} \left(1 - \cos \left(2\pi \frac{n}{N} \right) \right) \quad \text{for } n = 0 \dots N - 1 \quad (13.5)$$

Apparently the verb form ‘to Hann the data’ was used first; afterward people started to speak of ‘Hanning the signal’, and in the end the analogy with the Hamming window (see below) caused the adoption of the misnomer ‘Hanning window’. The Hanning window is also sometimes called the ‘cosine squared’, or ‘raised cosine’ window (use the ‘m’ index to see why). The Hanning window’s main lobe is twice as wide as that of the rectangular window, and at least three spectral lines will always be excited, even for the best case. The noise bandwidth is 1.5, the highest sidelobe is 32 dB down, and the sidelobes drop off by 18 dB per octave.

The Hamming window is named in honor of the applied mathematician Richard Wesley Hamming, inventor of the Hamming error-correcting codes, creator of one of the first programming languages, and author of texts on numerical analysis and digital filter design.

$$w_n = 0.54 - 0.46 \left(1 - \cos \left(2\pi \frac{n}{N} \right) \right) \quad \text{for } n = 0 \dots N - 1 \quad (13.6)$$

The Hamming window is obtained by modifying the coefficients of the Hanning window in order to precisely cancel the first sidelobe, but suffers from not becoming precisely zero at the edges. For these reasons the Hamming window has its highest sidelobe 42 dB below the main lobe, but asymptotically the sidelobes only decay by 6 dB per octave. The noise bandwidth is 1.36, close to that of the triangular window.

Continuing along similar lines one can define the Blackman-Harris family of windows

$$w_n = a_0 - a_1 \cos \left(2\pi \frac{n}{N} \right) + a_2 \cos \left(2\pi \frac{2n}{N} \right) - a_3 \cos \left(2\pi \frac{3n}{N} \right) \dots \quad (13.7)$$

and optimize the parameters in order to minimize sidelobes. More complex window families include the Kaiser and Dolph-Chebyshev windows, which have a free parameter that can be adjusted for the desired trade-off between sidelobe height and main-lobe width. We superpose several commonly used windows in Figure 13.6.

Let’s see how these windows perform. In Figure 13.7 we see the periodogram spectral estimate of a single worst-case sinusoid using several different windows. We see that the rectangular window is by far the worst, and that the triangular and then the Hanning windows improve upon it.

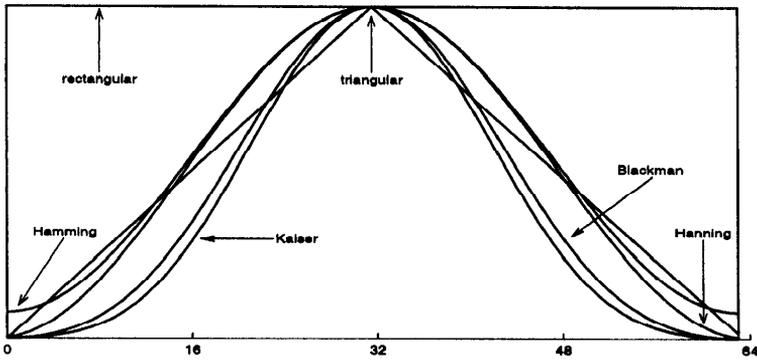


Figure 13.6: Various window functions. Depicted are 64-point rectangular, triangular, Hanning, Hamming, Blackman, and Kaiser windows.

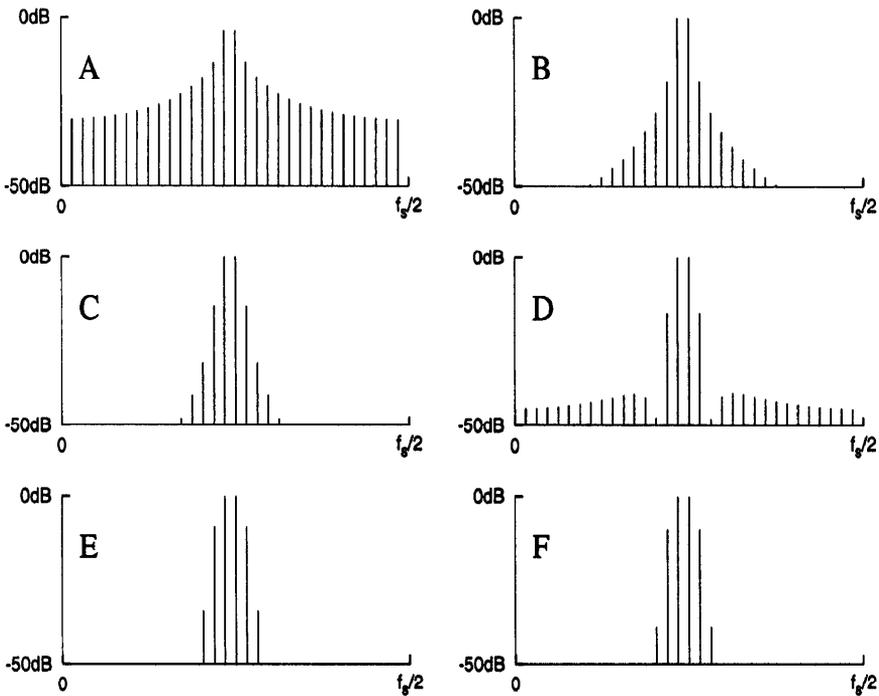


Figure 13.7: Periodogram of worst-case single sinusoids using various window functions, namely (A) rectangular, (B) triangular, (C) Hanning, (D) Hamming, (E) Blackman-Harris, and (F) Kaiser. Each periodogram is normalized such that its maximum height corresponds to 0 dB.

Afterward the choice is not clear cut. The Blackman and Kaiser windows reduce the sidelobe height, but cannot simultaneously further reduce the main lobe width. The Hamming window attempts to narrow the main lobe, but ends up with higher distant sidelobes. Not shown is a representative of the Dolph-Chebyshev family, which as can be assumed for anything bearing the name Chebyshev, has constant-height sidelobes.

Which window function is best? It all depends on what you are trying to do. Rectangular weighting could be used for sinusoids of precisely the right frequencies, but don't expect that to ever happen accidentally. If you are reasonably sure that you have a single clean sinusoid, this may be verified and its frequency accurately determined by using a mixer and a rectangular window STFT; just remember that the signal's frequency is the combination of the bin's frequency and the mixer frequency. An even trickier use of the rectangular window is for the probing of linear systems using synthetically generated pseudorandom noise inputs (see Section 5.4). By using a buffer length precisely equal to the periodicity of the pseudorandom signal we can ensure that all frequencies are just right and the rectangular weighted STFT spectra are beautiful. Finally, rectangular windows should be used when studying *transients* (signals that are nonzero only for a short time). We can then safely place the entire signal inside the buffer and guarantee zero signal values at the buffer edges. In such cases the rectangular window causes the least distortion and requires the least computation.

For general-purpose frequency displays the Hanning and Hamming windows are often employed. They have lower sidebands and lower pass-band ripple than the rectangular window. The coefficients of the Hanning window needn't be stored, since they are derivable from the FFT's twiddle factor tables. Another trick is to overlap and average adjacent buffers in such a way that the time weighting becomes constant.

A problem we haven't mentioned so far is two-tone separability. We sometimes need to separate two closely spaced tones, with one much stronger than the other. Because of main lobe width and sidelobe height, the weaker tone will be covered up and not noticeable unless we choose our window carefully. For such cases the Blackman, Dolph-Chebyshev, or Kaiser windows should be used, but we will see stronger methods in the following sections.

EXERCISES

- 13.4.1 Convert the Hanning and Hamming windows to symmetric ‘m’ notation and explain the names ‘cosine squared’ and ‘raised cosine’ often applied to the former. Express the Hanning window as a convolution in the frequency domain. What are the advantages of this approach?
- 13.4.2 Plot the periodograms for the same window functions as in Figure 13.7, but for a best-case sinusoid (e.g., for $N = 64$, a sinusoid of frequency $15/64$).
- 13.4.3 Plot periodograms of the logistics signal for various $1 \leq \lambda < 3.57$, as was done in Section 5.5. Which window is best? Now use λ that give for 3, 5, and 6 cycles. Which window should be used now?
- 13.4.4 Try to separate two close sinusoids, both placed in worst-case positions, and one much stronger than the other. Experiment with different windows.

13.5 Finding a Sinusoid in Noise

As we mentioned above, frequency estimation is simplest when we are given samples of a single clean sinusoid. Perhaps the next simplest case is when we are told that the samples provided are of a single sinusoid with additive uncorrelated white noise; but if the SNR is low this ‘simple’ case is not so simple after all. To use averaging techniques as discussed in Section 5.3 one would have to know a priori how to perform the registration in time before averaging. Unfortunately, this would require accurate knowledge of the frequency, which is exactly what we are trying to measure in the first place! We could perform an FFT, but that would only supply us with the frequency of the nearest bin; high precision would require using a large number of signal points (assuming the frequency were constant over this time interval), and most of the computation would go toward finding bins of no interest. We could calculate autocorrelations for a great number of lags and look for peaks, but the same objections hold here as well.

There are more efficient ways of using the autocorrelations. Pisarenko discovered one method of estimating the frequencies of p sinusoids in additive white noise using a relatively small number of autocorrelation lags. This method, called the Pisarenko Harmonic Decomposition (PHD), seems to provide an infinitely precise estimate of these frequencies, and thus belongs to the class of ‘super-resolution’ methods. Before discussing how the PHD circumvents the basic limitations of the uncertainty theorem, let’s derive it for the simple case of a single sinusoid ($p = 1$).

We assume that our signal is exactly of the form

$$s_n = A \sin(\omega n + \phi) + \nu_n \quad (13.8)$$

where ν is the uncorrelated white noise. Its autocorrelations are easily derived

$$C_s(m) = \langle s_n s_{n+m} \rangle = \frac{A^2}{2} \cos(\omega m) + \sigma_\nu^2 \delta_{m,0} \quad (13.9)$$

and the first few lags are given by the following.

$$\begin{aligned} C_s(0) &= \frac{A^2}{2} + \sigma_\nu^2 \\ C_s(1) &= \frac{A^2}{2} \cos(\omega) \\ C_s(2) &= \frac{A^2}{2} \cos(2\omega) = \frac{A^2}{2} (2 \cos^2(\omega) - 1) \end{aligned}$$

The noise only influences the lag zero term (energy) due to the assumption of white noise. Any deviation from whiteness causes the other lags to acquire noise-related terms as well.

Were the noise to be zero, we could simply calculate

$$\omega = \cos^{-1} \left(\frac{C_s(1)}{C_s(0)} \right)$$

but this fails miserably when noise is present. Can we find an expression that uses only nonzero lags, and is thus uninfluenced by the noise? Pisarenko's method uses only the two lags $m = 1$ and $m = 2$. Using the trigonometric identity $\cos(2\omega) = 2 \cos^2(\omega) - 1$ it is easy to show that

$$2C_s(1)c^2 - C_s(2)c - C_s(1) = 0$$

where we have denoted $c = \cos(\omega)$. This is a simple quadratic, with solutions

$$c = \frac{C_s(2)}{4C_s(1)} \pm \frac{1}{2} \sqrt{\frac{C_s^2(2)}{4C_s^2(1)} + 2} \quad (13.10)$$

only one of which leads to the correct solution (see exercise 13.5.2). We thus find

$$\omega = \cos^{-1} \left(\frac{C_s(2)}{4C_s(1)} + \frac{1}{2} \operatorname{sgn}(C_s(1)) \sqrt{\frac{C_s^2(2)}{4C_s^2(1)} + 2} \right) \quad (13.11)$$

which is the PHD estimate for the digital angular frequency (the analog frequency is obtained by dividing by 2π and multiplying by the sampling frequency).

The PHD expression we have just found is not a frequency *estimate* at all. Assuming the noise is perfectly white, it is an infinitely precise measurement of the frequency. Of course there is no problem with this infinite precision since we assumed that we have exact values for the two autocorrelation lags $C_s(1)$ and $C_s(2)$. Obtaining these exact values requires knowing the signal over all time, and therefore the uncertainty theorem does allow infinitely precise predictions. However, even when we use empirical autocorrelations (equation (9.11)) calculated using only N samples the prediction still seems to be perfectly precise. Unlike periodogram methods there is no obvious *precision* reduction with decreasing N ; but the *accuracy* of the prediction decreases. It is straightforward, but somewhat messy, to show that the variance of the PHD estimator is inversely proportional to the size of the buffer and the square of the SNR ($\text{SNR} = \frac{A^2}{2\sigma^2}$).

$$\sigma_{PHD}^2 = \frac{\cos^2(2\omega) + \cos^2(\omega)}{\sin^2(\omega)(\cos(2\omega) + 2)^2} \frac{1}{N \text{SNR}^2}$$

The somewhat complex frequency-dependent prefactor means that the estimate is more accurate near DC ($\omega = 0$) and Nyquist ($\omega = \pi$), and there is a small dip near the middle of the range. More interesting is the N dependence; the proper Δf is the standard deviation, and so we have a strange $(\Delta f)^2 \Delta t$ uncertainty product. Even more disturbing is the SNR dependence; as the SNR increases the error decreases even for small N . It is obvious that this error only reflects better noise cancellation with more data points, and not true uncertainty theorem constraints.

So it seems that the PHD really does beat the uncertainty theorem. The explanation is, however, deceptively simple. We made the basic assumption that the signal was exactly given by equation (13.8). Once the parameters of the sinusoid are known, the signal (without the noise) is known *for all times*. The uncertainty product effectively has $\Delta t = \infty$ and can attain infinitesimal frequency precision. This is the idea behind all model-based super-resolution methods. The data is used to find the parameters of the model, and the model is assumed to hold for all time. Thus, assuming that the assumption holds, the uncertainty theorem is robbed of its constraining influence.

EXERCISES

- 13.5.1 Derive the expression (13.9) for the autocorrelation (use exercise 9.2.12).
- 13.5.2 Exploiting the fact that we want $0 \leq \omega < \pi$ show that the proper solution of the quadratic has the sign of $C_s(1)$.
- 13.5.3 In the text we quoted the variance of the error of the PHD estimation. What about its bias? Find this numerically for various buffer sizes.
- 13.5.4 The PHD is a second-order frequency estimator in the sense that the highest autocorrelation lag it utilizes is $m = 2$. Using the trigonometric identity $\cos(\omega) + \cos(3\omega) = 2\cos(\omega)\cos(2\omega)$ prove that $C_s(1) - 2C_s(2)c + C_s(3) = 0$. Show that this leads to the following third-order estimator.

$$\omega = \cos^{-1} \left(\frac{C_s(1) + C_s(3)}{2C_s(2)} \right)$$

- 13.5.5 Compare the third-order estimator of the previous exercise with the PHD by generating sinusoids in various amounts of white noise and estimating their frequencies. Which is better for low SNR? High SNR? Small buffer size? Large buffer size?

13.6 Finding Sinusoids in Noise

The previous section dealt with the special case of a single sinusoid in noise. Here we extend the PHD to multiple sinusoids. The needed formalism is a bit more mathematically demanding (involving the roots of functions that are derived from the eigenvector of the signal covariance matrix belonging to the smallest eigenvalue), so we approach it cautiously.

In order to derive the PHD for the sum of p sinusoids in uncorrelated white noise,

$$s_n = \sum_{i=1}^p A_i \sin(\omega_i n) + \nu_n$$

we first rederive the $p = 1$ case in a different way. Recall that exponentials and sinusoids obey difference equations; those of real exponentials involve a single previous value, while sinusoids obey recursions involving two previous values. From equation (6.52) we know that a clean sinusoid $x_n = A \sin(\omega n)$ obeys the following recursion

$$x_n = a_1 x_{n-1} + a_2 x_{n-2} \quad \text{where} \quad \begin{cases} a_1 = 2 \cos(\omega) \\ a_2 = -1 \end{cases} \quad (13.12)$$

(we have simply defined $a_1 = -c_1$ and $a_2 = -c_2$). We will call a_1 and a_2 *recursion coefficients*. Given the recursion coefficients we can write the equation

$$1 - a_1 z^{-1} - a_2 z^{-2} = 0 \quad \text{or} \quad z^2 - 2 \cos(\omega)z + 1 = 0 \quad (13.13)$$

which has the following solutions.

$$z = \frac{1}{2} \left(2 \cos(\omega) \pm \sqrt{4 \cos^2(\omega) - 4} \right) = \cos(\omega) \pm i \sin(\omega) = e^{\pm i\omega}$$

Thus those z that solve equation (13.13) (i.e., the roots of the polynomial therein specified) are on the unit circle, and their angles are the frequencies (both positive and negative) of the original signal. This is a link between the recursion coefficients and the frequencies of the signal that obeys the recursion.

The connection between this and the PHD is easiest to understand in vector notation. We define the vectors

$$\begin{aligned} \underline{x} &= (x_n, x_{n-1}, x_{n-2}) \\ \underline{a} &= (1, -a_1, -a_2) \end{aligned}$$

so that equation (13.12) is written

$$\underline{x} \cdot \underline{a} = 0 \quad (13.14)$$

i.e., the clean signal vector and the recursion coefficient vector are orthogonal. Now the noisy signal is $\underline{s} = \underline{x} + \underline{\nu}$. This signal has mean zero (since we assume the noise to be zero mean) and its covariance matrix is thus the signal autocorrelation matrix

$$\underline{\underline{V}}_s = \langle \underline{s} \underline{s}^t \rangle = \langle (\underline{x} + \underline{\nu})(\underline{s}^t + \underline{\nu}^t) \rangle = \underline{\underline{V}}_x + \sigma_\nu^2 \underline{\underline{I}} \quad (13.15)$$

where $\underline{\underline{V}}_s$ is a 3-by-3 square matrix with elements $V_{s,i,j} = C_s(i-j)$. The first term $\underline{\underline{V}}_x = \langle \underline{x} \underline{x}^t \rangle$ is a symmetric Toeplitz matrix, the matrix $\underline{\underline{I}}$ is the 3-by-3 identity matrix, and σ_ν^2 is the variance of the noise. It is now easy to see that

$$\underline{\underline{V}}_s \underline{a} = \left(\langle \underline{x} \underline{x}^t \rangle + \sigma_\nu^2 \right) \underline{a} = \langle \underline{x} (\underline{x} \cdot \underline{a}) \rangle + \sigma_\nu^2 \underline{a} = \sigma_\nu^2 \underline{a} \quad (13.16)$$

which shows that \underline{a} is an eigenvector of the covariance matrix. Since eigenvalues of $\underline{\underline{V}}_x$ are nonnegative, σ_ν^2 must be the smallest eigenvalue of $\underline{\underline{V}}_s$. We

thus see that the frequency of a sinusoid in additive noise can be determined by diagonalizing its autocorrelation matrix. This is a specific case of the desired formulation of the PHD.

Theorem: The Pisarenko Harmonic Decomposition

Given a signal s that is the sum of p sinusoids and uncorrelated white noise,

$$s_n = \sum_{i=1}^p \left(A_i e^{i\omega_i n} + A_i^* e^{-i\omega_i n} \right) + \nu_n$$

denote by $\underline{\underline{V}}_s$ the $(2p + 1)$ -dimensional covariance matrix of this signal, and by \underline{a} the eigenvector of $\underline{\underline{V}}_s$ that belongs to the minimal eigenvalue. Then the roots of $1 - \sum_{k=1}^p a_k z^{-k}$ are of the form $z_i = e^{\pm i\omega_i}$. ■

We can now understand the term *decomposition* that appears in the name PHD. The decomposition is that of the covariance matrix into signal-related and noise-related parts (see equation (13.15)) which implies the splitting of the $(2p + 1)$ -dimensional space of signal vectors into orthogonal signal (equation (13.14)) and noise subspaces.

The proof of the general p case is similar to that of the $p = 1$ case. The key idea is that signals consisting of p real exponentials or sinusoids obey difference equations; those of real exponentials involve p previous values, while p sinusoids obey recursions involving $2p$ previous values. It's easier to understand this by first considering the exponential case.

$$x_n = \sum_{i=1}^p A_i e^{q_i n}$$

This can be expressed as the combination of p previous values.

$$x_n = \sum_{k=1}^p a_k x_{n-k} \tag{13.17}$$

Substituting the definition

$$x_n = \sum_{k=1}^p a_k \left(\sum_{i=1}^p A_i e^{q_i(n-k)} \right) = \sum_{i=1}^p A_i \left(\sum_{k=1}^p a_k e^{-q_i k} \right) e^{q_i n}$$

and thus

$$\sum_{k=1}^p a_k e^{-q_i k} = 1$$

we see that

$$1 - \sum_{k=1}^p a_k z^{-k}$$

has roots $z = e^{q_i}$. Similarly for the sum of sinusoids

$$x_n = \sum_{i=1}^p \left(A_i e^{i\omega n} + A_i^* e^{-i\omega n} \right)$$

we leave it as an exercise to show that

$$x_n = \sum_{k=1}^{2p} a_k x_{n-k} \quad (13.18)$$

where

$$1 - \sum_{k=1}^{2p} a_k z^{-k} \quad (13.19)$$

has roots $z_i = e^{\pm i\omega_i}$.

In practice we do not have the real covariance matrix, and Pisarenko's method uses the usual empirical estimates for $C_s(0), C_s(1), C_s(2), \dots, C_s(2p)$. Once the covariance matrix has been estimated, we diagonalize it or use any available method for finding the eigenvector belonging to the minimal eigenvalue. This produces the recursion coefficients with which we can build the polynomial in (13.19) and find its roots numerically. Finally we obtain the desired frequencies from the angles of the roots.

The PHD is only one of several frequency estimation methods that use eigenvector decomposition of the signal covariance matrix. Another popular eigenvector method, called *MUSIC* (MULTiple SInal Classification), provides a full spectral distribution, rather than the p discrete lines of the PHD.

Alternative approaches are based on inverting the covariance matrix rather than diagonalizing it. Baron de Prony worked out such an algorithm for a similar problem back in 1795! Prony wanted to approximate N equally-spaced data points by the sum of p real exponentials (as in equation (13.17)). There are precisely $2p$ free parameters in the parametric form, so he needed $N = 2p$ data points. Were the $e^{q_i t}$ factors known, finding the A_i would be reduced to the solution of p simultaneous linear equations; but the q_i appear in an exponent, creating a very nonlinear situation. Prony's idea was to find the q_s first, using the recursion relating the data values. For exponentials the recursion is equation (13.17) for all n . Given N signal values, x_0, x_1, \dots, x_{N-1} , we consider only the $N - p$ equations for which all the required signal values are in the buffer.

$$\begin{aligned}
 x_p &= a_1 x_{p-1} + a_2 x_{p-2} + \cdots + a_p x_0 \\
 x_{p+1} &= a_1 x_p + a_2 x_{p-1} + \cdots + a_p x_1 \\
 &\vdots \\
 x_{N-1} &= a_1 x_{N-2} + a_2 x_{N-3} + \cdots + a_p x_{N-p-1}
 \end{aligned}$$

This can be written in matrix form

$$\begin{pmatrix}
 x_{p-1} & x_{p-2} & x_{p-3} & \cdots & x_0 \\
 x_p & x_{p-1} & x_{p-2} & \cdots & x_1 \\
 x_{p+1} & x_p & x_{p-1} & \cdots & x_1 \\
 \vdots & \vdots & \vdots & \vdots & \vdots \\
 x_{N-2} & x_{N-3} & x_{N-4} & \cdots & x_{N-p-1}
 \end{pmatrix}
 \begin{pmatrix}
 a_1 \\
 a_2 \\
 a_3 \\
 \vdots \\
 a_p
 \end{pmatrix}
 =
 \begin{pmatrix}
 x_p \\
 x_{p+1} \\
 x_{p+2} \\
 \vdots \\
 x_{N-1}
 \end{pmatrix}$$

and for $N = 2p$ the matrix is square Toeplitz and the equations can be readily solved for the a_k . Once we have the a_k we can find the roots of the polynomial, and retrieve the q_i . Thereafter the A_i can be found as explained above. Thus Prony's method reduces the solution of a very nonlinear problem to the solution of two linear problems and the (nonlinear) operation of finding the roots of a polynomial.

EXERCISES

- 13.6.1 Why are the eigenvalues of $\underline{\underline{C_x}}$ nonnegative?
- 13.6.2 Complete the proof of the PHD for general p . To do this prove equation (13.18) and the claim about the roots.
- 13.6.3 Specialize the PHD back to $p = 1$ and show that we obtain our previous PHD equation (13.11).
- 13.6.4 What is the computational complexity of the PHD for general p ?
- 13.6.5 Prony's method as described works only for noiseless signals. How can it be extended to the noisy case?
- 13.6.6 Extend Prony's method to p sinusoids.

13.7 IIR Methods

Armed with the ideas acquired in the previous sections, we return to the problem of estimating the entire power spectral distribution from samples in the time domain. In Section 4.6 we saw that the DFT periodogram can be a powerful spectral *estimator*, but does not produce the exact spectrum due to the signal only being observed for short times. In Section 13.2 we saw that the STFT is essentially a bank of FIR filters. Can we improve on the periodogram by using a bank of IIR filters?

Recall that the DFT is simply the zT

$$s(z) = zT(s_n) = \sum_{n=-\infty}^{\infty} s_n z^{-n}$$

calculated on the unit circle. Thus corresponding to the moving STFT there is a STzT

$$s_m(z) = \sum_{n=m-N+1}^m s_n z^{-n} \quad (13.20)$$

where we have not explicitly shown a window function and have chosen the causal indexing convention. At time $n = 0$ this reduces to

$$s_0(z) = \sum_{n=-N+1}^0 s_n z^{-n} = \sum_{n=0}^{N-1} s_{-n} z^n \quad (13.21)$$

an $(N-1)^{\text{th}}$ degree polynomial in z . By comparison, the full zT is an infinite Laurent series,

$$s(z) = \sum_{n=-\infty}^{\infty} s_n z^{-n} = \sum_{n=1}^{\infty} s_n z^{-n} + \sum_{n=0}^{\infty} s_{-n} z^n$$

and the STzT can be considered to be a polynomial approximation to these infinite sums.

What kind of approximation to the infinite Laurent series is the polynomial? It is obviously an *all-zero* approximation since no poles in the z -plane can be produced, only zeros. Spectra with no discrete lines (delta functions) are well approximated by such polynomials, but spectra with sharp resonances are not. Sharp features such as delta functions are better captured by an approximation such as

$$s(z) \approx \frac{1}{\sum_{m=0}^M b_m z^m} \quad (13.22)$$

which is obviously an *all-pole* approximation to the full zT . All-pole approximations may efficiently describe resonances that would take dozens of coefficients in the all-zero model; and like the Pisarenko estimate the frequencies of the poles may be measured to higher resolutions than the STFT allows. To capture both zeros and poles in the true spectrum we had best consider an ARMA model.

$$s(z) \approx \frac{\sum_{n=0}^N a_n z^n}{\sum_{m=0}^M b_m z^m} \quad (13.23)$$

In order to use the all-pole model of equation (13.22) we need a method of finding the coefficients b_m , but these are precisely the LPC coefficients of Section 9.9. We saw there how to set up the Yule-Walker equations and solve them using the Levinson-Durbin recursion. Once we have them, what is the explicit connection between the LPC coefficients and the AR spectrum of the signal? From

$$H(z) = \frac{G}{1 + \sum_{m=1}^M b_m z^{-m}} = \frac{G}{\sum_{m=0}^M b_m z^{-m}}$$

it is straightforward to obtain the power spectrum by restricting to the unit circle.

$$|H(\omega)|^2 = \frac{G^2}{|1 + \sum_{k=1}^p a_k e^{-i\omega k}|^2} = \frac{G^2}{|\sum_{k=0}^p a_k e^{-i\omega k}|^2} \quad (13.24)$$

Which type of approximation is best, all-zero, all-pole, ARMA? The answer depends on the problem at hand. Speech signals tend to have spectra with resonant peaks called formants caused by the geometry of the vocal tract (see Section 19.1). Such spectra are most naturally approximated by all-pole models. All-zero DFT based methods are better for spectra containing narrow valleys but no peaks, such as noise that passed through notch filters. In any case arbitrary spectra can be approximated by either all-pole or all-zero models by using high enough orders. From this point of view, the incentive behind choosing a model is one of efficiency.

Yet there is another reason for choosing an all-pole model. The Wiener-Khintchine theorem relates the power spectrum to the infinite set of autocorrelations $C_s(m)$ for all lags m . In practice we can compute only a limited number of autocorrelations, and would like to estimate the power spectrum based on these. We might assume that all unknown autocorrelations are exactly zero,

$$S(\omega) = \sum_{m=-\infty}^{\infty} C_s(m) e^{-im\omega} \rightarrow \sum_{m=-M}^M C_s(m) e^{-im\omega} \quad (13.25)$$

which is not a bad assumption if the autocorrelations die off rapidly enough. This is easily seen to be an all-zero approximation, and leads to the blurring of sharp spectral lines. In 1967, John Burg introduced an alternative assumption, that the spectral estimate should be the most random spectrum consistent with the lags we *do* have. By 'most random' we mean the spectrum with the highest 'entropy', and thus this technique is called *maximum entropy* spectral analysis.

The reasoning behind the maximum entropy principle is easy to understand. DFT methods assume that all data that has not been observed either consist of periodic repetition of the data we have seen or are identically zero. There is usually little physical evidence for such assumptions! Maximum entropy means that we should remain as open minded as possible regarding unseen data. Indeed Burg's method actually tells us to use the most *unpredictable* extrapolation of the data possible. There are many possible spectra that are consistent with the data we have collected, each corresponding to a different extrapolation of the data; maximum entropy insists that the most likely spectrum is that corresponding to the least constraints on the unknown data. In other words we should assume that the uncollected data is as random as possible.

What type of approximation corresponds to the maximum entropy assumption? In Section 18.6 we will see, and if you have studied thermodynamics you already know, that maximum randomness means maximization of the 'entropy'. We assume that the entropy

$$H \equiv \int \ln S(\omega) d\omega \quad (13.26)$$

is maximized under the constraint that the *observed* autocorrelation lags (those with lags $|m| \leq M$) do indeed obey Wiener-Khintchine.

$$C_s(m) = \frac{1}{2\pi} \int S(\omega) e^{im\omega} \quad (13.27)$$

The integral in equation (13.26) depends on all the autocorrelations, not just the known ones, and the maximum we seek is for all possible values of the unknown autocorrelations. We differentiate with respect to all the autocorrelations with $|m| > M$ and set equal to zero.

$$0 = \frac{\partial H}{\partial C_s(m)} = \int \frac{d \ln S}{dS} \frac{\partial S(\omega)}{\partial C_s(m)} d\omega = \int \frac{1}{S(\omega)} e^{-im\omega} d\omega$$

We see that the Fourier coefficients of the reciprocal of $S(\omega)$ are zero for $|m| > M$ (i.e., the inverse spectrum is a finite Fourier series). Accordingly, the maximum entropy power spectrum can be written as the reciprocal of a finite Fourier series, that is, is all-pole.

EXERCISES

- 13.7.1 Generate a signal that is the sum of a small number of sinusoids and noise. Find the PSD via the periodogram. Solve the LPC equations and derive the spectrum using equation (13.24). Compare the results. Experiment by placing weak spectral lines close to strong ones (recall exercise 13.4.4).
- 13.7.2 Record some speech and compute its all-pole spectrum. What features do you observe? Can you recognize different sounds from the PSD?

13.8 Walsh Functions

As we saw in Section 3.4, the Fourier transform is easily computable because of the orthogonality of the sinusoids. The sinusoids are in some senses the simplest orthogonal family of functions, but there are other families that are simple in other ways. The Walsh functions, the first few of which are depicted in Figure 13.8, are an interesting alternative to the sinusoids. They are reminiscent of square waves, but comprise a complete orthogonal family. Like square waves all of the signal values are ± 1 ; due to this characteristic the Walsh transform can be computed without any true multiplications at all.

It is conventional to define the Walsh functions recursively. For the unit interval $0 \leq t \leq 1$ we define

$$\begin{aligned} \text{wal}^{[0]}(t) &= 1 & (13.28) \\ \text{cal}^{[k]}(t) = \text{wal}^{[2k]}(t) &= \text{wal}^{[k]}(2t) + (-1)^k \text{wal}^{[k]}(2t - \tfrac{1}{2}) \\ \text{sal}^{[k+1]}(t) = \text{wal}^{[2k+1]}(t) &= \text{wal}^{[k]}(2t) - (-1)^k \text{wal}^{[k]}(2t - \tfrac{1}{2}) \end{aligned}$$

and assume all of the functions to be zero to $t < 0$ and $t > 1$. After thus defining the functions on the unit interval we extend the definitions periodically to the entire t axis. Note that the 'wal' functions are a single family like the complex exponentials, while the 'sal' and 'cal' functions are analogous

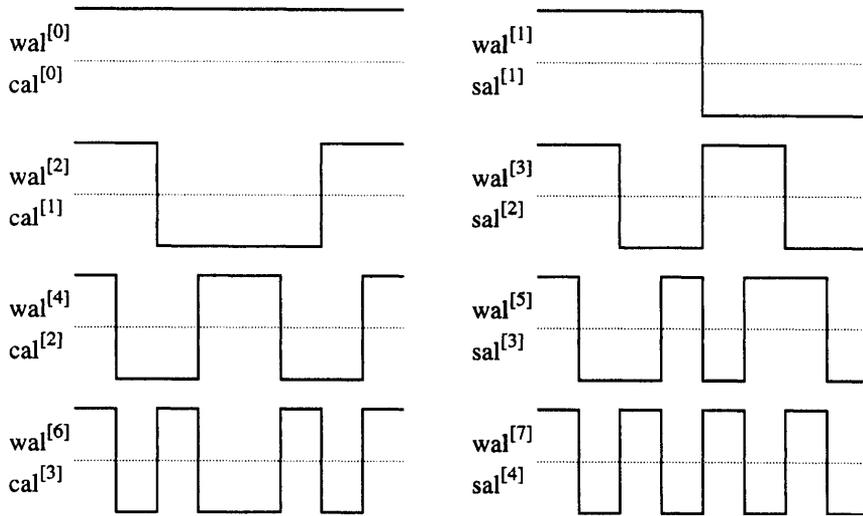


Figure 13.8: The first eight Walsh functions in order of increasing sequency. The cal functions are on the left and the sal functions on the right.

to sine and cosine. The label k equals the number of transitions in the unit interval and is called the *sequency*.

The value-by-value product of two Walsh functions is always a Walsh function

$$\text{wal}^{[i]}(t) \text{wal}^{[j]}(t) = \text{wal}^{[k]}(t)$$

where k is obtained by bit-by-bit xor of i and j . From this and the fact that all Walsh functions except $\text{wal}^{[0]}$ are DC-free it is easy to prove the required orthonormality property.

$$\int_0^1 \text{wal}^{[i]}(t) \text{wal}^{[j]}(t) dt = \delta_{i,j}$$

There is also a discrete version of this property.

$$\sum_{n=0}^{N-1} \text{wal}^{[i]}\left(\frac{n}{N}\right) \text{wal}^{[j]}\left(\frac{n}{N}\right) = N\delta_{i,j}$$

Analogously to the Fourier transform we can expand arbitrary signals as combinations of Walsh functions, thus defining the Walsh transform. Hence signals can be interpreted as functions in the time domain or sequency domain. In DSP we are more interested in the discrete Walsh transform (DWT)

$$\begin{aligned}
 X_k &= \sum_{n=0}^{N-1} x_n \text{wal}^{[k]} \left(\frac{n}{N} \right) \\
 x_n &= \frac{1}{N} \sum_{k=0}^{N-1} X_k \text{wal}^{[k]} \left(\frac{n}{N} \right)
 \end{aligned}
 \tag{13.29}$$

where the normalization was chosen according to our usual convention. This looks very similar to the DFT. The two-point transform is identical to that of the DFT

$$\begin{aligned}
 X_0 &= x_0 + x_1 \\
 X_1 &= x_0 - x_1
 \end{aligned}$$

but the four-point transform is simpler.

$$\begin{aligned}
 X_0 &= x_0 + x_1 + x_2 + x_3 \\
 X_1 &= x_0 + x_1 - x_2 - x_3 \\
 X_2 &= x_0 - x_1 - x_2 + x_3 \\
 X_3 &= x_0 - x_1 + x_2 - x_3
 \end{aligned}$$

Note that Walsh transforms are computed without nontrivial multiplications, requiring only N^2 additions. Analogously to the FFT, a fast Walsh transform (FWT) can be defined, reducing this to $O(N \log N)$ additions.

EXERCISES

- 13.8.1 Plot $\text{sal}(t)$ and overlay it with $\sin(2\pi t)$; similarly plot $\text{cal}(t)$ and overlay $\cos(2\pi t)$. What is the relationship?
- 13.8.2 What is the connection between the Walsh functions and the Hadamard matrices defined in exercise 14.5.3?
- 13.8.3 Find a nonrecursive formula for $\text{wal}^{[k]}(t)$.
- 13.8.4 Write a program that computes the decimation in sequency fast Walsh transform (see Section 14.3).
- 13.8.5 Since the DWT can be computed without multiplications and using only real arithmetic, it would be useful to be able to obtain the DFT from the DWT. How can this be done?
- 13.8.6 In the Fourier case multiplication is related to convolution. What is the analogous result for Walsh transforms?

13.8.7 Another purely real transform is the Hartley transform

$$\begin{aligned} X(f) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} x(t) \operatorname{cas}(\omega t) dt \\ x(t) &= \int_{-\infty}^{\infty} X(f) \operatorname{cas}(\omega t) df \end{aligned}$$

where we defined $\operatorname{cas}(t) = \cos(t) + \sin(t)$. Note that the Hartley transform is its own inverse (to within normalization). Similarly we can define the discrete Hartley transform.

$$\begin{aligned} X_k &= \frac{1}{N} \sum_{n=0}^{N-1} x_n \operatorname{cas}\left(\frac{2\pi nk}{N}\right) \\ x_n &= \sum_{k=0}^{N-1} X_k \operatorname{cas}\left(\frac{2\pi nk}{N}\right) \end{aligned}$$

How do you retrieve the power spectrum from the Hartley transform? Obtain the DFT from the discrete Hartley transform. Develop a fast Hartley transform (see Section 14.3).

13.9 Wavelets

No modern treatment of spectral analysis could be complete without mentioning wavelets. Although there were early precursors, wavelet theory originated in the 1980s when several researchers realized that spectral analysis based on basis functions that are localized in both frequency and time could be useful and efficient in image and signal processing.

What exactly is a wavelet? A basic wavelet is a signal $\psi(t)$ of finite time duration. For example, a commonly used basic wavelet is the sinusoidal pulse

$$\psi(t) = w(t) e^{i\omega_0 t} \quad (13.30)$$

where $w(t)$ is any windowing function such as a rectangular window, a sinc window or a raised cosine window. Such a pulse is only nonzero in the time domain in the vicinity of t_0 and only nonzero in the frequency domain near ω_0 . The STFT is based on just such functions with $w(t)$ being the window chosen (see Section 13.4) and ω_0 the center of a bin. From the basic wavelet we can make scaled and translated wavelets using the following transformation

$$\psi(\tau, t_0, t) = \frac{1}{\sqrt{\tau}} \psi\left(\frac{t - t_0}{\tau}\right) \quad (13.31)$$

where the prefactor normalizes the energy. The time translation t_0 is simple to understand, it simply moves the wavelet along the time axis. The time duration of the wavelet is proportional to τ ; conversely, you can think of the scaling transformation compressing the time scale for $\tau > 1$ and stretching it for $\tau < 1$. The center frequency is inversely proportional to τ (i.e., the frequency axis scales in the opposite way to the time axis).

What about the wavelet's bandwidth? Since the nonzero bandwidth results from the finite time duration via the uncertainty theorem, the bandwidth must scale inversely to τ . This last statement can be made more explicit by borrowing the filter design concept of the Q .

$$Q \equiv \frac{\Delta f}{f_0} \quad (13.32)$$

Since the center frequency $f = 2\pi\omega_0$ and the bandwidth both scale inversely with τ , all the wavelets $\psi(\tau(t), t_0, t)$ have the same Q .

We can now build a transform based on these wavelets by replacing the infinite-duration sinusoids of the FT by finite-duration wavelets.

$$S(\tau, t_0) = \frac{1}{\sqrt{\tau}} \int s(t)\psi(\tau, t_0, t) dt \quad (13.33)$$

The essential difference between the constant Q wavelet transform and Fourier transform is depicted in Figure 13.9. The DFT divides the frequency axis into equal bandwidth bins, while the wavelet transform bins have constant Q and thus increase in bandwidth with increasing frequency. The center frequencies of the wavelet transform are equally spaced on a logarithmic frequency axis, compressive behavior much like that of our senses (see Section 11.2). While the STFT is matched to artificial signals engineered to equally partition the spectrum, the wavelet transform may be more suited to 'natural' signals such as speech.

Are there cases where the wavelet transform is obviously more appropriate than the STFT? Assume we need to analyze a signal composed of short pulses superposed on sinusoids of long duration. We need to measure both the frequencies of the sinusoids as well as the time durations of the pulses. The uncertainty theorem restricts our accurately measuring the frequency of the pulses, but not that of the steady sinusoids; but to use the STFT we are forced into making a choice. If we use long windows we can accurately measure the frequencies, but blur the pulse time information; if we use short windows we can note the appearances and disappearances of the pulses, but our frequency resolution has been degraded. Using the wavelet transform the

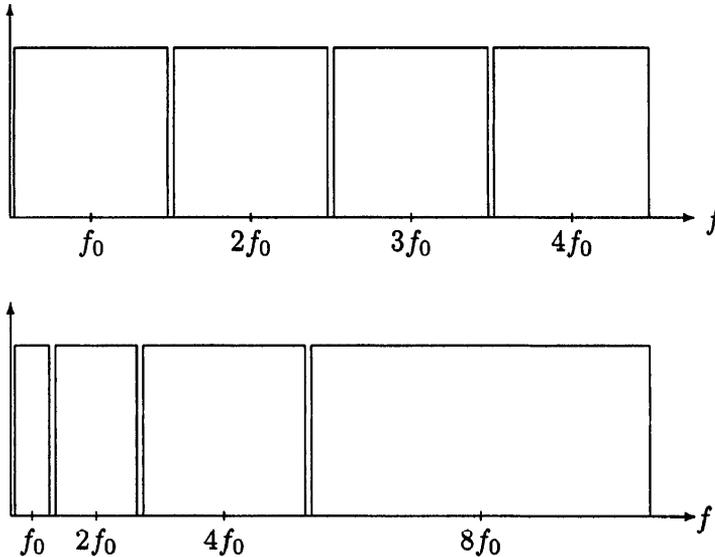


Figure 13.9: Comparison of Fourier and wavelet transforms. The top figure depicts a four bin DFT while the bottom is a four-bin wavelet transform. Note that the FT bins have constant bandwidth while those of the wavelet transform have constant Q . For the purposes of illustration we have taken the basic wavelet to be rectangular in the frequency domain.

time resolution gets better with higher frequency, while the frequency resolution becomes better at low frequencies (longer time durations). So using a single wavelet transform we can perform both measurements.

Digital wavelet transforms can be computed efficiently (a fast wavelet transform) using the pyramid algorithm, which extends the recursive computation of the logarithmic spectrum discussed in Section 13.2. We employ a pair of filters called **Quadrature Mirror Filters (QMFs)**. The QMF pair consists of a low-pass FIR filter that passes the lower half of the spectrum and a high-pass FIR filter that passes the upper half. The two filters are required to be mirror images of each other in the spectral domain, and in addition they must guarantee that the original signal may be recovered. The simplest QMF pair is $(\frac{1}{2}, \frac{1}{2})$ and $(\frac{1}{2}, -\frac{1}{2})$, the first being low-pass, the second high-pass, and their sum obviously the original signal. The pyramid algorithm works as follows. First we apply the QMF filters to the incoming signal, creating two new signals of half the original bandwidth. Since these signals are half bandwidth, we can decimate them by a factor of two with-

out loss of information. The decimated output of the high-pass is retained as the signal of the highest bin, and the QMF filters applied to lower band signal. Once again both signals are decimated and the higher one retained. The process is repeated, halving the bandwidth at each iteration, until all the desired outputs are obtained. The name ‘pyramid’ refers to the graph depicting the hierarchical relationship between the signals.

What is the computational complexity of the pyramid algorithm? If the QMF filters have M coefficients, the first iteration requires $2MN$ multiplications to produce N outputs. The second iteration operates on decimated signals and so requires only MN multiplications to produce outputs corresponding to the same time duration. Each iteration requires half the computations of the preceding, so even were we to compute an infinite number of iterations the number of multiplications would be $2 + 1 + \frac{1}{2} + \frac{1}{4} + \dots = 4$ times MN . So the wavelet transform is $O(N)$ (better than the FFT), and no complex operations are required!

EXERCISES

- 13.9.1 Use a raised cosine times a sine as a basic wavelet and draw the scaled wavelets for various τ . Compare these with the basis functions for the STFT.
- 13.9.2 A digital QMF pair obeys $|H^{[hp]}(f)| = |H^{[lp]}(\frac{f}{2} - f)|$, where $H^{[lp]}(f)$ and $H^{[hp]}(f)$ are the frequency responses of the low-pass and high-pass filters. Show that $h_n^{[hp]} = (-1)^n h_n^{[lp]}$ or $h_n^{[hp]} = (-1)^n h_{-n}^{[lp]}$ for odd length filters and similar statements for even length ones. Show that the latter form is consistent with the wavelets being an orthogonal basis for even length filters.
- 13.9.3 Can the original signal really be recovered after QMF filtering and *decimation* have been applied?
- 13.9.4 Derive an efficient procedure for computing the inverse digital wavelet transform.
- 13.9.5 Build a signal consisting of two close sinusoids that pulse on and off. Similarly build a signal that consists of a single sinusoid that appears as two close pulses. Try to simultaneously measure frequency and time phenomena using the STFT and the wavelet transform.
- 13.9.6 Compare the wavelet transform with the time-frequency distributions discussed in Section 4.6.

Bibliographical Notes

Kay and Marple have written a good tutorial of modern approaches to spectral analysis [128], and each has written a book as well [127, 123].

The periodogram was first introduced by Schuster [233] and became an even more indispensable tool after the introduction of the FFT. Blackman and Tukey wrote early book on the practical calculation of power spectra [19]. The bible of windows was written by Harris [93].

As noted in the bibliographical notes to Chapter 6, Yule [289] formulated the Yule-Walker equations for signals containing one or two sinusoidal components in the late 1920s, in an attempt to explain the 11-year periodicity of sunspot numbers. Walker, calling Yule's earlier work 'an important extension of our ideas regarding periodicity', expanded on this work, discovering that the autocorrelations were much smoother than the noisy signal itself, and suggesting using the 'correlation periodogram' as a substitute for the Schuster periodogram. He applied this technique to the analysis of air pressure data and could rule out as spurious various claimed periodicities.

Wiener was instrumental in explaining why the Schuster periodogram did not work well for noisy signals [276, 277], but this was not widely appreciated at the time. A highly interesting historical account is given by Robinson [223].

Pisarenko's original article is [196]. Various authors have analyzed the performance of the PHD [227, 255, 285].

Officer of the U.S. Navy and Harvard mathematician Joseph Leonard Walsh presented his functions in 1923 [268]. The standard text is [12] and a short introduction can be found in [13]. The conversion between DWT and DFT was expounded in [256, 257]. Hartley, who was in charge of telephone research at Bell Labs and responsible for an early analog oscillator, presented his transform in 1942 [94]. The DHT and FHT were published in [24, 25]. The standard text is [27].

Wavelets already have a rich literature. For DSP purposes we recommend the review article by Rioul and Vetterli [221].