DISCREETLY FINE TIMES
with
DISCRETE FOURIER TRANSFORMS
(DFT's with DFT's)
or
"WHY DOES THAT FFT OUTPUT LOOK SO WEIRD???"

"Spectral analysis" means describing a time-varying signal by its power spectrum, otherwise known as its Fourier spectrum or its frequency spectrum. It should be obvious that such spectra are essential for astronomers: for example, spectra tell us not only about pulsar periods, but also about chemical composition in stars and Doppler velocities. But the description of signals in terms of spectra or Fourier components has importance in many fields, and this becomes obvious on just a little reflection. A prime example: a bridge is subject to shaking by earthquakes; if its natural resonance frequency is close to the frequency of ground movement, you've got trouble!

Spectral analysis is an integral part of some of our labs. We will sample signals at regular intervals with our computers, a process known as "discrete sampling", and our computers will change the voltages to digital numbers, a process known as "digitization". We'll then do a discrete Fourier transform to calculate the power spectrum. There are some intricacies with this process…

Contents

0. THE INTEGRAL FOURIER TRANSFORM: TIME $\leftrightarrow$ FREQUENCY

0.1. Classical Fourier transforms

We use the Fourier transform (FT) to convert a time-varying voltage $E(t)$ from the time to the frequency domain $E(\nu)$, and *vice versa*. The formally correct equations are

$$E(\nu) = \int_{-\infty}^{\infty} E(t) e^{[2\pi i]\nu t} dt \tag{0.1a}$$

and, to go the other way,

$$E(t) = \int_{-\infty}^{\infty} E(\nu) e^{-[2\pi i]\nu t} d\nu \tag{0.1b}$$

In our case, we are going to modify this a bit so that we don't run into infinities. We will write. . .

$$E(\nu) = \lim_{T\rightarrow\infty} \frac{1}{2T} \int_{-T}^{+T} E(t) e^{[2\pi i]\nu t} dt \; . \tag{0.2}$$

We are often interested in the *power spectrum.* Power goes as amplitude squared, so the power spectrum $P(\nu) \propto E(\nu)^2$. A small complication: $E(\nu)$ can be complex, but the power spectrum is just watts per Hz and must be real. We take care of this by defining

$$P(\nu) = E(\nu) \times [E(\nu)]^* \; , \tag{0.3}$$

where the superscripted $[E(\nu)]^*$ means the complex conjugate. This multiplication is the equivalent of "detection".

1. EFFECTS OF T BEING FINITE: INTEGRAL TRANSFORMS

There's a fundamental fact of the real world: we don't live forever, and even if we did TAC's wouldn't allow us infinite telescope time for our precious projects, so we never can allow $T \rightarrow \infty$. This frustrates the old politicians who want absolute control over everything, forever. Less seriously in some circles—and more in others—it has two major ramifications for Fourier transform devotees:

1. It limits the frequency resolution.
2. It produces "spectral leakage": The result at a given frequency in the Fourier transform spectrum is contaminated by those at *all other* frequencies in the spectrum!

We treat these two ramifications in the following subsections.

1.1. Spectral Resolution

Equations 0.1 and 0.2 are fine as they stand, but when we sample a signal T is finite. This limits our spectral resolution. To understand this, consider the analytic solution of equation 0.2 for a monochromatic cosine wave of frequency ν_s, that is $E(t) = \cos(2\pi\nu_s t)$. For $T \rightarrow \infty$, $E(\nu) = 0$ except for $\nu = \nu_s$; this is commonly expressed by the term "Dirac delta function". But more generally, we have as the analytic solution

$$E(\nu) = \frac{\sin(2\pi(\nu - \nu_s)T)}{2\pi(\nu - \nu_s)T} \; . \tag{1.1}$$

It is simpler to write $\delta\nu = (\nu - \nu_s)$ and write

$$E(\nu) = \frac{\sin(2\pi\delta\nu T)}{2\pi\delta\nu T} \; . \tag{1.2}$$

$$P(\nu) = \left(\frac{\sin(2\pi\delta\nu T)}{2\pi\delta\nu T}\right)^2 \; . \tag{1.3}$$

and we see that this is just the famous $\frac{\sin x}{x}$ type behavior. This quantitatively expresses the degradation in frequency resolution when you sample for a finite time; roughly, the spectral resolution is the width of this function, which is $\Delta\nu \sim \frac{1}{T}$.

In optical and IR astronomy, the need for high spectral resolution leads to the requirement of long time delays, which in turn means long path lengths. For grating and prism spectrometers, this can be achieved only by using physically large devices. For a Fabry-Perot, the long path is achieved by multiple reflections. In terms of the conventional definition of resolving power, $\frac{\Delta\lambda}{\lambda} \sim \frac{\lambda}{size}$. There's no way around this!

1.2. Spectral Leakage

Not only is the infinitesimally-narrow delta function replaced by a function of width $\sim \frac{1}{2T}$ but also the function $\frac{\sin x}{x}$ has "sidelobes": it spreads power out over a huge range of frequencies, albeit—after squaring, anyway—at a low level. The spread-out power gets weaker as you move away from the signal, that is as $\delta\nu$ increases—but it doesn't decrease monotonically. Rather, the sidelobes mean that there the spread-out power goes to zero periodically for $\delta\nu = \frac{1}{2T}$. In colloquial terms, the power of the monochromatic sine wave "leaks" away in the power spectrum to all frequencies. Not surprisingly, this is called "spectral leakage". It's covered in NM chapter 13.4.

You can reduce spectral leakage by using a "window function" $W(t)$ that is not "square". What on Earth does this mean? In equation 0.2, when you use a finite T you are integrating over a "window" of total length $2T$, as follows:

$$E(\nu) = \frac{1}{2T} \int_{-T}^{+T} W(t) E(t) e^{[2\pi i]\nu t} dt \ . \tag{1.4}$$

In equation 0.2, the window function $W(t)$ is unity; when we let $W(t) = 1$ for the time window of length $2T$, then this is called a "square window function". A square window function produces "sharp edges" in the sense that the function $E(t)$ has its full amplitude for all t within the window and then cuts off precipitously at the ends of the window. *It's this sudden cutoff—the sharp edge—that produces spectral leakage.*

We can reduce spectral leakage by making $W(t)$ a function that produces a *gradual* cutoff so as to eliminate the sharp edge. We make $W(t)$ a function that gradually falls to zero at the window edges $-T$ and T. As one possible example among many, consider the Hanning weighting function, which is commonly used in radio astronomy:

$$W(t) = \frac{1}{2} \left(1 + \cos \frac{\pi t}{T} \right) \ . \tag{1.5}$$

This drastically reduces the sidelobes. This comes at a price, though: the spectral resolution gets worse—by about a factor of two. The reason is that the window function $W(t)$ goes to zero at the ends of the time window $2T$, which effectively makes the window shorter. *Note:* We also use the term "weighting function" for $W(t)$.

2. DISCRETELY SAMPLED SIGNALS

When I was a teenager we had vinyl records. They contained *continuously-sampled* signals that were directly imprinted on the vinyl. Today we focus on CD's. These contain *discretely-sampled* signals that are digitized and written on to the little silver disk. The fact that CD's contain discretely sampled data means that the signal is known only for certain specific times. In particular, the signal is not sampled *between* these times. Does this mean that CD's don't sound as good as the vinyl records?

Regarding the calculation of our power spectrum with a FT, you'd think that the replacement of an integral by a sum would be straightforward, with no subtleties. But this *isn't the case.* There *are* subtleties, and they are *crucial.* Understanding them, and dealing with them properly, makes your CD's sound just as good as vinyl. So read on: we'll first provide a qualitative introduction and then go into details.

2.1. Discretely-Sampled Signals: The Nyquist Criterion.

When your CD music is recorded, or in astronomy when we observe a pulsar or sample an incoming signal to determine the spectrum, we sample the incoming signal sequentially in time, at regular intervals called the *sampling interval* $\Delta t = t_{smpl}$. Equivalently, we are sampling at a regular rate called the sampling frequency $\nu_{smpl} = \frac{1}{t_{smpl}}$.

In discrete sampling, we first require that the signal be limited in bandwidth. That is, the highest frequency in its spectrum must be limited to an upper cutoff, which we call the bandwidth B; thus, the frequencies in the signal extend from 0 to B Hz. Then we must sample the signal periodically at a fast enough rate—specifically, we require ($\nu_{smpl} \geq 2B$ Hz). This is the "Nyquist criterion". If the Nyquist criterion is violated, we have the problem of *aliasing*, which means that signals with frequencies $\nu > B$ *appear* as *lower* frequencies. Remember those movies showing cars and the wheels that seem to rotate backwards? That's a real-world example of aliasing: the movie frames discretely sample the scene at too slow a rate to faithfully reproduce the wheel's rotation. You can understand aliasing by looking at Figure 2.1.

To put it another way: When we sample at rate ν_{smpl}, the maximum signal frequency that can be faithfully reproduced is $\frac{\nu_{smpl}}{2}$. This is called the Nyquist frequency and here we denote this by $f_N = \frac{\nu_{smpl}}{2}$. Clearly, the signal's bandwidth B must satisfy $B \leq f_N$. All this is called the *sampling theorem.*

How rapidly must music be sampled for recording on a CD? The human ear responds to something like 20 kHz. To prevent aliasing, we need to sample at twice this, at about 40 kHz.

2.2. A Popular MISCONCEPTION!!!

Somebody, sometime, will probably tell you that aliasing has to do with *discrete Fourier transforms.* Tell them to go jump in the lake. It has to do with the sampling rate. Period. Aliasing occurs if you don't sample fast enough, and affects your results *whether or not* you are dealing with Fourier transforms. If you don't believe me, just look again at Figure 2.1. And remember what

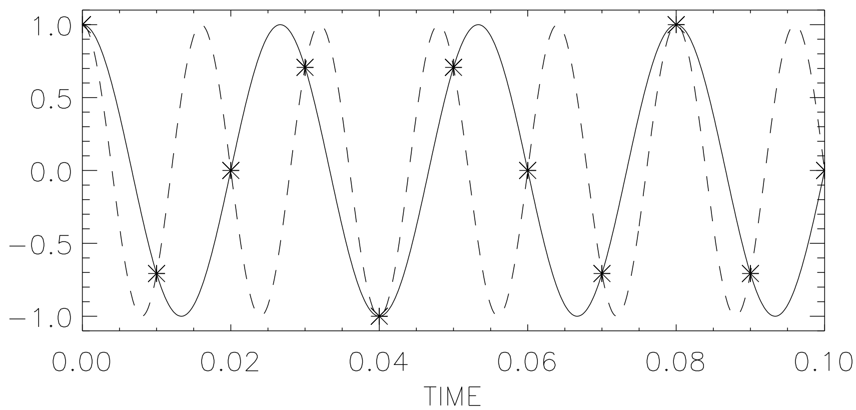

Fig. 2.1.— Aliasing at its worst. The sample interval is $t_{smpl} = 0.01$ s, so the sample frequency is $\nu_{smpl} = 100$ Hz. The Nyquist frequency is $f_N = 50$ Hz. The stars are the datapoints. The two signals shown are at frequencies 37.5 and 62.5, i.e. $f_N \pm 12.5$ Hz. Which signal do the datapoints represent?

happens to rotating wheels in movies.

3. DISCRETE SAMPLING AND THE FOURIER TRANSFORM

We use the Fourier transform to convert from the time to the frequency domain, and *vice versa*. For continuously-sampled signals we use the Fourier integral, equation 0.2. For discretely-sampled signals we have to replace this by a summation. This is the *discrete Fourier transform*, or DFT.

3.1. The maximum recoverable bandwidth: the Nyquist frequency

OK, we've sampled our signal at regular intervals t_{smpl}, meaning that the sampling frequency is $\nu_{smpl} = \frac{1}{t_{smpl}}$ and $f_N = \frac{\nu_{smpl}}{2}$. If we do this for time $2T$ then we obtain $2J = \frac{2T}{t_{smpl}}$ independent points. From these we wish to compute the spectrum.

If we want to digitally compute the Fourier transform, then the first thing we must realize that the original $2J$ channels provide only J spectral points! This seems like a loss of information, but it's not: the spectrum is *complex*, so those J complex numbers contain $2J$ independent numbers.

With sample frequency ν_{smpl} we obtain the spectrum for the range of frequencies $0 \rightarrow \frac{\nu_{smpl}}{2}$ or $0 \rightarrow f_N$; this is known as the *bandwidth*, denoted by B. With J independent points, the separation (and the frequency resolution) is $\Delta\nu = \frac{\nu_{smpl}}{2J}$. Fortunately, $\frac{\nu_{smpl}}{2J} = \frac{1}{2Jt_{smpl}} = \frac{1}{2T}$; this confirms our earlier discussion in §1.1 about frequency resolution.

But more importantly for the present discussion, when sampling at rate ν_{smpl} you cannot recover a spectrum wider in bandwidth than $f_N = \frac{\nu_{smpl}}{2}$.

Amazingly, a discretely sampled time series occurs where you'd least expect it: in a common, garden-variety optical analog device, namely the classical optical grating spectrometer. The incoming light strikes the grating at angle θ away from the normal and for the first order reflection leaves at the same angle. Thus the difference in time delay from one groove to the next is $\frac{2s\cos\theta}{c}$, where s is the groove separation. You don't have any information for time differences between these values. So this is exactly equivalent to t_{smpl}, so the maximum fractional bandwidth is $\frac{f_N}{f} = \frac{1}{4\frac{s}{\lambda}\cos\theta}$. To attain a substantial fractional bandwidth, e.g. $\frac{f_N}{f} > 1/2$ say, we require the spacing $s < \lambda/2\cos\theta$. Moreover, to avoid aliasing it's absolutely necessary to insert a filter in front of the grating to limit the input bandwidth.

3.2. Summary: The Two Fundamental Parameters in Discrete Sampling and Fourier Transforms

The above two parameters are the fundamental ones.

(1) To prevent aliasing, we must satisfy the sampling theorem: the total signal bandwidth B must be small enough , so $B \leq f_N$, or $B \leq \frac{\nu_{smpl}}{2}$.[1] Usually you need to limit the bandwidth with a filter.

(2) In the Fourier-transformed power spectrum, the spectral resolution is the reciprocal of the total time over which the FT is computed: $\Delta\nu = \frac{1}{T_{tot}}$.

4. THE DISCRETE FOURIER TRANSFORM AND DISCRETE SAMPLING

With the DFT, we have to replace the Fourier integral in equation 0.2 by a summation. Let's do this with a one-to-one correspondence of the terms. First we rewrite equation 0.2 to make it easier for a direct comparison:

$$E(\nu) = \frac{1}{2T}\int_{-T}^{+T} E(t)e^{[2\pi i]\nu t}dt \; . \tag{4.1}$$

In our discretely-sampled case we can replace t by jt_{smpl}, defined for $j = -J \rightarrow J$; ν by $\frac{k\nu_{smpl}}{2J}$;

[1]For complex input, which can occur in radio astronomy and some other applications, the discussion becomes more interesting.

and dt by t_{smpl}. We would calculate $E(\frac{k\nu_{smpl}}{2J})$ for $k = -J \rightarrow J$, so the summation looks like...

$$E(\frac{k\nu_{smpl}}{2J}) = \frac{1}{2Jt_{smpl}} \sum_{j=-J}^{J-1} E(jt_{smpl}) e^{[2\pi i]\nu_{smpl} t_{smpl} \frac{jk}{2J}} t_{smpl} \ . \tag{4.2}$$

Here we've taken $dt = \Delta t = t_{smpl}\Delta j = t_{smpl}$ (i.e., $\Delta j = 1$). The product $t_{smpl}\nu_{smpl} = 1$, so we can simplify our notation by eliminating these variables, replacing $\frac{k\nu_{smpl}}{2J}$ by the much simpler k, and replacing jt_{smpl} by just j and writing...

$$E(k) = \frac{1}{2J} \sum_{j=-J}^{J-1} E(j) e^{[2\pi i]\frac{kj}{2J}} \ . \tag{4.3}$$

4.1. IMPORTANT DETAIL Regarding Limits of Summation and Periodicities

Are you paying attention? If so, you should be asking why the upper limit in equation 4.3 on the sum is $J-1$ instead of J.

One reason is that a sum from $-J \rightarrow J$ has $(2J+1)$ samples. But we have only $2J$ samples. So we can't possibly sum from $-J \rightarrow J$.

This doesn't matter at all. To begin with, look at equation 4.3 carefully: you can verify for yourself that the trig portion—that is, the $e^{2[\pi i]\frac{kj}{2J}}$ term—is *periodic* in both j and k, with period $2J$. That is, you can use either k or $(k+2J)$—you'll get the same answer. Same with j.[2] And one other thing: the discrete Fourier transform makes the implicit, intrinsic, completely irrevocable *assumption* that the input signal $E(j)$ is *also* periodic with period $2J$. Thus the *entire quantity being summed* is periodic with period $2J$. This means, also, that the *result* of the summation, namely $E(k)$, is *also* periodic with period $2J$. The intrinsic, irrevocable assumption that $E(j)$ is periodic means that we only need to know $2J$ values of $E(j)$; the $j = 2J$ value is equal to the $j = 0$ value.

Now, everybody knows[3] that when we replace an integral by a digital summation we regard each sample as centered on a bin of width *unity* $\times \Delta t$. However, the samples at the end, with $j = -J$ and $j = J$, must have bins of width $\frac{1}{2} \times \Delta t$, so we're supposed to include both $E(-J)$ and $E(J)$ in the sum, each with a weight of $\frac{1}{2}$. But because of the $2J$ periodicity, we can instead include just one and use a weight of unity.

[2] With the qualification that k is an integer, i.e. that you are calculating the result only for integral multiples of $\frac{\nu_{smpl}}{2J}$. We discuss this in more detail below.

[3] If you don't believe me, go look at your elementary calculus book, in the chapter where integration was introduced.

The fact that $E(j)$ is periodic with period $2J$ allows equation 4.3 to be written in its more conventional form

$$E(k) = \frac{1}{2J} \sum_{j=0}^{2J-1} E(j) e^{[2\pi i] \frac{kj}{2J}} \text{ or } E(k) = \frac{1}{N} \sum_{n=0}^{N-1} E(n) e^{[2\pi i] \frac{kn}{N}} \ . \tag{4.4}$$

However, this form is valid only for integral values of k; see §6.2.

4.2. Again: The Periodic Nature of the Discrete FT

Above we discussed the periodic nature of both $E(j)$ and $E(k)$. Let's delve deeper.

By its very nature, the Fourier transform wants to integrate to ∞. But the input signal doesn't go on forever! We have to resolve this basic incompatibility: the input signal is sampled for a finite time, but the Fourier transform needs it to go on forever.

There's only one way to resolve this incompatibility: we must assume that the input signal *does* go on forever, and because we don't know what happens outside the sampling interval $-J \rightarrow J$, the only sensible thing to do is to assume that the input signal is *periodic* with period $2J$.[4]

This assumption of "forever", together with the associated $2J$ periodicity, leads to the necessary *result* that the spectrum $E(k)$ and its power spectrum $P(k) = E(k) \times [E(k)]^*$ *also* go on forever, $k = -\infty \rightarrow +\infty$, and are periodic with period $2J$.

Because of these periodicities, we gain complete information on the spectrum by restricting our attention to *windows* of length $2J$: these windows are the finite intervals in (j, k) between the dotted lines in figure 5.1. They can begin and end anywhere—all that matters is that their length is $2J$.

5. THAT WEIRD-LOOKING POWER SPECTRUM—IT'S JUST A MATTER OF STANDARD CONVENTION

Above we let the indices j and k run from $-J \rightarrow J - 1$. But in the real world of numerical computing, we don't use negative indices. You might think that the reasonable way to handle this would be to displace the whole set of $2J$ values of $E(j)$ and $E(k)$ upwards by J so that the indices run from $0 \rightarrow 2J - 1$ (this would be perfectly compatible with IDL's indexing scheme). But this

[4]You might be surprised: why not assume the signal is *zero* outside the interval? *Answer*: most signals don't stop simply because we stop sampling them—for example, a pulsar. *More Fundamental Answer*: the math requires this assumption!

would put the $t = 0$ or $f = 0$ point out in the middle, at j or $k = J$, and this isn't very convenient for lots of reasons.

Instead, realize that the FT is periodic in j with period $2J$. Therefore, in equation 4.2 it doesn't matter whether you sum from $j = -J \rightarrow J - 1$, from $j = 0 \rightarrow 2J - 1$, or even (god forbid!) from $j = -3J + 7 \rightarrow -J + 6$. So we might as well just sum from $j = 0 \rightarrow 2J - 1$ and not displace anything at all. *This is the standard convention, and it leads to the standard way in which FT arrays are stored in memory*—not just in IDL but in just about *every* software package. It has the great advantage that the $t = 0$ or $f = 0$ point is the first one in the array.

Above we were discussing "the FT array", without specifying whether it was the input or output array. The arrangement for the *input array* works in exactly the same way as that for the *output* array. And it doesn't matter whether the independent variable for the array is time or frequency. *All FT arrays, whether input or output and no matter what the independent variable, are arranged identically with respect to the negative and positive values.*

5.1. Enough Preaching! Let's Try an Example in IDL

Here we take an example with $J = 32$, with the discretely-sampled input signal being the sum of three cosine waves: $E(j) = \cos(\pi \frac{8j}{32}) + 0.5[\cos(\pi \frac{7j}{32}) + \cos(\pi \frac{9j}{32})]$. Thus we have power at three frequencies: $k = 7, 8, 9$. There is twice as much voltage, thus four times the power at $k = 8$ than there is at $k = 7, 9$. The power spectrum consists of these three nonzero points, plus a bunch of zeros at all other frequencies.

In IDL, we denote $E(j)$ by **EJ** and generate this 64-element array with the statements...

$$\mathbf{EJ} = \mathbf{cos}(\mathbf{!pi} * \mathbf{8} * \mathbf{findgen}(\mathbf{64})/\mathbf{32}) \tag{5.1a}$$

$$\mathbf{EJ} = \mathbf{EJ} + \mathbf{0.5} * \mathbf{cos}(\mathbf{!pi} * \mathbf{7} * \mathbf{findgen}(\mathbf{64})/\mathbf{32}) + \mathbf{cos}(\mathbf{!pi} * \mathbf{9} * \mathbf{findgen}(\mathbf{64})/\mathbf{32})) \tag{5.1b}$$

Figure 5.1 (top) shows the plot of the 64-element array **EJ**, with the 64 crosses marking the computed points (or, in our parlance, the discretely-sampled points). The interference of the sine waves makes the sampled signal look like a wave packet of frequency $k = 8$, attenuated at the middle of the time interval where $j \sim 32$.

The signal $E(j)$ must be periodic with period $2J = 64$ and it must go on forever. Figure 5.1 (bottom) shows this for a larger slice of time in which $j = -128 \rightarrow 128$. Our points are computed for IDL indices $0 \rightarrow 63$; this window is shown by the dotted lines on Figure 5.1 (bottom).

We take the Fourier transform $[E(k) = FT(E(j))]$. In IDL, we denote $E(k)$ by **EK** and use IDL's Fast Fourier Transform (FFT) procedure...

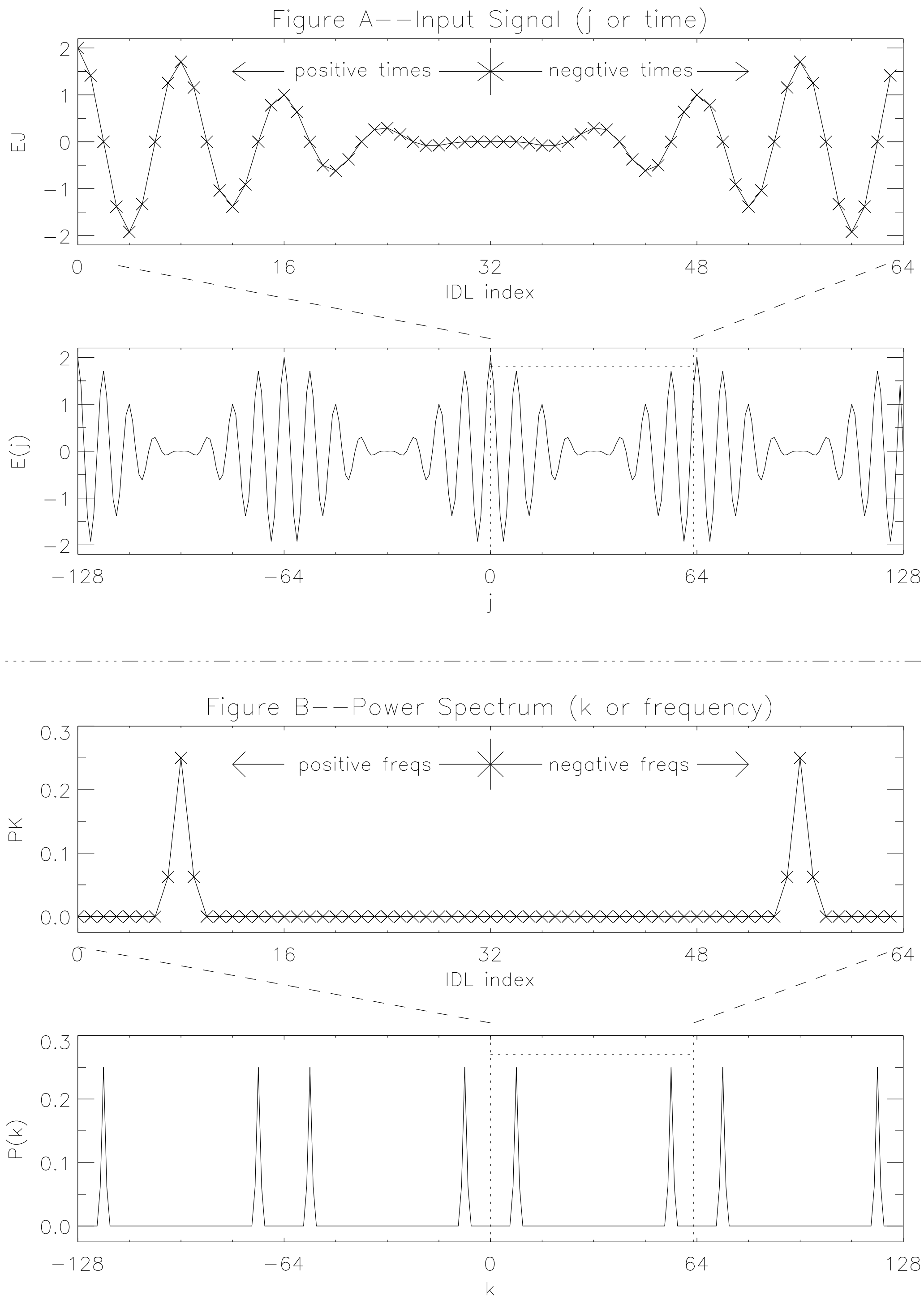

Fig. 5.1.— Upper plots in Figs A and B show a 64-point time series and power spectrum. Lower plots show a portion of an infinitely-long periodic time series and power spectrum from which the 64-point ones are extracted.

$$\mathbf{EK} = \mathbf{fft}(\mathbf{EJ}) \tag{5.2a}$$

and get then the power spectrum $P(k)$ (**PK** in IDL)...

$$\mathbf{PK} = \mathbf{float}(\mathbf{EK} * \mathbf{conj}(\mathbf{EK})) \tag{5.2b}$$

For convenience, we use the **float** to discard the imaginary portion of the product, which is zero—this makes **PK** real, so that in later operations we don't have to deal with a complex array (e.g., to plot the power spectrum we can type **plot, PK** instead of **plot, float(PK)**).

Figure 6.1 (top) shows the plot of the 64-element array **PK**, with the 64 crosses marking the computed points returned by IDL. The *positive* frequencies lie in the IDL index range $1 \rightarrow 32$ and the *negative* ones in the range $32 \rightarrow 63$. As expected, there is nonzero power at only three positive frequencies, the ones with IDL indices 7, 8, 9. There is also power at the three corresponding negative frequencies, the ones with IDL indices 55, 56, 57.

The power spectrum must be periodic with period $2J = 64$ and it must go on forever. Figure 6.1 (bottom) shows this for a larger slice of frequency in which $k = -128 \rightarrow 128$. IDL indices for **PK** are $0 \rightarrow 63$; this window is shown by the dotted lines on Figure 6.1 (bottom).

5.2. Is this weirdness perfectly clear?

Probably not. So, in the following two subsections we go through this again in excruciating detail. We provide first a detailed verbal description of the arrangement, and then the shortest of short numerical examples. In the verbal description we focus on the output array to make things specific, but we just as well could have focused on the input array and replaced the word "frequency" by "time". As you'll see, this leads to something surprising... we'll discuss it explicitly (5.2.3).

5.2.1. Verbal Description

Again, we let the time for $E(t)$ run from $t = -T \rightarrow +T$, with $T = Jt_{smpl}$. The corresponding frequencies run from $f = -f_N \rightarrow +f_N$, with $f_N = \frac{\nu_{smpl}}{2}$.

All FT arrays are arranged so that the *first* $J+1$ channels contain the *positive-frequency* portion of the transform. Again, here we focus on the output array. Thus, for channel $k = 0 \rightarrow J$ the frequency of channel k is $\nu_k = +\frac{kf_N}{J}$. Thus, channel 0 contains the $\nu = 0$ result, channel 1 the $\nu = +\frac{f_N}{J}$ result, ..., up to channel J which contains the maximum frequency $\nu = +f_N$.

The remaining $J-1$ channels contain the *negative-frequency* portion of the transform. As we go from channel J to $J+1$ we would go to the next highest *positive*-frequency point; but because the FT is periodic with period $2J$, this must be identical to the corresponding point at *negative* frequency. This means that channel J contains *not only* the result for $\nu = +f_N$, but *also* the result for $\nu = -f_N$. And the remaining $J-1$ higher channels contain the rest of the *negative*-frequency points, so for $k = J \rightarrow (2J-1)$ the frequency is $\nu_k = -f_N + \frac{(k-J)f_N}{J}$. Thus channel J has $\nu = -f_N$, channel $(J+1)$ has $\nu = -f_N + \frac{f_N}{J}$, $\ldots$, and channel $(2J-1)$ has the $\nu = -\frac{f_N}{J}$ result. If you were to consider channel number $2J$ it would contain the next highest frequency, which is $\nu = 0$; of course, this is identical to the result in channel 0.

Note that frequency can *always be considered to increase with channel number*: you can even regard the big backwards jump from $+f_N$ to $-f_N$ at channel J as *not* being a jump because the periodic nature of the transform means that the results for these two frequencies must be identical, and successively higher negative frequencies are equivalent to successively higher positive frequencies above $+f_N$.

So any FT array, for example the spectrum, contains $2J$ independent points. More generally, the FT spectrum could contain $4J$ points, or $6J$ points, etc. We could calculate as many points as we wish. However, because the FT is periodic, with period $2J$, channel $2J+1$ would contain the same result as in channel 1, etc. So there is no sense in calculating more than $2J$ points, because the calculations would be redundant.

5.2.2. An Example

Consider a simple case with $J = 4$; you've taken 8 time samples. Then the proper way to arrange the input to the FT is the following, where the *left-hand matrix contains the IDL indices* and the *right hand the times or frequencies*:

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 \times \frac{T}{J} \\ 1 \times \frac{T}{J} \\ 2 \times \frac{T}{J} \\ 3 \times \frac{T}{J} \\ \pm 4 \times \frac{T}{J} \\ -3 \times \frac{T}{J} \\ -2 \times \frac{T}{J} \\ -1 \times \frac{T}{J} \end{bmatrix} \tag{5.3a}$$

and the output looks like

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix} \leftrightarrow \begin{bmatrix} 0 \times \frac{f_N}{J} \\ 1 \times \frac{f_N}{J} \\ 2 \times \frac{f_N}{J} \\ 3 \times \frac{f_N}{J} \\ \pm 4 \times \frac{f_N}{J} \\ -3 \times \frac{f_N}{J} \\ -2 \times \frac{f_N}{J} \\ -1 \times \frac{f_N}{J} \end{bmatrix} \tag{5.3b}$$

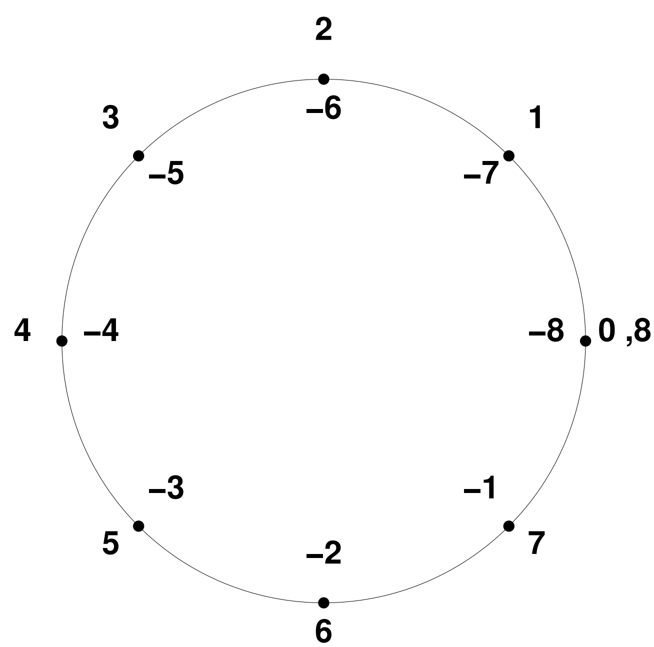

Fig. 5.2.— Envisioning the window as a circle makes the relationship between negative and positive frequencies or times. The numbers are the multipliers of the times and frequencies in equations 5.3.

We have been envisioning the FT array in linear space. However, the DFT is composed of trig functions restricted to a given range in frequency and time. It's helpful to step back and change our view from linear space to *circular* space—to get back to fundamentals and recall that trig functions are defined on the *circle*. As pictured explicitly in Figure 5.2, negative and positive frequencies, and repeated windows, simply correspond to going around the circle more than once, or in the opposite direction.

5.2.3. *Wait a Minute! You Mean the TIME Isn't CONTINUOUS?*

Above we said—and we meant it—that in our verbal description we focused on *frequency*, but we could just as well have focused on *time*. And in equation 5.3 above, where we explicitly listed the time versus the IDL index, the time *begins* at 0 for IDL index 0, runs up to $3 \times \frac{T}{J}$ for IDL index 3, and then "wraps around" to *negative* times for IDL index ≥ 4. So it seems that time versus IDL index is discontinuous—just like frequency.

Yes it's true!! If you take a bunch of time samples and compute the FT, you should put the negative time samples into the upper index ranges.

But for the calculation of just the power spectrum, it doesn't matter whether you bother with this reshuffling or not—you get the same answer whether or not you reshuffle. What it *does* matter for is the cases for which you are really interested in the *phase* of the FT output array. Remember that the output array $E(k)$ in equation 4.2 is complex; the ratio of imaginary to real parts gives the phase of the signal. This phase is defined with respect to $t = 0$. If you want the phase to be correctly defined, and if you want to regard the samples as extending from $-T \rightarrow T$ instead of $0 \rightarrow 2T$, then you must reshuffle the discrete time samples according to the above prescription.

Of course, the power spectrum doesn't have any phase information: it's only a specification of the power versus frequency. In other words, detected signals have no phase information. In our lab work we generally don't care about the absolute phase of the signal, so you have no need to carry out the index reshuffling before computing the FT.

6. THE SPECTRUM AT AN ARBITRARY FREQUENCY

Equation 4.3 provides results for discretely-spaced frequencies at intervals $\Delta\nu = \frac{\nu_{smpl}}{2J}$. It's often nice to have results for arbitrary frequencies. One way is the brute-force approach of equation 4.2, which we repeat here in slightly modified form:

$$E(\frac{\kappa\nu_{smpl}}{2J}) = \frac{1}{2J} \sum_{j=-J}^{J-1} E(j) e^{[2\pi i] \frac{j\kappa}{2J}} . \tag{6.1}$$

Here we have eliminated t_{smpl} and ν_{smpl} on the right hand side and replaced k by κ to emphasize the fact that κ can be a non-integer. (For non-integral k, you *must* carry the sum from $(-J \rightarrow J-1)$ and not $(0 \rightarrow 2J-1)$; see §6.2).

This is fine as long as you don't want to compute a lot of frequencies. However, suppose you want to create four interpolated points per original point so that you have a good visual representation of the spectrum. Each point requires $\sim 2J$ operations, so this can be a lot of computing.

Instead of calculating the points from the whole time series, you can interpolate using the spectral points themselves. In principle, you need to include *all* points when computing these interpolations. For uniform weighting, the proper interpolation formula in the frequency domain is (Brault & White)

$$E(\nu) = \frac{1}{2J} \sum_{j=0}^{2J-1} E(j) \frac{\sin[\pi(\nu - \nu_j)(2Jt_{smpl})]}{\tan[\pi(\nu - \nu_j)t_{smpl}]} \tag{6.2}$$

Note that $(2Jt_{smpl}) = 2T =$ the total time; many texts use T as the total time.

If you are willing to live with approximate results, then you can save a lot of computer time by including just a few original data points on each side of the point you're calculating because most of the contribution comes from those nearby points. For example, a good approximation is

$$E(\nu) = \sum_{j_{nearby}} E(j) \frac{\sin[\pi(\nu - \nu_j)(2T)]}{\pi(\nu - \nu_j)(2T)} \tag{6.3}$$

where j_{nearby} is a set of nearby frequencies in the original spectrum chosen as a compromise between accuracy and speed. This equation makes sense: you are weighting the spectral points in proportion to their sidelobe response in equation 1.1. Restricting j_{nearby} to the nearest few spectral channels to ν is sufficient for many purposes.

Note: Strange as it may seem, the denominators in equations 6.2 and 6.3 are written correctly.

6.1. Two examples

We consider two examples. Both have 64 datapoints; note that $64 = 2^6$.

6.1.1. The first example: a spike centered on integral k

First, the example of $[E(j) = \cos(2\pi \frac{\nu_{smpl}}{8} t)]$. This is a monochromatic wave whose frequency is exactly one-quarter the Nyquist frequency, i.e. when plotted it is exactly one-quarter the way between 0 and f_N. It falls *exactly* on an integral value of k. We show the resulting $E(\frac{k\nu_{smpl}}{2J})$ by the stars in Figure 6.1 (top), and we show the almost-continuous curve from fractional values of κ in equation 6.1 as the solid curve. The stars are nonzero *only* for one single value of k. This is because the signal frequency is *exactly centered* on the frequency represented by this value of k.

But look at the solid curve. You might have thought that this would be *symmetric* about the signal frequency; this is what the analytic equation 1.1 seems to suggest. *But it's not symmetric.* The reason is that, because the signal is real, the spectrum is Hermetian. Hermetian means that the

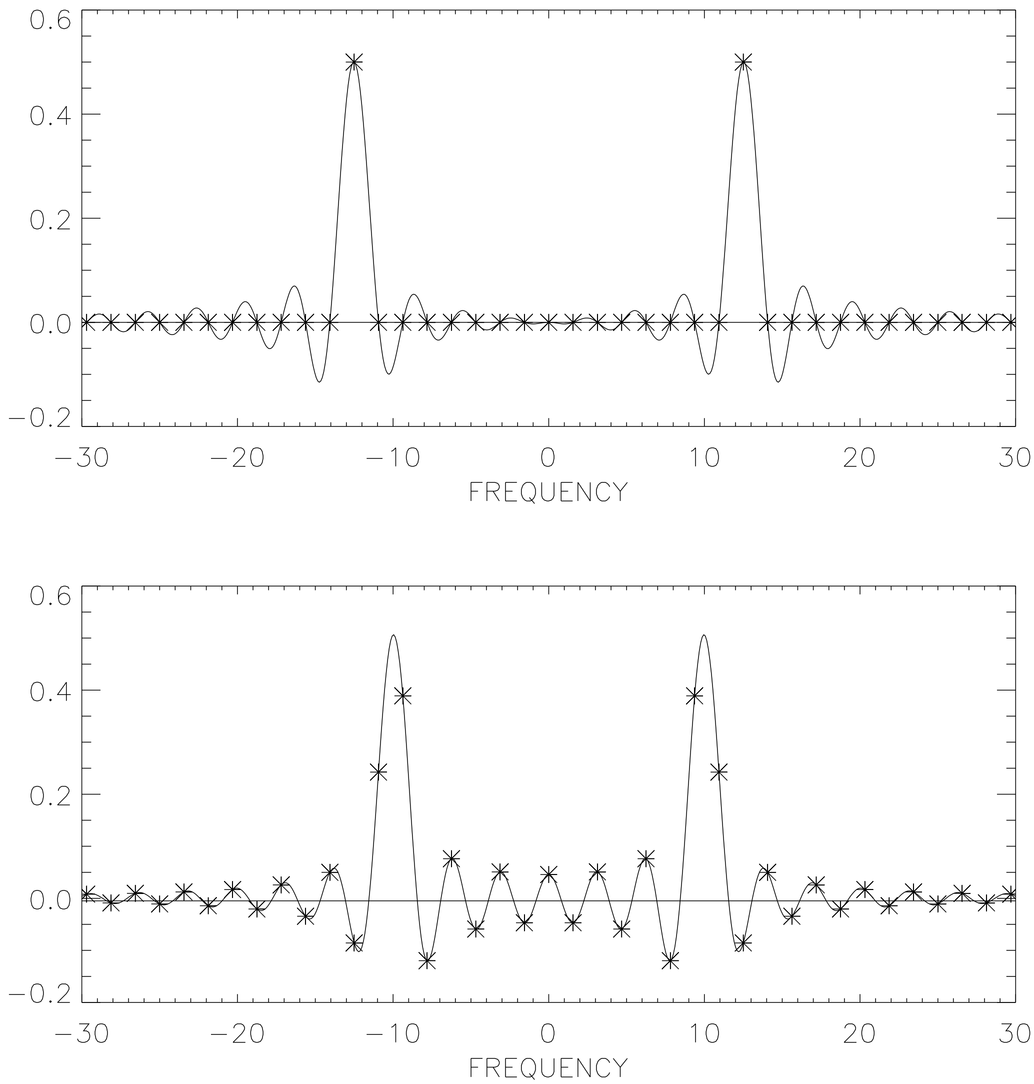

Fig. 6.1.— Monochromatic signals with frequency 12.5 Hz (top) and 10.0 Hz (bottom). The sample interval is $t_{smpl} = 0.01$ s, so the sample frequency is $\nu_{smpl} = 100$ Hz and the Nyquist frequency is $f_N = 50$ Hz. Stars are from the FFT, located at frequencies $(-f_N + k\Delta\nu)$ $[\Delta\nu = \frac{1}{2T}]$; the solid curve joins much more closely spaced points calculated using equation 6.1.

negative frequency real part is equal to the positive frequency real part. (In this case, the signal is symmetric in time, so the imaginary components are zero.) The sidelobes of the negative frequency part interfere with those of the positive frequency part, which causes the asymmetry. If you include the negative frequencies in the analytic calculation, then you also find that the sidelobes are not symmetric.

Similar asymmetries in sidelobe structure occur in the high end of the spectrum because of aliasing: The sidelobes of the $\frac{\sin x}{x}$ function go on forever, so they exist at frequencies *beyond the Nyquist frequency.* They are aliased back into the displayed spectrum and this also produces asymmetry.

6.1.2. The second example: a spike not centered on integral k

Second, the example of $[E(j) = \cos(2\pi \frac{\nu_{smpl}}{10} t)]$. This is a monochromatic wave whose frequency is exactly one-tenth the Nyquist frequency, i.e. when plotted it is exactly one-fifth the way between 0 and f_N. But, in contrast to the previous example, it does *not* fall on an integral value of k. Again we show the resulting $E(\frac{k\nu_{smpl}}{2J})$ by the stars in Figure 6.1 (bottom), and we show the almost-continuous curve from fractional values of κ in equation 6.1 as the solid curve. The stars are nowhere zero because the signal is centered on a non-integral value of κ. Comments about the sidelobe asymmetry apply here too, of course.

6.2. The repeating windows for nonintegral k

We have extensively discussed the periodic nature of the discrete FT in §4. Specifically, equation 4.3 shows this periodicity: both the input signal and the output spectrum are periodic with period $2J$. But for the input signal, this periodicity exists only if k (κ in equation 6.1) is an integer.[5]

If κ is not an integer, then the interpolated spectra (i.e., for nonintegral κ) at frequency offsets of $J\nu_{smpl}$ (equal to $2Jf_N$) are *not* identical. Fundamentally this is an effect of the time-shift property of FT's. Nevertheless, for the integral values of k, which provide the basic spectral information, the spectra at frequency offsets of $J\nu_{smpl}$ (equal to $2Jf_N$) *are* identical. Here the digital world triumphs over the time-shift theorem!

7. THE FAST FOURIER TRANSFORM

The Fourier transform as defined in equation 4.4 requires of order $(2J)^2$ operations: $2J$ frequencies to be computed, each of which requires a sum over $2J$ datapoints. Many applications generate huge values of J and the computing time becomes impractically long.

Enter the FFT. The fundamental idea is to split the FT into smaller chunks. Suppose, for example, you have 2^N datapoints. You split the FT into $\frac{2^N}{2}$ chunks each of which contains only two

---

[5]Similarly, for the spectrum this periodicity exists only if j is an integer—but j is always an integer!

numbers; you FT each pair; then you put the chunks back together again. This works and requires only $\sim N \log_2 N$ operations. This has two advantages: The obvious one is computing time. The less obvious one is numerical accuracy: fewer operations means smaller roundoff errors in the final result.

Suppose you have an arbitrary number M of datapoints. IDL's FFT routine factors M into as many chunks as possible. It's most efficient for chunks that are powers of 2, 3, or 5. The more factors, the faster the runtime. People tend to think of and always use powers of 2, and this is indeed the fastest kind of FFT, but the presence of other factors can be quite acceptable.

This can lead to large apparent anomalies in runtime. You might have M equal to some large integral power of 2. If then you add just one more point, M might be a prime number! (An easy, but not interesting, example is $M = 16$). The difference in computing time can be enormous. IDL's FFT doesn't warn you about these matters, so you have to think of them yourself—beforehand!

If you have an awkward value of M, then you can create a nice power-of-two value either by cutting out datapoints (do you really want to do this???), or by padding your datapoints with enough zeros to produce the requisite power-of-two condition.

7.1. On Padding with Zeros

Consider padding with zeros. Suppose you are taking datapoints as a function of time t. To retain the proper phase of the signal you need to pad the signal at its beginning and end, symmetrically with equal numbers of zeros at negative and positive times. Recall, however, that in the FFT algorithm the $t = 0$ point is shifted to the beginning. This means that the two ends of the datastream abut at the middle of the shifted array. This, in turn, means that you need to add the zeros *all in the middle* of the array that you use in the FFT procedure.

8. CORRELATION AND CONVOLUTION

Two important theorems regarding FT's:

1. The convolution theorem:

$$FT[s * r] = FT(s) \cdot FT(r) = S(f)R(f) \tag{8.1a}$$

where the capital letter functions mean the FT versions and the convolution is defined as

$$[s * r](t) = \int_{-\infty}^{\infty} s(t_s) r(t - t_s) dt_s \tag{8.1b}$$

In words, this reads: The FT of the convolution of two functions is the product of the FT's of the functions. Regard convolution as the smoothing of one function, the singal $s(t_s)$, by another, the response function $r(\Delta t)$. .

The classical example of convolution is in electric circuits. A signal varies with time ("signal time") t_s. It passes through some electronic black box, for example an RC circuit. This circuit has impulse response $r(\Delta t) = e^{-\Delta t/RC}$, where Δt is the time after the impulse is applied; the FT is $\frac{RC}{1+[2\pi i]RC\nu}$, so it acts as a low pass filter, attenuating high frequencies.

A good astronomical example is atmospheric seeing: the star image, which is infinitesimally sharp, is blurred by the atmospheric seeing. If the seeing is Gaussian, then the observed star image is the convolution of its true image with the atmospheric Gaussian. This example, as many, has a symmetric response function, in contrast to the above RC circuit example.

An important aspect of convolution is the reversal of the sense of "signal time" t_s, or the independent variable whatever it is, for the response function r in the integral. The response function gets flipped. This flipping seems strange—but it's important. For symmetric response functions, as we often encounter in astronomy, this doesn't matter—but be aware!

2. The correlation theorem:

$$FT[corr(s,r)](\tau) = FT(s(t)) \cdot [FT(r(t))]^* = S(f)R^*(f) \tag{8.2a}$$

where the correlation is defined as

$$[corr(s,r)](\tau) = \int_{-\infty}^{\infty} s(t)r(t+\tau)dt \tag{8.2b}$$

In words, this reads: The FT of the crosscorrelation of two functions is the product of the FT of one function by the complex conjugate of the FT of the other.

The classical and most important example of crosscorrelation is in deriving power spectra. Here, we take two time series, compute their integrated product as a function of the delay τ; the power spectrum is the FT of this cross correlation function.

This theorem is particularly important for the *autocorrelation* function, namely the crosscorrelation of a function with itself. Here, the theorem reads: "The power spectrum, defined as the FT of the signal times its complex conjugate, is equal to the FT of the signal's autocorrelation function." This has wide use in radio astronomy and, also, in spectral interferometry. There are two methods of calculating power spectra: First, the classical one, the FT of the signal times its complex conjugate; this is called the FX method [Fourier Transform, then multiplication (detection)]. Second, the XF method: (multiplication, then FT). The second method is popular with radio astronomers because it is easy to design a hugely parallel processor to do auto- and crosscorrelation.

Aside from the use of the correlation theorem in spectral analysis, there are two important applications of these theorems. One is in calculating convolutions. If you have a big CCD image and want to calculate what it would look like under various conditions of atmospheric seeing, you need to convolve the seeing function with the image. This requires $\sim N^2$ operations, where N is the number of pixels. Using FFT techniques, you cut this to $\sim N \log_2 N$. The other is in deconvolution: the product of two FT's convolves, while the ratio of two FT's deconvolves—it's magic! There are issues regarding noise and zeros in the denominator, though! See NM §13.1.

These theorems are identical if the response function r is symmetric. We will assume this to be the case and focus the discussion on autocorrelation as the specific example.

8.1. Digital Calculation of the Autocorrelation Function.

The way to digitally calculate the autocorrelation function $A(\tau)$ of the time-dependent function $E(t)$ is to carry its definition, which is by an analytic integral, to numerical summation in the standard way. We begin shifting the origin of the time axis so that all times are positive, which is the way we usually think of our samples, so we write

$$A(\tau) = \lim_{T \to \infty} \frac{1}{2T} \int_0^{+2T} E(t) E(t + \tau) dt \; . \tag{8.3}$$

For example, if $E(t) = \sin(2\pi\nu t)$, then $A(\tau) = cos(2\pi\nu\tau)$. This illustrates the general property that autocorrelation removes all phase information. (It has to: it's symmetric in τ, so its FT is always real!)

Now let's translate this into a digital sum. We have $2N$ discrete samples separated uniformly in time by $\Delta t = t_{smpl} = \frac{1}{\nu_{smpl}}$. Recall your elementary calculus in which an integral was defined by cutting up the x-axis into tiny bits and taking a sum over the bits. Here we are integrating over time, so it makes sense to make the "tiny bit" equal to the sample time t_{smpl}. In terms of sample number n, we can write $t = nt_{smpl}$ and $dt = t_{smpl}$, so $E(t) = E(nt_{smpl})$ and, for convenience, we just write $E(n)$. Similarly, we will calculate $A(\tau)$ only for discrete values of τ whose increment is also t_{smpl}; we write $\tau = jt_{smpl}$ and, as for E, we write $A(j)$ instead of $A(jt_{smpl})$. With all this, the direct analog of equation 8.3 in summation form is

$$A(j) = \lim_{N \to \infty} \frac{1}{2N} \sum_{n=0}^{2N-1} E(n) E(n + j) \; . \tag{8.4}$$

This looks innocent enough, but it misses a fundamental fact of the real world: we don't live forever, so we can't let $N \to \infty$. No problem; we just get rid of the $\lim_{N \to \infty}$ and write the even simpler form

$$A(j) = \frac{1}{2N} \sum_{n=0}^{2N-1} E(n)E(n+j) \ . \tag{8.5}$$

But wait! We have just $2N$ samples of E—that is, $E(n)$ is defined only for $n = 0 \rightarrow 2N-1$. So in the sum, whenever $n + j > 2N - 1$, we're in trouble—we have no samples to put in the sum! In other words, when N is finite, you have the problem of "end effects". What to do?

Now's the time to go back and review §4 and figure 5.1. There we stressed that the summation form of the FT implicitly, and necessarily, assumes that both the input and output arrays are *periodic* outside the fundamental window of length $2N$—and the period is just $2N$. So it's obvious what to do: when $n + j > 2N - 1$, you use $E(n + j - 2N)$.

Similarly, $A(j)$ is periodic with period $2N$. Thus, $A(j)$ is defined for the interval $j = 0 \rightarrow 2N - 1$. And, of course, this periodicity makes it easy to generate values for $j < 0$.

8.2. !!!!!!!!!WARNING!!!!!!!!!

Re-read the paragraphs immediately above. They state that the end effects are no problem because the math automatically "wraps around" in the calculation of correlation functions. This means that the *beginning* of the data stream gets correlated with the *end* of the data stream! Generally speaking, you don't want this to happen because the beginning and end are distinct and totally unrelated!

What do do? Pad the beginning and end symmetrically with zeros! (Be sure to look at §7.1). This ensures that the two ends of the data stream do not interact. And make sure you use enough!! See NM discussion §13.1.

8.3. Calculating correlation functions in IDL

IDL provides several native routines for calculating correlation functions. You have to be very careful, though. *Read their documentation* before blindly forging ahead. Here we summarize.

You might be tempted to use the routines **a_correlate** and **c_correlate** for auto- and crosscorrelation. Be aware that these don't do what you think. First, they subtract the mean before doing the calculation—something you don't want, if for no other reason that the zeros you so carefully padded with become *nonzero* when the mean is subtracted! Second, they have sliding limits on the sums, meaning that different numbers of terms are included for different delays. I'd stay away from these procedures if I were you. Frankly, I can't imagine any circumstance for which they would be useful.

What you want is IDL's **convol** routine, which convolves two arrays and allows you to "edge_wrap", which is the equivalent of wrapping around as you need to do to account properly for the edge effects in correlation as discussed above in §8.1. **convol** works on all types of arrays, but uses *only* the clunky non-FFT technique. You call this function as **result = convol(array1, array2, /center /edge_wrap)**. *Catch number 1*: **array2** needs to be smaller than **array1**. If your arrays have the same size, you need to expand **array1** (padding) or condense **array2**. When padding, be aware of the edge effects of convolution: if the edges of the datastream are nonzero and you pad with zeros, convolution will give you a large meaningless contribution at the edges! *Catch number 2:* The center keyword is *set by default*, in which case **convol** does a *correlation* instead of a *convolution*! To force a convolution, explicitly set **center=0** in the call to **convol**; in this case, for a symmetric kernal **array2** the output is shifted from the input by $\frac{M}{2}$, where M is the size of **array2**.

Or, to make it quicker, use the FFT technique! For one dimensional arrays this is easy (but you'd better experiment with some known simple functions to make sure you understand the details). For two dimensional arrays, you can use the Goddard library's **convolve** function, which gives you the choice of using either the FFT or the standard (clunky time consuming) technique. Unfortunately, **convolve** uses FFT *only* on 2-d images; for 1-d arrays it uses **convol**, which has ramifications for padding. *Important: with the FFT technique, both the signal* **array1** *and the kernal* **array2** *must be the same size, so you need to pad both.*

8.4. Calculating the Fourier Transform of the Autocorrelation Function

We do this using equation 4.3 using the variables appropriate here, that is...

$$P(k) = \frac{1}{2J} \sum_{j=-J}^{J-1} A(j) e^{[\pi i] \frac{kj}{J}} \ . \tag{8.6}$$

Here, the frequency $\nu = \frac{k\nu_{smpl}}{2J}$, and for convenience we write $P(k)$ instead of $P(\frac{k\nu_{smpl}}{2J})$.

Now let's notice that, with a suitable change of variables in our above equation (2a), you can easily determine that $A(\tau) = A(-\tau)$: the autocorrelation function is *symmetric* in τ. This means that the imaginary portion of its FT is automatically zero. So, in taking the FT, you don't even have to specify that we want just the real part of the result! *BUT* symmetrizing a digitally sampled $A(\tau)$ is a bit tricky and you need to follow the prescription in §9.

8.5. The FX versus XF methods: Not Entirely Equivalent!!

The correlation theorem says that the FX and XF methods of calculating power spectra should provide identical results. Not many people realize that this isn't exactly the case.

The reason is that the theorem is proved for integration to *infinity*. In fact, we integrate only over some range of time $2T$. This limits the spectral resolution as discussed in §1: the spectrum is convolved by the FT of the weighting function. The differences between the FX and XF methods arise only in this realm.

There's a fundamental difference between applying weighting functions in the two methods of getting power spectra. In the FX method, you apply the weighting function $W(t)$ to the *voltage*, that is *before* the Fourier transform; and then you "detect" the signal by squaring (really, by multiplying by its complex conjugate). So the weighting function is also "squared". Alternatively, in the XF method, you apply $W(t)$ to the *correlation function*, which is equivalent to the *detected* voltage; the "squaring" has already taken place, so the weighting function does *not* get "squared". Figure 8.1 illustrates the difference.

These applications of $W(t)$ *are not equivalent* and, furthermore, *cannot be made to be equivalent.* One strange result is in the XF method, a monochromatic signal produces $\frac{\sin x}{x}$ type sidelobes in the power spectrum, and these go *negative.* The power spectrum can have *negative power!*. (Of course, it's totally meaningless). This can never happen in the FX method.

In the final analysis—which is too much to discuss here, but the essence is that $W(t) < 1$ so "squaring" it means that it gets smaller—this means that, for identical weighting functions $W(t)$, the leakage is always much smaller with the FX method. You can always make this up by using a more severe weighting function in the XF method, but you lose a bit more resolution than with the FX method.

9. COSINE AND SIN TRANSFORMS

The Fourier transform is by its intrinsic definition a complex operation. However, there are many instances when you need to take a cosine or sin transform. This is straightforward, but it's worth spending some space on this because almost everybody gets it wrong.

Suppose you have J datapoints and you wish to take the cosine transform using the FFT method. That is, you use equation 4.3, which we reproduce here:

$$E(k) = \frac{1}{2J} \sum_{j=-J}^{J-1} E(j) e^{[\pi i] \frac{kj}{J}} \ . \qquad (9.1)$$

To take the cosine transform, you need to make sure that the argument $E(j)$ is symmetric in j. The datapoints $D(j)$ are defined only for $j \geq 0$. Defining the symmetric counterpart would seem to be easy: just define

$$E(j) = D(j) \ ; E(-j) = D(j) \ , (j \geq 0) \ . \qquad (9.2)$$

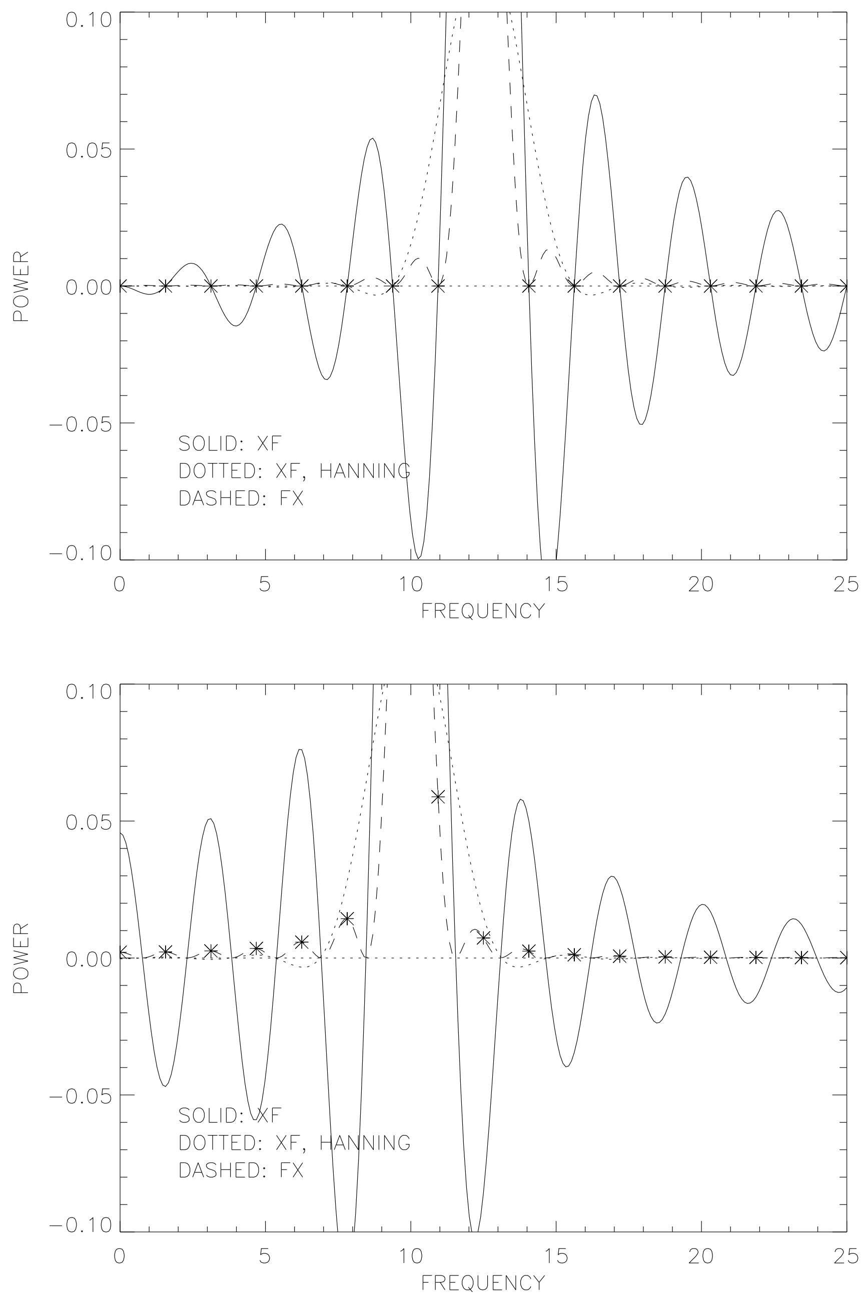

Fig. 8.1.— Comparison of FX and XF methods for a monochromatic signal. $\nu_{smpl} = 100$ Hz and $2T = 0.64$ sec. Solid line is XF method, which can have negative sidelobes; dotted line is XF method with Hanning weighting. Dashed line is FX method. Stars are square-window FFT output spectral points.

This makes the signal symmetric, so that when you take the digital transform using either the FFT or a direct transform you have to get a pure cosine transform.

But you immediately run into a problem if you wish to use the most efficient version of the FFT, for which the number of datapoints needs to be a power of two: the above symmetrization operation produces an odd number of datapoints. Specifically, if you start with J datapoints, you end up with $2J - 1$ datapoints. You have a "missing datapoint".

To get around this difficulty, look at equation 5.3. There you see that the missing datapoint has $j = +J$ and, also, $j = -J$. Because of the periodic nature of the DFT, these two datapoints must be equal to the one and only missing datapoint. You need to set this unknown missing datapoint to a reasonable number. The proper choice for this number is important only insofar as it should produce no discernible impact on the derived Fourier transform.

You might be tempted to set the missing datapoint equal to zero. However, this is the wrong choice! The signal may have a nonzero Fourier component at the adjacent datapoints where $j = \pm(2J - 1)$. Setting the missing datapoint equal to zero then produces a spike at $j = \pm(2J)$, and this spike produces a channel-to-channel oscillation in the derived Fourier spectrum. The proper choice for the missing datapoint is the average of the two values at $j = \pm(2J - 1)$.

Similar comments apply to doing a sin transform using the FFT, except that you need to antisymmetrize the signal. NM §12.3 discusses specific routines for cosine and sin transforms, but IDL does not have these implemented as native procedures.

10. SOME FINAL POINTS

10.1. What's This Business About *Negative Frequencies*?

There are some cases in which one can distinguish between negative and positive frequencies. Specifically, these are cases in which the input to the FT is *complex*. To be complex, the input must have both a real and imaginary part: in other words, each sample consists of two numbers, and these two numbers can be regarded as the real and imaginary parts of a complex number. If you take AY120B, you will encounter such a case.

More probably, you encounter this case in the movies when you see a rotating wheel. The real axis is horizontal and the imaginary is vertical. If the wheel moves backwards the *true* frequency is negative, forwards is positive: if the wheel *appears* to move backwards when it is moving forwards, that's aliasing! And you wouldn't know the wheel appears to go backwards without having both the horizontal and vertical—i.e. real and comples—information.

10.2. For Real Inputs, How Do These Negative Frequencies Enter the Power Calculation?

In the vast majority of applications, the samples consist only of one number: each time sample represents a real voltage (or a real number of photons), and there is nothing imaginary—or complex (mathematically speaking, that is)—about them. But it is perhaps surprising that the FT *output* numbers *are* complex: the imaginary part is *not* zero. The phase angle of each complex number represents the phase of that Fourier component with respect to $t = 0$. For the case of real numbers as input, the outputted complex numbers have a simplification: the imaginary parts are odd and the real parts even (in other words, the negative-frequency number is the complex conjugate of the positive-frequency number).

This means that when you use the complex output spectral numbers to calculate the corresponding power numbers (by $P(k) = E(k) \times [E(k)]^*$), negative and positive frequencies have identical powers. The proper way to combine the powers for the negative and positive frequencies is simply to add them; but because the numbers are identical, it's equivalent to simply use twice the full value of, say, the positive-frequency number. It should be obvious that there is only one number representing zero frequency, so you should *not* multiply this by two.

Thus, in the example above in §5.2, after calculating $P(k) = E(k) \times [E(k)]^*$, your power spectrum is most simply given by the first P_k ($k = 0$) and twice the next four values of $P(k)$ ($k = 1 \rightarrow 4$).

10.3. A Detail on Normalization and "Going Back and Forth"

In IDL, the FFT is normalized by multiplying the sum by $\frac{1}{2J}$, exactly as we've done in equation 4.2. Not all FFT routines do the normalization in this way. This way has the advantage that the scaling of the output is the same as that of the input—in other words, it's sort of like taking an average, because we divide by the number of points that contribute to the sum.

As we've mentioned in §1, you should know that *apart from normalization constants*, you can convert willy-nilly back and forth from frequency to time by applying FT's in succession. That is, $E(k) = FT(E(j))$ and $E(j) = FT^{-}(E(k))$. Here the superscript minus sign indicates using the negative complex exponential in the transform, as in equation 0.1b; this is called the *inverse Fourier transform*. More graphically, $E(j) = FT^{-}[FT(E(j))]$.

With the normalization used by IDL, the inverse transform must *not* multiply the sum by $\frac{1}{2J}$. Of course, IDL's inverse transform does all this correctly. In IDL, you invoke the inverse transform by using the **inverse** keyword—see IDL's help on the **fft** procedure.