RECOVERING SIGNAL FREQUENCIES WITH THE FOURIER TRANSFORM

KENNETH L. HO

Suppose that we are given a signal of the form

$$f(t) = \sum_i A_i \sin(2\pi\omega_i t), \tag{1}$$

i.e., f is the superposition of fundamental waves of amplitudes A_i and frequencies ω_i, and that we are only allowed to sample f at a finite number of equally-spaced time points. How might we determine, on the basis of the samples that we have collected, the identities of the frequencies composing f?

A commonly used tool for this task is the Fourier transform. In the continuous setting, the Fourier transform[1] is the mapping

$$\hat{f}(\omega) = \int_0^\infty f(t)\, e^{-2\pi\imath\omega t}\, dt \in \mathbb{C}$$

of f into the frequency domain, where $|\hat{f}(\omega)|$ gives the "contribution" of the fundamental wave of frequency ω to f. To see this a little more explicitly, it is instructive to introduce also the inverse Fourier transform

$$f(t) = \int_{-\infty}^\infty \hat{f}(\omega)\, e^{2\pi\imath\omega t}\, d\omega,$$

which is just an integral decomposition of f into its fundamental components. Viewed as a generalization of (1), $\hat{f}(\omega)$ clearly represents the amplitude of the wave of frequency ω.

In the posited situation, however, we do not have access to the entire signal f, but only a discrete number of samples $f_1, \dots, f_n$, where $f_i = f(t_i)$, with $t_{i+1} - t_i = \Delta t$, where Δt is the intersample time. Consequently, we will not use the Fourier transform as described above, but instead the discrete Fourier transform (DFT), defined by

$$F_j = \sum_{k=0}^{n-1} f_k e^{-2\pi\imath jk/n}, \quad j = 0, \dots, n-1.$$

Similarly to the continuous case, $|F_j|$ gives the contribution of "frequency" j to the sample signal. But what physical frequency does j correspond to?

Clearly, the maximum frequency $\omega_{\max}$ that we can hope to capture is limited by the sampling frequency $1/\Delta t$. But due to something called the Nyquist condition, we can achieve only half of this—intuitively, it takes two points to represent each wave cycle: one for the "peak" and another for the "trough". Therefore,

$$\omega_{\max} = \frac{1}{2\Delta t}. \tag{2}$$

Now what about the minimum frequency $\omega_{\min}$? For real signals f, there is a symmetry between positive and negative frequencies, so in fact $\omega_{\min} = -\omega_{\max}$. Since the DFT has n components, equally spaced from $\omega_{\min}$ to $\omega_{\max}$, the resolution between the discrete frequencies is

$$\Delta\omega = \frac{\omega_{\max} - \omega_{\min}}{n} = \frac{1}{n\Delta t} \equiv \frac{1}{T}, \tag{3}$$

where T is the total time period over which we sample f.

[1] There are many definitions of the Fourier transform—the one that we use here is certainly not the most common, though it is the most convenient for our purposes.

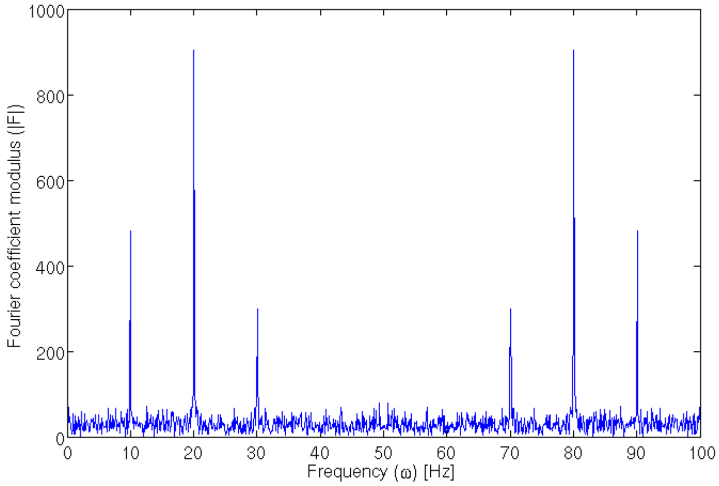

FIGURE 1. Moduli of Fourier transform coefficients $|F|$ of f sampled with $T = 10$ s and $\Delta t = 0.01$ s as a function of the frequency ω (including negative frequencies; Hz).

So which components of F correspond to the negative frequencies? To answer this, we consider the periodicity of the DFT kernel:

$$e^{-2\pi\imath(n-j)/n} = e^{-2\pi\imath}e^{-2\pi\imath(-j)/n} = e^{-2\pi\imath(-j)/n}$$

since $\exp(2\pi\imath) = 1$, so $F_{n-j} = F_{-j}$ is actually the amplitude for "frequency" $-j$. Hence, the DFT is redundant, and the first $\lceil n/2 \rceil$ components cover all non-negative frequencies captured. Thus, for $j = 0, \ldots, \lceil n/2 \rceil$, the physical frequency corresponding to F_j is $j\Delta\omega$. From this perspective, the maximum frequency that we can recover is

$$\omega_{\max} = \lceil n/2 \rceil \Delta\omega = \frac{\lceil n/2 \rceil}{n\Delta t} \sim \frac{1}{2\Delta t},$$

which, of course, is consistent with (2). Observe that (2) and (3) provide guidelines on how to select T and Δt.

Let's do an example now where we actually use this to recover the fundamental frequencies of the signal

$$f(t) = \sin(2\pi \cdot 10 \cdot t) + 2\sin(2\pi \cdot 20 \cdot t) - 0.7\sin(2\pi \cdot 30 \cdot t),$$

i.e., f is composed of waves of 10, 20, and 30 Hz, if t is given in seconds. We will additionally corrupt f with white noise to simulate an imperfect sensor. The DFT is computed in MATLAB by the command `fft`, for the fast Fourier transform—this is how the DFT is almost always computed in practice.

To ensure that we can capture the highest frequency of 30 Hz[2], we use (2) to derive the condition $\Delta t \leq 1/60 \approx 0.0167$ s; we hence choose $\Delta t = 0.01$ s for simplicity. For moderate resolution, we set $T = 10$ s. The moduli $|F_j|$ of the transformed signal are shown in Figure 1. Strong peaks about 10, 20, and 30 Hz clearly pinpoint the constituent frequencies. However, there are also peaks about 70, 80, and 90 Hz; as discussed above, these correspond to the negative frequencies -30, -20, and -10 Hz, respectively, as can be seen from the fact that the amplitudes of the corresponding frequencies are identical—this is the sense in which we mean that the DFT is redundant.

We can clean this up a bit by dropping off the second half of the DFT data. There are a few details in doing this cleanly (primarily involving promoting n up to the next power-of-two); these are all given in the function `fft_freq`[3]:

[2] Note that we don't usually have such knowledge *a priori*.

[3] All codes can be downloaded at `http://www.courant.nyu.edu/~ho/teaching/2011/nyu/spring/comput-med-biol/`.

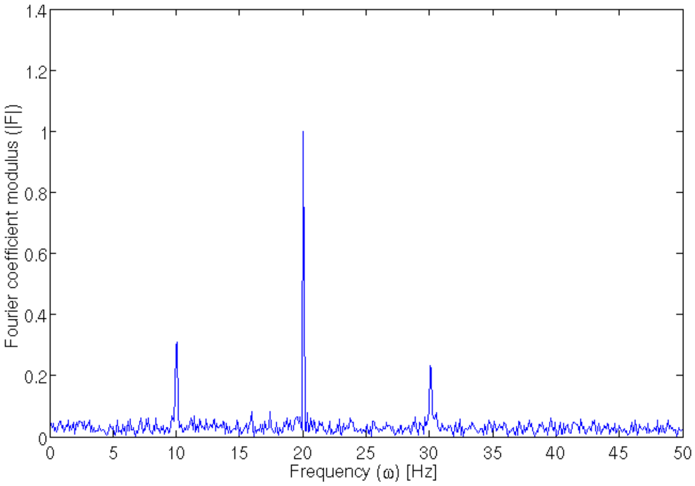

Figure 2. Moduli of Fourier coefficients $|F|$ of f sampled with $T = 10$ s and $\Delta t = 0.01$ s as a function of the frequency ω (Hz).

```
%FFT_FREQ  Compute a FFT restricted to non-negative frequencies and return
%          the frequencies associated with each Fourier component.
%
%   X = FFT_FREQ(X,T,DT) returns the discrete Fourier transform of X
%   sampled over the time interval T in time increments of DT, restricted
%   to non-negative frequencies only. The time interval T must have the
%   form [TMIN, ..., TMAX]; only the first and last elements are used.
%
%   [X,W] = FFT_FREQ(X,T,DT) also returns the frequencies W associated with
%   each component of X.
%
%   See also FFT.
function [X, w] = fft_freq(x, t, dt)
    n = ceil((t(end) - t(1)) / dt);
    nfft = 2^nextpow2(n);
    X = fft(x,nfft) / n;
    X = X(1:ceil(nfft/2));
    w = 1/(2*dt) * linspace(0, 1, ceil(nfft/2));
end
```

The script `demo_fft_freq` below demonstrates its use:

```
%% initialize
f = @(t)(     sin(2*pi*10*t) + ...     % input signal
           2 *sin(2*pi*20*t) - ...
          0.7*sin(2*pi*30*t));
T = 10;                                % sampling time interval
dt = 0.01;                             % sampling time period
%% compute
n = ceil(T/dt);                        % number of sample points
t = linspace(0, T, n);                 % sampling times
x = f(t) + randn(1,n);                 % sampled noisy signal
[X, w] = fft_freq(x, t, dt);           % compute FFT
%% plot
figure
plot(w, abs(X))                        % frequency contributions
xlabel('Frequency␣(\omega)␣[Hz]')
ylabel('Fourier␣coefficient␣modulus␣(|F|)')
```

Figure 2 shows the Fourier coefficients for the same sampling parameters as Figure 1, but with only the first half of the DFT data shown. Finally, to show the effects of resolution, we give two more plots: Figure 3 with $T = 1$ s, and Figure 4 with $T = 100$ s.

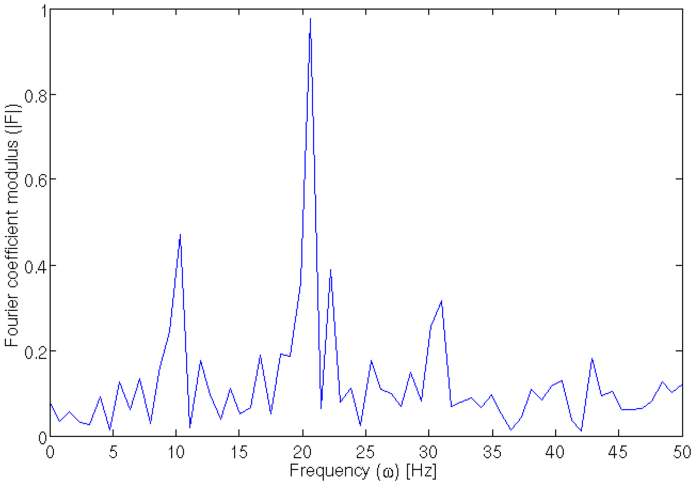

Figure 3. Moduli of Fourier coefficients $|F|$ of f sampled with $T = 1$ s and $\Delta t = 0.01$ s as a function of the frequency ω (Hz).

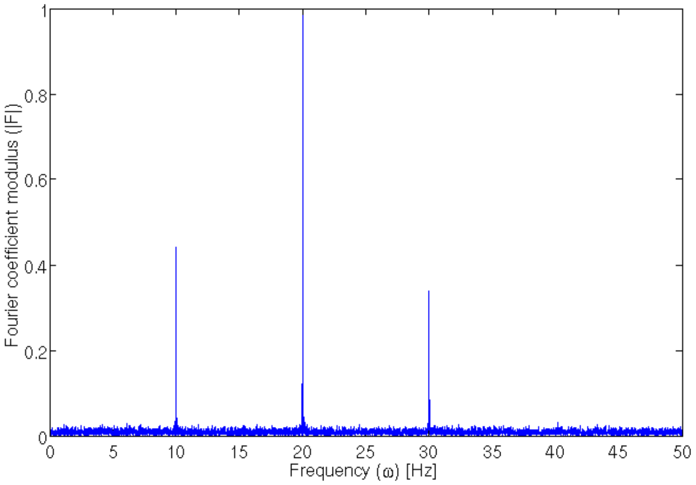

Figure 4. Moduli of Fourier coefficients $|F|$ of f sampled with $T = 100$ s and $\Delta t = 0.01$ s as a function of the frequency ω (Hz).