Lecture 27
Filtering

- Recursive and non recursive filters.
- Linear predictive coding.
- The Lomb periodogram.
- Evaluating integrals using FFTs.

Linear filters:

Given a sequence of input x_k a linear filter produces an output y_k,

$$y_n = \sum_{k=0}^{M} c_k \; x_{n-k} + \sum_{j=1}^{N} d_j \; y_{n-j}$$

The output generically depends on the M sampled values of x and possibly on N of the already filtered values of y. if N=0 the filter is called “non-recursive”. The Fourier transform of the filter is,

$$\mathcal{H}(f) = \frac{\sum_{k=0}^{M} c_k e^{-2\pi i k (f\Delta)}}{1 - \sum_{j=1}^{N} d_j e^{-2\pi i j (f\Delta)}}$$

To design a filter one usually wants the inverse, that is, the c’s and d’s as a function of H. This is an “inverse problem” and is therefore hard.

One clearly has to make compromises sin H(f) is a function and the d’s and c’s are a finite number of parameters to adjust. The subject of digital filter design is about those compromises. We will sketch a couple of basic techniques to get you started.

Non-recursive filters:

In this case the denominator of the formula for H(f) is unity and one is left with a Fourier transform. If one simply does the inverse transform one can solve for the c's.

This is not a good idea. The reason is that the resulting H(f) so constructed will oscillate (usually strongly) between the frequencies f_k where it gets "pinned down" by the equation. Like interpolating polynomials of high order do. This is not good for a filter.

A better idea is to take a rather large number M, perform the Fourier transform and then keep only the initial coefficients. That way one cuts out the high frequency oscillations. One then transforms back to check that the filter indeed is doing what one wants. If it does not, one can try keeping a larger number of coefficients.

A more sophisticated version of this idea is the Remes Exchange algorithm that produces the best Chebyshev approximation to a given Response with a fixed number of coefficients.

Recursive filters, stability:

Recursive filters, those that depend on their own output, tend to be superior to non-recursive ones. Why? A non-recursive filter is a polynomial in the variable 1/z, $z=e^{2\pi i f \Delta}$. A recursive filter is a rational function in 1/z. As we learnt with Padé, one can do better with rational approximations than with polynomial ones.

Recursive filters have, however, a potential drawback. A non-recursive filter is guaranteed that if one "turns off" the signal x_k the output will eventually turn off as well. Recursive filters do not guarantee this, in fact, they can develop modes that blow up over time.

Designing a recursive filter is therefore not only a hard inverse problem: it is an inverse problem with stability issues.

To test the stability we apply the same criteria we have applied before in the course. The definition of a linear fiter: $y_n = \sum_{k=0}^{M} c_k \, x_{n-k} + \sum_{j=1}^{N} d_j \, y_{n-j}$

can be viewed as a recursion relation on the y's.

For the recurrence relation to be stable, let us ignore the c's, and require that the roots of the characteristic polynomial,

$$z^N - \sum_{j=1}^{N} d_j z^{N-j} = 0$$

be less than unity, |z|<1. To simplify the discussion let us reparametrize

$$w \equiv \tan[\pi(f\Delta)] = i\left(\frac{1-e^{2\pi i(f\Delta)}}{1+e^{2\pi i(f\Delta)}}\right) = i\left(\frac{1-z}{1+z}\right)$$

This change of variable maps the Nyquist interval to the real line.

The inverse transform is, $z = e^{2\pi i(f\Delta)} = \frac{1+iw}{1-iw}$

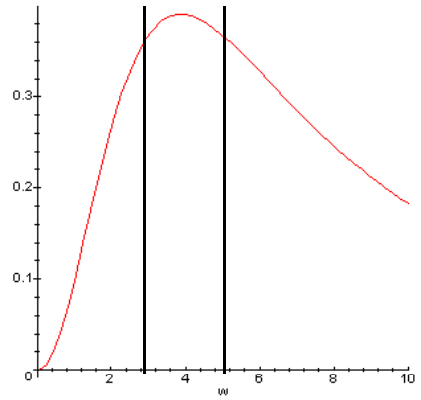
And the condition |z|<1 implies Im(w)>0.

This reparametrization allows the following construction (called the "bilinear transformation method"): construct the square of the filter, $|H(f)|^2=H(f)H^*(f)=H(f)H(-f)$. Such function will have symmetric roots on top and bottom of the real axis. Then construct H(f) by choosing products of the poles in the upper part. Then go back and compute the c's and d's.

Example:

Suppose you want to construct a band pass filter, that is a filter that falls off quickly for frequencies lower than w=a and bigger than w=b. A function that accomplishes this is,

$$|\mathcal{H}(f)|^2 = \left(\frac{w^2}{w^2+a^2}\right)\left(\frac{b^2}{w^2+b^2}\right)$$



If you want sharper cutoffs, just take a power of this function. The poles are obviously at w=+/-ia and w=+/-ib. We then choose the poles in the bottom of the complex plane to construct,

$$\mathcal{H}(f) = \left(\frac{w}{w-ia}\right)\left(\frac{ib}{w-ib}\right) = \frac{\left(\frac{1-z}{1+z}\right)b}{\left[\left(\frac{1-z}{1+z}\right)-a\right]\left[\left(\frac{1-z}{1+z}\right)-b\right]}$$

Or,

$$\mathcal{H}(f) = \frac{-\frac{b}{(1+a)(1+b)} + \frac{b}{(1+a)(1+b)}z^{-2}}{1 - \frac{(1+a)(1-b)+(1-a)(1+b)}{(1+a)(1+b)}z^{-1} + \frac{(1-a)(1-b)}{(1+a)(1+b)}z^{-2}}$$

From the last expression we can just read off the coefficients,

$$c_0 = -\frac{b}{(1+a)(1+b)}$$

$$c_1 = 0$$

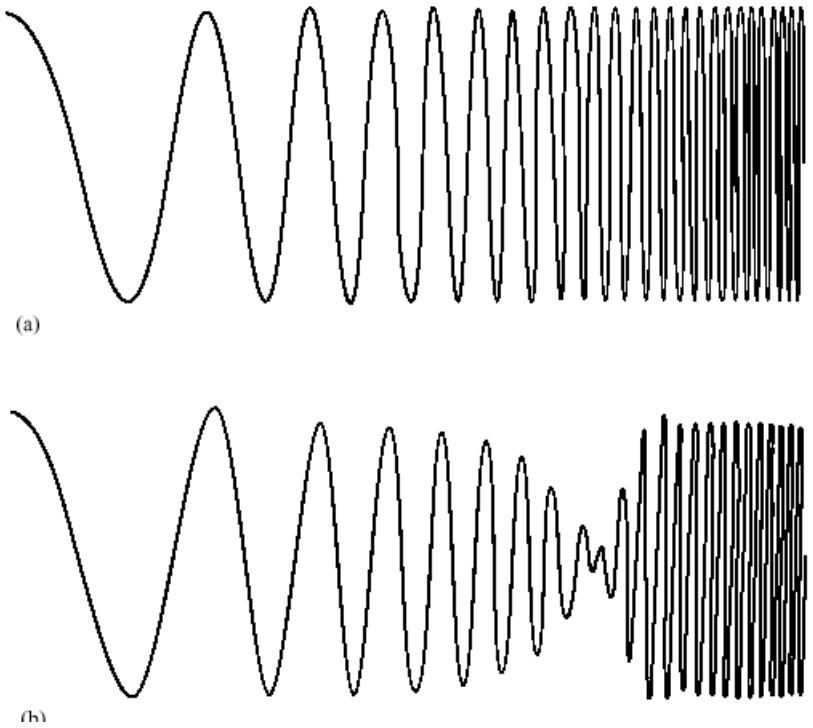
$$c_2 = \frac{b}{(1+a)(1+b)}$$

$$d_1 = \frac{(1+a)(1-b)+(1-a)(1+b)}{(1+a)(1+b)}$$

$$d_2 = -\frac{(1-a)(1-b)}{(1+a)(1+b)}$$

And this completes the design of the filter.

A “chirp” signal (ramp-up in frequency) passed through the “notch” filter.



Linear prediction and linear predictive coding (LPC):

Suppose you have a set of measured values y', related to an underlying set of true values y, through some noise,

$$y'_\alpha = y_\alpha + n_\alpha$$

(we use a Greek index to denote that the discussion does not require equal spacing in the measurement). Suppose we wish to compute, based on the measured noisy values, the value at some point, linearly,

$$y_\star = \sum_\alpha d_{\star\alpha} y'_\alpha + x_\star$$

Where the x refers to the discrepancy.

If we considered y_{star} to be one of the y_α's we have the previously considered problem of optimal filtering. If we don't choose it that way, we have the problem of linear prediction.

A natural way of minimizing the discrepancy is in the statistical mean square sense, seeking d's that minimize,

$$\begin{aligned}\langle x_\star^2 \rangle &= \left\langle \left[\sum_\alpha d_{\star\alpha}(y_\alpha + n_\alpha) - y_\star \right]^2 \right\rangle \\ &= \sum_{\alpha\beta} (\langle y_\alpha y_\beta \rangle + \langle n_\alpha n_\beta \rangle) d_{\star\alpha} d_{\star\beta} - 2 \sum_\alpha \langle y_\star y_\alpha \rangle \, d_{\star\alpha} + \langle y_\star^2 \rangle\end{aligned}$$

Where we have used that the signal is uncorrelated with the noise.

$$\langle n_\alpha y_\beta \rangle = 0.$$

It is useful to introduce a matrix notation for these calculations,

$$\phi_{\alpha\beta} \equiv \langle y_\alpha y_\beta \rangle \qquad \phi_{\star\alpha} \equiv \langle y_\star y_\alpha \rangle \qquad \eta_{\alpha\beta} \equiv \langle n_\alpha n_\beta \rangle$$

And for point-to-point uncorrelated noise one sometimes has

$$\langle n_\alpha n_\beta \rangle = \langle n_\alpha^2 \rangle \, \delta_{\alpha\beta}$$

If we minimize the previous expression by equating to zero its derivative with respect to the d's, we get, in matrix notation,

$$\sum_\beta \left[\phi_{\alpha\beta} + \eta_{\alpha\beta}\right] d_{\star\beta} = \phi_{\star\alpha} \qquad y_\star \approx \sum_{\alpha\beta} \phi_{\star\alpha} \left[\phi_{\mu\nu} + \eta_{\mu\nu}\right]^{-1}_{\alpha\beta} y'_\beta$$

For instance, if we take star to be one of the alpha's and assume that the matrices are diagonal, we get,

$$y_\gamma = \frac{\phi_{\gamma\gamma}}{\phi_{\gamma\gamma} + \eta_{\gamma\gamma}} y'_\gamma$$

Which we can recognize as the formula for the Wiener filter if we take phi->S^2 and eta->N^2

That is, in Fourier space autocorrelations become squares of Fourier amplitudes. But we see the above discussion is more general and valid in any space.

The Lomb periodogram:

Suppose you have a signal sampled at irregular intervals. The techniques we have discussed up to now for doing Fourier transforms all required equal spacing. In particular the power spectrum calculation

One could interpolate a regularly sampled signal from an irregular one. Or in cases in which there are “holes in the data” in a regularly sampled signal, try to fill the holes by setting the signal to zero or to a constant. These techniques do not work terribly well. Extrapolation assumes properties of the signal that might not be there. Filling holes tends to overweight the low frequencies.

Lomb proposed a formula for a modified periodogram that works with irregularly sampled data. One starts by computing the mean and the variance of the data,

$$\overline{h} \equiv \frac{1}{N}\sum_{1}^{N} h_i \qquad \sigma^2 \equiv \frac{1}{N-1}\sum_{1}^{N}(h_i - \overline{h})^2$$

Then the Lomb periodogram is given by

$$P_N(\omega) \equiv \frac{1}{2\sigma^2}\left\{\frac{\left[\sum_j (h_j - \overline{h})\cos\omega(t_j-\tau)\right]^2}{\sum_j \cos^2\omega(t_j-\tau)} + \frac{\left[\sum_j (h_j - \overline{h})\sin\omega(t_j-\tau)\right]^2}{\sum_j \sin^2\omega(t_j-\tau)}\right\}$$

Where the "shift" tau is given by,

$$\tan(2\omega\tau) = \frac{\sum_j \sin 2\omega t_j}{\sum_j \cos 2\omega t_j}$$

The theoretical construction leading to this periodogram is rather elaborate, so we won't discuss it here. We will concentrate on what it accomplishes. The above formula gives the same result, for a given frequency ω, as if one went and estimated the power in that frequency by fitting

$$h(t) = A\cos\omega t + B\sin\omega t$$

to the given signal, using the least-squares method. We therefore see that the method weighs things in a "per frequency" basis. The usual periodogram weighed power in a frequency bin. This obviously does not make sense for irregular data.

An interesting use of the Lomb periodogram is to separate signals from noise. It can be showed that, if the signal is just white noise, the probability that the Lomb periodogram would yield a value between z and z+dz is proportional to exp(-z).

Computing Fourier integrals using FFT:

On many occasions one is interested in computing integrals of the form,

$$I = \int_a^b e^{i\omega t} h(t) dt \ ,$$

With h(t) a continuous function, but not necessarily periodic.

The big computational breakthrough implied by the FFT method makes it tempting to bring it to bear on this problem. The problem has subtleties. To see how they arise, let us go about it naively.

The obvious thing to try is to start by dividing the interval into M subintervals (with M large),

$$\Delta \equiv \frac{b-a}{M}\,, \quad t_j \equiv a + j\Delta\,, \quad h_j \equiv h(t_j)\,, \quad j = 0, \ldots, M$$

We then just approximate the integral by the sum

$$I \approx \Delta \sum_{j=0}^{M-1} h_j \exp(i\omega t_j)$$

And we know that such formulas are first-order accurate in the spacing of the points. For certain values of M and omega, this expression is identical to that of a discrete Fourier transform. In particular for M even and ,

$$\omega_m \Delta \equiv \frac{2\pi m}{M} \qquad m = 0, 1, \ldots, M/2 - 1.$$

$$I(\omega_m) \approx \Delta e^{i\omega_m a} \sum_{j=0}^{M-1} h_j e^{2\pi i m j/M} = \Delta e^{i\omega_m a} [\mathrm{DFT}(h_0 \ldots h_{M-1})]_m$$

One has to be very careful with all this. The problem lies in the oscillatory nature of the integrands of the integrals we are trying to compute,

$$I = \int_a^b e^{i\omega t} h(t) dt \ ,$$

If h(t) is smooth and omega is large, the resulting integral is typically very small, easily swamped by roundoff and truncation error. The results get worse and worse for increasing omega.

Let us try a more sophisticated approach. Given the values $h(t_i)$ in the interval of interest, we use them to interpolate an approximation to h(t). Such interpolation could be linear, cubic, etc. The bottomline is that one is then left with a bunch of simple integrals that can be computed analytically multiplied times the values of the function h_i. Let us see how this works explicitly.

If one does polynomial interpolation, the formula for the value of the function is a polynomial in the variable t where the coefficients are linear functions in the values of the function h_i. A (rather unconventional) way to write this is,

$$h(t) \approx \sum_{j=0}^{M} h_j \, \psi\left(\frac{t-t_j}{\Delta}\right) + \sum_{j=\text{endpoints}} h_j \, \varphi_j\left(\frac{t-t_j}{\Delta}\right)$$

Where ψ(s) is a "kernel function" that is zero for large (+ or -) s and is non-zero in the interval in which we interpolate h(t). Normally one needs special expressions in the endpoints of the interval, since there one would not be able to "center" the interpolation scheme. We write this explicitly in the expression above.

We now insert this expression in the integral of interest and get,

$$I \approx \Delta e^{i\omega a}\left[W(\theta)\sum_{j=0}^{M} h_j e^{ij\theta} + \sum_{j=\text{endpoints}} h_j \alpha_j(\theta)\right]$$

Where we have defined, $s=(t-t_j)/\Delta$ in the first sum and $s=(t-a)/\Delta$ in the last sum. Also $\theta=\omega\Delta$

The integrals, $W(\theta) \equiv \int_{-\infty}^{\infty} ds\, e^{i\theta s}\psi(s)$ $\alpha_j(\theta) \equiv \int_{-\infty}^{\infty} ds\, e^{i\theta s}\varphi_j(s-j)$

are easy to compute analytically for a given interpolation scheme. The first sum in

$$I \approx \Delta e^{i\omega a}\left[W(\theta)\sum_{j=0}^{M} h_j e^{ij\theta} + \sum_{j=\text{endpoints}} h_j \alpha_j(\theta)\right]$$

Can now be evaluated using FFT methods. Numerical recipes lists the explicit form of the alpha's and W's for a few popular interpolation methods.

Summary

- Recursive filters can be unstable, but one can stabilize them.
- Linear predictive coding provides a general framework to discuss filters.
- Lomb's formula provides a periodogram for unevenly spaced data.
- There are tricks to compute Fourier integrals using FFTs.