Class 27. Fourier Transforms, Part 2

- For each n discrete frequencies must compute sum of k discrete times $\Longrightarrow \mathcal{O}(N^2)$ operation.
- Can we do better? Yes!

The Fast Fourier Transform (FFT)

- Cf. *NRiC* §12.2.
- Strategy: divide and conquer. Notice

$$\begin{aligned} H_n &= \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N} \\ &= \sum_{k=0}^{N/2-1} h_{2k} e^{2\pi i (2k) n/N} + \sum_{k=0}^{N/2-1} h_{2k+1} e^{2\pi i (2k+1) n/N} \\ &= \sum_{k=0}^{N/2-1} h_{2k} e^{2\pi i k n/(N/2)} + e^{2\pi i n/N} \sum_{k=0}^{N/2-1} h_{2k+1} e^{2\pi i k n/(N/2)} \\ &= H_n^e + e^{2\pi i n/N} H_n^o, \end{aligned}$$

  where H_n^e and H_n^o are periodic in n with length $N/2$.
- Can continue this process:

$$\begin{aligned} H_n^e &= H_n^{ee} + e^{2\pi i n/(N/2)} H_n^{eo} \\ H_n^o &= H_n^{oe} + e^{2\pi i n/(N/2)} H_n^{oo}, \end{aligned}$$

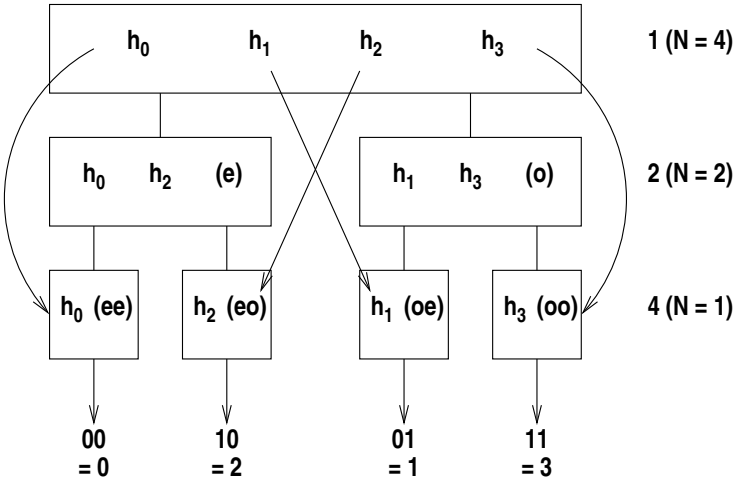
  where each of H_n^{ee}, H_n^{eo}, H_n^{oe}, and H_n^{oo} are periodic in n, length $N/4$.
- If N is a power of 2 (zero-pad your data if not!), can continue process until you get transforms of length 1. What is FT of length 1? Just identity op that copies its one input number (h_k) into its output slot! I.e.,

$$\sum_{k=0}^{0} h_k e^{2\pi i k n/(N/N)} = h_k.$$

- End up with $\log_2 N$ pattern of e's & o's such that, e.g.,

$$H_n^{eooeeeoe\cdots eoo} = h_k,$$

  for some k. (This doesn't depend on n, since it's periodic with length 1.)
- Trick is to figure out which k corresponds to which pattern of e's & o's. Solution: reverse (left-to-right) pattern of e's & o's and let $e \equiv 0$, $o \equiv 1$—this is the binary representation of k! Why? Schematically,



Algorithm

- Rearrange input data in bit-reversed order. This gives the 1-pt FTs. Combine adjacent pairs to get 2-pt FTs, then adjacent pairs to get 4-pt FTs, and so on until you get desired N-pt FT.
- Each combination requires $\mathcal{O}(N)$ ops to perform. There are $\log_2 N$ combinations, $\therefore$ method is $\mathcal{O}(N \log_2 N)$ (assuming bit-reversal sorting is also $\mathcal{O}(N \log_2 N)$, which it is).
- This is called "decimiation-in-time" (Cooley-Tukey FFT). Could also do decimation-in-frequency (Sande-Tukey FFT). Can also stop with base-4, base-8, or even base-prime (for arbitrary N) to exploit various optimizations.

NRiC Implementation

- *NRiC* implements Cooley-Tukey as `four1()`:

```
void four1(float data[],unsigned long nn,int isign)
```

- Use `isign = +1` for forward transform, `-1` for inverse (times N). Here `nn` is N. `data` is `2*nn` elements long: 1st, 3rd, ... elements are *real* components; 2nd, 4th, ... are imaginary.
- Transform returned in `data` in same fashion, but using frequency-ordering conventiond discussed earlier (cf. *NRiC* Fig. 12.2.2).
- If input function is pure real, can gain efficiencies (e.g., `twofft()`, `realfft()`).
- For 2-D, e.g., 2-D grid $0 \le k_1 \le N_1 - 1$, $0 \le k_2 \le N_2 - 1$:

$$H(n_1, n_2) = \sum_{k_2=0}^{N_2-1} \sum_{k_1=0}^{N_1-1} h(k_1, k_2) e^{2\pi i k_2 n_2 / N_2} e^{2\pi i k_1 n_1 / N_1}.$$

  - Pull sub-2 expression out of sub-1 summation $\Rightarrow$ FFT-on-2 $\{$FFT-on-1 $[h(k_1, k_2)]\}$.

Discrete convolution (*NRiC* §13.1, `convlv()`)

- Convolution with a response function of finite duration N:

$$(r \star s)_j \equiv \sum_{k=-N/2+1}^{N/2} s_{j-k} r_k.$$

- Convolution theorem:

$$\sum_{k=-N/2+1}^{N/2} s_{j-k} r_k \Longleftrightarrow S_n R_n.$$

  Here s_j is periodic with period N and the r_k's are stored in wrap-around order.

Discrete correlation (*NRiC* §13.2, `correl()`)

- Correlation of two sampled functions g_k and h_k, each of period N:

$$\mathrm{Corr}(g, h)_j \equiv \sum_{k=0}^{N-1} g_{j+k} h_k.$$

- Correlation theorem:

$$\mathrm{Corr}(g, h)_j \Longleftrightarrow G_k H_k^\star.$$

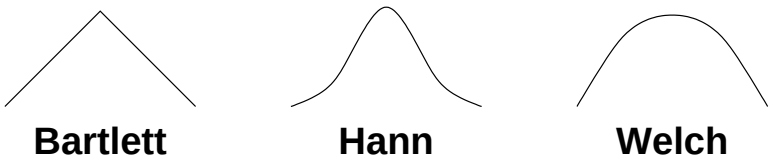
Power spectrum estimation (*NRiC* §13.4, `spctrm()`)

- A *periodogram* is the discrete analog of the power spectrum.
- Since f_k's are discrete, expect $P(f_k)$ to be some kind of average of $P(f)$ over a narrow window function centered on f_k:

$$W(s) = \frac{1}{N^2} \left[\frac{\sin(\pi s)}{\sin(\pi s/N)} \right]^2,$$

  where s is the frequency offset, in bins. Note $\lim_{s\to\infty} W = (\pi s)^{-2}$. $W(s)$ is actually the square of the DFT of the unity (square) window function.

- There will be significant leakage if f is not a pure sine wave.
- To minimize leakage, choose a "rounder" window. E.g.,



- Other methods for power spectrum analysis exist, e.g., maximum entropy "all poles" method (*NRiC* §13.7), Bayesian analysis, etc.

Unevenly spaced data points

- Cf. *NRiC* §13.8, `period()` and `fasper()`.
- Can interpolate onto a regular grid, but may get spurious power at low frequencies.
- Instead, use Lomb normalized periodogram (see *NRiC*).
  - Normally $\mathcal{O}(N^2)$ but can use approximation to get $\mathcal{O}(N \log N)$.
  - May remove aliasing (i.e., get real power above f_c).