Fast Fourier transform

There are three kinds of Fourier transforms: finite discrete, infinite discrete, and continuous. The first is defined by

$$G_j = \sum_{k=0}^{N-1} e^{-2\pi ikj/N} g_k \tag{1}$$

where i denotes the imaginary unit and $j = 0, 1, \ldots, N-1$. Because of

$$\sum_{k=0}^{N-1} e^{2\pi ik(j-\ell)/N} = N\delta_{j\ell} \tag{2}$$

we can proove

$$g_k = \frac{1}{N} \sum_{j=1}^{N-1} e^{2\pi ijk/N} G_j \ . \tag{3}$$

Only the finite discrete Fourier transform is of interest in computational physics since a continuum has to be approximated by an interval, and an interval by a finite set of representative points.

To be specific, we think about a time interval, $t_k = k\tau$ for $k = 0, 1, \ldots, N-1$. $f_j = j/N\tau$ are frequencies, and we may rewrite (1) and (2) into

$$G_j = G(f_j) = \sum_{k=0}^{N-1} e^{-2\pi if_j t_k} g_k \tag{4}$$

and

$$g_k = g(t_k) = \frac{1}{N} \sum_{j=0}^{N-1} e^{2\pi if_j t_k} G_j \ . \tag{5}$$

Let us simulate a very noisy cosine signal. We set

```
1   % this file is noisy_cos.m
2   fbar=50;
3   tau=0.001;
4   N=1024;
5   t=tau*[0:N-1];
6   R=2.0;
7   g=cos(2*pi*fbar*t)+R*randn(size(t));
8   plot(t,g,'.');
9   axis([min(t),max(t),-4,4]);
10  print -deps2 'ncos1.eps';
```

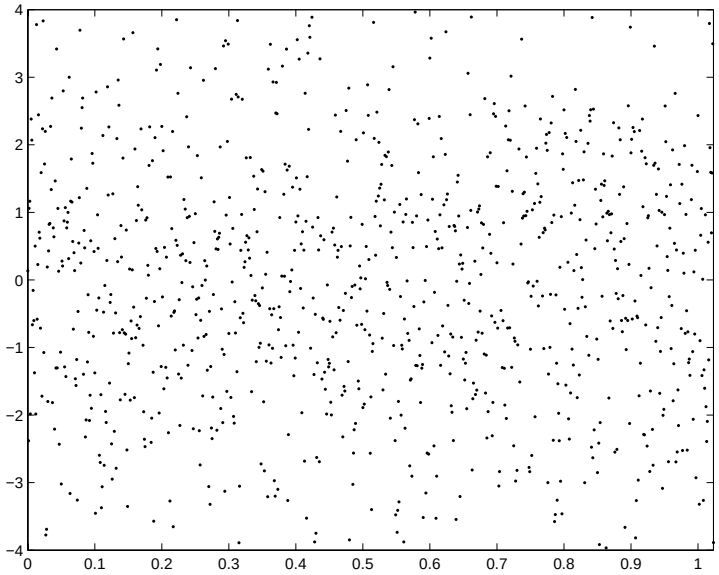

Figure 1: Noisy cosine (50 Hz) vs. time sampled at millisecond steps.

Would you recognize the cosine signal in Fig. 1?

We continue by implying the fast Fourier transform `fft`:

```
11  G=fft(g);
12  GG=fftshift(G);
13  f=[-N/2+1:N/2]/N/tau;
14  plot(f,abs(GG).^2);
15  print -deps2 'ncos2.eps';
```

In Fig. 2 we have plotted the spectral power $S = |G(f)|^2$. The prominent peaks at $f = \pm 50$ Hz are evident. The remaining spectral power is more or less constant which is an indicator for white noise. Note that we have shifted $G = G(f)$ such that the DC contribution ($f = 0$) becomes centered.

How can one extract the signal so convincingly from a lot of noise? Not by looking at the sampled data. They appear to be random. However, by performing a Fourier analysis, the different harmonic contributions are excited with their proper phases, so that they add up.

If you know that the signal is spoilt by white noise, you may remove it to a large extent. You might say

```
>> H=G.*(abs(G)>150);
>> h=ifft(H);
```

`ifft` denotes the inverse fast Fourier transform as described by (3).

You can do a lot more with the fast Fourier transform. Here are some examples:

- Differentiation: Fourier transform the signal and multiply by $2\pi if$, and back Fourier

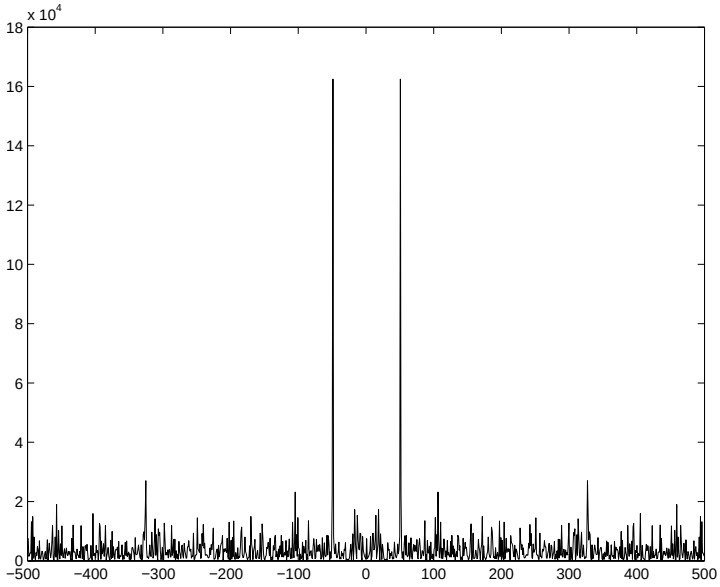

Figure 2: Spectral power of noisy cosine vs. frequency (Hz).

transform. Thereby a much better precision can be achieved than by difference quotients.

- Deconvolution: Often the output $b(t) = \int ds\, r(s)a(t-s)$ is the convolution of a signal a and a transmission function r. If the transmission function is known, then, from $B(f) = R(f)A(f)$, the signal can be reconstructed. Just divide the Fourier transformed output by the Fourier transformed transmission function, and back Fourier transform.
- Differential equations with constant coefficients: Differentiation operators become multiplication operators after Fourier transformation, and differential equations become algebraic equations.
- Data filtering and smoothing: remove the high frequency components.

The fast Fourier transform is an algorithm which takes into account that an operation on $2N$ data points are two operations on N data points. This reduces complexity from N^2 to $N\log(N)$ which makes all the difference.