Superoscillations

At the end of class this week, we talked a little bit about "superoscillations" and their potential use in radar for mine detection. Lets quickly recap what these are.

Let $f(t)$ denote some signal (*i.e.*, a real-valued, square-integrable function of one variable), and consider taking its Fourier transform. To make things easier, lets take $f : [0, 1] \to \mathbb{R}$ and enforce Dirichlet boundary conditions $f(0) = f(1) = 0$. In this case, we may represent f via a discrete Fourier series:

$$f(t) = \sum_{n=1}^{\infty} f_n \phi_n(t) \qquad ; \qquad \phi_n(t) := \sqrt{2}\sin(\omega_n t) \quad \text{where} \quad \omega_n = n\pi \tag{1}$$

We say that $f(t)$ is *bandlimited* if there is a maximum frequency component ω_N in $f(t)$. In other words, if $f(t)$ is bandlimited, then there is a $N > 1$ beyond which the Fourier coefficients f_n vanish:

$$f(t) = \sum_{n=1}^{N} f_n \phi_n(t) \tag{2}$$

The frequency ω_N is called the *bandlimit*.

The term "superoscillations" refers to the phenomenon whereby a bandlimited signal can locally oscillate (arbitrarily) faster than its bandlimit. To see how this arises, lets think of building functions using the Fourier basis. Suppose we are given N basis functions, $\{\phi_n(t)\}_{n=1}^{N}$. Then, we can build bandlimited functions by choosing the values of the coefficients f_n. Suppose we want to construct a function such that $f(t_i) = y_i$ for $i = 1, ..., N$. Using equation (2), we must have

$$y_i = \sum_{j=1}^{N} f_j \phi_j(t_i). \tag{3}$$

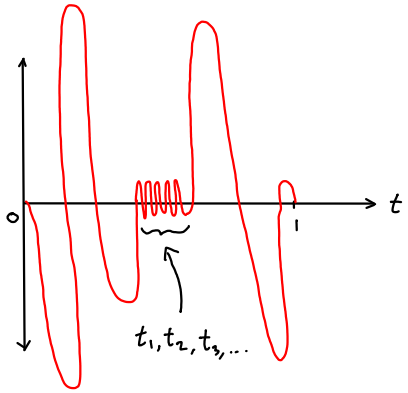
Notice, however, that this is just a matrix equation,

$$\mathbf{y} = \Phi \mathbf{f}, \tag{4}$$

where the components of Φ are $\Phi_{ij} = \phi_j(t_i)$. Hence, the requisite coefficients f_j are given by $\mathbf{f} = \Phi^{-1}\mathbf{y}$.

We are (almost) totally free to pick the N t_i's and y_i's. In particular, there's nothing stopping us from picking the t_i's as close together as we want and choosing, for example, $y_1 = -1$, $y_2 = +1$, $y_3 = -1$, $y_4 = +1$, etc. Therefore, a function that we construct in this way can be made to locally oscillate arbitrarily quickly. It's just linear algebra!

Of course, something has to give and there's a price to pay. As discussed in class, a superoscillatory signal will always have huge lobes whose size is exponentially larger than the amplitude of the superoscillations (see the figure below). Since the power of a signal goes like its amplitude squared, we see that the power required to generate superoscillations is exponential in their amplitude. (It turns out the power required is also exponential in the length of the superoscillating region.)



Problem: Suppose $f(t)$ is a superoscillatory signal, and consider taking a *windowed Fourier transform* around the superoscillations, *i.e.*, if the superoscillations begin at t_1 and end at t_2, take the Fourier transform of $B(t)f(t)$, where $B(t)$ is the box function

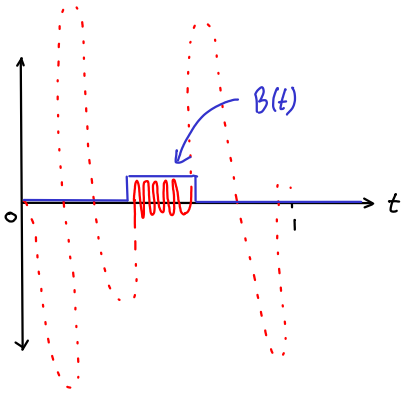
$$B(t) = \begin{cases} 1 & t_1 < t < t_2 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

(see the figure below). If the bandlimit of $f(t)$ is ω_N, one finds that the Fourier coefficients of $B(t)f(t)$ do not vanish for $n > N$. In other words, the superoscillations themselves are definitely high-frequency oscillations.

Why do we not see these high-frequency components in the Fourier transform of $f(t)$? In other words, how is it that they vanish when we remove the window? Where do they "go?"

Hint 1: What happens if you take the complementary windowed transform, i.e., *the Fourier transform of* $\bar{B}(t)f(t)$ *where* $\bar{B}(t) = 1$ *for* $0 \leq t \leq t_1$ *and* $t_2 \leq t \leq 1$ *and vanishes everywhere else?*

Hint 2: The Fourier transform is linear.



Bonus: Use the software package of your choice to generate an (intelligible) plot of a superoscillatory signal. My suggestion is to follow this write-up and to build a function on the interval $[0, 1]$ with Dirichlet boundary conditions. $N = 10$ is a good cutoff to use. Also, don't choose your points t_j symmetrically about the midpoint of the interval. A little bit of asymmetry will make your numerics more robust. This part is a bonus, since the main challenge that you'll be up against is machine precision, which is a computational mathematics problem.

Super Bonus: Convert your superoscillatory signal into an audio signal. I think MATLAB has a package that will let you do this. I really want to hear what superoscillations sound like.