Lab 04 Supplement: Filters In Action Solutions

EE 20 Spring 2014
University of California, Berkeley

1 Introduction

Below are the solutions to the supplement to Lab 4. Please attempt the exercises by yourself or with a group before reading on!

2 A More General Comb Filter

From the plots, we see that the number of peaks and notches is given by N, and their locations are

	$\alpha > 0$	$\alpha < 0$
Peaks	$\omega_P = 2\pi \cdot \frac{2k}{2N}$	$\omega_P = 2\pi \cdot \frac{2k+1}{2N}$
Notches	$\omega_N = 2\pi \cdot \frac{2k+1}{2N}$	$\omega_N = 2\pi \cdot \frac{2k}{2N}$

,

where $k \in \mathbb{Z}$. We also see that the magnitude of α determines the "steepness" of the filter and the gain at peak and notch frequencies:

$$|\alpha| \to 1 \Rightarrow |H(\omega_P)| \to 2, |H(\omega_N)| \to 0,$$
$$|\alpha| \to 0 \Rightarrow H(\omega) = 1, \forall\omega$$

We can also determine a more explicit relationship via the expression for the frequency response. Let's look at the case where $\alpha > 0$:

$$\begin{aligned} H(\omega_P) &= 1 + \alpha e^{-i2\pi\cdot\frac{2k}{2N}N} \\ &= 1 + \alpha e^{-i2\pi kN} \\ &= 1 + \alpha \end{aligned}$$

The last equality results from the fact that N is an integer, thus kN is an integer, and a complex exponential with integer multiple of 2π phase is equal to one. Similarly, we see that

$$\begin{aligned} H(\omega_N) &= 1 + \alpha e^{-i2\pi\cdot\frac{2k+1}{2N}N} \\ &= 1 + \alpha e^{-i2\pi k - i\pi} \\ &= 1 - \alpha, \end{aligned}$$

where we also used the fact that a complex exponential with integer multiple of π phase is equal to -1.

3 The Impulse Response & Convolution

3.1 Commutativity of Convolution

$$(x \star h)[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Let $m = n - k \Rightarrow k = n - m$, then summing over k from $-\infty$ to ∞ is the same as summing over m from ∞ to $-\infty$:

$$(x \star h)[n] = \sum_{m=\infty}^{-\infty} x[n-m]h[m]$$

Since the order of summation doesn't matter, we have

$$\begin{aligned}(x \star h)[n] &= \sum_{m=-\infty}^{\infty} h[m]x[n-m] \\ &= (h \star x)[n].\end{aligned}$$

To find the impulse response of the comb filter, we will use the definition of the impulse response, $x[n] = \delta[n] \Rightarrow y[n] = h[n]$. Letting $x[n] = \delta[n]$ in the comb filter LCCDE $y[n] = x[n] + \alpha x[n-N]$, we have

$$h[n] = \delta[n] + \alpha\delta[n-N].$$

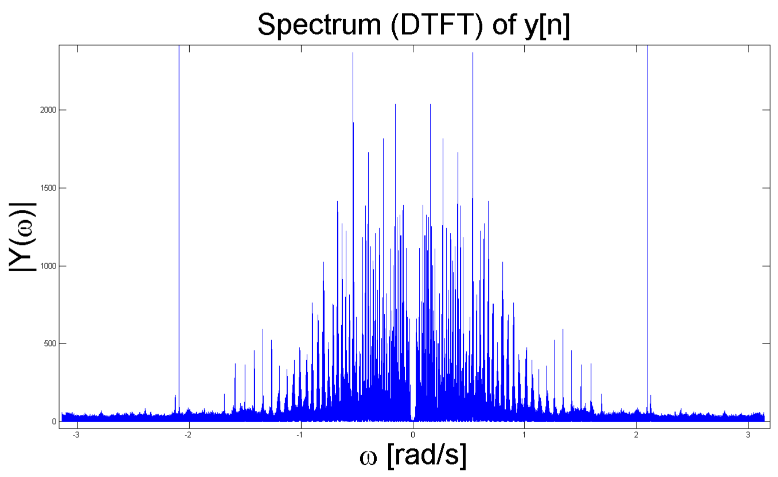
Qualitatively, this filter takes the input and adds to it a scaled & shifted version of itself, where the scaling factor is α and the shifting amount is N.

4 Noise

For MATLAB solutions, please refer to the script `solutions.m`, which can be found at `https://github.com/nosaesa/ee20`.

4.1 A Single Tone

With your newly gained knowledge of the Fourier Transform, let's take a look at the frequency content of $y[n]$:



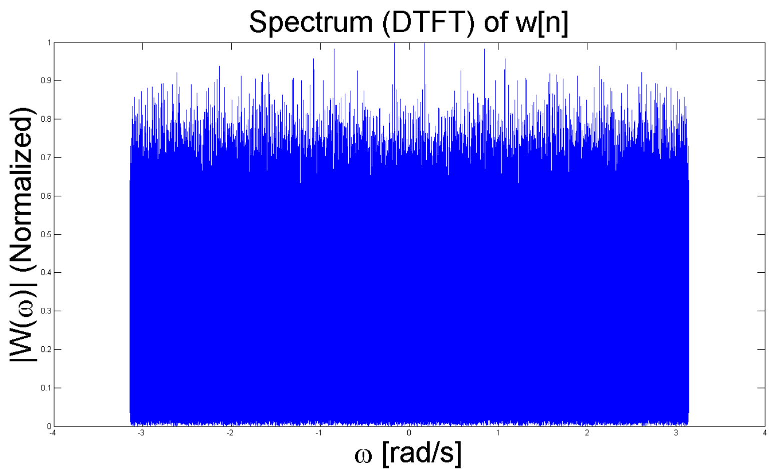
Note that, if you try this yourself, you'll have to zoom in quite far to see the signal – the noise is very overpowering! As discussed in the lab, we see that there is very low frequency content very close to $\omega = 0$ or above $\omega = \frac{\pi}{2}$, and thus we see how helpful the Fourier Transform is as a filter design tool! From this plot and the plots of the comb filter frequency response in the lab, we see that a comb filter with $N = 3$ and $\alpha = -1$ will notch out the noise while somewhat preserving the signal. This results in an impulse response of $h[n] = \delta[n] - \delta[n-3]$.

4.2 Is It Really That Easy?

Hopefully you are now convinced that, under the assumption that the noise is a single frequency which is not a large contributor to the signal we'd like to filter, it really is that easy! We can simply take the Fourier Transform of the signal to find the exact frequency of the offending noise and remove it. In fact, if we know the single noise frequency is the strongest frequency in the signal, we can remove the noise with no filtering at all – it's sort of a hack that's not reliable in the real world, but refer to `solutions.m` to check it out.

4.3 Random Noise

To convince yourself of the "hates-all-frequencies-equally" property of random noise, observe the spectrum of the random noise $w[n]$, which was generated with MATLAB's `wgn` function (requires the Communications System Toolbox, but documentation is available online):



From the plot, it certainly seems like the noise has higher amounts of some frequencies than others.. It is at this point that we must be a bit more rigorous when we talk about randomness: when we say the noise "hates all frequencies equally", we mean it is *all frequencies have the same probability of being present* in the noise. Again, if you're not comfortable with these concepts, don't worry about it. The study of noise in the frequency domain is quite advanced material, and you will not be expected to know anything pertaining to it in EE 20.