Music 421A
Spring 2019-2020
Homework #6
One-week assignment

Theory Problems

1. (5 pts) In the class overheads we have the derivation that the mean-square of a stationary stochastic process $v(n)$ is its variance plus its mean squared:

$$\begin{aligned}\hat{r}_v(0) &\stackrel{\Delta}{=} \mathcal{E}\{v^2(n)\} \stackrel{\Delta}{=} \mathcal{E}\{[\tilde{v}(n)+\mu_v]^2\} \\ &= \mathcal{E}\{\tilde{v}^2(n)\} + 2\mathcal{E}\{\tilde{v}(n)\}\mu_v + \mu_v^2 \\ &= \boxed{\sigma_v^2 + \mu_v^2}\end{aligned}$$

   where $v(n)$ is a real. Derive the corresponding formula for the case of complex $v(n)$.

2. (2 pts) Let $e(n)$ denote a sample of white noise. Then its power spectral density $S_e(\omega)$ is constant from $-\pi$ to π. By the stretch/repeat DTFT theorem, the signal $x = \textsc{Stretch}_2(e)$ also has a constant power spectral density, yet every other sample is constrained to be zero. Is $x(n)$ also white noise? Explain.

3. (8 pts) Find the (cyclic) unbiased autocorrelation of the following sequences in $\mathbf{R}^8$:

$$\begin{array}{llll}(a) & [1,0,0,0,0,0,0,0] & (b) & [1,1,0,0,0,0,0,0] \\ (c) & [1,1,1,1,0,0,0,0] & (d) & [1,1,1,1,1,1,1,1]\end{array}$$

4. (5 pts) Find the variance σ_y^2 of the three-point moving-average of a unit-variance white noise sequence $e(n)$: $y(n) = [e(n) + e(n-1) + e(n-2)]/3$.

5. (5 pts) Consider estimating the Power Spectral Density (PSD) of a noise sample $x(n)$ using Welch's method with a Hann window. Write down the formula for the sample PSD, including a formula for the normalizing constant needed due to using a Hann window in place of the rectangular window.

6. (5 pts) Show that *pink noise* has the same average power in any octave. [Hint: "average power" is the same thing as "sample variance". Recall that the sample variance within a frequency band is obtained by integrating the sample power spectral density across that band.]

Lab Assignment

1. (28 pts) **Resynthesizing colored noise with an LTI filter**
   Colored noise can be modeled as a filtered white noise. Here we estimate the filter given a recording of colored noise so that we can resynthesize the noise by passing white noise through the filter.

   (a) (10 pts) Download the sound file `airplane_noise.wav`[1] and compute and plot the power spectral density using Welch's method. Set the window length to $M = 2048$. In your work, do not use functions in Matlab, such as `pwelch` or `periodogram` (but you may refer to them).

   (b) Design a filter that, when driven by unit-variance white noise, will output, in steady state, a colored noise having substantially the same PSD as that of `airplane_noise.wav`. The filter may be FIR or IIR, but let the number of multiplies per sample be 9 (that is, one could design a length 9 FIR filter, 8th order allpole filter, or four-pole, four-zero IIR filter, for example). Use whatever filter-design method you believe to be best for this problem. Plot an overlay of

      i. the PSD of `airplane_noise.wav` computed in part (a) above,

      ii. (5 pts) the magnitude-frequency response of your filter, and

      iii. (5 pts) the PSD of the steady-state filter output when driven by white noise (either in one overlay or two separate overlays, as preferred). Include your matlab listing(s).

      iv. (5 pts) Discuss the relative merits and shortcomings of the FIR filter design method(s) you tried.

      v. (3 pts) Discuss any understood particularly large deviations. For the 1/f PSD example discussed in class, one might say the following about the simple third-order filter used in that example:

         > Below 20 Hz the response deviates from desired because our specified band-pass only went down to 20 Hz. Below 20 Hz is a "don't care" band, and we simply want it to gracefully approach a flat response near dc. We could also have imposed a zero at dc, which is nice for removing any nonzero mean in the noise, but this would have placed high numerical stress on a 1/f roll-off starting at 20 Hz. Instead we can use a series dc blocker as usual when we want to make sure the dc component is kept to zero. Above around 18 kHz, the response flattens out, deviating strongly from the 1/f roll-off. This is again by design because we don't hear much above that frequency, so the filter design "let's go" around there and approaches its natural asymptote of a constant. From a Bode plot perspective, both the dc and infinite-frequency limits should be constants, because we have an equal number of poles and zeros in the design. We also see a noticeable ripple in the roll-off between 20 Hz and 18 kHz. Since it is only on the order of a dB, it is likely not to be distinguishable audibly from a more accurate design. More accuracy can be achieved by using more poles and zeros to produce more ripples at smaller amplitudes, as shown in the given paper on exponentially distributed real pole-zero pairs used for this purpose.

[1] `http://ccrma.stanford.edu/~jos/sasp/hw/airplane_noise.wav`

General Hints and Guidelines:

The amplitude response of the desired filter is given by the square-root of the power spectral density. To fully specify the filter, however, we also need a phase response. The simplest solution is to define the phase response as zero, corresponding to a "zero-phase" impulse response. A zero-phase FIR filter can be designed by the *window method* as discussed in class. If you choose this method, report some measure of estimated time aliasing in the desired impulse response prior to applying the window.

A more typical choice in practice, especially when working with analog filters, is to compute the *minimum-phase* phase response from the desired magnitude response. It can be shown that the minimum-phase spectrum corresponding to any magnitude spectrum is given by the *Hilbert transform* of the log-magnitude spectrum.[2] The Matlab `hilbert` function can be used here because the spectrum of any discrete-time signal is periodic.

When the desired frequency response is minimum phase, we can estimate the filter coefficients using `invfreqz` in Matlab/Octave, among others. (Say `help filterdesign` in Matlab for more suggestions.) Note that you can specify a weighting factor in `invfreqz` to give, e.g., more weight to low frequencies than high frequencies (as is typically done in audio filter design).

[2] `http://ccrma.stanford.edu/~jos/filters/Hilbert_Transform_Relations.html`