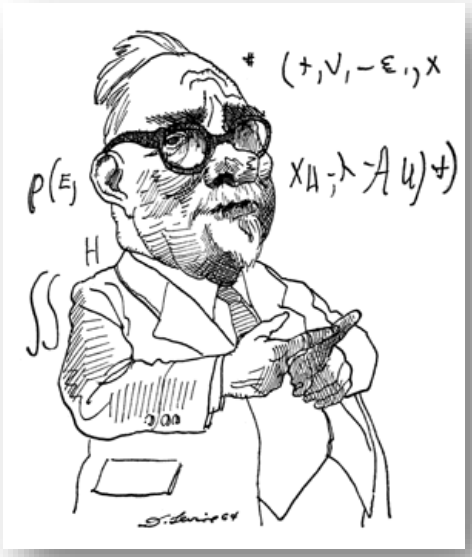
Introduction to Audio-visual Processing

Lab 4: Wiener Filter

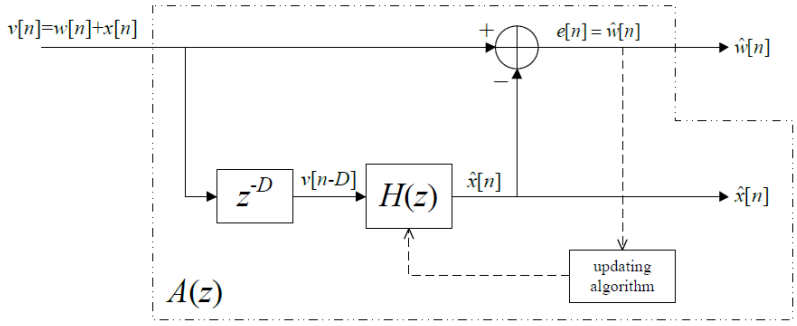


Norbert Wiener; drawing by David Levine

Course 20-21/Q2

1. Wiener filtering to eliminate a narrowband interference

In this first part, we will implement a Wiener filter to eliminate a sinusoidal interference x[n] that corrupts a desired speech signal w[n] using next scheme that allows to separate two additive and uncorrelated components with different power spectral density widths.



If the sinusoid were deterministic, or at least the frequency known, a very narrow band pas filter centered at the sinusoid frequency would separate the desired signal from the interference, cancelling it. However, assuming that the frequency of the sinusoid is not known, this solution cannot be implemented. Alternatively, a Wiener filter will be used. Furthermore, if we also assume that the frequency of the sinusoid can be time varying, the problem will not be stationary and a solution that would track the frequency variations of the interference will be required. In this case, an adaptive filter, implemented by means of an LMS algorithm, will allow to autonomously search and cancel the sinusoid.

1.1. Wiener filter implementation when the autocorrelation function is known

For academic purposes we will first evaluate the behavior of the system assuming that w[n] is a zero mean white Gaussian process with variance σ_w^2 and x[n] a sinusoid whose amplitude $\sqrt{2P}$ and frequency Ω_0 are known, and whose phase θ is a (0,2π) uniformly distributed random variable.

$$\mathrm{v[n]} = \mathrm{x[n]} + \mathrm{w[n]} = \sqrt{2P}\cos(2\pi \mathrm{F}_0 n + \theta) + w[n]$$

In this case the autocorrelation function for v[n] is:

$$R_v[m] = P\ \cos(2\pi \mathrm{F}_0 m) + \sigma_w^2 \cdot \delta[m]$$

i) Review the MATLAB code. Set the number of coefficients of the Winer filter to Q=10 and the frequency of the interference to $F_0 = 1/16$. The rest of parameters are D=1, $\sigma_w^2 = 1$ and $P = 0.5$. Check that the autocorrelation matrix R_x and the crosscorrelation vector r_{xd} are calculated according to the theory results seen in class.

ii) Calculate the impulse response of the Wiener filter $h_{opt}[n]$ and the impulse response of the global filter A(z), which is the filter encompassed within the dashed line in the figure: $h_a[n] = \delta[n] - h_{opt}[n - D]$.

iii) Plot the frequency response of the Wiener filter and discuss if you have obtained the expected result.

iv) Plot the frequency response of the global filter and discuss if you have obtained the expected result.

v) Plot the interfered signal v[n], the sinusoidal interference x[n], and the signals at the output of the Wiener filter and the global filter. Comment the results.

vi) Change the number of coefficients of the Wiener filter and repeat the design. Comment the results.

1.2. Wiener filter implementation with the LMS algorithm

In real applications the autocorrelation matrix R_x and the crosscorrelation vector r_{xd} are not known. Moreover, in most cases non-stationary signals are present (for example speech signals or interferences that change in time). In those cases, an adaptive filter is required to identify and track the tie varying correlation signals. In this practice we will implement an adaptive filter based on LMS algorithm.

1.2.1. The LMS algorithm and its convergence analysis

First, we will continue with the academic exercise of a zero mean white Gaussian process interfered by a sinusoid. Review the MATLAB code and check the simplicity of the LMS algorithm.

i) Execute the code.

- Figure 11 plots the evolution of the two first coefficients vs. the number of iterations. Check the convergence of those coefficients.
- Figures 13 and 14 plot the frequency responses of the Wiener filter and the global filter, respectively, after 3N/4 iterations (it is assumed that the convergence has been achieved at that time instant). Check that you obtain similar results to those obtained in the previous section.

ii) The MATLAB code calculates and reports a *reference* mu constant calculated as:

$$\mu = \frac{2}{Q \cdot \hat{r}_v[0]}$$

Using this reference value, play with the μ constant to check the convergence, the speed convergence, and the misadjustement. Experiment with values of μ close to upper bound that guarantees the convergence (in average) and much lower than this limit.

1.2.2. Filtering a speech signal

Let us analyze a more realistic situation in which the interfered signal is a speech signal. Using `Audio_Read()` function load the `'Force.mp3'` file that will substitute the white Gaussian process w[n].

i) Execute the MATLAB code to design the Wiener filter implementing the LMS algorithm, plot the filter frequency responses and the different signal waveforms. You can also listen the different signals using the MATLAB function `soundsc(signal,Fs)` where `Fs` is the sampling frequency (Fs=44100 for the audio file `'Force.mp3).` Appreciate how the interference is eliminated.

ii) Analyze the frequency responses of the filters and justify its behavior.

iii) Uncomment the lines that suddenly modify the frequency of the sinusoid after N/2 iterations (into the loop of the LMS algorithm) or generate a chirp as interference and check how the filter autonomously adapts to the changes.

iv) For D=1 you can see that there will be no signal at the output (if you listen `westimate` signal you can see that is highly attenuated), $\widehat{w}[n] = 0$. How can you explain this behavior?