Linear Regression and Wiener Filtering

1. Introduction

The topic of Wiener filtering sets the stage for the topic of Kalman filtering; the latter being simply real time implementation of the former. To arrive at this point required at least a minimal understanding of the following topics:

T1: Random Variables ; T2: *wss* Random Processes ; T3: Dynamical Systems; T4: Signal Processing.

Each one of these topics is covered in at least one entire course; typically within an electrical engineering curriculum. Yet, here we are, in the 8th week of a single course. And the student is expected to have a basic understanding of all four of these topics. Some students in the class may have had a course related to T1. Others may have had a course related to T3. The catalogue prerequisites for this course include *either* T1 *or* T3. Why is this? Well, simply, to require both T1 and T3 would likely not garner a sufficient number of qualified students to run the course.

The point of the above discussion is to encourage you, the student, to rest assured that your cohorts are probably as uncomfortable as you are with your lack of a firm grasp of any of the above topics. [There's an old saying: Missouri (misery) loves company. That's why god created Kansas! ☺] If it seems as though topics have not received due attention, and have been covered in an incomplete, and even chaotic fashion, it's because that is the case. Yes, one could always try to lessen the chaos. However, given the fact that all four topics must be addressed within eight weeks, along with the fact that they all must be connected to one another, it is only natural to expect some amount of chaos.

In this lecture we will attempt to connect the dots, so to speak. The medium for this effort is Wiener filtering. We will begin this effort by revisiting the concept of linear models in relation to random variables. We will then couch this model in the framework of a random process. This will then be followed by bringing a model for a dynamical system into the picture.

2. The Linear Model Problem

Let $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^{tr}$ and $\mathbf{y} = [y_1 \quad y_2 \quad \cdots \quad y_n]^{tr}$ be two *n*-dimensional random variables, and consider the linear model:

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{x} + \mathbf{b} \tag{1}$$

We desire to find the expression for **A** and **b** that minimize the *mean-squared error* $mse = E\|\mathbf{y} - \hat{\mathbf{y}}\|^2$.

Fact 1: The value for **A** will satisfy the *orthogonality condition* (i) $Cov[(\mathbf{y} - \hat{\mathbf{y}})\mathbf{x}^{tr}] = \mathbf{0}$ (i). The value for **b** will satisfy the *unbiased condition* (ii) $E(\mathbf{y}) = E(\hat{\mathbf{y}})$. We will now use these facts to obtain the explicit expressions for **A** and **b**.

$$Cov[(\mathbf{y} - \hat{\mathbf{y}})\mathbf{x}^{tr}] = \mathbf{0} \quad \Rightarrow \quad Cov(\mathbf{y}\mathbf{x}^{tr}) = Cov(\hat{\mathbf{y}}\mathbf{x}^{tr}) = Cov[(\mathbf{A}\mathbf{x} + \mathbf{b})\mathbf{x}^{tr}] = \mathbf{A}Cov(\mathbf{x}\mathbf{x}^{tr}) \quad \Rightarrow \quad \mathbf{A} = Cov(\mathbf{x}\mathbf{x}^{tr})^{-1}Cov(\mathbf{y}\mathbf{x}^{tr}) . \tag{2a}$$

$$E(\mathbf{y}) = E(\hat{\mathbf{y}}) = E(\mathbf{A}\mathbf{x} + \mathbf{b}) + \mathbf{A}E(\mathbf{x}) + \mathbf{b} \quad \Rightarrow \quad \mathbf{b} = E(\mathbf{y}) - \mathbf{A}E(\mathbf{x}) . \tag{2b}$$

Equations (2) are usually expressed in the notational form: $\mathbf{A} = \boldsymbol{\Sigma}_{\mathbf{xx}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}}$ and $\mathbf{b} = \boldsymbol{\mu}_{\mathbf{y}} - \mathbf{A}\boldsymbol{\mu}_{\mathbf{x}}$. (2c)

3. Linear Modeling in the Context of Random Processes

Now suppose that **x** and **y** are segments of a jointly *wide sense stationary* (*wss*) 2-dimensional random process $\{[x_t \quad y_t]^{tr}\}_{t=-\infty}^{\infty}$. For convenience, we will assume that $\mu_x = \mu_y = 0$. Then

$$\mathbf{\Sigma}_{xx} = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) & \cdots & R_{xx}(n-1) \\ R_{xx}(1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & R_{xx}(1) \\ R_{xx}(n-1) & \cdots & R_{xx}(1) & R_{xx}(0) \end{bmatrix} \stackrel{\Delta}{=} \mathbf{R}_{xx} \quad \text{and} \quad \mathbf{\Sigma}_{yx} = \begin{bmatrix} R_{yx}(0) & R_{yx}(1) & \cdots & R_{yx}(n-1) \\ R_{yx}(-1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & R_{yx}(1) \\ R_{yx}(-n+1) & \cdots & R_{yx}(-1) & R_{yx}(0) \end{bmatrix} \stackrel{\Delta}{=} \mathbf{R}_{yx}. \quad (3)$$

In this setting, it follows that the optimal predictor is: $\hat{\mathbf{y}} = \mathbf{A}\mathbf{x}$ where $\mathbf{A} = \mathbf{R}_{xx}^{-1}\mathbf{R}_{yx}$. (4)

At this point, note the similarity of (4) with (2c). Since we implicitly assume that all wss process have zero mean, then the equation $\mathbf{b} = \boldsymbol{\mu}_{\mathbf{y}} - \mathbf{A}\boldsymbol{\mu}_{\mathbf{x}}$ in (2c) becomes moot (i.e. it is no longer relevant). Now, recall that $E(XY) = Cov(X,Y) + \mu_X \mu_Y$. Since the means are zero, we have $E(XY) = Cov(X,Y)$. Hence, (4) is simply a special case of (2c), where then means are zero, and the indices associated with the random variables are time indices.

3. The Signal-Plus-Noise Problem

Now, let's assume: $\mathbf{x} = \mathbf{s} + \mathbf{n}$ (5)

where **s** and **n** are mutually independent signal and noise processes. In this setting we are attempting to predict **y=s** using **x**. Hence, in this notation, (4) becomes:

$$\hat{\mathbf{s}} = \mathbf{A}\mathbf{x} \text{ where } \mathbf{A} = \mathbf{R}_x^{-1}\mathbf{R}_s = \mathbf{R}_{(s+n)}^{-1}\mathbf{R}_s. \quad (6)$$

Even though (6) is optimal (i.e. *mmse*) it does not offer much insight into how the structure of the signal and noise influence the accuracy of the estimate. For this reason, we will work in the frequency domain. But first, we will delve a bit into the eigenstructure of (6). Recall the following:

FACT 1: Let **Q** be a symmetric matrix. Then $\mathbf{Q} = \mathbf{V}_\mathbf{Q}\mathbf{\Lambda}_\mathbf{Q}\mathbf{V}_\mathbf{Q}^*$ where $\lambda_k^{(Q)}$ is the k^{th} real eigenvalue of $\mathbf{\Lambda}_\mathbf{Q} = diag\{\lambda_1^{(Q)} \quad \lambda_2^{(Q)} \quad \cdots \quad \lambda_n^{(Q)}\}$ and where $\mathbf{v}_k^{(Q)}$ is the k^{th} eigenvector of $\mathbf{V}_Q = [\mathbf{v}_1^{(Q)} \quad \mathbf{v}_2^{(Q)} \quad \cdots \quad \mathbf{v}_n^{(Q)}]$. Moreover, $\mathbf{V}_\mathbf{Q}\mathbf{V}_\mathbf{Q}^* = \mathbf{V}_\mathbf{Q}^*\mathbf{V}_\mathbf{Q} = \mathbf{I}$. (In words, the eigenvectors are orthonormal to one another.) Finally, we also have $\mathbf{Q}^{-1} = \mathbf{V}_\mathbf{Q}\mathbf{\Lambda}_\mathbf{Q}^{-1}\mathbf{V}_\mathbf{Q}^*$.

[If you haven't noticed, we are also bringing the topic of matrix theory into play. ☹]

In order to apply this fact to (6), write $\mathbf{R}_s = \mathbf{V}_s\mathbf{\Lambda}_s\mathbf{V}_s^*$ and $\mathbf{R}_x = \mathbf{V}_x\mathbf{\Lambda}_x\mathbf{V}_x^*$. Then (6) becomes:

$$\hat{\mathbf{s}} = \mathbf{A}\mathbf{x} \text{ where } \mathbf{A} = \mathbf{V}_x\mathbf{\Lambda}_x^{-1}\mathbf{V}_x^*\mathbf{V}_s\mathbf{\Lambda}_s\mathbf{V}_s^*. \quad (7)$$

Now, in general, (7) does not appear to offer any further insight than does (6). In an effort to pursue such insight, consider the following example.

***Example* 1.** Suppose that the noise is a white noise process with power σ_n^2. Then $\mathbf{R}_n = \sigma_n^2\mathbf{I}$, and so

$\mathbf{R}_x = \mathbf{R}_s + \mathbf{R}_n = \mathbf{R}_s + \sigma_n^2\mathbf{I}$. In this case: $\mathbf{R}_x = \mathbf{V}_x\mathbf{\Lambda}_x\mathbf{V}_x^* = \mathbf{V}_s(\mathbf{\Lambda}_s + \sigma_n^2\mathbf{I})\mathbf{V}_s^*$. Hence, (7) becomes:

$$\hat{s} = \mathbf{A}\mathbf{x} \text{ where } \mathbf{A} = \mathbf{V}_s(\mathbf{\Lambda}_s + \sigma_n^2\mathbf{I})^{-1}\mathbf{\Lambda}_s\mathbf{V}_s^* = \mathbf{V}_s\mathbf{\Lambda}_r\mathbf{V}_s^* \text{ and where } \lambda_k^{(r)} = \lambda_k^{(s)}/[\lambda_k^{(s)} + \sigma_n^2]. \quad (8)$$

Now write $\mathbf{x} = \mathbf{V}_x\mathbf{V}_x^*\mathbf{x}$. Substituting this into (8) gives:

$$\hat{\mathbf{s}} = \mathbf{A}\mathbf{x} = \mathbf{V}_s(\mathbf{\Lambda}_r\mathbf{V}_s^*x) = \mathbf{V}_s(\mathbf{\Lambda}_r\mathbf{X}) \qquad \text{where } \mathbf{X} = \mathbf{V}_s^*\mathbf{x}. \quad (9a)$$

The form (9) offers insight as to the nature of the estimator $\hat{s}$. Write (9a) as:

$$\hat{\mathbf{S}} \overset{\Delta}{=} \mathbf{V}_s^*\hat{s} = \mathbf{\Lambda}_r(\mathbf{V}_s^*x) \overset{\Delta}{=} \mathbf{\Lambda}_r\mathbf{X}. \quad (9b)$$

In the transformed domain (using the transformation matrix $\mathbf{V}_s^*$), equation (10) shows that the kth element of $\hat{\mathbf{S}}$ is:

$$\hat{S}_k = \lambda_k^{(r)}X_k = \frac{\lambda_k^{(s)}}{\lambda_k^{(s)} + \sigma_n^2}\,X_k. \quad (9c)$$

For a value of k such that $\lambda_k^{(s)} >> \sigma_n^2$ [i.e. a high signal-to-noise ratio (SNR)], we have $\hat{S}_k \cong X_k$. At a value of k such that $\lambda_k^{(s)} << \sigma_n^2$ (i.e. a low SNR), we have $\hat{S}_k \cong 0$. More generally, (9c) states that $\hat{S}_k = SNR_kX_k$. Hence, with a little thought, it should become clear that (9c) is intuitively appealing.

In fact, as we will now show, (9c) is a *Wiener filter*, with the exception that $\mathbf{V}_s^*$ is not the DFT matrix $\mathbf{F}^*$. The question of just how does $\mathbf{V}_s^*$ relate, if at all, to $\mathbf{F}^*$, is a bit complicated to address in this set of notes. All I will say here is that:

As the matrix dimension $n \to \infty$, $\mathbf{V}_s^*$ gets 'close and closer' to $\mathbf{F}^*$, where the term 'close' is used in a type of 'weak' sense. In a similar sense, the eigenvalues of $\mathbf{R}_x = \mathbf{V}_x\mathbf{\Lambda}_x\mathbf{V}_x^*$ get closer and closer to the *psd* values for **x**.

[NOTE: I currently (as of 10/14/19) have a manuscript that quantifies 'closeness' in the above statements being reviewed by the journal *Linear Algebra and Its Applications*. I mention this because as of this date, the term 'closeness' has been vague, at the very least.]

This question is much more amenable to an investigation using simulations. The following problem is very approachable in relation to this course:

Problem: Given a segment $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^{tr}$ of a *wss* random process, where $\mathbf{x} = \mathbf{s} + \mathbf{n}$, and where the *psd*s of the signal and noise processes are known, how suboptimal is the Wiener filter solution to the *mmse* solution, as a function of the dimension, *n*.

Sub-Problem 1: How suboptimal is it in the case where the noise is white and the signal is AR(1)?

Linear Regression in the Frequency Domain in the context of a Linear System

For $\Delta = 2\pi/n$ define the *Fourier transform* matrix:

$$\mathbf{F} \overset{\Delta}{=} \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-i\Delta} & \cdots & e^{-i(n-1)\Delta} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & e^{-i(n-1)\Delta} & \cdots & e^{-i(n-1)^2\Delta} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \cdots & \mathbf{f}_n \end{bmatrix}. \tag{10}$$

Note that the $1/\sqrt{n}$ factor in (10) was chosen under the assumption that the processes involved are *regular wss* random processes.

FACT 2: $\mathbf{F}^*\mathbf{F} = \mathbf{F}\mathbf{F}^* = \mathbf{I}$.

In words, Fact 2 means that **F** is a *unitary* matrix. Let $\mathbf{w} = \mathbf{F}\mathbf{v}$. Then $\|\mathbf{w}\|^2 = \mathbf{w}^*\mathbf{w} = \mathbf{(Fv)}^*\mathbf{(Fv)} = \mathbf{v}^*\mathbf{F}^*\mathbf{F}\mathbf{v} = \mathbf{v}^*\mathbf{v} = \|\mathbf{v}\|^2$. What this means is that a unitary transformation is 'size-preserving'. The 'size' of **w**, as measured by $\|\mathbf{w}\|^2$, is the same as the 'size' of **v**. If we take 'size' to be the energy contained in a process, then this says that we can sum up the energy in time or in frequency. In either case, we get the same energy. Computationally, (10) is almost the 'fft' command in *Matlab.* Note, however, that because we included the $1/\sqrt{n}$ factor, it is not exactly the same. In *Matlab* there is no such factor. Instead, a factor of $1/n$ is included in the 'ifft' command.

The Wiener filter solution, analogous to (9) is:

$$\hat{\hat{s}} = \mathbf{F(HX)} \qquad \text{where} \quad \mathbf{X} = \mathbf{F}^*\mathbf{x} \text{ and } \mathbf{H} = diag\{h_0, \cdots, h_{n-1}\} \text{ with } h_k = \frac{S_s(\omega_k)}{S_s(\omega_k) + S_n(\omega_k)}. \tag{11}$$

A comparison between (11) and (9c) is worth noting. In fact, as the matrix size $n \to \infty$ the eigenvalues in (9c) do, indeed, converge In distribution to the *psd* elements in (11). Hence, for large *n*, the key difference between (11) and (9c) is that (11) uses the DFT matrix, while (9c) uses the eigenvector matrix.

4. Using Matlab to Construct a Linear Model

We will address this in relation to the 3-D random variable, (X_1, X_2, Y), only because this setting is simple, and yet readily illustrates the general construction. Denote the data associated with (X_1, X_2, Y) as $(\mathbf{X_1}, \mathbf{X_2}, \mathbf{Y})$, as defined above (1.2). To estimate (μ_1, μ_2, μ_Y), we use the command:

$$m12y = mean(\mathbf{X_1}, \mathbf{X_2}, \mathbf{Y}).$$

This command will result in $(\hat{\mu}_1, \hat{\mu}_2, \hat{\mu}_Y)$. To arrive at the covariance estimates, we use the command:

$$c12y = \text{cov}(\mathbf{X_1}, \mathbf{X_2}, \mathbf{Y}).$$

This command will result in

$$\begin{pmatrix} \hat{\sigma}_1^2 & \hat{\sigma}_{12} & \hat{\sigma}_{1Y} \\ \hat{\sigma}_{12} & \hat{\sigma}_1^2 & \hat{\sigma}_{2Y} \\ \hat{\sigma}_{1Y} & \hat{\sigma}_{2Y} & \hat{\sigma}_Y^2 \end{pmatrix}$$

Let the 2×2 matrix shaded in green be denoted as $\widehat{\Sigma}_{XX}$, and let the 2×1 matrix shaded in blue be denoted as $\widehat{\Sigma}_{XY}$ Then (1.8e) can be written as:

$$\widehat{\Sigma}_{XX}\begin{pmatrix}\widehat{\beta}_1\\ \widehat{\beta}_2\end{pmatrix}=\widehat{\Sigma}_{XY} \qquad \Rightarrow \qquad \begin{pmatrix}\widehat{\beta}_1\\ \widehat{\beta}_2\end{pmatrix}=\widehat{\Sigma}_{XX}^{-1}\ \widehat{\Sigma}_{XY}\,.$$

The *Matlab* command for obtaining $\widehat{\Sigma}_{XX}$ is c12 = c12y(1:2,1:2), and the command for $\widehat{\Sigma}_{XY}$ is cxy=c12y(1:2,3). Hence, the *Matlab* command for arriving at $(\widehat{\beta}_1\,,\,\widehat{\beta}_2)$ is: b12= c12^-1 * cxy. Once we have $(\widehat{\beta}_1\,,\,\widehat{\beta}_2)$, the *Matlab* command to obtain $\widehat{\beta}_0$ is: b0 = m12y(3) – b12' * m12y(1:2)'.

WARNING: This setting addresses covariances. However, for *wss* random processes that are known a priori to be zero-mean processes, the use of sample covariances, as opposed to sample expected values, can cause numerical problems (e.g. negative *psd* estimates). So, the question then becomes: How can the above covariance method be used in relation to correlations? To answer this question, write $R(\tau)=E(X_tX_{t+\tau})=Cov(X_t,X_{t+\tau})+E(X_t)E(X_{t+\tau})$. Hence, if we know a priori that the mean is zero, then $R(\tau)=E(X_tX_{t+\tau})=Cov(X_t,X_{t+\tau})$. The problem with using Matlab's 'cov' command is that it automatically subtracts the sample mean. So the moral of the story here is: Do not use the 'cov' command. Rather, use the 'xcorr' command to obtain estimates of the correlations, and then use the 'toeplitz' command to obtain the associated matrices. Although, keep in mind that if one uses the 'unbiased' flag in the 'xcorr' command, then it is possible that the estimated *psd* values could become negative. ☹

Example. Consider the following difference equation models for a signal process s_k and noise process n_k:

$s_k=0.9\,s_{k-1}+u_k$; $n_k=-0.5\,n_{k-1}+v_k$, where u_k and v_k are white noise processes. Assume that the white noise variances are such that the signal and noise variances are each equal to one. This is the situation wherein the total signal-to-noise ratio (SNR) is one, or 0 dB.

Remark: Since the above are discrete-time processes, they do not have Laplace transforms. Rather, they have z-transforms. Specifically: $Z(s_k)=Z\{0.9\,s_{k-1}+u_k\}\Rightarrow S(z)=0.9z^{-1}S(z)+U(z)\Rightarrow H(z)=\dfrac{S(z)}{U(z)}=\dfrac{1}{1-0.9z^{-1}}$. This system transfer function has a poll at $z_1=0.9$. If we assume that this discrete-time process was obtained by sampling a continuous-time process using a sampling period $\Delta=1\text{sec.}$, then the relation between the Laplace variable *s* and the Z-variable *z* is: $z=e^{s\Delta}$. In this case, the discrete-time and continuous-time system poles relate as: $z_1=e^{s_1\Delta}$. Writing $s_1=\sigma_1+i\omega_1$ gives $z_1=e^{(\sigma_1+i\omega_1)\Delta}=e^{\sigma_1\Delta}e^{i\omega_1\Delta}$. So a LHP continuous-time pole corresponds to a discrete-time pole inside the unit circle. In the case of this signal, the discrete-time pole 0.9 is, indeed, in the unit circle, and so this corresponds to a stable system. The corresponding continuous-time poles is: $s_1=\ln(z_1)/\Delta=\ln(0.9)=-0.1054$. This corresponds to a first order transfer function with -3dB bandwidth 0.1054 rad/sec. This is also called a *lowpass* filter. On the other hand, the noise process has $H(z)=\dfrac{N(z)}{U(z)}=\dfrac{1}{1+0.5z^{-1}}$. The pole is -0.5. Were we to desire to recover the associated continuous-time

pole, we would have $s_1 = \ln(z_1)/\Delta = \ln(-0.5) = -0.6931 + 3.1416i$. Notice that the pole is in the LHP, but that it has an imaginary part equal to π. However, since the sampling interval $\Delta = 1\text{sec.}$, the sampling frequency $\omega_s = 2\pi$, and the Nyquist frequency is $\omega_N = \pi$. The problem here is that, if one were to compute or plot the FRF associated with $H(z = e^{i\omega\Delta}) = \dfrac{1}{1+0.5e^{-i\omega\Delta}}$, one would discover that this is a *highpass* filter. As such, one needs to beware of casually mapping the discrete-time pole back to a continuous-time pole, since aliasing becomes a major complicating issue.

(a) Compute the numerical values for σ_u^2 and σ_w^2

<u>*Solution*</u>: For an AR(1) process $y_k \Rightarrow \sigma_w^2 = (1-\alpha^2)\,\sigma_y^2$. So, since $\sigma_s^2 = \sigma_n^2 = 1$:

$$\boxed{\sigma_u^2 = (1-0.9^2)1 = 0.19} \;\&\; \boxed{\sigma_v^2 = [1-(-0.5)^2]1 = 0.75}$$

(b) Use the Wiener-Kinchin Theorem to obtain an explicit expression for the *psd* of the AR(1) process:

$$y_k = \alpha\, y_{k-1} + u_k\,. \quad (*)$$

<u>*Solution*</u>: Recall that $Y(i\omega) = \lim_{T/2\to\infty} \dfrac{1}{\sqrt{T}} \int_{-T/2}^{T/2} y(t)e^{-i\omega t}dt \cong \lim_{n\to\infty} \dfrac{1}{\sqrt{(n+1)\Delta}} \sum_{k=-n/2}^{n/2} y_k e^{-i\omega(k\Delta)}\Delta$ where $T = n\Delta$.

Hence, the Fourier transform of $v_k = y_{k-1}$ is:

$$V(i\omega) \cong \lim_{n\to\infty} \frac{1}{\sqrt{(n+1)\Delta}} \sum_{k=-n/2}^{n/2} v_k e^{-i\omega(k\Delta)}\Delta = \lim_{n\to\infty} \frac{e^{-i\omega\Delta}}{\sqrt{(n+1)\Delta}} \sum_{k=-n/2}^{n/2} y_{k-1} e^{-i\omega[(k-1)\Delta]}\Delta = e^{-i\omega\Delta} Y(i\omega)\,.$$

Hence, the FT of (*) results in: $Y(i\omega) = \dfrac{U(i\omega)}{1-\alpha e^{-i\omega}}$ for $\Delta = 1\text{sec}$. [c.f. also the above Remark.]

Hence, from the WK Theorem:

$$S_y(\omega) = E[|Y(i\omega)|^2] = \frac{E[|U(i\omega)|^2]}{|1-\alpha e^{-i\omega}|^2} = \frac{\sigma_u^2}{1+\alpha^2-2\alpha\cos(\omega)} \quad \text{for } |\omega| < \omega_s/2 = \pi/\Delta = \pi\,.$$

(c) Using the values in (a), along with the given AR bandwidth parameters,

to arrive at *psd* overlaid plots for the signal and noise *pds's*.

<u>*Solution*</u>: [See code @ (c).]

Notice that even though the total SNR=0dB, the spectral SNR is frequency dependent. At low frequencies the SNR is ~10dB, while at high frequencies it is ~-15dB.

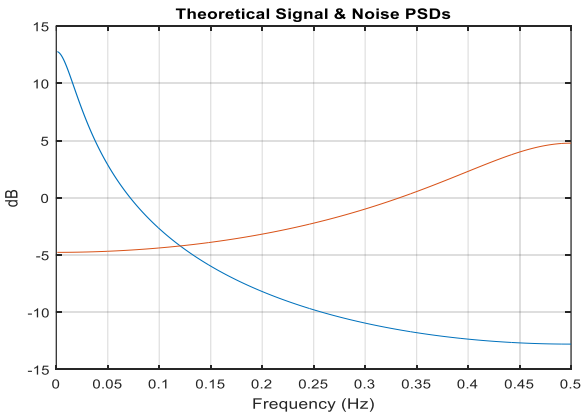

Figure 2(b) Signal and noise *psd's.*

(d) Use your results in (b) to obtain a plot of the theoretical Wiener filter for the signal. [Do not use dB units.]

Solution: [See code @ (d).]

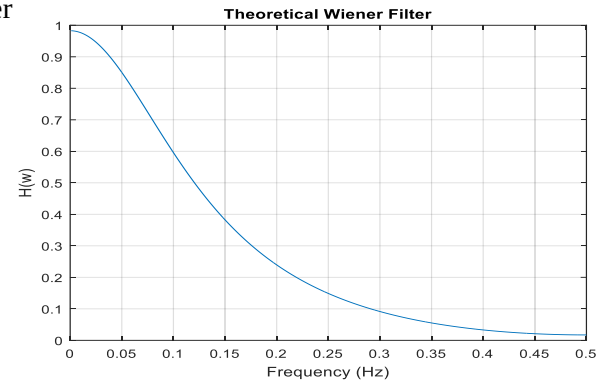

Figure 2(c) Theoretical Wiener filter.

(e) Use Matlab to generate a 1024-point sample of the signal-plus-noise wss process. [Generate an additional 1000 points, and then discard the first 1000 values so that the process can be assumed to be stationary.] Then run this process through the Wiener filter in (c) to obtain a signal estimate. Overlay the plots of the time domain signal-plus-noise, the signal and the estimated signal processes.

Solution: [See code @ (e).]

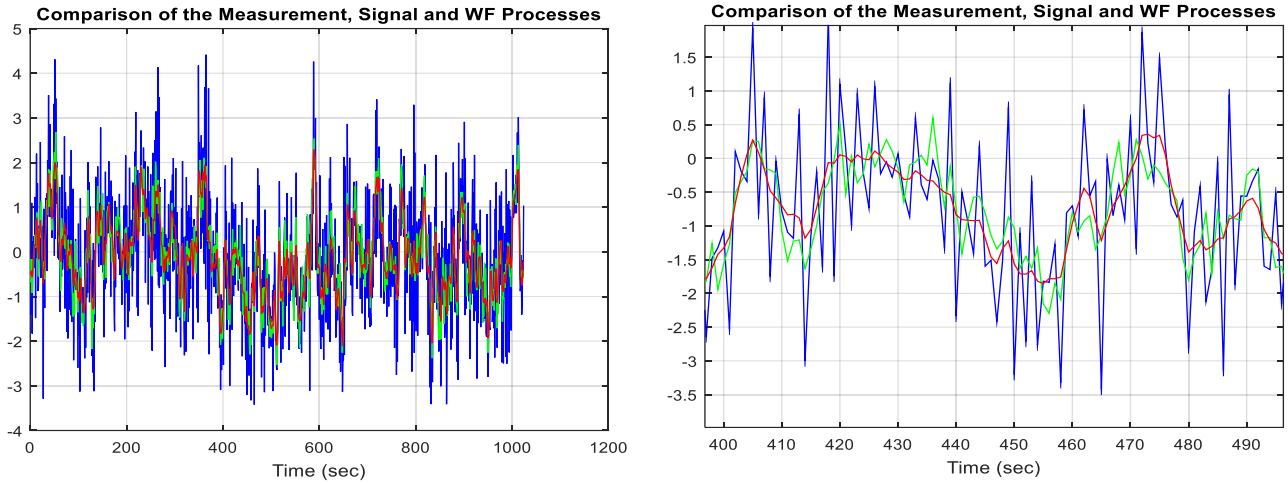

Figure 2(e) Comparison of the measurement, signal and WF estimate of the signal. Entire data set (left), and zoomed region (right).[Blue=measurement; Green=Signal; Red=WF estimate.]

APPENDIX Matlab Code

```
%LECTURE_14_EXAMPLE.m
%Part(c)
fv=0.001:0.001:0.5;
wv=2*pi*fv;
Ss=0.19*(1.81 - 1.8*cos(wv)).^-1;
Sn=0.75*(1.25 + cos(wv)).^-1;
SsdB=10*log10(Ss);
SndB=10*log10(Sn);
Smat=[SsdB ; SndB];
figure(3)
plot(fv,Smat);
```

```
xlabel('Frequency (Hz)')
ylabel('dB')
title('Theoretical Signal & Noise PSDs')
grid
%Part(d)
r=Ss./Sn;
H=(1+r.^-1).^-1;
figure(4)
plot(fv,H)
xlabel('Frequency (Hz)')
ylabel('H(w)')
title('Theoretical Wiener Filter')
grid
%Part(e)
ns=0.19^0.5*randn(1,2024);
nn=0.75^0.5*randn(1,2024);
n=[ns ; nn];
a=[0.9 ; -0.5];
xm1=n(:,1);
x=xm1;
for i=2:2024
   xi=a.*xm1 + n(:,i);
   x=[x xi];
   xm1=xi;
end
s=x(1,1001:2024);
n=x(2,1001:2024);
y=s+n;
fv=0:1/1024:1-1/1024;
wv=2*pi*fv;
Ss=0.19*(1.81 - 1.8*cos(wv)).^-1;
Sn=0.75*(1.25 + cos(wv)).^-1;
r=Ss./Sn;
H=(1+r.^-1).^-1;
Y=fft(y);
Shat=H.*Y;
shat=ifft(Shat);
shat=real(shat);
sv=[y;s;shat];
tv=1:1024;
figure(5)
plot(tv,y,'b',tv,s,'g',tv,shat,'r')
xlabel('Time (sec)')
title('Comparison of the Measurement, Signal and WF
Processes')
grid
```

Example continued. Now consider the prediction problem in the time domain. Recall:

$\hat{\mathbf{s}} = \mathbf{A}\mathbf{x}$ where $\mathbf{A} = \mathbf{R}_x^{-1}\mathbf{R}_s = \mathbf{R}_{(s+n)}^{-1}\mathbf{R}_s$ where $\mathbf{x} = \mathbf{s} + \mathbf{n}$ is the measurement data. We know that $R_s(k) = \alpha_s^k \sigma_s^2$ and $R_n(k) = \alpha_n^k \sigma_n^2$. Hence, $\mathbf{R}_s = Toeplitz\{R_s(0), \cdots, R_s(n-1)\}$ and $\mathbf{R}_n = Toeplitz\{R_n(0), \cdots, R_n(n-1)\}$. Hence,

$$\mathbf{R}_x = \mathbf{R}_{s+n} = \mathbf{R}_s + \mathbf{R}_n = Toeplitz\{R_x(0), \cdots, R_x(n-1)\}.$$