Delft University of Technology
Faculty of Electrical Engineering, Mathematics, and Computer Science
Circuits and Systems Group

EE 4C03 STATISTICAL DIGITAL SIGNAL PROCESSING AND MODELING

26 January 2017, 18:30–21:30

Open book exam: copies of the book by Hayes and the course slides allowed. No other materials allowed.

This exam has four questions (40 points)

Question 1 (10 points)

Consider a first-order AR process that is generated by the difference equation

$$y(n) = a\,y(n-1) + w(n),$$

where $|a| < 1$ and $w(n)$ is a zero mean white noise random process with variance σ_w^2.

(a) Find the unit sample response of the filter that generates $y(n)$ from $w(n)$.

(b) Find the autocorrelation sequence of $y(n)$.

(c) What is the variance σ_y^2 of the output process?

(d) Find the power spectrum of $y(n)$.

(e) What is the 4×4 autocorrelation matrix $\mathbf{R}_y$? Give 3 properties of this matrix.

(f) In general, if the $p \times p$ autocorrelation matrix $\mathbf{R}_y$ of some WSS random process $y(n)$ is *singular*, then what can you say about that process?

Solution

(a) The transfer function is $H(z) = \dfrac{1}{1 - az^{-1}}$. The impulse response of the filter is $h(n) = a^n u(n)$.

(b) (cf example 3.4.1.) You could evaluate $r_y(k) = r_x(k) * h(k) * h^*(-k)$ where $r_x(k) = \sigma_w^2 \delta(k)$. Alternatively:

$$P_y(z) = P_x(z)H(z)H^*(1/z^*) = \sigma_w^2 \frac{1}{1 - az^{-1}} \frac{1}{1 - az} = \cdots = \frac{\sigma_w^2}{1 - a^2} \left(\frac{1}{1 - az^{-1}} + \frac{z}{1 - az} \right)$$

Taking the inverse z-transform gives: $r_y(k) = \dfrac{\sigma_w^2}{1 - a^2} a^{|k|}$.

(c) $\sigma_y^2 = \mathrm{E}\{|y(n)|^2\} = r_y(0) = \dfrac{\sigma_w^2}{1 - a^2}$.

(d) The power spectrum is

$$P_y(e^{j\omega}) = \sigma_w^2 \frac{1}{1 + a^2 - 2a\cos(\omega)}$$

(e)

$$\mathbf{R}_y = \frac{\sigma_w^2}{1 - a^2} \begin{bmatrix} 1 & a & a^2 & a^3 \\ a & 1 & a & a^2 \\ a^2 & a & 1 & a \\ a^3 & a^2 & a & 1 \end{bmatrix}$$

This matrix is Hermitian, positive (semi) definite, and Toeplitz. Hence its eigenvalues are real and non-negative.

(f) The process becomes a predictable, harmonic (periodic) process. Here, this occurs for $a = e^{j\phi}$ on the complex unit circle. The poles are on the unit circle; this is then a marginally stable process.

Question 2 (10 points)

Consider the general second-order pole-zero model $H(z)$ given by

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}.$$

We want to fit this model to the convolution of two signals $x(n) = s(n) * t(n)$, where $s(n) = 0.1^n u(n)$ and $t(n) = 0.5^n u(n-1)$.

(a) Use the property that the z-transform of $\alpha^n u(n)$ is $1/(1 - \alpha z^{-1})$ and that of $x(n-k)$ is $z^{-k}X(z)$ to determine the z-transform of $x(n) = s(n) * t(n)$. Can we fit $H(z)$ to this model? If so, what are the parameters a_1, a_2, b_0, b_1 and b_2?

(b) Given the samples $x(0), x(1), x(2), \ldots$, determine the parameters of $H(z)$ using Padé's method. How many samples are needed?

Hint: The following expression might be useful for this:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

(c) Give the equations to estimate $H(z)$ using Prony's method? There is no need to derive the actual solution. How many samples are needed in this case?

(d) For this problem, is Prony's model going to change if we increase the number of signal values that we consider? Why or why not?

(e) Compare the three filter models and explain.

Solution

(a) First of all, the z-transform of $s(n)$ is

$$S(z) = \frac{1}{1 - 0.1z^{-1}}$$

and the z-transform of $t(n)$ is

$$T(z) = \frac{0.5z^{-1}}{1 - 0.5z^{-1}}.$$

Hence, the z-transform of $x(n)$ is

$$X(z) = S(z)T(z) = \frac{0.5z^{-1}}{(1 - 0.1z^{-1})(1 - 0.5z^{-1})} = \frac{0.5z^{-1}}{1 - 0.6z^{-1} + 0.05z^{-2}}.$$

This clearly fits the model $H(z)$, where $a_1 = -0.6$, $a_2 = 0.05$, $b_0 = 0$, $b_1 = 0.5$, and $b_2 = 0$.

(b) The Padé equations are given by

$$\begin{bmatrix} x(0) & 0 & 0 \\ x(1) & x(0) & 0 \\ x(2) & x(1) & x(0) \\ x(3) & x(2) & x(1) \\ x(4) & x(3) & x(2) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ 0 \\ 0 \end{bmatrix}.$$

We first solve the last two equations for a_1 and a_2, leading to

$$\begin{bmatrix} x(2) & x(1) \\ x(3) & x(2) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = - \begin{bmatrix} x(3) \\ x(4) \end{bmatrix}.$$

Plugging in the specific values, we obtain

$$\begin{bmatrix} 0.3 & 0.5 \\ 0.155 & 0.3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = - \begin{bmatrix} 0.155 \\ 0.078 \end{bmatrix}.$$

The solution is given by

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} -0.6 \\ 0.05 \end{bmatrix}.$$

Then we solve the first three equations for b_0, b_1, and b_2, leading to

$$\begin{aligned} b_0 &= x(0) = 0, \\ b_1 &= x(1) + a_1 x(0) = 0.5, \\ b_2 &= x(2) + a_1 x(1) + a_2 x(0) = 0. \end{aligned}$$

(c) The Prony method is using many more equations. For instance, if we consider the first 6 values of $x(n)$, we obtain

$$\begin{bmatrix} x(0) & 0 & 0 \\ x(1) & x(0) & 0 \\ x(2) & x(1) & x(0) \\ x(3) & x(2) & x(1) \\ x(4) & x(3) & x(2) \\ x(5) & x(4) & x(3) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Solving the last three equations for a_1 and a_2 in a least squares sense, we obtain

$$\begin{bmatrix} 1 \\ a_1 \\ a_2 \end{bmatrix} = -\left(\begin{bmatrix} x(2) & x(3) & x(4) \\ x(1) & x(2) & x(3) \end{bmatrix} \begin{bmatrix} x(2) & x(1) \\ x(3) & x(2) \\ x(4) & x(3) \end{bmatrix} \right)^{-1} \begin{bmatrix} x(2) & x(3) & x(4) \\ x(1) & x(2) & x(3) \end{bmatrix} \begin{bmatrix} x(3) \\ x(4) \\ x(5) \end{bmatrix}.$$

Similar least squares problems are obtained if more $x(n)$ samples are used. The values for b_0, b_1, and b_2 are obtained as before.

(d) The filter is not going to improve since it is already the best possible representation without any errors.

(e) All filter models are the same (exact models) and they are the best possible models. (This is not the case if the samples are from a stochastic process.)

Question 3 (10 points)

Let $x(n) = \sin(\omega_0 n + \phi) + w(n)$ be a (real-valued) sinusoid in noise. Here, ω_0 and ϕ are deterministic (non-random) unknown parameters. We are given data $x(n)$, $\quad n = 0, \cdots, N-1$, and we wish to estimate the frequency ω_0 using the MUSIC algorithm.

(a) Consider first the noise-free case. Define a data vector

$$\mathbf{x}(n) = [x(n),\ x(n+1),\ x(n+2),\ x(n+3)]^T$$

and a $4 \times (N-3)$ data matrix

$$\mathbf{X} = [\mathbf{x}(0),\ \mathbf{x}(1),\ \cdots,\ \mathbf{x}(N-4)].$$

What is the structure of this matrix in terms of the unknown parameters? What is the rank of this matrix?

Hint: you need to use the property $\sin(\alpha) = \frac{1}{2j}(e^{j\alpha} - e^{-j\alpha})$.

(b) Define the 4×4 data covariance matrix $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}(n)^H\}$.

What is a suitable corresponding estimate $\hat{\mathbf{R}}$ in terms of the given data $\mathbf{x}(n)$?

Consider the eigenvalue decomposition of $\hat{\mathbf{R}}$. For the noise-free case, what is the rank of this matrix?

What changes to the eigenvalues if we add white noise $w(n)$ with power σ_n^2?

(c) Briefly explain the MUSIC algorithm for this case. How is ω_0 estimated?

(d) What is the smallest size of the covariance matrix that can be used (if it is known in advance that there is only one sinusoid)?

(e) Let $\mathbf{P}$ be the matrix

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

When applied to a vector, this matrix reverses the entries of the vector.

In a technique called "forward backward averaging", $\hat{\mathbf{R}}$ is replaced by $\hat{\mathbf{R}} + \mathbf{P}\hat{\mathbf{R}}\mathbf{P}$. Explain why this can work, and why this leads to a better estimate of ω_0.

Solution

(a) We have the data matrix

$$\mathbf{X} = [\mathbf{x}(0),\ \mathbf{x}(1),\ \cdots,\ \mathbf{x}(N-4)] = \begin{bmatrix} x(0) & x(1) & \cdots & x(N-4) \\ x(1) & x(2) & \cdots & x(N-3) \\ x(2) & x(3) & \cdots & x(N-2) \\ x(3) & x(4) & \cdots & x(N-1) \end{bmatrix}$$

We can write $x(n) = \frac{1}{2j}e^{j\phi}e^{j\omega_0 n} - \frac{1}{2j}e^{-j\phi}e^{-j\omega_0 n} = \alpha e^{j\omega_0 n} + \alpha^* e^{-j\omega_0 n}$, where $\alpha = \frac{1}{2j}e^{j\phi}$. The structure of $\mathbf{x}(n)$ is

$$\mathbf{x}(n) = \begin{bmatrix} \alpha e^{j\omega_0 n} \\ \alpha e^{j\omega_0 (n+1)} \\ \alpha e^{j\omega_0 (n+2)} \\ \alpha e^{j\omega_0 (n+3)} \end{bmatrix} + \begin{bmatrix} \alpha^* e^{-j\omega_0 n} \\ \alpha^* e^{-j\omega_0 (n+1)} \\ \alpha^* e^{-j\omega_0 (n+2)} \\ \alpha^* e^{-j\omega_0 (n+3)} \end{bmatrix} = \alpha e^{j\omega_0 n} \begin{bmatrix} 1 \\ e^{j\omega_0} \\ e^{j\omega_0 \cdot 2} \\ e^{j\omega_0 \cdot 3} \end{bmatrix} + \alpha^* e^{-j\omega_0 n} \begin{bmatrix} 1 \\ e^{-j\omega_0} \\ e^{-j\omega_0 \cdot 2} \\ e^{-j\omega_0 \cdot 3} \end{bmatrix}$$

The structure of $\mathbf{X}$ is

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ e^{j\omega_0} & e^{-j\omega_0} \\ e^{j\omega_0 \cdot 2} & e^{-j\omega_0 \cdot 2} \\ e^{j\omega_0 \cdot 3} & e^{-j\omega_0 \cdot 3} \end{bmatrix} \begin{bmatrix} \alpha & \alpha e^{j\omega_0} & \cdots & \alpha e^{j\omega_0 (N-3)} \\ \alpha^* & \alpha^* e^{-j\omega_0} & \cdots & \alpha^* e^{-j\omega_0 (N-3)} \end{bmatrix}$$

From this factorization, it is seen that it is a rank-2 matrix.

(b) We can estimate $\mathbf{R} = E\{\mathbf{x}(n)\mathbf{x}(n)^H\}$ by averaging the available data vectors $\mathbf{x}(n)\mathbf{x}(n)^H$, i.e., $\hat{\mathbf{R}} = \frac{1}{N-3}\mathbf{X}\mathbf{X}^H$. This gives

$$\hat{\mathbf{R}} = \frac{1}{N-3}\mathbf{X}\mathbf{X}^H = \begin{bmatrix} 1 & 1 \\ e^{j\omega_0} & e^{-j\omega_0} \\ e^{j\omega_0 \cdot 2} & e^{-j\omega_0 \cdot 2} \\ e^{j\omega_0 \cdot 3} & e^{-j\omega_0 \cdot 3} \end{bmatrix} \hat{\mathbf{R}}_s \begin{bmatrix} 1 & e^{-j\omega_0} & e^{-j\omega_0 \cdot 2} & e^{-j\omega_0 \cdot 3} \\ 1 & e^{j\omega_0} & e^{j\omega_0 \cdot 2} & e^{j\omega_0 \cdot 3} \end{bmatrix}$$

where $\hat{\mathbf{R}}_s$ is some 2×2 matrix that will converge to the identity matrix for large N.

From the factorization of $\hat{\mathbf{R}}$, it is clear that it is a rank-2 matrix.

If we add white noise, the rank becomes full. The two eigenvalues that were zero are now nonzero: in expectation they become σ_n^2. Also the two largest eigenvalues are augmented by something close to σ_n^2.

(c) MUSIC computes an eigenvalue decomposition of $\hat{\mathbf{R}}$ and selects the eigenvectors $\mathbf{V}_n = [\mathbf{v}_3,\ \mathbf{v}_4]$ corresponding to the *smallest* eigenvalues. They span the noise subspace and are orthogonal to the signal subspace, spanned by

$$\begin{bmatrix} 1 & 1 \\ e^{j\omega_0} & e^{-j\omega_0} \\ e^{j\omega_0 \cdot 2} & e^{-j\omega_0 \cdot 2} \\ e^{j\omega_0 \cdot 3} & e^{-j\omega_0 \cdot 3} \end{bmatrix}$$

Subsequently, MUSIC defines a vector

$$\mathbf{e}(\omega) = \begin{bmatrix} 1 \\ e^{j\omega_0} \\ e^{j\omega_0 \cdot 2} \\ e^{j\omega_0 \cdot 3} \end{bmatrix}$$

which is a template vector for the vectors in the signal subspace that we look for. For the right ω_0, we know that $\mathbf{e}(\omega_0)$ and $\mathbf{e}(-\omega_0)$ are orthogonal to $\mathbf{V}_n$. Thus, MUSIC forms the cost function

$$\xi(\omega) = \frac{1}{\mathbf{e}(\omega)^H \mathbf{V}_n \mathbf{V}_n^H \mathbf{e}(\omega)}$$

and looks for the maxima of this function over ω. There are two maxima, for ω_0 and for $-\omega_0$.

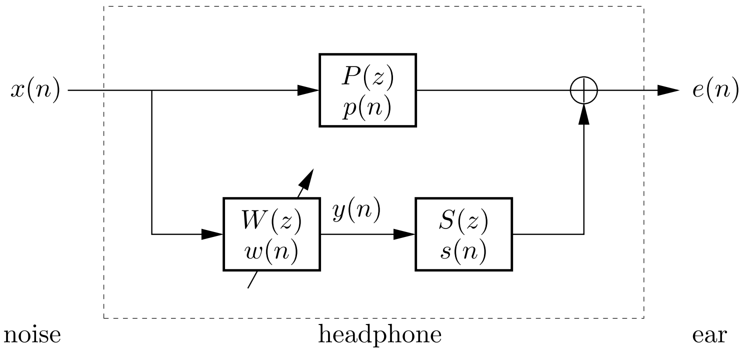
(d) We need to have a noise subspace of at least dimension 1, so the smallest size is 3×3.

(e) If we apply $\mathbf{P}$ to $\mathbf{X}^*$, we obtain a matrix with a similar structure:

$$\begin{aligned}
\mathbf{PX}^* &= \begin{bmatrix} e^{-j\omega_0 \cdot 3} & e^{j\omega_0 \cdot 3} \\ e^{-j\omega_0 \cdot 2} & e^{j\omega_0 \cdot 2} \\ e^{-j\omega_0} & e^{j\omega_0} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha^* & \alpha^* e^{-j\omega_0} & \cdots & \alpha^* e^{-j\omega_0 (N-3)} \\ \alpha & \alpha e^{j\omega_0} & \cdots & \alpha e^{j\omega_0 (N-3)} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 \\ e^{j\omega_0} & e^{-j\omega_0} \\ e^{j\omega_0 \cdot 2} & e^{-j\omega_0 \cdot 2} \\ e^{j\omega_0 \cdot 3} & e^{-j\omega_0 \cdot 3} \end{bmatrix} \begin{bmatrix} e^{-j\omega_0 \cdot 3} & \\ & e^{j\omega_0 \cdot 3} \end{bmatrix} \begin{bmatrix} \alpha^* & \alpha^* e^{-j\omega_0} & \cdots & \alpha^* e^{-j\omega_0 (N-3)} \\ \alpha & \alpha e^{j\omega_0} & \cdots & \alpha e^{j\omega_0 (N-3)} \end{bmatrix}
\end{aligned}$$

Also note that $\mathbf{P}(\mathbf{X}^* \mathbf{X}^{H*})\mathbf{P} = \mathbf{P}\hat{\mathbf{R}}^*\mathbf{P} = \mathbf{P}\hat{\mathbf{R}}\mathbf{P}$, since it is a real-valued matrix. Thus, $\hat{\mathbf{R}} + \mathbf{P}\hat{\mathbf{R}}\mathbf{P}$ is still rank-2 and its column span is generated by the same two vectors $\mathbf{e}(\omega_0)$ and $\mathbf{e}(-\omega_0)$ as before. MUSIC will perform the same on this matrix. But because it is formed by averaging twice as many samples, it will perform better in the presence of noise.

Question 4 (10 points)



In acoustic noise control (ANC) for headphones, we want to actively generate anti-noise to cancel the noise such that only the useful signal remains. Suppose that $x(n)$ is the noise we want to cancel and which can be picked up with a microphone outside the headphone. This noise $x(n)$ should be canceled inside the ear so what we actually want to cancel is $x(n)$ after traveling to the inside ear, via what is called the primary path (modeled using the transfer function $P(z)$). The anti-noise that is generated by the headphone is denoted by $y(n)$ but this signal also has to travel to the inside ear, via what is called the secondary path (modeled using the transfer function $S(z)$). Hence, we actually want to have that $S(z)Y(z) = -P(z)X(z)$, so that the noise is perfectly canceled inside the ear. Unless otherwise stated, it is assumed that the primary and secondary paths are known, i.e., $P(z)$ and $S(z)$ are known.

(a) Suppose we aim to generate $y(n)$ from $x(n)$ using a linear filter $w(n)$, i.e., $y(n) = w(n) * x(n)$. Give the expression for the optimal transfer function of the filter $w(n)$, i.e., derive the optimal $W(z)$, where we assume no particular filter structure.

What complication do you see with this solution?

(b) Let us now use the theory of optimal filtering to compute $W(z)$. This time we assume that all filters ($P(z)$, $S(z)$, and $W(z)$) are causal FIR filters with a finite order. First, derive the expression for the mean square error $E\{e^2(n)\}$ where $e(n) = s(n) * y(n) + p(n) * x(n) = s(n) * w(n) * x(n) + p(n) * x(n)$. From this expression, derive the Wiener-Hopf equations for $w(n)$ by taking the derivative towards $w(n)$ for all its taps.
Hint: Use the commutative property of the convolution and write the convolution using a matrix-vector product. More specifically, we can write the convolution between $x_1(n)$ and $x_2(n)$ as $\mathbf{x}_1^T\mathbf{X}_2 = \mathbf{x}_2^T\mathbf{X}_1$, where $\mathbf{x}_1 = [x_1(0), \ldots, x_2(N_1)]^T$, $\mathbf{x}_2 = [x_2(0), \ldots, x_2(N_2)]^T$, $\mathbf{X}_1$ is an $(N_2 + 1) \times (N_1 + N_2 + 1)$ Toeplitz matrix based on $\mathbf{x}_1$, and $\mathbf{X}_2$ is an $(N_1 + 1) \times (N_1 + N_2 + 1)$ Toeplitz matrix based on $\mathbf{x}_2$.

(c) Can you transform problem (b) into a classical optimal filtering problem with desired signal given by $d(n)$ and input of the filter $w(n)$ given by $u(n)$? What is $d(n)$ and $u(n)$ in this case?

(d) Based on the above, give the LMS update equations for the causal FIR filter $w(n)$.

(e) Suppose now that the error signal $e(n)$ is measured using a second microphone inside the headset. What would be the advantage of such a second microphone for the LMS algorithm in terms of the knowledge of $P(z)$ and/or $S(z)$?

Give the LMS update equation for this case.

Solution

(a) Since $Y(z) = W(z)X(z)$ and since we need $X(z)P(z) = -Y(z)S(z)$ it is clear that the optimal filter if given by

$$W(z) = -\frac{P(z)}{S(z)}.$$

The problem with this expression is that we generally don't know $P(z)$ and $S(z)$.

(b) Suppose the filter $p(n)$ has order L_p, the filter $s(n)$ has order L_s and the filter $w(n)$ has order L_w. Then the first term of $e(n)$ can be written in matrix-vector form as

$$s(n) * w(n) * x(n) = \mathbf{s}^T\mathbf{W}\mathbf{x}_1(n),$$

where $\mathbf{s} = [s(0), s(1), \ldots, s(L_s)]^T$, $\mathbf{W}$ is an $(L_s+1) \times (L_s+L_w+1)$ convolution matrix using the filter $\mathbf{w} = [w(0), w(1), \ldots, w(L_w)]^T$, and $\mathbf{x}_1(n) = [x(n), x(n-1), \ldots, x(n-L_s-L_w)]^T$. The second term of $e(n)$ can be written as

$$p(n) * x(n) = \mathbf{p}^T \mathbf{x}_2(n),$$

where $\mathbf{p} = [p(0), p(1), \ldots, p(L_p)]^T$, and $\mathbf{x}_2(n) = [x(n), x(n-1), \ldots, x(n-L_p)]^T$. From these expressions, the MSE can be written as

$$\xi = \mathbf{s}^T \mathbf{W} \mathbf{R}_{x_1,x_1} \mathbf{W}^T \mathbf{s} + 2\mathbf{s}^T \mathbf{W} \mathbf{R}_{x_1,x_2} \mathbf{p} + \mathbf{p}^T \mathbf{R}_{x_2,x_2} \mathbf{p}.$$

Then we use the fact that the convolution of $s(n)$ and $w(n)$ is commutative, which means that $s(n) * w(n) = w(n) * s(n)$, or $\mathbf{s}^T \mathbf{W} = \mathbf{w}^T \mathbf{S}$, where $\mathbf{S}$ is an $(L_w+1) \times (L_s+L_w+1)$ convolution matrix using the filter $\mathbf{s}$. Hence, the MSE can also be written as

$$\xi = \mathbf{w}^T \mathbf{S} \mathbf{R}_{x_1,x_1} \mathbf{S}^T \mathbf{w} + 2\mathbf{w}^T \mathbf{S} \mathbf{R}_{x_1,x_2} \mathbf{p} + \mathbf{p}^T \mathbf{R}_{x_2,x_2} \mathbf{p}.$$

Taking the derivative towards $\mathbf{w}$ and setting this derivative to zero, we obtain the Wiener-Hopf equations. These are given by

$$\mathbf{S} \mathbf{R}_{x_1,x_1} \mathbf{S}^T \mathbf{w} = -\mathbf{S} \mathbf{R}_{x_1,x_2} \mathbf{p}.$$

(c) The classical Wiener-Hopf equations with input $u(n)$ and desired response $d(n)$, look like

$$\mathbf{R}_{u,u} \mathbf{w} = \mathbf{R}_{d,u},$$

where $\mathbf{R}_{u,u} = E\{\mathbf{u}(n)\mathbf{u}^T(n)\}$ and $\mathbf{R}_{d,u} = E\{d(n)\mathbf{u}(n)\}$, with $\mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-L_w)]^T$. This corresponds to the previous Wiener-Hopf equations if we set $\mathbf{u}(n) = \mathbf{S}\mathbf{x}_1(n)$ and $d(n) = -\mathbf{p}^T \mathbf{x}_2$. Hence, if we use the classical Wiener-Hopf equations with as input $u(n) = s(n) * x(n)$ and as desired response $d(n) = -p(n) * x(n)$, then we obtain the Wiener-Hopf equations from (b).

(d) Now it is easy to derive the LMS update equations, since we know the input $u(n)$ and the desired response $d(n)$. Hence, we obtain

$$e(n) = \hat{\mathbf{w}}^{(n)} \mathbf{u}(n) - d(n) = \hat{\mathbf{w}}^{(n)} \mathbf{S} \mathbf{x}_1(n) - \mathbf{p}^T \mathbf{x}_2(n),$$
$$\hat{\mathbf{w}}^{(n+1)} = \hat{\mathbf{w}}^{(n)} - \mu \mathbf{u}(n) e(n) = \hat{\mathbf{w}}^{(n)} - \mu \mathbf{S} \mathbf{x}_1(n) e(n).$$

(e) In that case, we can directly use the error $e(n)$ in the LMS algorithm and we don't need to know the primary path.