ECE551, Fall 2016: Homework Problem Set #6
Due Nov. 1 2016 by 12:30pm

1. **DTFT of Auto-correlation and Cross-correlation**
Let $\{x[\cdot]\}$ and $\{w[\cdot]\}$ be uncorrelated WSS sequences (i.e their cross-correlation vanish) For $y := x + w$, show that the DTFT of the correlation sequences $c_{x,y}[k]$ and $a_y[k]$ satisfy

$$C_{x,y}(\omega) = A_x(\omega) \qquad\qquad A_y(\omega) = A_x(\omega) + A_w(\omega)$$

2. **Highly Correlated Random Processes**
Let $A, B \sim \mathcal{N}(0,1)$ be uncorrelated Gaussian variables. Consider the random sequences:

$$x_1[n] = \begin{cases} A & \text{even } n \\ B & \text{odd } n \end{cases} \qquad x_2[n] = \begin{cases} A & n \geq 0 \\ B & n < 0 \end{cases} \qquad \begin{cases} x_3[n+1] & = \frac{1}{2}x_3[n] + A \\ x_3[0] & = A \end{cases}$$

   (a) Determine if each process is WSS, and if periodic.

   (b) For each of the three processes, find a linear predictor of $x[n+1]$ based on $x[n]$ and $x[n-1]$, and the prediction error (which may vary with n if $\{x[n]\}$ is not WSS)

3. **Adaptive Filter and LMS**
An adaptive filter w of length L is fed with a WSS input $\{x[\cdot]\}$ along with the reference

$$d[n] = \alpha_1 x[n-1] + \alpha_2 x[n-2] \qquad\qquad \alpha_1, \alpha_2 \in \mathbb{R}.$$

The auto-correlation is known, and given as

$$\mathbb{E}(x[0]x[m]) = 2^{-|m|} + 4^{-|m|}$$

   (a) Which of the two cost functions is suitable for this adaptive filter design ? explain.

   (b) For your selected cost, find the optimal filter coefficients vector w_{opt}, and $C(w_{opt})$.

   (c) Determine the gradient $\nabla_w C(w)$ of your selected cost, write the gradient descend equations for $\hat{w}[n]$ and find the limit of $\hat{w}[n]$.

   (d) Write the LMS update equations for $\hat{w}[n]$, and find a sufficient condition for $\mathbb{E}(\hat{w}[n])$ to converge. Does that guarantee convergence of $\hat{w}[n]$? explain.

4. **Regularized Wiener Filter and Leaky LMS**
Consider an adaptive filter w of length L with $\{x[\cdot]\}$ and $\{d[\cdot]\}$ jointly WSS with known correlations, and R_x that is a singular matrix.

   (a) Explain why the LMS might diverge ("explode") in this scenario. **Hint:** gradient.

The regularized Wiener filter is an adaptation strategy that penalizes both the mean-square-error as well as the norm of w, i.e the (slightly modified) cost is now

$$C_\lambda(w) = \mathbb{E}\{|e[n]|^2\} + \lambda\|w\|^2 \qquad\qquad \lambda > 0 \qquad (1)$$

   (b) Determine the gradient $\nabla_w C_\lambda(w)$, and the minimum of $C_\lambda(w_{opt})$.

   (c) Explain why the minimum of $C_\lambda(w)$ is unique, even though R_x is singular.

(d) Determine the analogous LMS update equations of C_λ (known as *leaky LMS*) and find a sufficient bound on the step-size μ for which mean-convergence is guaranteed.

(e) Let $\{x[\cdot]\}$ be a WSS process with $a_x[k] = \frac{3}{4} + \frac{1}{4}(-1)^k$. Find a Wiener filter of length $L = 3$ for the one-step prediction of x, and find the leaky Wiener filter with $\lambda = 0.1$.

5. **Python Problem - Wiener's LMS**
Let $\{x[\cdot]\}$ be a random process governed by the equation

$$x[n+1] = \alpha x[n] + s[n] - 0.5s[n-1] \qquad\qquad x[-1] = 0 \tag{2}$$

where $\{s[n]\}$ is a standard white Gaussian noise and α is real. As usual, we want to linearly estimate $\hat{x}[n+1]$ based on $X[n] = \begin{bmatrix} x[n], & \ldots, & x[n-L+1] \end{bmatrix}^T$.

(a) Formulate the prediction problem as an adaptive filter diagram. Clearly state the input x, the reference d and the cost function C.

(b) Compute the autocorrelation $\mathbb{E}(x[0]x[m])$, and write a function that generates R_x, R_{xd}, and the associated Wiener filter
Note: whenever $X(z) = H(z)S(z)$ we have $A_x(z) = H(z)H(z^{-1})A_s(z)$.

```
def probabilistic_Wiener(L,alpha):
    Rx = # Write your code here
    Rxd = # Write more code here
    return w_opt
```

(c) Complete the following code that creates a realization of $\{x[n]\}$ of length N

```
from numpy import zeros
from numpy.random import randn # Basically a white Gaussian generator
def gen_x(alpha,N):
    x = zeros(N)
    s_now, s_prev = randn() ,randn() # s[0], s[-1]
    x[0] = s_now-0.5*s_prev # First iteration assumes x[-1]=0
    for n in ?
        s_prev, s_now = s_now, randn() # Promote the noise
        x[n] = ?
    return x
```

(d) Write a function that statistically estimates $\hat{R}_x$ and $\hat{R}_{xd}$ and a corresponding $\hat{w}$.

```
def statistical_Wiener(L,x):
    Rx, Rxd = np.zeros((L,L)), np.zeros(L)
    # Write your code here
    return w
```

(e) Implement the LMS algorithm on x of order L. As output, return an array of size $N \times L$ containing the values of $\hat{w}[n]$, and an array of length N of predicted $\hat{x}[n+1]$.

```
def LMS(x, L, mu):
    # Fill in the blank
    return w, x_est
```

(f) Test your three adaptive filters with $L = 2,\ 4,\ 7$, $\alpha = 0,\ 0.9$, and appropriate μ (deduce from R_x). Plot the actual $x[n+1]$, the estimated $\hat{x}[n+1]$, and the temporal error $e[n]$. For the LMS, also find the distance $\|\hat{w}[n] - w_{opt}\|^2$ where w_{opt} is the Wiener filter. Comment on the nature of its convergence (monotone or not, fluctuates, rate).

(g) For the LMS, experiment with a couple of different values of μ for three different combinations of L and α.

Note: those are many different cases. You can plot multiple graphs on the same axis, use different colors and line styles, especially if you want to compare them (e.g different values of μ). You can also use subplots.

```
N = 500
cases = [(alpha, L) for alpha in [0, 0.9] for L in [2,4,7]]
for alpha,L in cases:
     x = gen_x(alpha,N)
     w_opt = probabilistic_Wiener(L,alpha)
     w_stat = statistical_Wiener(L,x)

     x_prob = # filter x using w_opt
     x_stat = # ...and using w_stat

     mu = ?
     x_lms, w_hat = LMS(x,L,mu)

     # Plot a lot of stuff
     print("Estimated Wiener Error:", norm(w_stat-w_opt))
```

6. **Python Problem - AR System Identification**

Download the file data.csv from the course website.

In this problem, you are given a sample realization of an unknown AR system

$$x[n+1] = w^T X[n] + s[n]$$

where s is a white Gaussian noise with $\sigma^2 = 1$.

Your goal is to estimate w and its length L, in a statistical method of your choice (LMS or statistical Wiener).

Use the following code to get started. Use your training - good luck !

```
x = np.loadtxt('data.csv',delimiter=',')

# Fill in some smart and beautiful Python code here

print('The estimated order is:', L)
print('The estimated filter:', w)
```