Carnegie Mellon ADSP Electrical & Computer ENGINEERING

Advanced Digital Signal Processing (18-792)

Fall Semester 2024

Department of Electrical and Computer Engineering

Problem Set 8

Issued: 10/31/24
Due: 11/08/24 at 0100 via Gradescope

Reminder: 18-792 Quiz 2 will take place in class on November 20. The format will be similar to that of Quiz 1. The coverage is expected to be through Problem Set 9.

Reading: This problem set covers the core topics in linear prediction and its implementation. The core source material is Secs. 7.1 and 7.2 in the class notes, which in turn are derived from Secs. 8.1 through 8.3.2 of Rabiner and Schafer (RS), both of which are available on the Web. You are also encouraged to look over the classic review article by Makhoul on linear prediction, also on the course website. Be aware of the difference in notational convention used by Makhoul which is discussed in detail at the beginning of the MATLAB problems.

Problem 8.1: An all-zero filter has a transfer function of

$$H(z) = \frac{X(z)}{W(z)} = 1 - 0.9z^{-1}$$

A white noise WSS random process $w[n]$ with variance equal to 1 is input to the filter, and $x[n]$ is the output random process.

(a) Using random process properties and your knowledge of the filters response, obtain the values of $\phi_{xx}[m]$, the autocorrelation function of $x[n]$, the system's output for lags $|m| \leq 3$.

(b) Using the Levinson-Durbin recursion technique, and the autocorrelation function for $x[n]$ that you calculated in part (a), obtain the α and the k coefficients for the first-, second-, and third-order LPC model of the filter. Please obtain this result by hand (using a calculator or its computer equivalent), rather than using routines in MATLAB . For each order of the model, sketch the locations of the poles of the z-transform of the system. Using MATLAB , sketch the magnitude of the DTFT in decibels (dB) for each of the three models.

Problem 8.2: (a former quiz problem)

In class we developed the basic LPC equations that minimized the forward mean square error

$$\xi^2 = \mathbb{E}\left\{e_P^2[n]\right\} = \mathbb{E}\left\{\left(x[n] - \sum_{k=1}^{P} \alpha_k^{(P)} x[m-k]\right)^2\right\}$$

and we will introduce the concept of backward error in our discussion of lattice filters. It is also possible to consider a non-causal operation that predicts the current sample of $x[n]$ from both past and future samples in a two-sided fashion:

$$\hat{x}[n] = \sum_{k=1}^{P} \alpha_k^{(P)} (x[n-k] + x[n+k])$$

The corresponding mean-square error for such a system would be

$$\xi^2 = \mathbb{E}[(x[n] - \hat{x}[n])^2] = \mathbb{E}\left\{\left(x[n] - \left[\sum_{k=1}^{P} \alpha_k^{(P)} (x[n-k] + x[n+k])\right]\right)^2\right\}$$

(a) Obtain a set of matrix-vector equations that will specify the set of coefficients α_k when solved.

Your equations should be similar to (but not identical to) the standard LPC equations that are solved, for example, by Levinson-Durbin recursion. The matrices and vectors in the equations should be expressed in terms of the coefficients α_k and $\phi[m] = \phi_{xx}[m] = \mathbb{E}[x[n]x[n+m]]$, the auto-correlation function of $x[n]$.

You may assume that $x[n]$ is a real wide-sense stationary random process.

(b) Can your answer to part (a) be obtained using Levinson-Durbin recursion? Why or why not?

(c) Can your answer to part (a) be obtained using Cholesky decomposition? Why or why not?

Problem 8.3:

A particular filter has a transfer function of the form

$$H(z) = \frac{G}{1 - 0.3167z^{-1} - 0.1361z^{-2} + 0.7290z^{-3}} = \frac{G}{A(z)}$$

Without computing the correlation coefficients, obtain numerical values for the reflection coefficients associated with the denominator polynomial $A(z)$.

MATLAB Problems

Note: In working the problems, turn in a printout of your results, a copy of the MATLAB or Python code you developed to work the problem, as well as any additional comments you'd like to add.

WARNING!! The texts by Oppenheim and Schafer, the Lim and Oppenheim book, and the boosk by Rabiner and Schafer have all consistently used the filter notation convention

$$H(z) = \frac{\sum_{l=0}^{M} b_l z^{-l}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

We have used this notation in class and in the course notes with some variants (such as using the coefficients α_k instead of a_k).

Unfortunately, MATLAB uses the convention used in the classic review article on linear prediction and lattice filtering by John Makhoul (1975):

$$H(z) = \frac{\sum_{l=0}^{M} b_l z^{-l}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

This means that the coefficients of the denominator polynomial a_k and the reflection coefficients k_i produced by MATLAB are opposite in sign to what we have seen in Rabiner and Schafer and used in class discussions. In turning in your homework beginning with Problem C8.1 you must use the OSYP/LO/RS/RMS/18-792 conventions! Please save yourself and us a lot of aggravation by being careful about this annoying inconsistency! (Thanks!)

Problem C8.1: You will be provided with a main file called `main_8_1.m` or `main_8_1.py` that you must complete for this problem.

Consider again the all-zero filter you examined in Problem 8.1,

$$H(z) = 1 - 0.9z^{-1}$$

Using the MATLAB routines `lpc` or `levinson`, along with `freqz`, sketch the magnitude of the DTFT of the best-fit all-pole approximation to $H(z)$ with 5 poles, 10 poles, 15 poles, and 20 poles. How many poles do you think are needed for an adequate all-pole approximation to the original transfer function with a single zero?

Note: For Python, use `scipy.linalg.solve_toeplitz` (there are third-party libraries that implement Levinson-Durbin in Python but we recommend this as a simple alternative) to solve for the alphas and `scipy.signal.freqz` to sketch the magnitude.

Problem C8.2: In this problem we will compare the resulting model obtained by analyzing a segment of the speech waveform using the autocorrelation and covariance methods. (We will consider a third method called the "partial correlation method" next week.) In each case we will analyze the 20-ms segment of the "Welcome to DSP-I" utterance corresponding to samples 17000 through 17319. Turn in the MATLAB or Python script you write and the frequency response plots you obtain for the two methods. Please use the audio in the file `welcome16k.wav`, which is included in the folder with code templates on the web. Please work this problem using the file `main_8_2.m` or `main_8_2.py` that is provided.

(a) **Autocorrelation method:**

You will obtain the autocorrelation analysis of the segment in the following fashion.

- Apply a Hamming window to samples 17000 through 17319 of the welcome utterance.
- Compute the autocorrelation coefficients for lags 0 through 14 of your segment in the usual fashion.
- Complete the MATLAB or Python function we provide called `levinson792` that performs Levinson-Durbin recursion on these autocorrelation coefficients. (While this function is already accomplished by the MATLAB routine `levinson`, please do not use the code in `levinson`. You can, of course, use `levinson` or `lpc` to check your results.)
- Using your routine, compute the coefficients α_k for the polynomial $A(z) = 1 - \sum_{k=1}^{P} \alpha_k^{(P)} z^{-k}$. Use a model order of $P = 14$.
- Compute the LPC gain parameter using the procedures discussed in RS Sec. 8.2 and Sec. 7.2.5 in the class notes.

Using the results of this analysis, please answer the following questions:

1. Obtain the frequency response of the resulting model by using the MATLAB routine `freqz` or the Python routine `scipy.signal.freqz`.
2. What are the values of the LPC coefficients, the value of the gain parameter, the values of the reflection coefficients, and the locations of the poles resulting from the autocorrelation analysis?

(b) **Covariance method:**

You will obtain the covariance analysis of the segment in the following fashion.

- Obtain the lags of the autocorrelation function for the covariance method according to the procedures described in RS Sec. 8.1.2, and especially RS Eq. 8.28b. In this case, let the $m = 0$ point correspond to sample 17000 and the $m = N - 1$ point correspond to sample number 17319.
- While the LPC coefficients are normally obtained in the covariance method using Cholesky decomposition, (which is represented in MATLAB by the `chol` function), you can solve for the LPC coefficients most easily by using the `inv` function in MATLAB (or `np.linalg.inv`

in Python) to invert the covariance matrix in the Wiener-Hopf Equation (RS Eq. 8.29b). Again, use a model order of $P = 14$ for these calculations as well.

- Compute the LPC gain parameter using the procedures discussed in RS Sec. 8.2.

Using the results of this analysis, please answer the following questions:

1. Obtain the frequency response of the resulting model by using the MATLAB routine `freqz` or the Python routine `scipy.signal.freqz`.
2. What are the values of the LPC coefficients, the value of the gain parameter, the values of the reflection coefficients, and the locations of the poles resulting using the covariance method?
3. In what ways do you think that the covariance method might provide a better representation for the random process?