Level I Signal Modeling and Adaptive Spectral Analysis

1 Learning Objectives

- Students will learn about autoregressive signal modeling as a means to represent a stochastic signal. This differs from using a transform, such as the Fourier transform or wavelet transform, which is used to map a signal from one domain to another.
- The well known Kalman estimator will be briefly studied, since a variation of it can be viewed as a parametric estimation process which relies on an autoregressive (AR) model in its internal the signal estimation process.
- Students will learn about parametric modeling of Level I data for each range gate of a radars sample volume. As storage capabilities become more prevalent, such as KOUN, massive amounts of Level I data can be stored and analyzed.
- Students will learn how autoregressive parameters are related to spectral analysis. Such all-pole modeling is very effective in representing peaks or bumps in a signals spectrum.
- Students will learn how to develop an adaptive technique that relies on autoregressive signal modeling to estimate the spectrum for a range gate of data. In particular, when a range gate has more than one type of scatterer, multiple peaks in the Doppler spectrum may appear. Thus the autoregressive parameters can be used to represent the peaks in the spectrum, which is very similar to the modeling of voice data.

2 Introduction

Signal modelling is a broad class of processing techniques where a stochastic signal of interest is modelled as a certain type of process, such as an autoregressive moving-average (ARMA) process or as a collection of sinusoids. In fitting the signal to the model, several parameters are obtained which can then be used in a variety of ways, including spectral analysis, frequency estimation, and adaptive signal processing. The focus here will be on modelling a signal as an autoregressive (AR) process, also known as an all-pole model, which is a special class of the more-general ARMA process model. Using the AR model, the location of multiple peaks within the frequency spectrum can be obtained, which can be used in a weather radar to estimate the velocity of meteorological targets in the presence of moving biological clutter. The AR model can also be used within the framework of the Kalman filter, a powerful adaptive filter that is employed in a wide variety of applications.

2.1 Autoregressive Modelling

In digital signal processing, the input-output relation of a linear time-invariant (LTI) system is given (in the z domain) by

$$Y(z) = \frac{B(z)}{A(z)}X(z) = H(z)X(z) \tag{1}$$

where $H(z)$ is called the filter response. In an LTI system, $A(z)$ and $B(z)$ are polynomials, so $H(z)$ can be written as

$$H(z) = \frac{\sum_{i=1}^{n} b_n z^{-n}}{\sum_{j=1}^{m} a_m z^{-m}} \tag{2}$$

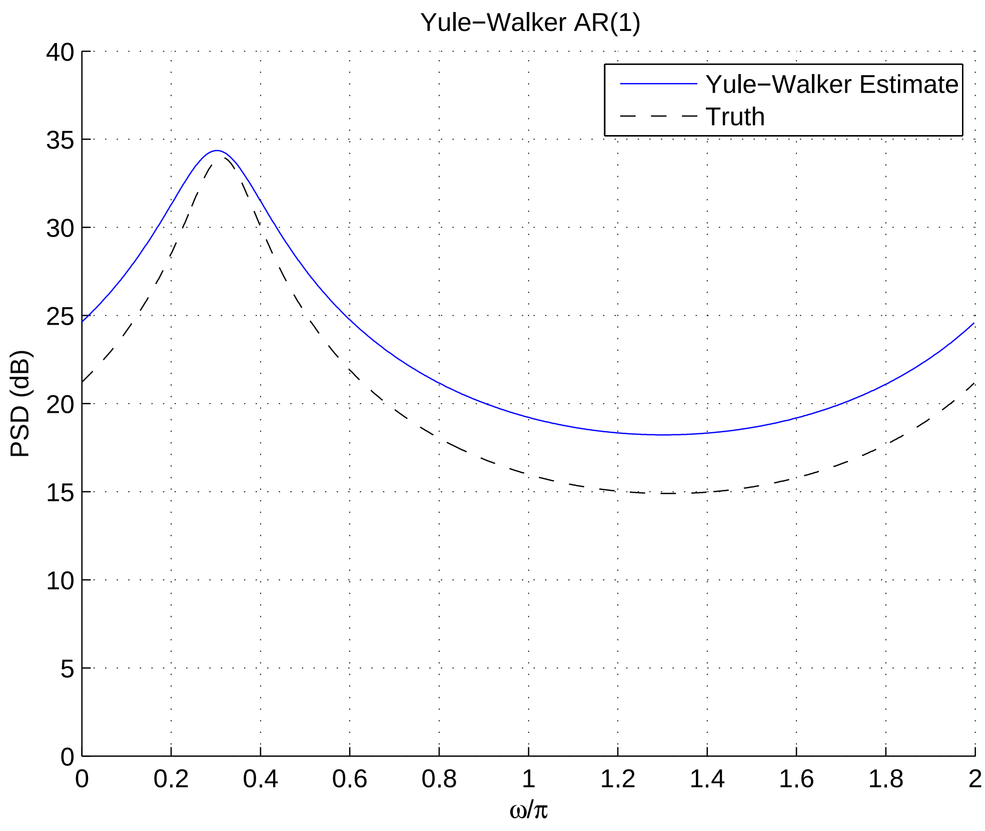

Figure 1: Using the Yule-Walker equations to estimate a signals autoregressive coefficients

The polynomials in the numerator and denominator can be factored and rewritten as

$$H(z) = \frac{\prod_{i=1}^{n}(1 - \beta_n z^{-n})}{\prod_{j=1}^{m}(1 - \alpha_m z^{-m})} \tag{3}$$

where the β_n are the zeros of the filter response and the α_m are the poles (where the filter response tends to ∞). This is thus often referred to as the pole-zero model. Using (1), the power spectral densities (PSD) of x and y are related as

$$\phi_y(z) = \left\| \frac{B(z)}{A(z)} \right\|^2 \phi_x(z) \tag{4}$$

It can be shown that any PSD can be approximated arbitrarily closely by a rational PSD which can be factored as [5, chap. 9]

$$\phi(z) = \left\| \frac{B(z)}{A(z)} \right\|^2 \sigma^2 \tag{5}$$

where σ^2 is an arbitrary constant. Comparing (4) and (5), we see that the PSD of a signal $y(t)$ can be thought of as the output PSD resulting from passing a signal with a PSD of σ^2, which corresponds to white noise with power σ^2, through a filter $H(z)$. Therefore, $y(t)$ itself can be modelled as a process that results from passing white noise through a rational filter $H(z)$, known as an ARMA process. By determining the coefficients of the filter $H(z)$, we can estimate the frequency content of $y(t)$.

The process of determining all of the coefficients of an ARMA model is a non-trivial problem. Even if the order (i.e.. n and m in (2) above) is known, obtaining the coefficients of $B(z)$, the moving-average (MA) component, is a non-linear estimation problem. However, we can simply ignore the MA component (i.e.. $B(z) = 1$) and still obtain good results, especially for signals which consist mainly of narrow peaks [5, chap. 9]. Estimating the AR coefficients can be done using the Yule-Walker equations, which have a fast solution algorithm known as Levinson-Durbin recursion. The details of this process is beyond the scope here, but the details are available in [5, chap. 9]. Instead, we will utilize the function MATLAB provides to calculate the AR coefficients using the Yule-Walker equations. Figure 1 shows the actual PSD of an AR(1) process and the PSD as estimated using the Yule-Walker equations.

2.2 Kalman Filter

The filter now known as the Kalman filter was first proposed by R. E. Kalman in 1960 [3]. The Kalman filter is a state-based filtering process, which means that it works by estimating the (hidden) state of the system, x, using direct and indirect observations, z, of this state. The filter can be implemented as a basic two step process:

- Propagate the current set of state variables (and their covariances, P) using a state transition matrix (or process model), M.
- Adjust the current state estimates based on new observations, using the observation matrix (or operator), H, to produce observation estimates from the state estimates.

In addition to the two matrices above, the Kalman filter also requires estimates of the model error (i.e.. the error in predicting the next system state from its current state using the specified model) as well as the error in the observations. By having these error covariances, the Kalman filter is able to produce a statistically optimal (minimum variance) estimate of the system state. Because the Kalman filter relies upon this matrix formulation, both the state transition and the observation operator are required to be linear. Extensions to the traditional Kalman filter, such as the Extended Kalman Filter (EKF) and the Unscented Kalman Filter have been developed to address this limitation of linearity. The EKF in particular has proven useful as the basis for finding locations using the Global Positioning System [1]. The following set of equations summarize the Kalman filter [4, chap. 27]. Given the following:

$$\begin{aligned}
\text{Model:}\quad & \mathbf{x_{k+1}} = \mathbf{M_k x_k} + \mathbf{w_{k+1}} \\
& E(\mathbf{w_k}) = 0 \\
& Cov(\mathbf{w_k}) = \mathbf{Q_k} \\
& Cov(\mathbf{x_0}) = \mathbf{P_0} \\[1em]
\text{Observation:}\quad & \mathbf{z_k} = \mathbf{H_k x_k} + \mathbf{v_{k+1}} \\
& E(\mathbf{v_k}) = 0 \\
& Cov(\mathbf{v_k}) = \mathbf{R_k} \\[1em]
\text{Initial Conditions:}\quad & \mathbf{\hat{x}_0} = E(\mathbf{x_0}) \\
& \mathbf{\hat{P}_0} = \mathbf{P_0}
\end{aligned} \tag{6}$$

the Kalman filter estimates the state as follows:

$$\begin{aligned}
\text{Forecast:}\quad & \mathbf{x_k^f} = \mathbf{M_{k-1}\hat{x}_{k-1}} \\
& \mathbf{P_k^f} = \mathbf{M_{k-1}\hat{P}_{k-1}M_{k-1}} + \mathbf{Q_k} \\[1em]
\text{Observation Correction:}\quad & \mathbf{\hat{x}_k} = \mathbf{x_k^f} + \mathbf{K_k}[\mathbf{z_k} - \mathbf{H_k x_k^f}] \\
& \mathbf{K_k} = \mathbf{P_k^f H_k^T}[\mathbf{H_k P_k^f H_k^T} + \mathbf{R_k}]^{-1} \\
& \mathbf{\hat{P}_k} = [\mathbf{I} - \mathbf{K_k H_k}]\mathbf{P_k^f}
\end{aligned} \tag{7}$$

(6) and (7) above boil down to the following steps:

- Begin with initial guesses for the state ($\hat{x}_0$) and the state covariance ($\hat{P}_0$).
- From the current estimate of the state, predict the next state using the model, M_k. Similarly, adjust the state covariance using the model and add the model error to the estimate of the state covariance.
- Adjust the estimate of the state using the available observations. This involves calculating observations from the current state estimate using the observation operator, H_k. The difference between the calculated and true observations is often called the "innovation vector". The estimate of the current state is corrected by this innovation vector weighted by K_f, which is known as the Kalman gain.
- The Kalman gain is essentially calculated as the ratio of the error in the state estimate to the sum of the error in the state estimate and the error in the observations. If the error in the state estimate is high, the Kalman gain is larger, giving new observations greater weight in adjusting the state estimate. Conversely, if the observation error is large, the Kalman gain is smaller, giving less adjustment to the state estimate.
- The state covariance matrix is adjusted using the Kalman gain.

The true importance here is that if we can formulate a model for our system or process of interest, the Kalman filter can combine observations with this model to yield more optimal estimates of the state of the system. In fact, the Kalman filter can be combined with the auto-regressive modelling discussed above to filter noise out of observations of an AR process.

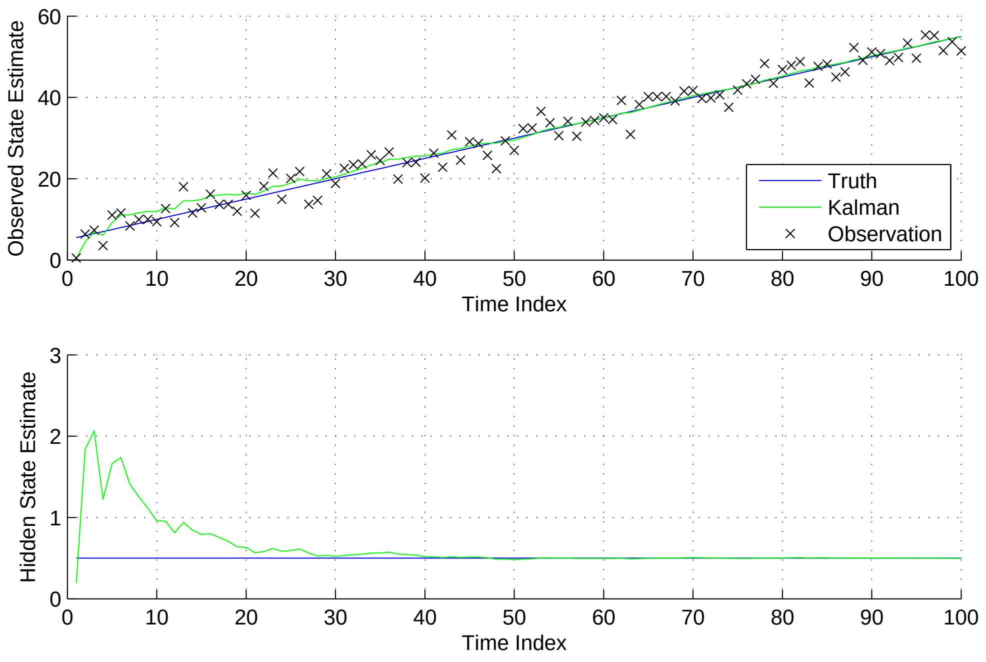

Figure 2: Top: The Kalman Filter estimates of a directly observed variable (position) compared with observations and the true value. Bottom: The Kalman Filter estimates of the hidden variable (velocity).

As an example of how to develop the Kalman filter for an application, we'll look at a simple linear model with two variables, only one of which will be observed. This could correspond, for instance, with using noisy observations of an object's position to make estimates of its true position and (c (constant) velocity. We start by writing the equations that define this system, where x_k and v_k are the object's position and velocity, respectively, at time k:

$$x_{k+1} = v_k \triangle t + x_k \tag{8}$$

$$v_{k+1} = v_k \tag{9}$$

We can write this in a matrix form compatible with (6) above as:

$$\mathbf{x_{k+1}} = \mathbf{M_k x_k} = \begin{bmatrix} \triangle t & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_k \\ x_k \end{bmatrix} \tag{10}$$

Since the model here is exact (given that the object has constant velocity), the covariance matrix for the model error is all zeros. For the observation operator, since we directly observe the position of the object, but do not observe the velocity, we have:

$$\mathbf{z_{k+1}} = \mathbf{H_k x_k} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} v_k \\ x_k \end{bmatrix} \tag{11}$$

The covariance matrix for the observations here, since we observe only a single scalar, becomes simply the variance of the position operators. All of this information together is sufficient to implement the Kalman filter for a set of observations. The results of using the Kalman filter on a set of generated position data is shown in Figure 2. This shows that after some initial time to settle down, the filter generates estimates of position closer to the true values than the observations. Also, while it is not directly observed (hence "hidden"), the filter's estimate of the velocity converges to the true value.

2.3 Collected Data

Time-series data were collected with the NOAA/NSSL research S-band radar (KOUN) on September 7, 2004 at 11 pm local time (04 UTC). The radar was in a dual polarization mode, thus simultaneously transmitting and receiving waves of horizontal and vertical polarizations. For each radial of data, 128 samples were collected at a pulse repetition time of 780μs. Consequently, the unambiguous range, R$_a$ and velocity, V$_a$ were 117 km and 35 m s^{-1}. Approximately 468 range gates per radial were collected.

The data are organized in the following format: The term raw is followed by the experiment number, which is then followed by the date. Next the time of day is given by UTC. A custom built VCP was prepared and denoted by a 5 digit code, which is known as 64091. Five elevations for this VCP were given and denoted by 01, 02 ,..., 05 corresponding to 0.5, 1.5, 2.5, 4, and 6 degrees. A full circle of radials were collected, providing 360 radials. These are denoted by the next three digit number. Finally, the azimuth angle is provided. Since the data are stepped in approximately 1.0 degree increments, the last number in the series follows a nice progression. The last number should be divided by 10 to arrive at the right angle. For example for radial 250, the azimuth is 356.5 degrees. The azimuth data begins with approximately 6 degrees of offset. Zero degrees corresponds to due north. In the data given below, each radial requires 944 KB of storage.

3 Hands-On Activities

1. To start, let's examine the performance of using the Yule-Walker equations to estimate the frequency peak of an AR(1) process, using MATLAB's `aryule` command.
   (a) Generate 128 samples of an AR(1) process with a single complex pole (for instance at $-0.4419 - 0.6666j$). This shape of this process's spectrum *roughly* resembles the Gaussian shape of a weather radar spectrum (a single broad peak). These samples can be generated by passing Gaussian white noise (see `randn`) of a given power (σ^2) through MATLAB's `filter` command. What are the filter coefficients in this simple case?
   (b) Now, pass your generated data into MATLAB's `aryule` command, which will return estimates of the filter coefficients and input noise power. How well do the estimated values compare with the true values? Do the estimates improve with increasing number of samples? What if you only have a few samples (say 32)?

(c) Compare the PSD from the estimate with the expected PSD for the process. Is there good general agreement, especially with regard to the location of the peak? Note that in MATLAB, `freqz` will give you the frequency response of a filter based on the coefficients. Combine this with the `aryule` output and equation (5) to get the PSD.

(d) Compare the pole-zero plots (really just pole plots) of the original filter and and that estimated from the Yule-Walker equations. (`zplane` will generate the plot for you. The pole is simply the second coefficient returned by `aryule`.) Is there good agreement in the location of the pole within the z-plane?

(e) Now repeat these steps, but add some white noise to the output of the `filter` command to simulate observation noise. Note that the ratio of the σ^2 of the original input white noise to that of the observation noise is equivalent to the signal-to-noise ratio (SNR). How does the performance of the Yule-Walker estimator change as a function of the SNR?

2. Next, we'll increase the complexity to using real data with multiple peaks. We can use the auto-regressive modelling to find multiple peaks within the PSD, which allows us to estimate velocity values even in the presence of clutter.

   (a) Download the compressed archive of data from http://www.ou.edu/radar/exp21(KOUNdata,sept2004,1800radials).zip

   (b) Examine the PSD's of the data in the lowest elevation cut to find a radar gate where there is strong mix of weather radar signal and ground clutter (i.e.. significant peaks at 0 and one other frequency). MATLAB's `periodogram` command will be helpful here.

   (c) Pass the time series data for this gate to the `aryule` command. *Be sure to use the appropriate order.*

   (d) Use `tf2zp` to convert the obtained filter coefficients to poles. How well do the frequency locations of the poles correspond to the peaks of the periodogram? (Hint: you'll need to use `angle` on the poles.)

   (e) Repeat this for a radar gate that has more than 2 peaks due to contamination from biological clutter.

3. Changing gears, we turn our attention to the Kalman filter. We'll start with a simple application.

   (a) Download the simple MATLAB Kalman filter code here: http://www.ou.edu/radar/kalman.m This function returns new estimates of the state and its covariance matrix given their initial values, the model and observation operators with their respective covariances, and an observation vector from a single time. To run this function on a set of observations, multiple iterations of calling the function are necessary.

   (b) Calculate a set of true values of position for an object moving with constant velocity. Add noise of a chosen variance (σ^2) to these truth values using `randn`.

   (c) Using (10) and (11) above, feed the generated observations into the Kalman filter function. Use the σ^2 from the noise above as the observation covariance. How does the filter perform at estimating the state? Is there good agreement with the truth values?

   (d) How does the performance of the filter change as you increase/decrease the variance in the observations? Is there some sensitivity to the initial values given to the filter?

   (e) How does the Kalman filter perform if you use an observation covariance value different from the true value? Is an accurate estimate of the error necessary to get good performance, or is an approximation sufficient? This is a realistic scenario, since we do not always know how much error/noise is present in our observations.

4. Now, let's use a simple application of the Kalman filter to some radar data, using the AR(1) model from above..

   (a) Find a radar gate that is well modelled as a single peak. Use `aryule` to estimate the parameters of the corresponding AR(1) process for this data.

   (b) Using the output of `aryule`, construct the relevant matrices for the Kalman filter and pass these into the `kalman` function. This will essentially work as a noise cancellation filter for the radar time series data. (*Hint:* for an AR(1) process, the new value is the old value scaled by the filter coefficient plus noise. This fits well with the model forecast equation used in the Kalman filter. Also, the state here is directly observed, so the observation operator is `[1.0]`. The real effort is finding a good estimate of the observation error, so that observations as correctly weighted.)

(c) Compare the periodograms of the raw and filtered data. How well does the Kalman filter work at removing noise?

(d) Can you derive the equations for the using the Kalman filter with an AR(2) process? (See [2, p.378] for a more thorogh derivation of the Kalman filter of an AR(1) process, which should serve as a guide for deriving the AR(2) version.)

References

[1] Grewal, M. S., L. R. Weill, and A. P. Andrews, *Global Positioning Systems, Inertial Navigation, and Integration*. 2ed. Hoboken: Wiley Interscience, 2007.

[2] Hayes, M. H., *Statistical Digital Signal Processing and Modelling*. Hoboken: John Wiley and Sons, 1996.

[3] Kalman, R. E., ”A New Approach to Linear Filtering and Prediction Problems.” *Trans. ASME, J. Basic Eng.*, Ser. 82D, pp. 35-45, March 1960.

[4] Lewis, J. M., S. Lakshmivarahan, and S. K. Dhall., *Dynamic Data Assimilation: A Least Squares Approach*. Cambridge: Cambridge University Press, 2006.

[5] Stoica, P. and R. Moses, *Spectral Analysis of Signals*. Upper Saddle River: Pearson Prentice Hall, 2005.