1 Lecture Outline

Reading: Chapter 5.

- Comments on autocorrelation method
- The Levinson recursion and its associated properties
- Lattice filter formulation of the inverse filter
- Linear Prediction Analysis of stochastic speech sounds

2 Review of All-pole Modeling of Deterministic Signals

Our basic source/filter model for a single impulse or periodic impulse train source input is shown below. We assume the transfer function from the glottis to the lips output is modeled as an all-pole filter

$$H(z) = \frac{A}{1 - \sum_{k=1}^{p} a_k z^{-k}}. \quad (1)$$

Figure 1: Source/filter model for impulse or impulse train source signal

3 Linear Prediction Analysis of Stochastic Speech Sounds

We motivated LPA with the observation that for a single impulse or periodic impulse train input to an all-pole vocal tract, the prediction error is zero "most of the time." Such analysis therefore appears not to be applicable to speech sounds with fricative or aspirated sources modeled as a stochastic process (often white noise). We will see that LPA can be formulated for the stochastic case where a white noise input takes on the role of the single impulse.

Consider a speech sound where turbulence is created at a constriction in the vocal tract. One model of the generation of this sound is the excitation of an all-pole vocal tract with white noise, $u_o[n]$ (zero mean, unit variance). Thus our autoregressive (AR) model is given by

$$\begin{aligned} S(z) &= \sum_{k=1}^{p} a_k S(z) z^{-k} + A U_o(z) \\ &\updownarrow \\ s[n] &= \sum_{k=1}^{p} a_k s[n-k] + A u_o[n]. \end{aligned} \quad (2)$$

In the unvoiced speech case, $u_o[n]$ is modeled as a zero mean, unit variance, white Gaussian noise sequence. The all-pole filter $H(z)$, defined by model parameters $\{a_k\}$, leads to a difference equation representation of the output

$$s[n] = \sum_{k=1}^{p} a_k s[n-k] + A u_o[n] \tag{3}$$

where $u_o[n]$ and $s[n]$ are sample functions of the random process. In this case, $s[n]$ is an IIR-filtered random sequence or autoregressive (AR) process. We can use previous definitions for linear predictor, one-step prediction, and prediction error in formulating the problem for finding the LPCs for stochastic speech sounds.

4 Problem Formulation and the Normal Equations

We define the prediction error

$$\begin{aligned} e[n] &= \text{actual} - \text{prediction} \\ &\equiv s[n] - \sum_{k=1}^{p} \alpha_k s[n-k]. \end{aligned} \tag{4}$$

For a short segment of N samples of unvoiced speech, find $\{\alpha_k\}$ that minimizes the mean-squared error over the segment

$$J = E\left\{e[n]^2\right\} \tag{5}$$

i.e. find $\{\alpha_k\}$ which leads to the minimum mean-squared prediction error. Thus we must find the vector of LPCs $\alpha = [\alpha_1, ..., \alpha_p]^T$ such that

$$J = E\left\{\left(s[n] - \alpha^T \mathbf{s}[n-1]\right)^2\right\} \tag{6}$$

is minimized. Here $\mathbf{s}[n-1] = [s[n-1], ..., s[n-p]]^T$ is the sample regressor. We expand:

$$J = E\left\{\left(s[n] - \alpha^T \mathbf{s}[n-1]\right)^2\right\}$$

$$J = E\left\{\left(s[n] - \alpha^T \mathbf{s}[n-1]\right)\left(s[n] - \mathbf{s}^T[n-1]\alpha\right\}\right]$$

$$J = E\left\{s[n]^2\right\} - 2\alpha^T E\left\{\mathbf{s}[n-1]s[n]\right\} + \alpha^T E\left\{\mathbf{s}[n-1]\mathbf{s}^T[n-1]\right\}\alpha$$

$$J = E\left\{s[n]^2\right\} - 2\alpha^T \mathbf{r} + \alpha^T \mathbf{R}\alpha \tag{7}$$

where the cross-correlation vector

$$\mathbf{r} \equiv E\left\{\mathbf{s}[n-1]s[n]\right\} \tag{8}$$

and the autocorrelation matrix.

$$\mathbf{R} \equiv E\left\{\mathbf{s}[n-1]\mathbf{s}^T[n-1]\right\}. \tag{9}$$

Define the k-th autocorrelation as

$$r[k] \equiv E\left\{s[n-k]s[n]\right\}. \tag{10}$$

The cross-correlation vector

$$\begin{aligned} \mathbf{r} &\equiv E\left\{\mathbf{s}[n-1]s[n]\right\} \\ &= \left[r[1], r[2], ..., r[p]\right]^T. \end{aligned} \tag{11}$$

The autocorrelation matrix is then a Toeplitz matrix

$$\begin{aligned} \mathbf{R} &\equiv E\left\{\mathbf{s}[n-1]\mathbf{s}^T[n-1]\right\} \\ &= \begin{bmatrix} r[0] & r[1] & \cdots & r[p-1] \\ r[1] & r[0] & \cdots & r[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r[p-1] & r[p-2] & \cdots & r[0] \end{bmatrix}. \end{aligned} \tag{12}$$

In order to minimize the MSE

$$J = E\left\{s[n]^2\right\} - 2\alpha^T\mathbf{r} + \alpha^T\mathbf{R}\alpha \tag{13}$$

we require the gradient vector to be zero

$$\begin{aligned} \mathbf{0} &= \partial J/\partial\alpha \\ &= -2\mathbf{r} + 2\mathbf{R}\alpha. \end{aligned} \tag{14}$$

We thus have the normal equations for the stochastic case

$$\mathbf{R}\alpha = \mathbf{r}. \tag{15}$$

The solution to the normal equations yields the LPCs

$$\alpha = \mathbf{R}^{-1}\mathbf{r} \tag{16}$$

where $\mathbf{R}$ is given by (12) and $\mathbf{r}$ is given by (11). Comments:

- For unvoiced speech, we can compute the parameters (LPCs)

$$\alpha = \mathbf{R}^{-1}\mathbf{r} \tag{17}$$

  of an all-pole filter which models the spectral envelope

$$\hat{H}(z) = \frac{A}{1 - \sum_{k=1}^{p} \alpha_k z^{-k}} \tag{18}$$

- In practice, $\mathbf{R}$ and $\mathbf{r}$ are estimated from the window of unvoiced speech samples

5 The Levinson Recursion and Its Associated Properties

The Levinson algorithm is implemented in MATLAB as

```
[alpha,e] = levinson(r,p)
```

Computation of the LPCs and gain is implemented in MATLAB as

```
[alpha,e] = lpc(s,p);
A = sqrt(e); (?)
```

We will use `lpc` throughout this course.

A direct matrix inversion of (12) is costly and requires $O(p^3)$ multiplies. The Levinson recursion, however, requires $O(p^2)$ multiplies and as we will see, leads to a direct link to the concatenated lossless tube model.

The basic idea is to build an order $i+1$ solution from an order i solution until we reach the desired model order p. Each recursion solves the ith pole problem finding the solution that minimizes the mean-squared prediction error (MSE) for each order predictor by updating the lower-order solution.

Recursion

Step 1: Initialize the prediction error and predictor coefficients of an order-zero predictor, i.e.

$$\begin{aligned} \alpha_0^0 &= 0 \\ E^0 &= r[0]. \end{aligned} \tag{19}$$

Because we assume the initial predictor is zero, i.e. $\alpha_k = 0$, the prediction sequence $\tilde{s}[n] = 0$. Thus MSE, (for simplicity we drop the $_n$)

$$\begin{aligned} E^0 &= \sum_{m=0}^{N_w+p-1} e^2[m] \\ &= \sum_{m=0}^{N_w+p-1} s^2[m] \\ &= r[0]. \end{aligned} \tag{20}$$

Step 2: Form a weighting factor (PARtial CORrelation coefficients) for the ith pole model as

$$k_i = \frac{1}{E^{i-1}} \left\{ r[i] - \sum_{j=1}^{i-1} \alpha_j^{i-1} r[i-j] \right\}. \tag{21}$$

The autocorrelations, $r[i]$ are computed from the (windowed) signal itself. (Later we will show how to compute the k_i without computing the autocorrelations.)

Step 3: Using step 2, update the coefficients for the ith pole model as

$$\begin{aligned} \alpha_i^i &= k_i \\ \alpha_i^j &= \alpha_j^{i-1} - k_i \alpha_{i-j}^{i-1}, \ 1 \le j \le i-1. \end{aligned} \tag{22}$$

Here the highest-order coefficient, α_i^i is set to k_i.

Step 4: Update the minimum MSE for the ith pole model as

$$E^i = \left(1 - k_i^2\right) E^{i-1}. \tag{23}$$

Step 5: Repeat steps 2-4 for $i = 1, 2, \ldots p$. At the pth step, we have the desired pth order predictor coefficients

$$\alpha_j^* = \alpha_j^p. \tag{24}$$

Here * denotes optimum and the desired all-pole model is given by

$$\begin{aligned} H(z) &= \frac{A}{A(z)} \\ &= \frac{A}{1-\sum_{k=1}^{p}\alpha_k^* z^{-k}}. \end{aligned} \tag{25}$$

(We still need to calculate the gain, A.)

6 Properties of Levinson

Property #1: The PARCOR coefficients, $|k_i| < 1$.

Property #2: The roots of the predictor polynomial, $A(z)$ lie strictly inside the unit circle. Thus the poles of $H(z)$ are strictly inside the unit-circle and $H(z)$ is minimum phase. This can be shown using Property #1. Because of this, Levinson can also be used as a general stability test for a filter $H(z)$ and is cheaper than factoring. To do so, start with the filter coefficients, α_k and run the recursions backward to get k_i. If $|k_i| < 1$, then the roots of $A(z)$ lie strictly inside the unit circle. This is equivalent to running a Jury stability test.

Property #5 (Autocorrelation Matching): Consider a measurement $s_n[m] = s[n+m]w[m]$ with autocorrelation, $r_n[\tau]$. Let $h[n]$ denote the impulse response associated with the estimated transfer function $H(z)$ of order p obtained from the autocrrelation method of linear prediction using the measurement, $s_n[m]$. Denote the autocorrelation of $h[n]$ by $r_h[\tau]$. Suppose the energy in $h[n]$ equals the energy in $s_n[m]$, i.e. $r_h[0] = r_n[0]$. Then

$$r_h[\tau] = r_n[\tau], \ |\tau| \leq p. \tag{26}$$

We will use Property #5 to compute the filter gain A.

6.1 Computation of Gain

One approach to compute the gain A is to require that the energy in the all-pole impulse response $h[n]$ equals the energy in the measurement $s[n]$, i.e. $r_h[0] = r_n[0]$. This constraint results in matching the autocorrelation of the two sequences for $|\tau| \leq p$.

We begin with the all-pole transfer function,

$$H(z) = \frac{A}{1-\sum_{k=1}^{p}\alpha_k z^{-k}} \tag{27}$$

where α_k are computed with Levinson using $r_n[\tau]$. With (27), we have

$$h[m] = \sum_{k=1}^{p}\alpha_k h[m-k] + A\delta[m]. \tag{28}$$

Note the since we assume $h[m]$ is causal, $h[0] = A$. Multiplying (28) with itself and summing over all m results in

$$\begin{aligned}\sum_{m=-\infty}^{\infty} h[m]h[m] &= \sum_{k=1}^{p}\alpha_k \sum_{m=-\infty}^{\infty} h[m-k]h[m] + A\sum_{m=-\infty}^{\infty}\delta[m]h[m] \\ &= \sum_{k=1}^{p}\alpha_k \sum_{m=-\infty}^{\infty} h[m-k]h[m] + A^2. \end{aligned} \tag{29}$$

We compactly express (29) as

$$r_h[0] = \sum_{k=1}^{p}\alpha_k r_h[k] + A^2. \tag{30}$$

Under the equal energy assumption, $r_h[0] = r_n[0]$, Property #5 holds and (30) becomes

$$A^2 = r_n[0] - \sum_{k=1}^{p}\alpha_k r_n[k] \tag{31}$$

or

$$A = \left\{ r_n[0] - \sum_{k=1}^{p}\alpha_k r_n[k] \right\}^{1/2}. \tag{32}$$

Using speech samples and the associated autocorrelation sequence, we now can use Levinson to compute the LPC coefficients, $\{\alpha_k\}$ and (32) to compute A in

$$H(z) = \frac{A}{1 - \sum_{k=1}^{p}\alpha_k z^{-k}}. \tag{33}$$

7 Lattice Filter Formulation of the Inverse Filter

The lattice filter formulation has an implementation advantage compared to other direct form methods. The advantage lies in fixed-point representation of reflection coefficients (Levinson property #1) and robustness to coefficient quantization.

The pth order model is given by

$$\begin{aligned} H(z) &= \frac{A}{A(z)} \\ &= \frac{A}{1 - \sum_{k=1}^{p}\alpha_k z^{-k}} \end{aligned} \tag{34}$$

where $A(z)$ is referred to as the *inverse filter* or *prediction error filter*.

Consider the ith order prediction error filter

$$A(z) = 1 - \sum_{k=1}^{i}\alpha_k^{(i)} z^{-k} \tag{35}$$

which we can think of as the ith stage of the Levinson recursion. The output of $A^{(i)}(z)$ for the input $s_n[m]$ is (for convenience we drop $_n$)

$$e^{(i)}[m] = s[m] - \sum_{k=1}^{i} \alpha_k^{(i)} s[m-k]. \tag{36}$$

We can think of $e^{(i)}[m]$ as the *forward prediction error* for the ith order prediction error filter, i.e. we predict $s[m]$ forward from the past samples $s[m-i], ..., s[m-1]$.

Figure 2: Forward and backward prediction errors

Now define the *backward prediction error* as

$$b^{(i)}[m] = s[m-i] - \sum_{k=1}^{i} \alpha_k^{(i)} s[m-i+k] \tag{37}$$

which represents the prediction of $s[m-i]$ backward from the future samples $s[m], ..., s[m-i+1]$. With some work we have the following relations

$$e^i[m] = e^{i-1}[m] - k_i b^{i-1}[m-1] \tag{38}$$
$$b^i[m] = b^{i-1}[m-1] - k_i e^{i-1}[m] \tag{39}$$

which represent the forward and backward prediction errors for the ith order predictor in terms of the prediction errors for the $i-1$-th order predictor. The relations are jump-started with the 0th order predictor giving

$$e^0[m] = s[m] \tag{40}$$
$$b^0[m] = s[m]. \tag{41}$$

The equations are graphically represented with a lattice network. The PARCOR coefficients k_i are obtained

Figure 3: Figure 5.9b

with the Levinson recursion. As it turns out these can also be computed as

$$k_i = \frac{\sum_{n=-\infty}^{\infty} e^{i-1}[n] b^{i-1}[n]}{\sum_{n=-\infty}^{\infty} \left(e^{i-1}[n]\right)^2 \sum_{n=-\infty}^{\infty} \left(b^{i-1}[n]\right)^2} \tag{42}$$

which does not involve the autocorrelations and these can be used in Levinson.

8 Demo of LPA

Demo of LPA (Step 1):

Create a filter with resonances (formants) at 500, 1000, 2000, and 4000Hz.

```
% Build the transfer function
fs = 16000;  % sample rate
w = 2*pi*[500;1000;2000;4000]/fs;  % pole angles
p = 0.99*exp(j*w);  % poles
p = [p;conj(p)];  %  4 pole pairs = 8 poles
a = poly(p);  % get filter coefs
polezero(1,a)
```

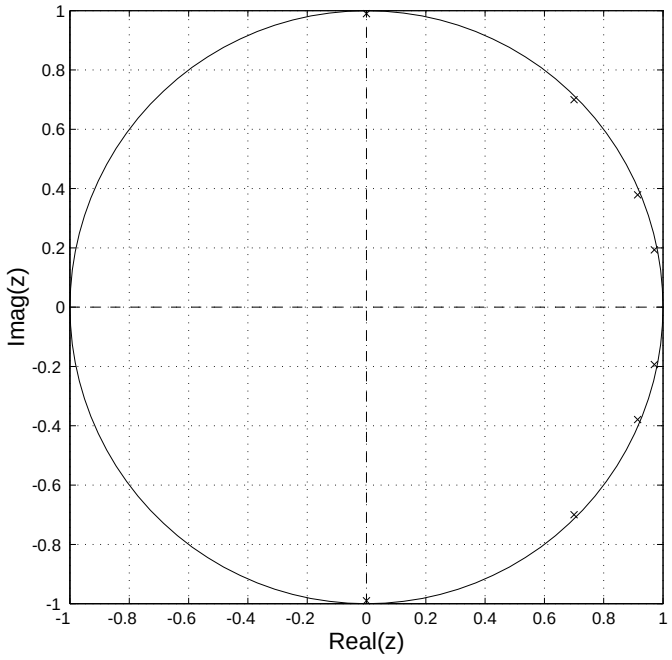

Figure 4: Spectrogram (wideband) of a chirp

Demo of LPA (Step 2): First, filter zero mean, unit-variance white Gaussian noise through filter

$$H(z) = 1/A(z) = 1/\left(1 - \sum_{k=1}^{p} a_k z^{-k}\right) \tag{43}$$

where the a_k are given in the previous step. Second, compute periodogram (DSP Toolkit).

```
N = 3*fs;
s = ar_synthesizer(a,N,1); % zero mean, unit-variance input
S = periodogram(s);
periodogram_plot(S,0,fs);
```

Demo of LPA (Step 2 Alternate): Same as prior slide but compute periodogram with Welch's method (DSP Toolkit)

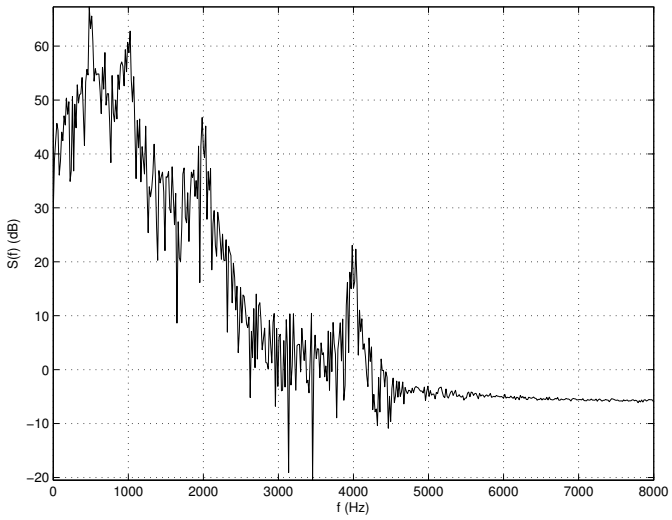

Figure 5: Spectrogram (wideband) of a chirp

```
N = 3*fs;
s = ar_synthesizer(a,N,1); % zero mean, unit-variance input
Nw = 1024;window = hamming(Nw);
S = welch2(s,window,Nw/2);
periodogram_plot(S,0,fs);
```

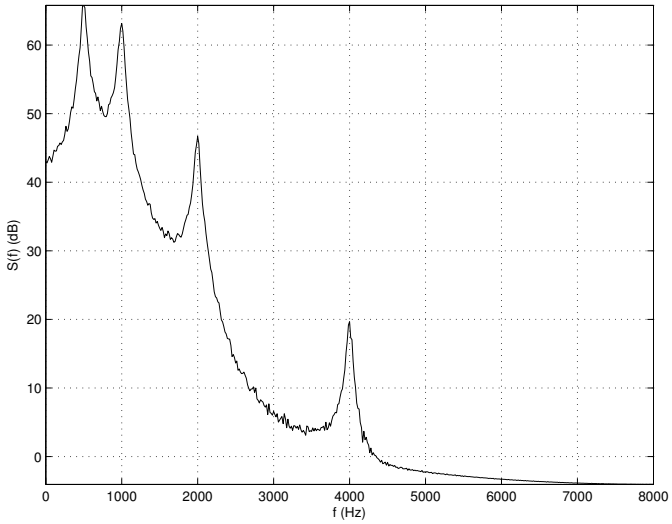

Figure 6: Spectrogram (wideband) of a chirp

Demo of LPA (Step 2 Alternate): Same as prior slide but compute periodogram with Welch's method (MATLAB SP Toolbox)

```
N = 3*fs;
s = ar_synthesizer(a,N,1); % zero mean, unit-variance input
Nw = 1024;window = hamming(Nw);
[S,f] = pwelch(s,window,Nw/2,Nw,fs);
S = S*fs; % undo MATLAB sample rate normalization
```

```
S = [S;S(512:-1:2)]; % pwelch only returns 0 <= f <= fs/2
periodogram_plot(S,0,fs);
```

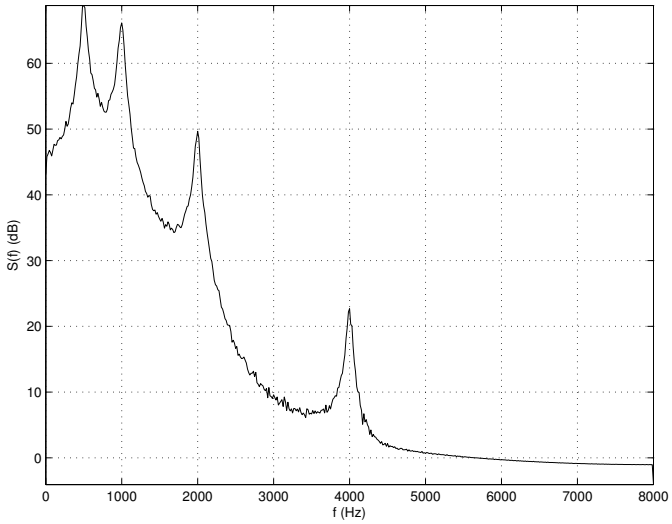

Figure 7: Spectrogram (wideband) of a chirp

Demo of LPA (Step 3):

```
p = 20;  % model order
```

Build a model of the AR process using levinson2 (Toolkit)...

```
[alpha,A,k] = levinson2(s,p); % DSP Toolkit
alpha = [1;-alpha];
```

...or using levinson (MATLAB SP Toolbox)

```
r = xcorr(s,p,'biased');
r = r(p+1:2*p+1);
[alpha,e,k] = levinson(r,p); % build model using MATLAB's levinson
A = sqrt(e); % filter gain
```

...or using lpc (MATLAB SP Toolbox)

```
[alpha,e] = lpc(s,p); % build model using MATLAB's lpc
A = sqrt(e); % filter gain
```

Plot the original power spectrum and magnitude response of the model (next slide)

```
[H,f] = freqz(A,alpha,N,fs);  % response of model
hold on;plot(f,20*log10(abs(H)),'r');hold off;  % overlay
```

Demo of LPA (Step 3) cont:

Since we are dealing with a random process, your mileage (plots) will vary.

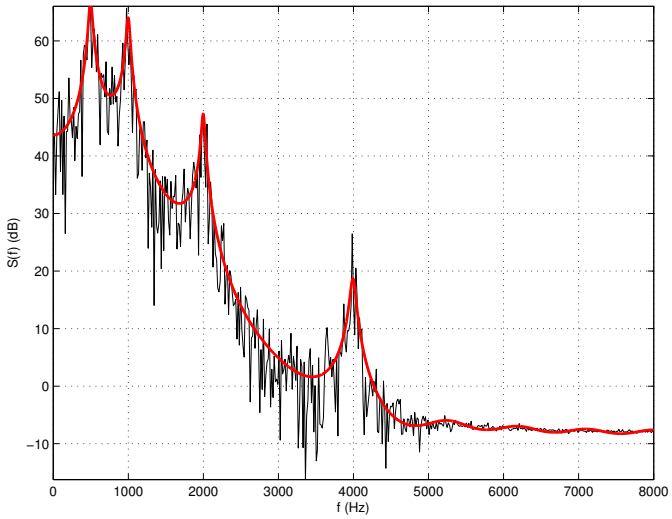

Figure 8: Spectrogram (wideband) of a chirp