Lecture 11: Predictive Coding of Speech at Low Bit Rates, Background: LPC

Mark Hasegawa-Johnson

ECE 537: Speech Processing, Fall 2022

Outline

IIR Filter

Let's start with a general second-order IIR filter, which you would implement in one line of python like this:

$$y[n] = x[n] + a_1 y[n-1] + a_2 y[n-2]$$

By taking the Z-transform of both sides, and solving for $Y(z)$, you get

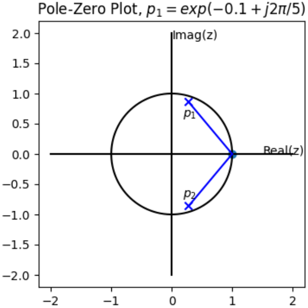
$$H(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{1}{(1 - p_1 z^{-1})(1 - p_1^* z^{-1})},$$

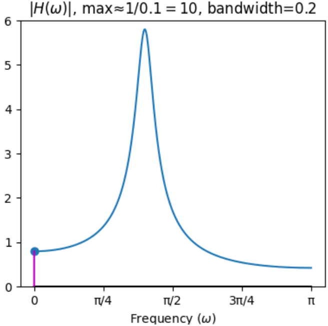
where p_1 and p_1^* are the roots of the polymomial $z^2 - a_1 z - a_2$. (For the rest of this lecture, we'll assume that the polynomial has complex roots, because that's the hardest case).

Frequency Response of an All-Pole Filter

We get the magnitude response by just plugging in $z = e^{j\omega}$, and taking absolute value:

$$|H(\omega)| = |H(z)|_{z=e^{j\omega}} = \frac{\left|e^{2j\omega}\right|}{\left|e^{j\omega} - p_1\right| \times \left|e^{j\omega} - p_1^*\right|}$$





Outline

Inverse Z transform

Suppose you know $H(z)$, and you want to find $h[n]$. How can you do that?

How to find the inverse Z transform

Any IIR filter $H(z)$ can be written as...

- a **sum** of **exponential** terms, each with this form:

$$G_\ell(z) = \frac{1}{1 - az^{-1}} \quad \leftrightarrow \quad g_\ell[n] = a^n u[n],$$

- each possibly **multiplied** by a **delay** term, like this one:

$$D_k(z) = b_k z^{-k} \quad \leftrightarrow \quad d_k[n] = b_k \delta[n-k].$$

How to find the inverse Z transform

Remember that multiplication in the frequency domain is convolution in the time domain, so

$$
\begin{aligned}
b_k z^{-k} \frac{1}{1 - az^{-1}} &\leftrightarrow (b_k \delta[n-k]) * (a^n u[n]) \\
&= b_k a^{n-k} u[n-k]
\end{aligned}
$$

Step #1: The Products

So, for example,

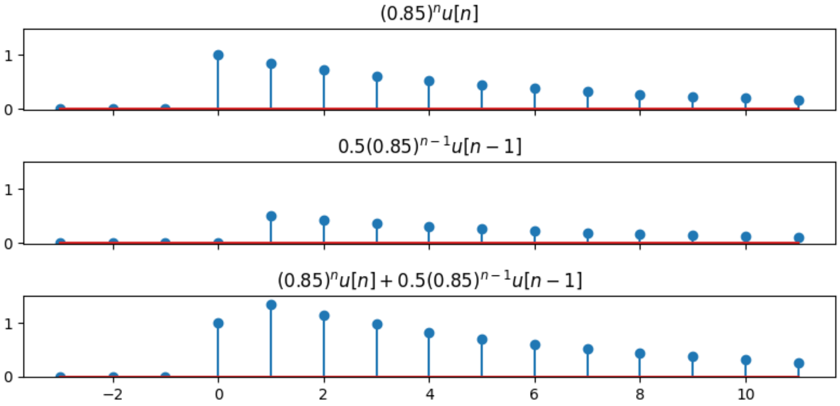
$$H(z) = \frac{1+bz^{-1}}{1-az^{-1}} = \left(\frac{1}{1-az^{-1}}\right) + bz^{-1}\left(\frac{1}{1-az^{-1}}\right)$$

and therefore

$$h[n] = a^n u[n] + ba^{n-1}u[n-1]$$

Step #1: The Products

So here is the inverse transform of $H(z) = \frac{1+0.5z^{-1}}{1-0.85z^{-1}}$:



Step #1: The Products

In general, if

$$G(z) = \frac{1}{A(z)}$$

for any polynomial $A(z)$, and

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{A(z)}$$

then

$$h[n] = b_0 g[n] + b_1 g[n-1] + \cdots + b_M g[n-M]$$

Step #2: The Sum

Now we need to figure out the inverse transform of

$$G(z) = \frac{1}{A(z)}$$

You already know it for the first-order case ($A(z) = 1 - az^{-1}$).
What about for the general case?

Step #2: The Sum

The method is this:

1. Factor $A(z)$:
$$G(z) = \frac{1}{\prod_{\ell=1}^{N}\left(1 - p_\ell z^{-1}\right)}$$
2. Assume that $G(z)$ is the sum of first-order fractions:
$$G(z) = \frac{C_1}{1 - p_1 z^{-1}} + \frac{C_2}{1 - p_2 z^{-1}} + \cdots$$
3. Find the constants, C_ℓ, that make the equation true.
4. . . . and the inverse Z transform is
$$g[n] = C_1 p_1^n u[n] + C_2 p_2^n u[n] + \cdots$$

Outline

A General Second-Order IIR Filter

Suppose we have a general second-order IIR filter:

$$y[n] = x[n] + a_1 y[n-1] + a_2 y[n-2]$$

Its Z-transform is

$$\begin{aligned} Y(z) &= X(z) + a_1 z^{-1} Y(z) + a_2 z^{-2} Y(z) \\ &= \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} X(z) \end{aligned}$$

So, if p_1 and p_1^* are the roots of the quadratic,

$$H(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2}} = \frac{1}{(1 - p_1 z^{-1})(1 - p_1^* z^{-1})}$$

Partial Fraction Expansion

In order to find the impulse response, we do a partial fraction expansion:

$$H(z) = \frac{1}{(1-p_1z^{-1})(1-p_1^*z^{-1})} = \frac{C_1}{1-p_1z^{-1}} + \frac{C_2}{1-p_1^*z^{-1}}$$

When we multiply both sides by the denominator, we get:

$$1 = C_1(1-p_1^*z^{-1}) + C_2(1-p_1z^{-1})$$

Notice that the above equation is actually two equations: $1 = C_1 + C_2$, and $0 = C_1p_1^* + C_2p_1$. Solving those two equations, we get,

$$C_1 = \frac{p_1}{p_1 - p_1^*}, \quad C_2 = \frac{p_1^*}{p_1^* - p_1}$$

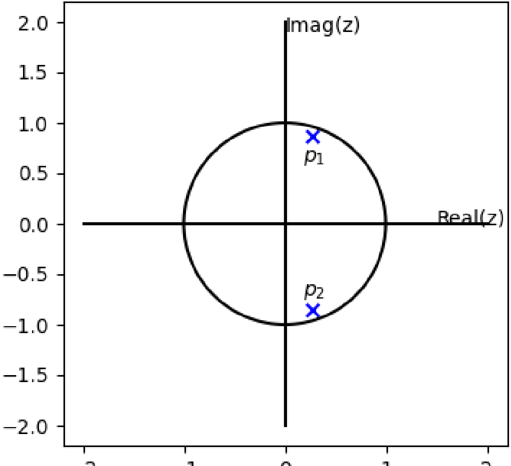
Impulse Response of a Second-Order IIR

. . . and so we just inverse transform.

$$h[n] = C_1 p_1^n u[n] + C_1^*(p_1^*)^n u[n]$$

Example: Causal Stable IIR Filter

Let's assume that the filter is causal and stable, meaning that p_1 is inside the unit circle, $p_1 = e^{-\sigma_1 + j\omega_1}$.



Example: Stable Resonator

Remember that p_1 and p_1^* are the zeros of a polynomial whose coefficients are a_1 and a_2:

$$H(z) = \frac{1}{(1-p_1z^{-1})(1-p_1^*z^{-1})} = \frac{1}{1-a_1z^{-1}-a_2z^{-2}},$$

so

$$\begin{aligned} a_1 &= 2e^{-\sigma_1}\cos\omega_1 \\ a_2 &= -e^{-2\sigma_1} \end{aligned}$$

Impulse Response of a Causal Stable Filter

To find the impulse response, we just need to find the constants in the partial fraction expansion. Those are

$$C_1 = \frac{p_1}{p_1 - p_1^*} = \frac{p_1}{e^{-\sigma_1}\left(e^{j\omega_1} - e^{-j\omega_1}\right)} = \frac{e^{j\omega_1}}{2j\sin(\omega_1)}$$

and

$$C_1^* = -\frac{e^{-j\omega_1}}{2j\sin(\omega_1)}$$

Impulse Response of a Second-Order IIR

Plugging in to the impulse response, we get

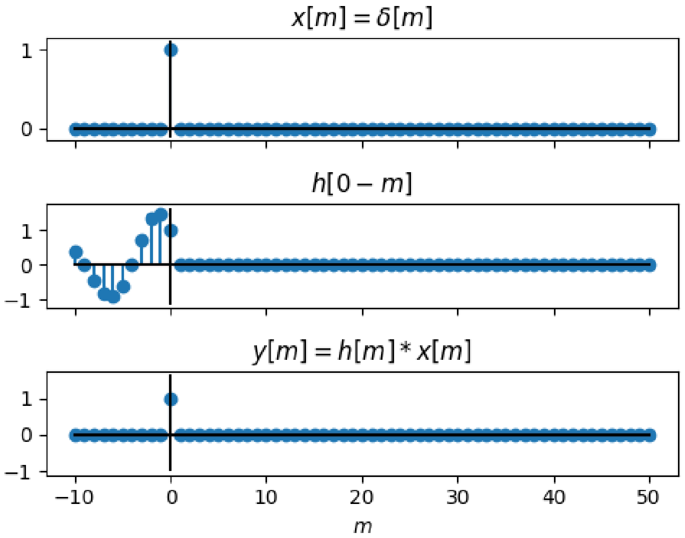
$$\begin{aligned}
h[n] &= C_1 p_1^n u[n] + C_1^* (p_1^*)^n u[n] \\
&= \frac{1}{2j\sin(\omega_1)} \left(e^{j\omega_1} e^{(-\sigma_1 + j\omega_1)n} - e^{-j\omega_1} e^{(-\sigma_1 - j\omega_1)n} \right) u[n] \\
&= \frac{1}{2j\sin(\omega_1)} e^{-\sigma_1 n} \left(e^{j\omega_1(n+1)} - e^{-j\omega_1(n+1)} \right) u[n] \\
&= \frac{1}{\sin(\omega_1)} e^{-\sigma_1 n} \sin(\omega_1(n+1)) u[n]
\end{aligned}$$

Impulse Response of a Second-Order IIR

$$h[n] = 2|C_1|e^{-\sigma_1 n}\sin(\omega_1(n+1))u[n]$$

- The constant is $2|C_1| = 1/\sin\omega_1$. It's just a scaling constant, it's not usually important to remember what it is.
- The $e^{-\sigma_1 n}\sin(\omega_1 n)u[n]$ part is what's called a "damped sinusoid," meaning a sinusoid that decays exponentially fast as a function of time. That's really the most important part of this equation.
- The fact that it's $\sin(\omega_1(n+1))$ instead of $\sin(\omega_1 n)$ is not really very important, but if you want, you can remember that it's necessary because, at $n=0$, $\sin(\omega_1 n)=0$, but $\sin(\omega_1(n+1))\neq 0$.

Impulse Response of a Second-Order IIR



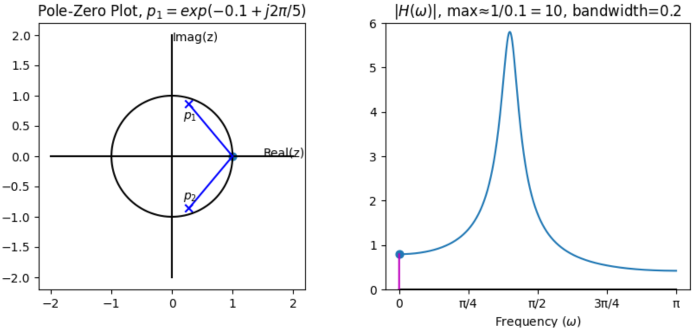
A Damped Resonator is Stable

A damped resonator is stable: any finite input will generate a finite output.

$$H(\omega) = H(z)|_{z=e^{j\omega}} = \frac{1}{(1-e^{-\sigma_1+j(\omega_1-\omega)})(1-e^{-\sigma_1+j(-\omega_1-\omega)})}$$

The highest peak of the frequency response occurs at $\omega \approx \pm\omega_1$, where you get

$$H(\omega_1) = \frac{1}{(1-e^{-\sigma_1})(1-e^{-\sigma_1-2j\omega_1})} \approx \frac{1}{1-e^{-\sigma_1}} \approx \frac{1}{\sigma_1}$$



Pole-Zero Plot, $p_1 = exp(-0.1 + j2\pi/5)$
Imag(z)
Real(z)
p_1
p_2
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
−1
0
1
2
$|H(\omega)|$, max≈1/0.1 = 10, bandwidth=0.2
6
5
4
3
2
1
0
π/4
π/2
3π/4
π
Frequency (ω)

Outline

Speech

Voiced speech is made when your vocal folds snap shut, once every 5-10ms. The snapping shut of the vocal folds causes a negative spike in the air pressure just above the vocal folds, like this:

$$e[n] = G\delta[n-n_0] + G\delta[n-n_0-T_0] + G\delta[n-n_0-2T_0] + \cdots$$

where T_0 is the pitch period (5-10ms), n_0 is the time alignment of the first glottal pulse, G is some large negative number, and I'm using $e[n]$ to mean "the speech excitation signal."

Speech

The speech signal echoes around inside your vocal tract for awhile, before getting radiated out through your lips. So we can model speech as

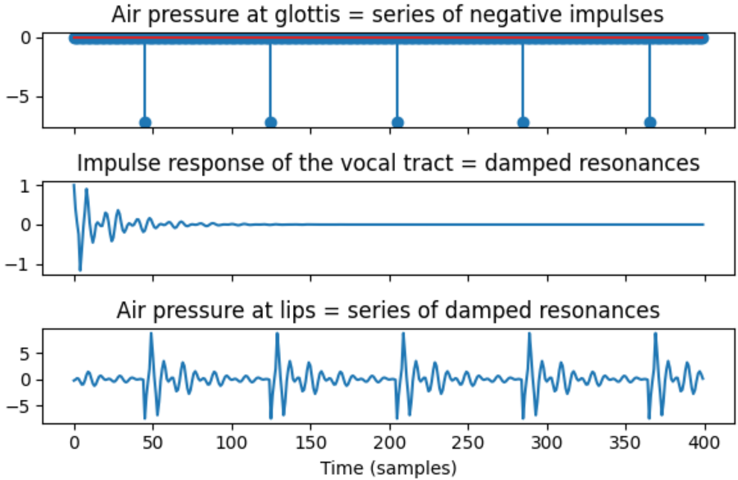
$$s[n] = e[n] + a_1 s[n-1] + a_2 s[n-2] + \cdots$$

where $a_1, a_2, \ldots$ are the reflection coefficients inside the vocal tract, and $s[n]$ is the speech signal. In the frequency domain, we have

$$S(z) = H(z)E(z) = \frac{1}{A(z)}E(z) = \frac{1}{1 - \sum_m a_m z^{-1}}E(z)$$

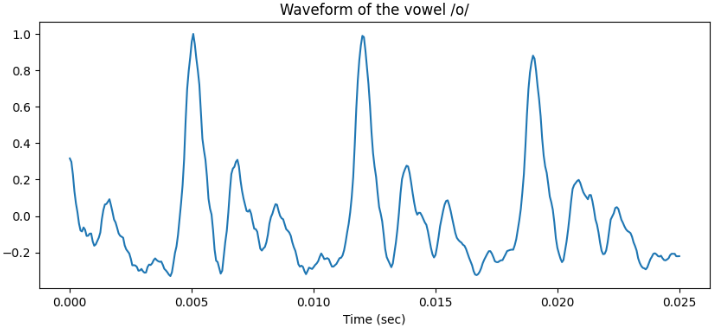
Speech: The Model

Speech is made when we take a series of impulses, one every 5-10ms, and filter them through a resonant cavity (like a bell).



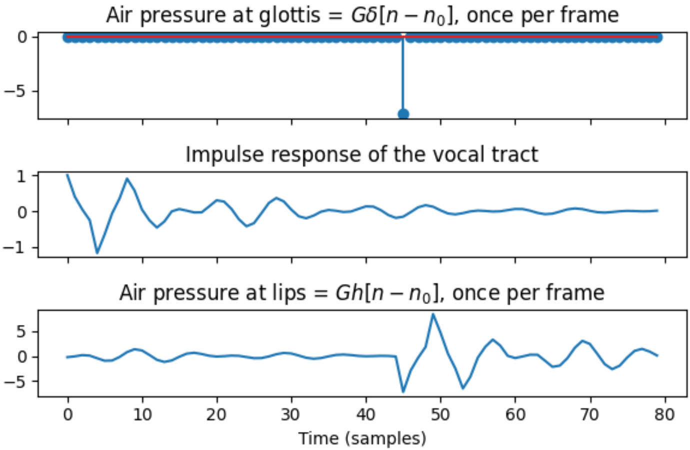
Speech: The Real Thing

For example, here's a real speech waveform (the vowel /o/):



Speech: The Model

Here's the model again, zoomed in on just one glottal pulse:



Inverse Filtering

If $S(z) = E(z)/A(z)$, then we can get $E(z)$ back again by doing something called an **inverse filter:**

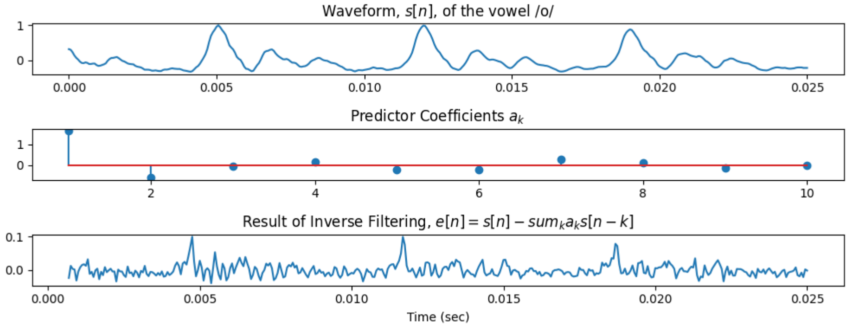
$$\textbf{IF: } S(z) = \frac{1}{A(z)}E(z) \quad \textbf{THEN: } E(z) = A(z)S(z)$$

The inverse filter, $A(z)$, has a form like this:

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

where p is twice the number of resonant frequencies. So if speech has 4-5 resonances, then $p \approx 10$.

Inverse Filtering



Inverse Filtering

This one is an all-pole (feedback-only) filter:

$$S(z) = \frac{1}{1-\sum_{k=1}^{p} a_k z^{-k}} E(z)$$

That means this one is an all-zero (feedforward only) filter:

$$E(z) = \left(1 - \sum_{k=1}^{p} a_k z^{-k}\right) S(z)$$

which we can implement just like this:

$$e[n] = s[n] - \sum_{k=1}^{p} a_k s[n-k]$$

Outline

Linear Predictive Analysis

This particular feedforward filter is called **linear predictive analysis**:

$$e[n] = s[n] - \sum_{k=1}^{p} a_k s[n-k]$$

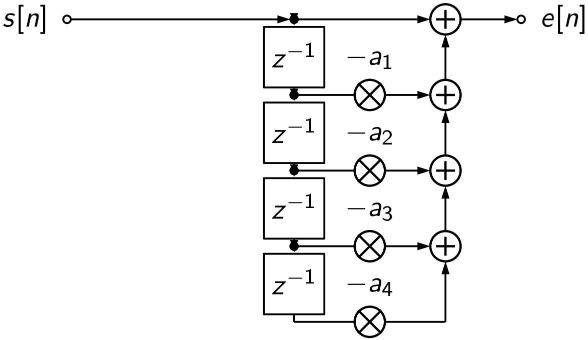
It's kind of like we're trying to predict $s[n]$ using a linear combination of its own past samples:

$$\hat{s}[n] = \sum_{k=1}^{p} a_k s[n-k],$$

and then $e[n]$, the glottal excitation, is the part that can't be predicted:

$$e[n] = s[n] - \hat{s}[n]$$

Linear Predictive Analysis Filter



Linear Predictive Synthesis

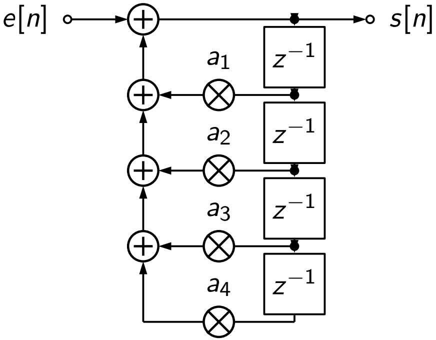
The corresponding feedback filter is called **linear predictive synthesis**. The idea is that, given $e[n]$, we can resynthesize $s[n]$ by adding feedback, because:

$$S(z) = \frac{1}{1-\sum_{k=1}^{p} a_k z^{-k}} E(z)$$

means that

$$s[n] = e[n] + \sum_{k=1}^{p} a_k s[n-k]$$

Linear Predictive Synthesis Filter



Outline

Finding the Linear Predictive Coefficients

Things we don't know:

- The timing of the unpredictable event (n_0), and its amplitude (G).
- The coefficients a_k.

It seems that, in order to find n_0 and G, we first need to know the predictor coefficients, a_k. How can we find a_k?

Finding the Linear Predictive Coefficients

Let's make the following assumption:

- Everything that can be predicted is part of $\hat{s}[n]$. Only the unpredictable part is $e[n]$.

Finding the Linear Predictive Coefficients

Let's make the following assumption:

- Everything that can be predicted is part of $\hat{s}[n]$. Only the unpredictable part is $e[n]$.
- So we define $e[n]$ to be:

$$e[n] = s[n] - \sum_{k=1}^{p} a_k s[n-k]$$

- ...and then choose a_k to make $e[n]$ as small as possible.

$$a_k = \text{argmin} \sum_{n=-\infty}^{\infty} e^2[n]$$

Finding the Linear Predictive Coefficients

So we've formulated the problem like this: we want to find a_k in order to minimize:

$$\mathcal{E} = \sum_{n=p+1}^{N+p} e^2[n] = \sum_{n=p+1}^{N+p} \left(s[n] - \sum_{m=1}^{p} a_m s[n-m] \right)^2$$

The Orthogonality Principle

If we differentiate $d\mathcal{E}/da_k$, we get

$$\frac{d\mathcal{E}}{da_k} = 2\sum_{n=p+1}^{N+p}\left(s[n] - \sum_{m=1}^{p} a_m s[n-m]\right) s[n-k] = 2e[n]s[n-k]$$

If we then set the derivative to zero, we get what's called the **orthogonality principle**. The orthogonality principle says that the optimal coefficients, a_k, make the error **orthogonal** to the predictor signal ($e[n] \perp s[n-k]$), by which we mean that

$$0 = \sum_{n=p+1}^{N+p} e[n]s[n-k] \quad \text{for all } 1 \le k \le p$$

This is a set of p linear equations (for $1 \le k \le p$) in p different unknowns (a_k). So it can be solved.

Autocorrelation

In order to write the solution more easily, let's define something called the "autocovariance," $\phi(i,k)$:

$$\phi(i,k) = \sum_{n=p+1}^{N+p} s[n-i]s[n-k]$$

In terms of the autocorrelation, the orthogonality equations are

$$0 = \phi(0,k) - \sum_{m=1}^{p} a_m \phi(m,k) \quad \forall\, 1 \le k \le p$$

which can be re-arranged as

$$\phi(0,k) = \sum_{m=1}^{p} a_m \phi(m,k) \quad \forall\, 1 \le k \le p$$

Matrices

Since we have p linear equations in p unknowns, let's write this as a matrix equation:

$$\begin{bmatrix} \phi(0,1) \\ \phi(0,2) \\ \vdots \\ \phi(0,p) \end{bmatrix} = \begin{bmatrix} \phi(1,1) & \phi(1,2) & \cdots & \phi(1,p) \\ \phi(2,1) & \phi(2,2) & \cdots & \phi(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(p,1) & \phi(p,2) & \cdots & \phi(p,p) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}$$

Notice that this matrix is symmetric:

$$\phi(i,k) = \phi(k,i) = \sum_{n=p+1}^{N+p} s[n-i]s[n-k]$$

Matrices

Since we have p linear equations in p unknowns, let's write this as a matrix equation:

$$\vec{c} = \Phi\vec{a}$$

where

$$\vec{c} = \begin{bmatrix} \phi(0,1) \\ \phi(0,2) \\ \vdots \\ \phi(0,p) \end{bmatrix}, \quad \Phi = \begin{bmatrix} \phi(1,1) & \phi(1,2) & \cdots & \phi(1,p) \\ \phi(2,1) & \phi(2,2) & \cdots & \phi(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(p,1) & \phi(p,2) & \cdots & \phi(p,p) \end{bmatrix}.$$

Matrices

Since we have p linear equations in p unknowns, let's write this as a matrix equation:

$$\vec{c} = \Phi\vec{a}$$

and therefore the solution is

$$\vec{a} = \Phi^{-1}\vec{c}$$

Finding the Linear Predictive Coefficients

So here's the way we perform linear predictive analysis:

1. Create the matrix Φ and vector $\vec{c}$:

$$\vec{c} = \begin{bmatrix} \phi(0,1) \\ \phi(0,2) \\ \vdots \\ \phi(0,p) \end{bmatrix}, \quad \Phi = \begin{bmatrix} \phi(1,1) & \phi(1,2) & \cdots & \phi(1,p) \\ \phi(2,1) & \phi(2,2) & \cdots & \phi(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(p,1) & \phi(p,2) & \cdots & \phi(p,p) \end{bmatrix}.$$

2. Invert Φ.

$$\vec{a} = \Phi^{-1}\vec{c}$$

Autocorrelation versus Covariance Methods

- The method I've just described is called the **covariance method** for solving LPC. It requires inverting Φ (or, equivalently, finding its Cholesky decomposition) which is an $\mathcal{O}\{p^3\}$ operation.
- The computational complexity can be reduced to $\mathcal{O}\{p^2\}$ (using the Levinson-Durbin recursion) if we assume that $\phi(i,k)=\phi(0,i-k)=R[i-k]$; $R[i-k]$ is called the autocorrelation, and this method is called the **autocorrelation method**. This is the same thing as assuming that the averaging window is very long:

$$\phi(i,k)=\sum_{n=p+1}^{N+p} s[n-i]s[n-k] \stackrel{?}{=} \sum_{n=p+1}^{N+p} s[n]s[n-(i-k)] = \phi(0,i-k)$$

Autocorrelation versus Covariance Methods

- The covariance method is more accurate: it finds exactly the predictor coefficients that are optimal for the window $p+1 \leq n \leq N+p$. The autocorrelation method is a little less accurate, especially for small analysis windows.
- With the normal covariance method, $A(z)$ often has roots outside the unit circle, especially for small analysis windows. This causes unstable speech synthesis, which makes your output go to $\hat{s}[n] =$**FLT_MAX**.
- The Atal article describes a modified covariance method that has the extra accuracy of regular covariance method, but that also guarantees a stable synthesis filter.
- Recommendation: don't use $\vec{a} = \Phi^{-1}\vec{c}$. If you're going to use the covariance method, use the modified method described by Atal.

High-Frequency Correction

The Atal article also talks about a correction for the high-frequency roll-off of many A-to-D converters. Looking up that reference, we find that the HF correction is just

$$\Phi = \Phi + \lambda \epsilon_p D,$$

where λ is a regularization constant ($\lambda \approx 0.1$), ϵ_p is the error residual obtained from LPC analysis without the correction, and D is a matrix with $3/8$ the main diagonal, $-1/4$ on each first off-diagonal, and $1/16$ on each second off-diagonal.

Outline

Main Equations

- Inverse filter:

$$H(z) = \frac{C_1}{1 - p_1 z^{-1}} + \frac{C_1^*}{1 - p_1^* z^{-1}}$$

$$h[n] = C_1 p_1^n u[n] + C_1^* (p_1^*)^n u[n]$$

- Orthogonality principle: a_k minimizes

$$\sum_{n=-\infty}^{\infty} e^2[n] = \sum_{n-\infty}^{\infty} \left(s[n] - \sum_{m=1}^{p} a_m s[n-m] \right)^2$$

if and only if $e[n] \perp s[n-k]$, meaning

$$\sum_{n=-\infty}^{\infty} e[n] s[n-k] = 0$$

- p linear equations in p unknowns:

$$\vec{a} = \Phi^{-1} \vec{c}$$