Linear Predictive Coding is All-Pole Resonance Modeling

Hyung-Suk Kim

Center for Computer Research in Music and Acoustics, Stanford University

1 Why another article on LPC?

Linear predictive coding (LPC) is a widely used technique in audio signal processing, especially in speech signal processing. It has found particular use in voice signal compression, allowing for very high compression rates. As widely adopted as it is, LPC is covered in many textbooks and is taught in most advanced audio signal processing courses. So why another article on LPC?

Despite covering LPC during my undergraduate coursework in electrical engineering, it wasn't until implementing LPC for my own research project that I understood what the goals of LPC were and what it was doing mathematically to meet those goals. A great part of the confusion stemmed from the somewhat cryptic name, Linear-Predictive-Coding. "Linear" made sense, however "Predictive" and "Coding", sounded baffling, at least to the undergraduate me. What is it trying to predict? And coding? What does that mean?

As we will find in the following sections, the name LPC does make sense. However, it is one catered to a specific use, speech signal transmission, possibly obscuring other applications, such as cross-synthesis. It takes a moment to understand the exclamation, "I used linear predictive coding to cross-synthesize my voice with a creaking ship!".

Thus the purpose of this article is to explain LPC from a general perspective, one that makes sense to me and hopefully to others trying to grasp what LPC is. Another purpose is to present a complete derivation in simple linear algebra terms before mentioning other concepts and approaches.

Finally, I prefer calling LPC, All-Pole Resonance Modeling.

2 The LPC Model

The original objective of LPC was to model human voice production. LPC is a source-filter model in that there is a sound source that goes through a filter. (Figure 1) The source, $e(n)$, models the vocal chords, while the resonant filter,

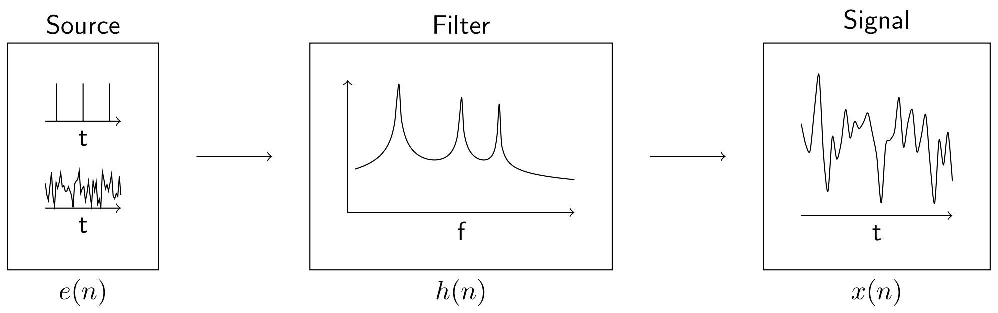

Figure 1: The LPC Model

$h(n)$ models the vocal tract. The resulting signal is,

$$x(n) = h(n) * e(n). \tag{1}$$

There are two possible signals for the source: an impulse train or random white noise. These signals model pitched sounds and plosive/fricatives respectively. The common characteristic for both impulse train and white noise is that they are spectrally flat; all spectral information is modeled in the filter. The keen reader will notice the source signal is labeled $e(n)$. This was chosen so for reasons to be revealed in the following sections.

LPC assumes the filter is a p-th order all-pole filter. Though not physiologically exact, it provides an extendable method for modeling resonances. This also allows for a tractable solution when estimating $h(n)$ from $x(n)$, which we will cover in §4.

Though initially developed for speech signals, the assumption of a spectrally flat source signal and a resonant filter applies well to modeling signals from most tonal instruments as well as many naturally occuring sounds.

3 The LPC Problem

For a filter design problem, like that of designing sound with an analog synthesizer, one would control filters, $h(n)$, with a source signal, $e(n)$, to create an interesting result, $x(n)$. However, for the problem at hand, the only known is $x(n)$, the resulting signal of the system. From $x(n)$, we need to estimate $h(n)$ and $e(n)$.

Let's take a moment to think about a system with p poles only. For a discrete time system, each pole corresponds to a delay, thus the system has memory and the current sample $x(n)$ will be the result of the current input $e(n)$ and past samples $x(n-k)$, $k = 1 \ldots p$. Inversely, this means that $h(n)$ imposes a relation between the past samples and the current sample. If we have enough examples of this imposed relation, we should be able to extract some information about the filter. Let's push this idea further.

We can formulate the relation between the input and output as,

$$\begin{aligned} X(z) &= H(z)E(z) \\ &= \frac{1}{1-\sum_{k=1}^{p} a_k z^{-k}} E(z). \end{aligned} \tag{2}$$

With a simple reordering, $X(z)$ can be expressed in terms of the input $E(z)$ and past samples $z^{-k}X(z)$.

$$X(z) = \sum_{k=1}^{p} a_k z^{-k} X(z) + E(z) \tag{3a}$$

$$\overset{\mathcal{Z}}{\Leftrightarrow} \quad x(n) = \sum_{k=1}^{p} a_k x(n-k) + e(n) \tag{3b}$$

The problem has now become that of finding the coefficients a_k, $k = 1 \dots p$. Again for reasons to be revealed, let's leave $e(n)$ as it is for now.

On a sidenote, this all-pole approach is also known as *auto-regression* (AR), since it is finding the future value of itself from the past.

4 Finding the Resonances

If we have $N \gg p$ samples, which is easily the case, then we have N equations for a_k, thus a_k is overdetermined.[1] Let's transform equation 3b to matrix form.

For $\mathbf{x}_i = \begin{bmatrix} x(n-1+i) \cdots x(n-p+i) \end{bmatrix} \in \mathbb{R}_{1\times p}$ and

$$\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix} \in \mathbb{R}_{p\times 1},$$

we get the following N equations.

$$\begin{aligned} x(n) &= \mathbf{x}_0 \cdot \mathbf{a} + e(n) \\ x(n+1) &= \mathbf{x}_1 \cdot \mathbf{a} + e(n+1) \\ &\vdots \\ x(n+N) &= \mathbf{x}_N \cdot \mathbf{a} + e(n+N) \end{aligned} \tag{4}$$

Forming the variables by stacking the variables,

[1]The order of the all-pole filter, p, sets a lower bound for the number of samples LPC needs to effectively model the signal.

$$\mathbf{b} = \begin{bmatrix} x(n) \\ \vdots \\ x(n+N) \end{bmatrix} \in \mathbb{R}_{N\times 1},\ \mathbf{e} = \begin{bmatrix} e(n) \\ \vdots \\ e(n+N) \end{bmatrix} \in \mathbb{R}_{N\times 1} \text{ and } \mathbf{A} = \begin{bmatrix} \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} \in \mathbb{R}_{N\times p},$$

then replacing the variables in equation 4 and rearranging the matrix multiplication term, we get

$$\begin{aligned} \mathbf{e} &= \mathbf{b} - \mathbf{A}\cdot\mathbf{a} && \text{(5a)} \\ &= \mathbf{b} - \hat{\mathbf{b}} && \text{(5b)} \end{aligned}$$

where $\hat{\mathbf{b}} \equiv \mathbf{A}\cdot\mathbf{a}$.

The equation above is known as *linear regression*, where $\hat{\mathbf{b}}$ is an estimate of $\mathbf{b}$ based on $\mathbf{A}$ with weights $\mathbf{a}$. The error of estimation or residual is $\mathbf{e}$.

As mentioned before, the equation is overdetermined and there is no exact solution, we can only try to find a "good fit". A well studied solution or "fit" to this problem is *least squares*, the solution of choice for LPC. For the least squares solution, we choose the values a_k that minimize $\|e(n)\|^2$, the power of the residual $\mathbf{e}$. The solution is simple to compute:

$$\begin{aligned} \mathbf{a} &= \mathbf{A}^\dagger\mathbf{b} \\ &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} && \text{(6)} \end{aligned}$$

where $\mathbf{A}^\dagger$ is the Moore-Penrose pseudoinverse.

5 Residuals

Now that we have found values for $\mathbf{a}$, by equation 6, we can calculate the residuals $\mathbf{e}$. If we have done a good job of modeling the resonances, we should be left with either an impulse train of a certain frequency(pitched), a random noise signal(unpitched) or a mixture of the two. If not, some of the resonance will have leaked into $\mathbf{e}$, and it will not be spectrally flat.

We first need to determine whether the signal is pitched or unpitched which requires pitch detection. Pitch detection is a widely studied subject. It is not a trivial task and is beyond the scope of this article. Simple methods include, zero-crossing detection and auto-correlation methods. More sophisticated methods use cepstrums, adaptive filtering or perceptual models. For the pitched case, we need the power of the source signal along with the fundamental frequency. For a random signal, all we need to measure is the variance of the samples σ^2, which is also the power of the source signal, $e(n)$.

It is known that 3 peak frequencies of the vocal tract, called formants, are enough to descriminate most vowels. Thus setting $p = 6$ will work well for speech signal. This is one reason LPC works well with speech. Choosing the optimal value p for other classes of signals is an open problem.

6 Analysis/Resynthesis

We have found the coefficients a_k and have a model for the signal. Great! However, the solution from §4 will only produce a single resonant sound, like "ah", not the whole sentence "I'm cool like that.". Real world signals, like music and speech, are time varying. So what do we do? Divide and conquer! Specifically, we need to cut the signal into small chunks and model each chunk using the equations from §4. This also adds a few conditions to the equations covered in §4, but the overall process holds.

The usual approach in audio processing is to use a technique called overlap-add (OLA). For OLA, we window the signal with a window function $w(n)$ that has a constant OLA property.

$$\sum_{m=-\infty}^{\infty} w(n - mR) = 1 \tag{7}$$

Equation 7 is the constant OLA equation, where R is called the step size. Thus instead of running LPC on the full signal $x(n)$, we use a windowed version $x_m(n) = w(n)x(n+mR)$. Because of the constant overlap-add property, simply adding the windowed signals will return the original signal.

$$\sum_{m=-\infty}^{\infty} w(n - mR)x(n) = x(n) \tag{8}$$

A window function of length N is zero outside the window. Thus the windowing imposes the condition that any value outside length N is zero. This means for $\mathbf{x}_i$ in equation 4, $x(n-m+i) = 0$, if $n-m+i < 0$ or $n-m+i \leq N$, where $m = 1 \ldots p$.

We mention this because the terms in equation 5b now become an auto-correlation which is often how LPC is explained. It is also known as the Yule-Walker equations. This approach allows for faster implementations using FFT to calculate the auto-correlations.

Now let's look at an example using LPC on speech signal. Let's choose a window of length $N = 240$, which is the length of a 30ms window at 8kHz, reasonable values for speech signals. Let's also use 50% OLA, which means $R = 0.5N = 120$. Since it is speech and we are only interested in 3 peaks (the formants), so we choose $p = 6$. For each step, 120 samples, after LPC, we are left with 7 numbers; the 6 coefficients a_k and the variance of the residual σ^2. Thus we have reduced the amount of data from 120 samples to 7 (about 17 to 1). That's quite impressive. In a sense, we have encoded the original audio into a compact form.

To decode the signal, we run the coefficients through the LPC model in Figure 1. Specifically, using the variance σ^2 to control the source, and the coefficients a_k for the filter, we get an estimated signal $\hat{x}_m(n) = w(n)\left(e(n) * h_m(n)\right)$. We then overlap-add the decoded windowed signals to obtain the full signal $\hat{x}(n)$. This can be viewed as playing an LPC synth with LPC encoded parameters. We will explore this idea in §8.

7 Decyphering the name: LPC

Now that we have covered how LPC works, let's review the name.

Linear Prediction. The system in Figure 1 is a linear system. We use least squares which solves linear equations. Actually, the system is using *linear prediction* where in equations 3b and 5b, we are using the past values of $x(n)$ *linearly* to find the coefficients a_k that best estimate or *predict* the current value.

Coding. In §6, we covered how LPC can be used to encode(analyze) and decode(resynthesize) speech signals. So LPC can be viewed as a *coding* algorithm. Another example of a well known audio coding algorithm is the MP3 codec(coder/decoder). The two audio coding schemes make an interesting comparison. Where LPC tries to model how the sound is created (source modeling), MP3 models how the sound is perceived (listener modeling).

8 Cross-synthesis

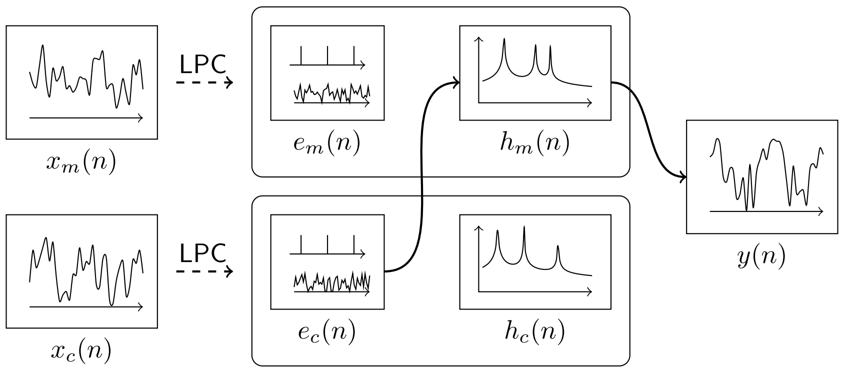

Figure 2: Cross-synthesis

Viewing LPC as a coding method can obscure other possible uses. Looking at LPC from an analysis/synthesis perspective allows for different applications. A good example is that of cross-synthesis. (Figure 2) After running a signal through LPC, we are left with a model of the filter, $h(n)$ and a model of the source, $e(n)$. The filter controls the resonance or tone of the signal, while the source controls the utterance. If we run two different signals through LPC, it would be possible to use the source of one model (carrier, $e_c(n)$) and the filter of the other model (modulator, $h_m(n)$) to create a different, hopefully interesting, output signal $y(n) = h_m(n) * e_c(n)$, thus cross-synthesis.

This creates particularly interesting results when taking the filter from speech signals and the source from a spectrally rich sound texture such as wind, or creaking wood.

9 Next Steps

As useful as LPC has proven to be, it is not without limitations. For one, the audio signal should fit the source-filter model for LPC to work well. A signal with many sounds mixed may not work so well.

Further improvements in sound quality can be acheived without significantly increasing the data needed by using *excitation codes*. This approach is called Code-Excited Linear Prediction (CELP).

The keen reader may have noticed that within a windowed segment, there is no change in power. The assumption is that the signal within a windowed segment is stationary. A transient or attack will become smeared by a windows width. Thus, LPC itself is not suited for sounds with many transients such as fire crackling. A recent approach to address this problem is that of time-frequency LPC (TFLPC) or cascaded time-frequency linear prediction (CTFLP), where the time envelope of the source signal after LPC is modeled in the time-domain also using an all-pole model.

Despite the limitations, LPC is a good starting point for analyzing and modeling sounds and worth understanding.