UNUSUAL TEACHING SHORT-CUTS TO THE LEVINSON AND LATTICE ALGORITHMS

Thierry Dutoit

Faculté Polytechnique de Mons, 9, rue de Houdain, B-7000 Mons
email : thierry.dutoit@fpms.ac.be

ABSTRACT

The Levinson and Lattice algorithms are taught in many signal processing curricula. If only one, the fact that every cell phone solves Yule-Walker equations every 10 ms justifies it all. These algorithms, however, tend to be hard to conceptualize in a few mental images. This paper proposes two such short-cut views, geometric in the wide sense, which proved to help students "see" the essence of these tools.

1. INTRODUCTION

The Levinson and Lattice algorithms are taught in many signal processing curricula. If only one, the fact that every cell phone solves Yule-Walker equations every 10 ms justifies it all.

These algorithms are usually described in algebraic terms, using equations which, although they are easy to follow from one to the other, do not lead to simple mental images : students agree with the equations, but do not really "see" the essential principles of the algorithms.

These equations, however, have given birth to dozens of profitable interpretations (see [1] for a review), from maximum likelihood to polynomial approaches, through the extensively used spectral interpretation. All of them share the same mathematical reality observed in various lights. We present here two somewhat original interpretations of the Levinson and Lattice algorithms, using matrix-based and geometric-oriented approaches respectively. As far as we know, they are usually not presented as such in textbooks, although the geometric-oriented approach to linear prediction is sometimes used (see [2], [3], [4] for instance).

Before examining them, we recall here the algebraic and geometric principles behind the Yule-Walker equations.

Let us recall that Yule-Walker equations of order p appear in linear prediction analysis, when trying to obtain a set of prediction coefficients $\{a_1,a_2,...a_p\}$ verifying, in the least squares sense:

$$s(n) = \sum_{i=1}^{p} - a_i s(n-i) + f(n) \qquad \text{for } n = p, p+1, \ldots, N-1$$

with $a_1,\ldots,a_p$ chosen so as to minimize $\sum_{n=p}^{N-1} f(n)^2$

where *s(0), s(1),...s(N-1)* are N samples of the analyzed signal (with N greater than p) and *f(n)* is the prediction residual (or error). This can be written, in vector notation:

$$\mathbf{s_p} = \sum_{i=1}^{p} - a_i \mathbf{s_{p-i}} + \mathbf{f_p}$$

as the search for the linear combination of the prediction vectors $\mathbf{s_0}$, ..., $\mathbf{s_{p-1}}$ that is closest to $\mathbf{s_p}$, with $\mathbf{s_i} = [s(i), s(i+1),..., s(N-p+i-1)]^T$ (Fig. 1). As $\mathbf{s_p}$ is not, in general, included in the subspace $\{\mathbf{s_0},..., \mathbf{s_{p-1}}\}$, this decomposition is performed under the constraint of minimizing the norm $\alpha_\mathbf{p}$ of $\mathbf{f_p} = [f(p), f(p+1),..., f(N-1)]^T$, the (forward) prediction error vector.

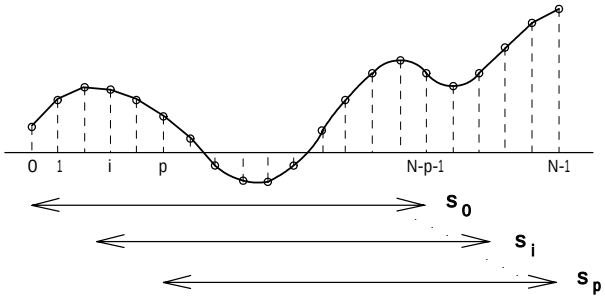

Fig. 1. Prediction vectors.

Equation (1) leads to a well-known matrix form:

$$\mathbf{s_p} = \mathbf{\Phi a} + \mathbf{f_p}$$

in which $\mathbf{\Phi}$ is the matrix of column vectors $\mathbf{s_0}$,..., $\mathbf{s_{p-1}}$ and $\mathbf{a}$ is the vector $[-a_1, -a_2, ..., -a_p]^T$.

The linear combination that minimizes α_p is naturally obtained by projecting $\mathbf{s_p}$ orthogonally to the prediction subspace $\{\mathbf{s_0},..., \mathbf{s_{p-1}}\}$ (Fig 2), which simply express that :

$$\mathbf{f_p} \perp \mathbf{s_j} \qquad \text{for } j = 0,1,\ldots,p-1$$

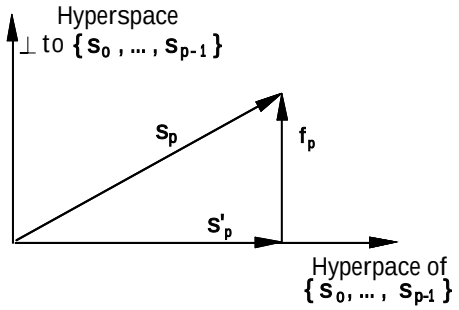

Fig. 2. The (forward) prediction error vector.

$$\mathbf{s_p}\,\mathbf{s_j} = \mathbf{s'_p}\,\mathbf{s_j}$$

$$= \sum_{i=1}^{p} -a_i \mathbf{s_{p-i}}\,\mathbf{s_j} \qquad \text{for } j = 0,1,\ldots,p-1$$

or, in matrix form,

$$\mathbf{\Phi^T \Phi\, a} = \mathbf{\Phi^T s_p}$$

$$\mathbf{a} = (\mathbf{\Phi^T \Phi})^{-1} \mathbf{\Phi^T s_p}$$

in which $(\mathbf{\Phi^T \Phi})^{-1}$ is the pseudo-inverse of Φ, i.e. the inverse of the covariance-like matrix $\mathbf{\Phi^T \Phi}$, well known in least squares theory [5].

2. SHORT-CUT TO THE LEVINSON ALGORITHM

Up to now, the very peculiar relationship among $\mathbf{s_i}$ vectors, which originates in the way they have been extracted from the analysis frame, has not been taken into account. It is known, however, to lead to important simplifications, provided the covariance-like matrix $\mathbf{\Phi^T \Phi}$ is further assumed to be Toeplitz symmetric.

Let us first define a *circular sequence of vectors* {$\mathbf{s_0}$, $\mathbf{s_1}$, ..., $\mathbf{s_p}$} as any sequence of vectors constructed as shown in Fig. 1, and the covariance-like matrix of which is Toeplitz symmetric. Clearly, such vectors have identical norms, and the scalar product $\mathbf{s_i}\,\mathbf{s_j}$ depends only on *i-j*. Any circular sequence is thus entirely defined by its last vector $\mathbf{s_p}$ and its order *p*, from which any other vector $\mathbf{s_i}$ is deduced by a circular right shift of order *p-i* of its coordinates. Obviously, the circular sequence introduced in Section 1 is generated by $\mathbf{s_p}$ = (*s(0),...,s(N-2p-1),0,..., 0*). Linear prediction analysis based on this assumption is termed is usually termed as operated in the *autocorrelation framework.*

It is then very easy to see why this assumption leads to inversion algorithms such as Levinson's whose complexity is of the order of $O(p^2)$ rather than the usual $O(p^3)$ complexity.

As a matter of fact, the Levinson algorithm is based on a recursive derivation of $\{a_1,a_2,\ldots a_p\}$ for order $m+1$ based on the result for order m Let.us write both problems as:

$$\begin{bmatrix} \phi_x(0) & \phi_x(1) & \ldots & \phi_x(m-1) & \phi_x(m) \\ \phi_x(1) & \phi_x(0) & \ldots & \phi_x(m-2) & \phi_x(m-1) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \phi_x(m-1) & \phi_x(m-2) & \ldots & \phi_x(0) & \phi_x(1) \\ \phi_x(m) & \phi_x(m-1) & \ldots & \phi_x(1) & \phi_x(0) \end{bmatrix} \begin{bmatrix} a_1^{m+1} \\ a_2^{m+1} \\ \ldots \\ a_m^{m+1} \\ a_{m+1}^{m+1} \end{bmatrix} = - \begin{bmatrix} \phi_x(1) \\ \phi_x(2) \\ \ldots \\ \phi_x(m) \\ \phi_x(m+1) \end{bmatrix}$$

and :

$$\begin{bmatrix} \phi_x(0) & \phi_x(1) & \ldots & \phi_x(m-1) \\ \phi_x(1) & \phi_x(0) & \ldots & \phi_x(m-2) \\ \ldots & \ldots & \ldots & \ldots \\ \phi_x(m-1) & \phi_x(m-2) & \ldots & \phi_x(0) \end{bmatrix} \begin{bmatrix} a_1^m \\ a_2^m \\ \ldots \\ a_m^m \end{bmatrix} = - \begin{bmatrix} \phi_x(1) \\ \phi_x(2) \\ \ldots \\ \phi_x(m) \end{bmatrix}$$

It is striking to notice that, thanks to its Toeplitz structure, the matrix of the first set of equations contains twice the (smallest) matrix of the second : once in the upper left corner, and once in the lower right one. Since there is a additional equation in the first set, however, the solutions for order *m* and *m+1* apparently do not share a simple relationship.

Let us augment the second set of equations with a new *last* equation, obtained by boldly assuming $a_{m+1}^{m+1}=0$:

$$\begin{bmatrix} \phi_x(0) & \phi_x(1) & \ldots & \phi_x(m-1) & \phi_x(m) \\ \phi_x(1) & \phi_x(0) & \ldots & \phi_x(m-2) & \phi_x(m-1) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \phi_x(m-1) & \phi_x(m-2) & \ldots & \phi_x(0) & \phi_x(1) \\ \phi_x(m) & \phi_x(m-1) & \ldots & \phi_x(1) & \phi_x(0) \end{bmatrix} \begin{bmatrix} a_1^m \\ a_2^m \\ \ldots \\ a_m^m \\ 0 \end{bmatrix} = - \begin{bmatrix} \phi_x(1) \\ \phi_x(2) \\ \ldots \\ \phi_x(m) \\ \phi_x(m+1)+\mu \end{bmatrix} \qquad (1)$$

Using the solutions for order *m* as the first-guess solutions for order *m+1* then automatically verifies the *m* first equations, but not the last one (except by chance). We thus denote by μ the difference between the desired value of the last element of the independent term and it desired value $\phi_x(m+1)$.

Similarly, one can add a new *first* equation to the set of *m*th order equations, and boldly assume $a_1^{m+1}=1$:

$$\begin{bmatrix} \phi_x(0) & \phi_x(1) & \ldots & \phi_x(m-1) & \phi_x(m) \\ \phi_x(1) & \phi_x(0) & \ldots & \phi_x(m-2) & \phi_x(m-1) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \phi_x(m-1) & \phi_x(m-2) & \ldots & \phi_x(0) & \phi_x(1) \\ \phi_x(m) & \phi_x(m-1) & \ldots & \phi_x(1) & \phi_x(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1^m \\ \ldots \\ a_{m-1}^m \\ a_m^m \end{bmatrix} = - \begin{bmatrix} \alpha \\ 0 \\ \ldots \\ 0 \\ 0 \end{bmatrix}$$

Here, using the *i*th coefficient of the solution for order *m* as the *i+1*th coefficient of the first-guess solution for order *m+1* verifies the *m* last equations (in which the independent term has been set to 0 for complying with the *m*th order equations). However, the first equation leads to a non-zero independent term, which we denote α.

In order to obtain the exact solution for order *m+1*, we simply notice that , thanks to the symmetry of its matrix, inverting the order of all equations in our problem does not change the matrix itself:

$$\begin{bmatrix} \phi_x(0) & \phi_x(1) & \dots & \phi_x(m-1) & \phi_x(m) \\ \phi_x(1) & \phi_x(0) & \dots & \phi_x(m-2) & \phi_x(m-1) \\ \dots & \dots & \dots & \dots & \dots \\ \phi_x(m-1) & \phi_x(m-2) & \dots & \phi_x(0) & \phi_x(1) \\ \phi_x(m) & \phi_x(m-1) & \dots & \phi_x(1) & \phi_x(0) \end{bmatrix} \begin{bmatrix} a_m^m \\ a_{m-1}^m \\ \dots \\ a_1^m \\ 1 \end{bmatrix} = - \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \alpha \end{bmatrix} \quad (2)$$

Hence, it is now easy to compute a new set of equations, as the linear combination (1)+k_{m+1}(2) of sets (1) and (2) which eliminates μ in (1) while leaving the other independent term components untouched:

$$\phi_x(m+1) + \mu_m + k_{m+1}\alpha_m = \phi_x(m+1) \qquad k_{m+1} = -\mu_m / \alpha_m$$

As a result, the solution of the $m+1$ th order problem naturally comes out as a function of the solution of the mth order problem:

$$a_i^{m+1} = a_i^m + k_{m+1} a_{m+1-i}^m \qquad a_{m+1}^{m+1} = k_{m+1}$$

which immediately shows that the total computational complexity of the algorithm will be $O(p^2)$.

3. SHORT-CUT TO THE LATTICE ALGORITHM

In this Section, we show how the Lattice algorithm (and filter) can be btained by geometric examination of the linear prediction problem, after first recalling the vector-based interpretation of PARCOR coefficients.

3.1. Vector-based derivation of PARCOR coefficients

Just like forward predictions of order m, which result from the projection of $\mathbf{s_p}$ on the vector subspace of the m *previous* vectors $\{\mathbf{s_{p-1}}, .., \mathbf{s_{p-m}}\}$, it is possible to define *backward predictions* of order m, as the linear decomposition of $\mathbf{s_{p-m-1}}$ on the vector subspace of the m *following* vectors $\{\mathbf{s_{p-m}}, .., \mathbf{s_{p-1}}\}$. The associated backward prediction error vectors $\mathbf{g_m}$, the norms of which are the square roots of the *backward prediction error energies* β_m, are thus the orthogonal components of $\mathbf{s_{p-m-1}}$ on $\{\mathbf{s_{p-m}}, .., \mathbf{s_{p-1}}\}$ (Fig. 3).

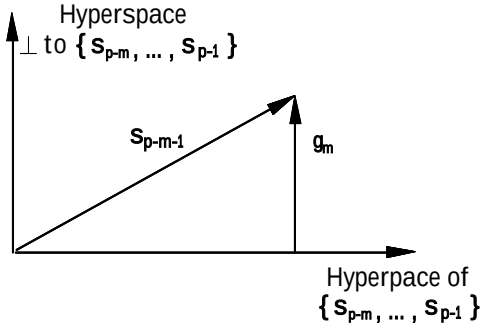

Fig. 3. The backward prediction error vector.

This time, $\{\mathbf{g_0}, \mathbf{g_1}, ..., \mathbf{g_m}\}$ are orthogonal to one another, by construction. What is more, they span the same subspace as $\{\mathbf{s_{p-m}}, .., \mathbf{s_{p-1}}\}$, so that $\{\mathbf{g_0}, \mathbf{g_1}, ..., \mathbf{g_m}\}$ is an orthogonal basis of $\{\mathbf{s_{p-m}}, .., \mathbf{s_{p-1}}\}$. The corresponding matrix, $\mathbf{G}$, is orthogonal.

This feature results in a considerable simplification in their computation: $\mathbf{g_m}$ is also the orthogonal component of $\mathbf{s_{p-m-1}}$ on $\{\mathbf{g_0}, \mathbf{g_1}, ..., \mathbf{g_{m-1}}\}$, so that successive backward prediction error vectors can be computed recursively, according to the well-known Grahm-Schmidt orthogonalization process:

$$\mathbf{g_0} = \mathbf{s_{p-1}}$$

$$\mathbf{g_m} = \mathbf{s_{p-m-1}} - \sum_{i=0}^{m-1} g_i \, \mathbf{g_i} \qquad \text{for } m = 1 \dots p-1$$

$$\text{with} \quad g_i = \frac{\mathbf{s_{p-m-1}}\ \mathbf{g_i}}{\|\mathbf{g_i}\|^2}$$

This can be turned into a simultaneous recursive computation of the forward prediction errors themselves. Assuming $\mathbf{f_{m-1}}$ and $\mathbf{g_{m-1}}$ are known—that is, the respective orthogonal components of $\mathbf{s_p}$ and $\mathbf{s_{p-m}}$ on $\{\mathbf{s_{p-m-1}}, .., \mathbf{s_{p-1}}\}$—it is easy to see (Fig. 4.) that $\mathbf{f_m}$ can be expressed as:

$$\mathbf{f_m} = \mathbf{f_{m-1}} + k_m \, \mathbf{g_{m-1}} \quad (3)$$

in which k_m is the opposite of the projection of $\mathbf{f_{m-1}}$ on $\mathbf{g_{m-1}}$:

$$k_m = -\frac{\mathbf{f_{m-1}}\ \mathbf{g_{m-1}}}{\|\mathbf{g_{m-1}}\|^2} \quad (4)$$

$\{k_0, k_1, .., k_p\}$ are the well-known PARCOR coefficients. They actually have a wider geometrical interpretation than suggested by (4). Since $\mathbf{f_{m-1}} = \mathbf{f_{m-2}} + k_{m-1}\,\mathbf{g_{m-2}}$, and $\mathbf{g_{m-2}} \perp \mathbf{g_{m-1}}$, equation (4) can be rewritten as:

$$k_m = -\frac{\mathbf{f_{m-2}}\ \mathbf{g_{m-1}}}{\|\mathbf{g_{m-1}}\|^2}$$

and so on, down to

$$k_m = -\frac{\mathbf{f_0}\ \mathbf{g_{m-1}}}{\|\mathbf{g_{m-1}}\|^2} = -\frac{\mathbf{s_p}\ \mathbf{g_{m-1}}}{\|\mathbf{g_{m-1}}\|^2}$$

In other words, PARCOR coefficients are the negative of the projections of $\mathbf{s_p}$ on the backward prediction error vectors. As a result, $\{k_1, k_2, ..., k_p\}$ carry the same information as $\{a_1, a_2, ..., a_p\}$; they can be interchanged by a simple change of co-ordinates in the $\{\mathbf{s_{p-m}}, .., \mathbf{s_{p-1}}\}$ subspace.

$$\mathbf{s'_p} = \sum_{i=1}^{p} - a_i \mathbf{s_{p-i}} = \sum_{i=1}^{p} - k_i \mathbf{g_i}$$

$\Phi \mathbf{a} = \mathbf{G}\,\mathbf{k}$, where $\mathbf{k}$ is the vector $[-k_1, -k_2, ..., -k_p]$

This is illustrated in Fig. 4.

Such a k_i to a_i transformation is hardly useful in practice, since a lattice version of the synthesis filter

$1/A_p(z)$ can be derived in the particular case of the autocorrelation method; its coefficients are precisely the k_i.

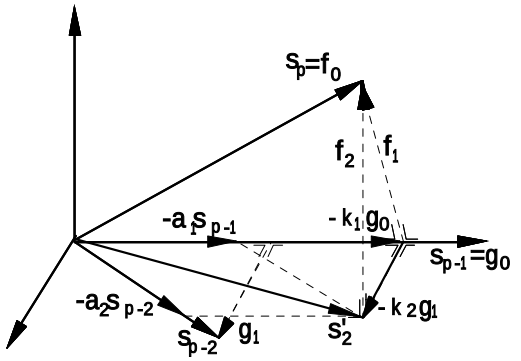

Fig. 4. Prediction and PARCOR coefficients are different expressions of the same decomposition process.

3.2. Vector-based derivation of the lattice algorithm

Because PARCOR coefficients are computed on the basis of the angles between vectors in a circular sequence of vector and of their norm, they are insensitive to orthogonal transformations applied on the initial circular sequence. They are therefore related to *circular figures* (in an *N*-dimension space) rather than to circular sequences themselves. As such, subsequences of vectors in a circular sequence (which are themselves circular sequences) have identical PARCOR coefficients, provided their order is the same. Furthermore, reversing a circular sequence has no effect on the related PARCORs.

These properties are turned into account in the Levinson algorithm to speed up the computation of the backward prediction error vectors. As a matter of fact, it suffices to notice that

1. Computing $\mathbf{g_m}$—the orthogonal component of $\mathbf{s_{p-m-1}}$ on $\{\mathbf{s_{p-m-2}}, .., \mathbf{s_{p-1}}\}$—is equivalent to computing the orthogonal component of $\mathbf{s_{p-m}}$ on $\{\mathbf{s_{p-m-1}}, .., \mathbf{s_p}\}$, which we shall denote as $\mathbf{g'_m}$, and circularly shifting the coordinates of the result one element to the right. This one element coordinate shift is an orthogonal transformation described by a *circulant* matrix **S**.

2. Computing $\mathbf{g'_m}$ can itself be achieved with a similar strategy as in equation (3), that is by using the orthogonal component of $\mathbf{s_{p-m}}$ on $\{\mathbf{s_{p-m-1}}, .., \mathbf{s_{p-1}}\}$, i.e. $\mathbf{g_{m-1}}$, and adding it a correction vector parallel to the orthogonal component of $\mathbf{s_p}$ on $\{\mathbf{s_{p-m-1}}, .., \mathbf{s_{p-1}}\}$, i.e. to $\mathbf{f_{m-1}}$. In other words, $\mathbf{g'_m}$ can be considered as the *m*th *forward* prediction error vector of the circular sequence $\{\mathbf{s_p}, \mathbf{s_{p-1}}, .., \mathbf{s_{p-m}}\}$. This sequence has the same PARCORs as its reversed sequence, $\{\mathbf{s_{p-m}}, \mathbf{s_{p-m+1}}, .., \mathbf{s_p}\}$—the k_i computed with (4).

Remarks 1 and 2 lead to the very important relation:

$$\mathbf{g_m} = \mathbf{S}\,\mathbf{g'_m} = \mathbf{S}(\mathbf{g_{m-1}} + k_m\,\mathbf{f_{m-1}})$$

which, when used in combination with (3) and (4), makes it possible to solve the linear prediction problem with a single recursion over the prediction order. This method is known as the *lattice algorithm*, since it is completely defined by the lattice inverse filter of Fig. 4, which should be understood as a sequence of blockwise operations that progressively produce forward and backward prediction error vectors from a given analysis frame. Delay blocks in this filter are the expression of the one element coordinate shift suggested in the previous paragraphs.

4. REFERENCES

[1] MARKEL, J.D., and A.H. GRAY Jr, (1976), *Linear Prediction of Speech*, Springer Verlag, New York, pp. 10-42.

[2] ALEXANDER, S.T., (1986), Adaptive Signal Processing: Theory and Applications, Springer-Verlag, New-York, pp.123-141.

[3] KROON, P., (1985), *Time-Domain Coding of (Near) Toll Quality Speech at Rates below 16 kB/s*, Ph.D. dissertation, Techniche Hogeschool, Delft.

[4] CADZOW, J. A., (1990), "Signal Processing via Least Squares Error Modeling", *IEEE ASSP Magazine*, October, pp 12-31.

[5] GOLUB, G.H., and C.F. VAN LOAN, (1989), *Matrix Computations*, Johns Hopkins University Press, London, p. 243.

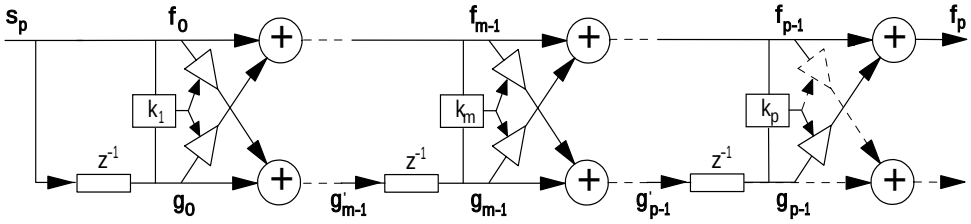

Fig. 4. The well-known lattice filter (and implicitly, the corresponding algorithm).