A Fast Implementation of Burg's Method

Koen Vos

August, 2013

Abstract

In this paper we derive an efficient implementation of the Burg method for maximum entropy spectral analysis. The new algorithm has a complexity of about $Nm+5m^2$ multiplications, compared to $3Nm-m^2$ for the conventional implementation, for data length N and order m. The algorithm also allows for straightforward regularization.

Introduction

Burg's method for maximum entropy spectral analysis [1, 2] is a popular tool in spectral estimation, speech processing, radar, geophysics and other fields. It combines a high prediction gain with poles that guarantee stable filters.

While the Burg method is typically described as directly operating on time series data, it can also be formulated in terms of autocorrelations, as this paper will show. In that case, however, the correlations change between computing reflection coefficients, and need to be adjusted to account for what Burg called "end effects" [2]. As a consequence it is not possible to express the Burg method in a single set of equations akin to the Yule-Walker equations. Perhaps this has hindered application of techniques for efficient inversion of close-to-Toeplitz matrices, where such techniques have previously been used for the covariance method [3] and forward-backward prediction [4].

This paper derives an implementation of Burg's method in terms of autocorrelations, and shows how properties of the correlation matrix enable a fast implementation. We first describe the conventional Burg method before going into the details of the new method.

Burg Method

Conventionally, Burg's method operates on the time series data. Each iteration finds a scalar reflection coefficient k_i that minimizes forward and backward prediction error vectors $\mathbf{f}_i$ and $\mathbf{b}_i$[1]. Burg's method is shown below in its basic form.

[1]Bold lower-case letters indicate column vectors, capitals represent matrices, and $(\cdot)^*$ denotes complex conjugation.

Algorithm 1: Conventional Burg Method

For an input signal $x_0, x_1, \ldots, x_{N-1}$, compute the m reflection coefficients as follows:

0. *Initialization*

$$i = 0 \tag{1}$$

$$\mathbf{f}'_0 = \mathbf{b}'_0 = [x_0, x_1, \ldots, x_{N-1}]^T. \tag{2}$$

1. Remove the first element of $\mathbf{f}'_i$ and last element of $\mathbf{b}'_i$

$$\mathbf{f}_i = \mathbf{f}'_i(1 : N-i-1) \tag{3}$$

$$\mathbf{b}_i = \mathbf{b}'_i(0 : N-i-2). \tag{4}$$

2. Calculate the reflection coefficient

$$k_i = -\frac{2\mathbf{b}_i^H \mathbf{f}_i}{\mathbf{f}_i^H \mathbf{f}_i + \mathbf{b}_i^H \mathbf{b}_i}. \tag{5}$$

3. **if** $(i == m)$

   Finished

   endif

4. Update the prediction errors

$$\mathbf{f}'_{i+1} = \mathbf{f}_i + k_i \mathbf{b}_i \tag{6}$$

$$\mathbf{b}'_{i+1} = \mathbf{b}_i + k_i^* \mathbf{f}_i. \tag{7}$$

5. Increment iteration counter: $i = i + 1$

6. Back to step (1).

The choice of the reflection coefficient in (5) minimizes the resulting combined forward/backward prediction error energy $\mathbf{f}'^H_{i+1}\mathbf{f}'_{i+1} + \mathbf{b}'^H_{i+1}\mathbf{b}'_{i+1}$. This lets us use the orthogonality principle to find an efficient recursion [5] for the denominator in Eq. (5)

$$\begin{aligned} den_i &\equiv \mathbf{f}_i^H \mathbf{f}_i + \mathbf{b}_i^H \mathbf{b}_i && (8)\\ &= \mathbf{f}_i'^H \mathbf{f}'_i + \mathbf{b}_i'^H \mathbf{b}'_i \\ &\quad - |\mathbf{f}'_i(0)|^2 - |\mathbf{b}'_i(N-i-1)|^2 && (9)\\ &= (1 - |k_{i-1}|^2) den_{i-1} \\ &\quad - |\mathbf{f}'_i(0)|^2 - |\mathbf{b}'_i(N-i-1)|^2. && (10) \end{aligned}$$

Although not used in the Burg method itself, prediction error coefficients $\mathbf{a}_i$ can be found from the reflection coefficients using the Levinson recursion [6]

$$\mathbf{a}_{i+1} = \begin{bmatrix}\mathbf{a}_i \\ 0\end{bmatrix} + k_i J \begin{bmatrix}\mathbf{a}_i^* \\ 0\end{bmatrix}, \tag{11}$$

initialized as $\mathbf{a}_0 = 1$, and where we introduced the exchange matrix

$$J = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \iddots & 1 & 0 \\ 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}, \tag{12}$$

which flips the vector or matrix to its right upside down (or, that to its left, horizontally). The size of J is $(i+1)\times(i+1)$ and presumed clear from the context.

Fast Implementation

We now derive an implementation of Burg's method that operates on the forward-backward correlation matrix, without explicitly calculating the forward/backward prediction error vectors. Since the method is order recursive, the derivation expresses variables in terms of the previous iteration's outcome. We begin by stacking the time series data in an $(N-i)\times(i+1)$ Toeplitz matrix

$$X_i = \begin{bmatrix} x_i & x_{i-1} & \cdots & x_0 \\ x_{i+1} & x_i & \cdots & x_1 \\ \vdots & \vdots & \ddots & \vdots \\ x_{N-1} & x_{N-2} & \cdots & x_{N-i-1} \end{bmatrix}. \tag{13}$$

Note that X_i grows one wider and one shorter with every iteration. Given prediction error coefficients $\mathbf{a}_i$, we obtain the Burg forward and conjugate backward prediction errors $\mathbf{f}_i$ and $\mathbf{b}_i^*$ as

$$\begin{bmatrix}\mathbf{f}_{i+1} \\ \mathbf{b}_{i+1}^*\end{bmatrix} = \begin{bmatrix} X_{i+1} \\ X_{i+1}^* J \end{bmatrix} \mathbf{a}_{i+1} \tag{14}$$

$$= \begin{bmatrix} X_{i+1} \\ X_{i+1}^* J \end{bmatrix} \begin{bmatrix}\mathbf{a}_i \\ 0\end{bmatrix} + k_i \begin{bmatrix} X_{i+1} J \\ X_{i+1}^* \end{bmatrix} \begin{bmatrix}\mathbf{a}_i^* \\ 0\end{bmatrix}, \tag{15}$$

where the second equality comes from the Levinson recursion (11) and the fact that $J^2 = I$. Minimizing the energy of the left-hand vector with respect to k_i gives the solution

$$k_i = -\frac{\begin{bmatrix}\mathbf{a}_i^H & 0\end{bmatrix}\left(X_{i+1}^H X_{i+1} J + J X_{i+1}^T X_{i+1}^*\right)\begin{bmatrix}\mathbf{a}_i^* \\ 0\end{bmatrix}}{\begin{bmatrix}\mathbf{a}_i^T & 0\end{bmatrix}\left(J X_{i+1}^H X_{i+1} J + X_{i+1}^T X_{i+1}^*\right)\begin{bmatrix}\mathbf{a}_i^* \\ 0\end{bmatrix}} \tag{16}$$

$$= -\frac{\begin{bmatrix}\mathbf{a}_i^H & 0\end{bmatrix} J R_{i+1} \begin{bmatrix}\mathbf{a}_i^* \\ 0\end{bmatrix}}{\begin{bmatrix}\mathbf{a}_i^T & 0\end{bmatrix} R_{i+1} \begin{bmatrix}\mathbf{a}_i^* \\ 0\end{bmatrix}}, \tag{17}$$

where we defined the forward-backward correlation matrix

$$R_i \equiv J X_i^H X_i J + X_i^T X_i^*. \tag{18}$$

This correlation matrix is persymmetric and Hermitian, *i.e.*, $J R_i J = R_i^T = R_i^*$. These properties, together with the observation that the upper-left $i \times i$ submatrix of R_i can be expressed as R_{i-1} plus a rank-2 update ΔR_i, enable the following recursion

$$\mathbf{g}_i \equiv R_{i+1} \begin{bmatrix}\mathbf{a}_i^* \\ 0\end{bmatrix} \tag{19}$$

$$= \begin{bmatrix} R_i + \Delta R_{i+1} \\ \mathbf{r}_{i+1}^T \end{bmatrix} \mathbf{a}_i^* \tag{20}$$

$$= \begin{bmatrix} R_i \left(\begin{bmatrix}\mathbf{a}_{i-1}^* \\ 0\end{bmatrix} + k_{i-1}^* J \begin{bmatrix}\mathbf{a}_{i-1} \\ 0\end{bmatrix} \right) + \Delta R_{i+1} \mathbf{a}_i^* \\ \mathbf{r}_{i+1}^T \mathbf{a}_i^* \end{bmatrix} \tag{21}$$

$$= \begin{bmatrix} \mathbf{g}_{i-1} + k_{i-1}^* J \mathbf{g}_{i-1}^* + \Delta R_{i+1} \mathbf{a}_i^* \\ \mathbf{r}_{i+1}^T \mathbf{a}_i^* \end{bmatrix}, \tag{22}$$

where the rank-2 update matrix is given by

$$\Delta R_{i+1} = -\begin{bmatrix} x_i \\ \vdots \\ x_0 \end{bmatrix} \begin{bmatrix} x_i^* & \cdots & x_0^* \end{bmatrix} - \begin{bmatrix} x_{N-1}^* \\ \vdots \\ x_{N-i-1}^* \end{bmatrix} \begin{bmatrix} x_{N-i-1} & \cdots & x_{N-1} \end{bmatrix}. \tag{23}$$

Note that the product $\Delta R_{i+1} \mathbf{a}_i^*$ can be computed with $4(i+1)$ multiplications by first calculating the inner products of the row vectors in Eq. (23) with $\mathbf{a}_i^*$. The vector $\mathbf{r}_{i+1}^T$ contains the first $i+1$ elements of the last row of R_{i+1}, and satisfies the recursion

$$\mathbf{r}_{i+1} = \begin{bmatrix} 2c_{i+1} \\ \mathbf{r}_i - \begin{bmatrix} x_{N-1}^* \\ \vdots \\ x_{N-i}^* \end{bmatrix} x_{N-i-1} - \begin{bmatrix} x_0 \\ \vdots \\ x_{i-1} \end{bmatrix} x_i^* \end{bmatrix}, \tag{24}$$

which uses the autocorrelation

$$c_i = \begin{bmatrix} x_0 & \cdots & x_{N-i-1} \end{bmatrix} \begin{bmatrix} x_i^* \\ \vdots \\ x_{N-1}^* \end{bmatrix}. \tag{25}$$

The above recursions let us write the fast Burg algorithm as follows.

Algorithm 2: Fast Burg Method

For an input signal $x_0, x_1, \ldots, x_{N-1}$, compute the m reflection coefficients and prediction error coefficients $\mathbf{a}_m$ as follows:

0. *Initialization*

   For $j = 0 : m$

$$c_j = \begin{bmatrix} x_0 & \cdots & x_{N-j-1} \end{bmatrix} \begin{bmatrix} x_i^* \\ \vdots \\ x_{N-1}^* \end{bmatrix} \tag{26}$$

   End

$$i = 0 \tag{27}$$

$$\mathbf{a}_0 = 1 \tag{28}$$

$$\mathbf{g}_0 = \begin{bmatrix} 2c_0 - |x_0|^2 - |x_{N-1}|^2 \\ 2c_1 \end{bmatrix} \tag{29}$$

$$\mathbf{r}_1 = 2c_1 \tag{30}$$

1. Compute the reflection coefficient

$$k_i = -\frac{\begin{bmatrix} \mathbf{a}_i^H & 0 \end{bmatrix} J \mathbf{g}_i}{\begin{bmatrix} \mathbf{a}_i^T & 0 \end{bmatrix} \mathbf{g}_i} \tag{31}$$

2. Update the prediction coefficients

$$\mathbf{a}_{i+1} = \begin{bmatrix} \mathbf{a}_i \\ 0 \end{bmatrix} + k_i J \begin{bmatrix} \mathbf{a}_i^* \\ 0 \end{bmatrix} \tag{32}$$

3. Increment the iteration counter: $i = i + 1$

4. **if** $(i == m)$

   Finished

   endif

5. Update $\mathbf{r}_{i+1}$: Eq. (24).

6. Compute $\Delta R_{i+1} \mathbf{a}_i^*$: Eq. (23).

7. Update $\mathbf{g}_i$: Eq. (22).

8. Back to step (1).

Complexity

The number of complex multiplications in the algorithm above is approximately $Nm + 5m^2$, for data of length N and prediction order m. Most of the computational work goes into calculating the autocorrelations (26); the remaining operations are on short vectors of size equal to the iteration number plus one. In contrast, the conventional Burg method updates and correlates forward and backward prediction error vectors of size (near) N in each iteration. This leads to approximately $3Nm - m^2$ multiplications. The new algorithm thus reduces complexity whenever $m < N/3$, which is the typical case. For example, a 16th order analysis of a 20 ms speech frame sampled at 16 kHz uses about 15100 multiplications for the standard method, and 6400 for the new one. For long data sets and high filter orders, the complexity can be reduced further by computing the autocorrelations with the Fast Fourier Transform. This brings the complexity down to order $\mathcal{O}(NlogN + m^2)$.

Regularization

Since the new method operates on autocorrelations, it is trivial to add a regularization coefficient by multiplying the zero-lag autocorrelation c_0 defined in Eq. (26) with a value slightly larger than one. Such regularization is known as Tikhonov regularization in general least-squares problems. It has the effect of adding a white noise floor to the data, which improves numerical stability and reduces fluctuations in prediction coefficients.

Conclusion

We derived a new implementation of the Burg method that in typical applications requires fewer operations. The implementation is used in the SILK speech codec [7] and the Opus speech and audio codec [8].

References

[1] J. P. Burg, "Maximum Entropy Spectral Analysis," presented at the 37th Annual International Meeting, Soc. of Explor. Geophysics, Oklahoma, Oct. 1967.

[2] J. P. Burg, "Maximum Entropy Spectral Analysis," PhD thesis, Stanford University, 1975. Available from http://sepwww.stanford.edu/theses/sep06/

[3] M. Morf, et al. "Efficient solution of covariance equations for linear prediction." Acoustics, Speech and Signal Processing, IEEE Transactions on 25.5: 429-433, 1977.

[4] L. Marple Jr, "A new autoregressive spectrum analysis algorithm." Acoustics, Speech and Signal Processing, IEEE Transactions on 28.4: 441-454, 1980.

[5] N. Andersen. "Comments on the performance of maximum entropy algorithms." Proceedings of the IEEE 66.11: 1581-1582, 1978.

[6] N. Levinson, "The Wiener RMS error criterion in filter design and prediction." J. Math. Phys., v. 25, pp. 261278, 1947.

[7] http://en.wikipedia.org/wiki/SILK

[8] http://www.opus-codec.org