Stanford Exploration Project, Report 65, January 13, 1998, pages 67–89

Generalized adaptive deconvolution

Gilles Darche[1]

keywords: *algorithm, deconvolution, reflection coefficients*

ABSTRACT

In signal processing, the lattice algorithms have been widely applied to spectral analysis. Their best known example in geophysics is Burg's filtering. In fact, lattice structures can be used to construct algorithms solving any kind of prediction problems. In this paper, I derive two of these algorithms. They generalize two classical adaptive algorithms used in spectral analysis: the least-squares lattice (LSL) algorithm, and Burg's adaptive algorithm. I will apply these algorithms to the problem of multiples removal with non-stationary data; these applications will show the superiority of Burg's algorithm.

INTRODUCTION

The general prediction problem consists in predicting a data sequence $x(t)$ from the past values of another data sequence $y(t)$. In spectral analysis (or spiking deconvolution) problems, we try to estimate $y(t)$ from its own immediately past values. This is usually solved with Levinson algorithm, in terms of a prediction filter, solution of a Toeplitz system involving the autocorrelations of the data $y(t)$ at different lags.

However, we are most often interested in the residuals of the prediction, not in the filter. The alternative proposed in spectral analysis by Burg (1975) was to use a new set of unknowns, the backward and forward residuals, and auxiliary variables, the reflection coefficients; the filters themselves are never computed.

Many improvements to Burg's method have been proposed. One of them, proposed by Hale (1981), considers a time-adaptive formalism (Hale, 1981). Lee et al. (1981) and Friedlander (1982) proposed an adaptive algorithm for spectral analysis called the least-squares lattice (LSL) algorithm, which is similar in concept, though not in formulation, to Burg's algorithm. The advantage of adaptive methods is their applicability to data with statistical variations. But, indeed, I have never seen applications of these various developments to fields other than spectral analysis.

Noticing this restriction, I generalize the LSL algorithm and Hale's adaptive ver-

[1]**email:** not available

sion of Burg's algorithm to the general prediction problem, in which we predict a data sequence $x(t)$ from any other sequence $y(t)$. With this extension, I am better able to solve the prediction problem for non-stationary data.

In the first part of the paper, I describe the basic LSL algorithm used for spectral analysis. The main idea of this algorithm is that the optimal residual at a given time depends only on the *past* values of the data, not on the future values. It can also include exponential tapering. The description of the algorithm is rather long, but is necessary to understand the concepts I use throughout the paper. After the presentation of the algorithm, I extend it to the general prediction problem, using simple geometrical considerations.

In the second part of the article, I describe also the principle of the adaptive Burg's algorithm. Here, the optimal residuals depend on both future and past values of the data, but a time-varying exponential weighting is used to minimize the energy of the residuals. The extension to the general prediction problem requires however a slight transform of this algorithm, but is also relatively easy.

Finally, I apply these algorithms to the elimination of the water-bottom multiples and peglegs on a marine shot-gather. The process is actually run in the $\tau - p$ domain, to make the predictive modeling more reliable. This application will indeed show the superiority of the generalized Burg's algorithm, essentially because it is better stabilized, using both future and past values.

At the end of this paper, I add the sketches of the two algorithms.

THE LSL ALGORITHM

I present here first the basic LSL algorithm as it is used for spectral analysis: its formalism, the definitions of the forward and backward residuals, and the recursion formulas. The proofs for the recursions implied are long, and heavy: I prefer to refer to Lee et al. (1981) for elegant geometric demonstrations. The recursion formulas also provide us with a statistical tool, describing the time-statistical variations in the data, which I will briefly explain. Finally, I derive a simple version of the general LSL algorithm; Friedlander (1982) had indeed derived such an algorithm, but in a "normalized" formalism, not easy to understand intuitively.

The basic LSL algorithm: concepts and formalism

Suppose we want to predict a data sequence $y(t)$ $(t = 0, \cdots, T_{max})$ from its own past values. The basic concept of the LSL algorithm is that, to estimate the optimal residual at time T, we solve the prediction problem in a least-squares sense on the window $[0, T]$, and keep only the residual at time T. To compute the residual at time $T+1$, we would solve the problem on the window $[0, T+1]$, and again compute the residual at time $T+1$ from that solution. This sequence explains why the final residuals depend only on the past data.

In mathematical formulation, we try to find n filter coefficients $f_{n,T}(1), \cdots, f_{n,T}(n)$, minimizing:

$$\sum_{t=0}^{T} [\varepsilon_{n,T}(t)]^2 \quad \text{where} \quad \varepsilon_{n,T}(t) = y(t) - \sum_{i=1}^{n} f_{n,T}(i) y(t-i) \ .$$

I define the data vectors y_T, the residual vector $\varepsilon_{n,T}$ and the convolution matrix $A_{1,n,T}$:

$$y_T = \begin{pmatrix} y(0) \\ y(1) \\ \vdots \\ y(T) \end{pmatrix} \ , \ \varepsilon_{n,T} = \begin{pmatrix} \varepsilon_{n,T}(0) \\ \varepsilon_{n,T}(1) \\ \vdots \\ \varepsilon_{n,T}(T) \end{pmatrix} \ , \ A_{1,n,T} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ y(0) & 0 & & \vdots \\ y(1) & y(0) & \ddots & \\ & y(1) & \ddots & 0 \\ & & \ddots & y(0) \\ & & & y(1) \\ \vdots & \vdots & \vdots & \vdots \\ y(T-1) & y(T-2) & \cdots & y(T-n) \end{pmatrix}$$

The prediction coefficients minimizing the residual energy are then equal to:

$$f_{n,T} = [f_{n,T}(1), \cdots, f_{n,T}(n)]' = (A'_{1,n,T} A_{1,n,T})^{-1} A'_{1,n,T} y_T \ .$$

We are solving a minimization problem on the data window $[0, T]$, as indicated by the subscript T. Especially, the filter coefficients $f_{n,T}$ depend on T: if we work on the window $[0, T+1]$, the least-squares solution will be different, giving $f_{n,T+1}$. The second subscript n means that we are solving a problem for a filter of length n.

In the classical approach, the final outputs are $\varepsilon_{n,T_{max}}(T)$: we solve the problem on the whole window $[0, T_{max}]$. In this case, the final residuals also depend on the future data, because the filter coefficients are computed with the whole set of data ($t = 0, \cdots, T_{max}$). On the other hand, the concept of the LSL algorithm is that the final outputs are the residuals $\varepsilon_{n,T}(T)$ for all $T = 0, \cdots, T_{max}$: this enables the algorithm to adjust to variations in the statistics of the data.

So, the LSL algorithm uses recursions for the residuals when we increase the order n (order-updating), and the time T (time-updating). We can also include exponential weighting, if we use a positive factor $\lambda \leq 1$ to minimize:

$$\sum_{t=0}^{T} \lambda^{T-t} [\varepsilon_{n,T}(t)]^2 \ = \ \sum_{t=0}^{T} \lambda^{T-t} [y(t) - \sum_{i=1}^{n} f_{n,T}(i) y(t-i)]^2 \ .$$

The matrix $A'_{1,n,T} A_{1,n,T}$ would be the usual Toeplitz matrix involved in the Wiener-Levinson algorithm, if we took only the data up to $t = T - n$ into account, but the form of $A_{1,n,T}$ is necessary for the time-update recursions. This matrix $A_{1,n,T}$ will actually never be built, because the aim of lattice methods is to work only with the residuals $\varepsilon_{n,T}$, and not with the filter coefficients.

Forward residuals

The forward residuals $\varepsilon_{n,T}$ can be expressed in terms of orthogonal projections. Effectively, defining $P_{1,n,T} = A_{1,n,T}(A'_{1,n,T}A_{1,n,T})^{-1}A'_{1,n,T}$, we have:

$$\varepsilon_{n,T} = y_T - A_{1,n,T}f_{n,T} = (I - P_{1,n,T})y_T \ .$$

Because $P^2_{1,n,T} = P'_{1,n,T} = P_{1,n,T}$, the operator $P_{1,n,T}$ is an **orthogonal projector**. It projects the data vector y_T on the space $Y_{1,n,T}$ spanned by the columns of $A_{1,n,T}$. These columns are simply the past observations (or shifted data), Zy_T to $Z^n y_T$, where Z is the unit-delay shift operator.

The *predictable* part of y_T is $P_{1,n,T}y_T$, the **orthogonal projection** of y_T on $Y_{1,n,T}$. The *residuals* $\varepsilon_{n,T}$ are then orthogonal to this space: they are called the **orthogonal complement** of y_T on $Y_{1,n,T}$. Their geometric interpretation is illustrated in Figure **??**. Defining the operator $P^{\perp}_{1,n,T} = I - P_{1,n,T}$, the forward residuals $\varepsilon_{n,T}$ are $P^{\perp}_{1,n,T}y_T$; $P^{\perp}_{1,n,T}$ is also an orthogonal projector.

So, we need to compute $P_{1,n,T}y_T$. But this projection is easy to compute, if we already know an orthogonal basis $u_1, \cdots, u_n$ of the space $Y_{1,n,T}$. Effectively, the projection of y_T is:

$$P_{1,n,T}y_T = \frac{y'u_1}{u'_1u_1}u_1 + \cdots + \frac{y'u_n}{u'_nu_n}u_n \ .$$

The backward residuals will in fact form this orthogonal basis.

Backward residuals

We define the vectors of backward residuals $r_{k,T-1}$ with the recursions:

$$\begin{cases} r_{0,T-1} & = & Zy_T \\ r_{k,T-1} & = & Z^{k+1}y_T - \dfrac{r'_{0,T-1}(Z^{k+1}y_T)}{r'_{0,T-1}r_{0,T-1}}r_{0,T-1} \cdots - \dfrac{r'_{k-1,T-1}(Z^{k+1}y_T)}{r'_{k-1,T-1}r_{k-1,T-1}}r_{k-1,T-1} \ . \end{cases} \tag{1}$$

The index $T-1$ appears because these residuals can all be expressed as Z applied to a known vector: $r_{0,T} = y_T$ for example. These recursions are in fact equivalent to performing a Gram-Schmidt orthogonalization of the space $(Zy_T, \cdots, Z^n y_T)$. Then, for any k, the space $(r_{0,T-1}, \cdots, r_{k,T-1})$ is equal to $(Zy_T, \cdots, Z^{k+1}y_T)$; also, $r_{k,T-1}$ is orthogonal to $(r_{0,T-1}, \cdots, r_{k-1,T-1})$. So, with these recursions, we see that $r_{k,T-1}$ is the **orthogonal complement** of the shifted data $Z^{k+1}y_T$ on $(Zy_T, \cdots, Z^k y_T)$:

$$r_{k,T-1} = P^{\perp}_{1,k,T}(Z^{k+1}y_T) \qquad \text{for all } k = 0, \cdots, n-1 \ .$$

In other words, $r_{k,T-1}(t)$ is the residual obtained from the optimal prediction of $y(t-(k+1))$ with the *future* values $y(t-1), \cdots, y(t-k)$; this explains the term "backward" residuals. The geometrical interpretations of $\varepsilon_{k,T}$ and $r_{k,T-1}$ appear together on Figure **??**.

Geometrical interpretation of the residuals $\varepsilon_{k,T}$ and $r_{k,T-1}$.
$\varepsilon_{k,T}$ is the orthogonal complement of the data y_T on $Y_{1,k,T}$.
The predictable part of y_T of order k is the orthogonal projection of y_T on $Y_{1,k,T}$.
$r_{k,T-1}$ is the orthogonal complement of the shifted data $Z^{k+1}y_T$ on $Y_{1,k,T}$.
The new space of regressors $Y_{1,k+1,T}$ is $Y_{1,k,T} \oplus Z^{k+1}y_T$.
$\varepsilon_{k+1,T}$ is the orthogonal complement of y_T on $Y_{1,k+1,T}$.
$\varepsilon_{k+1,T} - \varepsilon_{k,T}$ is equal to $r_{k,T-1}$ multiplied by the reflection coefficient K^r_{k+1}.

Finally, knowing that the forward residuals $\varepsilon_{k,T}$ are the orthogonal complement of y_T on $(Zy_T, \cdots, Z^k y_T)$, so on $(r_{0,T-1}, \cdots, r_{k-1,T-1})$, we have the recursive formulas:

$$\left\{ \begin{array}{lcl} \varepsilon_{0,T} & = & y_T \\ \varepsilon_{k,T} & = & y_T - \dfrac{r'_{0,T-1}y_T}{r'_{0,T-1}r_{0,T-1}} r_{0,T-1} \cdots - \dfrac{r'_{k-1,T-1}y_T}{r'_{k-1,T-1}r_{k-1,T-1}} r_{k-1,T-1} \ . \end{array} \right. \tag{2}$$

Recursions: order updating

Here, T is fixed, and the order of the prediction filter increases from 1 to the desired order. The recursions implied for the backward residuals need the weighting to be uniform (or exponential), essentially because the recursions need a shift of these residuals; this condition is also explained in Lee et al. (1981). These recursions are similar to those of Burg's algorithm, and are directly derived from equations (1) and (2):

$$\begin{array}{c} \varepsilon_{k+1,T} = \varepsilon_{k,T} - \dfrac{\Delta_{k+1,T}}{R^r_{k,T-1}} r_{k,T-1} \ , \qquad r_{k+1,T} = r_{k,T-1} - \dfrac{\Delta_{k+1,T}}{R^\varepsilon_{k,T}} \varepsilon_{k,T} \\ \text{where} \left\{ \begin{array}{lcl} \Delta_{k+1,T} & = & r'_{k,T-1}y_T = \varepsilon'_{k,T} r_{k,T-1} \\ R^r_{k,T-1} & = & r'_{k,T-1} r_{k,T-1} \\ R^\varepsilon_{k,T} & = & \varepsilon'_{k,T}\varepsilon_{k,T} \end{array} \right. \end{array} \tag{3}$$

$\Delta_{k+1,T}$ is called the **partial correlation** of the residuals; $R^r_{k,T-1}$ is the **covariance** of the backward residuals, $R^\varepsilon_{k,T}$ is the **covariance** of the forward residuals. Notice that updating of the forward residuals has already been illustrated by Figure **??**. We can also update these covariances as follows:

$$R^\varepsilon_{k+1,T} = R^\varepsilon_{k,T} - \frac{\Delta^2_{k+1,T}}{R^r_{k,T-1}} \quad \text{and} \quad R^r_{k+1,T} = R^r_{k,T-1} - \frac{\Delta^2_{k+1,T}}{R^\varepsilon_{k,T}} \tag{4}$$

Finally, we can define two reflection coefficients $K^\varepsilon_{k+1,T}$ and $K^r_{k+1,T}$, to give a compact expression of equation (**??**):

$$K^\varepsilon_{k+1,T} = \frac{\Delta_{k+1,T}}{R^\varepsilon_{k,T}} \quad \text{and} \quad K^r_{k+1,T} = \frac{\Delta_{k+1,T}}{R^r_{k,T-1}} \ ,$$

$$\varepsilon_{k+1,T} = \varepsilon_{k,T} - K^r_{k+1,T} r_{k,T-1} \quad \text{and} \quad r_{k+1,T} = r_{k,T-1} - K^\varepsilon_{k+1,T}\varepsilon_{k,T} \ .$$

Notice also that we have *two* reflection coefficients. Burg's algorithm only uses *one* coefficient. Normally, for an infinite stationary sequence y_T, these coefficients should be equal. However, in most practical applications, this is not the case. Burg's algorithm corresponds in fact to a particular choice of the reflection coefficient, which minimizes the total energy $\sum_t(\varepsilon_{p+1}^2(t) + r_{p+1}^2(t))$:

$$\frac{2}{K_{p+1}} = \frac{1}{K_{p+1}^{\varepsilon}} + \frac{1}{K_{p+1}^{r}}\,, \qquad \text{or} \qquad K_{p+1} = \frac{2K_{p+1}^{\varepsilon}K_{p+1}^{r}}{K_{p+1}^{\varepsilon} + K_{p+1}^{r}} = \frac{2\sum \varepsilon_p(t) r_p(t)}{\sum \varepsilon_p^2(t) + \sum r_p^2(t)}\,.$$

The advantage of Burg's algorithm is that the reflection coefficients K_p are always bounded (by 1), preventing the algorithm from exploding numerically. On the other hand, the backward residuals it uses don't provide us with an orthogonal basis of the space of the regressors, and we cannot use projection operators to solve the prediction problem.

Recursions: time updating

In this part, the length of the filter is fixed, but we increase the length of the processing window (from $T-1$ to T). Equation (3) tells us how to compute $\varepsilon_{k+1,T}$ from $\varepsilon_{k,T}$. But actually, we are only interested in the $\varepsilon_{k,T}(T)$, and we don't want to compute, nor store, the other values $\varepsilon_{k,T}(t)$ $(t = 0,\cdots,T-1)$ which form the vector $\varepsilon_{k,T}$. It seems difficult then to compute $\Delta_{k+1,T}$, $R_{k,T}^{\varepsilon}$, or $R_{k,T-1}^{r}$!

However, time-updating formulas also exist for these values, which only imply $\varepsilon_{k,T}(T)$ and $r_{k,T-1}(T)$. Their derivation is long, and I refer to Lee et al. (1981) for an elegant proof. These authors use two other intermediate variables, the angles $\theta_{1,k,T}$, and $\theta_{0,k,T}$; especially, $\theta_{1,k,T}$ is defined as:

$$\sin^2\theta_{1,k,T} = \pi_T'(P_{1,k,T}\pi_T) \qquad \text{with } \pi_T = (0,\cdots,0,1)'$$

The recursions are now:

$$\left\{\begin{array}{lcl}
\Delta_{k+1,T} & = & \Delta_{k+1,T-1} + \dfrac{\varepsilon_{k,T}(T).r_{k,T-1}(T)}{\cos^2\theta_{1,k,T}} \\[2ex]
R_{k,T}^{\varepsilon} & = & R_{k,T-1}^{\varepsilon} + \dfrac{\varepsilon_{k,T}^2(T)}{\cos^2\theta_{1,k,T}} \\[2ex]
R_{k,T}^{r} & = & R_{k,T-1}^{r} + \dfrac{r_{k,T}^2(T)}{\cos^2\theta_{0,k-1,T}} \\[2ex]
\cos^2\theta_{1,k+1,T} & = & \cos^2\theta_{1,k,T} - \dfrac{r_{k,T-1}^2(T)}{R_{k,T-1}^{r}} \\[2ex]
\cos^2\theta_{0,k,T} & = & \cos^2\theta_{1,k,T} - \dfrac{\varepsilon_{k,T}^2(T)}{R_{k,T-1}^{\varepsilon}}
\end{array}\right. \tag{5}$$

In conclusion, the recursions (3), (4), and (5) give us the entire set of recursions we need for the basic LSL algorithm. The initialization procedures are precised in the sketch of the general LSL algorithm I add at the end of the paper. The most important one is the choice of an a priori value for the covariances, which will stabilize the process by avoiding divisions by 0: it has more or less the stabilizing effect of a prewhitening process on the data. After a few samples, the algorithm should not be sensitive anymore to this a priori value.

Exponential weighting

The weighting consists of using a tapering coefficient λ to minimize the weighted norm of the residuals:

$$\sum_{t=0}^{T} \lambda^{T-t} \varepsilon_{n,T}^2(t) \ .$$

It forces the algorithm to forget the "old data" in the minimization problem: it extends the adaptive properties of the algorithm. We can use the recursion results derived up to now, simply adapted to another choice of scalar product:

$$x.y = \sum_t \lambda^{T-t} x(t) y(t) \ .$$

Actually, only the covariances and correlation recursions are modified:

$$\begin{aligned} \Delta_{k+1,T} &= \lambda \Delta_{k,T} + \frac{\varepsilon_{k,T}(T).r_{k,T-1}(T)}{\cos^2 \theta_{1,k,T}} \\ R^{\varepsilon}_{0,T} &= R^{r}_{0,T} = \lambda R^{\varepsilon}_{0,T-1} + y^2(T) \end{aligned} \qquad (6)$$

Statistical signification of $\theta_{1,k,T}$

The angle $\theta_{1,k,T}$ has a statistical interpretation. Effectively, with the definition of $P_{1,k,T}$, we see that:

$$\sin^2 \theta_{1,k,T} = (A'_{1,k,T}\pi_T)'(A'_{1,k,T}A_{1,k,T})^{-1}(A'_{1,k,T}\pi_T) \ .$$

Notice first that $A'_{1,k,T}\pi_T = (y(T-1), \cdots, y(T-k))'$. Moreover $A'_{1,k,T}A_{1,k,T}$ contains the lags $0, \cdots, k-1$ of the autocorrelation of the data: thus, it is the covariance matrix $R^y_{k,T}$ of the data (up to time T). So, the value $\exp(-0.5 \sin^2 \theta_{1,k,T})$ can be compared to the probability of the sequence $\tilde{y} = (y(T-1), \cdots, y(T-k))'$ according to the covariance matrix $R^y_{k,T}$:

$$\text{Proba}(\tilde{y}) = \frac{1}{||2\pi R^y_{k,T}||^{\frac{n}{2}}} \exp(-\frac{1}{2}\tilde{y}'(R^y_{k,T})^{-1}\tilde{y}) = \frac{1}{||2\pi R^y_{k,T}||^{\frac{n}{2}}} \exp(-\frac{1}{2}\sin^2 \theta_{1,k,T}) \ .$$

A small value of $\theta_{1,k,T}$ means that the previous samples don't deviate from the general statistics of the data. On the contrary, a sudden burst of noise, or a new strong seismic arrival, will produce a large value of $\sin^2 \theta_{1,k,T}$. This occurrence will also perturb strongly the covariances $R^{\varepsilon}_{k,T}$, $R^{r}_{k,T}$, and the correlation $\Delta_{k+1,T}$, whose updatings involve a division by $\cos^2 \theta_{1,k,T}$ (small in that case). So the variable $\sin^2 \theta_{1,k,T}$ can be assimilated to a likelihood variable, and used for detection of unexpected events.

General LSL algorithm

We are interested in the prediction of a time series x_T from a related process y_T. We want to minimize:

$$\sum_{t=0}^{T}(\varepsilon^x_{n,T}(t))^2 = \sum_{t=0}^{T}[x(t) - f_{n,T}(1)y(t-1) - \cdots - f_{n,T}(n)y(t-n)]^2 \ .$$

Once again, we are interested only in the residual $\varepsilon^x_{n,T}(T)$, so that the residuals are affected only by past values, and not by future values. This minimization problem leads to:

$$\begin{aligned} f_{n,T} &= (A'_{1,n,T}A_{1,n,T})^{-1}A'_{1,n,T}.x_T \\ \varepsilon^x_{n,T} &= x_T - A_{1,n,T}f_{n,T} = (I - P_{1,n,T}).x_T \end{aligned}$$

If we are interested in the residuals $\varepsilon^x_{k,T}$ themselves, then we have to find the **orthogonal complement of** x_T on the space $Y_{1,k,T} = (Zy_T, \cdots, Z^k y_T)$. So, the idea is to use the backward residuals of the self-prediction of y_T to form an orthogonal basis of the space $Y_{1,k,T}$, and then project the vector x_T onto this particular basis. Consequently, we will lead two recursions at the same time: one to compute the variables $\varepsilon_{k,T}$ and $r_{k,T}$ of the *self-prediction* of y_T, and the other to compute the residuals $\varepsilon^x_{k,T}$ of the joint prediction of x_T from y_T.

The order-updating of the residuals $\varepsilon^x_{k,T}$ is very similar to the updating of the residuals $\varepsilon_{k,T}$. Effectively,

$$\begin{aligned} \varepsilon^x_{k+1,T} &= x_T - P_{1,k+1,T}.x_T \\ &= x_T - \frac{r'_{0,T-1}x_T}{r'_{0,T-1}r_{0,T-1}}r_{0,T-1} - \cdots - \frac{r'_{k,T-1}x_T}{r'_{k,T-1}r_{k,T-1}}r_{k,T-1} \\ &= \varepsilon^x_{k,T} - \frac{r'_{k,T-1}x_T}{r'_{k,T-1}r_{k,T-1}}r_{k,T-1} \ . \end{aligned} \tag{7}$$

We can define the crosscorrelation $\Delta^x_{k+1,T}$ and the reflection coefficient $K^x_{p+1,T}$:

$$\Delta^x_{k+1,T} = r'_{k,T-1}x_T \ , \qquad \text{and} \qquad K^x_{p+1,T} = \frac{\Delta^x_{k+1,T}}{R^r_{k,T-1}} \ .$$

The time-updating is similar to equation (5). I use here an equation derived from Lee et al. (1981), saying that for any vectors u_T and v_T, we have the relationship:

$$u'_T(P^{\perp}_{1,k,T}v_T) - u'_{T-1}(P^{\perp}_{1,k,T-1}v_{T-1}) = [u'_T(P^{\perp}_{1,k,T}\pi_T)].[\pi'_T(P^{\perp}_{1,k,T}v_T)].\frac{1}{\cos^2\theta_{1,k,T}} \ . \tag{8}$$

This equation, used with $(u_T, v_T) = (y_T, Z^{k+1}y_T)$, leads to the time-updating of $\Delta_{k+1,T}$ (equation (5)) in the self-prediction of y_T. Here, using the equation (8) with $u_T = x_T$ and $v_T = Z^{k+1}y_T$, we get the time-update recursion:

$$\Delta^x_{k+1,T} = \Delta^x_{k+1,T-1} + \frac{\varepsilon^x_{k,T}(T)r_{k,T-1}(T)}{\cos^2\theta_{1,k,T}} \ . \tag{9}$$

The use of exponential tapering also gives a recursion similar to equation (6):

$$\Delta^x_{k+1,T} = \lambda \Delta^x_{k+1,T-1} + \frac{\varepsilon^x_{k,T}(T) r_{k,T-1}(T)}{\cos^2\theta_{1,k,T}} \; .$$

In conclusion, the recursions (7) and (9), joined to the previous recursions (3), (4), and (5) of the basic LSL algorithm, give us the entire set of recursions necessary for the general LSL algorithm.

Conclusion

The complete set of recursions (equations (3), (5), (4), (7), (9)), along with the initializations, is formulated in the sketch of the general LSL algorithm at the end of the paper (appendix A). As in any recursive method, we have access to the residuals for all intermediate lengths of the prediction filter. The basic concept is that a residual at time T is computed from the minimization problem on the window $[0, T]$, and does not depend on future data, which is a first way to make the algorithm adaptive. The second way is to introduce an exponential tapering, which forces the algorithm to forget the "old data".

ADAPTIVE BURG-TYPE FILTERING

In this part, I present first the adaptive version of Burg's algorithm as it was first described by Hale (1981). I will try to keep the same notations as those used in the description of the LSL algorithm. Actually, the derivation of this algorithm seems to me more intuitive than mathematically rigorous, but the algorithm is a good approximation of the exact solution. Then, I will slightly modify this algorithm, to use two reflection coefficients as in the LSL algorithm, rather than one as in Burg's formalism. Finally, I will extend the algorithm to the general prediction problem.

Adaptive prediction error filtering

We still have a data sequence $y(t)$ $(t = 0, \cdots, T_{max})$. For each time T and order k, we define the forward and backward residuals $\varepsilon_k(T)$ and $r_k(T)$; a single reflection coefficient K_k and Burg's recursions are used:

$$\left\{ \begin{array}{l} \varepsilon_0(T) = y(T) \, , r_0(T) = y(T) \\ \varepsilon_k(T) = \varepsilon_{k-1}(T) - K_k r_{k-1}(T-1) \\ r_k(T) = r_{k-1}(T-1) - K_k \varepsilon_{k-1}(T) \end{array} \right. \tag{10}$$

To compute the reflection coefficients K_k, we minimize the weighted energy of the forward and backward residuals of order k. The weights are centered on the particular time T:

$$\begin{aligned}\mathcal{E}(K_k) &= \sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}(\varepsilon_k^2(t) + r_k^2(t)) \\ &= \sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}[(\varepsilon_{k-1}(t) - K_k r_{k-1}(t-1))^2 + (r_{k-1}(t-1) - K_k \varepsilon_{k-1}(t))^2] \, .\end{aligned}$$

It is important to notice that the residuals depend here on the whole set of data, since the energy $\mathcal{E}(K_k)$ is a summation on the whole time window. However, this expression of the energy induces an adaptive formalism, since more emphasis is put on the residuals near the time T of interest than of the residuals far from this time. Minimizing this expression with respect to K_k leads to a time-dependent reflection coefficient:

$$K_{k,T} = \frac{2\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}\varepsilon_{k-1}(t) r_{k-1}(t-1)}{\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}(\varepsilon_{k-1}^2(t) + r_{k-1}^2(t-1))} \, . \tag{11}$$

Then Hale (1981) showed that the reflection coefficients $K_{k,T}$ can be easily computed, if the numerator and denominator of expression (11) are split between past and future summations. For example, the numerator becomes:

$$\begin{aligned}N_k(T) &= 2\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}\varepsilon_{k-1}(t) r_{k-1}(t-1) \\ &= 2\sum_{t=0}^{T-1} \lambda^{|T-t|}\varepsilon_{k-1}(t) r_{k-1}(t-1) \; + \; 2\sum_{t=T}^{T_{max}-1} \lambda^{|T-t|}\varepsilon_{k-1}(t) r_{k-1}(t-1) \\ &= N_k^-(T) \; + \; N_k^+(T) \, .\end{aligned}$$

In the same way, we can split the denominator between $D_k^-(T)$ (summation up to $T-1$), and $D_k^+(T)$ (summation from T to T_{max}). Then, it is straightforward to show that:

$$\left\{\begin{array}{lcll} N_k^-(T+1) &=& \lambda(2\varepsilon_{k-1}(T) r_{k-1}(T-1) + N_k^-(T)) & ; \; N_k^-(k) = 0. \\ D_k^-(T+1) &=& \lambda(\varepsilon_{k-1}^2(T) + r_{k-1}^2(T-1) + D_k^-(T)) & ; \; D_k^-(k) = 0. \\ N_k^+(T-1) &=& 2\varepsilon_{k-1}(T) r_{k-1}(T-2) + \lambda N_k^+(T) & ; \; N_k^+(T_{max}) = 0. \\ D_k^+(T-1) &=& \varepsilon_{k-1}^2(T) + r_{k-1}^2(T-2) + \lambda D_k^+(T) & ; \; D_k^+(T_{max}) = 0. \end{array}\right. \tag{12}$$

These recursions, together with the recursions (10) and the expression (11), form the adaptive version of Burg's algorithm.

Is this algorithm rigorous?

My concern about the rigor of the algorithm comes mainly from the form of the recursions (10). Hale supposes that we can extend the exact recursions of the non-adaptive Burg's algorithm to the adaptive case. However, some contradictions appear.

First, to minimize the energy $\mathcal{E}(K_k)$, we consider that the same reflection coefficient is used for all the residuals at different times; however, the minimization results in a time-dependent coefficient. Second, the recursions (10) on the backward residuals should be valid only when uniform or exponential weighting is used; here the weights are pseudo-exponential (because they are symmetric), and the recursions should not be valid. So, I have the impression that this algorithm will effectively *decrease* the energy of the residuals, but might as well not minimize it, because we don't give enough degrees of freedom to let these residuals evolve from iteration to iteration.

I have tried to use the same formalism as for the LSL algorithm, but I could not find any logical relation between the two algorithms. The ideal would be to use again the residuals $\varepsilon_{k,T}(t)$ and $r_{k,T}(t)$, where the subscript T reappears, because we want to minimize the energy:

$$\sum_{t=0}^{T_{max}} \lambda^{|T-t|} \varepsilon_{k,T}^2(t) \ .$$

Unlike in Burg's adaptive filtering, I prefer to keep the backward residuals outside the summation, so that we minimize their energy at the same time, and separately: they are still an orthogonal basis of the space of regressors. The final output is again $\varepsilon_{n,T}(T)$: the residual at time T for the minimization problem with the weights centered on T. Actually, the definitions of the forward and backward residuals in term of projection operators are still the same. We should just use a different scalar product:

$$x.y = \sum_{t=0}^{T_{max}} \lambda^{|T-t|} x(t)y(t) \qquad \text{rather than} \qquad x.y = \sum_{t=0}^{T} \lambda^{T-t} x(t)y(t) \ .$$

However, some problems appear in the order-updating recursions. First, as I said, the updating of the backward residuals is not rigorous: it implies a shift of these residuals, and we need an orthogonality condition to be preserved even after this shift. This is not true with the weights of Burg's adaptive algorithm. This problem had already been approached by Claerbout (1977), who derived a Burg-type algorithm for weighted prediction problems. However, Claerbout recognized that he didn't know exactly what this algorithm was solving. I tried it on a simple numerical problem, and I found an error of 20 % between the exact solution and the solution of the algorithm, which confirms that the solution of an arbitrary weighted problem cannot be reached with a totally recursive algorithm.

Moreover, the recursions in Hale's algorithm involve the values $r_{k-1,T-1}(t)$ ($t = 0, \cdots, T_{max}$) to compute the reflection coefficient $K_{k,T}^r$, while the recursion (11) uses the values $r_{k-1,t-1}(t)$ (for all t).

So, I must concede that I could not derive any mathematical formulation of Hale's algorithm comparable to the LSL algorithm. That lack of formulation makes me say that Burg's adaptive algorithm is rather intuitive.

To conclude this part, I recognize that this algorithm, exact for $\lambda = 1$, is a good approximation of the ideal solution (which may not be recursive) for λ close to 1.

Effectively, some theoretical obstacles disappear for such values of λ. First, in the minimization of $\mathcal{E}(K_k)$, we can at first hand consider that K_k is the same for all times t, even if it results in a time-varying expression: the reflection coefficient will actually be slowly varying. Then, when deriving the recursions, the weighting will be uniform on a large scale of time, which makes Burg-type recursions more plausible.

So, even if this algorithm does not exactly minimize the residuals, it will give a good approximation of the residuals. Actually, Hale himself noticed in his paper (1981) that it was necessary to keep λ close to 1.

Modification of Burg's adaptive algorithm

In Burg's algorithm, the backward residuals don't form an orthogonal basis of the space of the regressors $(Z^y, \cdots, Z^n y)$. Effectively, we minimize the energies of the forward and backward residuals not separately, but together. Thus, the recursions implied for the backward residuals do not correspond to a Gram-Schmidt orthogonalization of the space of the regressors.

Therefore, my idea is to use two reflection coefficients, K_k^r and K_k^ε, as in the LSL algorithm, so that the forward and backward residuals are minimized separately. This is simply done by writing:

$$\left\{\begin{array}{l} \varepsilon_0(T) = y(T)\ ,\ r_0(T) = y(T) \\ \varepsilon_k(T) = \varepsilon_{k-1}(T) - K_k^r r_{k-1}(T-1) \\ r_k(T) = r_{k-1}(T-1) - K_k^\varepsilon \varepsilon_{k-1}(T) \end{array}\right. \tag{13}$$

Now, to compute the reflection coefficients K_k^r and K_k^ε, we minimize separately the energies of the forward and backward residuals:

$$\left\{\begin{array}{lllll} \mathcal{E}^\varepsilon(K_k^r) & = & \sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}\varepsilon_k^2(t) & = & \sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}[\varepsilon_{k-1}(t) - K_k^r r_{k-1}(t-1)]^2 \\ \mathcal{E}^r(K_k^\varepsilon) & = & \sum_{t=0}^{T_{max}-1} \lambda^{|T-t|} r_k^2(t) & = & \sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}[r_{k-1}(t-1) - K_k^\varepsilon \varepsilon_{k-1}(t)]^2 \end{array}\right.$$

This yields to the following time-varying formulations of the reflection coefficients:

$$K_{k,T}^r = \frac{\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}\varepsilon_{k-1}(t) r_{k-1}(t-1)}{\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|} r_{k-1}^2(t-1)}\ , \tag{14}$$

$$K_{k,T}^\varepsilon = \frac{\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}\varepsilon_{k-1}(t) r_{k-1}(t-1)}{\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|} \varepsilon_{k-1}^2(t)}\ . \tag{15}$$

Both coefficients have the same numerator; actually, Burg's coefficient is their geometric mean. Once again, these coefficients can be easily computed if we split their numerators and denominators into past and future summations. Indexing the denominators with r and ε as we did with the reflection coefficients, we have now:

$$\left\{\begin{array}{lcll} N_k^-(T+1) & = & \lambda(\varepsilon_{k-1}(T)r_{k-1}(T-1)+N_k^-(T)) & ; \quad N_k^-(k)=0. \\ D_k^{\varepsilon-}(T+1) & = & \lambda(\varepsilon_{k-1}^2(T)+D_k^{\varepsilon-}(T)) & ; \quad D_k^{\varepsilon-}(k)=0. \\ D_k^{r-}(T+1) & = & \lambda(r_{k-1}^2(T-1)+D_k^{r-}(T)) & ; \quad D_k^{r-}(k)=0. \\ N_k^+(T-1) & = & \varepsilon_{k-1}(T)r_{k-1}(T-2)+\lambda N_k^+(T) & ; \quad N_k^+(T_{max})=0. \\ D_k^{\varepsilon+}(T-1) & = & \varepsilon_{k-1}^2(T)+\lambda D_k^{\varepsilon+}(T) & ; \quad D_k^{\varepsilon+}(T_{max})=0. \\ D_k^{r+}(T-1) & = & r_{k-1}^2(T-2)+\lambda D_k^{r+}(T) & ; \quad D_k^{r+}(T_{max})=0. \end{array}\right. \tag{16}$$

These recursions, joined to the formulas (13) and (15), give my modified version of Burg's adaptive algorithm. The advantage of Burg's algorithm is that it is more stable, because the reflection coefficients are always smaller than one. However, this version allows me to construct an orthogonal basis for the space of regressors of the general prediction problem, and thus leads to the corresponding algorithm.

Generalized Burg-type adaptive algorithm

I come back to the general problem: estimating a time series $x(t)$ from the past values of another general series $y(t)$. As we did in the LSL algorithm, we need to project the data vector x onto the space of regressors $(Zy, \cdots, Z^n y)$. If $\lambda = 1$, the backward residuals in the self-prediction of y form an exact orthogonal basis of this space. So, the residuals $\varepsilon_k^x(t)$ in the joint prediction of x from y are the orthogonal complement of the series $x(t)$ on this basis, and verify:

$$\varepsilon_{k+1}^x(T) = x(T) - a_0 r_0(T-1) - \cdots - a_k r_k(T-1) = \varepsilon_k^x(T) - a_k r_k(T-1)\ .$$

In the more general case (when λ is close to 1), I consider that this formulation still holds. Using a reflection coefficient K_k^x, we get:

$$\varepsilon_0^x(T) = x(T)\ , \qquad \varepsilon_k^x(T) = \varepsilon_{k-1}^x(t) - K_k^x r_{k-1}(t-1)\ . \tag{17}$$

To compute K_k^x, we minimize the energy:

$$\mathcal{E}^x(K_k^x) \;=\; \sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}(\varepsilon_k^x(t))^2 \;=\; \sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}[\varepsilon_{k-1}^x(t) - K_k^x r_{k-1}(t-1)]^2\ .$$

As I already said, this expression of the energy forces the residuals to depend on past *and* future data, because the summation symbol extends to the whole time window. But the formalism is still adaptive, because the weighting is non-uniformly centered around the time of interest T. The minimization with respect to K_k^x leads to the expression:

$$K_{k,T}^x = \frac{\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|}\varepsilon_{k-1}^x(t) r_{k-1}(t-1)}{\sum_{t=0}^{T_{max}-1} \lambda^{|T-t|} r_{k-1}^2(t-1)}\ . \tag{18}$$

The denominator is indeed the one involved in the computation of the reflection coefficient $K^r_{k,T}$, and we already saw how we can compute it recursively. The numerator can also be split into past and future summations $N_k^{x-}(t)$ and $N_k^{x+}(t)$ verifying:

$$\left\{ \begin{array}{lcll} N_k^{x-}(T+1) & = & \lambda(\varepsilon^x_{k-1}(T) r_{k-1}(T-1) + N_k^{x-}(T)) & ; \; N_k^{x-}(k) = 0. \\ N_k^{x+}(T-1) & = & \varepsilon^x_{k-1}(T) r_{k-1}(T-2) + \lambda N_k^{x+}(T) & ; \; N_k^{x+}(T_{max}) = 0. \end{array} \right. \tag{19}$$

In conclusion, the idea of the algorithm is to perform at the same time an adaptive self-prediction of the data $y(t)$ with my modified version of Burg's adaptive algorithm, and the adaptive prediction of x from y with the recursions derived in this section. The self-prediction of y is processed with the recursion formulas (13), (15), and (16). The prediction of x is performed with the recursions (17), (18), and (19).

Conclusion

The complete set of recursions for the general Burg-type adaptive algorithm is reformulated at the end of the paper (appendix B). Though a residual depends on the whole data set, more emphasis is put in the minimization on the residuals near the time of interest, by using an exponential weighting centered on this particular time. However, I expect to achieve good results only if the exponent λ is chosen close to 1, since the algorithm appears to me as an approximation of the exact algorithm.

EXAMPLE: MULTIPLES REMOVAL

Theoretical model

In this case, we are trying to predict the data sequence x_T from its lagged past values. To do that, we are minimizing:

$$\sum_{t=0}^{T} (\varepsilon_{n,T}(t))^2 = \sum_{t=0}^{T} [x(t) - f_{n,1} x(t-L) - \cdots - f_{n,n} x(t - L - (n-1))]^2 .$$

L is the lag of the prediction. So we can apply my adaptive algorithms, taking $y_T = Z^L x_T$, or $y(t) = x(t-L)$. To have an adequate prediction model, I take a marine shot-gather, where I will try to remove the water-bottom multiples and peglegs. L should be the two-way traveltime in the water trench.

However, the prediction model is not adequate in the time-offset domain, because the lag of the water-bottom multiples is not constant on a non-0 offset trace. Actually, this lag is a function of the angle of incidence θ of the wavefront at the receiver. As the parameter $p = \sin\theta / V$ is constant along a ray, this lag depends only upon the p-parameter of the incident ray. So, to make the prediction model more appropriate, it is better to transform the data to the $\tau - p$ domain with a slant-stack transform, and apply the prediction process in this domain.

Calling V_W the water velocity, the variation of the lag L with the p-parameter is given by the relationship:

$$L(p) = L(0).\sqrt{1 - p^2 V_W^2} \ .$$

Thus, after having transformed the input gather to the $\tau - p$ domain, I will apply the adaptive algorithms on each trace, with $y_T = Z^L x_T$, and L given by the previous relation. However, I recognize that this method should not be efficient for large p-parameters, because the lag L will be very small (close to 1): the process will try at the same time to remove the multiples and to compress the signal. But for my purpose, which is a comparison between different adaptive methods, I consider it is sufficient.

Results on the data

I use the shot-gather wz27 (Western Geophysical, Alaska), which is described and displayed on Figure 1. The ocean floor being hard, the water-bottom multiples and peglegs are strong, and you can notice them on Figure 1; the lag at 0-offset is approximately 120 ms.

The figure 1 also displays the $\tau - p$ transform of this shot-gather. The strong events at $p = 0.7 \ s.km^{-1}$ are due to the presence of strong direct arrivals, and head-waves. The water-bottom peg-legs are still very easy to recognize. Notice also that the event at 1.5 seconds is stronger than the previous events. Actually, it seems that the primaries are stronger with time; this property justifies the use of adaptive algorithms.

To see the effects of this process, I apply three algorithms, with the same length for the filter ($n = 30$). The first one consists of a block method: I solve the minimization problem on time windows of 800 msec (200 samples), with a classical Levinson algorithm; the windows overlap by half their length. The second is the LSL algorithm, with no tapering ($\lambda = 1$). It is interesting to observe what happens before the event at 1.5 sec, because the LSL residuals before this event are supposed not to be influenced by it. The third algorithm is the general Burg's adaptive filtering, with $\lambda = 0.97$, and no windowing. Each of these algorithms is applied on each trace separately.

I display the results on Figure 2, with which we can compare the outputs of the three processes with the input in the $\tau - p$ domain on Figure 1.

The Burg-type algorithm seems more robust that the LSL algorithm. The different multiples are efficiently suppressed. Effectively, the Burg-type algorithm takes into account the events after 1.5 sec: they provide a lot of information on the multiplication process, because they are stronger than the previous events. On the other hand, up to this time (1.5 sec), the LSL algorithm does not take them into account, so that the process is less stable. Notice also a better lateral continuity on the residuals of

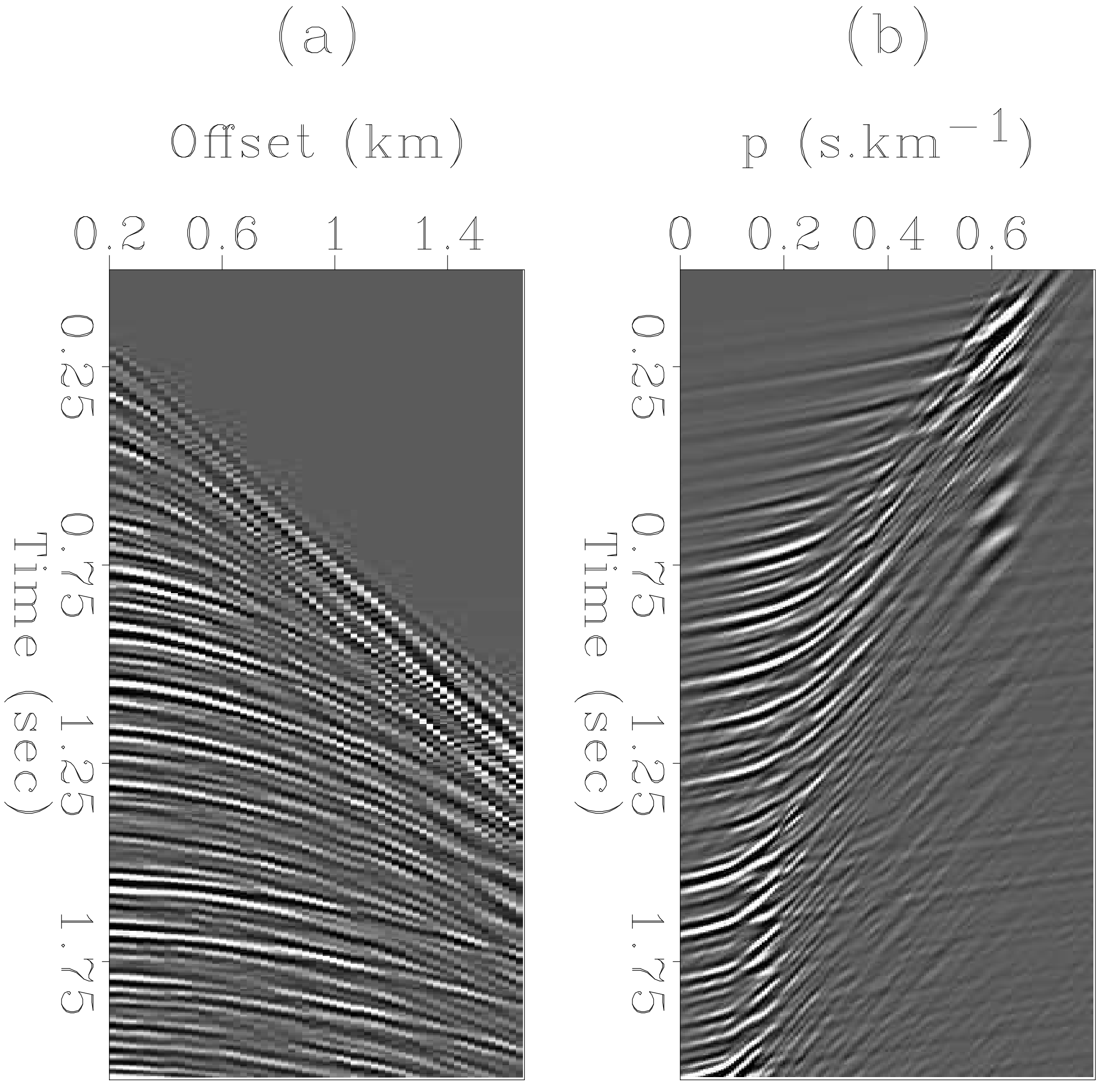

Figure 1: Shot-gather wz27. (a) Time-offset section: dt=4 $msec$, dx=50 m. (b) $\tau - p$ domain: p=0., dp=0.02 $s.km^{-1}$. In both cases, strong peglegs multiples can be seen, with a lag approximately equal to 120 msec. gilles1-H27input [NR]

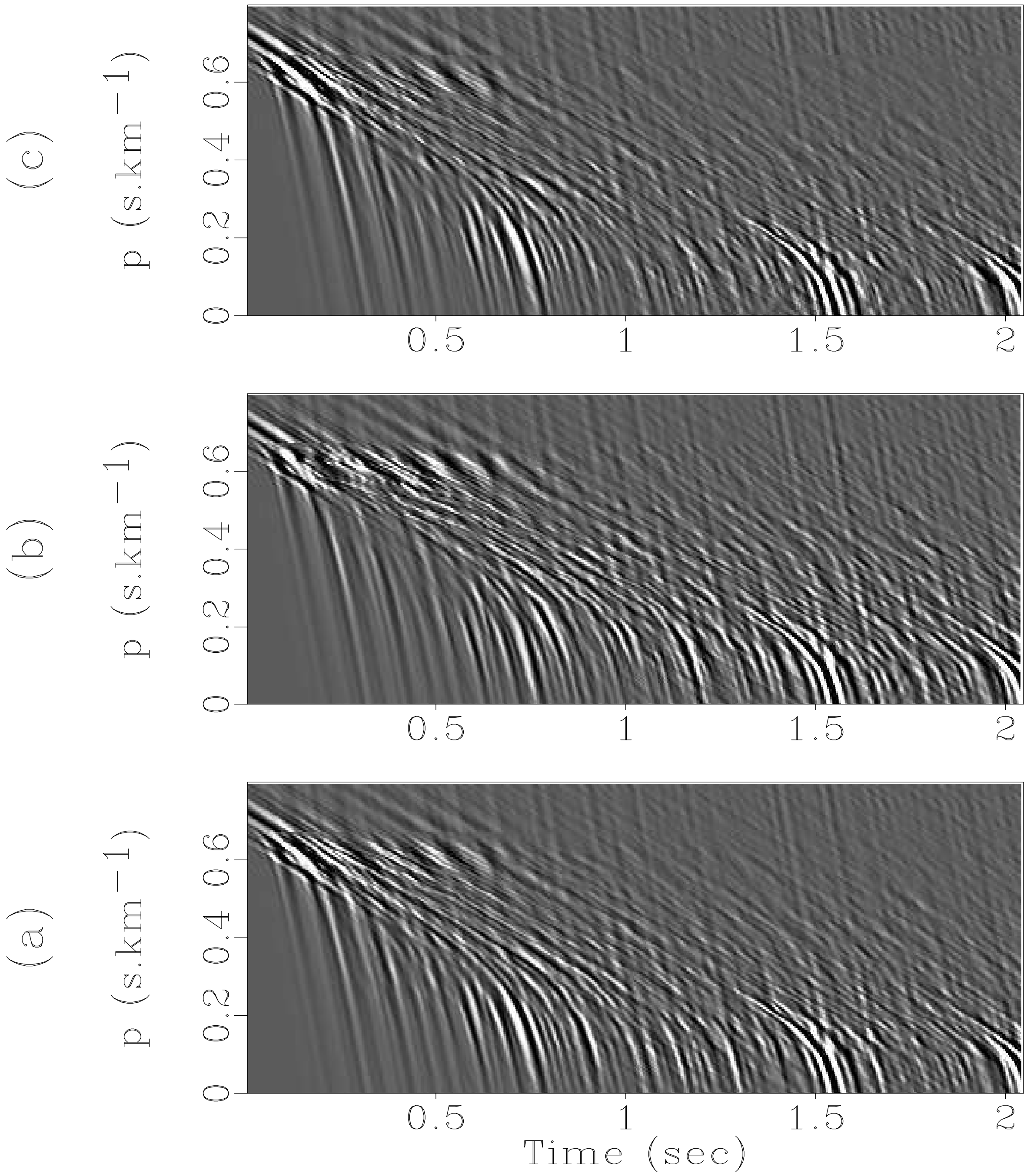

Figure 2: Removal processing in the $\tau - p$ domain. (a) Block method. (b) LSL algorithm. (c) Adaptive Burg-type algorithm. The peglegs are better attenuated by the LSL and Burg-type methods than by the block algorithm. However, a new undetermined event (1.2 sec) appears after LSL processing; it is not present on the output of the Burg-type algorithm. gilles1-H27topdemul [NR]

the Burg-type algorithm. This suggests a better numerical stability of this algorithm, less sensitive to incoherent noise.

A doubt could still persist about the undetermined event at 1.2 sec on the residuals of the LSL algorithm. However, I transformed back the residuals in the $\tau - p$ domain to the time-offset domain, as shown on Figure 3. After applying a velocity analysis on the LSL residuals, it appeared that this event was a pegleg multiple of another primary at 0.7sec. Was it an artifact of the process, or a real unpredictable multiple? I don't know indeed, but it confirms the instability of this algorithm.

Suggested improvements

Actually, I came to the idea of the Burg-type algorithm when I noticed the instability of the LSL algorithm. But I still tried to improve the LSL method. Particularly noticing the lack of lateral continuity, I tried to derive a multichannel formalism. Actually, the formalism developed by Friedlander (1982) is a multichannel formalism. It still estimates the optimal residuals for each trace, as though we wanted to estimate one optimal filter for each trace. However, it considers also the correlations between traces. For example, if we have m traces, the covariances R^r, R^ε, and Δ are $m \times m$ matrices, and the algorithm would require the inversion of these $m \times m$ matrices at each iteration! As you see, it would be extremely fastidious and impractical to implement such an algorithm, as soon as we have more than 4 or 5 traces.

Because of this difficulty, I tested multichannel procedures similar to Burg's multichannel processing, in which we average the reflection coefficients along offset. Different averagings can be used: either the coefficients themselves, or their numerators and denominators separately, or even median filtering (Woodward and Dellinger, 1984). But, when applied to the LSL algorithm, these procedures simply don't work, because the time-update recursions cannot be averaged. The reflection coefficients quickly become larger than 1, and the algorithm "explodes".

However, this kind of multichannel processing can be applied to my Burg-type algorithm, as in the exact Burg (non-adaptive) algorithm, by averaging the numerators and denominators of the reflection coefficients.

CONCLUSION

I have presented here ways in which two classical spectral-analysis adaptive algorithms can be modified so that they are suitable to any prediction problem. The first one, the LSL algorithm, solves an exact least-squares problem on windows of increasing size; its residuals don't depend on future data. The other one, a Burg-type algorithm, uses a tapering of the residuals to make the process adaptive, but seems less rigorous mathematically. However, the better results are obtained by the Burg-type algorithm, which is better stabilized than the LSL algorithm.

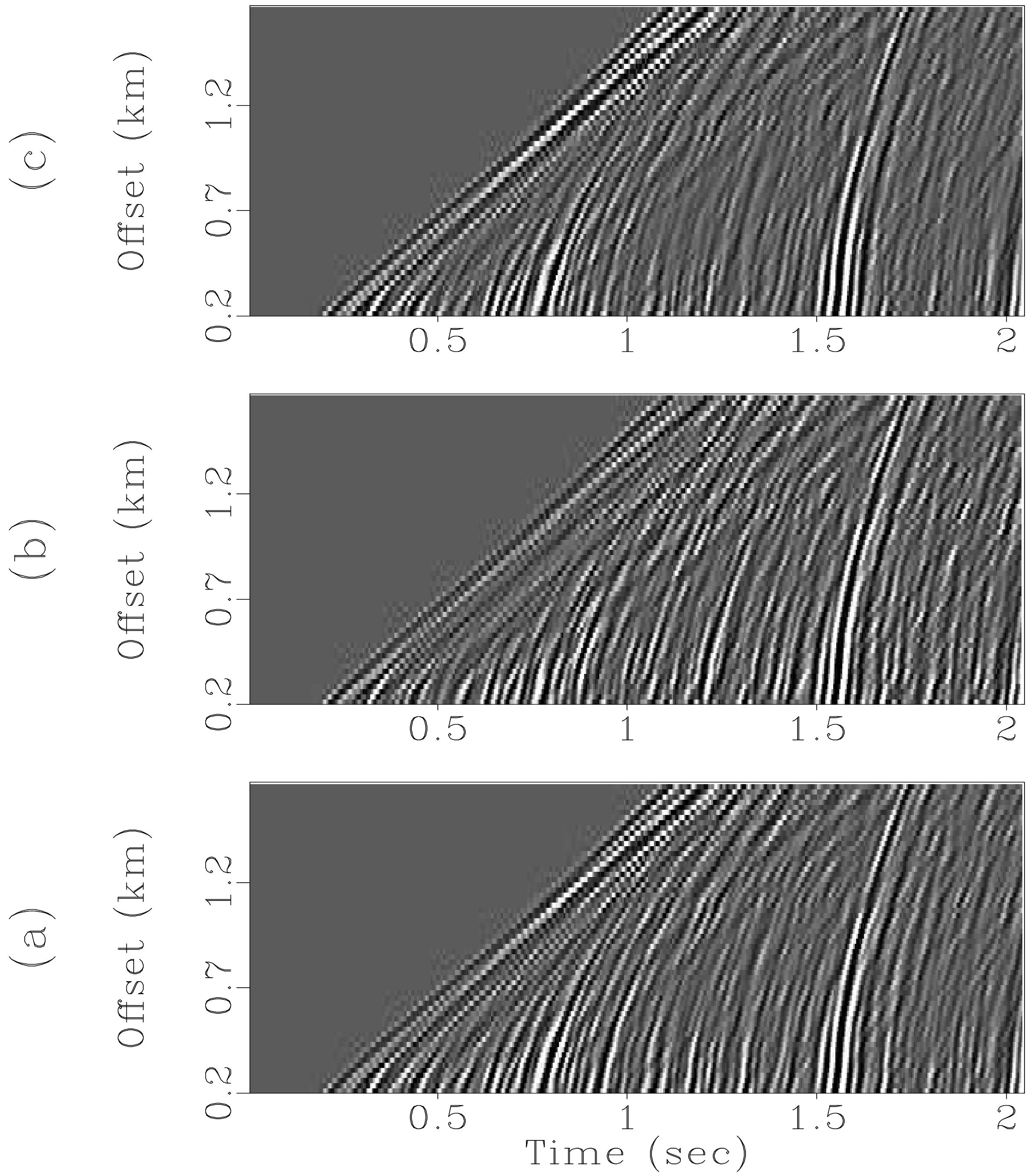

Figure 3: Removal processing, output in the time-offset domain. (a) Block method. (b) LSL algorithm. (c) Adaptive Burg-type algorithm. The outputs of Figure 2 are transformed back to the time-offset domain. The Burg-type method is more efficient in the elimination process. In particular, after velocity analysis, the undetermined event at 1.2 sec on (b) appeared to be a water-bottom pegleg of the primary at 0.7 sec. gilles1-H27result [NR]

REFERENCES

Burg, J.P., 1975, Maximum entropy spectral analysis: SEP **6**.

Claerbout, J.F., 1977, Weighted Burg-Levinson recursion with noise: SEP **11**, 167-172.

Friedlander, B., 1982, Lattice filters for adaptive processing: Proceedings of the IEEE, vol. **70-8**, 829-867.

Hale, D., 1981, Q and adaptive prediction error filters: SEP **28**, 209-232.

Lee, D., Morf, M., and Friedlander, B., 1981, Recursive least squares ladder estimation algorithms: IEEE, vol. ASSP **29-3**, 627-642.

Woodward, M., and Dellinger, J., 1984, Median spectra: SEP **41**, 35-50.

APPENDIX A: GENERAL LSL ALGORITHM

I present here the LSL algorithm for adaptive prediction of joint-process recursion, where x_T is predicted from y_T. Giving a non-0 a-priori value σ^2 to $R^{\varepsilon}_{0,-1}$ and $R^{r}_{0,-1}$ avoids divisions by 0. The algorithm is:

$$\texttt{Input parameters :}$$
$$\left[\begin{array}{lcl} n & = & \texttt{order of the prediction filter} \\ x_T,\ y_T & = & \texttt{input data sequence until time } T \\ \lambda & = & \texttt{exponential weighting factor} \\ \sigma^2 & = & \texttt{a}-\texttt{priori variance (for stabilization)} \end{array}\right.$$

$$\texttt{Variables :}$$
$$\left[\begin{array}{l} \varepsilon_p, r_p:\ \texttt{forward}-\texttt{backward residuals of order } p \texttt{ of } y_T \\ \varepsilon^x_p:\ \texttt{forward residuals of order } p \texttt{ in the prediction of } x_T \texttt{ from } y_T \\ \Delta_{p,t}:\ \texttt{correlation between } \varepsilon_p \texttt{ and } r_p \\ \Delta^x_{p,t}:\ \texttt{correlation between } \varepsilon^x_p \texttt{ and } r_p \\ R^{\varepsilon}_{p,t}, R^{r}_{p,t}:\ \texttt{covariances of } \varepsilon_p \texttt{ and } r_p \\ K^{\varepsilon}_{p,t}, K^{r}_{p,t}:\ \texttt{forward}-\texttt{backward reflection coefficients of } y_T \\ K^{x}_{p,t}:\ \texttt{forward reflection coefficient of } x_T \\ \gamma_{p,t}:\ \texttt{likelihood variable } \cos^2\theta_{1,p+1,t+1} \end{array}\right.$$

$$\texttt{Initializations :}$$
$$\left[\begin{array}{l} R^{\varepsilon}_{0,-1} = R^{r}_{0,-1} = \sigma^2 \\ \quad \texttt{For } t = 0,\cdots,T: \\ \qquad \varepsilon_0(t) = r_0(t) = y(t)\ ;\ \varepsilon^x_0(t) = x(t)\ ;\ \gamma_{-1,t} = 1.\ ;\ R^{\varepsilon}_{0,t} = R^{r}_{0,t} = \lambda R^{\varepsilon}_{0,t-1} + y^2(t) \end{array}\right.$$

$$\texttt{Time updating :}$$
$$\left[\begin{array}{l} \texttt{For } t = 0,\cdots,T: \\ \qquad \left[\begin{array}{l} \texttt{For } p = 0,\cdots,min(n,t)-1: \\ \quad \Delta_{p+1,t} = \lambda\Delta_{p+1,t-1} + \varepsilon_p(t).r_p(t-1)/\gamma_{p-1,t-1} \\ \quad \Delta^x_{p+1,t} = \lambda\Delta^x_{p+1,t-1} + \varepsilon^x_p(t).r_p(t-1)/\gamma_{p-1,t-1} \\ \quad \gamma_{p,t} = \gamma_{p-1,t} - r^2_p(t)/R^r_{p,t} \\ \quad K^r_{p+1,t} = \Delta_{p+1,t}/R^r_{p,t-1} \\ \quad \varepsilon_{p+1}(t) = \varepsilon_p(t) - K^r_{p+1,t}.r_p(t-1) \\ \quad R^{\varepsilon}_{p+1,t} = R^{\varepsilon}_{p,t} - K^r_{p+1,t}.\Delta_{p+1,t} \\ \quad K^{\varepsilon}_{p+1,t} = \Delta_{p+1,t}/R^{\varepsilon}_{p,t} \\ \quad r_{p+1}(t) = r_p(t-1) - K^{\varepsilon}_{p+1,t}.\varepsilon_p(t) \\ \quad R^r_{p+1,t} = R^r_{p,t-1} - \Delta_{p+1,t}.K^{\varepsilon}_{p+1,t} \\ \quad K^x_{p+1,t} = \Delta^x_{p+1,t}/R^r_{p,t-1} \\ \quad \varepsilon^x_{p+1}(t) = \varepsilon^x_p(t) - K^x_{p+1,t}.r_p(t-1) \end{array}\right. \end{array}\right.$$

APPENDIX B: GENERAL BURG-TYPE ADAPTIVE ALGORITHM

I present here the Burg-type algorithm for adaptive prediction of joint-process recursion, where x_T is predicted from y_T. Giving a non-0 a-priori value σ^2 to the denominators avoids divisions by 0. The algorithm is:

$$\texttt{Input parameters}: \ \texttt{as in the LSL algorithm}$$

$$
\begin{array}{l}
\texttt{Variables}: \\
\left[\begin{array}{l}
\varepsilon, r: \ \texttt{forward}-\texttt{backward residuals of } y_T \\
\varepsilon^x: \ \texttt{forward residuals in the prediction of } x_T \texttt{ from } y_T \\
N^-, N^+, N^{x-}, N^{x+}: \ \texttt{numerators in the reflection coefficients} \\
D^{\varepsilon-}, D^{\varepsilon+}, D^{r-}, D^{r+}: \ \texttt{denominators in the same coefficients} \\
K^\varepsilon, K^r: \ \texttt{forward}-\texttt{backward reflection coefficients of } y_T \\
K^x: \ \texttt{forward reflection coefficient of } x_T
\end{array}\right.
\end{array}
$$

$$
\begin{array}{l}
\texttt{Initializations}: \\
\left[\ \texttt{For } t = 0, \cdots, T: \ \varepsilon(t) = r(t) = y(t) \ ; \ \varepsilon_x(t) = x(t) \right.
\end{array}
$$

$$
\begin{array}{l}
\texttt{Order updating}: \\
\left[\begin{array}{l}
\texttt{For } p = 2, \cdots, n: \\
\quad N^-(p) = N^{x-}(p) = N^+ = N^{x+} = 0 \\
\quad D^{\varepsilon-}(p) = D^{r-}(p) = D^{\varepsilon+} = D^{r+} = \sigma^2 \\
\quad \left[\begin{array}{l}
\texttt{For } t = p, \cdots, T-1: \\
\quad N^-(t+1) = \lambda.(N^-(t) + \varepsilon(t).r(t-1)) \\
\quad D^{\varepsilon-}(t+1) = \lambda.(D^{\varepsilon-}(t) + \varepsilon^2(t)) \\
\quad D^{r-}(t+1) = \lambda.(D^{r-}(t) + r^2(t-1)) \\
\quad N^{x-}(t+1) = \lambda.(N^{x-}(t) + \varepsilon^x(t).r(t-1))
\end{array}\right. \\
\\
\quad \left[\begin{array}{l}
\texttt{For } t = T, \cdots, p: \\
\quad N^+ = \lambda.N^+ + \varepsilon(t).r(t-1) \\
\quad D^{\varepsilon+} = \lambda.D^{\varepsilon+} + \varepsilon^2(t) \\
\quad D^{r+} = \lambda.D^{r+} + r^2(t-1) \\
\quad N^{x-} = \lambda.N^{x-} + \varepsilon^x(t).r(t-1) \\
\quad K^\varepsilon = (N^- + N^+)/(D^{\varepsilon-} + D^{\varepsilon+}) \\
\quad K^r = (N^- + N^+)/(D^{r-} + D^{r+}) \\
\quad K^x = (N^- + N^+)/(D^{x-} + D^{x+}) \\
\quad \varepsilon old = \varepsilon(t) \\
\quad \varepsilon(t) = \varepsilon old - K^r.r(t-1) \\
\quad \varepsilon^x(t) = \varepsilon^x(t) - K^x.r(t-1) \\
\quad r(t) = r(t-1) - K^\varepsilon.\varepsilon old
\end{array}\right.
\end{array}\right.
\end{array}
$$