Adaptive prediction error filters

Rustam Akhmadiev

ABSTRACT

The theory and the applications of prediction error filters are well known. Most of the time, however, they are built to be stationary (not varying in space and time) and therefore, they carry some global statistical information of the signals. Hence, these filters cannot be expected to be optimal in a nonstationary environment. Here I address the problem of designing a nonstationary prediction error filter (PEF) using a gradient adaptive lattice and recursive least-squares filters. Their one- and two-dimensional applications (deconvolution) to nonstationary signals show better whitening properties compared to conventional stationary PEFs.

INTRODUCTION

Signals recorded on seismic records are not stationary. The frequency content of the seismic waves change as they propagate through the subsurface because of the anelastic attenuation and scattering. At the same time recorded data is nonstationary in space because different events have different slopes that change with time as well. That is why it is natural to try to find filters that adapt to the local characteristics of the signals.

Prediction error filters arise from the theory of linear prediction, which deals with the problem of estimating the future samples of the signal using a linear combination of its previous samples weighted by sought-for filter coefficients. The error of this prediction turns out to be white (not correlated), while the filter has the inverse spectrum of the signal (that allows estimation of the original signal). Consequently, one of the most common applications of these filters is deconvolution – the process that tries to widen the spectrum by removing the convolutional effects of the bandlimited signal. Having the information of the original signals, these filters can also be used for multidimensional interpolation.

There are different ways of finding a PEF, all based on minimizing the norm of the prediction error. Probably the very first successful way of solving this problem was based on an efficient way of solving normal equations (finding the inverse of a Toeplitz matrix) using the Levinson-Durbin algorithm (Claerbout, 1985). This problem can also be approached using optimization theory tools such as methods of steepest descent, conjugate gradient, etc (Claerbout, 2014). While being very efficient and stable, the gradient-based methods, however, require significant effort to be implemented in a nonstationary environment. On the other hand, it turns out that

the methods based on the original Levinson-Durbin recursion are very fast, efficient and easily adjusted to account for nonstationarity.

One of these filters considered herein is called the lattice predictor filter and arises from the Levinson-Durbin algorithm. It is well studied in the electrical engineering society, it can be easily extended to be adaptive and it is known to be the most efficient structure to give orthogonal (not correlated) forward and backward prediction errors.

Another popular adaptive algorithm is called recursive least squares (RLS) that essentially is a way of adaptively finding the inverse correlation matrix (coming from the normal equations) as the data streams through the filter.

LATTICE FILTER

The lattice filter can be understood from the point of view of the Levinson-Durbin recursion. The filter coefficients are updated in a following way:

$$\mathbf{a}_m = \begin{bmatrix} \mathbf{a}_{m-1} \\ 0 \end{bmatrix} + K_m \begin{bmatrix} 0 \\ \mathbf{a}^B_{m-1} \end{bmatrix}, \tag{1}$$

where $\mathbf{a}_{m-1}$ is the prediction-error filter of order $(m-1)$ and size m on the previous iteration, $\mathbf{a}^B_{m-1}$ has the same coefficients in the reverse order, K_m are reflection coefficients and $\mathbf{a}_m$ is the updated prediction-error filter of order (m) and size $m+1$. Let us consider the action of the PEF of order m to the input $\mathbf{u}_{m+1} = [u(n), u(n-1), \ldots, u(n-m)]$ of size $m+1$ and with corresponding lags equal to $[0, 1, \ldots, m]$. This produces a forward prediction error of order m:

$$f_m(n) = \mathbf{a}^T_m \mathbf{u}_{m+1}(n)$$

Looking at the terms in the Equation 1 separately:

$$\begin{bmatrix} \mathbf{a}^T_{m-1} & 0 \end{bmatrix} \mathbf{u}_{m+1}(n) = \begin{bmatrix} \mathbf{a}^T_{m-1} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_m(n) \\ u(n-m) \end{bmatrix} = \mathbf{a}^T_{m-1} \mathbf{u}_m(n) = f_{m-1}(n),$$

$$\begin{bmatrix} 0 & \mathbf{a}^{BT}_{m-1} \end{bmatrix} \mathbf{u}_{m+1}(n) = \begin{bmatrix} 0 & \mathbf{a}^{BT}_{m-1} \end{bmatrix} \begin{bmatrix} u(n) \\ \mathbf{u}_m(n-1) \end{bmatrix} = \mathbf{a}^{BT}_{m-1} \mathbf{u}_m(n-1) = b_{m-1}(n-1).$$

Therefore, to estimate the forward prediction error $f_m(n)$ (where m represents the filter length $m+1$ at time n), we need the forward prediction error $f_{m-1}(n)$ at time n corresponding to the lower filter of order $m-1$ and backward prediction error $b_{m-1}(n-1)$ at the previous sample $n-1$:

$$\begin{aligned} f_m(n) &= f_{m-1}(n) + K_m b_{m-1}(n-1), \\ b_m(n) &= b_{m-1}(n-1) + K_m f_{m-1}(n) \end{aligned} \tag{2}$$

For this procedure and the resulting filter to be stable (and minimum-phase) the prediction-error power should be decreasing or at least stay the same. Update equations for the prediction-error of the Levinson recursion (Yilmaz, 2001) show that this requires the reflection coefficients $|K_i| \leq 1$ at every iteration $i = 1, \ldots, m$.

To find the coefficients K_m, we should find the minimum of objective function:

$$E_m(n) = ||\mathbf{f}_m(n)||^2 + ||\mathbf{b}_m(n)||^2,$$

where $\mathbf{f}_m(n)$ and $\mathbf{b}_m(n)$ are now the vectors containing the forward and backward prediction errors of m-order accumulated up to the n-th sample.

$$\begin{aligned} E_m(n) &= ||\mathbf{f}_m(n)||^2 + ||\mathbf{b}_m(n)||^2 = \\ &= ||\mathbf{f}_{m-1}(n) + K_m\mathbf{b}_{m-1}(n-1)||^2 + ||\mathbf{b}_{m-1}(n-1) + K_m\mathbf{f}_{m-1}(n)||^2 = \\ &= (||\mathbf{f}_{m-1}(n)||^2 + ||\mathbf{b}_{m-1}(n)||^2)(1 + K_m^2) + 4K_m\mathbf{b}_{m-1}^T(n-1)\mathbf{f}_{m-1}(n). \end{aligned}$$

Taking the derivative of the previous expression with respect to K_m:

$$\frac{\partial E_m}{\partial K_m} = 2K_m(||\mathbf{f}_{m-1}(n)||^2 + ||\mathbf{b}_{m-1}(n)||^2) + 4\mathbf{b}_{m-1}^T(n-1)\mathbf{f}_{m-1}(n).$$

Therefore, the optimal K_m minimizing the power of forward and backward errors can be chosen by equating the derivative to zero:

$$K_m = -\frac{2\mathbf{b}_{m-1}^T(n-1)\mathbf{f}_{m-1}(n)}{||\mathbf{f}_{m-1}(n)||^2 + ||\mathbf{b}_{m-1}(n)||^2}.$$

What this says is that the reflection coefficients (controlling the PEF and the prediction error) are computed based on all the previous samples. The idea for making this process adaptive is to introduce a "forgetting" parameter $0 < \beta < 1$ for the denominator estimate at a current sample n to become (taking single-pole average):

$$\begin{aligned} E_{m-1}(n) &= ||\mathbf{f}_{m-1}(n)||^2 + ||\mathbf{b}_{m-1}(n)||^2 \\ &= \beta E_{m-1}(n-1) + (1-\beta)(f_{m-1}^2(n) + b_{m-1}^2(n)). \end{aligned} \tag{3}$$

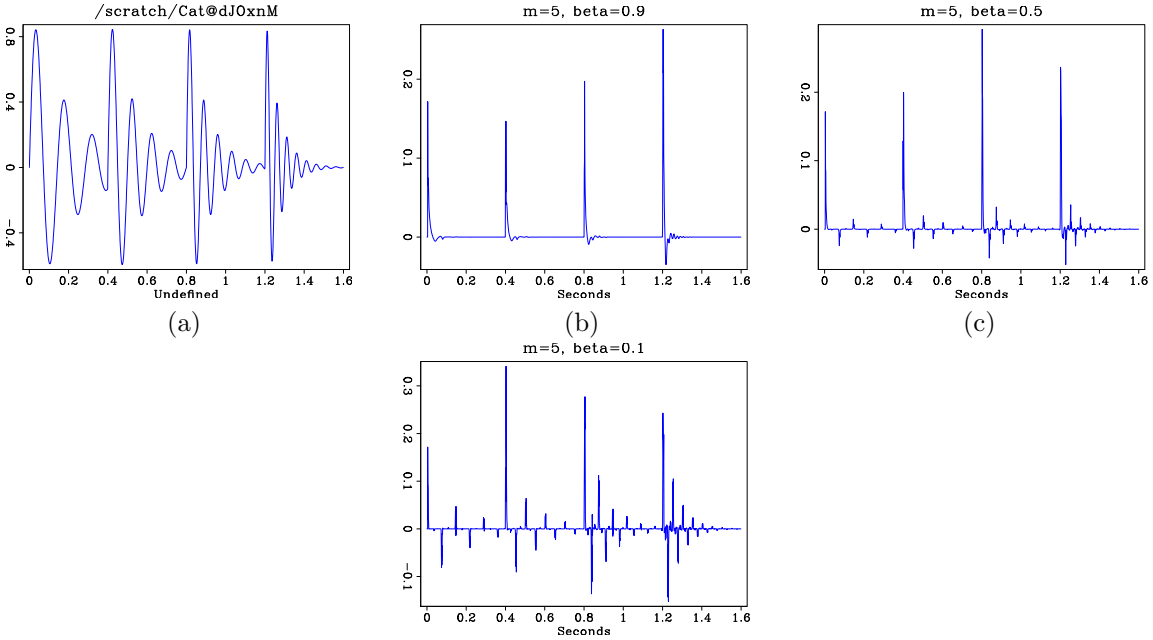
Treating the numerator in the same manner and allowing reflection coefficients to depend on the sample number $K_m(n)$ it is possible to obtain the following recursive formula (Haykin, 2002):

$$K_m(n) = K_m(n-1) - \frac{1-\beta}{E_{m-1}(n)}(f_{m-1}(n)b_m(n) + b_{m-1}(n-1)f_m(n)), \tag{4}$$

which is similar to the update equations of the normalized LMS algorithm (Paleologu et al., 2008). This allows the algorithm to respond accordingly depending on the powers of forward and backward errors. If the errors are small ($E(n)$ is small) then

the step-size $(1-\beta)/E(n)$ is large and the filter adapts rapidly and if the errors are large, the step-size is small and the filter doesn't respond to variations as quickly.

To imitate nonstationarity, I created a simple single trace with four damped sinusoids with four distinct frequencies increasing with time (Figure 5a). We can see that after filter is applied, the signal has indeed been compressed, the amplitudes, however, are not really reliable. The effect of forgetting parameter β is also obvious – the smaller its value, the more adaptive the filter is, which results in overfitting (Figure 1d).



(a) (b) (c) (d)

Figure 1: Lattice filter applied to single nonstationary trace (with a filter size $m = 5$, β–forgetting parameter): (a) – original trace, (b) – $\beta = 0.9$, (c) – $\beta = 0.5$, (d) – $\beta = 0.1$. [**ER**]

I then applied this filter to the field stacked data on Figure 2. Again, after filtering, we can see increased resolution. However, we have to be careful again with the forgetting parameter so as to not create "fake" reflections.

Multidimensional lattice filter

Lattice filters are implicitly solving the problem of finding the inverse of the autocorrelation matrix of the signal involved in the normal equations of linear prediction. Therefore, as PEFs they are carrying information of the signal spectra. Multidimensional filters are similar and probably even more important for the real applications.

Extending lattice filters to two dimensions is not a trivial task. There have been several successful attempts of solving this problem (e.g. (Parker and Kayran, 1984)).

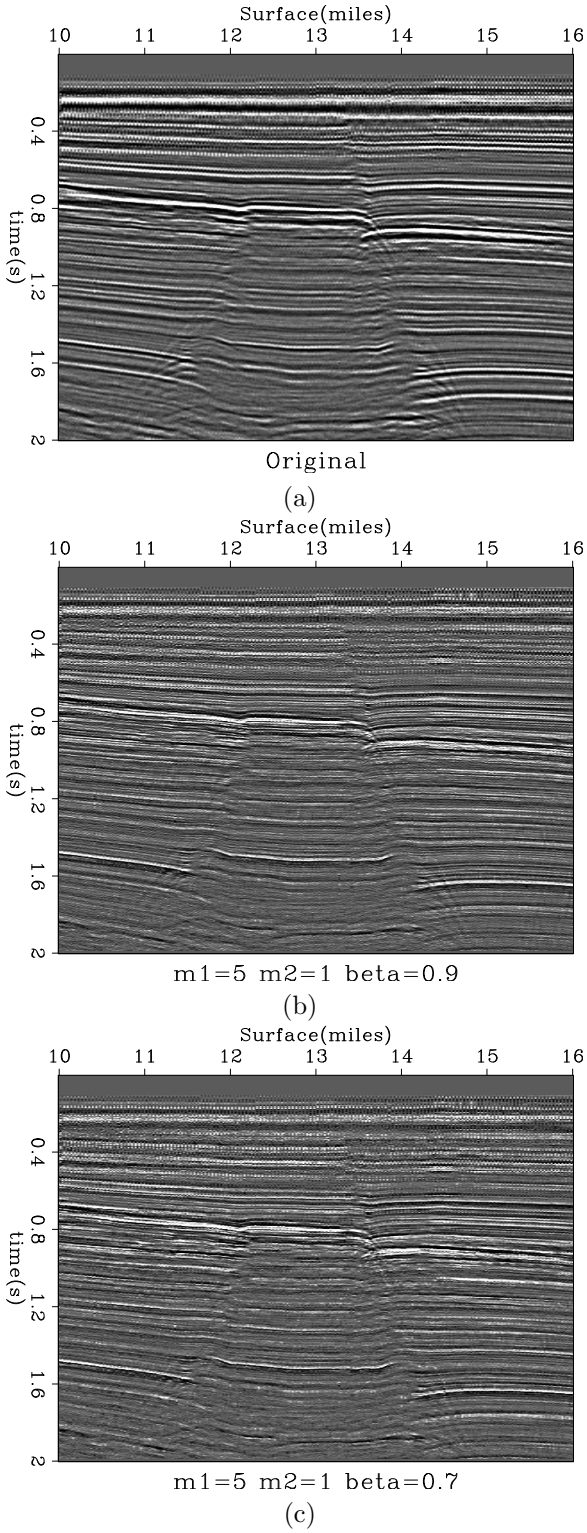
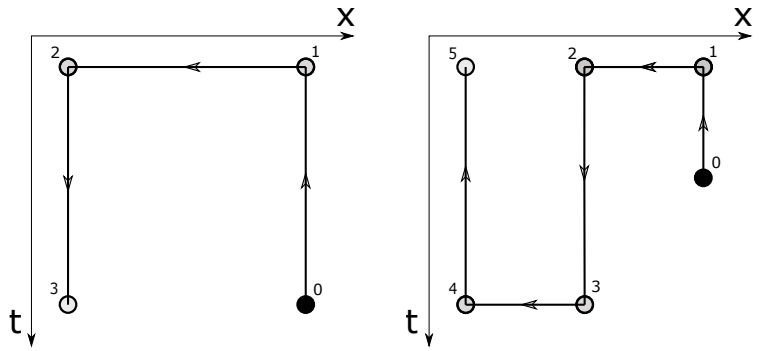


Figure 2: Lattice filter applied to raw stack (with a filter size $m = 5$, β–forgetting parameter): (a) – original stack, (b) – $\beta = 0.9$, (c) – $\beta = 0.7$. [**ER**]

If in the one-dimensional case there are two fields involved (forward and backward prediction errors), in 2D there are even more fields that need to be updated (4 in case of a 2x2 quarter-plane filter). Naturally, the reflection coefficients now become vectors depending on the filter size. Finding optimal values for them in this case involves a matrix inversion at every lattice stage m and at every sample n. However, it was shown in (Kayran, 1996) that the problem of finding multidimensional lattice filter may be solved using one-dimensional lattice approach.

The first step is to locally order the 2D signal into a 1D array (Figure 3). In this case, ordering corresponding to an assymetric plane was used. After this, the first iteration of the algorithm is to combine all the neighboring pairs within this 1D vector to give the first estimate of the forward and backward errors at all the points. The second iteration combines the pair of points jumping across one sample, the third – jumping across two samples, etc. We iterate until the jump is equal to the whole filter length. To make this filter adaptive, the reflection coefficients are updated by the same Equation 4.



(a) (b)

Figure 3: Ordering of 2D signal into 1D array: (a)–quarter-plane model, (b)–assymetric plane. Evaluating the prediction error at the 0-th sample (shown by black circle) involves previous samples (shown in white circles). Numbers correspond to ordering of the samples involved in calculations. [**NR**]

The result of applying a 2D lattice filter of size 2x3 on the stacked data is shown in Figure 4. Two-dimensional prediction error filters destroy the correlation in two dimensions, which is why the continous reflections are suppressed. However, because the dip of the events is changing in space and time, crossing events are harder to predict. By changing the forgetting parameter it is possibe to control the adaptiveness of the filter and significantly remove the correlated (in space and time) events. Making this parameter even smaller than shown in the figure removes most of the events, making the output white noise. Increasing the filter size might also help in predicting more slopes, because it will be capturing more spatial information.

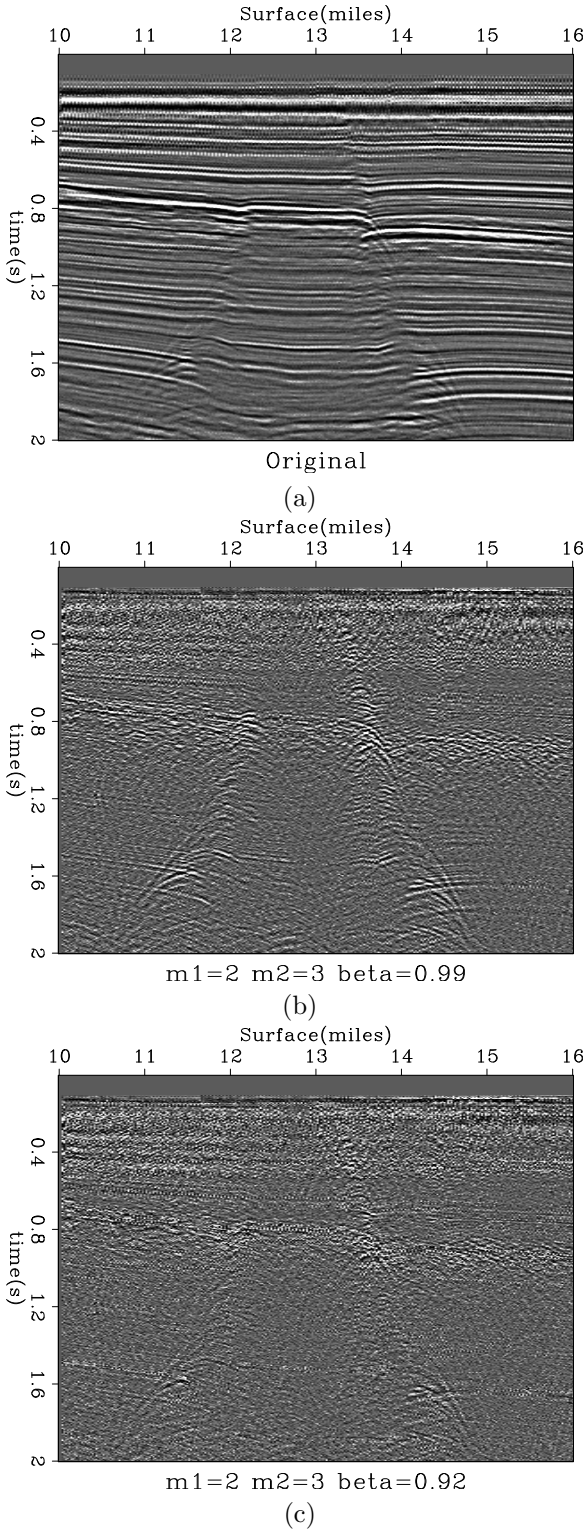


Figure 4: 2D lattice filter applied to raw stack (with filter size $m_1 = 2, m_2 = 3$, β–forgetting parameter): (a) – original stack, (b) – $\beta = 0.99$, (c) – $\beta = 0.92$. [**ER**]

RECURSIVE LEAST-SQUARES (RLS) FILTER

The recursive least-squares algorithm is a general extension of the least-squares method that allows computation of adaptive filters as the data streams through the filter. It is known to have better convergence properties (towards optimal Wiener solution) than the least-mean square (LMS) algorithm due to the intrinsic use of the inverse correlation matrices.

The first step to allow RLS filter adaptation is to introduce a forgetting parameter $0 < \lambda \leq 1$ in the error-norm estimate at a given sample n:

$$\begin{aligned} E(n) &= \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 \\ &= \sum_{i=1}^{n} \lambda^{n-i} |d(i) - y(i)|^2 \\ &= \sum_{i=1}^{n} \lambda^{n-i} |d(i) - \mathbf{w}^T(n)\mathbf{u}(i)|^2, \end{aligned} \tag{5}$$

where $d(i)$ is the desired response at i-th sample, $y(i) = \mathbf{w}^T(n)\mathbf{u}(i)$ is the output of adaptive filter $\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{m-1}(n)]^T$ estimated at sample n acting on the past m input samples $\mathbf{u}(i) = [u(i), u(i-1), \ldots, u(i-m+1)]^T$.

Because of the ill-posed nature of the problem, a regularization term is included in the cost function as follows

$$E(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 + \delta\lambda^n ||\mathbf{w}(n)||^2.$$

Finding the zeroes of derivative of this cost function with respect to the filter coefficients $\mathbf{w}$ leads to normal equations with an autocorrelation matrix $\mathbf{R}(n)$, and cross-correlation $\mathbf{x}(n)$ of the desired output $d(i)$ with the input vector $\mathbf{u}(i)$:

$$\begin{aligned} &\mathbf{R}(n)\mathbf{w}(n) = \mathbf{x}(n), \\ &\mathbf{R}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}(i)\mathbf{u}^T(i) + \delta\lambda^n \mathbf{I}, \\ &\mathbf{x}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{u}(i)d(i). \end{aligned}$$

It is straightforward to see now that adding the regularization term has essentially the effect of adding white noise to the data and makes the correlation matrix diagonally dominant to assure its stable inverse. It is also easy to note the following recursion relations (Haykin, 2002):

$$\begin{aligned} \mathbf{R}(n) &= \lambda\mathbf{R}(n-1) + \mathbf{u}(n)\mathbf{u}^T(n), \\ \mathbf{x}(n) &= \lambda\mathbf{x}(n-1)\mathbf{u}(n)d(n). \end{aligned}$$

Computing the optimal filter weights requires the inverse correlation matrix $\mathbf{R}^{-1}(n) = \mathbf{P}(n)$. An efficient way of finding it is using the previously shown recursion and the Woodbury matrix inversion lemma. The resulting formulas are:

$$\begin{aligned}
\mathbf{P}(n) &= \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^T(n)\mathbf{P}(n-1), \\
\mathbf{k}(n) &= \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^T(n)\mathbf{P}(n-1)\mathbf{u}(n)} = \mathbf{P}(n)\mathbf{u}(n), \\
\mathbf{w}(n) &= \mathbf{w}(n-1) + \mathbf{k}(n)[d(n) - \mathbf{u}^T(n)\mathbf{w}(n-1)] = \mathbf{w}(n-1) + \mathbf{k}(n)\xi(n).
\end{aligned} \tag{6}$$

Here, $\xi(n)$ is the a priori estimation error – the error at a current sample n estimated using the old filter weights from the previous sample $n-1$; $\mathbf{k}(n)$ is the gain vector that is the input $\mathbf{u}(n)$ transformed by the inverse correlation matrix $\mathbf{P}(n)$. We see that the filter weights are updated by adding in a priori prediction error scaled by $\mathbf{k}(n)$.

The equations 6 constitute the RLS algorithm. We start off from the initial filter $\mathbf{w}(0) = \mathbf{0}$ and initial correlation matrix $\mathbf{R}(0) = \delta\mathbf{I}$. Then, after choosing the forgetting parameter λ ,the algorithm proceeds to giving the prediction errors.

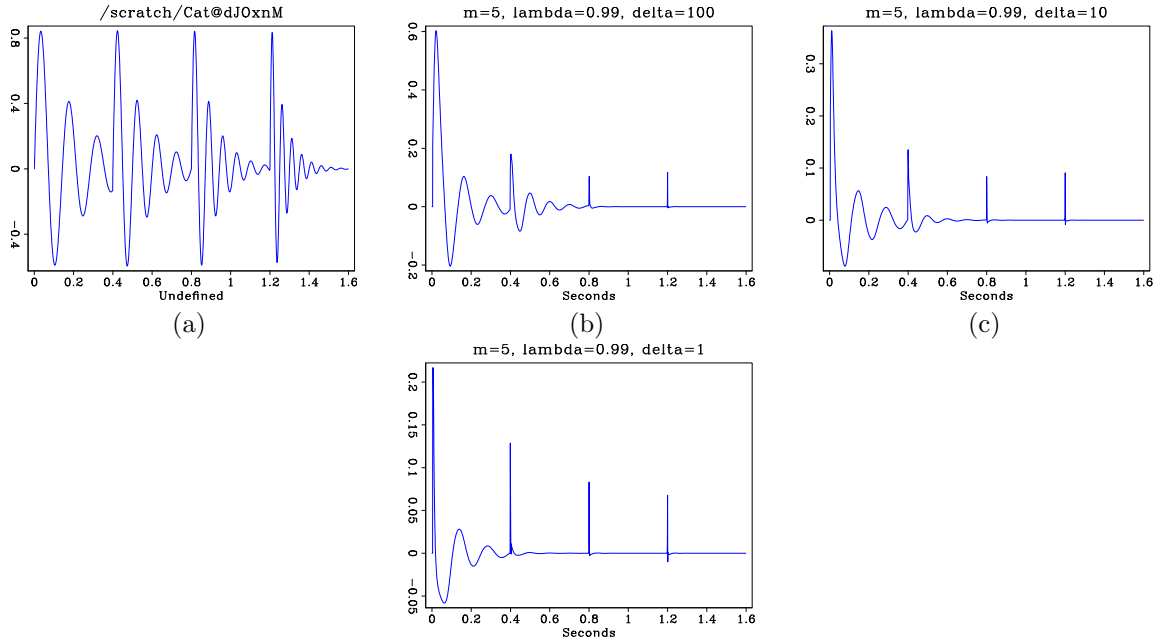


Figure 5: RLS filter applied to single nonstationary trace (with a filter size $m = 5$, forgetting parameter $\lambda = 0.9$ and initialization constant δ: (a) – original trace, (b) – $\delta = 100$, (c) – $\delta = 10$, (d) – $\delta = 1$. [**ER**]

We applied the RLS filter to a single nonstationary trace (Figure 5). Here, we observe different results by choosing different initialization constants δ. Apparently, decreasing it leads to faster convergence, however in the high signal-to-noise environment this might lead to instabilities.

Applying the filter to the raw stacked data (Figure 6) increased resolution throughout the whole section since the filter was changing along the time axis and adapting to the changes in the signal.

Adding filter coefficients in the spatial direction leads to suppressing spatially correlated events (mostly reflections). Ideally the output of the PEF should be uncorrelated white noise. This can be achieved by modifying the forgetting parameter and making the filter adapt faster.

CONCLUSION

Prediction error filters are powerful tools for capturing statistical information about the signals. I have implemented nonstationary versions able to adapt to local (multidimensional) changes of the waves. Gradient adaptive lattice and recursive least-squares prediction-error filters have shown to be efficient and fast. However, their stability, parametrization and whitening properties are yet to be investigated.

ACKNOWLEDGEMENTS

I would like to thank Jon Claerbout for bringing the topic of adaptive PEFs into the SEP group and his continuous interest in this study.

REFERENCES

Claerbout, J. F., 1985, Fundamentals of geophysical data processing: Blackwell Science Inc.

———, 2014, Geophysical image estimation by example.

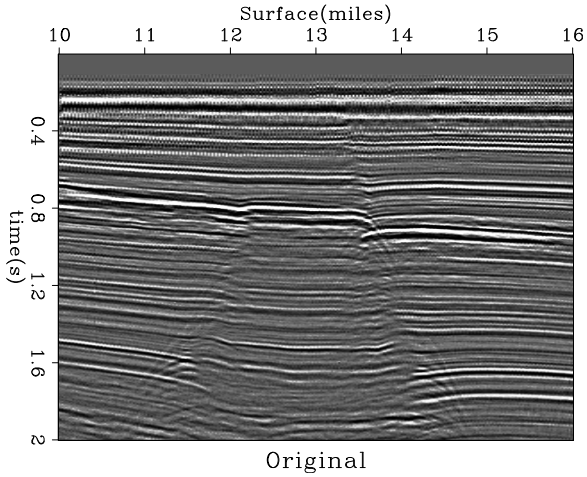
Haykin, S. S., 2002, Adaptive filter theory: Prentice Hall.

Kayran, A., 1996, Two-dimensional orthogonal lattice structures for autoregressive modeling of random fields: IEEE transactions on signal processing, **44**, 963–978.

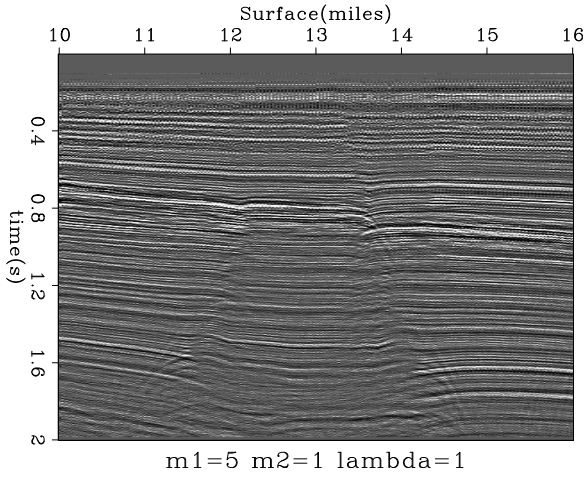
Paleologu, C., S. Ciochina, A. A. Enescu, and C. Vladeanu, 2008, Gradient adaptive lattice algorithm suitable for fixed point implementation: 2008 The Third International Conference on Digital Telecommunications, 41–46.

Parker, S., and A. Kayran, 1984, Lattice parameter autoregressive modeling of two-dimensional fields-part i: The quarter-plane case: IEEE transactions on acoustics, speech, and signal processing, **ASSP-32**, 872–885.

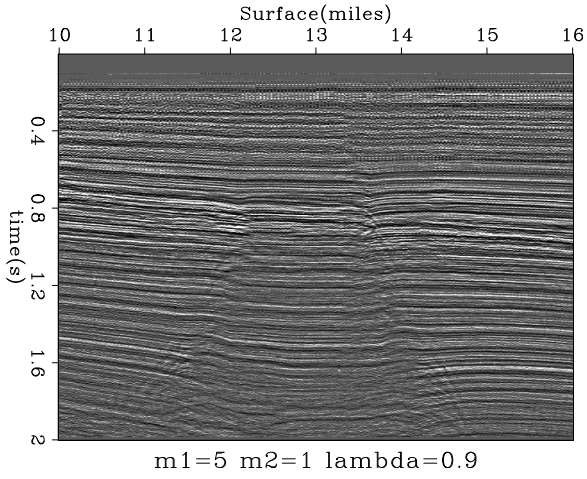
Yilmaz, ʹO.., 2001, Seismic data analysis: Society of Exploration Geophysicists.



(a)



(b)



(c)

Figure 6: 1D RLS filter applied to raw stack (with filter size $m = 5$, forgetting parameter λ and initialization constant $\delta = 10$: (a) – original stack, (b) – $\lambda = 1$, (c) – $\lambda = 0.9$). [**ER**]

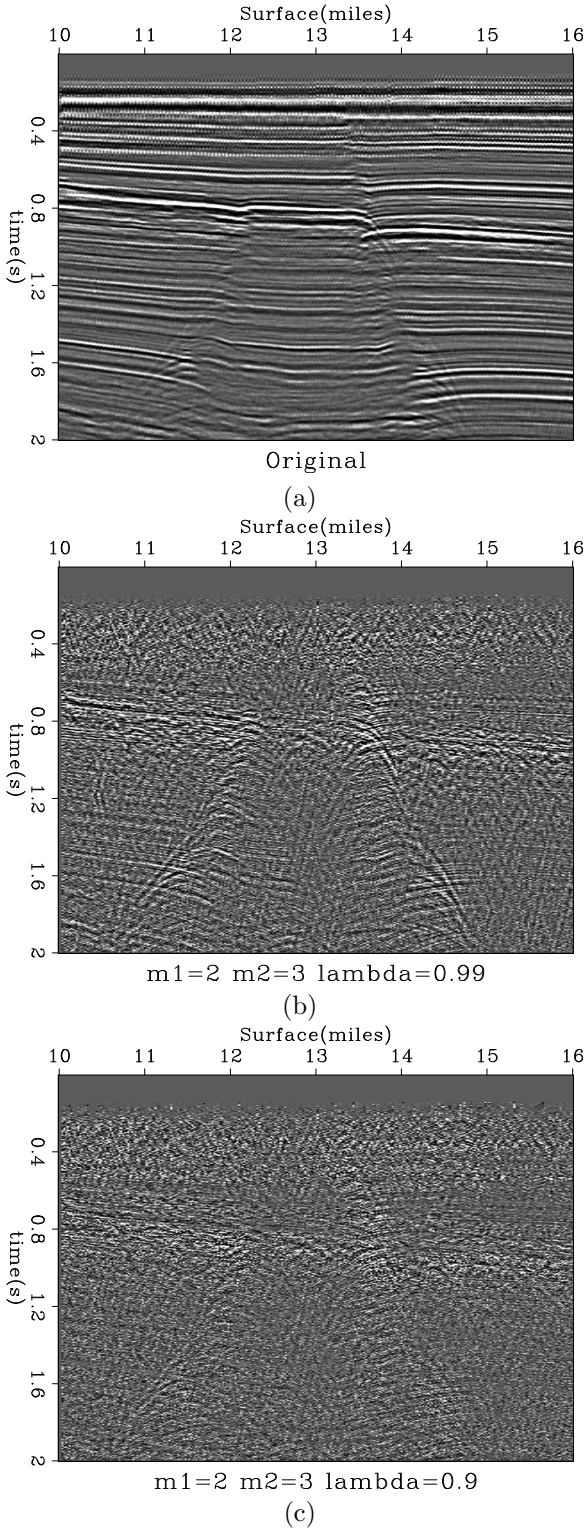


Figure 7: 2D RLS filter applied to raw stack (with filter size $m_1 = 2, m_2 = 3$, forgetting parameter λ and initialization constant $\delta = 10$: (a)– original stack, (b) – $\lambda = 0.99$, (c) – $\lambda = 0.9$). [**ER**]