TV-PEF design : Algorithmic aspects, a fast update scheme and 2D path dependence

Rahul Sarkar

ABSTRACT

Prediction error filters (PEFs) have found applications in several important areas of geophysics, including noise removal and interpolation of seismic data. Non-stationary time variant PEFs or TV-PEFs are special in this family of filters because they can model non-stationary processes which allow us to more accurately capture most seismic data. However, computing and applying the TV-PEFs using the usual method of least squares, becomes prohibitively expensive for large data sets. This paper investigates a recent computationally efficient streaming formulation designed to overcome this difficulty. I show with 1D examples, that the TV-PEFs obtained using streaming are capable of producing results comparable or better than other existing algorithms. I generalize the streaming result to an arbitrary path based update scheme for the TV-PEF coefficients in 2D. It is also shown with numerical examples that the results depend on the path traversed through the data. The performance of the streaming algorithm when applied to the 2D missing data interpolation problem is also considered.

INTRODUCTION

Stationary and non-stationary PEFs have become ubiquitous in autoregressive model fitting of multidimensional data. Recorded surface seismic data is an example of a multidimensional data set where each recorded seismic trace represents a time series for the corresponding pair of source and receiver coordinates. Stationary models, even with their inherent limitations, have been used quite successfully in several applications such as removal of noise, sharpening the spectrum of recorded seismic data, missing data interpolation and more. A nice compilation of such interesting results from the world of geophysics is contained in the book *Geophysical Image Estimation by Example (GIEE)*, Chapter 7 (Claerbout, 2014). *"Seismic Data Analysis"* by Yilmaz (2001) also contains a nice review of prediction filter estimation for geophysical data.

Seismic data is highly non-stationary in both time and space. This is because as the seismic wave travels through the earth before being recorded at the surface, it undergoes changes in its spectral characteristics due to frequency dependent effects such as absorption and dispersion. The spatially varying reflectivity of the earth also introduces additional features into the recorded seismic data, that varies from trace to trace. In order to account for these effects, a non-stationary autoregressive model would fit the changing spectral characteristics of the data better than a stationary model. Crawley (2001) demonstrates an interesting application of TV-PEFs in removing data aliasing without the need for low-pass filtering, achieved by interpolating traces in between with TV-PEFs. Curry (2008) and Curry (2005) provide other interesting applications such as interpolating near offsets using multiples and multidimensional data interpolation on frequency slices. A standard approach to estimate non-stationary TV-PEFs for a data set involves dividing the data into small patches and then fitting a stationary autoregression model to each patch. However, if the characteristics of the data are changing too quickly, one needs to use smaller patch sizes. Successful attempts along these lines have been performed by estimating TV-PEFs in *"micropatches"* followed by the smoothing of filter coefficients for stability (Crawley et al., 1998, 1999). The PEFs thus obtained were termed *"adaptive PEFs"*. Curry (2008) solves the problem of estimating TV-PEFs for irregularly sampled data using a multigrid approach, where data are simultaneously binned into different grids and then TV-PEFs are estimated for all the different binnings simultaneously. However, all of these methods rely on solving a least squares fitting problem, every time the TV-PEF coefficients are generated for a particular patch or location. For very large data sets that are typical in the seismic industry today, this strategy becomes computationally too expensive. In addition, if the number of patches is too large, the cost of storing TV-PEF coefficients becomes comparable to the data itself.

To make the problem computationally feasible, Ruan et al. (2015) recently introduced the idea of evaluating and storing the TV-PEF coefficients on a coarse grid. In that case, one would only need to solve the least squares problem a few times given by the number of locations in the coarse grid. Once the TV-PEFs are computed and stored, one can approximate the filter coefficients at any intermediate location through linear interpolation. This method succeeds in overcoming the computational bottleneck, but is likely to be error-prone if the TV-PEFs are changing too fast, as linear interpolation will no longer be a valid approximation for the filter coefficients at the intermediate locations. Around the same time, Fomel et al. (2016) came up with an alternative formulation of the TV-PEF estimation problem in 1D which has a simple analytical solution. It is shown that the resulting algorithm involves a few vector dot products and is extremely computationally efficient. The filters evaluated this way are termed *"streaming TV-PEFs"*.

This paper explores this new result and some of its consequences. We show that the algorithm can be easily generalized to 2D. One may even generalize it to higher dimensions trivially. I explore the path dependence of the results obtained using the streaming algorithm. The analysis is done on a 2D data set, but the conclusions are true even in higher dimensions.

We will begin with an overview of stationary and non-stationary auto-regressive models and eventually build up to the streaming TV-PEF formulation. Some examples of PEF applications on 1D data sets are used to compare streaming with other stationary and non-stationary autoregressive model estimation algorithms. We then generalize the technique to 2D and analyze its path dependence based on some simple continuous non-intersecting curves. Finally, some 2D interpolation examples based on streaming TV-PEFs are considered.

STATIONARY AUTOREGRESSIVE MODELS

We begin with a basic review of stationary autoregressive models. Let us consider a data vector $\mathbf{d} \in \mathbb{R}^N$. The elements of $\mathbf{d}$ are denoted by d_i , for $i = 1, \ldots, N$. An autoregressive model of order p for the data vector $\mathbf{d}$ is given by a vector of coefficients $\mathbf{a}$ with elements denoted by a_i , for $i = 1, \ldots, p$ such that the following equation holds true:

$$d_t = \sum_{i=1}^{p} a_i d_{t-i} + n_t \quad \forall\, t = 1, \ldots, N \, . \tag{1}$$

The vector $\mathbf{a}$ is also known as the vector of *"prediction error filter"* coefficients. Note that in the above equation (1) for $t < p$ we can have $t-i \leq 0$, so for equation (1) to make sense for all t we define $d_{t-i} = d_1$ if $t-i \leq 0$. The term n_t in the equation is the error term (also known as the *"residual"* or *"prediction error"*). It can be shown that the vector $\mathbf{n} \in \mathbb{R}^N$ with elements given by n_t has a white spectrum, as a result of least squares minimization, which is explained below (also explained in GIEE, page 182, (Claerbout, 2014)).

The system of equations in (1) can be written in matrix form $\mathbf{d} = \mathbf{D^T a} + \mathbf{n}$ as in equation (2), where $\mathbf{D^T}$ is the matrix that pre-multiplies the vector $\mathbf{a}$. The reason for writing the matrix as a *"transpose"* will become clear soon.

$$\begin{bmatrix} d_1 \\ . \\ . \\ . \\ d_r \\ . \\ . \\ . \\ d_N \end{bmatrix} = \begin{bmatrix} d_1 & d_1 & . & d_1 & d_1 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ d_{r-1} & d_{r-2} & . & d_{r-p+1} & d_{r-p} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ d_{N-1} & d_{N-2} & . & d_{N-p+1} & d_{N-p} \end{bmatrix} \begin{bmatrix} a_1 \\ . \\ . \\ . \\ a_p \end{bmatrix} + \begin{bmatrix} n_1 \\ . \\ . \\ . \\ n_r \\ . \\ . \\ . \\ n_N \end{bmatrix} . \tag{2}$$

Equation (2) can be solved easily using *least squares* by minimizing the L_2 norm of the residual $\mathbf{n} = \mathbf{d} - \mathbf{D^T a}$. The solution is formally written as:

$$\mathbf{a}_* = \arg\min_{\mathbf{a}} ||\mathbf{n}||_2^2 = \arg\min_{\mathbf{a}} ||\mathbf{d} - \mathbf{D^T a}||_2^2 = (\mathbf{DD^T})^{-1}\mathbf{Dd} \, . \tag{3}$$

It is often a good idea to regularize the solution (see for e.g Golub et al. (1999)), as the matrix $\mathbf{DD^T}$ can turn out to be ill-conditioned. One regularized solution to the least squares problem is written as $\mathbf{a}_* = (\mathbf{DD^T} + \epsilon^2 \mathbf{I})^{-1}\mathbf{Dd}$, where $\epsilon > 0$ is a small regularization term. In practice, one would solve the linear system $(\mathbf{DD^T} + \epsilon^2\mathbf{I})\mathbf{a}_* = \mathbf{Dd}$ using iterative methods, like *conjugate gradients* (see Hestenes and Stiefel (1952), Golub and Van Loan (2012) and Paige and Saunders (1982) for some good iterative methods) without actually computing the inverse of $(\mathbf{DD^T}+\epsilon^2\mathbf{I})^{-1}$. We can now denote the residual corresponding to the solution $\mathbf{a}_*$ as $\mathbf{n}_*$, i.e, $\mathbf{n}_* = \mathbf{d} - \mathbf{D^T}\mathbf{a}_*$. We also define the quantity $\mathbf{d}_*$, which is the *reconstructed data*, as $\mathbf{d}_* = \mathbf{D^T}\mathbf{a}_*$. In the discussion that follows, we assume that the order p of the autoregression model is either known or has been determined through a suitable method. Akaike (1969, 1974) and Hannan and Quinn (1979) describe some good methods to do this.

NON-STATIONARY AUTOREGRESSIVE MODELS

We now extend the idea of fitting an autoregressive model to non-stationary data sets. Figure 1 shows an example of a non-stationary signal, which is an 1D data set created by extracting a time slice at t = 1 s from a 2D seismic stack.

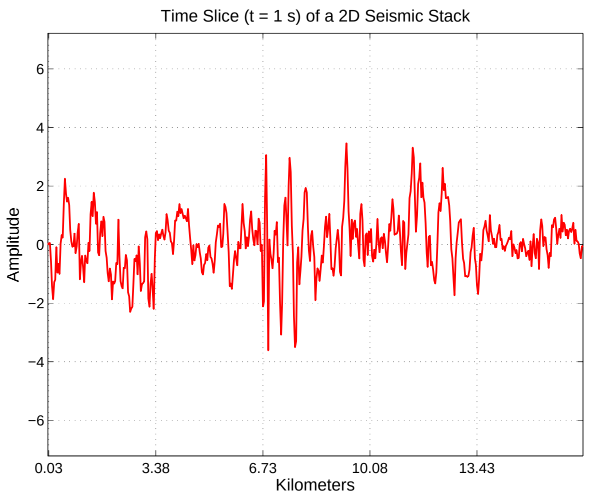


Figure 1: A time slice (t = 1 s) extracted from a 2D seismic stack (courtesy of Western Geophysical, 1974 vintage, Gulf of Mexico) illustrating non-stationary signals. [**CR**]

Local Stationarity

As can be seen from the figure, the characteristics of this data set, change with time. Of course, one could just try to fit a stationary autoregression model to this data and evaluate the results. But, it makes intuitive sense that since the characteristics of the data are changing with time, one should also allow the autoregression model to be a function of time. The first step in trying to do this is to introduce the concept of *"local stationarity"*. Note that in this discussion, we are using *"time"* as a generic independent variable with respect to which the signal is varying; so in the general case, the signal could also be a function of any independent variable like space.

The idea of local stationarity stems from the observation that any non-stationary signal can be thought to be stationary in the neighborhood of each sample. This allows us to take pieces of the signal in small windows and then fit a prediction-error filter model to each piece of the data. For simplicity, we assume that the window size remains constant throughout the data set. In 1D, this process plays out as follows. Suppose we want to fit a time variable prediction error filter (TV-PEF) of order p to the 1D data $\mathbf{d}$ in Figure 1. Let us assume that at every location in the data set, local stationarity holds over b data samples. Thus b defines the window size. For a location i where we wish to find the PEF coefficients, we solve a stationary autoregression problem for the b consecutive data points numbered $i-b+1,\ldots,i$. If we denote the prediction error filter vector at data sample i by $\mathbf{a^{(i)}}$, then we have the linear system of equations given in equation (4).

$$\begin{bmatrix} d_{i-b+1} \\ . \\ . \\ . \\ d_r \\ . \\ . \\ . \\ d_i \end{bmatrix} = \begin{bmatrix} d_{i-b} & d_{i-b-1} & . & d_{i-b-p+2} & d_{i-b+1-p} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ d_{r-1} & d_{r-2} & . & d_{r-p+1} & d_{r-p} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ d_{i-1} & d_{i-2} & . & d_{i-p+1} & d_{i-p} \end{bmatrix} \begin{bmatrix} a_1^{(i)} \\ . \\ . \\ . \\ a_p^{(i)} \end{bmatrix} + \begin{bmatrix} n_{i-b+1} \\ . \\ . \\ . \\ n_r \\ . \\ . \\ . \\ n_i \end{bmatrix} . \tag{4}$$

This equation is similar to equation (2), and hence following similar notation coupled with the fact that we are solving a stationary autoregression problem to determine the filter coefficients at sample i, we write equation (4) symbolically as $\mathbf{d^{(i)}} = \mathbf{D^{(i)T}a^{(i)}} + \mathbf{n^{(i)}}$. The least squares solution is found in a similar manner as follows:

$$\mathbf{a_*^{(i)}} = \arg\min_{\mathbf{a^{(i)}}} ||\mathbf{n^{(i)}}||_2^2 = \arg\min_{\mathbf{a^{(i)}}} ||\mathbf{d^{(i)}} - \ \mathbf{D^{(i)T}a^{(i)}}||_2^2 = (\mathbf{D^{(i)}D^{(i)T}})^{-\mathbf{1}}\mathbf{D^{(i)}d^{(i)}} \ . \tag{5}$$

As in the case of stationary data, it is a good idea to regularize the solution, which is given by $\mathbf{a_*^{(i)}} = (\mathbf{D^{(i)}D^{(i)T}}+\epsilon^2\mathbf{I})^{-\mathbf{1}}\mathbf{D^{(i)}d^{(i)}}$. Once we have obtained solutions for $\mathbf{a_*^{(i)}}$ for all $i = 1,\ldots,N$, we have obtained all the TV-PEF coefficients. Let us denote the j^{th} row of the matrix $\mathbf{D^{(i)T}}$ as $\mathbf{u_j^{(i)T}}$. We can now compute the residual or the prediction error for the whole data. Denoting the residual vector as $\mathbf{n}_*$, the i^{th} element is given by $n_{*i} = d_i - \mathbf{u_b^{(i)T}a_*^{(i)}}$, where d_i is the i^{th} element of the data vector $\mathbf{d}$.

The Rank-2 Update Formula for 1D TV-PEFs

In this section, we introduce a fast rank-2 update formula to find the regularized solution to the TV-PEF estimation problem at data sample i, given by $\mathbf{a_*^{(i)}} = (\mathbf{D^{(i)}D^{(i)T}}+\epsilon^2\mathbf{I})^{-\mathbf{1}}(\mathbf{D^{(i)}d^{(i)}})$. The key idea comes from the observation that the matrix inverse that we need to compute for the i^{th} data sample $(\mathbf{D^{(i)}D^{(i)T}}+\epsilon^2\mathbf{I})^{-\mathbf{1}}$, is different from the one we need to compute for the $(i-1)^{th}$ data sample $(\mathbf{D^{(i-1)}D^{(i-1)T}}+\epsilon^2\mathbf{I})^{-\mathbf{1}}$ by only a rank-2 update.

$$\begin{aligned}(\mathbf{D^{(i)}D^{(i)T}}+\epsilon^2\mathbf{I})^{-\mathbf{1}} &= \left[\epsilon^2\mathbf{I} + \sum_{j=1}^{b}\mathbf{u_j^{(i)}u_j^{(i)T}}\right]^{-1} \\ &= (\mathbf{D^{(i-1)}D^{(i-1)T}}+\epsilon^2\mathbf{I} - \mathbf{u_1^{(i-1)}u_1^{(i-1)T}} + \mathbf{u_b^{(i)}u_b^{(i)T}})^{-\mathbf{1}} \ .\end{aligned} \tag{6}$$

This form is important because we can now use the *Sherman-Morrison-Woodbury* formula (see Sherman and Morrison (1949, 1950), Woodbury (1950), Hager (1989) for more details). To present the result in a clean way, we introduce the following matrices:

$$\mathbf{A} = (\mathbf{D^{(i-1)}D^{(i-1)T}}+\epsilon^2\mathbf{I}) \ ; \ \mathbf{U} = \begin{bmatrix}\mathbf{u_1^{(i-1)}u_b^{(i)}}\end{bmatrix} \ ; \ \mathbf{C} = \begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix} \ . \tag{7}$$

Equation (6) then becomes

$$\begin{aligned}&(\mathbf{D^{(i)}D^{(i)T}}+\epsilon^2\mathbf{I})^{-\mathbf{1}} = (\mathbf{A}+\mathbf{UCU^T})^{-\mathbf{1}}\\ &= \mathbf{A^{-1}} - \mathbf{A^{-1}U(C^{-1}+U^TA^{-1}U)^{-1}U^TA^{-1}} \ .\end{aligned} \tag{8}$$

The matrix $(\mathbf{C^{-1}+U^TA^{-1}U})^{-\mathbf{1}}$ in equation (8) is a 2×2 matrix, so its inverse is trivial to calculate. We then have the result that if $\mathbf{A^{-1}} = (\mathbf{D^{(i-1)}D^{(i-1)T}}+\epsilon^{\mathbf{2}}\mathbf{I})^{-\mathbf{1}}$ is known, then $(\mathbf{D^{(i)}D^{(i)T}}+\epsilon^2\mathbf{I})^{-\mathbf{1}}$ can be calculated easily. Specifically, the new inverse at data sample i is different from the old inverse at data sample $i-1$ by a rank-2 update, as seen from equation 8.

The Weighted Non-Stationary TV-PEF Problem

In this section, we introduce the *weighted non-stationary autoregression problem.* Here we only introduce the problem and just say a few key words. For more details, the reader is referred to Claerbout (2016). The central idea is that if we evaluate the TV-PEF coefficients at data sample i, then the assumption that the data is modeled well by the same TV-PEF gets weaker and weaker as one moves further away from data sample i. If we look at equation (4), we can see that all the equations in the regression have equal importance. We would like to correct for the loss of stationarity away from i by modifying these equations. Following Claerbout (2016), an easy and natural way to do this is to down weight equations farther from i. This brings us to the weighted non-stationary autoregression problem.

Instead of solving equation (4), we will instead create a new system of equations by multiplying each row j of the equation by a weight term $w_j = j/b$.

$$\begin{bmatrix} w_1 d_{i-b+1}\\ .\\ .\\ .\\ w_r d_r\\ .\\ .\\ .\\ w_b d_i\end{bmatrix} = \begin{bmatrix} w_1 d_{i-b} & w_1 d_{i-b-1} & . & w_1 d_{i-b-p+2} & w_1 d_{i-b+1-p}\\ . & . & . & . & .\\ . & . & . & . & .\\ . & . & . & . & .\\ w_r d_{r-1} & w_r d_{r-2} & . & w_r d_{r-p+1} & w_r d_{r-p}\\ . & . & . & . & .\\ . & . & . & . & .\\ . & . & . & . & .\\ w_b d_{i-1} & w_b d_{i-2} & . & w_b d_{i-p+1} & w_b d_{i-p}\end{bmatrix} \begin{bmatrix} a_1^{(i)}\\ .\\ .\\ .\\ a_p^{(i)}\end{bmatrix} + \begin{bmatrix} w_1 n_{i-b+1}\\ .\\ .\\ .\\ w_r n_r\\ .\\ .\\ .\\ w_b n_i\end{bmatrix} \ . \tag{9}$$

If we collect all the weight terms into a diagonal matrix $\mathbf{W} : W_{jj} = w_j$, then we can write equation (9) in matrix form as $\mathbf{Wd^{(i)}} = \mathbf{WD^{(i)T}a^{(i)}} + \mathbf{Wn^{(i)}}$. As before, the system can be solved using least squares, and the regularized solution for the TV-PEF coefficients at data sample i is given by $\mathbf{a}_*^{\mathbf{(i)}} = (\mathbf{D^{(i)}W^2D^{(i)T}} + \epsilon^2\mathbf{I})^{-\mathbf{1}}\mathbf{D^{(i)}Wd^{(i)}}$. Once we have obtained solutions $\mathbf{a}_*^{\mathbf{(i)}}$ for all $i = 1, \ldots, N$, the predicted data and the residual can be calculated in the same way as before. Therefore, we now have a way to solve the weighted non-stationary autoregression problem, at least in principle.

Breakdown of The Rank-2 Update Formula

The weighted non-stationary autoregression problem introduced above can be solved in principle by solving the linear regression in equation (9) at each step i using iterative methods. While this is better than having to compute the matrix inverse appearing in the formula at every step i, it is still a lot of computation. Note that in the unweighted case, the rank-2 update formula gave us a way to quickly update the inverse of the matrix appearing in the solution, using the matrix inverse from step $i-1$. But this approach breaks down for the weighted case. The problem becomes clear when we look at the matrices we need to invert at step $i-1$ versus at step i, namely $(\mathbf{D^{(i-1)}W^2D^{(i-1)T}} + \epsilon^2\mathbf{I})$ and $(\mathbf{D^{(i)}W^2D^{(i)T}} + \epsilon^2\mathbf{I})$ respectively. It is easy to see that in this case

$$\begin{aligned}
&(\mathbf{D^{(i)}W^2D^{(i)T}} + \epsilon^2\mathbf{I}) - (\mathbf{D^{(i-1)}W^2D^{(i-1)T}} + \epsilon^2\mathbf{I}) \\
&= \mathbf{u_b^{(i)}u_b^{(i)T}} + \sum_{j=1}^{b-1}(w_j^2 - w_{j+1}^2)\mathbf{u_j^{(i)}u_j^{(i)T}} - w_1^2\mathbf{u_1^{(i-1)}u_1^{(i-1)T}} \\
&= \mathbf{u_b^{(i)}u_b^{(i)T}} - \sum_{j=1}^{b-1}\frac{2j+1}{b^2}\mathbf{u_j^{(i)}u_j^{(i)T}} - \frac{1}{b^2}\mathbf{u_1^{(i-1)}u_1^{(i-1)T}} \ .
\end{aligned} \tag{10}$$

This difference is no longer a rank-2 update. The actual rank of this difference depends on the dimension of the set spanned by the vectors $\{\mathbf{u_1^{(i)}}, \ldots, \mathbf{u_b^{(i)}}, \mathbf{u_1^{(i-1)}}\}$, which is expected to be much larger than 2. As a consequence, the difference of the inverse of the matrices in equation (10) is also not just a rank-2 update, and hence it is no longer possible to update the matrix inverses at every step quickly, in the weighted non-stationary autoregression problem. So the question arises, what can we do to speed up the calculation of TV-PEFs. There does not seem to be an easy way to do this with the way the problem has been posed so far, so an alternative approach is presented next.

An Alternative Formulation - Streaming TV-PEFs

A detailed discussion of this formulation is presented by Fomel et al. (2016). We present only the minimal details of this formulation, as the examples coming up next in this paper uses this method.

The basic assumption behind this formulation is that the TV-PEF coefficients change from one location to the next smoothly. For example, in the 1D case, if the TV-PEF coefficients $\mathbf{a}_*^{\mathbf{(i-1)}}$ at location $i-1$ are known, then the TV-PEF coefficients $\mathbf{a}_*^{\mathbf{(i)}}$ at location i should be closely related to $\mathbf{a}_*^{\mathbf{(i-1)}}$. Therefore, when solving for $\mathbf{a}_*^{\mathbf{(i)}}$, instead of solving the system of equations in (9), we will instead solve the following optimization problem:

$$\begin{aligned} \mathbf{a}_*^{\mathbf{(i)}} &= \arg\min_{\mathbf{a}^{\mathbf{(i)}}} \ ||d_i - \mathbf{u}_{\mathbf{b}}^{\mathbf{(i)T}} \mathbf{a}^{\mathbf{(i)}}||_2^2 + \gamma^2 ||\mathbf{a}^{\mathbf{(i)}} - \mathbf{a}_*^{\mathbf{(i-1)}}||_2^2 \ , \\ \text{where } \ \mathbf{u}_{\mathbf{b}}^{\mathbf{(i)T}} &= \begin{bmatrix} d_{i-1} & d_{i-2} & . & . & d_{i-p+1} & d_{i-p} \end{bmatrix} \ , \text{ as introduced earlier.} \end{aligned} \tag{11}$$

The first term in the objective function in this regularized least squares problem is the data fitting term at location i, which measures the prediction error. The second term regularizes the solution by controlling the difference between $\mathbf{a}_*^{\mathbf{(i)}}$ and $\mathbf{a}_*^{\mathbf{(i-1)}}$, through the relative weighting parameter γ. The second term controls the notion of stationarity in the problem. For example, if γ is large, then $\mathbf{a}_*^{\mathbf{(i)}}$ will change slowly with i, which will imply that stationarity is valid over longer distances. On the other hand, if γ is small, then it will allow $\mathbf{a}_*^{\mathbf{(i)}}$ to change much more rapidly with i, meaning that stationarity will be valid over shorter distances. The implications of this formulation are profound because it turns out that the optimization problem in equation (11) has a simple answer given by,

$$\mathbf{a}_*^{\mathbf{(i)}} = \mathbf{a}_*^{\mathbf{(i-1)}} + \left[\frac{d_i - \mathbf{u}_{\mathbf{b}}^{\mathbf{(i)T}} \mathbf{a}_*^{\mathbf{(i-1)}}}{\gamma^2 + \mathbf{u}_{\mathbf{b}}^{\mathbf{(i)T}} \mathbf{u}_{\mathbf{b}}^{\mathbf{(i)}}} \right] \mathbf{u}_{\mathbf{b}}^{\mathbf{(i)}} \ . \tag{12}$$

This result is the *"the rank-1 update formula"* and is explained in great detail in Fomel et al. (2016). The most important aspect of equation (12) is that the terms $\mathbf{u}_{\mathbf{b}}^{\mathbf{(i)T}} \mathbf{a}_*^{\mathbf{(i-1)}}$ and $\mathbf{u}_{\mathbf{b}}^{\mathbf{(i)T}} \mathbf{u}_{\mathbf{b}}^{\mathbf{(i)}}$ appearing in the equation are simply vector dot products. This new formulation yields a cheap way to update our TV-PEF coefficients using this new formulation. The only question that arises is how to choose the parameter γ. Fomel et al. (2016) provides some preliminary insights on this aspect. I'll simply make the comment that for all results to follow in this paper, γ was chosen by testing different values.

The rank-1 update idea presented in this section does not need to store more than one copy of the TV-PEFs. It updates them at every location i very cheaply based on its value at location $i-1$ using equation (12), and applies them to the data to compute the predicted data and the residual. Once that is done, we move on to the next sample location and so on. The TV-PEFs computed using this algorithm are also referred to as *"streaming TV-PEFs"*. We will use this name to refer to this technique for the remainder of this paper.

The streaming TV-PEFs have the important property that the filter at a particular sample can predict the data with high accuracy only in the neighborhood of the sample. However, the quality of the prediction with the same filter coefficients (at a particular sample) will get worse as one moves further away from the sample. Hence it will not work globally. This disparity between local and global prediction capabilities is important to note, but it is not unexpected due to the very design of non-stationary prediction error filters. This is an important fact that will become evident when we study the 2D missing data prediction problem.

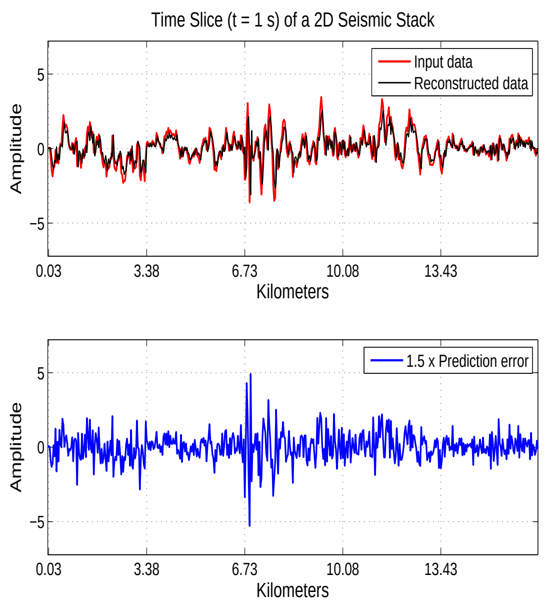
1D Examples of Applications of PEFs vs TV-PEFs

In this section, some comparisons are presented that illustrate the differences between the application of stationary PEFs and the different versions of non-stationary TV-PEFs discussed in this paper, on a 1D data set. The 1D data consisting of a single seismic trace, already introduced and plotted in Figure 1, is chosen for this study.

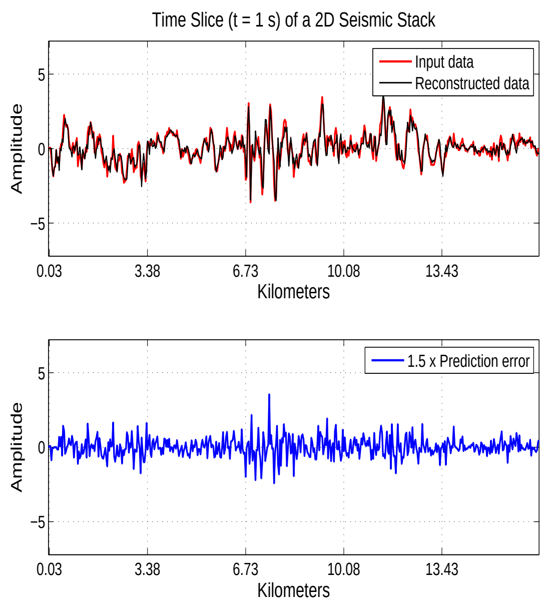
Figures 2(a), 2(b), 2(c) and 2(d) show the results. Figure 2(a) is the result of estimating a stationary autoregression model for the data. Figures 2(b), 2(c) and 2(d) are the results of estimating non-stationary autoregression models for the data using the different techniques outlined in this paper. In all the cases, the order of the autoregression model used is set to $p = 5$. Figure 2(b) shows a clear improvement in predictive power as compared to Figure 2(a) by simply switching from stationary PEFs to non-stationary TV-PEFs with a choice of the window size $b = 10$. The prediction is improved even further when we switch to a weighted non-stationary autoregression model, with all other parameters kept the same as in the case of Figure 2(b). The weighted autoregression fitting result is illustrated in Figure 2(c). We notice that the prediction error has been greatly diminished. Finally, the result of applying streaming TV-PEFs is shown in Figure 2(d). The value of γ in this case was tweaked to match the result of 2(c). Note that Figures 2(c) and 2(d) are almost indistinguishable, but 2(d) has been computed at a fraction of the computational cost of 2(c).

APPLICATIONS OF STREAMING TV-PEF IN 2D

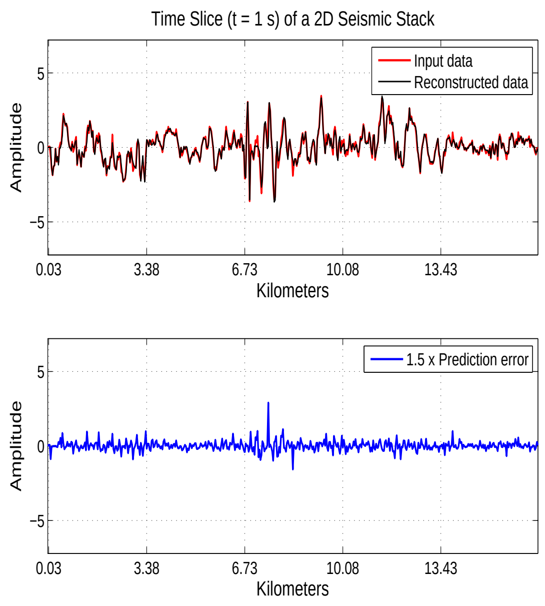
In this section, we extend the streaming TV-PEF idea to two dimensions. Stationary 2D PEFs have been discussed in GIEE, Chapter 7 (Claerbout, 2014). Here we only consider the non-stationary extension of the problem in the context of streaming TV-PEFs.



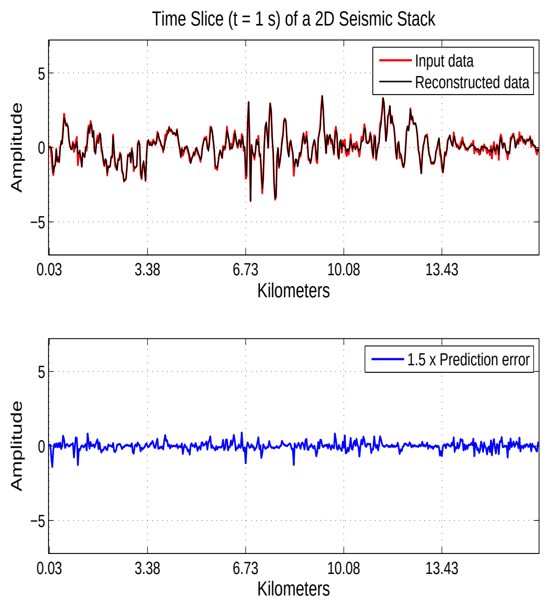
(a)



(b)



(c)



(d)

Figure 2: This figure illustrates the differences between the applications of stationary PEFs and the different kinds of non-stationary PEFs on the 1D data set of Figure 1. Panel (a) is the result of application of stationary autoregression, panel (b) corresponds to the result of non-stationary autoregression, panel (c) corresponds to the result weighted non-stationary autoregression and panel (d) is the result of the application of streaming non-stationary autoregression. In each figure, the red curve is the original data, the black curve is the predicted data and the blue curve is the prediction error. The prediction errors have all been scaled up by a factor of 1.5 for easy viewing. [**CR**]

Streaming 2D TV-PEFs

Let us consider a 2D grid of points where our data values are defined. As an example, one such grid of size 10×10 is shown in Figure 3, where the grid points are numbered 1 through 10 in each dimension. Suppose P_1 is a point in this grid where the TV-PEF coefficients have already been determined. Now consider another point P_2 that is adjacent to P_1 in this grid, where we wish to determine the new TV-PEF coefficients. If we denote the coordinates of the point P_1 as (i, j), then this means that the coordinates of P_2 can be any of the 8 points with coordinates $(i+1, j)$, $(i-1, j)$, $(i, j+1)$, $(i, j-1)$, $(i-1, j-1)$, $(i-1, j+1)$, $(i+1, j-1)$, $(i+1, j+1)$.

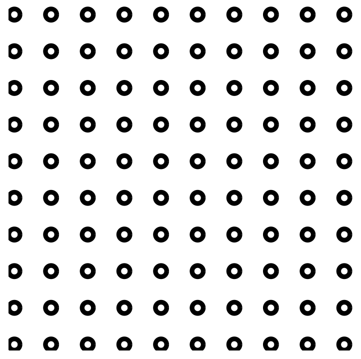


Figure 3: A conceptual 2D 10×10 grid of points where data values are defined. [**CR**]

Let us recall the streaming TV-PEF formulation in 1D that was set up in equation (11). We rewrite it below:

$$\begin{aligned} \mathbf{a}_*^{(\mathbf{i})} &= \arg\min_{\mathbf{a}^{(\mathbf{i})}} \; ||d_i - \mathbf{u}_\mathbf{b}^{(\mathbf{i})\mathbf{T}} \mathbf{a}^{(\mathbf{i})}||_2^2 + \gamma^2 ||\mathbf{a}^{(\mathbf{i})} - \mathbf{a}_*^{(\mathbf{i-1})}||_2^2 \;, \\ \text{where } \; \mathbf{u}_\mathbf{b}^{(\mathbf{i})\mathbf{T}} &= \begin{bmatrix} d_{i-1} & d_{i-2} & . & . & d_{i-p+1} & d_{i-p} \end{bmatrix} \;, \text{ for 1D case.} \end{aligned} \tag{13}$$

It is straightforward to extend this idea to 2D. Let us first discuss what the data fitting term will be. Denote the TV-PEF coefficients at locations P_1 and P_2 by $\mathbf{a}_*^{(\mathbf{P_1})}$ and $\mathbf{a}_*^{(\mathbf{P_2})}$ respectively. The order of the 2D TV-PEFs are assumed to be $p_1 \times p_2$ along the respective dimensions. Denoting the data value at P_2 by $d^{(P_2)}$, the data fitting term is then given by $||d^{(P_2)} - \mathbf{u}_\mathbf{b}^{(\mathbf{P_2})\mathbf{T}} \mathbf{a}^{(\mathbf{P_2})}||_2^2$. Here $\mathbf{u}_\mathbf{b}^{(\mathbf{P_2})\mathbf{T}}$ is the row vector containing the data values whose dot product with the vector of TV-PEF coefficients at P_2, given by $\mathbf{a}_*^{(\mathbf{P_2})}$, produces the predicted data value at P_2. Details on how to determine the vector $\mathbf{u}_\mathbf{b}^{(\mathbf{P_2})\mathbf{T}}$ given a location P_2 and filter size can be found in the book GIEE (Claerbout, 2014), but this is relatively straightforward to do. We now turn to the regularization term. In exact analogy to the 1D case, we define it as $||\mathbf{a}^{(\mathbf{P_2})} - \mathbf{a}_*^{(\mathbf{P_1})}||_2^2$. We can then set up our optimization problem for 2D TV-PEFs, as follows:

$$\mathbf{a}_*^{(\mathbf{P_2})} = \arg\min_{\mathbf{a}^{(\mathbf{P_2})}} \; ||d^{(P_2)} - \mathbf{u}_\mathbf{b}^{(\mathbf{P_2})\mathbf{T}} \mathbf{a}^{(\mathbf{P_2})}||_2^2 + \gamma^2 ||\mathbf{a}^{(\mathbf{P_2})} - \mathbf{a}_*^{(\mathbf{P_1})}||_2^2 \;. \tag{14}$$

As in the 1D case, the parameter γ controls the notion of stationarity in the 2D case. Together with the regularization term, it controls the update in the TV-PEF coefficients from P_1 to P_2. The solution to the optimization problem in equation (14) is now given by:

$$\mathbf{a}_*^{(\mathbf{P_2})} = \mathbf{a}_*^{(\mathbf{P_1})} + \left[\frac{d^{(P_2)} - \mathbf{u_b}^{(\mathbf{P_2})\mathbf{T}} \mathbf{a}_*^{(\mathbf{P_1})}}{\gamma^2 + \mathbf{u_b}^{(\mathbf{P_2})\mathbf{T}} \mathbf{u_b}^{(\mathbf{P_2})}} \right] \mathbf{u_b}^{(\mathbf{P_2})} \; . \tag{15}$$

Note how equation (12) and (15) are similar to one another. Thus we can see now that the update to the TV-PEF coefficients in the 2D case is also very cheap to compute involving only a few vector dot products.

Path Dependence of TV-PEF Updates in 2D

Equation (15) gives us a way to update the TV-PEF coefficients at a new location P_2, if we know the coefficients at a neighboring location P_1. We see that in the 2D case, there are 8 such possibilities of choosing a neighboring point. In general, if we have a continuous non-intersecting path that traverses the 2D grid of points where the data values are defined, like in Figure 3, then the answer to the TV-PEF estimation problem will depend on the chosen path. We plot 4 such simple non-intersecting paths for the 10×10 grid in Figures 4(a), 4(b), 4(c) and 4(d) but there are finitely many more.

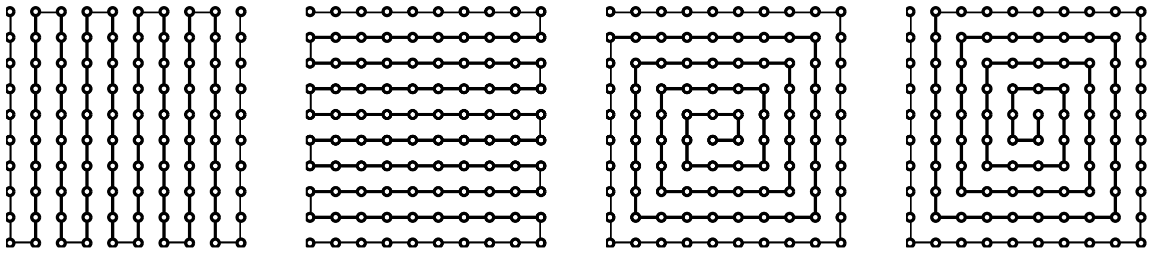


Figure 4: In this figure, we plot 4 different continuous non-intersecting paths that traverse a 2D grid of points of size 10×10. The paths in the four panels are given the following names - (a): N-S-N, (b): W-E-W, (c): Clockwise Spiral and (d): Anti-Clockwise spiral. [**CR**]

In the 1D case, we also have two possible paths for traversing through the 1D array, one that goes from the beginning of the array to the end of the array, and the other that goes in the reverse direction. In our formulation of the problem, we chose the first alternative and neglected the second. But in 2D the multitude of possible non-intersecting paths through the 2D grid can no longer be ignored. We are thus naturally led to the study of path dependence of the TV-PEF updates based on equation (15). However, given a path, one could easily compute the TV-PEF coefficients along each successive location in the path, and calculate the predicted data and the prediction errors (residuals) at each location. Once we have traversed through the entire path, assuming that the path visits all locations in the 2D grid, we would have calculated the prediction errors at all locations. Note that this method is extremely efficient both in terms of storage and computational effort involved at every step. The computational aspect has already been discussed. As for storage, we only need to store one copy of the TV-PEF coefficients in memory, which are continuously updated from one location to the next while traversing through the path. This again justifies the use of the name *"streaming TV-PEFs"* for this algorithm.

Examples of Path Dependence of TV-PEFs in 2D

In this section, we take a 2D seismic stack section and apply the streaming TV-PEF algorithm to it based on the four different paths shown in Figures 4(a), 4(b), 4(c) and 4(d). The data set to be used is shown in Figure 5, which is a 2D seismic stack of size 350×275. Although the original paths were for a 2D grid of size 10×10, one can easily imagine them for the size of the 2D stack. Nothing about the paths fundamentally changes.

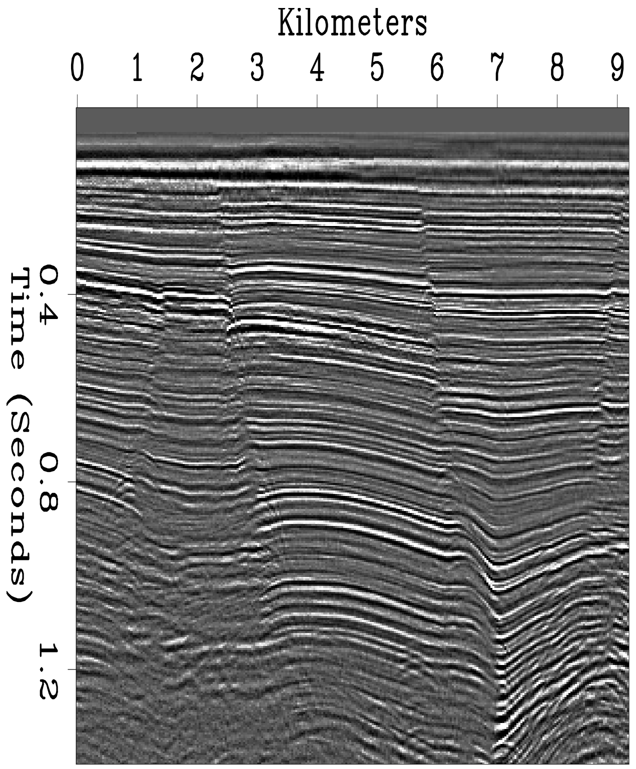
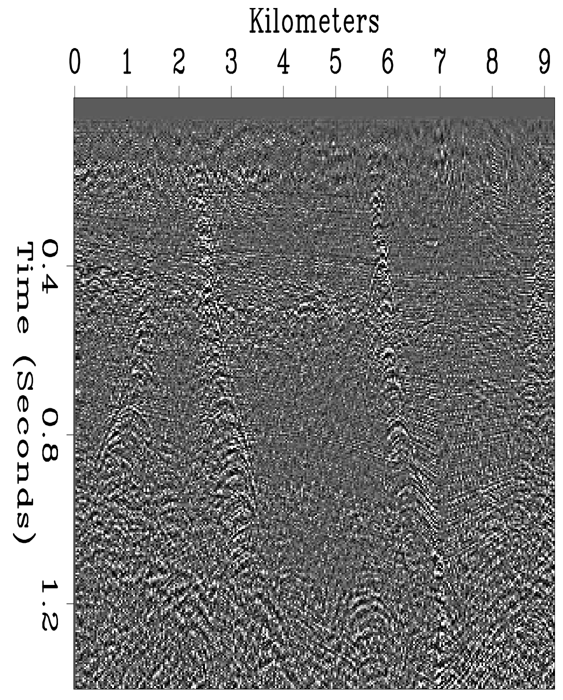


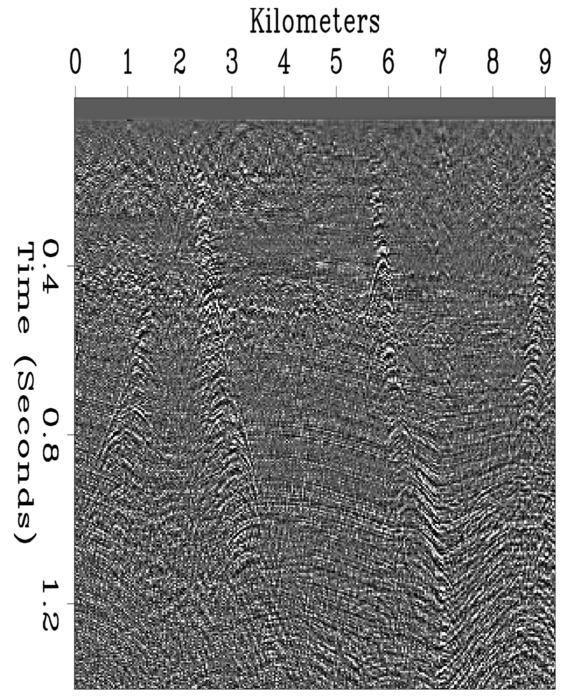
Figure 5: A 2D stack section used in the study of 2D TV-PEFs (courtesy of Western Geophysical, 1974 vintage, Gulf of Mexico). [**ER**]

For this test, we chose a TV-PEF filter of order 5×5, leading to a 22 coefficient 2D TV-PEF, with an implicit 1 and two implicit zeros (for the shape of these filters, see GIEE, Chapter 7 (Claerbout, 2014)). We used the same value of the paramter γ for all paths. The results are plotted in Figures 6(a), 6(b), 6(c) and 6(d). The key observation in the figures is that all the results are slightly different from one another, although by comparing each image to the original 2D stack section in Figure 5 we can see that all of them are qualitatively giving the correct answer. However, we can also see that the amount of coherent energy contained in the residuals corresponding to different paths varies. Figure 6(a), the N-S-N path, shows the least amount of coherent energy (flat events). But on the other 3 panels the coherent energy is much more prominent. This is because the parameter choice γ was made to optimize results of the N-S-N path corresponding to panel (a), and then the same γ was applied to the other paths for comparison. This then suggests that even the choice of γ should depend on the path used for traversal through the data.

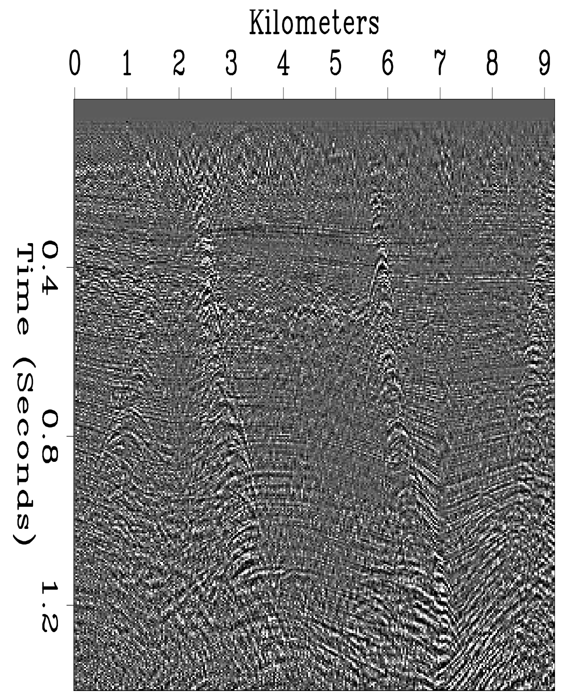
A natural question that arises then is whether we can define the streaming TV-PEFs to be path independent because we would not want our prediction errors to change every time we take a different path through the data, as the previous example suggests that the residuals can end up being non-white. A simple way to address this is to average the TV-PEFs over many different possible paths. But then how should the parameter γ be chosen for each path, especially in light of the observation made in this section that γ indeed depends on the path? Should they be kept them the



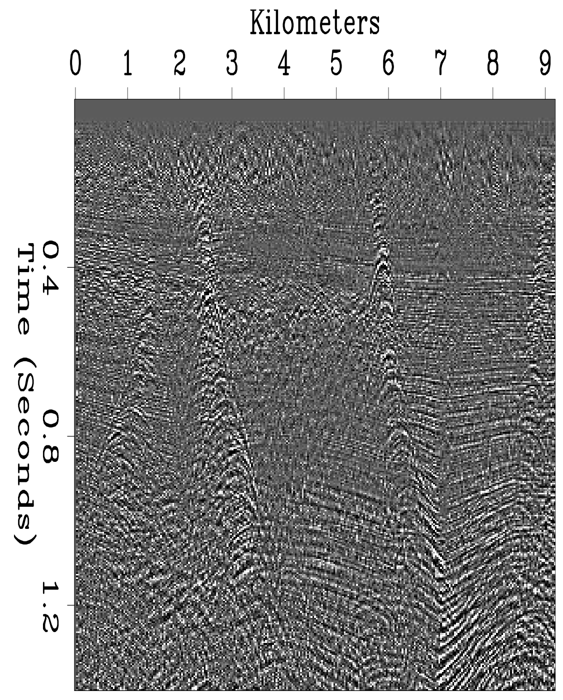
(a)



(b)



(c)



(d)

Figure 6: This figure shows the residuals obtained by the application of streaming TV-PEFs to the 2D seismic stack in Figure 5, for the 4 different paths introduced in Figures 4(a), 4(b), 4(c) and 4(d). The paths that correspond to the four panels are - (a): N-S-N, (b): W-E-W, (c): Clockwise and (d): AntiClockwise respectively. The filter size and choice of γ were kept the same for all the paths. The residuals have all been scaled by a factor of 3 in this display. [**ER**]

same? These questions have not been answered, and are left to a future study.

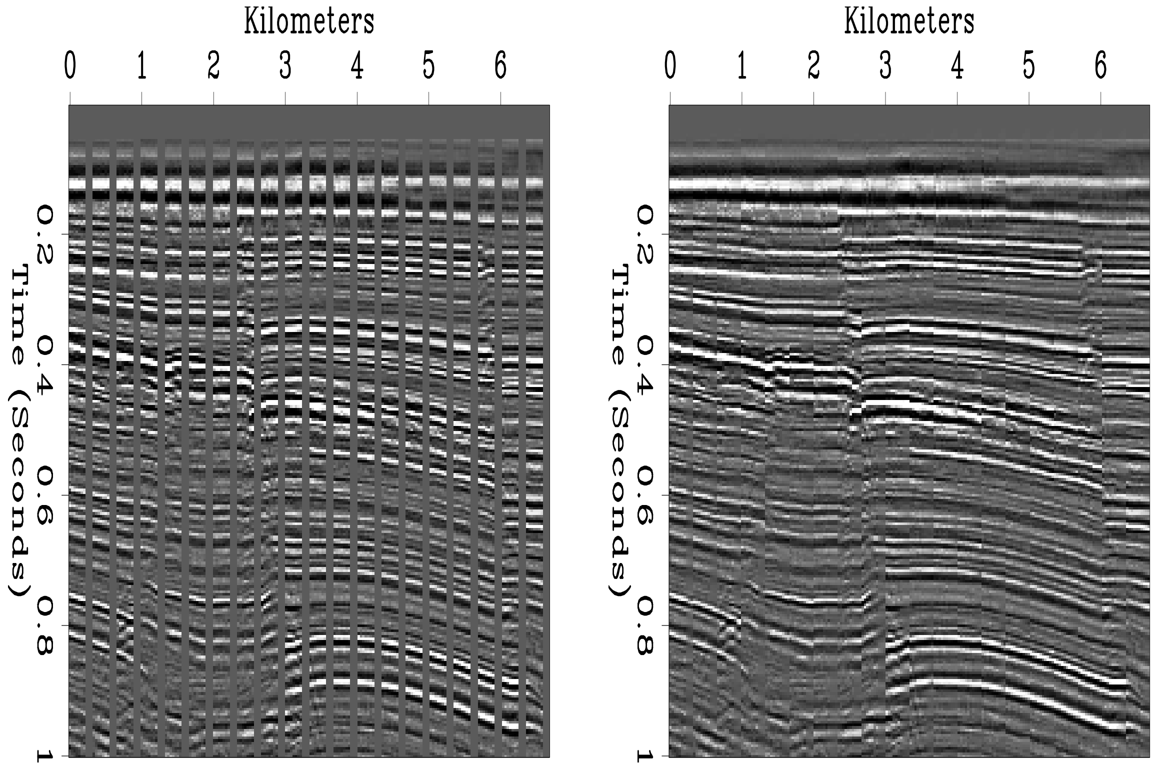
Missing Data Interpolation Using 2D TV-PEFs

In this section, we will look at the application of 2D TV-PEFs to missing data interpolation problems, using the idea of streaming. Interpolation problems are a common way to test the prediction capabilities of a PEF (or TV-PEF) model, and hence are a test of the model's accuracy and performance. We will look at two different examples. In the first example, we will take the data set in Figure 5, and introduce a few holes in the data. We will then attempt to fill them back using streaming 2D TV-PEFs. In the second example, we will attempt to extrapolate data off the edge to the right with the same data set.

Interpolating Holes in The Data

In this example, we take a cropped version of the data set introduced in Figure 5. We introduce a few missing traces in the data by zeroing out 3 consecutive traces at an interval of every 10 traces, and the result is shown in Figure 7(a). Then we interpolated the missing traces using 2D TV-PEFs derived using the streaming algorithm. The path chosen to traverse through the data was from left to right. This means that for the very first missing trace, we go from left to right in each row, until we completely interpolate the trace. Note that this path is different from any of the other paths introduced so far, because now we always go from left to right in each row. Once the first missing trace is interpolated, we recursively apply the process to interpolate all the other missing traces. The final result after the interpolation is shown in Figure 7(b). Examining the Figures 7(a) and 7(b) closely, one can observe that the traces interpolated this way are a good approximation for filling out small gaps. However, one can also observe small discontinuities in some places at the right boundary of every missing trace patch. This is because, the 2D TV-PEFs are interpolating each missing trace based on the character of the traces to its left. Because of non-stationarity of the signal, the traces to the right of the missing patch no longer match the interpolated traces in areas where the lack of stationarity is severe (even over a spacing of just 3 traces !).

The fact that this happens is directly related to the fact that the streaming TV-PEFs are excellent for local prediction but lack global prediction capabilities, as was pointed out previously. It may be possible to overcome this problem by interpolating the data over many different paths, and then averaging the results. For example, one could interpolate the same data set in Figure 7(a) by first reflecting it left to right (by taking a mirror image about a vertical plane), and then performing interpolation the same way followed by reflecting the result again from left to right. This could then be averaged with the result in Figure 7(b). For holes of more general shapes, one could devise many such paths to do the interpolation along and then average the results. These ideas are currently under active research and will be published at a later time.



(a) (b)

Figure 7: Interpolation of missing traces using 2D TV-PEF streaming from left to right. Left panel (a) is the original data with 3 consecutive missing traces every 10 traces apart. The right panel (b) shows the data set after interpolation. [**ER**]

Extrapolating Data Off The Edge

In this example, we explore the behavior of 2D streaming TV-PEFs when we extrapolate data off the edge. The input data set is created by zeroing out 50 traces from the right edge of a 2D seismic stack, as shown in Figure 8(a). After several tests, the best extrapolation strategy was found to be along paths from left to right, one column at a time. So to interpolate the first missing column, we start streaming for each row from the left to the right and upon reaching the column to be extrapolated, we use the TV-PEFs calculated for each row of the last filled column, to predict the data values for the next missing column. This process is iterated for each new missing column. The result is shown in Figure 8(b). The extrapolated data blows up for certain rows at the extreme edge. This happens because if the data samples are growing in amplitude before the first missing column is encountered, then the TV-PEFs continue to predict this trend in the extrapolation zone and we end up with large amplitudes. To prevent this from happening one can damp the TV-PEF coefficients used in the extrapolation. The details of this are covered in Fomel et al. (2016), but the idea is to damp the filter coefficients as $a_{ij} \leftarrow a_{ij} \times (1-\epsilon)^{|i|+|j|}$ for every new interpolated column, where the implicit unit spike of the 2D filter is at $(i,j) = (0,0)$ and some fixed $0 < \epsilon < 1$. The end result of extrapolation with damping is shown in Figure 8(c). The amplitudes of the extrapolated data values now decay smoothly with increasing extrapolation distance and the regions of the data with larger amplitudes take longer to decay than regions with relatively smaller amplitudes, which is completely expected.

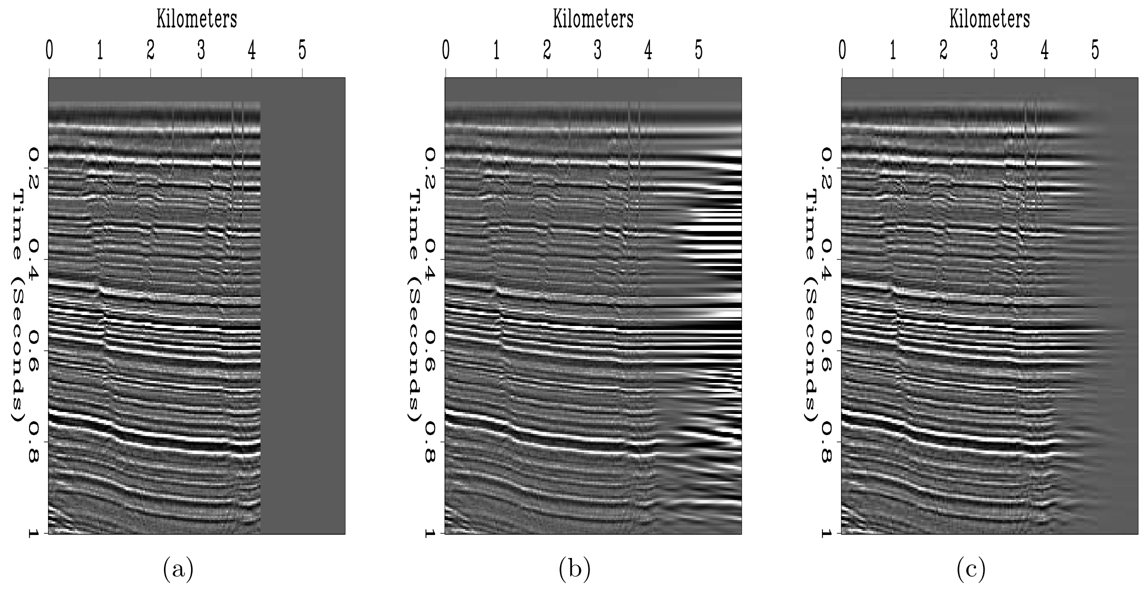


Figure 8: Extrapolation of missing traces using 2D TV-PEF streaming from left to right. Left panel (a) is the original data with 50 missing traces at the right edge. The middle panel (b) shows the data set after extrapolation without damping. The right panel (c) shows the data set after extrapolation with damping. [**ER**]

DISCUSSIONS AND CONCLUSIONS

In this paper, we explored some of the computational challenges of non-stationary prediction error filter design. It was seen that the weighted non-stationary autoregression problem could not be solved using a fast algorithm due to the breakdown of the rank-2 update, as compared to the non-stationary autoregression problem. However, as the former is clearly superior compared to the latter in predicting non-stationary signals, we wanted to come up with an alternative formulation that captures the features of the weighted non-stationary autoregression problem, but is easier to solve computationally. This led to the streaming TV-PEF idea, where the solution is based on a rank-1 update formula. With this method, one can update the TV-PEF coefficients from one location to the next at the cost of just a few vector dot products.

We investigated some of the consequences of using this algorithm to update the TV-PEF coefficients in 1D, and saw how they compared with stationary PEFs and other non-stationary TV-PEF design algorithms, both in terms of computational cost and accuracy. We studied the path dependence of the results using streaming TV-PEFs when applied to 2D problems. It was seen that the choice of path is directly related to the choice of γ, as the prediction errors corresponding to different paths for the same γ were different. In particular, the residuals along certain paths were closer to being random identical independent distributions than others. Some missing data interpolation problems were also studied in 2D using the streaming TV-PEF idea. In the first example, we interpolated a few missing traces at a certain regular interval along the spatial axis. This example revealed how the TV-PEFs lack global prediction capabilities far away from the location where they were designed for. In the second example, we extrapolated data off the edge of a data set which showed that extrapolation without damping can lead to exponential growth. But this effect can be controlled by damping the filter coefficients with increasing extrapolation distance.

The role of the regularization parameter γ remains to be investigated in more detail. It would be nice to have some theoretical results that connect the weighted non-stationary autoregression problem of equation (9) to the streaming TV-PEF formulation of equation (11). Another aspect that remains to be investigated has to do with coming up with a framework to determine the streaming TV-PEF coefficients in 2D, that eliminates the path dependence of the answer. One possible way to eliminate path dependence is to average the TV-PEF coefficients over many possible paths, something that is currently under research.

ACKNOWLEDGMENTS

I'd like to thank Jon Claerbout for introducing me to the problem of prediction error filters in his class at Stanford, that ultimately led to the work done in this paper. He was extremely generous with his time and knowledge, and this work would not have been possible without his guidance and ideas. He also helped me to straighten out a lot of mathematical notation for a geophysical audience. Eileen Martin, Yinbin Ma, Stewart Levin and Bob Clapp meticulously proof-read my paper and suggested valuable ideas investigated in this paper, for which they deserve a special mention. A note of thanks goes to Prof. Biondo Biondi, my advisor at Stanford, for giving me an opportunity to come to Stanford to study. I also extend my gratitude to the students at SEP for helping me out with many aspects of SEPlib software, which probably saved me several hours. Finally, I'd like to thank all the sponsors of SEP for their generous support and for providing the data sets used in this work.

REFERENCES

Akaike, H., 1969, Fitting autoregressive models for prediction: Annals of the Institute of Statistical Mathematics, **21**, 243–247.

———, 1974, A new look at the statistical model identification: Automatic Control, IEEE Transactions on, **19**, 716–723.

Claerbout, J., 2014, Geophysical image estimation by example: lulu.com.

———, 2016, Streaming nonstationary prediction error (I) : SEP-Report, **163**, 265–269.

Crawley, S., 2001, Seismic trace interpolation with nonstationary prediction-error filters: PhD thesis, Stanford University.

Crawley, S., R. Clapp, and J. Claerbout, 1998, Decon and interpolation with nonstationary filters: SEP-Report, **97**.

Crawley, S., R. Clapp, J. Claerbout, et al., 1999, Interpolation with smoothly nonstationary prediction-error filters: Presented at the 1999 SEG Annual Meeting.

Curry, W., 2005, Interpolating with data-space prediction-error filters: SEP-Report, **120**, 237–247.

———, 2008, Interpolation with prediction-error filters and training data: PhD thesis, Stanford University.

Fomel, S., J. Claerbout, S. Levin, and R. Sarkar, 2016, Streaming nonstationary prediction error (II) : SEP-Report, **163**, 271–277.

Golub, G. H., P. C. Hansen, and D. P. O'Leary, 1999, Tikhonov regularization and total least squares: SIAM Journal on Matrix Analysis and Applications, **21**, 185–194.

Golub, G. H. and C. F. Van Loan, 2012, Matrix computations, volume **3**: JHU Press.

Hager, W. W., 1989, Updating the inverse of a matrix: SIAM review, **31**, 221–239.

Hannan, E. J. and B. G. Quinn, 1979, The determination of the order of an autoregression: Journal of the Royal Statistical Society. Series B (Methodological), 190–195.

Hestenes, M. R. and E. Stiefel, 1952, Methods of conjugate gradients for solving linear systems, volume **49**: Journal of Research of the National Bureau of Standards.

Paige, C. C. and M. A. Saunders, 1982, Lsqr: An algorithm for sparse linear equations and sparse least squares: ACM Transactions on Mathematical Software (TOMS), **8**, 43–71.

Ruan, K., J. Jennings, E. Biondi, R. G. Clapp, et al., 2015, Industrial scale high-performance adpative filtering with PEF applications: SEP-Report, **160**, 181–192.

Sherman, J. and W. J. Morrison, 1949, Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix: Annals of Mathematical Statistics, 621–621.

———, 1950, Adjustment of an inverse matrix corresponding to a change in one element of a given matrix: The Annals of Mathematical Statistics, **21**, 124–127.

Woodbury, M. A., 1950, Inverting modified matrices: Memorandum report, **42**, 106.

Yilmaz, Ö., 2001, Seismic data analysis, volume **1**: Society of Exploration Geophysicists Tulsa.