Sparse graph-regularized dictionary learning for suppressing random seismic noise

Lina Liu[1], *Jianwei Ma*[2], *Gerlind Plonka*[3]

ABSTRACT

We propose a new regularization method for the sparse representation and denoising of seismic data. Our approach is based on two components, a sparse data representation in a learned dictionary and a similarity measure for image patches that is evaluated using the Laplacian matrix of a graph.

Dictionary learning (DL) methods aim to find a data-dependent basis or a frame that admits a sparse data representation while capturing the characteristics of the given data. We propose two algorithms for dictionary learning based on clustering and singular value decomposition (SVD), called first and second dictionary construction (FDC and SDC). Besides employing an adapted dictionary we also consider a similarity measure for the local geometric structures of the seismic data using the Laplacian matrix of a graph. The proposed method achieves better denoising performance than existing denoising methods, both in terms of peak signal-to-noise ratio values and visual estimation of weak-event preservation. Comparisons of experimental results on field data using traditional FX deconvolution (FX-Decon) and curvelet thresholding methods are also provided.

[1]Department of Mathematics and Institute of Geophysics, Harbin Institute of Technology, Harbin, China, liulina_lx@163.com

[2]Department of Mathematics and Institute of Geophysics, Harbin Institute of Technology, Harbin, China, jma@hit.edu.cn

[3]Institute for Numerical and Applied Mathematics, University of Göttingen, Göttingen, Germany, plonka@math.uni-goettingen.de

INTRODUCTION

Two critical and widely studied problems in seismic data processing are (1) extracting the fundamental structures of seismic data by sparse representation, and (2) denoising. The two problems are closely related since for given noisy measurements, the extraction of the important data structures can be at the same time understood as a denoising process, where one has to separate the original seismic data from noise. This process is an ill-posed inverse problem, meaning that there are many significantly different possible solutions that come close to optimize it. Often, a first linear denoising (usually based on averaging of measurements) is already applied during the acquisition process in order o achieve data sets with lower noise level. To improve denoising results, regularization methods are required that provide stable and unique solutions. A regularization method is a technique for solving ill-posed inverse problems by formulating a minimization problem, where the objective function includes besides the fidelity term (that measures the distance to the given data) additional "regularization" terms that force special properties of the desired data. Therefore, regularization methods serve to impose desired features on subsurface images. The well-known Tikhonov regularization models force smoothness of the data and tend to produce models where discontinuities are blurred. They have been for instance applied to solve the acoustic inverse problem of crosshole seismology, see Reiter and Rodi (1996). Other regularization methods using total variation minimization can provide high-resolution images of the subsurface, where edges and discontinuities are properly preserved. For example, Anagaw and Sacchi (2012) applied total variation minimization to the problem of estimating acoustic velocity perturbations using a single scattering Born modeling operator. Another regularization method that has attracted a renewed interest and considerable attention in signal processing literature is L_1-norm regularization. Here a regularization term is included into the objective function that forces sparsity of the data in the spatial domain or in a transform domain. This approach is popular for seismic data processing methods such as seismic data denoising and interpolation, see Herrmann and Hennenfent (2008). However, the above mentioned methods do not exploit the full geometric structure characteristics of the data such as the spatial relative position of the events. To overcome this issue, additional regularization terms have to be incorporated.

In this paper, we propose a regularization approach with two additional constraints:
a) forcing sparsity of the data in a learned (data-dependent) dictionary, and
b) minimizing a similarity measure that is evaluated by a Laplacian matrix of a graph (graph regularization). Graphs are mathematical structures used to model pairwise relations between objects.
In the following, we explain these two new constraints in more detail.

a) In this paper, dictionaries are bases or frames of the finite dimensional space $\mathbb{R}^N$ of real vectors of length N or equivalently of $\mathbb{R}^{n\times n}$, the space of digital images with n rows and n columns, where $N = n^2$. A dictionary with k elements (or atoms) is a set of k vectors in $\mathbb{R}^N$ that can be simply represented by a matrix $\mathbf{D} \in \mathbb{R}^{N\times k}$, where in this case the k columns of length N are the atoms of $\mathbf{D}$. In other words, each column of the matrix $\mathbf{D}$ contains one (vectorized) $n \times n$ matrix atom. If $k \geq N$ and $\mathbf{D}$ has full rank N, then all vectors $\mathbf{y} \in \mathbb{R}^N$ can be represented as linear combinations of atoms in $\mathbf{D}$, i.e., there exists an $\mathbf{x} \in \mathbb{R}^k$ with $\mathbf{y} = \mathbf{D}\mathbf{x}$. If $k > N$ and $\mathbf{D}$ has full rank N, then the representation of $\mathbf{y}$ by dictionary atoms is not longer unique, and the dictionary is called over-complete. The dictionary matrix $\mathbf{D}$ can be seen as a transform matrix, and one is usually interested to construct $\mathbf{D}$ such that the given seismic data (either vectors or image patches) can be sparsely represented by this transform, i.e., the main structure can be already presented by a linear combination of a small number of atoms of the dictionary. Indeed, sparse transforms offer a nice way to implement sparse representations and denoising of seismic data. Whether denoising is effective or not strongly depends on dictionary selection or on the choice of the sparse transform. Wavelet transforms have been often used for solving seismic data denoising problems in Chanerley and Alexander (2002). Zhang and Ulrych (2003) presented a physical wavelet frame for seismic data denoising that used the special characteristics of seismic data. Nowadays, curvelets are one of most popular tools for seismic data representation and denoising, see Hennenfent and Herrmann (2006), Neelamani et al.

(2008), Ma and Plonka (2010). In addition to such sparse representation methods, Bonar and Sacchi (2012) applied a nonlocal means algorithm to seismic data denoising by using similar samples or pixels within the image regardless of their spatial proximity.

A new, popular, and highly effective approach for solving seismic denoising problems is the sparse representation of seismic data using a learned dictionary. Learned dictionaries adapt to the particular structure of the given data. Compared to denoising methods by Pan et al. (1999) and Starck et al. (2002) based on dictionaries being not data-dependent, a dictionary trained through a dictionary learning method can provide a sparser representation of seismic data. Different dictionary learning methods have already been applied to the seismic data denoising processing see Bechouche and Ma (2014) Engan et al. (1999). Kaplan et al. (2009) presented a review of sparse coding and its application to random noise attenuation. Yu et al. (2015) proposed a dictionary learning seismic data denoising method that used a data driven tight frame (DDTF). However, almost all patch-based dictionary learning denoising methods convert seismic data patches or volumes into one-dimensional (1D) vectors for training, and thereby lose the inherent 2D or 3D geometric structures of the seismic data.

In this paper, we will propose two new methods for dictionary learning using the singular value decomposition (SVD) and a clustering method. Our approach connects ideas of Zeng et al. (2015) with ideas borrowed from generalized wavelet constructions. For the two methods, we will use training patches from the given seismic data to teach the dictionary. For denoising problems in particular, these patches are often noisy. The two methods are simple to implement and may be of relevance not only for denoising but also for other seismic data applications. The first dictionary construction (FDC) employs the SVD of covariance matrices of similar training patches. The second dictionary construction (SDC) uses the similarity of training patches and a clustering method. The main idea of our two dictionary learning algorithms is the following. In the first step, we cluster the training patches into subsets according to a similarity measure using a binary tree structure. Then we build average patches (center matrices) for each subset. The dictionary elements are then constructed as linear combinations of small-rank approximations of the obtained average patches.

b) The second ingredient of our proposed regularization method is a term that measures similarity of patches. This similarity measure is determined by the Laplacian matrix of a graph and is therefore called graph regularization term. The graph is used as a tool for presenting the similarity between the training patches. In recent years, graph regularization has been applied for describing the relationship between image patches, see Elmoataz et al. (2008), Bougleux et al. (2009), Kheradmand and Milanfar (2014), e.g. by building a K-nearest neighbor graph to encode the geometric information in the data. Such graph-based methods have been applied for image denoising by Tang et al. (2013) and Yankelevsky and Elad (2016), for image representation by Zheng et al. (2011), and for image super-resolution by Lu et al. (2012). In this work, we will employ the assumption that if two training data patches are close with respect to the obtained similarity measure then their sparse dictionary representations are also close. Similar assumptions have been also used in various manifold learning methods to explore such structures, see Belkin and Niyogi (2003), Liu et al. (2014), Zheng et al. (2011).

Our denoising method is iterative. First we derive a dictionary of training patches. For the dictionary we use a randomly selected subset of the set of all patches of the given noisy seismic data. In the experiments in this paper we have taken 60 % of all patches for the first field data and 68 % for the second field data. Having fixed the dictionary we present the noisy data sparsely in this dictionary and solve the minimization problem, where the graph regularization term is applied to the sparse data in the transformed (dictionary) domain. We repeat this process with the resulting denoised data patches that now serve in a next iteration to obtain an improved dictionary etc.

The rest of the paper is organized as follows: In the next section we present our regularization method. We describe the minimization problem obtained by incorporating the sparse representation in a dictionary and in particular derive the graph regularization term that measures the similarity between image patches. We summarize the complete denoising algorithm in the section titled DENOISING METHOD (Algorithm 1). This algorithm can be applied with an arbitrary dictionary $\mathbf{D}$. In the following section titled METHODS FOR DICTIONARY LEARNING we describe the two new dictionary learning methods FDC and SDC. Finally we deal with the problem of how to solve the complete minimization problem arising from this regularization approach in section titled METHODS TO SOLVE THE MINIMIZATION PROBLEM, (Algorithm 2). The mathematical justification of Algorithm 2 can be found in Appendix A. Experimental results and comparative discussions using two kinds of field data are provided in the section titled EXPERIMENTS. Conclusions and final thoughts are provided in the section titled CONCLUSION. Appendix B contains an extensively described toy example for the two dictionary learning methods.

SPARSE REPRESENTATION BY LEARNED DICTIONARIES AND GRAPH REGULARIZATION

Let us first introduce some notation. Let $\{\mathbf{I}_1, \ldots, \mathbf{I}_m\}$ (e.g. $\mathbf{I}_j \in \mathbb{R}^{n\times n}$, where $j = 1, 2, \ldots, m$ represent m images, each consisting of n traces with n samples) represent a given training set of seismic data. Further, let $\mathbf{Y} := [\mathbf{y}_1, \ldots, \mathbf{y}_m]$ be an $N \times m$ matrix with $N = n^2$, where the columns $\mathbf{y}_j = \operatorname{vec}(\mathbf{I}_j) \in \mathbb{R}^N$ are the vectorized patches, i.e., we stack all columns of $\mathbf{I}_j$ into one vector $\mathbf{y}_j$ starting with the first. With $\mathbf{D} := [\mathbf{d}_1 \ldots \mathbf{d}_k] \in \mathbb{R}^{N\times k}$ we denote the dictionary (matrix), where each column $\mathbf{d}_i \in \mathbb{R}^N$ is one atom in the dictionary. Here, and elsewhere we use the notation := for (mathematical) definitions. The Frobenius norm of the matrix $\mathbf{Y}$ is the square root of the sum of all squared entries of $\mathbf{Y}$, i.e., we have $\|\mathbf{Y}\|_F^2 = \sum_{i=1}^m \|\mathbf{y}_i\|_2^2 = \sum_{i=1}^m \sum_{j=1}^N y_{i,j}^2$, where $\|\mathbf{y}_i\|_2 = \left(\sum_{j=1}^N y_{i,j}^2\right)^{1/2}$ denotes the usual Euclidean norm of $\mathbf{y}_i$ in $\mathbb{R}^N$.

Once we have determined a suitable dictionary $\mathbf{D}$, a sparsity promoting optimization problem can be formulated as

$$\min_{\mathbf{X}\in\mathbb{R}^{k\times m}} \left(\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \lambda\|\mathbf{X}\|_0\right), \tag{1}$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{k \times m}$ denotes the matrix of sparse coefficient vectors, such that the set of training patches $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$ is sparsely represented by $\mathbf{DX}$. Here, λ is a regularization parameter, and $\|\mathbf{X}\|_0$ counts the number of non-zero entries of $\mathbf{X}$. This optimization problem is "NP-hard" which means that it cannot be exactly solved in polynomial computation time. Therefore $\|\cdot\|_0$ is usually replaced by the convex norm $\|\cdot\|_1$, and the relaxed optimization problem reads

$$\min_{\mathbf{X} \in \mathbb{R}^{k \times m}} \left(\|\mathbf{Y} - \mathbf{DX}\|_F^2 + \lambda \|\mathbf{X}\|_1 \right), \tag{2}$$

where $\|\mathbf{X}\|_1 := \sum_{i=1}^m \|\mathbf{x}_i\|_1 = \sum_{i=1}^m \sum_{j=1}^k |x_{i,j}|$ is the sum of absolute values of all entries in $\mathbf{X}$. The model (2) has been widely used for data denoising in the last decade, and many algorithms have been developed to solve these optimization problems efficiently, see Beck and Teboulle (2009), Chambolle and Pock (2011) and Needell and Vershynin (2010).

We extend this model for sparse representation of seismic images by adding two important ingredients, **dictionary learning** and **graph regularization**.

Dictionary learning. We want to employ a dictionary $\mathbf{D}$ that is adaptively learned from the data. In machine learning and in other applications, see Aharon et al. (2006), Elad and Aharon (2006) and Dong et al. (2013), the following model has been studied already,

$$\min_{\mathbf{X} \in \mathbb{R}^{k \times m}, \mathbf{D} \in \mathbb{R}^{N \times k}} \left(\|\mathbf{Y} - \mathbf{DX}\|_F^2 + \lambda \|\mathbf{X}\|_* \right) \tag{3}$$

with $\|\cdot\|_*$ being either $\|\cdot\|_0$ or $\|\cdot\|_1$, and where K-SVD for dictionary learning is employed. The notion K-SVD has been coined by Aharon et al. (2006). This well-known dictionary learning method to solve (3) is a generalization of K-means clustering and combines SVD and the orthogonal matching pursuit (OMP) method. The approach is based on alternating optimization: For a fixed $\mathbf{D}$, an improved sparse matrix $\mathbf{X}$ is computed e.g. by the OMP method, a greedy algorithm that provides good solutions for (1). For fixed $\mathbf{X}$, the dictionary $\mathbf{D}$ is updated using singular value decomposition (SVD). However, K-SVD is very expensive, see Liu et al. (2017). In particular, many iteration steps are needed since the atoms of the dictionary are updated one by one, and each time the SVD of a large $N \times N$ matrix is needed.

At the same time, two-dimensional structures of the training patches are not explicitly used but all patches are considered only in vectorized form.

Therefore, other methods for dictionary learning came up aiming at a cheaper technique to replace the K-SVD. For example, in Cai et al. (2014) and Liu et al. (2017) a priori dictionary structure is imposed to reduce the number of parameters, such as block-wise Toeplitz structure or (directional) tensor-product frames. Running times of the three methods have been compared in Liu et al. (2017), and the two new methods are both significantly cheaper than the K-SVD method.

Motivated by ideas in Zeng et al. (2015), we will propose two different methods for adaptive dictionary learning in the Section titled METHODS FOR DICTIONARY LEARNING. These methods try to exploit the two-dimensional geometric structure of the training data in a more direct way and are essentially cheaper than K-SVD.

Graph regularization. The second ingredient is an extension of the functional in (3) by an additional term that measures the similarity between the image patches. Here we follow ideas in Zheng et al. (2011) and Yankelevsky and Elad (2016), and employ a graph that represents the internal topology of the training patches. A graph is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related".

For the given set of training patches $\mathbf{I}_1, \ldots, \mathbf{I}_m$, we construct a weighted undirected complete graph $G(V, E, \mathbf{W})$, where the finite set $V = \{\mathbf{I}_1, \ldots, \mathbf{I}_m\}$ of m vertices represents the given patches. We say that a graph is *undirected* if each edge of a graph is non-directional. Further, $E = V \times V$ is a set of weighted edges, i.e., each two patches $\mathbf{I}_i$, $\mathbf{I}_j$ are connected by an edge, and the corresponding weights are collected in the weight matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$. We measure the similarity of the training patches $\mathbf{I}_i$ and $\mathbf{I}_j$ using the Frobenius norm of its difference $\|\mathbf{I}_i - \mathbf{I}_j\|_F$. For a pre-defined K, we fix the K nearest neighbors of $\mathbf{I}_i$ (being different from $\mathbf{I}_i$ itself) by inspecting $\|\mathbf{I}_i - \mathbf{I}_k\|_F^2$ for $k \in \{1, \ldots, i-1, i+1, \ldots, m\}$. Then, $\mathbf{I}_j$ belongs to the K nearest neighbors of $\mathbf{I}_i$ if the distance $\|\mathbf{I}_i - \mathbf{I}_j\|_F^2$ belongs to the K smallest distances of the set of all $m-1$ obtained distances. We define the symmetric weight matrix $\mathbf{W} = (W_{i,j})_{i,j=1}^m$ by $W_{i,j} = 1$ if $\mathbf{I}_j$ is among the K nearest neighbors of $\mathbf{I}_i$ or if $\mathbf{I}_i$ is among the K nearest neighbors of $\mathbf{I}_j$. Otherwise, we set $W_{i,j} = 0$. In particular, $W_{i,i} = 0$ for $i = 1, \ldots, m$. Thus, each row (or column) of $\mathbf{W}$ contains at least K ones. The process of graph building is illustrated in Figure 1. The degree of each vertex $\mathbf{I}_i$, i.e., the number of all edges with weight 1 to the vertex $\mathbf{I}_i$ is given by $\Delta_i = \sum_{j=1}^m W_{i,j}$. Introducing the diagonal matrix $\mathbf{\Delta} = \mathrm{diag}(\Delta_1, \ldots, \Delta_m) \in \mathbb{R}^{m\times m}$, the Laplacian matrix of the graph G is now defined as the matrix $\mathbf{L} = \mathbf{\Delta} - \mathbf{W} \in \mathbb{R}^{m\times m}$. By construction, $\mathbf{L}$ is symmetric and positive semidefinite, its non-diagonal entries are non-positive, and the sum of all entries in each column (or row) is zero.

A direct computation shows that the term

$$\mathrm{Tr}(\mathbf{Y}\mathbf{L}\mathbf{Y}^T) = \sum_{i,j=1}^m W_{i,j}\|\mathbf{I}_i - \mathbf{I}_j\|_F^2 = \sum_{i,j=1}^m W_{i,j}\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 = \sum_{\mathbf{I}_i \sim \mathbf{I}_j} \|\mathbf{I}_i - \mathbf{I}_j\|_F^2$$

measures the similarity of neighborhood patches in the graph, where we have used the notation $\mathbf{I}_i \sim \mathbf{I}_j$ if $W_{i,j} = 1$. In other words, $\mathbf{I}_i \sim \mathbf{I}_j$ means that either $\mathbf{I}_i$ is one of the K nearest neighbors of $\mathbf{I}_j$, or $\mathbf{I}_j$ is one of the K nearest neighbors of $\mathbf{I}_i$, and in the last sum all terms with weight $W_{i,j} = 0$ are droped. Further, the trace $\mathrm{Tr}(\mathbf{A})$ of a quadratic matrix $\mathbf{A}$ denotes the sum of its diagonal entries. For each j, the vector $\mathbf{D}\mathbf{x}_j$ is assumed to be a good approximation to $\mathbf{y}_j$. Since the (fixed) transform matrix $\mathbf{D}$ induces a linear mapping, we can suppose that the vectors $\mathbf{x}_j$, $j = 1, \ldots, m$ possess a similar topological structure as $\mathbf{y}_j$, $j = 1, \ldots, m$, and particularly that, if $\mathbf{y}_i$ and $\mathbf{y}_j$ are K-neighbors with a small distance $\|\mathbf{y}_i - \mathbf{y}_j\|_2$, we also have that $\|\mathbf{x}_i - \mathbf{x}_j\|_2$ is small. Therefore, we incorporate the term

$$\mathrm{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T) = \sum_{i,j=1}^m W_{i,j}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \sum_{\mathbf{I}_i \sim \mathbf{I}_j} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$$

and obtain the new minimization problem

$$\min_{\mathbf{X}\in\mathbb{R}^{k\times m}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \alpha\,\mathrm{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T) + \lambda\|\mathbf{X}\|_1 \tag{4}$$

with a regularization parameter $\alpha \geq 0$, where the Laplacian matrix $\mathbf{L}$ and the learned dictionary $\mathbf{D}$ only depend on the training data $\mathbf{Y}$. This model is simpler than the model considered in Yankelevsky and Elad (2016), since we have not incorporated the graph Laplacian into the dictionary learning process.

The weight matrix $\mathbf{W}$ of the graph G can also be defined differently, as e.g. by

$$W_{i,j} = \begin{cases} \frac{1}{2\pi h^2} \exp\left(\frac{-\|\mathbf{I}_i - \mathbf{I}_j\|_F^2}{2h^2}\right) & \text{for } i \neq j \\ 0 & \text{for } i = j \end{cases}$$

using the Gaussian kernel and some parameter h, or alternatively by employing a thresholded kernel. Note, however, that the weight matrix $\mathbf{W}$ and hence the Laplacian matrix $\mathbf{L}$ are already determined by the vectorized training patches, since the Frobenius norm does not exploit any two-dimensional structures of the training patches $\mathbf{I}_1, \ldots, \mathbf{I}_m$ that cannot be observed after vectorization. However, we will propose dictionary learning methods that also incorporate two-dimensional characteristic features into the dictionary atoms.

DENOISING METHOD

The regularization model in (4) can now be employed to obtain a denoising method and at the same time a sparse data representation of the given seismic data $\mathbf{Y}$ within the dictionary $\mathbf{D}$. The procedure has three stages. First we derive a dictionary $\mathbf{D}$ that is learned from the given data, see Section METHODS FOR DICTIONARY LEARNING. Then we fix $\mathbf{D}$ and solve the minimization problem with regard to $\mathbf{X}$ using Algorithm 2, see Section METHODS TO SOLVE THE MINIMIZATION PROBLEM. For the special denoising application we employ the given noisy seismic data themselves to acquire the set of training patches, i.e., we take patches of the noisy data to obtain the set of training patches $\{\mathbf{I}_1, \ldots, \mathbf{I}_m\}$. The third stage contains the reconstruction of the denoised data $\mathbf{Y}_D$ using the learned dictionary $\mathbf{D}$ and the computed coefficient matrix $\mathbf{X}$. The new denoising scheme consists of the following steps that we present in Algorithm 1.

```
Algorithm 1: Denoising algorithm based on dictionary learning and graph regularization

Input
Noisy training data $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$
Number of iterations
Parameters $K$, $\alpha$ and $\lambda$

Algorithm
1: Set $\mathbf{Y}_D := \mathbf{Y}$.
Loop through steps 2-5 until the given number of iterations is achieved:
2: Compute the Laplacian matrix $\mathbf{L}$ for the given training set $\mathbf{Y}_D$.
3: Determine the dictionary $\mathbf{D}$ by a dictionary learning algorithm based on $\mathbf{Y}_D$.
4: Solve the minimization problem $\min_{\mathbf{X}\in\mathbb{R}^{k\times m}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \alpha\,\mathrm{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T) + \lambda\|\mathbf{X}\|_1$.
5: Reconstruct the data $\mathbf{Y}_D := \mathbf{D}\mathbf{X}$.

Output
Denoised data $\mathbf{Y}_D$
```

In the next sections, we describe the two essential steps 3 and 4 of the algorithm in more detail.

For step 2 to compute the Laplacian matrix $\mathbf{L} := (L_{i,j})_{i,j=1}^m$, we can proceed as follows. First we compute the symmetric matrix of all distances $\|\mathbf{I}_i - \mathbf{I}_j\|_F^2$. Then, in each row $i = 1, \ldots, m$, we order the nonzero values by size and collect the indices j corresponding to the K smallest distances in the index set I_i. For $i, j \in \{1, \ldots, m\}$ and $i \neq j$ we put

$$L_{i,j} := \begin{cases} -1 & \text{for } j \in I_i \,\text{or}\, i \in I_j, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we set $L_{i,i} = \sum\limits_{\substack{j=1\\ j\neq i}}^{m} |L_{i,j}|$.

METHODS FOR DICTIONARY LEARNING

We will propose here two dictionary learning methods to perform step 3 of Algorithm 1. Both are based on a special partition tree structure. We construct the dictionaries in two steps. First we compute a special tree structure to partition the set of our training patches. Then, we compute the dictionary based on the obtained subset partitions in the tree. The first step will be different for our two proposed methods, while the method to determine the dictionary from the tree structure will be the same. The second method uses a simpler similarity measure for the partition tree construction and is simpler to implement, but also gives less good results in applications.

The two methods use the set of training patches $\mathbf{I}_1, \ldots, \mathbf{I}_m \in \mathbb{R}^{n \times n}$ and we construct the dictionary elements $\mathbf{D}_\ell$, $\ell = 1, \ldots, k$ in the form of patches, such that $\mathbf{d}_\ell = \operatorname{vec} \mathbf{D}_\ell$ are the columns (atoms) of $\mathbf{D}$. The dictionary elements $\mathbf{D}_\ell$ will be linear combinations of low-rank approximations of suitable center matrices, which are averages of subsets of $\mathbf{I}_j$ with high similarity. This approach lets us incorporate 2-D features into the dictionary that cannot be simply found by employing only the vectorized training patches.

First dictionary construction (FDC). Motivated by the ideas in Zeng et al. (2015), we employ for the first dictionary learning method a top-bottom two-dimensional subspace partition (TTSP) as follows.

Step 1: Construction of the partition tree. For the given training set $\mathbf{I}_1, \ldots, \mathbf{I}_m \in \mathbb{R}^{n \times n}$ of image patches we compute the mean

$$\mathbf{C} := \frac{1}{m} \sum_{i=1}^{m} \mathbf{I}_i \in \mathbb{R}^{n \times n}$$

and the two non-symmetric $(n \times n)$-covariance matrices

$$\mathbf{C}_L := \frac{1}{m} \sum_{i=1}^{m} (\mathbf{I}_i - \mathbf{C})(\mathbf{I}_i - \mathbf{C})^T, \qquad \mathbf{C}_R := \frac{1}{m} \sum_{i=1}^{m} (\mathbf{I}_i - \mathbf{C})^T (\mathbf{I}_i - \mathbf{C}). \tag{5}$$

Observe that $\mathbf{C}_L$ and $\mathbf{C}_R$ possess the same eigenvalues. Now compute the normalized eigenvectors $\mathbf{u}$ and $\mathbf{v}$ for the maximal eigenvalue of $\mathbf{C}_L$ and $\mathbf{C}_R$,

$$\mathbf{u} := \operatorname*{argmax}_{\|\mathbf{x}\|_2=1} \mathbf{x}^T \mathbf{C}_L \mathbf{x}, \qquad \mathbf{v} := \operatorname*{argmax}_{\|\mathbf{x}\|_2=1} \mathbf{x}^T \mathbf{C}_R \mathbf{x},$$

representing the main structures of the training patches being not captured by the mean patch $\mathbf{C}$. Compute

$$s_i := \mathbf{u}^T \mathbf{I}_i \mathbf{v}, \qquad i = 1, \ldots, m.$$

These numbers measure how well each patch correlates with the structure, and will be used to partition the set of $\{\mathbf{I}_1, \ldots, \mathbf{I}_m\}$ into two partial sets. For this purpose we order these numbers by size,

$$s_{\ell_1} \leq s_{\ell_2} \leq \ldots \leq s_{\ell_m},$$

i.e., the new index set $\{\ell_1, \ldots, \ell_m\}$ is a permutation of the index set $\{1, 2 \ldots, m\}$ such that s_{ℓ_1} denotes the smallest number of the set $\{s_1, \ldots, s_m\}$, s_{ℓ_2} the second smallest, and so forth. We compute

$$\hat{\kappa} := \operatorname*{argmin}_{1 \leq \kappa \leq m-1} \left[\sum_{r=1}^{\kappa} \left(s_{\ell_r} - \frac{1}{\kappa} \sum_{\nu=1}^{\kappa} s_{\ell_\nu} \right)^2 + \sum_{r=\kappa+1}^{m} \left(s_{\ell_r} - \frac{1}{m-\kappa} \sum_{\nu=\kappa+1}^{m} s_{\ell_\nu} \right)^2 \right]. \tag{6}$$

Using $\hat{\kappa}$, the partition $\{\mathbf{I}_{\ell_1}, \ldots, \mathbf{I}_{\ell_{\hat{\kappa}}}\} \cup \{\mathbf{I}_{\ell_{\hat{\kappa}+1}}, \ldots, \mathbf{I}_{\ell_m}\}$ is derived. This is the clustering k-means method for the special case $k = 2$, that can be simply solved exactly for the set of numbers $\{s_{\ell_1}, \ldots, s_{\ell_m}\}$ by evaluating the term in (6) for each κ.

Having found this first partition, we then partition the two obtained subsets further using the same scheme. This procedure yields a binary tree. The root node of the tree is associated with the full set of training patches $\{\mathbf{I}_1, \ldots, \mathbf{I}_m\}$ and the two children nodes are associated with the subsets $\{\mathbf{I}_{\ell_1}, \ldots, \mathbf{I}_{\ell_{\hat{\kappa}}}\}$ and $\{\mathbf{I}_{\ell_{\hat{\kappa}+1}}, \ldots, \mathbf{I}_{\ell_m}\}$. We introduce the corresponding subsets of indices $\Lambda_1 := \{1, \ldots, m\}$ associated with the root node and $\Lambda_2 := \{\ell_1, \ldots, \ell_{\hat{\kappa}}\} \subset \Lambda_1$, $\Lambda_3 := \{\ell_{\hat{\kappa}+1}, \ldots, \ell_m\} \subset \Lambda_1$ associated with the two children nodes. Applying the partition to the subsets of image patches, number the index sets at the nodes of the tree by going through each layer from left to right. We stop the further partition of a subset once it contains less than a predefined number of elements.

While this partitioning procedure is equivalent to the first part of the TTSP algorithm proposed in Zeng et al. (2015), we compose the data-dependent dictionary differently from Zeng et al. (2015) as follows.

Step 2: Determine the dictionary from the partition tree. Each node in the tree is now associated with a subset of training patches $\{\mathbf{I}_j\}_{j\in\Lambda_k}$, where $\Lambda_k \subset \{1,\ldots,m\}$ denotes the subset of indices of these patches. Now, for each node of the tree, i.e., for each index set Λ_k, we compute the mean (center) matrix

$$\mathbf{C}_k := \frac{1}{|\Lambda_k|}\sum_{i\in\Lambda_k}\mathbf{I}_i$$

and the normalized eigenvectors to the maximal eigenvalue of $\mathbf{C}_k\mathbf{C}_k^T$ and $\mathbf{C}_k^T\mathbf{C}_k$,

$$\mathbf{u}_k := \operatorname*{argmax}_{\|\mathbf{x}\|_2=1}\mathbf{C}_k\mathbf{C}_k^T\mathbf{x}, \qquad \mathbf{v}_k := \operatorname*{argmax}_{\|\mathbf{x}\|_2=1}\mathbf{C}_k^T\mathbf{C}_k\mathbf{x}.$$

If λ_k denotes the maximal singular value of $\mathbf{C}_k$ then we have $\mathbf{C}_k\mathbf{C}_k^T\mathbf{u}_k = \lambda_k^2\mathbf{u}_k$, $\mathbf{C}_k^T\mathbf{C}_k\mathbf{v}_k = \lambda_k^2\mathbf{v}_k$. Thus, $\lambda_k\mathbf{u}_k\mathbf{v}_k^T$ is the best rank-1 approximation of $\mathbf{C}_k$, since $\mathbf{u}_k$ and $\mathbf{v}_k$ are the first vectors in the singular value decomposition of $\mathbf{C}_k$.

The dictionary is now determined as follows. We fix the first dictionary element

$$\mathbf{D}_1 := \mathbf{u}_1\mathbf{v}_1^T \tag{7}$$

capturing the main structure of the mean $\mathbf{C} = \mathbf{C}_1$. Further, for each pair of children nodes with index sets Λ_{2k} and Λ_{2k+1} to the same parent node with center matrices $\mathbf{C}_{2k}$ and $\mathbf{C}_{2k+1}$ we set

$$\tilde{\mathbf{D}}_k := \lambda_{2k}\mathbf{u}_{2k}\mathbf{v}_{2k}^T - \lambda_{2k+1}\mathbf{u}_{2k+1}\mathbf{v}_{2k+1}^T, \qquad \mathbf{D}_k := \frac{\tilde{\mathbf{D}}_k}{\|\tilde{\mathbf{D}}_k\|_F}, \tag{8}$$

thereby capturing the difference of main structures of $\mathbf{C}_{2k}$ and $\mathbf{C}_{2k+1}$. This procedure differs from Zeng et al. (2015).

Remarks. *1. The construction of the dictionary* $\mathbf{D}$ *can be done simultaneously during the construction of the partition tree. Having the matrix* $\mathbf{C} = \mathbf{C}_1$*, we can already determine the first dictionary element* $\mathbf{D}_1$ *according to (7). At the second level, after having found the* *partition of training sets into two subsets, we find the two corresponding center matrices* $\mathbf{C}_2$ *and* $\mathbf{C}_3$ *and can construct the second dictionary element* $\mathbf{D}_2$ *according to (8) etc. In Appendix B we give an extensive example of this construction and present the tree obtained by this procedure.*

2. Instead of approximating the centers $\mathbf{C}_k$ *by a rank-1 matrix, we can also use a more exact approximation with matrices of higher rank using the singular value decomposition. For sparse representation purposes and if the patches do not contain noise, one may even use the centers* $\mathbf{C}_k$ *directly instead of their approximations of smaller rank. In this case we obtain* $\mathbf{D}_1 = \mathbf{C}_1/\| \mathbf{C}_1\|_F$ *and* $\mathbf{D}_k = (\mathbf{C}_{2k} - \mathbf{C}_{2k+1})/\| \mathbf{C}_{2k} - \mathbf{C}_{2k+1}\|_F$*.*

3. Our construction in (7) and (8) avoids the problem of having dictionary elements being very similar. Strong similarity of dictionary elements can occur when the center matrices of the subsets in the nodes of the tree, or a small-rank approximation of these center matrices are employed as dictionary atoms, see Zeng et al. *(2015).*

4. Our dictionary construction can be understood as a generalized wavelet approach, where the dictionary elements for $k > 1$ *are "wavelet elements" while* $\mathbf{D}_1$ *is the low-pass element.*

Second dictionary construction (SDC). Here we propose a second method for dictionary learning.

Step 1: Construction of the partition tree. This time, we construct the partition tree in a different way, using the similarity of training patches as we did already for the graph regularization. We use the weights

$$w_{i,j} := \|\mathbf{I}_i - \mathbf{I}_j\|_F^2 \tag{9}$$

to measure the similarity between $\mathbf{I}_i$ and $\mathbf{I}_j$.

First we order the patches such that $\mathbf{I}_1$ has the smallest Frobenius norm. Now, we compute the weights $w_{1,j}$ for $j = 1, \ldots, m$ according to (9) and order them by size,

$$w_{1,\ell_1} \leq w_{1,l_2} \leq \ldots w_{1,\ell_m},$$

where $\ell_1 = 1$ since $w_{1,1} = 0$ is the smallest weight. Now, similarly as before in the first construction, we divide the set of training patches into two subsets $\{\mathbf{I}_{\ell_1}, \ldots, \mathbf{I}_{\ell_{\hat{\kappa}}}\}$ and $\{\mathbf{I}_{\ell_{\hat{\kappa}+1}}, \ldots, \mathbf{I}_{\ell_m}\}$ using

$$\hat{\kappa} := \operatorname*{argmin}_{1 \le \kappa \le m-1} \left[\sum_{r=1}^{\kappa} \left(w_{1,\ell_r} - \frac{1}{\kappa} \sum_{\nu=1}^{\kappa} w_{1,\ell_\nu} \right)^2 + \sum_{r=\kappa+1}^{m} \left(w_{1,\ell_r} - \frac{1}{m-\kappa} \sum_{\nu=\kappa+1}^{m} w_{1,\ell_\nu} \right)^2 \right].$$

We proceed to partition the obtained subsets using the same procedure and obtain a binary tree, where each node is associated with a subset of training patches.

Step 2: Determine the dictionary from the partition tree. We proceed as in the first construction and apply the rank-1 approximation of the mean $\mathbf{C} = \mathbf{C}_1$ as the first dictionary element as well as the normalized differences of the rank-1 approximations of the centers of each pair of two children in the partition tree as further dictionary elements.

We consider a toy example to illustrate the two dictionary learning methods in the appendix B (see also Figure 2).

Let us summarize the computational cost for the two dictionary learning methods FDC and SDC. For FDC we have to compute for the first partition the matrices $\mathbf{C}_L$ and $\mathbf{C}_R$ and need $\mathcal{O}(mn^3)$ operations. The computation of $\hat{\kappa}$ is the most expensive with $\mathcal{O}(m^2)$ operations. All further partition steps are cheaper since the number of patches decays. We obtain an overall computational cost of at most $\mathcal{O}(km(m+n^3))$ for the first step of FDC, where $\mathcal{O}(k)$ bounds the number of partitions in the tree, and k is the number of wanted dictionary elements. For the SDC method, we need to compute the weights $w_{i,j}$ in (7) with an effort of $\mathcal{O}(m^2n^2)$ first. The remaining effort is governed by computing $\hat{\kappa}$ with $\mathcal{O}(m^2)$ operations in the first partition and with less effort in the further partitions. Since the weights $w_{i,j}$ have to be computed only once, we get an overall computational cost of at most $\mathcal{O}(m^2(n^2+k))$.

The second step namely the determination of the dictionary from the tree is the same for both methods and needs at most $\mathcal{O}(n^3k)$ operations. For an over-complete dictionary we can assume that $k > n^2$ and the number m of training patches is much larger than k.

Therefore, both algorithms have a computational cost of $\mathcal{O}(m^2k)$. The second method is cheaper than the first if we assume that the weights $w_{i,j}$ in (9) are already known, since they are also needed for computing the graph regularization term.

METHODS TO SOLVE THE MINIMIZATION PROBLEM

In the fourth step of the proposed denoising Algorithm 1 we have to solve the minimization problem

$$\min_{\mathbf{X}\in\mathbb{R}^{k\times m}} \left(\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 + \alpha \operatorname{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T) + \lambda\|\mathbf{X}\|_1\right), \tag{10}$$

for given (noisy) training data $\mathbf{Y}$ and the given dictionary $\mathbf{D} = [\mathbf{d}_1 \ldots \mathbf{d}_k]$, where $\mathbf{d}_j = \operatorname{vec}\mathbf{D}_j \in \mathbb{R}^N$ are the dictionary elements constructed in the last section.

We suggest solving it using the split Bregman iteration in Goldstein and Osher (2009) and Plonka and Ma (2011), which is in the considered case equivalent to the Alternating Direction Method of Multipliers (ADMM), see Yankelevsky and Elad (2016). For other approaches see Chambolle and Pock (2011) and Lee et al. (2007). Since we have not found in the literature a suitable presentation of a computational method for the particular problem stated in equation (10), we present the mathematical details to solve this optimization problem in Appendix A. These considerations lead to the following Algorithm 2 to perform step 4 of Algorithm 1.

Algorithm 2 : Solving the minimization problem

Input

Noisy training data $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m] \in \mathbb{R}^{N \times m}$

Laplacian matrix $\mathbf{L} \in \mathbb{R}^{m \times m}$

Learned dictionary $\mathbf{D} \in \mathbb{R}^{N \times k}$

$\mathbf{X}^0 = \mathbf{Z}^0 = \mathbf{B}^0 = \mathbf{0}_{k \times m}$

Parameters $\lambda, \mu, \alpha > 0$

Number of iterations ℓ

Algorithm

```
Iterate until the given number of iterations is achieved:
1: Compute $\mathbf{X}^{\ell+1}$ as the solution of $(\mathbf{D}^T\mathbf{D} + \mu\mathbf{I})\mathbf{X} + \alpha\mathbf{X}\mathbf{L} = \mathbf{D}^T\mathbf{Y} + \mu(\mathbf{Z}^\ell - \mathbf{B}^\ell)$.
2: Compute $\mathbf{Z}^{\ell+1}$ componentwisely by employing soft shrinkage
    $z_{i,j}^{\ell+1} = \mathcal{T}_{\lambda/2\mu}(x_{i,j}^{\ell+1} + B_{i,j}^\ell) := \begin{cases} x_{i,j}^{\ell+1} + B_{i,j}^\ell - \frac{\lambda}{2\mu} & \text{for } (x_{i,j}^{\ell+1} + B_{i,j}^\ell) \geq \frac{\lambda}{2\mu}, \\ x_{i,j}^{\ell+1} + B_{i,j}^\ell + \frac{\lambda}{2\mu} & \text{for } (x_{i,j}^{\ell+1} + B_{i,j}^\ell) \leq -\frac{\lambda}{2\mu}, \\ 0 & \text{otherwise,} \end{cases}$
    for $i = 1, \ldots, k$, $j = 1, \ldots, m$.
3: Update $\mathbf{B}^{\ell+1} = \mathbf{B}^\ell - \mathbf{Z}^{\ell+1} + \mathbf{X}^{\ell+1}$.
```

Output X

The matrix equation

$$(\mathbf{D}^T\mathbf{D} + \mu\mathbf{I})\mathbf{X} + \alpha\mathbf{X}\mathbf{L} = \mathbf{D}^T\mathbf{Y} + \mu(\mathbf{Z}^\ell - \mathbf{B}^\ell) \tag{11}$$

that has to be solved with regard to the unknown $(k \times k)$ matrix $\mathbf{X}$ in the first step of Algorithm 2, has a very special structure. We show how it can be converted into a usual linear equation system. Let the Kronecker product of two matrices $\mathbf{A} = (a_{ij})_{i,j=1}^k \in \mathbb{R}^{k \times k}$ and $\mathbf{B} \in \mathbb{R}^{k \times k}$ be defined as

$$\mathbf{A} \otimes \mathbf{B} := \begin{bmatrix} a_{11}\mathbf{B} & \ldots & a_{1k}\mathbf{B} \\ \vdots & & \vdots \\ a_{k1}\mathbf{B} & \ldots & a_{kk}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{k^2 \times k^2},$$

and let $\operatorname{vec} \mathbf{X} = \mathbf{x} \in \mathbb{R}^{k^2}$ denote the vectorization of the matrix $\mathbf{X}$ formed by stacking the columns of $\mathbf{X}$ into a single vector. We can use now the property

$$\operatorname{vec}(\mathbf{A}\,\mathbf{X}\,\mathbf{B}) = (\mathbf{B}^T \otimes \mathbf{A}) \operatorname{vec} \mathbf{X} = (\mathbf{B}^T \otimes \mathbf{A})\, \mathbf{x},$$

see Horn and Johnson (1991), Lemma 4.3.1, and obtain

$$\begin{aligned} \operatorname{vec}\left((\mathbf{D}^T\mathbf{D} + \mu\mathbf{I})\mathbf{X}\right) &= \left(\mathbf{I}_k \otimes (\mathbf{D}^T\mathbf{D} + \mu\mathbf{I})\right)\mathbf{x}, \\ \operatorname{vec}(\alpha\mathbf{X}\mathbf{L}) &= \alpha\,(\mathbf{L} \otimes \mathbf{I}_k)\,\mathbf{x}, \end{aligned}$$

where $\mathbf{I}_k$ denotes the identity matrix of size $k \times k$. Thus (11) is equivalent with the linear equation system

$$\left[\left(\mathbf{I}_k \otimes (\mathbf{D}^T\mathbf{D} + \mu\mathbf{I})\right) + \alpha\,(\mathbf{L} \otimes \mathbf{I}_k)\right]\, \mathbf{x} = \operatorname{vec}\left(\mathbf{D}^T\mathbf{Y} + \mu(\mathbf{Z}^\ell - \mathbf{B}^\ell)\right). \tag{12}$$

Remarks. *1. In the process of solving the minimization problem (10), the parameters α, μ, and λ have to be determined. According to the graph regularization method in Yankelevsky and Elad (2016) and Lee et al. (2007), we have tested the values $\alpha \in [0.01, 100]$. Best results are obtained for $\alpha \in [1.2, 2.0]$. In our experiments we observed that the parameter μ does not strongly influence the denoising result. However, μ and λ are strongly related since we have to employ a thresholding in step 2 of the Algorithm 2 with the parameter $\frac{\lambda}{2\mu}$. The value $\frac{\lambda}{2\mu}$ determines the number of elements in the vector $\mathbf{Z}$ that are kept. If $\frac{\lambda}{2\mu}$ is too large, the essential information on the data will de deleted by the soft shrinkage method.*

2. The equation system (12) is of size $k^2 \times k^2$ but has a very special structure due to the Kronecker product of the involved matrices. For efficient solutions we refer to Bartels and Stewart (1972) and Bhatia and Rosenthal (1997), who have proposed a method with computational cost of $\mathcal{O}(k^3)$ which is comparable to usual costs for linear systems of size $k \times k$.

EXPERIMENTS

In this section, we demonstrate the seismic data denoising performance of the proposed FDC-graph and SDC-graph method in Algorithm 1 and compare it to the regularization method without the graph regularization term, i.e., with $\alpha = 0$. We highlight the advantages and disadvantages of using the graph regularization term on a field data set with known amounts of random noise added. Comparisons to a traditional FX (frequency domain in the time direction and spatial domain in trace direction) deconvolution (FX-Decon) algorithm in Canales (1984) and Gulunay (1986) and to state-of-the-art curvelet denoising by thresholding in Hennenfent and Herrmann (2006) are also provided. A Matlab implementation for the FX-Decon method is available from M.D. Sacchi, see `http://www-geo.phys.ualberta.ca/saig/SeismicLab`. The Matlab code for curvelet denoising can be found in the curvelet toolbox, see `http://www.curvelet.org`.

An objective quality metric is critical for comparing denoising methods. The peak-signal-to-noise ratio (PSNR) in dB is given by

$$PSNR = 20 * \log_{10}\left(\frac{\max(\mathbf{A})}{\text{std2}(\mathbf{A}-\mathbf{B})}\right),$$

where $\mathbf{B}$ presents the denoised image data and $\mathbf{A}$ is the clean image data, see e.g. Zoran and Weiss (2011). Here $\max(\mathbf{A})$ is just the largest value occurring in the gray scale image $\mathbf{A}$ and std2 computes the standard deviation of $\mathbf{A}-\mathbf{B}$. Computation time (in seconds), and the recovery error $\frac{\|\mathbf{A}-\mathbf{B}\|_F^2}{\|\mathbf{A}\|_F^2}$ are used as the performance measurements. The parameter K for building the graph Laplacian, where we need to fix the K nearest neighbors, is set to $K = 6$.

To qualitatively evaluate our algorithm's effectiveness on seismic data, we compare the following six algorithms for denoising of seismic data in Figure 4(b), where the noise standard deviation σ is 25: FX-Decon, Curvelet algorithm, FDC-OMP, SDC-OMP, FDC-Graph, and SDC-Graph. For all methods we tried to choose the parameters to achieve optimal denoising results. Here FDC-OMP and SDC-OMP denote the denoising methods that we obtain by solving the optimization problem (1) by orthogonal matching pursuit with a dictionary $\mathbf{D}$ learned by FDC or SDC. The minimization problem (1) even uses the superior semi-norm $\|\mathbf{X}\|_0$ instead of $\|\mathbf{X}\|_1$, but does not include the graph-regularization term $\alpha\text{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T)$. Similarly, FDC-Graph and SDC-Graph denote the methods obtained from

Algorithm 1, using the first (FDC) or the second (SDC) dictionary learning approach. To see the impact of the graph regularization term, we have taken the FDC-OMP and the SDC-OMP here instead of applying Algorithm 1 with $\alpha = 0$, since these methods work slightly better. We employ only 2 iterations in Algorithms 1 for FDC-OMP, SDC-OMP, FDC-Graph, and SDC-Graph.

The FX-Decon algorithm has been applied to a window of size 64×64 samples overlapping by 8 samples in both the time and spatial dimensions with filters of length 6. In our numerical experiments with the FX-Decon algorithm we have studied different window sizes from 16 to 64 and overlapping from 20% to 70%. We obtained the above parameters that achieve the best denoising result with respect to the PSNR measure. The curvelet denoising algorithm decomposes the noisy data into 5 different scales and different directions, and 91% percent of the small curvelet coefficients are removed by thresholding. For the seismic data in Figure 4(b), the FDC-Graph and SDC-Graph algorithms use a set of 961 patches that constitutes the training set, where the size of patches is 16×16. The regularization parameters taken in Algorithm 2 to solve in (14) (see Appendix A) have been empirically chosen to be $\alpha = 1.6$; $\lambda = 0.8$; $\mu = 0.05$ for FDC-Graph and $\alpha = 1.6$; $\lambda = 1.0$; $\mu = 0.04$ for SDC-Graph. For the dictionary learning part, the number of patches contained in each node determines the number of layers of the tree. We have set the minimal number of patches in a subset corresponding to one node to 6 for the FDC-Graph algorithm and to 16 for the SDC-Graph algorithm.

The dictionaries learned by the FDC and the SDC approach are shown in Figure 5, where the set of 961 training patches has been randomly selected from the noisy data in Figure 4(b). The main difference between FDC-OMP/ SDC-OMP and FDC-Graph/ SDC-Graph is that by graph regularization the geometric similarity of data patches is employed. We observe from the close-up window in Figure 6 that the results using the graph regularization methods maintain the even continuous and weak features of the original data. Figure 7 shows the separated noise, i.e., the difference between denoising results and field data one in Figure 4(a). From the single trace comparison in Figure 8 we see that all methods achieve strong amplitudes close to the clear data trace. But for weak amplitudes, the denoising results using the Graph regularization methods are much better compared to the methods without them. The statistical results with various noise standard deviation are shown in Figure 9. The graph regularization term is helpful for capturing the features of seismic data.

Finally, we applied FX-Decon, curvelet denoising, and the proposed FDC-Graph, and SDC-Graph algorithms to denoising to the second field data (the size of data is 256×256) in Figure 4(c). FX-Decon was applied to windows of size 50×50 samples overlapping on 16 samples in both dimensions and with filters of length 4. The Curvelet denoising algorithm decomposes the noisy data into 5 scales and 98% percent of the small curvelet coefficients are removed by thresholding. For the FDC-Graph and SDC-Graph algorithms we employ Algorithm 1 and Algorithm 2, where the parameters are empirically chosen to be $\alpha = 1.2$; $\lambda = 1.0$; $\mu = 0.04$ for FDC-Graph and $\alpha = 1.2$; $\lambda = 1.0$; $\mu = 0.03$ for SDC-Graph. The field data two does not have noise added. The noise there is the original noise recorded during acquisition.

The denoising results are presented in Figure 10 and separated noise of field data two is given in Figure 11. We see that the seismic event in the curvelet denoising result is smoother than for the other methods in the Figure 10. In Figure 11, the separated noise obtained by curvelet denoising contains still information of the signal while the FDC-Graph, and SDC-Graph algorithms work significantly better for noise separation. From the magnified windows in Figures 10 and 11, the FDC-Graph and SDC-Graph methods better recover the weak features found in the original data compared to the other methods.

For the two field data, the computational costs of the proposed algorithms are similar, since we have the same size of field data and have used the same size of training patches 16×16 as well as the same minimal number of patches in subsets in the dictionary construction as for the first test, see Figure 9(e).

CONCLUSION

In this paper we have proposed a regularization approach for seismic data processing using dictionary learning methods and graph regularization. We have focussed on applications in seismic image denoising. However, the concept of the proposed method based on learning an adaptive dictionary from the given data strongly differs from direct denoising methods that apply either local smoothing procedures or smoothing procedures based on space frequency filtering processes. The learned dictionary enables us to obtain a sparse data representation that can be used also for other imaging problems.

For denoising of seismic images, we have employed here the given noisy data to extract training patches for dictionary learning. This procedure made a iterative method necessary, where the dictionary is improved using only 2 iterations in Algorithm 1. Iterations can be completely avoided if the dictionary is learned from clean seismic images possessing structures that are similar to the significant structures of the data to be processed. Another possible alternative may be a hybrid method that uses e.g. FX-Decon at a first stage for denoising and employs the denoised data patches to learn a data-driven dictionary at the second stage.

Experimental results achieved on field seismic data show that our proposed method connecting graph regularization and dictionary learning provides improved denoising results compared to methods that only employ a learned dictionary. Incorporating the graph regularization term however requires a higher effort to solve the minimization problem at each iteration step. We have used here a split Bregman method to minimize the occurring functional which is more time consuming than orthogonal matching pursuit which can be applied for the simplified functional without graph regularization term. Still, the recovery for stronger noise levels is not satisfactory. Future work will focus on improving the computational speed of the algorithms and on a more efficient choice of optimal parameters. Furthermore, we will study this approach for other geophysical inversion problems such as migration, imaging, and full waveform inversion.

ACKNOWLEDGEMENTS

This work is supported by National Nature Science Foundation of China (grant number: NSFC 91330108, 41374121, 61327013), and the Fundamental Research Funds for the Central Universities (grant number: HIT.PIRS.A201501).

APPENDIX A

We introduce the mathematical details of the split Bregman algorithm to derive Algorithm 2, which is used to solve the fourth step of the proposed denoising Algorithm 1. Let's recall the problem in the fourth step of Algorithm 1,

$$\min_{\mathbf{X}\in\mathbb{R}^{k\times m}} \left(\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 + \alpha\,\mathrm{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T) + \lambda\|\mathbf{X}\|_1\right). \tag{13}$$

First, we introduce a further variable $\mathbf{Z}\in\mathbb{R}^{k\times m}$ and rewrite the problem as follows,

$$\min_{\mathbf{X},\mathbf{Z}\in\mathbb{R}^{k\times m}} \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 + \alpha\,\mathrm{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T) + \mu\|\mathbf{X}-\mathbf{Z}\|_F^2 + \lambda\|\mathbf{Z}\|_1 \tag{14}$$

with some fixed parameter $\mu > 0$. Now, let

$$E(\mathbf{X},\mathbf{Z}) := \lambda\|\mathbf{Z}\|_1 + \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 + \alpha\,\mathrm{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T),$$

such that (14) is of the form

$$\min_{\mathbf{X},\mathbf{Z}\in\mathbb{R}^{k\times m}} E(\mathbf{X},\mathbf{Z}) + \mu\|\mathbf{X}-\mathbf{Z}\|_F^2.$$

We introduce the so-called Bregman distance

$$D_E((\mathbf{X},\mathbf{Z}),(\mathbf{X}^\ell,\mathbf{Z}^\ell)) := E(\mathbf{X},\mathbf{Z}) - E(\mathbf{X}^\ell,\mathbf{Z}^\ell) - \langle \mathbf{P}^\ell, \mathbf{X}-\mathbf{X}^\ell\rangle - \langle \mathbf{Q}^\ell, \mathbf{Z}-\mathbf{Z}^\ell\rangle,$$

where $(\mathbf{P}^\ell,\mathbf{Q}^\ell)$ is a subgradient of E at $(\mathbf{X}^\ell,\mathbf{Z}^\ell)$, i.e., $\mathbf{P}^\ell \in \partial_{\mathbf{X}}E(\mathbf{X}^\ell,\mathbf{Z}^\ell)$ and $\mathbf{Q}^\ell \in \partial_{\mathbf{Z}}E(\mathbf{X}^\ell,\mathbf{Z}^\ell)$, where $\partial_{\mathbf{X}}E(\mathbf{X}^\ell,\mathbf{Z}^\ell)$ and $\partial_{\mathbf{Z}}E(\mathbf{X}^\ell,\mathbf{Z}^\ell)$ denote the subdifferentials of E at $(\mathbf{X}^\ell,\mathbf{Z}^\ell)$ with respect to $\mathbf{X}$ and $\mathbf{Z}$. Now, to solve (14), we replace $E(\mathbf{X},\mathbf{Z})$ by the Bregman distance and consider the iteration

$$\begin{aligned}(\mathbf{X}^{\ell+1},\mathbf{Z}^{\ell+1}) &= \operatorname*{argmin}_{\mathbf{X},\mathbf{Z}\in\mathbb{R}^{k\times m}} \left\{D_E((\mathbf{X},\mathbf{Z}),(\mathbf{X}^\ell,\mathbf{Z}^\ell)) + \mu\|\mathbf{X}-\mathbf{Z}\|_F^2\right\} \\ &= \operatorname*{argmin}_{\mathbf{X},\mathbf{Z}\in\mathbb{R}^{k\times m}} \left\{E(\mathbf{X},\mathbf{Z}) - \langle \mathbf{P}^\ell,\mathbf{X}\rangle - \langle \mathbf{Q}^\ell,\mathbf{Z}\rangle + \mu\|\mathbf{X}-\mathbf{Z}\|_F^2\right\}. \end{aligned} \tag{15}$$

This iteration is senseful, since the two additional terms $\langle \mathbf{P}^\ell, \mathbf{X}\rangle$ and $\langle \mathbf{Q}^\ell, \mathbf{Z}\rangle$ vanish for the desired solution $(\mathbf{X}, \mathbf{Z})$ of (14). From (15) we derive the necessary conditions

$$\begin{aligned} 0 &\in \partial_{\mathbf{X}} E(\mathbf{X}^{\ell+1}, \mathbf{Z}^{\ell+1}) - \mathbf{P}^\ell + 2\mu(\mathbf{X}^{\ell+1} - \mathbf{Z}^{\ell+1}), \\ 0 &\in \partial_{\mathbf{Z}} E(\mathbf{X}^{\ell+1}, \mathbf{Z}^{\ell+1}) - \mathbf{Q}^\ell - 2\mu(\mathbf{X}^{\ell+1} - \mathbf{Z}^{\ell+1}). \end{aligned}$$

Since $(\mathbf{P}^{\ell+1}, \mathbf{Q}^{\ell+1}) \in \partial E(\mathbf{X}^{\ell+1}, \mathbf{Z}^{\ell+1})$ we obtain the following recursions from these conditions,

$$\mathbf{P}^{\ell+1} = \mathbf{P}^\ell - 2\mu(\mathbf{X}^{\ell+1} - \mathbf{Z}^{\ell+1}), \qquad \mathbf{Q}^{\ell+1} = \mathbf{Q}^\ell + 2\mu(\mathbf{X}^{\ell+1} - \mathbf{Z}^{\ell+1}), \tag{16}$$

and particularly, $\mathbf{P}^{\ell+1} + \mathbf{Q}^{\ell+1} = \mathbf{P}^\ell + \mathbf{Q}^\ell$ for all ℓ. Now, introducing $\mathbf{B}^\ell := \frac{1}{2\mu}\mathbf{Q}^\ell$, the second equation in (16) implies the recursion

$$\mathbf{B}^{\ell+1} = \mathbf{B}^\ell - \mathbf{Z}^{\ell+1} + \mathbf{X}^{\ell+1}.$$

Moreover, by

$$\mu\|\mathbf{X} - \mathbf{Z} + \mathbf{B}^\ell\|_F^2 = \mu\|\mathbf{X} - \mathbf{Z}\|_F^2 + 2\mu\langle \mathbf{X} - \mathbf{Z}, \mathbf{B}^\ell\rangle + \mu\|\mathbf{B}^\ell\|_F^2$$

we can rewrite (15) as

$$\begin{aligned} (\mathbf{X}^{\ell+1}, \mathbf{Z}^{\ell+1}) &= \underset{\mathbf{X},\mathbf{Z}\in\mathbb{R}^{k\times m}}{\operatorname{argmin}} \Big\{ E(\mathbf{X}, \mathbf{Z}) + \mu\|\mathbf{X} - \mathbf{Z} + \mathbf{B}^\ell\|_F^2 - 2\mu\langle \mathbf{X} - \mathbf{Z}, \mathbf{B}^\ell\rangle - \langle \mathbf{P}^\ell, \mathbf{X}\rangle - \langle \mathbf{Q}^\ell, \mathbf{Z}\rangle \Big\} \\ &= \underset{\mathbf{X},\mathbf{Z}\in\mathbb{R}^{k\times m}}{\operatorname{argmin}} \left\{ E(\mathbf{X}, \mathbf{Z}) + \mu\|\mathbf{X} - \mathbf{Z} + \mathbf{B}^\ell\|_F^2 - \langle \mathbf{X}, \mathbf{P}^\ell + \mathbf{Q}^\ell\rangle \right\}, \end{aligned}$$

where we have used that $-2\mu\langle \mathbf{X} - \mathbf{Z}, \mathbf{B}^\ell\rangle = -\langle \mathbf{Q}^\ell, \mathbf{X}\rangle + \langle \mathbf{Q}^\ell, \mathbf{Z}\rangle$. Taking the initial matrices $\mathbf{B}^0 = \mathbf{P}^0 = \mathbf{Q}^0 = \mathbf{0}$, it follows that $\mathbf{P}^\ell + \mathbf{Q}^\ell = \mathbf{0}$ for all ℓ, and we arrive at the iteration

$$\begin{aligned} (\mathbf{X}^{\ell+1}, \mathbf{Z}^{\ell+1}) &= \underset{\mathbf{X},\mathbf{Z}\in\mathbb{R}^{k\times m}}{\operatorname{argmin}} \left\{ E(\mathbf{X}, \mathbf{Z}) + \mu\|\mathbf{X} - \mathbf{Z} + \mathbf{B}^\ell\|_F^2 \right\}, \\ \mathbf{B}^{\ell+1} &= \mathbf{B}^\ell - \mathbf{Z}^{\ell+1} + \mathbf{X}^{\ell+1}. \end{aligned}$$

Applying alternating minimization, this leads to the following iteration scheme,

$$\begin{aligned} \mathbf{X}^{\ell+1} &= \underset{\mathbf{X}\in\mathbb{R}^{k\times m}}{\operatorname{argmin}} \left\{ \|\mathbf{D}\mathbf{X} - \mathbf{Y}\|_F^2 + \alpha\operatorname{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T) + \mu\|\mathbf{X} - \mathbf{Z}^\ell + \mathbf{B}^\ell\|_F^2 \right\}, \\ \mathbf{Z}^{\ell+1} &= \underset{\mathbf{Z}\in\mathbb{R}^{k\times m}}{\operatorname{argmin}} \left\{ \lambda\|\mathbf{Z}\|_1 + \mu\|\mathbf{X}^{\ell+1} - \mathbf{Z} + \mathbf{B}^\ell\|_F^2 \right\}, \\ \mathbf{B}^{\ell+1} &= \mathbf{B}^\ell - \mathbf{Z}^{\ell+1} + \mathbf{X}^{\ell+1}. \end{aligned} \tag{17}$$

The functional in the first equation of (17) is differentiable, and we obtain the necessary condition

$$\mathbf{D}^T(\mathbf{D}\mathbf{X}-\mathbf{Y})+\alpha\mathbf{X}\mathbf{L}+\mu(\mathbf{X}-\mathbf{Z}^\ell+\mathbf{B}^\ell)=\mathbf{0},$$

i.e.,

$$(\mathbf{D}^T\mathbf{D}+\mu\mathbf{I})\mathbf{X}+\alpha\mathbf{X}\mathbf{L}=\mathbf{D}^T\mathbf{Y}+\mu(\mathbf{Z}^\ell-\mathbf{B}^\ell).$$

This equation is uniquely solvable if the eigenvalues $\alpha_1,\dots,\alpha_k$ of $(\mathbf{D}^T\mathbf{D}+\mu\mathbf{I})$ and the eigenvalues $\beta_1,\dots,\beta_k$ of $\alpha\mathbf{L}$ satisfy

$$\alpha_i+\beta_j\neq 0 \qquad \forall\, i=1,\dots,k,\ j=1,\dots,m,$$

see Bartels and Stewart, 1972 and Bhatia and Rosenthal (1997). This is true in our case since $(\mathbf{D}^T\mathbf{D}+\mu\mathbf{I})$ is positive definite for $\mu>0$ and $\mathbf{L}$ is positive semidefinit.

The second subproblem in (17) can be simply solved by component-wise shrinkage. For each single component $z_{i,j}^{\ell+1}$ of $\mathbf{Z}^{\ell+1}=(z_{i,j}^{\ell+1})_{i=1,j=1}^{k,m}$ we have

$$z_{i,j}^{\ell+1}=\underset{z\in\mathbb{R}}{\operatorname{argmin}}\,\{\lambda\,|z|+\mu|z-x_{i,j}^{\ell+1}-B_{i,j}^\ell|^2\}$$

i.e.,

$$0\in\lambda\,\frac{z^{\ell+1}}{|z^{\ell+1}|}+2\mu\left(z^{\ell+1}-x_{i,j}^{\ell+1}-B_{i,j}^\ell\right),$$

where $\frac{z}{|z|}$ denotes the set $[-1,1]$ if $z=0$. Hence, we find the solution by soft shrinkage,

$$z_{i,j}^{\ell+1}=\mathcal{T}_{\lambda/2\mu}(x_{i,j}^{\ell+1}+B_{i,j}^\ell):=\begin{cases} x_{i,j}^{\ell+1}+B_{i,j}^\ell-\frac{\lambda}{2\mu} & \text{for } (x_{i,j}^{\ell+1}+B_{i,j}^\ell)\geq\frac{\lambda}{2\mu},\\ x_{i,j}^{\ell+1}+B_{i,j}^\ell+\frac{\lambda}{2\mu} & \text{for } (x_{i,j}^{\ell+1}+B_{i,j}^\ell)\leq-\frac{\lambda}{2\mu},\\ 0 & \text{otherwise.}\end{cases}$$

APPENDIX B

In our paper, we have proposed two dictionary learning algorithms, called FDC and SDC. In order to illustrate the dictionary learning step more clearly, we will give a small toy example, where the training set contains the following 10 training patches of size 3×3,

$$\mathbf{I_1} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \mathbf{I_2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \mathbf{I_3} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{I_4} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{I_5} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \mathbf{I_6} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad \mathbf{I_7} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{I_8} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{I_9} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{I_{10}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

We construct the partition trees using the two algorithms in section METHODS FOR DICTIONARY LEARNING and show, how the dictionary construction can be directly incorporated into the partition tree construction. The example also shows that the trees obtained and thus the dictionary elements found slightly differ for the two approaches. For the explicit computations we round to 2 digits in matrix entries and to 4 digits for entries of singular values and singular vectors/eigenvectors.

FDC algorithm

First layer: We start with the training set $\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_{10}$. We compute the mean (center) matrix

$$\mathbf{C}_1 = \frac{1}{10} \sum_{i=1}^{10} \mathbf{I}_i = \begin{pmatrix} 0.6 & 0.5 & 0.4 \\ 0.2 & 0.7 & 0.8 \\ 0 & 0.1 & 0.6 \end{pmatrix}$$

and the normalized eigenvectors to the maximal eigenvalue of

$$\mathbf{C}_1 \mathbf{C}_1^T = \begin{pmatrix} 0.77 & 0.79 & 0.29 \\ 0.79 & 1.17 & 0.55 \\ 0.29 & 0.55 & 0.37 \end{pmatrix} \quad \text{and} \quad \mathbf{C}_1^T \mathbf{C}_1 = \begin{pmatrix} 0.40 & 0.44 & 0.40 \\ 0.44 & 0.75 & 0.82 \\ 0.40 & 0.82 & 1.16 \end{pmatrix}.$$

We obtain

$$\mathbf{u}_1 = [-0.5590, -0.7516, -0.3501]^T, \qquad \mathbf{v}_1 = [-0.3423, -0.5924, -0.7293]^T,$$

which are the singular vectors of $\mathbf{C}_1$ to the singular value $\lambda_1 = 1.4191$. Thus we get the first dictionary element according to (7),

$$\mathbf{D}_1 = \mathbf{u}_1\mathbf{v}_1^T = \begin{pmatrix} 0.19 & 0.33 & 0.41 \\ 0.26 & 0.45 & 0.55 \\ 0.12 & 0.21 & 0.26 \end{pmatrix},$$

and $\lambda_1\mathbf{D}_1$ represents the optimal rank-1 approximation of $\mathbf{C}_1$.

Second layer: The training set is now divided into two partial sets according to the description for FDC tree partitioning. This can be done as follows:

1. We calculate the two non-symmetric covariance matrices $\mathbf{C}_L$ and $\mathbf{C}_R$ in (5),

$$\mathbf{C}_L = \begin{pmatrix} 0.73 & -0.19 & -0.90 \\ -0.19 & 0.53 & 0.05 \\ -0.09 & 0.05 & 0.33 \end{pmatrix}, \qquad \mathbf{C}_R = \begin{pmatrix} 0.40 & 0.16 & -0.10 \\ 0.16 & 0.55 & 0.08 \\ -0.10 & 0.08 & 0.64 \end{pmatrix}.$$

2. We compute the normalized eigenvectors corresponding to the maximal eigenvalue of $\mathbf{C}_L$ and $\mathbf{C}_R$, $\mathbf{u} = [-0.8415, 0.5062, 0.1890]^T$, $\mathbf{v} = [-0.590, 0.4508, 0.8907]^T$.

3: We compute the one-dimensional projection representations $s_i := \mathbf{u}^T\mathbf{I}_i\mathbf{v}$ of all patches, obtaining

$$[s_1, \cdots, s_{10}]^T = [0.12, 0.28, 0.37, 0.90, -0.40, 0.15, 0.52, -0.68, -0.46, 0.85]^T.$$

We order these numbers by size, $s_{\ell_1} \leq s_{\ell_2} \leq \ldots \leq s_{\ell_{10}}$, and find the ordered set

$$\begin{aligned} & [s_{\ell_1}, \ldots, s_{\ell_{10}}]^T = [s_8, s_9, s_5, s_1, s_6, s_2, s_3, s_7, s_{10}, s_4]^T \\ = \; & [-0.68, -0.46, -0.40, 0.12, 0.15, 0.28, 0.37, 0.52, 0.85, 0.90]^T. \end{aligned}$$

4: We compute $\hat{\kappa}$ according to (6) and obtain here $\hat{\kappa} = 3$. Thus, the patches with the indices 8, 9, 5 corresponding the first three numbers s_8, s_9, s_5 in our ordered set are put into the first subset, and all other patches are put into the second subset. In this way the partition of index sets $\Lambda_1 = \Lambda_2 \cup \Lambda_3$ with $\Lambda_2 = \{5, 8, 9\}$, $\Lambda_3 = \{1, 2, 3, 4, 6, 7, 10\}$ is derived. The center matrices of the obtained two sets of patches are

$$\mathbf{C}_2 = \frac{1}{3}(\mathbf{I}_5 + \mathbf{I}_8 + \mathbf{I}_9), \quad \mathbf{C}_3 = \frac{1}{7}(\mathbf{I}_1 + \mathbf{I}_2 + \mathbf{I}_3 + \mathbf{I}_4 + \mathbf{I}_6 + \mathbf{I}_7 + \mathbf{I}_{10}).$$

We compute the maximal singular values $\lambda_2 = 1.8259$, $\lambda_3 = 1.3648$ of the center matrices $\mathbf{C}_2$, $\mathbf{C}_3$ and the first left and right singular vectors $\mathbf{u}_2$, $\mathbf{v}_2$ and $\mathbf{u}_3$, $\mathbf{v}_3$ of $\mathbf{C}_2$, $\mathbf{C}_3$, respectively. Then, from the formula (8) we obtain the second dictionary element $\mathbf{D}_2 = \frac{\tilde{\mathbf{D}}_2}{\|\tilde{\mathbf{D}}_2\|_F}$ with

$$\tilde{\mathbf{D}}_2 = \lambda_2 \mathbf{u}_2 \mathbf{v}_2^T - \lambda_3 \mathbf{u}_3 \mathbf{v}_3^T = \begin{pmatrix} 0.55 & 1.48 & 1.83 \\ 0.27 & 0.31 & 0.51 \\ 0.04 & -0.14 & -0.09 \end{pmatrix}.$$

Now, $\tilde{\mathbf{D}}_2$ is a rank-2 approximation of $\mathbf{C}_2 - \mathbf{C}_3$ and therefore represents the difference between main structures of $\mathbf{C}_2$ and $\mathbf{C}_3$ while the common significant structure is already captured by $\mathbf{D}_1$. Indeed, since $\lambda_1 \mathbf{D}_1$ approximates $\mathbf{C}_1 = \frac{3}{10}\mathbf{C}_2 + \frac{7}{10}\mathbf{C}_3$, the main structures of $\mathbf{C}_2$ and $\mathbf{C}_3$ can be well presented by $\mathbf{D}_1$ and $\mathbf{D}_2$.

Third layer: The partition procedure yields now the subsets of indices $\Lambda_4 = \{8\}$, $\Lambda_5 = \{5, 9\}$ of Λ_2, and the subsets $\Lambda_6 = \{3, 6\}$, $\Lambda_7 = \{1, 2, 4, 7, 10\}$ of Λ_3, see also Figure 3. We compute the center matrices of the corresponding partial sets of patches,

$$\mathbf{C}_4 = \mathbf{I}_8, \ \mathbf{C}_5 = \frac{1}{2}(\mathbf{I}_5 + \mathbf{I}_9), \ \mathbf{C}_6 = \frac{1}{2}(\mathbf{I}_3 + \mathbf{I}_6), \ \mathbf{C}_7 = \frac{1}{5}(\mathbf{I}_1 + \mathbf{I}_2 + \mathbf{I}_4 + \mathbf{I}_7 + \mathbf{I}_{10}).$$

We get the new dictionary element $\mathbf{D}_3 = \frac{\tilde{\mathbf{D}}_3}{\|\tilde{\mathbf{D}}_3\|_F}$ with

$$\tilde{\mathbf{D}}_3 = \lambda_4 \mathbf{u}_4 \mathbf{v}_4^T - \lambda_5 \mathbf{u}_5 \mathbf{v}_5^T = \begin{pmatrix} -0.72 & -0.21 & -0.05 \\ -0.42 & -0.10 & 0.01 \\ -0.15 & -0.20 & -0.26 \end{pmatrix},$$

where $\lambda_4 = 1.6180$ and $\lambda_5 = 1.9966$ are maximal singular values of the center matrices $\mathbf{C}_4$ and $\mathbf{C}_5$, and $\mathbf{u}_4$, $\mathbf{v}_4$ and $\mathbf{u}_5, \mathbf{v}_5$ are first singular vectors of $\mathbf{C}_4$, $\mathbf{C}_5$. Similarly, $\mathbf{D}_4 = \frac{\tilde{\mathbf{D}}_4}{\|\tilde{\mathbf{D}}_4\|_F}$

with

$$\tilde{\mathbf{D}}_4 = \lambda_6 \mathbf{u}_6 \mathbf{v}_6^T - \lambda_7 \mathbf{u}_7 \mathbf{v}_7^T = \begin{pmatrix} 0.05 & -0.10 & -0.13 \\ 0.44 & 0.17 & 0.07 \\ 0.35 & 0.30 & 0.25 \end{pmatrix},$$

where $\lambda_6 = 1.7184$ and $\lambda_7 = 1.2630$ are maximal singular values of center matrices $\mathbf{C}_6$ and $\mathbf{C}_7$ with corresponding singular vectors $\mathbf{u}_6$, $\mathbf{v}_6$, $\mathbf{u}_7$, $\mathbf{v}_7$. Since $\tilde{\mathbf{D}}_3$ is now a good approximation of $\mathbf{C}_4 - \mathbf{C}_5 = \mathbf{I}_8 - \mathbf{C}_5$, we can obtain a sparse approximation of $\mathbf{I}_8$ already using the dictionary elements $\mathbf{D}_1$, $\mathbf{D}_2$ and $\mathbf{D}_3$.

Fourth layer: The index subset Λ_5 is split into $\Lambda_8 = \{5\}$, $\Lambda_9 = \{9\}$, Λ_6 is split into $\Lambda_{10} = \{3\}$, $\Lambda_{11} = \{6\}$; and Λ_7 is split into $\Lambda_{12} = \{1, 7\}$ and $\Lambda_{13} = \{2, 4, 10\}$. Then $\mathbf{D}_5$, $\mathbf{D}_6$ and $\mathbf{D}_7$ can be learned from $\{\Lambda_8, \Lambda_9\}$, $\{\Lambda_{10}, \Lambda_{11}\}$, and $\{\Lambda_{12}, \Lambda_{13}\}$.

Fifth layer: The index subset Λ_{12} is split into $\Lambda_{14} = \{1\}$ and $\Lambda_{15} = \{7\}$, Λ_{13} is split into $\Lambda_{16} = \{2\}$ and $\Lambda_{17} = \{4, 10\}$, and we learn the dictionary elements $\mathbf{D}_8$ and $\mathbf{D}_9$.

Sixth layer: Finally, index subset Λ_{17} is split into $\Lambda_{18} = \{4\}$ and $\Lambda_{19} = \{10\}$, and we learn the dictionary element $\mathbf{D}_{10}$. The complete partition tree is given in Figure 3(a).

Remark. Please keep in mind that we have used here always rank-1 approximations of the center matrices since we assume that the training data are noisy in applications. If the training data are exact (as it is the case in this toy example), then we may use the center matrices themselves instead of their rank-1 approximations, getting the dictionary

$$\mathbf{D}_1 = \frac{\mathbf{C}_1}{\|\mathbf{C}_1\|_F}, \quad \mathbf{D}_2 = \frac{\mathbf{C}_2 - \mathbf{C}_3}{\|\mathbf{C}_2 - \mathbf{C}_3\|_F}, \quad \mathbf{D}_3 = \frac{\mathbf{C}_4 - \mathbf{C}_5}{\|\mathbf{C}_4 - \mathbf{C}_5\|_F}, \quad \mathbf{D}_4 = \frac{\mathbf{C}_6 - \mathbf{C}_7}{\|\mathbf{C}_6 - \mathbf{C}_7\|_F},$$

etc., and have for example already an exact sparse representation of $\mathbf{I}_8 = \mathbf{C}_4$ as a linear combination of $\mathbf{D}_1$, $\mathbf{D}_2$ and $\mathbf{D}_3$. Indeed, we find

$$\mathbf{I}_8 = \mathbf{C}_4 = \mathbf{C}_1 + \frac{7}{10}\left(\mathbf{C}_2 - \mathbf{C}_3\right) + \frac{2}{3}\left(\mathbf{C}_4 - \mathbf{C}_5\right).$$

Applying the SDC algorithm, we obtain a different partition tree, see Figure 3(b), where

$$\begin{aligned}
\Lambda_1 &= \{1,\dots,10\} \\
\Lambda_2 &= \{1,2,3,4,5,7,10\}, \quad \Lambda_3 = \{6,8,9\}, \\
\Lambda_4 &= \{2\}, \quad \Lambda_5 = \{1,3,4,5,7,10\}, \quad \Lambda_6 = \{8,9\}, \quad \Lambda_7 = \{6\} \\
\Lambda_8 &= \{1,4,5,7\}, \quad \Lambda_9 = \{3,10\}, \quad \Lambda_{10} = \{8\}, \quad \Lambda_{11} = \{9\} \\
\Lambda_{12} &= \{1\}, \quad \Lambda_{13} = \{4,5,7\}, \quad \Lambda_{14} = \{10\}, \quad \Lambda_{15} = \{3\}, \\
\Lambda_{16} &= \{4,5\}, \quad \Lambda_{17} = \{7\}, \\
\Lambda_{18} &= \{4\}, \quad \Lambda_{19} = \{5\}.
\end{aligned}$$

Observe that this approach provides a different partition tree, also the number of layers and the number of dictionary elements is different.

REFERENCES

Aharon, M., M. Elad, and A. Bruckstein, 2006, The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation: IEEE Transactions on Signal Processing, **54**, no. 11, 4311-4322, doi: 10.1109/TSP.2006.881199.

Anagaw, A. Y., and MD. Sacchi, 2012, Edge-preserving seismic imaging using the total variation method: Journal of Geophysics and Engineering, **9**, no. 2, 138-146.

Bartels, R.H., and G.W. Stewart, 1972, Solution of the matrix equation $AX + XB = C$: Commun. ACM **15**, no. 9, 820-826.

Bechouche, S., and J. Ma, 2014, Simultaneously dictionary learning and denoising for seismic data: Geophysics, **79**, A27-A31, doi: 10.1190/geo2013-0382.1.

Belkin, M., and P. Niyogi, 2003, Laplacian eigenmaps for dimensionality reduction and data representation: Neural Computation, **15**, no. 6, 1373-1396, doi:10.1162/089976603321780317.

Beck, A., and M. Teboulle, 2009, A fast iterative shrinkage-thresholding algorithm for linear inverse problems: SIAM Journal on Imaging Sciences, **2**, no. 1, 183-202, doi:10.1137/080716542.

Bhatia, R., and P. Rosenthal, 1997, How and why to solve the operator equation $AX+XB=Y$: Bull. London Math. Soc., **29**, no. 1, 1-21.

Bougleux, S., A. Elmoataz, and M. Melkemi, 2009, Local and nonlocal discrete regularization on weighted graphs for image and mesh processing: International Journal of Computer Vision, **84**, no. 2, 220-236, doi:10.1007/s11263-008-0159-z.

Bonar, D., and M. Sacchi, 2012, Denoising seismic data using the nonlocal means algorithm: Geophysics, **77**, no. 1, A5–A8, doi: 10.1190/geo2011-0235.1.

Cai, J. F., S. Huang, H. Ji, Z. Shen, and G. Ye, 2014, Data-driven tight frame construction and image denoising: Applied and Computational Harmonic Analysis, **37**, no. 1, 89-105.

Canales, L. L., 1984, Random noise reduction: Expanded Abstracts of Annual Meeting, Society of Exploration Geophysicists, 525-527, doi: 10.1190/1.1894168.

Chanerley, A. A., and N. A. Alexander, 2002, An approach to seismic correction which includes wavelet de-noising: Proceedings of the Sixth Conference on Computational Structures Technology, pp. 107-108.

Chambolle, A., and T. Pock, 2011, A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging: Journal of Mathematical Imaging and Vision **40**, no. 1, 120-145, doi:0.1007/s10851-010-0251-1.

Ding, C., and X. He, 2004, K-means clustering via principal component analysis: Proceedings of the Twenty-First International Conference on Machine Learning, 29, doi:10.1145/1015330.1015408.

Dong, W., L. Zhang, G. Shi, and X. Li, 2013, Nonlocally centralized sparse representation for image restoration: IEEE Transactions on Image Processing, **22**, no. 4, 1620-1630, doi:10.1109/TIP.2012.2235847.

Elad, M., and M. Aharon, 2006, Image denoising via sparse redundant representations over learned dictionaries: IEEE Transactions on Image Processing, **15**, no. 12, 3736-3754, doi:10.1109/TIP.2006.881969.

Engan, K., S. O. Aase, and J. H. Husoy, 1999, Method of optimal directions for frame design: IEEE International Conference on Acoustics, Speech, and Signal Processing, **5**, doi: 10.1109/ICASSP.1999.760624.

Elmoataz, A., O. Lezoray, and S. Bougleux, 2008, Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing: IEEE Transactions on Image Processing, **17**, no. 7, 1047-1060, doi:10.1109/TIP.2008.924284.

Goldstein, T., and S. Osher, 2009, The split Bregman method for L_1-regularized problems: SIAM J. Imaging Sciences, **2**, no. 2, 323-343, doi:10.1137/080725891.

Gulunay, N., 1986, FXDECON and complex Wiener prediction filter: SEG Technical Program Expanded Abstracts, pp. 279-281, doi: org/10.1190/1.1893128.

Hein, M., and M. Maier, 2006, Manifold denoising: Advances in Neural Information Processing Systems, **19**, 561-568.

Hennenfent, G., and F. J. Herrmann, 2006, Seismic denoising with nonuniformly sampled curvelets: Computing in Science and Engineering, **8**, 16-25, doi:10.1109/MCSE.2006.49.

Hennenfent, G., L. Fenelon, and F. J. Herrmann, 2010, Nonequispaced curvelet transform for seismic data reconstruction: A sparsity-promoting approach: Geophysics, **75**, no.6, WB203-WB210, doi:org/10.1190/1.3494032.

Herrmann, F. J., and G. Hennenfent, 2008, Non-parametric seismic data recovery with curvelet frames: Geophysical Journal International, **173**, no. 1, 233-248.

Horn, R. A, and C. R. Johnson, Topics in Matrix Analysis, Cambridge University Press, 1991.

Kheradmand, A., and P. Milanfar, 2014, A general framework for regularized, similarity-based image restoration: IEEE Transactions on Image Processing, **23**, no. 12, 5136-5151, doi:10.1109/TIP.2014.2362059.

Kaplan, S. T., M. D. Sacchi, and J. U. Tadeusz, 2009, Sparse coding for data-driven coherent and incoherent noise attenuation: SEG Technical Program Expanded Abstracts 2009, 3327-3331, doi:org/10.1190/1.3255551.

Lee, H., A. Battle, R. Raina, and A. Y. Ng, 2007, Efficient sparse coding algorithms:

Advances in Neural Information Processing Systems, **19**, 801-808.

Liu, X., D. Zhai, D. Zhao, G. Zhai, and W. Gao, 2014, Progressive image denoising through hybrid graph laplacian regularization: a unified framework: IEEE Transactions on Image Processing, **23**, no. 4, 1491-1503, doi:10.1109/TIP.2014.2303638.

Liu, L., G. Plonka, and J. Ma, 2017, Seismic data interpolation and denoising by learning a tensor tight frame: Inverse Problems, 33, 105011, doi: 10.1088/1361-6420/aa7773.

Lu, X., H. Yuan, P. Yan, Y. Yuan, and X. Li, 2012, Geometry constrained sparse coding for single image super-resolution: IEEE Conference on Computer Vision and Pattern Recognition, 1648-1655, doi10.1109/CVPR.2012.6247858.

Naghizadeh, M., 2010, A unified method for interpolation and de-noising of seismic records in the f-k domain: Expanded Abstracts of Annual Meeting, Society of Exploration Geophysicists, 3579-3583, doi: 10.1190/1.3513593.

Ma, J., and G. Plonka, 2010, The curvelet transform: IEEE Signal Processing Magazine, **27**, no.2, 118-133, doi: 10.1109/MSP.2009.935453.

Neelamani, R., A. Baumstein, D. Gillard, M. Hadidi, and W. Soroka, 2008, Coherent and random noise attenuation using the curvelet transform: The Leading Edge, **27**, no.2, 240–248, doi: 10.1190/1.2840373.

Needell, D., and R. Vershynin, 2010, Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit: IEEE Journal of Selected Topics in Signal Processing, **4**, no. 2, 310-316, doi:10.1109/JSTSP.2010.2042412.

Pan, Q., L. Zhang, G. Dai, and H. Zhang, 1999, Two denoising methods by wavelet transform: IEEE Transactions on Signal Processing, **47**, no. 12, 3401-3406, doi: 10.1109/78.806084.

Plonka, G., and J. Ma, 2011, Curvelet-wavelet regularized split Bregman iteration for compressed sensing, Int. J. Wavelets Multiresolut Inf. Process. **9**, no. 1, 79-110. doi:10.1142/S0219691311003955.

Reiter, DT., and W. Rodi, 1996, Nonlinear waveform tomography applied to crosshole seismic data: Geophysics, **61**, no. 3, 902-913, doi: 10.1190/1.1444015.

Starck, JL., EJ. Candes, and DL. Donoho, 2002, The curvelet transform for image denoising: IEEE Transactions on Image Processing, **11**, no. 6, 670-684, doi:10.1109/TIP.2002.1014998.

Tang, Y., Y. Shen, A. Jiang, N. Xu, and C. Zhu, 2013, Image denoising via graph regularized K-SVD: IEEE International Symposium on Circuits and Systems (ISCAS), 2820-2823, doi:10.1109/ISCAS.2013.6572465.

Yankelevsky, Y., and M. Elad, 2016, Dual graph regularized dictionary learning: IEEE Transactions on Signal and Information Processing over Networks, **2**, no. 4, 611-624, doi:10.1109/TSIPN.2016.2605763.

Yu, S., J. Ma, X. Zhang, and M. Sacchi, 2015, Denoising and interpolation of high-dimensional seismic data by learning tight frame: Geophysics, **80**, V119-V132, doi:10.1190/geo2014-0396.1.

Yu, S., J. Ma, and S. Osher, 2016, Monte Carlo data-driven tight frame for seismic data recovery: Geophysics, **81**, no. 4, V327-V340.

Zhang, R., and T. Ulrych, 2003, Physical wavelet frame denoising: Geophysics, **68**, 225-231, doi: 10.1190/1.1543209.

Zeng, X., W. Bian, W. Liu, J. Shen, and D. Tao, 2015, Dictionary pair learning on Grassmann manifolds for image denoising: IEEE Transactions on Image Processing, **24**, no. 11, 4556-4569, doi:10.1109/TIP.2015.2468172.

Zheng, M., J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, 2011, Graph regularized sparse coding for image representation. IEEE Transactions on Image Processing, **20**, no. 5, 1327-1336, doi:10.1109/TIP.2010.2090535.

Zoran, D., and Y. Weiss, 2011, From learning models of natural image patches to whole image restoration: Computer Vision (ICCV), 2011 IEEE International Conference on, 479-486, doi: 10.1109/ICCV.2011.6126278.

Figure Captions:

Figure 1: Example for a weighted and undirected graph, where with weight 1 for K-nearest neighbors.

Figure 2: The process for dictionary learning by FDC and SDC algorithm, where λ_i denotes the maximal singular value of center matrix $\mathbf{C}_i$ at each node, $\mathbf{u}_i$ and $\mathbf{v}_i$ are the first vectors in the singular value decomposition of $\mathbf{C}_i$.

Figure 3: Trees for (a) FDC and (b) SDC dictionary learning for the toy example in Appendix B.

Figure 4: Seismic Data: (a) field data one; (b) field data one with added random noise (noise standard deviation $\sigma = 25$); and (c) field data two.

Figure 5: Comparison of dictionaries learned through two methods: (a) FDC; (b) SDC.

Figure 6: Denoising results for field data with noise standard deviation $\sigma = 25$: (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.

Figure 7: Difference between denoising results and field data one in Figure 4(a): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.

Figure 8: Single trace comparison of the reconstructions of the field data one in Figure 4(a): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.

Figure 9: Denoising performance of FX-Decon, Curvelet, FDC-OMP, SDC-OMP, FDC-Graph and SDC-Graph algorithms for different noise standard deviations.

Figure 10: Denoising of field data two by FX-Decon (a), Curvelet (b), FDC-Graph (c) and SDC-Graph (d).

Figure 11: Difference between denoising results and field data two by FX-Decon (a), Curvelet (b), FDC-Graph (c) and SDC-Graph (d).

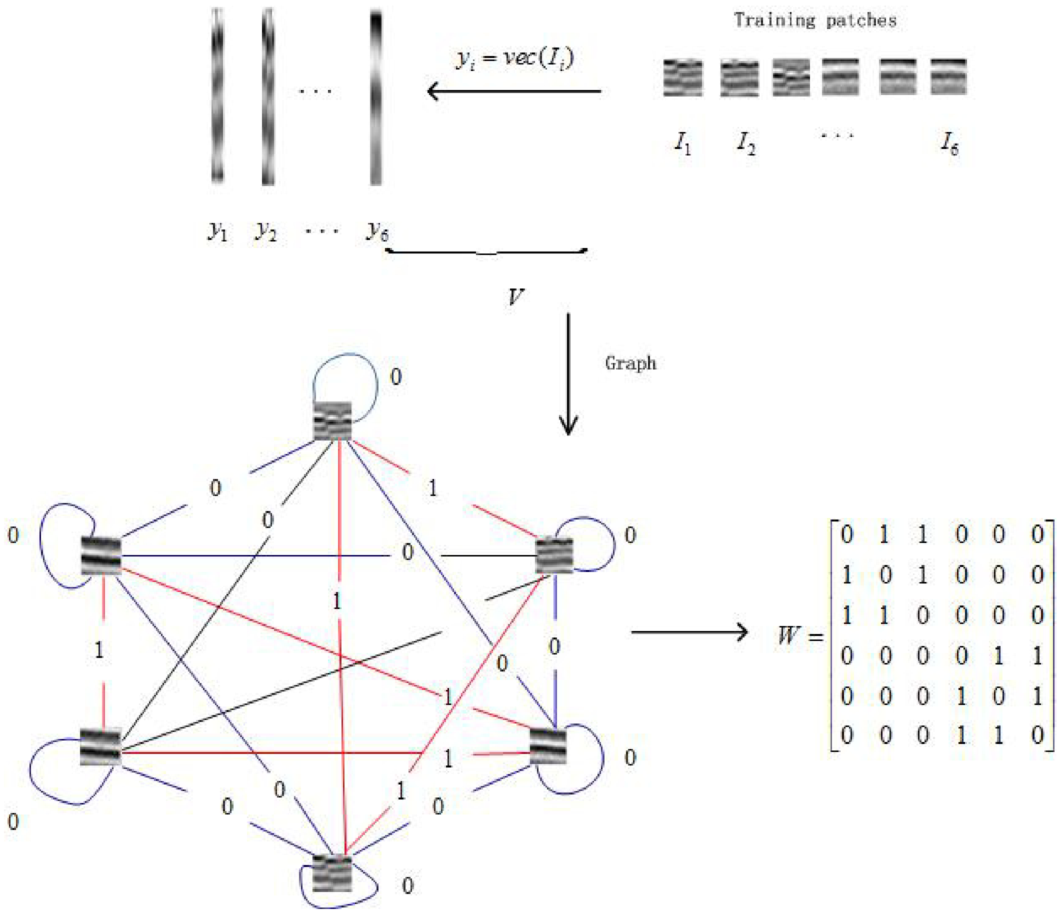

Training patches
$y_i = vec(I_i)$
I_1 I_2 ... I_6
y_1 y_2 ... y_6
V
Graph
$W = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$

Figure 1: Example for a weighted and undirected graph, where with weight 1 for K-nearest neighbors.

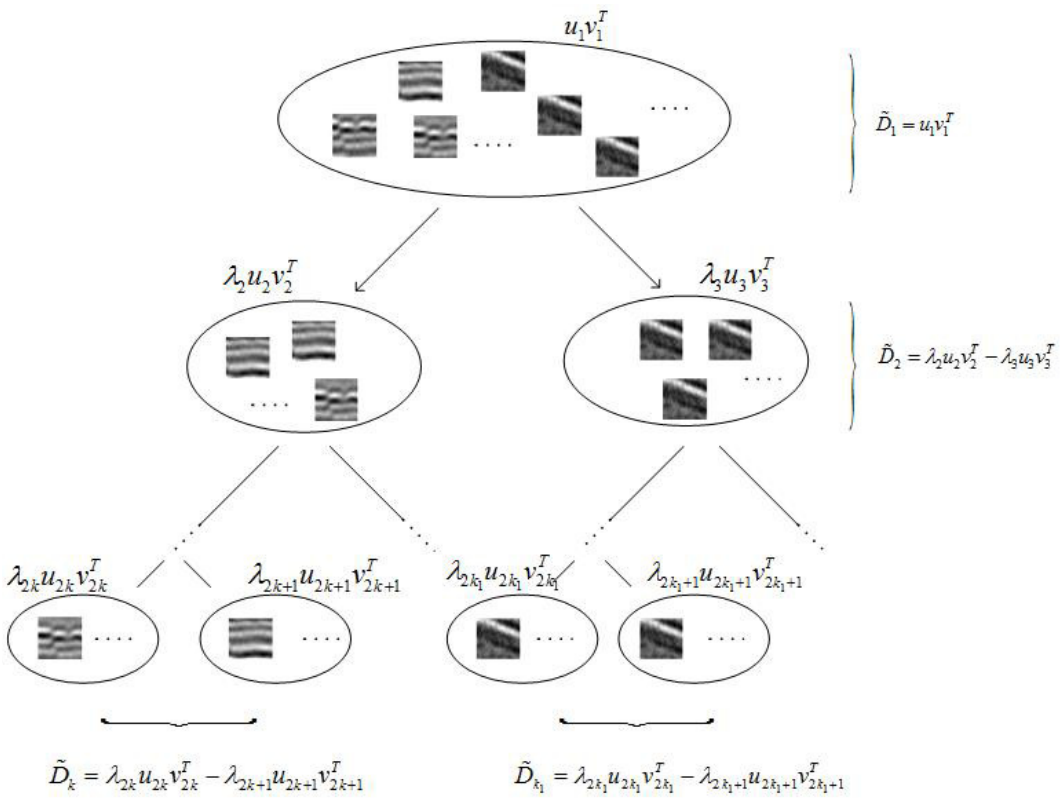
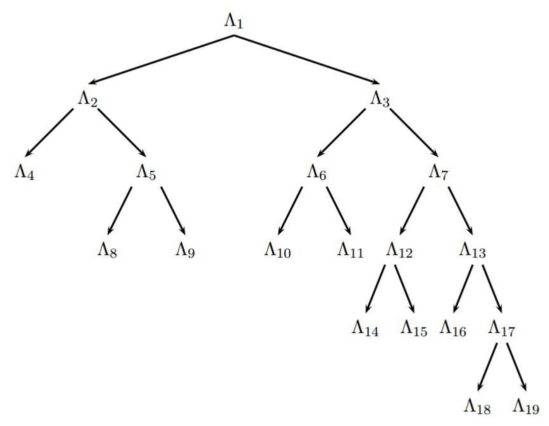
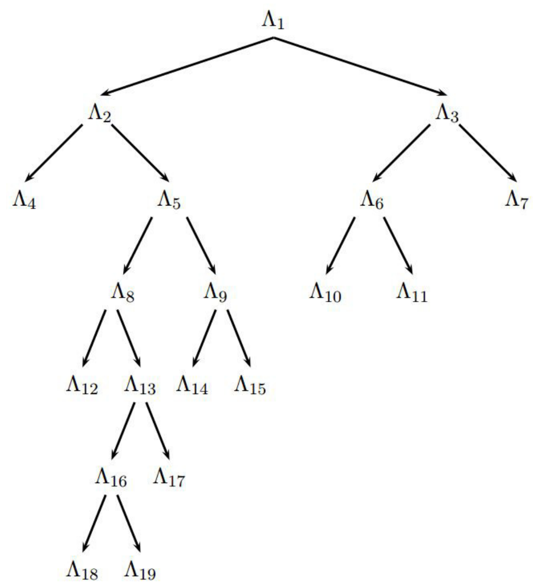

$u_1 v_1^T$
$\tilde{D}_1 = u_1 v_1^T$
$\lambda_2 u_2 v_2^T$
$\lambda_3 u_3 v_3^T$
$\tilde{D}_2 = \lambda_2 u_2 v_2^T - \lambda_3 u_3 v_3^T$
$\lambda_{2k} u_{2k} v_{2k}^T$
$\lambda_{2k+1} u_{2k+1} v_{2k+1}^T$
$\lambda_{2k_1} u_{2k_1} v_{2k_1}^T$
$\lambda_{2k_1+1} u_{2k_1+1} v_{2k_1+1}^T$
$\tilde{D}_k = \lambda_{2k} u_{2k} v_{2k}^T - \lambda_{2k+1} u_{2k+1} v_{2k+1}^T$
$\tilde{D}_{k_1} = \lambda_{2k_1} u_{2k_1} v_{2k_1}^T - \lambda_{2k_1+1} u_{2k_1+1} v_{2k_1+1}^T$

Figure 2: The process for dictionary learning by FDC and SDC algorithm, where λ_i denotes the maximal singular value of center matrix $\mathbf{C}_i$ at each node, $\mathbf{u}_i$ and $\mathbf{v}_i$ are the first vectors in the singular value decomposition of $\mathbf{C}_i$.

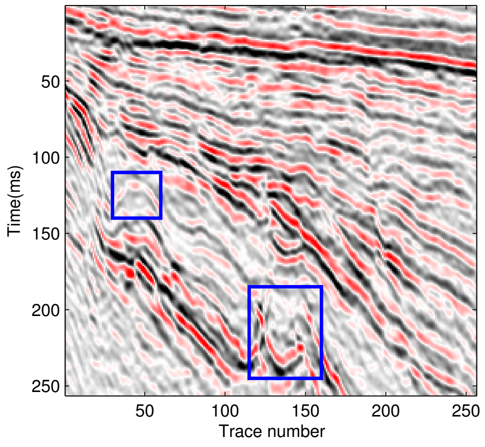


(a)

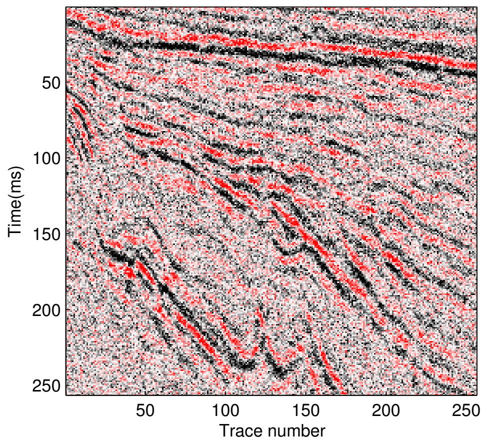


(b)

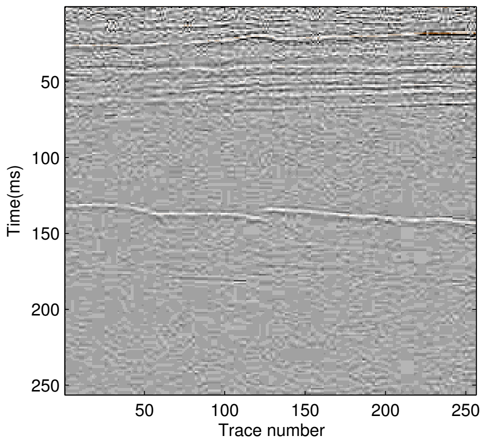
Figure 3: Trees for (a) FDC and (b) SDC dictionary learning for the toy example in Appendix B.



(a)

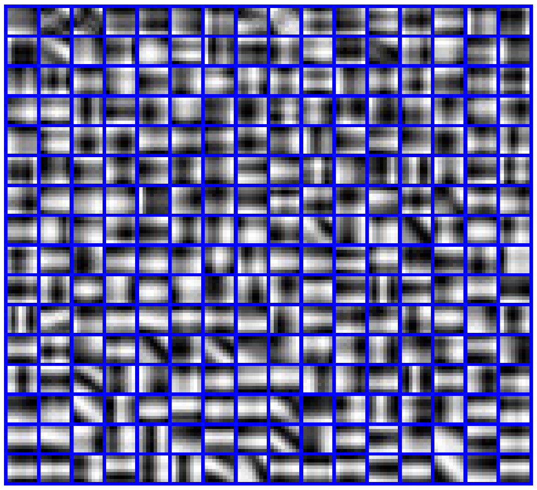


(b)

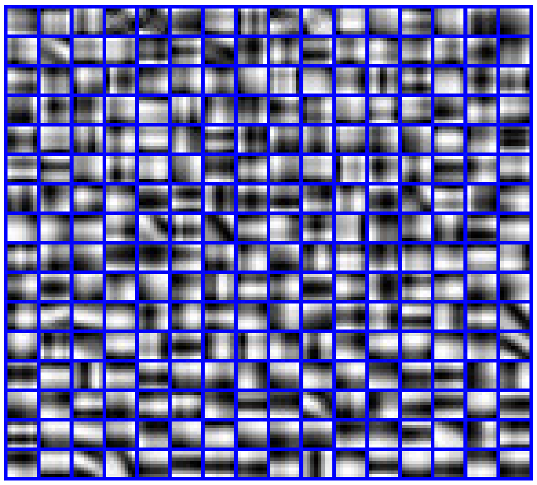


(c)

Figure 4: Seismic Data: (a) field data one; (b) field data one with added random noise (noise standard deviation $\sigma = 25$); and (c) field data two.

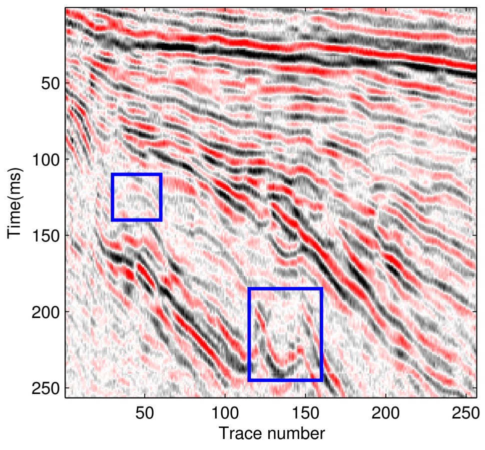


(a)

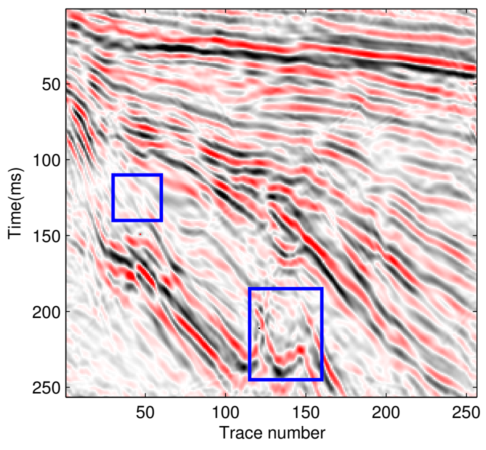


(b)

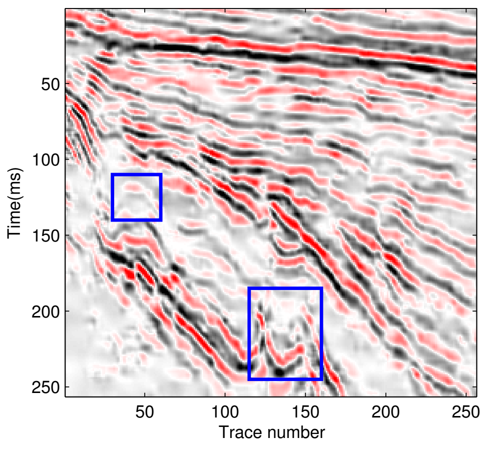
Figure 5:   Comparison of dictionaries learned through two methods: (a) FDC; (b) SDC.



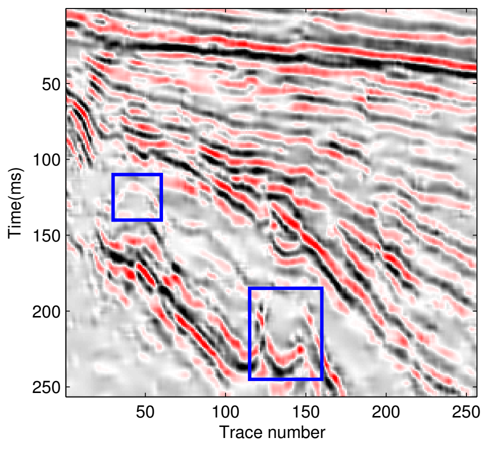
(a)



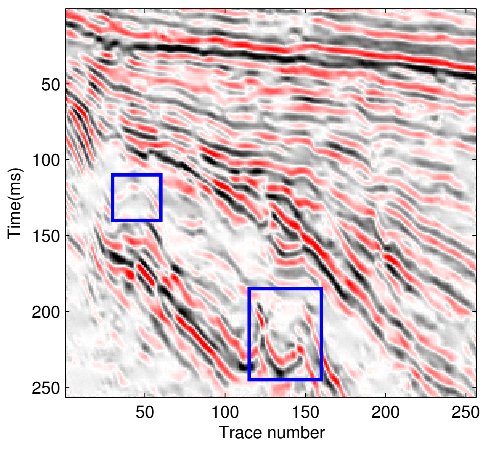
(b)



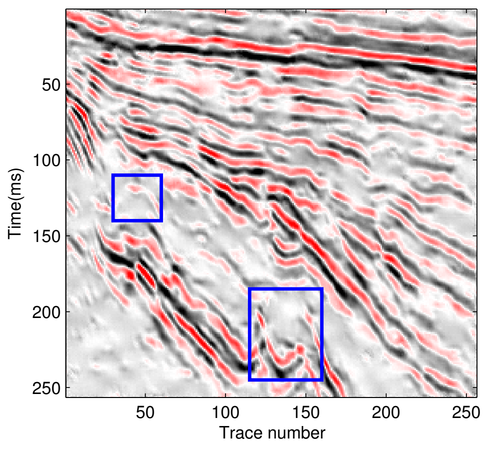
(c)



(d)

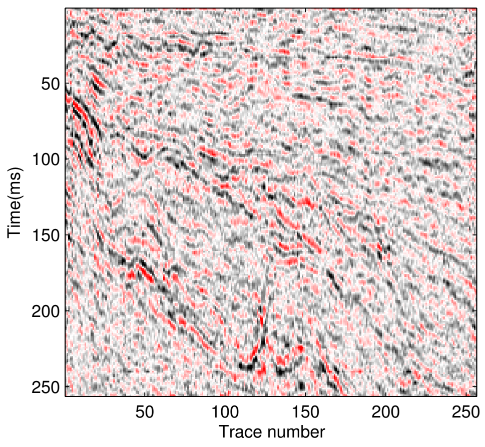


(e)

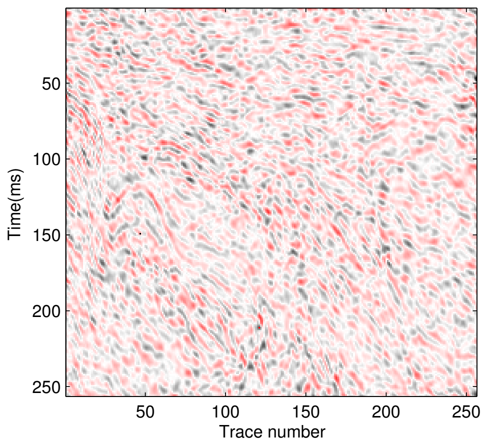


(f)

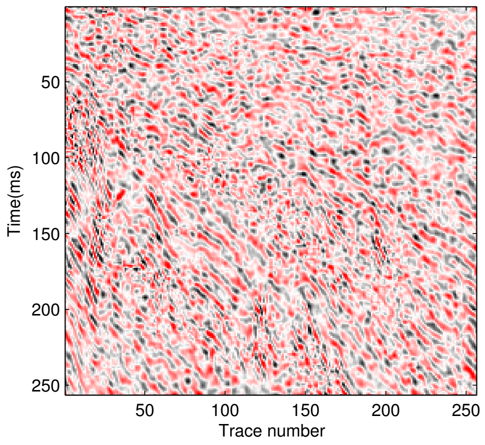
Figure 6: Denoising results for field data with noise standard deviation $\sigma = 25$: (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.



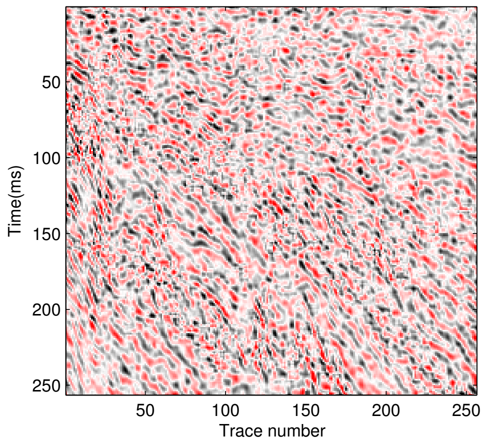
(a)



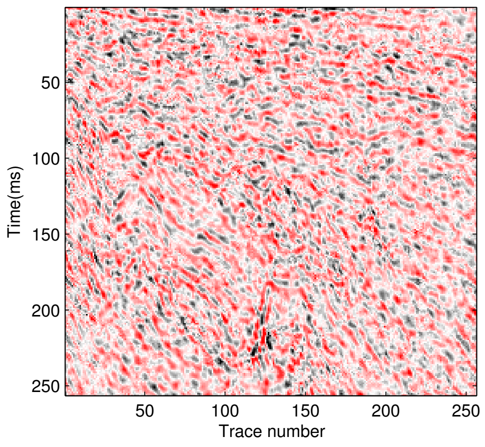
(b)



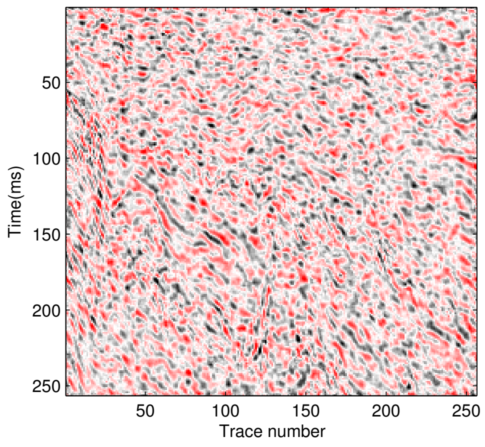
(c)



(d)

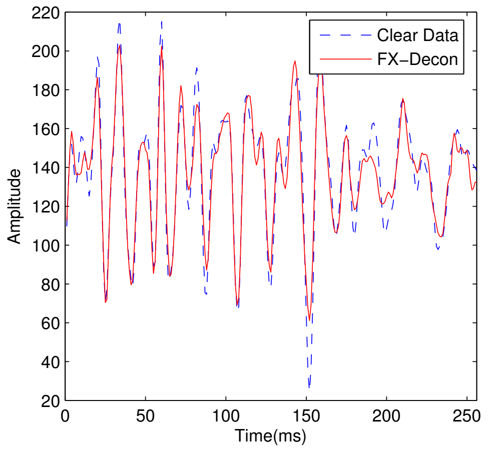


(e)

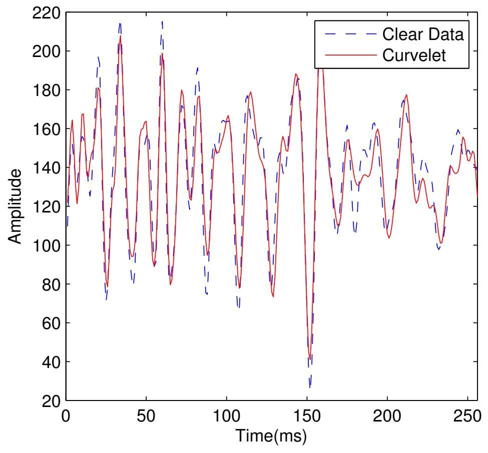


(f)

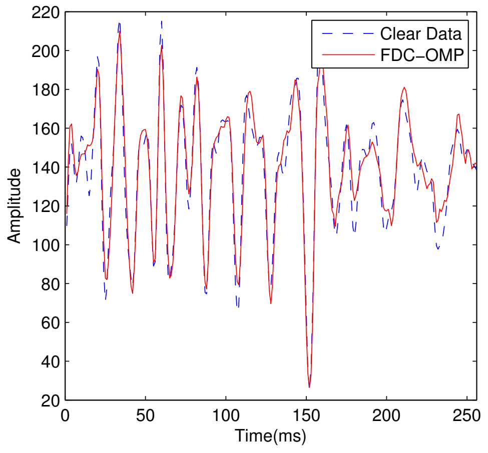
Figure 7: Difference between denoising results and field data one in Figure 4(a): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.



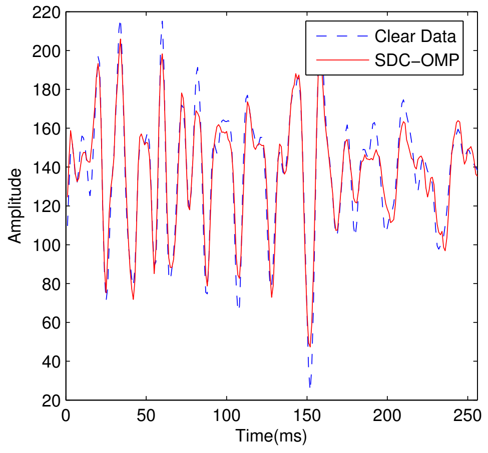
(a)



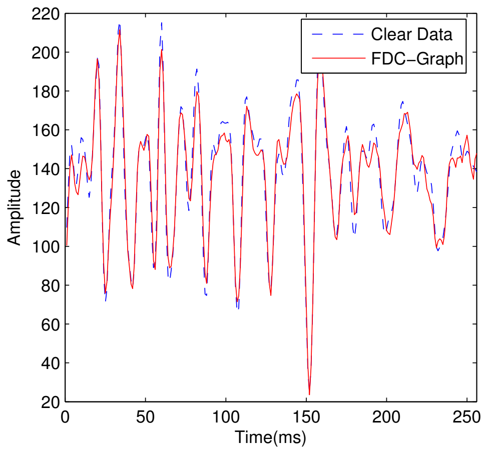
(b)



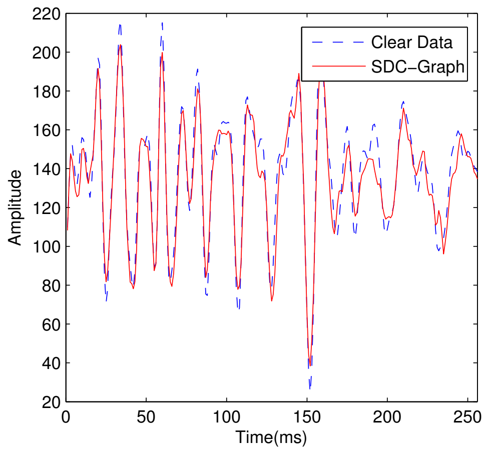
(c)



(d)

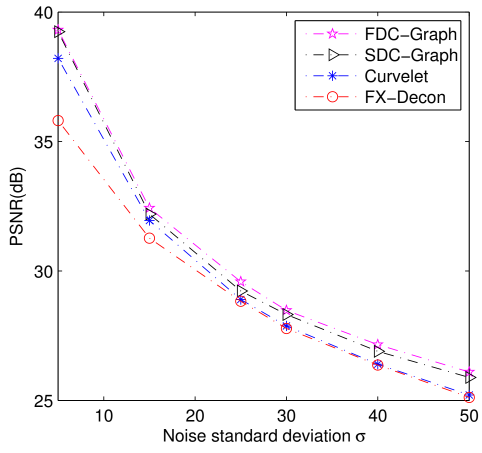


(e)

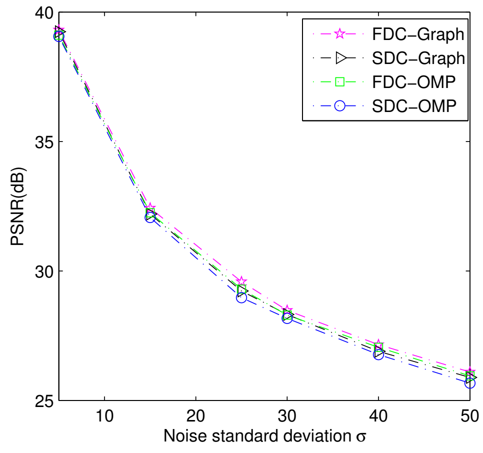


(f)

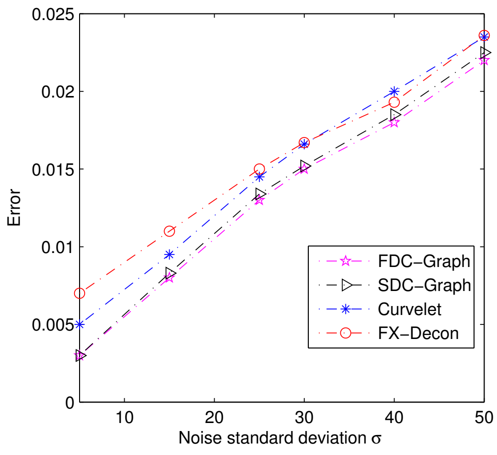
Figure 8: Single trace comparison of the reconstructions of the field data one in Figure 4(a): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.



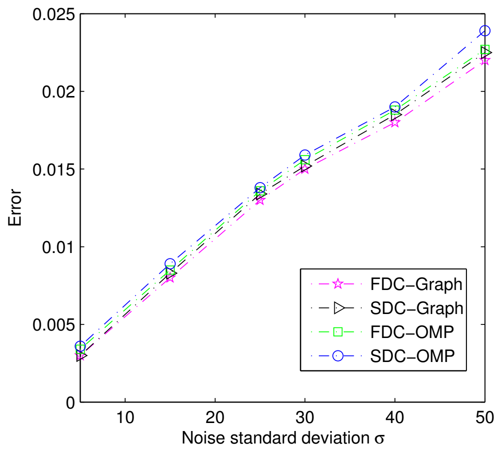
(a)



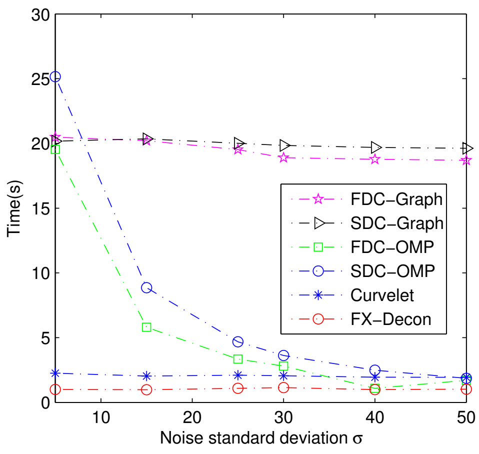
(b)



(c)



(d)



(e)

Figure 9: Denoising performance of FX-Decon, Curvelet, FDC-OMP, SDC-OMP, FDC-Graph and SDC-Graph algorithms for different noise standard deviations.

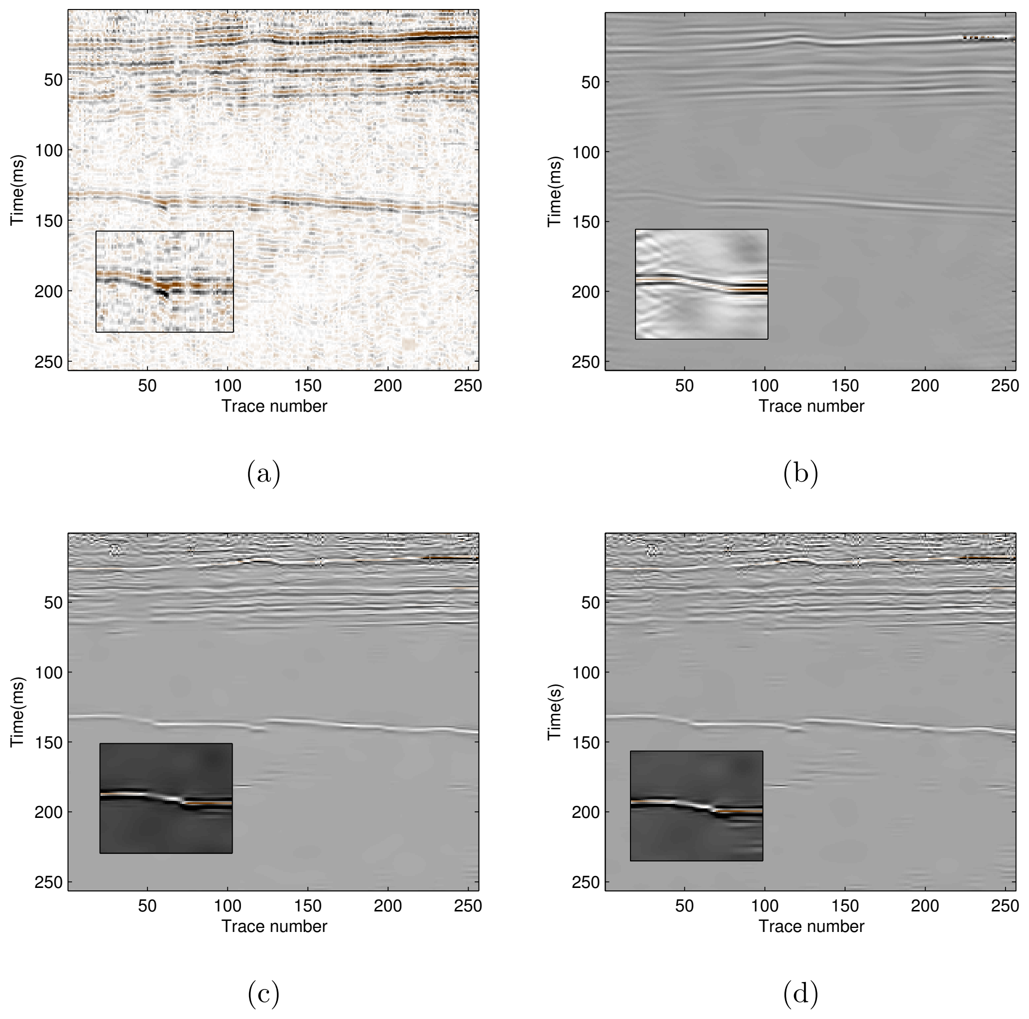
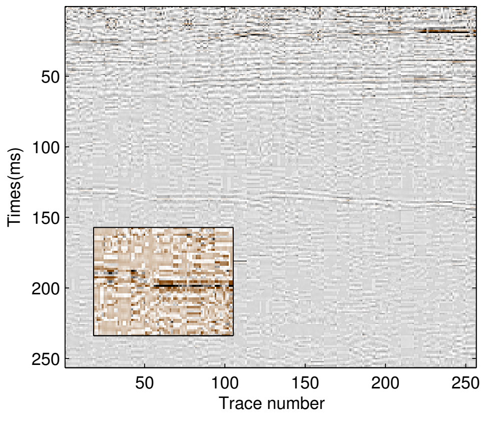
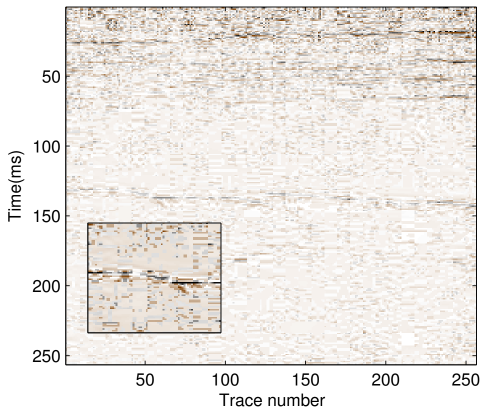


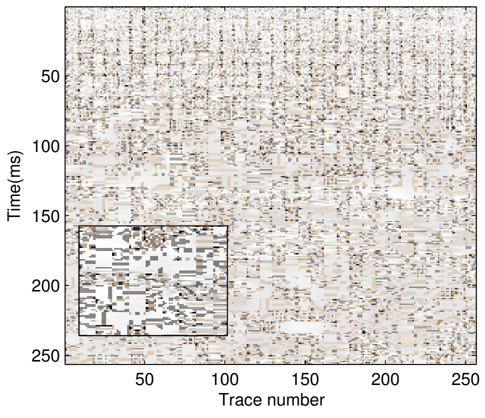
Figure 10: Denoising of field data two by FX-Decon (a), Curvelet (b), FDC-Graph (c) and SDC-Graph (d).



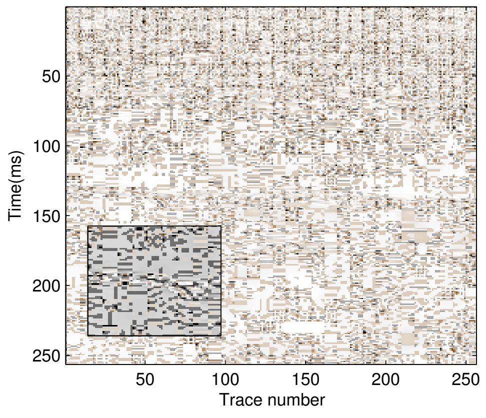
(a)



(b)



(c)



(d)

Figure 11: Difference between denoising results and field data two by FX-Decon (a), Curvelet (b), FDC-Graph (c) and SDC-Graph (d).