Dimensionality-reduced estimation of primaries by sparse inversion

by

Bander K. Jumah

B.Sc. Software Engineering, King Fahad University of Petroleum and Minerals, Dhahran, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Geophysics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

August 2014

Abstract

Data-driven methods—such as the estimation of primaries by sparse inversion suffer from the 'curse of dimensionality' that leads to disproportional growth in computational and storage demands when moving to realistic 3D field data. To remove this fundamental impediment, we propose a dimensionality-reduction technique where the 'data matrix' is approximated adaptively by a randomized low-rank factorization. Compared to conventional methods, which need passes through all the data possibly including on-the-fly interpolations for each iteration, our approach has the advantage that the passes is reduced to one to three. In addition, the low-rank matrix factorization leads to considerable reductions in storage and computational costs of the matrix multiplies required by the sparse inversion. Application of the proposed formalism to synthetic and real data shows that significant performance improvements in speed and memory use are achievable at a low computational overhead required by the low-rank factorization.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I wish to extend my most sincere gratitude and appreciation to my thesis advisor Felix J. Herrmann for his supervision of my research, creative ideas, encouragement and continues support through out my studies. Without him none of this would be possible.

In addition, I would like to thank my committee members Ozgur Ylmaz, Michael Friedlander and Phil Austin. I will always appreciate their input and feedback.

I would like also to thank my management at Saudi Aramco for sponsoring of my studies at UBC.

A special thanks goes to supervisor at Saudi Aramco, Saleh Al-Dossary for all his encouragement and support during my studies and work.

Dedication

To my parents,
my wife Sara,
my daughter Jude

Chapter 1

Introduction

Demand for oil and gas is increasing rapidly while new discoveries are more difficult to make because most of the relatively easy to find reservoirs are being depleted. This development combined with high oil prices and the decline of conventional oil reserves are the driving forces behind continued efforts of the oil and gas industry towards unconventional and more difficult to find and produce reservoirs. In order to achieve this ambition, state-of-the-art technologies, such as high-density, wide-azimuth seismic recording and imaging, are now being utilized to generate higher resolution images of the earth's subsurface.

Unfortunately, these new technologies generate enormous volumes of 3D data that require massive amounts of computational resources to store, manage, and processes. For example, it is nowadays not unusual to conduct seismic surveys that gather one million traces for a square mile compared 10,000 that were collected traditionally. This development not only makes the acquisition costly but it also makes processing these massive data volumes extremely challenging. These challenges become particularly apparent in the case of data-driven seismic inversion, e.g. Surface-Related Multiple Elimination (SRME, Berkhout and Verschuur, 1997) and Estimation of Primaries by Sparse Inversion (EPSI, van Groenestijn and Verschuur, 2009; Lin and Herrmann, 2011, 2012). These methods are known to be compute intensive because they rely on multi-dimensional convolutions that translate into massive dense matrix-vector multiplies, and therefore, suffer from the curse of dimensionality, where the size of the data volume increases exponentially with its dimension. For example, in the 3D case, having a data set with of 1000 sources and receivers in both directions results multiplications of dense matrices of size $1000^6 \times 1000^6$ for each frequency. Needless to say these matrices can not be stored are expensive to apply, and often require on-the-fly interpolation because acquired data is always incomplete.

In addition, working with massive data volumes creates communication bottlenecks that form a major impediment for iterative methods that requires multiples passes through all the data. For instance, this explains why current-day seismic data processing centers spend approximately the same

time on running SRME as on migration. EPSI generally requires more passes through the data than SRME, which explains the slow adaption of this new technology by industry.

In this paper, we present a dimensionality reduction technique that is aimed at limiting the number of passes over the data, including on-the-fly-interpolation, to one to three while reducing the memory imprint, accelerating the matrix multiplications, and leveraging parallel capabilities of modern computer architectures. A low-rank matrix factorization with the randomized singular-value decomposition (SVD, Halko et al., 2011) lies at the heart of our method. By selecting the rank adaptively for each angular frequency, our approach leads to approximations of monochromatic data matrices, where shot records are organized in columns, that lead matrix multiplies with a controllable error. The use of the randomized SVD allows us to limit passes over the data and permits matrix multiplies, which opens the way to use existing code bases for SRME.

A key step in the randomized SVD is formed by matrix probing (Halko et al., 2011; Chiu and Demanet, 2012; Demanet et al., 2012), where information on the range of a matrix is obtained by applying the matrix on random test vectors. The number of these vectors depends on the rank of the matrix, which in turn determines the degree of dimensional reduction and speedup yielding by the low-rank approximation. The dimensionality reduction also allows us to compute the SVD for very large problems sizes.

The paper is organized as follows. First, we briefly introduce EPSI by recasting Berkhout's data matrix into a vectorized form, which makes it conduce curvelet-domain sparsity promotion. Next, we identify that the matrix multiplies are the dominant cost and we show that these costs can be reduced by replacing the data matrix for each frequency by a low-rank factorization with SVDs. To get best accuracy, we propose an adaptive scheme that selects the appropriate ranks for each data matrix. Second, we introduce the randomized SVD based matrix probing, which allows to carry out the SVD on large systems from information by the matrix probing. We also discuss how matrix can be extended such that it no longer relies on full sampling but instead can work with data with missing shots. We show that this reduces the number of passes through the data significantly. Third, we address the increase in ranks as the temporal frequency increases by introducing Hierarchically Semi-Separable Matrix Representation (HSS, Lin et al., 2011) matrices. We conclude by performing tests of our method on synthetic and real seismic lines.

1.1 Theory

Estimation of Primaries by Sparse Inversion (EPSI) proposed by van Groenestijn and Verschuur (2009) is an important new development in the mitigation of surface-related multiples. As opposed to conventional multiple removal, where multiples are predicted and subtracted after matching, the surface-related multiples are mapped to the surface-free Green's function by carrying out a sparse inversion. In describing the EPSI formulation, we make use of Berkhout's detail-hiding monochromatic matrix notation (Figure 1.1), where each monochromatic wavefield is arranged into a matrix with columns representing common shot gathers and rows representing common receiver gathers. Hatted quantities correspond to the monochromatic quantities, and upper case variables denote matrices or linear operators. Multiplication of two hatted matrices, corresponds to convolution in the physical domain. EPSI describes the relation between the total up-going wavefield $\widehat{\mathbf{P}}$, the surface-free Green's function $\widehat{\mathbf{G}}$, and $\widehat{\mathbf{Q}}$ representing the source signature. The EPSI formulation is expressed as

$$\widehat{\mathbf{P}} = \widehat{\mathbf{G}}(\widehat{\mathbf{Q}} - \widehat{\mathbf{P}}). \tag{1.1}$$

Following the work of Lin and Herrmann (2011) the above formulation serves as a basis for an inversion problem, where we solve for an unknown vector $\mathbf{x}$, from an observation vector $\mathbf{b}$, with the relationship $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a linear operator. Before we introduce the multiple prediction operator, we first, define the convention of vectorized monochromatic wavefields, where columns of the monochromatic data matrix are stacked as a single long column. Vectorized wavefields are represented as lower case quantities e.g. $\mathbf{p}$ =vec($\mathbf{P}$), where vec stands for the linear operation that stacks the columns of a matrix into a long concatenated vector (matlabs colon operator). In order to utilize curevelts within our EPSI formulation, we need to reformulate our EPSI problem into a matrix-vector format instead of the matrix-matrix format using the following identity:

$$\mathrm{vec}\left(\mathbf{AXB}\right) = \left(\mathbf{B}^T \otimes \mathbf{A}\right) \mathrm{vec}\left(\mathbf{X}\right), \tag{1.2}$$

which holds for arbitrary matrices – of compatible sizes – $\mathbf{A}, \mathbf{X}$, and $\mathbf{B}$. In this expression, the symbol $\otimes$ refers to the Kronecker product. Equation 1.1, now can be rewritten as

$$\left((\widehat{\mathbf{Q}} - \widehat{\mathbf{P}})_i^T \otimes \mathbf{I}\right) \mathrm{vec}\left(\widehat{\mathbf{G}}_i\right) = \mathrm{vec}\left(\widehat{\mathbf{P}}_i\right), \; i = 1 \cdots n_f, \tag{1.3}$$

where $\mathbf{I}$ is the identity matrix. After the inclusion of the curvelet synthesis and temporal Fourier transforms ($\mathbf{F}_t = (\mathbf{I} \otimes \mathbf{I} \otimes \mathcal{F}_t)$ with $\mathcal{F}_t$ the temporal Fourier transform), we finally arrive at the following block-diagonal system

$$\underbrace{\mathbf{F}_t^* \begin{bmatrix} \left((\widehat{\mathbf{Q}} - \widehat{\mathbf{P}})_1^T \otimes \mathbf{I}\right) & & \\ & \ddots & \\ & & \left((\widehat{\mathbf{Q}} - \widehat{\mathbf{P}})_{n_f}^T \otimes \mathbf{I}\right) \end{bmatrix} \mathbf{F}_t}_{\mathbf{U}} \underbrace{\begin{bmatrix} \mathrm{vec}\,(\mathbf{G}_1) \\ \vdots \\ \mathrm{vec}\,(\mathbf{G}_{n_t}) \end{bmatrix}}_{\mathbf{x}} \approx \underbrace{\begin{bmatrix} \mathrm{vec}\,(\mathbf{P}_1) \\ \vdots \\ \mathrm{vec}\,(\mathbf{P}_{n_t}) \end{bmatrix}}_{\mathbf{b}}. \tag{1.4}$$

This equation is amenable to transform-domain sparsity promotion i.e., Equation 1.4 can now be written as

$$\mathbf{A}\mathbf{x} \approx \mathbf{b}, \tag{1.5}$$

In this expression, we use the symbol * to denote the Hermitian transpose or adjoint.

The above vectorized equation is amenable to transform-domain sparsity promotion by defining $\mathbf{A} := \mathbf{U}\mathbf{S}^*$, where $\mathbf{g} = \mathbf{S}^*\mathbf{x}$ is the transform-domain representation of $\mathbf{g}$. We use a combination of the 2D curvelet, along the source-receiver coordinates, and the 1D wavelet transform along the time coordinates. Using the Kronecker product again, we define $\mathbf{S} = \mathbf{C} \otimes \mathbf{W}$. With these definitions, we relate the sparsifying representation for the surface-free Green's function to the vectorized upgoing wavefield $\mathbf{b} = \mathrm{vec}\,(\mathbf{P})$ via $\mathbf{A}\mathbf{x} = \mathbf{b}$. This relationship forms the basis for our inversion.

Solving for the transform-domain representation of the wavefield $g(t, x_s, x_r)$ with t time, x_s the source, and x_r the receiver coordinates, now corresponds to inverting a linear system of equations where the monochromatic wavefield $\{(\widehat{\mathbf{Q}} - \widehat{\mathbf{P}})_i\}_{i=1\cdots n_f}$ and temporal wave-field $\{\mathbf{P}_i\}_{i=1\cdots n_t}$ are related — through the temporal Fourier transform—to the curvelet representation of the discretized wavefield vector $\mathbf{g}$ in the physical domain.

The linear operator $\mathbf{A}$ first applies the inverse curvelet-transform to the vector $\mathbf{x}$, yielding the discrete approximation of the wavefield $g(t, x_s, x_r)$, followed by the application of the temporal Fourier transform operator $\mathbf{F}_t$ and the "vectorized" matrix-matrix multiplication by $\left((\widehat{\mathbf{Q}} - \widehat{\mathbf{P}})_i^T \otimes \mathbf{I}\right)$ for each frequency, i.e., for $i = 1 \cdots n_f$. In the time domain, these operations correspond to a multidimensional convolution between discrete approximations of the wave-fields $g(t, x_s, x_r)$ and $u(t, x_s, x_r)$. Application of $\mathbf{A}^*$ is similar, except that it applies $\left((\widehat{\mathbf{Q}} - \widehat{\mathbf{P}})_i \otimes \mathbf{I}\right)$ for each frequency (with the

bar denoting complex conjugate), followed by the forward curvelet transform. This forward transform is applied to a vector that contains the result of a multidimensional correlation of two discrete wave-fields. The solution of the EPSI problem, is now of the following form

$$\tilde{\mathbf{x}} = \arg \min_{\mathbf{x}} \|\mathbf{x}\|_{\mathbf{1}} \text{ subject to } ||\mathbf{A}\mathbf{x} - \mathbf{b}||_{\mathbf{2}} \leq \sigma, \tag{1.6}$$

where σ is the error between the recorded and predicted data.

Solving optimization problems (cf. Equation 1.6) requires multiple iterations involving the application of $\mathbf{A}$, $\mathbf{A}^*$, and possibly $\mathbf{A}^*\mathbf{A}$. In real applications, application of these matrix-vector multiplies is challenging because (i) the matrices are full and extremely large, e.g. for each frequency the data matrix becomes easily $10^6 \times 10^6$ for $n_s = n_r = 1000$ (with n_s the number of sources and n_r the number of receivers), so they require lots of storage and computational resources; (ii) data is incomplete, which requires 'on-the-fly' interpolations that are costly but have the advantage that the data matrix does not need to be formed explicitly; and (iii) the solvers require multiple evaluations of $\mathbf{A}$, $\mathbf{A}^*$, and possibly $\mathbf{A}^*\mathbf{A}$. Each application of the operator $\mathbf{A}$ requires a complete pass over the data, where the data needs to be transferred into and out of the CPU. To reduce the storage and multiplication costs of these operations, we replace the data matrix $\hat{\mathbf{P}}$ by a low-rank approximation using Singular Value Decomposition **SVD**.

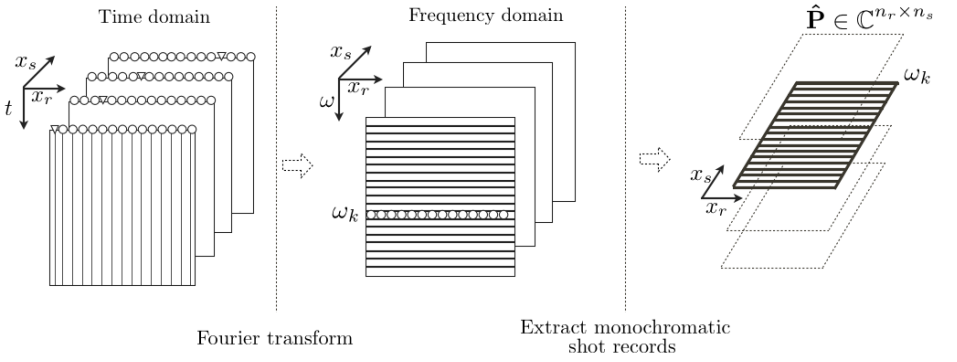


Figure 1.1: Extraction of monochromatic wavefield matrices, by transforming the data from the time domain into the frequency domain via The Fourier transform. Then for each frequency component ω_k a data matrix $\hat{\mathbf{P}} \in \mathbb{C}^{n_r \times n_s}$ is extracted (adapted from Dedem (2002)).

Chapter 2

Dimensionality-reduced estimation of primaries by sparse inversion

By applying dimensionality reduction techniques, we are able to represent large data matrices in terms of much smaller low-rank matrix factorizations. The most basic form of the low-rank matrix approximation can be represented as

$$\widehat{\mathbf{P}}_{m\times n} \approx \widehat{\mathbf{L}}_{m\times k}\widehat{\mathbf{R}}^{*}_{k\times n}, \tag{2.1}$$

where the $\approx$ sign refers to an approximation with a controllable error, $\widehat{\mathbf{P}}$ is the data matrix with m rows and n columns (n_s sources and n_r receivers in a seismic context), $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{R}}^{*}$ are the matrix decompositions, and k represents the rank of the data matrix. In order for the approximation to be low rank, k has to be much smaller than m and n i.e. ($k << min(m, n)$).

A special case of this low rank matrix factorization, is the Singular Value Decomposition (SVD) where the data matrix $\widehat{\mathbf{P}}$ is decomposed into 3 factors:

$$\widehat{\mathbf{P}}_{n_r\times n_s} \approx \widehat{\mathbf{U}}_{n_r\times k}\widehat{\mathbf{S}}_{k\times k}\widehat{\mathbf{V}}^{*}_{k\times n_s}, \tag{2.2}$$

where $\widehat{\mathbf{U}} \in \mathbb{C}^{n_r\times k}$ and $\widehat{\mathbf{V}} \in \mathbb{C}^{k\times n_s}$ are the left and right singular vectors of $\widehat{\mathbf{P}}$, respectively. The matrix $\widehat{\mathbf{S}} \in \mathbb{C}^{k\times k}$ is diagonal with non-negative singular values on the diagonal. An example of an SVD factorization of a monochromatic data matrix of size $n_s = n_r = 150$ at a frequency of 40Hz is shown in Figure 2.1, where the rank k is set to 20. Using this factorization we were able to represent the data in terms of much smaller matrices, containing fewer number of elements making it cheaper to store and apply the approximated data on other matrices. Depending on the rank k significant savings can be made in terms of memory and multiplication costs as summarized Table 2.1.

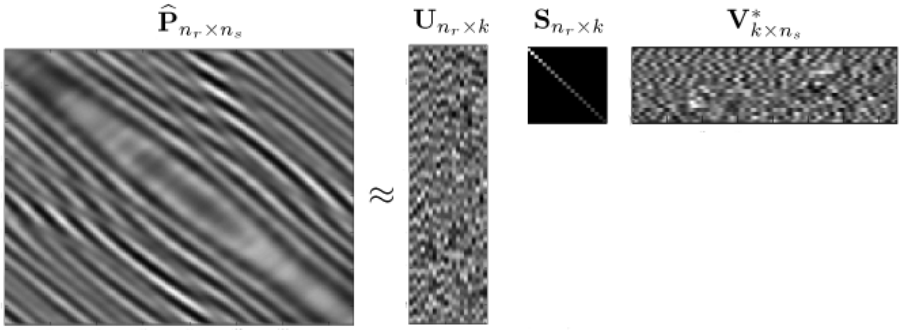


Figure 2.1: SVD factorization, of a frequency slice $\widehat{\mathbf{P}} \in \mathbb{C}^{n_r \times n_s}$ on left, and its SVD decompositions on right. $\mathbf{U}_{n_r \times k}$ and $\mathbf{V}^*_{k \times n_s}$ are left and right singular vectors of $\widehat{\mathbf{P}}$ respectively, $\mathbf{S}_{k \times k}$ holds the singular values of $\widehat{\mathbf{P}}$ on its diagonal.

	SVD approximation	Regular method
Matrix-matrix multiplication	$\mathcal{O}(kn(2n+k))$	$\mathcal{O}(n^3)$
Storage	$\mathcal{O}(k(2n+1))$	$\mathcal{O}(n^2)$

Table 2.1: Advantages of using low-rank approximations via SVD in terms of matrix-matrix multiplications and storage requirements.

2.1 Dimensionality-reduced EPSI

Recalling the EPSI formulation defined in equation 1.1, we can now replace each monochromatic data matrix $\widehat{\mathbf{P}}$ in our dataset with its decompositions: $\widehat{\mathbf{U}} \in \mathbb{C}^{n_r \times k}$, $\widehat{\mathbf{S}} \in \mathbb{C}^{k \times k}$ and $\widehat{\mathbf{V}} \in \mathbb{C}^{k \times n_s}$, giving us the following multiple prediction operator:

$$\mathbf{U} := \mathbf{F}_t^* \,\text{blockdiag}\left[\widehat{\mathbf{Q}} - \widehat{\mathbf{U}}\widehat{\mathbf{S}}\widehat{\mathbf{V}}^*\right]_{1 \cdots n_f} \mathbf{F}_t, \tag{2.3}$$

where we can rapidly apply the data matrix $\widehat{\mathbf{P}}$ and its adjoint $\widehat{\mathbf{P}}^*$ on any other matrix. Additional speed up is achieved in computing the product $\widehat{\mathbf{P}}^*\widehat{\mathbf{P}}$ by utilizing the orthogonality of the left singular vectors of $\widehat{\mathbf{P}}$ via performing $\widehat{\mathbf{P}}^*\widehat{\mathbf{P}} = \mathbf{V}\mathbf{S}^\mathbf{2}\mathbf{V}^*$, reducing the cost of this multiplication from $O(N^3)$ to approximately $O(N^2 k)$ operations,i.e, by a factor of $\frac{k}{N}$.

To successfully apply our dimensionally reduced system, we must strike a balance between the amount of speed up we are trying to achieve, and the

quality of our approximation. To demonstrate the effect of such trade off, two approximation examples – of a monochromatic data matrix at frequency of 40Hz with $n_s = n_r = 150$ – are shown in Figure 2.2 and in Figure 2.3 representing a good and a bad trade off of accuracy over speed up, respectively. In the first example, the rank k was set to 20, giving us good approximation quality, where we preserved the details of our monochromatic data matrix. However, in the second example, we tried to achieve a higher speed up by setting $k = 4$, resulting in a loss of significant details, especially on the diagonal of the data matrix. This loss in the diagonal details of the monochromatic data matrix corresponds to losing the curvature of the seismic events in the time domain, as shown in Figure 2.4. In this work, we will use the spectral norm (also know as the operator norm), as a means of measuring the accuracy of our low-rank approximation. It relates the energy of a vector, which is the result of a matrix-vector product, to the energy of the input vector. The spectral norm denoted by $\|.\|_s$, is computed by taking the maximum singular value of $\widehat{\mathbf{P}}$, and it is given by

$$\|\widehat{\mathbf{P}}\|_S = \lambda_1 = \max_{x \neq 0} \frac{\|\mathbf{P}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}, \tag{2.4}$$

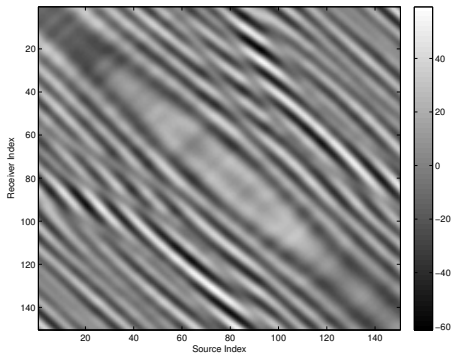
where $\widehat{\mathbf{P}}$ is the data matrix, $\mathbf{x}$ is an arbitrary input vector, and λ_1 is the maximum singular value of $\mathbf{P}$. The symbol $\|.\|_2$ indicates the l^2-norm, which is defined as the square root of the sum of the absolute values of an input vector $\mathbf{x}$, and its given by

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{k=1}^{n} |x_k|^2}. \tag{2.5}$$

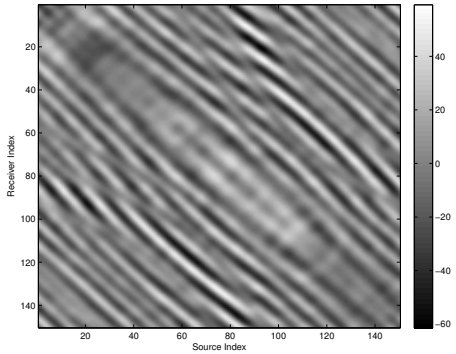
2.1.1 Adaptive data-matrix approximation

Using the spectral norm, we develop a data-adaptive low-rank approximation scheme according to the following two scenarios: namely we either approximate the data matrix with a fixed rank for each frequency slice or we select the rank adaptively to improve the quality of the approximation based on the spectral norm. We keep the total rank budget, which is the sum of the ranks for all frequency slices, the same so we achieve the same improvement in storage and matrix multiply costs.

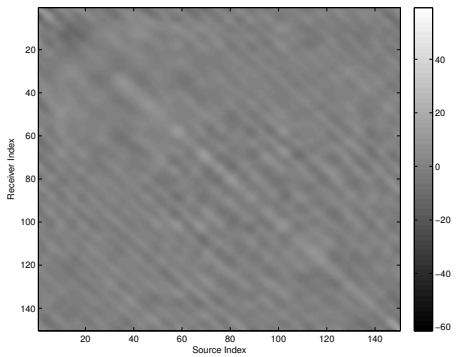
In approximating our datasets, we specify a percentage of the rank budget to be used in the approximation. This percentage will serve as



(a)

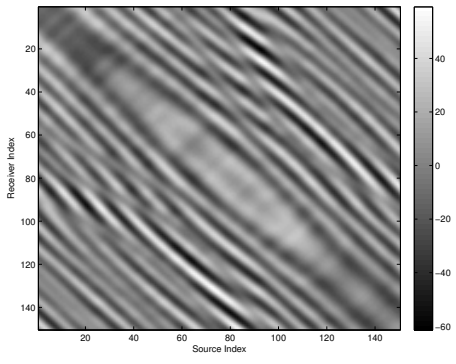


(b)

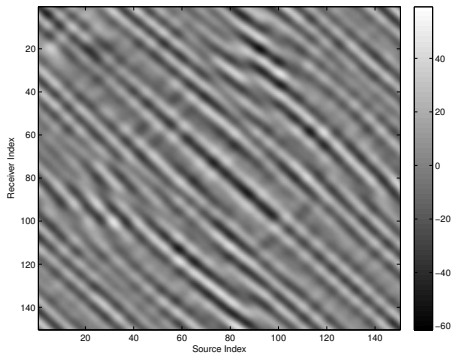


(c)

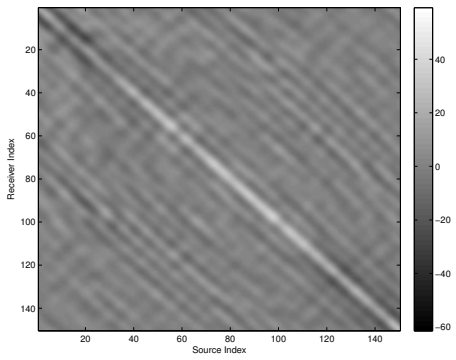
Figure 2.2: Example of a good SVD approximation, of a frequency slice with $n_s = n_r = 150$ at a frequency of $\omega = 50$Hz. (a) Frequency slice without approximation, (b) frequency slice approximated using 20 sources out of 150, and (c) difference between full data and the approximation. Comparing the original data, to its approximation we get a Signal-to-Noise Ratio (SNR) of 16dB, where $SNR = 20 \log_{10}(\frac{\|\mathbf{p}-\tilde{\mathbf{p}}\|_2}{\|\mathbf{p}\|_2})$.



(a)

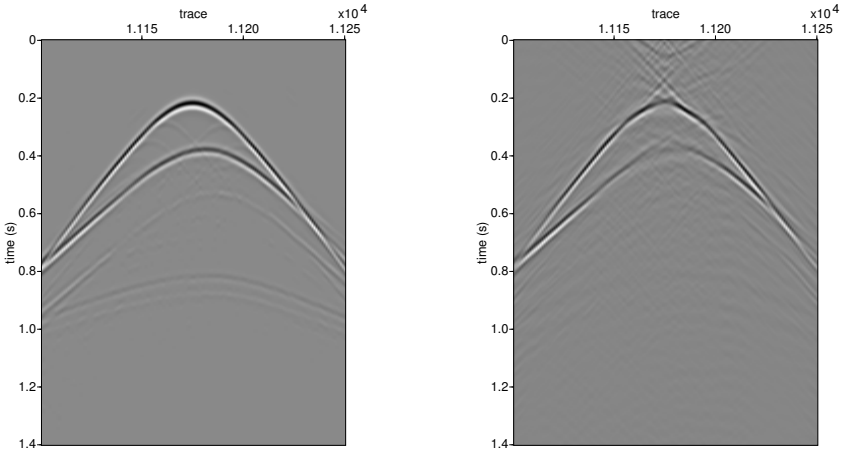


(b)



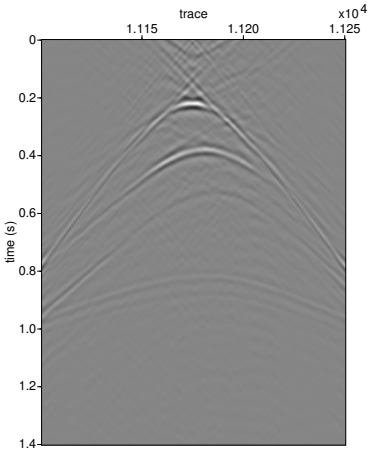
(c)

Figure 2.3: Example of a bad SVD approximation of a frequency slice, with $n_s = n_r = 150$ at a frequency of $\omega = 50$Hz. (a) Frequency slice without approximation, (b) frequency slice approximated using 5 sources out of 150, and (c) difference between full data and the approximation. Comparing the original data, to its approximation we get an SNR = 6dB.



(a)

(b)



(c)

Figure 2.4: Effect of losing the diagonal details in the frequency domain (Figure 5) on the data in the time domain. ($n_s = n_r = 150$ and 512 time samples with $\Delta t = .004s$). (a) Original shot gather, (b) shot gather of the approximated data, and (c) difference between full data and its approximation. Comparing the original data, to its approximation we get an SNR = 6dB.

a subsampling ratio δ, that will determine, the amount of speed up we achieve in the matrix-matrix products, the compression percentage of the data, and the quality of the approximation. The compression rate per frequency slice, is related to the subsampling ratio, by the following relation: $c = (1-\delta(2+\delta))*100$. Here, c is the compression percentage for the specified subsampling ratio δ. Matrix-matrix multiplication speed up is computed using the following relation: $s = \frac{1}{2\delta+\delta^2}$, where s, is the expected matrix-matrix multiplication speed up using the specified subsampling ratio δ.

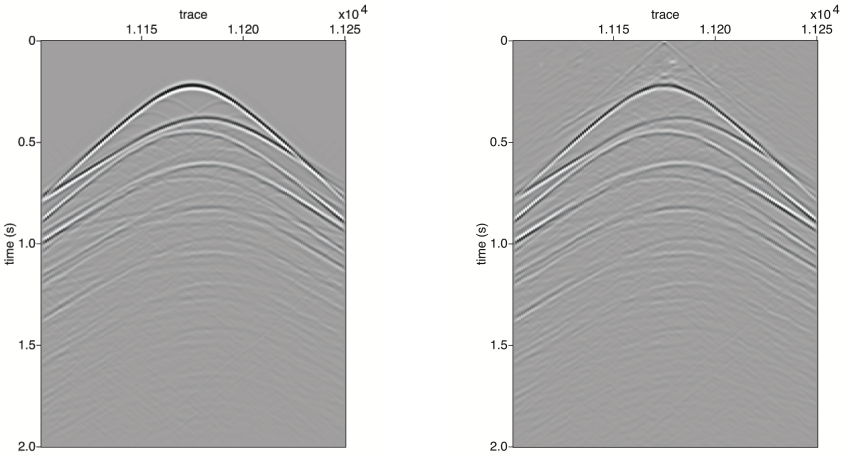
In the first approximation scenario, we distribute the approximation ranks, evenly for each frequency slice. Results of applying this approximation scenario are summarized in Table 2.2, where a synthetic dataset (of size $n_t \times n_s \times n_r = 150 \times 150 \times 512$) is approximated, using different subsampling ratios. As observed from the table, significant improvements in memory usage and efficiency were achieved. However, these improvements go at the expense of quality as shown in Figure 2.5.

Subsampling ratio δ	1/3	1/5	1/8	1/12
Mat-mat multiplication speed-up	1.4×	2.2×	4.7×	6.5×
Memory reduction	31%	56%	79%	86%
SNR dB	11	7	4	3

Table 2.2: Results of applying the first approximation scenario, where a synthetic dataset, of size $n_s = n_r = 150$, and $n_t = 512$, is approximated using a fixed rank approximation.

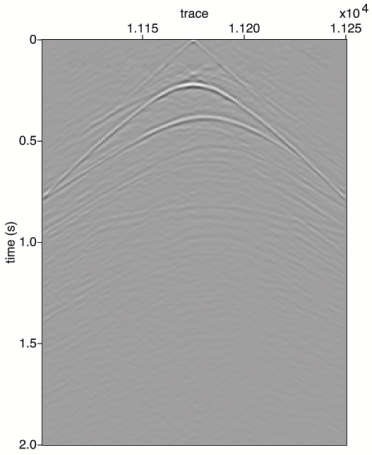
In the second scenario, we approximate the dataset based on the spectral norm of each monochromatic data matrix. Let us in our analysis first look at Figure 2.8, where each column represents the singular values of a frequency slice, at frequencies ranging between 1Hz and 250Hz. As noted from the figure, the most significant singular values are concentrated within the seismic band between frequencies 24Hz and 98Hz, while the remaining singular values are insignificant. This effect is mainly due to the behavior of the amplitude of the seismic wavelet, which varies with frequency (Figure 2.7). When we applied this approximation scenario, where we adaptively selected the rank of each monochromatic data matrix based on its spectral norm (Figure 2.8), we were able to achieve superior approximation qualities over the first approximation scenario as reflected by the SNR values in Table 2.3 and as shown by the shot gathers in Figure 2.9.

Using the SVD approximations, we were able to approximate our datasets,



(a)

(b)



(c)

Figure 2.5: Synthetic shot gather of size $n_s = n_r = 150$ and 512 time samples. a) Original shot gather , b) Shot gather using fixed rank approximation, with 9% of the rank budget. c) difference between the original data and its approximation.

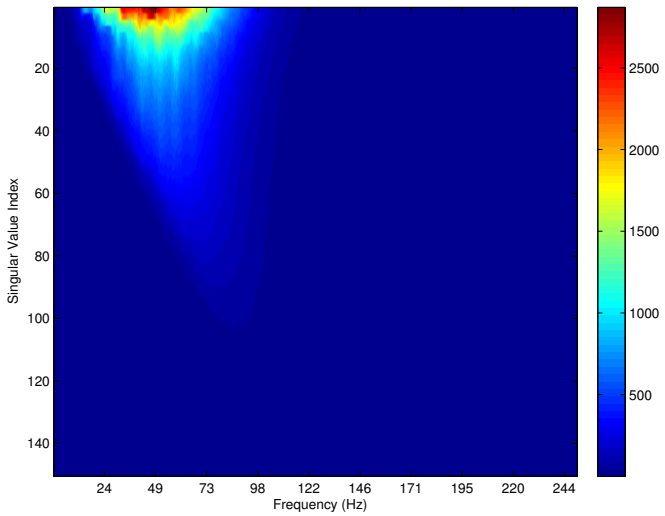


Figure 2.6: Singular values for each frequency slice, plotted as columns of the figure. Notice how the highest energies are concentrated within the seismic band.

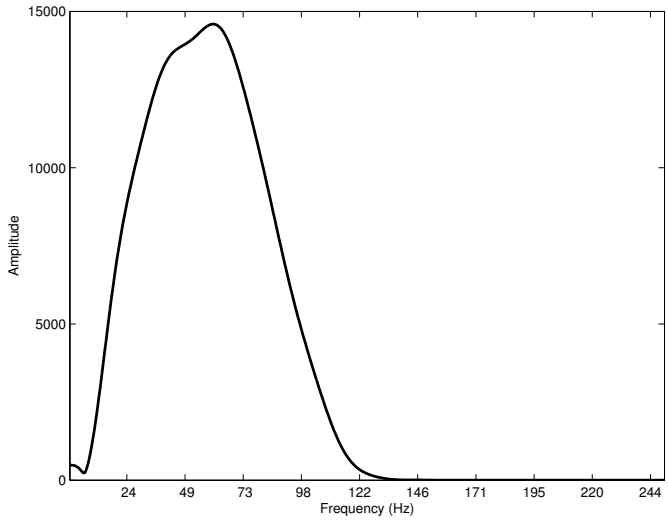


Figure 2.7: Amplitude of the source wavelet.

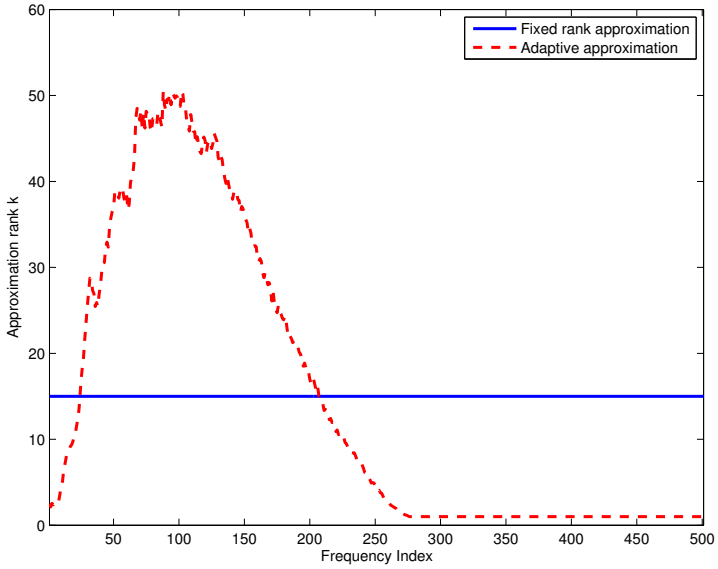
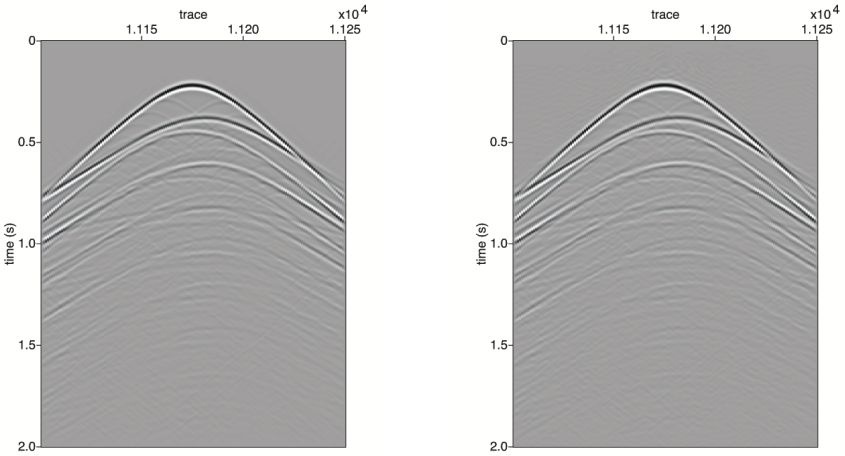
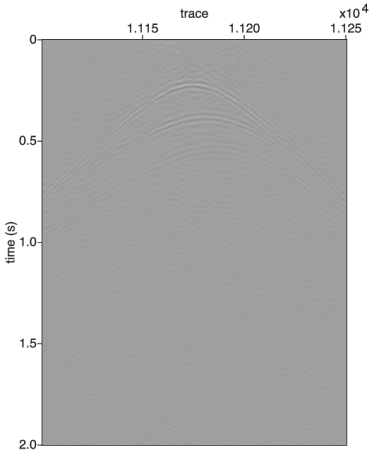


Figure 2.8: Distribution of approximation ranks based on the two scenarios (fixed and adaptive) using 10% of the total rank budget.



(a)

(b)



(c)

Figure 2.9: Synthetic shot gather of size $n_s = n_r = 150$ and 512 time samples. a) without approximation , b) adaptive rank approximation, using 9% of the rank budget. c) difference between the original data and its approximation.

Subsampling ratio δ	1/3	1/5	1/8	1/12
Speed-up	2×	5×	8×	11×
Memory Compression	31%	56%	79%	86%
SNR dB	20	18	11.4	8

Table 2.3: Results of applying the adaptive approximation scenario, where a synthetic dataset, of size $n_s = n_r = 150$, and $n_t = 512$ is approximated.

with high accuracy, allowing us to handle larger datasets, which will facilitate the handling of 3D data. However, our approach comes with an additional pre-processing step to perform the SVD operation. Efficient implementations of the SVD operation do exists for small data sizes, however, these implementations are expensive, when it comes to dealing with larger datasets, where communication becomes the real bottle neck. Classical SVD methods, require at least k passes over the data, requiring the data to be transferred into the CPU k number of times (Halko et al. (2011)). To overcome this challenge, we adapt randomization techniques, for large scale problems, which may not fit in the main memory, and might need to be retrieved in and out of the CPU.

Chapter 3

Randomized Singular Value Decomposition

To overcome the communication bottleneck, introduced by the modern computing environments. In approximating our data, we utilize randomized SVDs, which are efficient in approximating large data matrices. The main advantage in using the randomized SVD methods, is in their ability to factorize the data using only few passes over the data (1-3 passes) compared to k passes using the regular SVD methods. In this section we will look at 3 randomized SVD methods, developed by Halko et al. (2011). We will start the discussion by introducing the general framework these algorithms work in. Then we will show how this framework is improved, to handle matrices suffering from slow singular value decay. The third method, will show how the SVD computation cost can be reduced, using fast transforms. Finally we will show how the communication cost can be reduced further, by randomly sampling the data using the Dirac basis.

3.1 General framework

The general framework in developing the randomized SVD algorithms is composed of two stages: First, capturing the action of the data matrix on a small number of vectors, resulting in a smaller system to deal with. Second, from this reduced system, an approximation to the SVD is computed more cheaply. The two stages of the randomized SVD are described below:

Stage A Capturing the action of the data matrix. This stage makes use of random sampling and fast matrix-vector multiplies, to capture the action of the data matrix on a small number of random vectors, as given by:

$$\mathbf{Y} = \widehat{\mathbf{P}}\widehat{\mathbf{W}}, \tag{3.1}$$

where $\widehat{\mathbf{W}}_{n\times(p+k)}$ is a random Gaussian matrix, of mean 0 and variance 1. k is the rank of the input matrix $\widehat{\mathbf{P}}$, and p is an over-sampling parameter (typically set to a value between 5-10). Multiplication with a random set of vector, corresponds to randomly combining shots into 'supershots', using random superposition. This sampling technique, leads to a significantly reduced system, which will require smaller number of computations, at the cost of introducing a random cross-talk between the sources (van Leeuwen et al. (2011)). Figure 3.1, is an example of the sampling operation, where the data matrix $\widehat{\mathbf{P}}$, is sampled using a small set of random vectors $\widehat{\mathbf{W}}$:

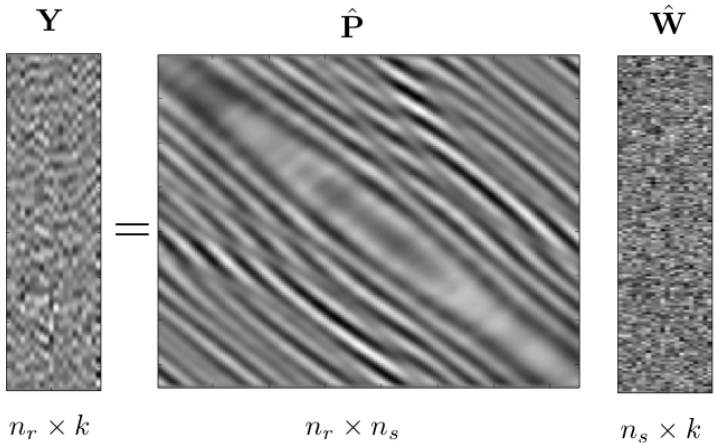


Figure 3.1: Supershots created by randomly picking and weighting (using random Gaussian weights) sequential shots.

The next step in this stage, is to find an approximate basis $\widehat{\mathbf{Q}}$, that provides a good approximation of the data matrix $\widehat{\mathbf{P}}$,i.e., we have:

$$\|\widehat{\mathbf{P}} - \widehat{\mathbf{Q}}\widehat{\mathbf{Q}}^*\widehat{\mathbf{P}}\|_2 < \epsilon, \tag{3.2}$$

where ϵ is a measure of the approximation error. The matrix $\widehat{\mathbf{Q}}$ is computed by the **QR** factorization, giving us:

$$\mathbf{Y} = \widehat{\mathbf{Q}}\widehat{\mathbf{R}}, \tag{3.3}$$

where, $\widehat{\mathbf{Q}} \in \mathbb{C}^{n_s \times k}$ is an orthonormal matrix and $\widehat{\mathbf{R}} \in \mathbb{C}^{k \times k}$ is an upper triangle matrix. From $\widehat{\mathbf{Q}}$, we can compute a reduced system as follows:

$$\mathbf{B}_{n_r \times k} = \widehat{\mathbf{Q}}_{n_r \times k}\widehat{\mathbf{P}}_{n_r \times n_s}. \tag{3.4}$$

This stage of the approximation is summarized as Algorithm 1.

Algorithm 1 Basic randomized SVD

Input: $\widehat{\mathbf{P}}_{n_r \times n_s}$, a target rank k and an over-sampling parameter p
Output Orthonormal matrix $\widehat{\mathbf{Q}}$ whose range approximates the range of $\widehat{\mathbf{P}}$

1. Draw a random Gaussian matrix $\widehat{\mathbf{W}}_{n_s \times (k+p)}$.
2. Form $\mathbf{Y}_{n_r \times (k+p)} = \widehat{\mathbf{P}}\widehat{\mathbf{W}}$.
3. Construct the orthonormal matrix $\widehat{\mathbf{Q}}_{n_r \times (k+p)}$ by computing **QR** factorization of $\mathbf{Y}$.

Stage B: Computing an approximation of the SVD. In this stage, we compute the approximate low-rank factorizations (using deterministic SVD) on the reduced system, that is the output of stage A. The advantage in this step, is that the full data matrix needs to be accessed for only 1 to 2 times, compared to k passes over the data, as required by the classical SVD (Halko et al. (2011)). More specifically we compute:

$$\mathbf{B}_{n_r \times k} = \widetilde{\mathbf{U}}_{k \times k}\widehat{\mathbf{S}}_{k \times k}\widehat{\mathbf{V}}^*_{k \times n_s}, \tag{3.5}$$

with:

$$\widehat{\mathbf{U}}_{n_r \times k} = \mathbf{Q}_{n_r \times k}\widetilde{\mathbf{U}}_{k \times k}. \tag{3.6}$$

Figure 3.2, describes how the left singular vectors $\widehat{\mathbf{U}}$ of a data matrix $\widehat{\mathbf{P}}$, are computed using the orthonormal matrix $\widehat{\mathbf{Q}}$.

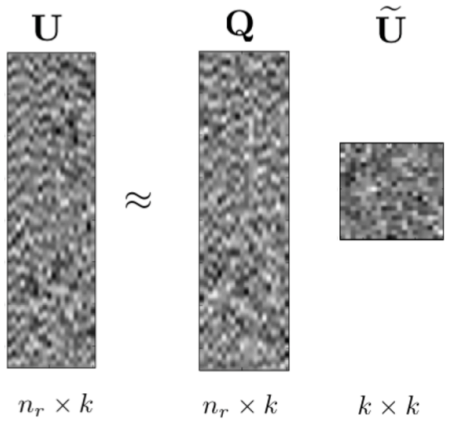


Figure 3.2: Approximation of left singular vectors $\widehat{\mathbf{U}}$ of a data matrix $\widehat{\mathbf{P}}$ using the orthonormal matrix $\widehat{\mathbf{Q}}$.

3.2 Slow decay of singular values

Algorithm 1, is most appropriate when used to approximate matrices with fast decaying singular values. However, it performs poorly, when approximating matrices exhibiting slow singular value decay (Halko et al. (2011)). In this case, singular vectors associated with the smaller singular values are known to interfere with the approximation and this leads to poor approximation a quality (Halko et al. (2011)). To reduce this interference, the singular values can, as proposed by (Halko et al. (2011)), be reweighed by considering powers of the matrix, i.e., we replace Equation 3.1 by

$$\mathbf{B} = (\widehat{\mathbf{P}}\widehat{\mathbf{P}}^*)^q \widehat{\mathbf{P}}\widehat{\mathbf{W}}, \tag{3.7}$$

with q chosen order of the power. As a result of raising the matrix to the q^{th} power, the singular values decay faster, (Figure 3.3), hence reducing the interference. This interference reduction comes on the cost of $(q+1)$ more passes over the data. For $q = 1$, the power iteration corresponds to:

$$\begin{aligned}
\mathbf{B} &= (\widehat{\mathbf{P}}^*\widehat{\mathbf{P}})^1 \widehat{\mathbf{P}}\widehat{\mathbf{W}}. \\
&= (\mathbf{USV}^*\mathbf{VSU}^*)\mathbf{USV}^*\widehat{\mathbf{W}}. \\
&= \mathbf{US^2U}^*\mathbf{USV}^*\widehat{\mathbf{W}}. \\
&= \mathbf{US^3V}^*\widehat{\mathbf{W}}.
\end{aligned}$$

Experience has shown, that it is usually enough to set q with values $q = 1, 2$.

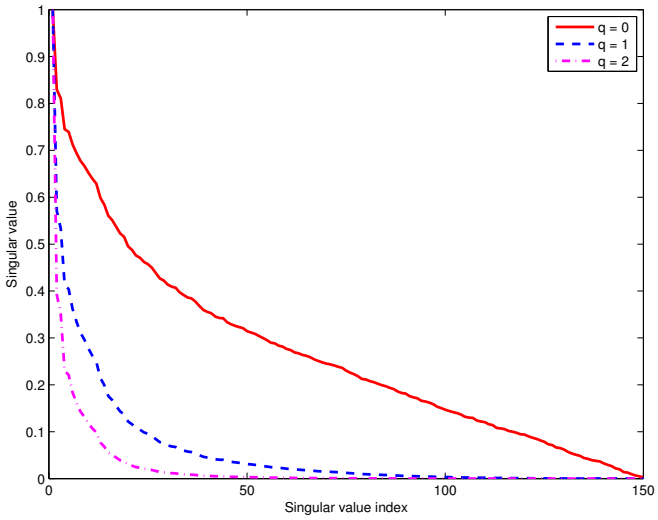


Figure 3.3: Effect of the power iteration on the decay of singular values for a frequency slice $\widehat{\mathbf{P}}_{150\times150}$. Algorithm 1 is equivalent to $q = 0$.

Algorithm 2 Power Iteration

Input: $\widehat{\mathbf{P}}_{n_r\times n_s}$, a target rank k and a number of iterations parameter q
Output Orthonormal matrix $\widehat{\mathbf{Q}}$ whose range approximates the range of $\mathbf{P}$

1. Draw a random Gaussian matrix $\widehat{\mathbf{W}}_{n_r\times k}$.
2. Form $\mathbf{Y}_0 = \widehat{\mathbf{P}}\widehat{\mathbf{W}}$.
3. Construct orthonormal matrix $\widehat{\mathbf{Q}}_0$ by computing $\mathbf{QR}$. factorization on $\mathbf{Y_0}$.
4. **for** $j = 1 \dots q$
5. Form $\widetilde{\mathbf{Y}} = \widehat{\mathbf{P}}^*\mathbf{Q_{j-1}}$ and compute its $\mathbf{QR}$ factorization $\widetilde{\mathbf{Y}}_j = \widetilde{\mathbf{Q}}_j\widetilde{\mathbf{R}}_j$.
6. Form $\mathbf{Y}_j = \widehat{\mathbf{P}}^*\widetilde{\mathbf{Q}}_j$ and compute its $\mathbf{QR}$ factorization $\mathbf{Y}_j = \widehat{\mathbf{Q}}_j\mathbf{R}_j$.
7. **end**
8. $\mathbf{Q} = \mathbf{Q_q}$.

3.3 Lowering the cost of matrix probing

For our application, the computational bottleneck in Algorithms 1 and 2, is in sampling the data matrix, i.e., $\mathbf{Y}_{m\times(k+p)} = \widehat{\mathbf{P}}\widehat{\mathbf{W}}$. Generating each sample will cost $O(mnk)$ operations, in addition to, a single pass over the data. To reduce the computational cost, we look into using two approaches. First, we look into utilizing fast transforms, such as the Fourier transform to sample the data matrix. In the second approach, we propose approximating the data matrix, by picking up random sources (columns) of the data matrix $\widehat{\mathbf{P}}$. This will help in reducing the number of operations, and the amount of data to be transferred, in order to perform the approximation.

Fast transforms

As proposed by (Halko et al. (2011)), we can replace the Gaussian dense matrix by the sub-sampled Fourier transform $SRFT$, which reduces the computational cost, of sampling the data to $O(mnlog(k))$ operations. The $SRFT$ matrix is constructed as follows:

$$\widehat{\mathbf{W}}_{SRFT} = \sqrt{\frac{n_s}{k}}\mathbf{DFR} \tag{3.8}$$

Where

- $\mathbf{D}$ is an $n_s \times n_s$ diagonal matrix, whose non-zeros elements, are randomly picked from a uniform unit circle,
- $\mathbf{F}$ is an $n_s \times n_s$ unitary discrete Fourier transform (DFT), and
- $\mathbf{R}$ is an $n_s \times k$ matrix, with k columns drawn randomly without replacement, from the columns of the $n_s \times n_s$ identity matrix

If we replace the random Gaussian matrices used in Algorithms 1 and 2, with the $SRFT$ matrices (see Algorithm 3), we reduce the computational cost, of the first stage of the low-rank approximation. However, we still suffer from the communication cost of transferring large volumes of information for each pass over the data and the cost of on-the-fly interpolation in the case of missing sources.

3.3.1 Data sampling via Dirac basis

To reduce the communication and on-the-fly interpolation bottlenecks, we propose to 'sample' the data matrix to obtain $\mathbf{Y}$ with randomly selected

Algorithm 3 Sub-sampled Fourier transform

Input: $\mathbf{P}_{n_r \times n_s}$, a target rank k and an over-sampling parameter p
Output Orthonormal matrix $\widehat{\mathbf{Q}}$ who's range approximates the range of $\widehat{\mathbf{P}}$

1. Draw a random **SRFT** matrix. $\widehat{\mathbf{W}}_{n_r \times (k+p)}$.
2. Form $\mathbf{Y}_{nr_\times (k+p)} = \widehat{\mathbf{P}}\widehat{\mathbf{W}}$.
3. Construct and return orthonormal matrix $\widehat{\mathbf{Q}}_{n_r \times (k+p)}$ by computing **QR**. factorization of $\mathbf{Y}$.

vectors from the identity basis. This corresponds to data matrices that are formed from seismic data with sources missing at random locations. Not only does this rid ourselves from having to form the fully-sampled data matrix by interpolation but it also reduces the cost of passing over the full data matrix. In this case we replace equation 3.1 with

$$\mathbf{Y} = \widehat{\mathbf{P}}\mathbf{R}$$

Where $\mathbf{R}_{n_s \times k}$ is a restriction matrix, drawn randomly without replacement, from the columns of an $n_s \times n_s$ identity matrix. To show that our choice of Dirac basis is appropriate in approximating the data matrix, we compute the mutual coherence between the singular vectors of a typical $\widehat{\mathbf{P}}$ and the vectors from the identity basis. The mutual coherence parameter μ (Herrmann and Hennenfent (2008), Herrmann and Hennenfent (2008), and Herrmann et al. (2011)) is given by

$$\mu(\mathbf{U}, \mathbf{R}) = \max_{1<i,j<n} \frac{< \mathbf{u}_i, \mathbf{r}_j >}{\|\mathbf{u}_i\|_2 \|\mathbf{r}_j\|_2}. \tag{3.9}$$

where, $\mathbf{u}_i$ and $\mathbf{v}_i$ representing the ith column of the first and second matrices, respectively. The more incoherent these two basis are w.r.t. each other, the better approximation we can achieve. To get an idea of how incoherent the Dirac basis w.r.t. the singular vectors of $\widehat{\mathbf{P}}$, we compute the mutual coherence for the Dirac basis as well as for $\widehat{\mathbf{W}}$ given by the Gaussian vectors. We make this comparison because the Gaussian vectors have the smallest mutual coherence. The results of this exercise are summarized in Figure 3.4 and show that the Gaussian vectors have the lowest mutual coherence, as expected followed by the Dirac basis and the Fourier basis. From these results, we can conclude that we can use missing shots instead of sampling the data matrix using the dense Gaussian basis allowing us to work with missing data e.g., the case of marine data (Li et al. (2011)).

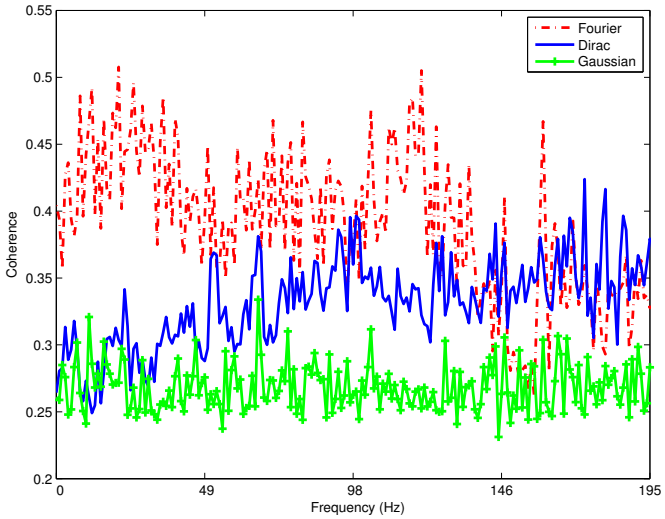


Figure 3.4: Coherence of the singular vectors of the data matrix with Fourier, Dirac, and Gaussian basis for all frequencies.

3.3.2 Computational costs

The goal of using randomized low-rank approximation is not to reduce the number of operations, but rather the goal is the pass efficiency, which counts the number of times the data needs to be loaded into the in-core memory. Classical approximation methods including SVD, need at least $O(k)$ passes over the data. In the previously described randomized algorithms, the number of these passes over the complete data, is significantly reduced. Both of, Algorithms 1 and 3, require only a single pass over the data. The advantage of using fast transforms,e.g. SRFT, is that the number of computations required to compute the orthonormal matrix $\widehat{\mathbf{Q}}$, is reduced from $2O(mnk)$ to $mnlog(k) + O(mnk)$ operations. Algorithm 2, requires additional q passes over the data to handle slow decaying singular values. Finally, our proposed modification to stage A, reduces the communication cost, where we only need to load random columns of the data into memory, instead of cycling through the whole data matrix. However stage B require an additional pass over the data and $O(mnk)$ operations to compute the factorizations. These costs are summarized in table 3.1 for all the mentioned algorithms.

	Number of operations	Number of passes over the data
Algorithm 1	$\mathcal{O}(2mnk$	1)
Algorithm 2	$\mathcal{O}(2qmnk + nk)$	$q + 1$
Algorithm 3	$\mathcal{O}mnlog(k) + (mnk)$	1
Stage B	$\mathcal{O}(2mnk)$	1
Classical SVD	$\mathcal{O}(mn)$	k

Table 3.1: Summery of the cost of applying the randomized approximation algorithms, in terms of number of operations to compute the factorization, and the number of passes over the data .

3.4 Examples

We now look at whether these randomized approximation algorithms actually work for our data by presenting two examples. First, we approximate a frequency slice exhibiting a fast decay in its singular values, and in the second example, we look at the approximation of a frequency slice with slow decay in its singular values. In both cases, we compute the approximated singular values, and compare them to the actual ones generated using the classical SVD method. We will also look at the behavior of the spectral approximation error, for each method. The approximation error is computed using:

$$e_{_k} = \|(\mathbf{I} - \widehat{\mathbf{Q}}\widehat{\mathbf{Q}}^*)\widehat{\mathbf{P}}\|_s, \tag{3.10}$$

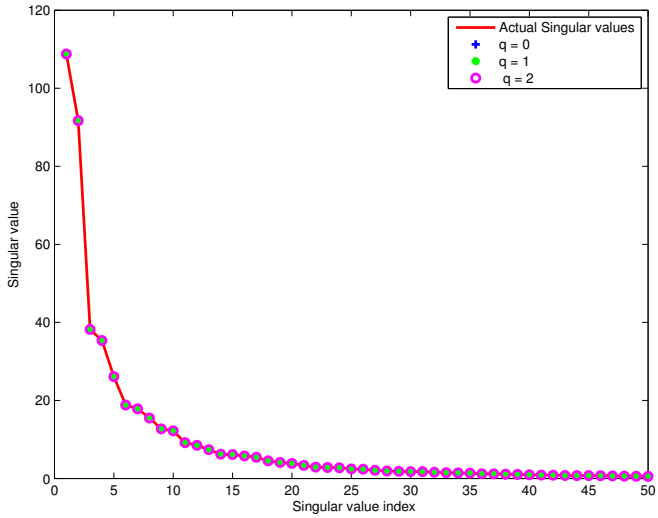
where e_k, is the approximation error,k is the rank, and $\widehat{\mathbf{Q}}$, is the orthonormal matrix –generated using the first stage of the randomized algorithms– which approximates the range of the data matrix $\widehat{\mathbf{P}}$.

Data matrix with fast decay in its singular values . In this example, we will approximate a frequency slice from our synthetic dataset, consisting of $n_r = n_s = 150$, at a frequency of 5Hz. Figure 3.5(a), shows the estimated singular values, using the different randomized algorithms described in this text. The approximation was done, using a rank of 50 columns (sources), and as noted from the figure, all of the estimated singular values match almost exactly the actual singular values. We observe from Figure 3.5(b), how the power iteration achieves much better approximation error, for each q iteration compared to $q = 0$, which is the most basic algorithm.

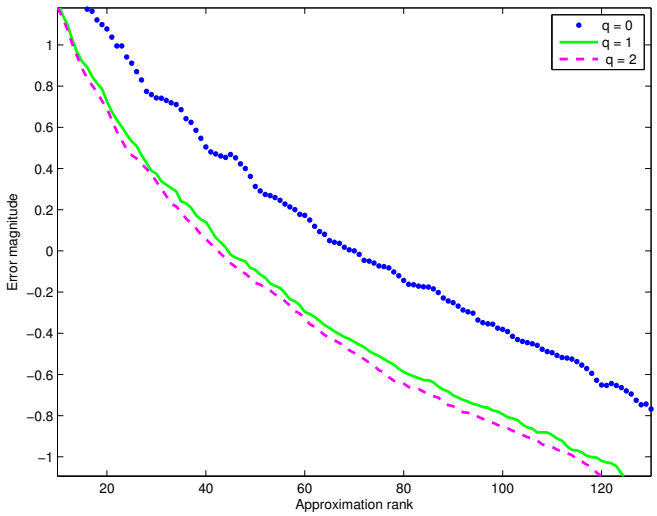
Data matrix with slowly decaying singular values. In this example we will approximate a frequency slice from the same synthetic dataset, at a frequency of 100Hz. The decay is much slower than the previous example, and therefore it is more difficult to approximate. In Figure 3.6(a), we notice how the first algorithm, where $q = 0$, generated an inaccurate estimation of the singular values. As we increase the number of iterations q, we get better approximations of the actual singular values, until we have the highest accuracy at $q = 2$. Figure 3.6(b), demonstrates how, the basic algorithm, achieved a poor approximation error, compared to the power iteration method, at $q = 1$ and $q = 2$.

Changing the sampling method. Figures 3.7(a) and 3.7(b), show the effect of changing the sampling method (i.e. simulations shots, missing shots or SRFT), on the quality of the approximation. The results are shown for Algorithm 2, with $q = 0$ in the first plot, and $q = 2$ in the second plot. When $q = 0$, all three sampling methods, provide very close approximation results. However, using simultaneous shots provides the best approximation quality, and the missing shots (Dirac basis), provides the poorest approximation quality amongst the 3 methods. As we increase the number of iterations q, the results for are all 3 methods, are close to each other, as noted from the second figure.

From the previous examples in this section, we observe how the quality of our low-rank approximation is dependent on the decay of singular values. The faster the decay, the less samples and passes over that data we need for the approximation. However, there are more challenging cases, where the singular values exhibit almost no decay. In such cases, the power method is not effective enough, as it will require more passes over the data, with a small effect on the approximation (see Figure 3.8). As mentioned previously, we are trying to avoid the communication bottleneck, and therefore would like to keep the number of passes over the data at a minimum. To address this issue, we suggest using a special type of matrix representation, know as Hierarchal Semi-Separable matrix (HSS) representation, in the next section.

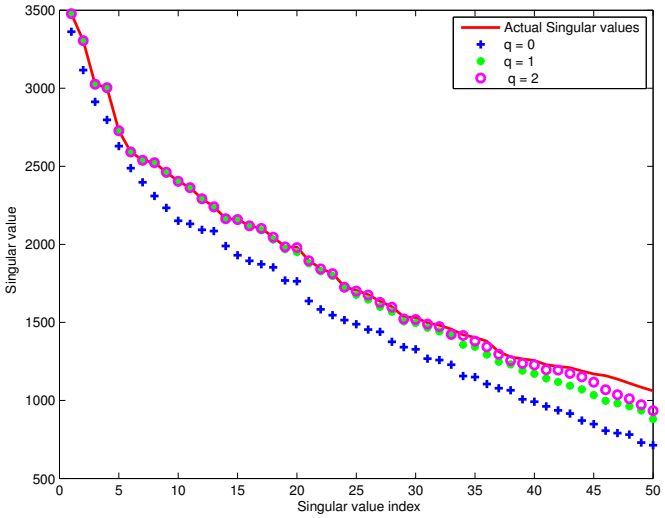


(a)

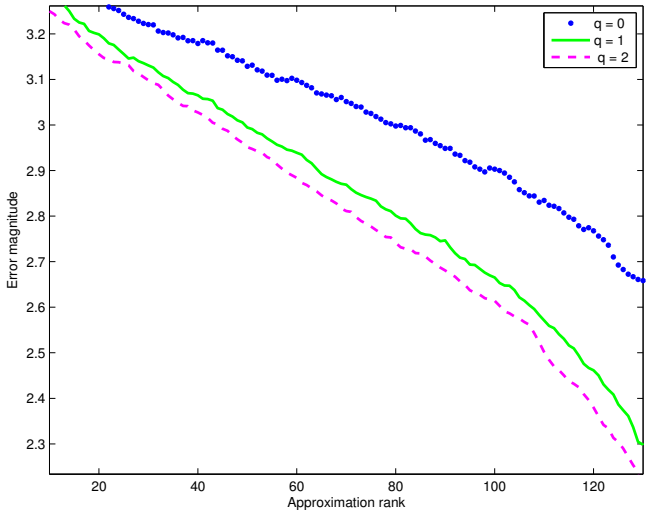


(b)

Figure 3.5: Singular value decay of a frequency slice, of $n_r = n_s = 150$ and $\omega = 5$Hz. (a) Approximation of the singular values, and (b) the behavior of the spectral approximation error.

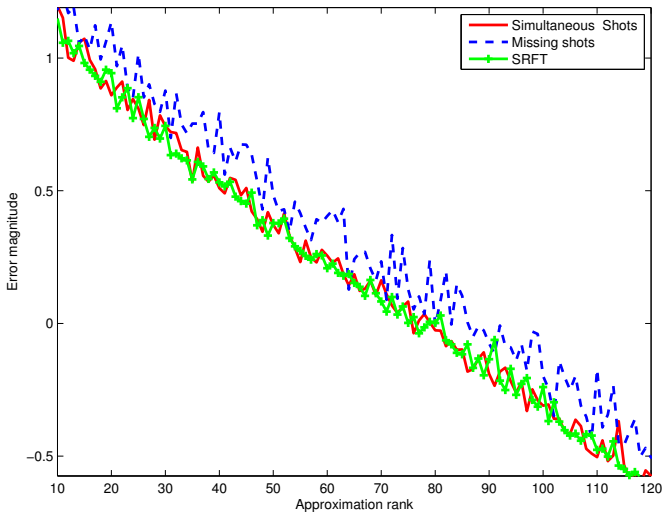


(a)

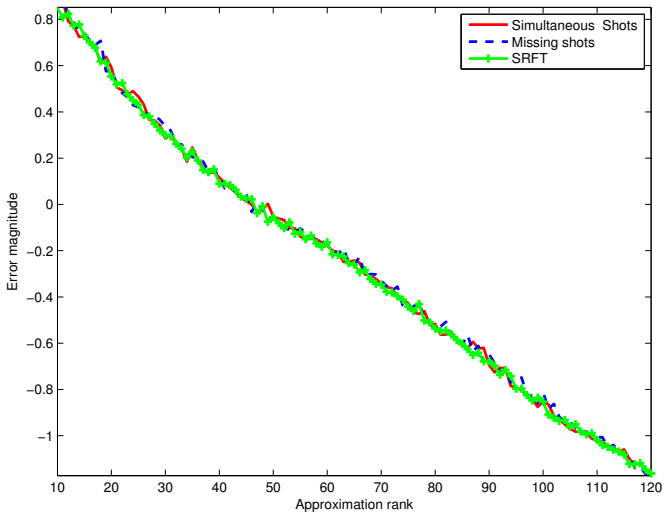


(b)

Figure 3.6: Singular value decay of a frequency slice, of $n_r = n_s = 150$ and $\omega = 100$Hz. (a) Approximation of the singular values, and (b) the behavior of the spectral approximation error.



(a)



(b)

Figure 3.7: Sampling methods (simulations, Fourier, and random shots), a) approximation error using $q = 0$, and b) approximation error at $q = 2$

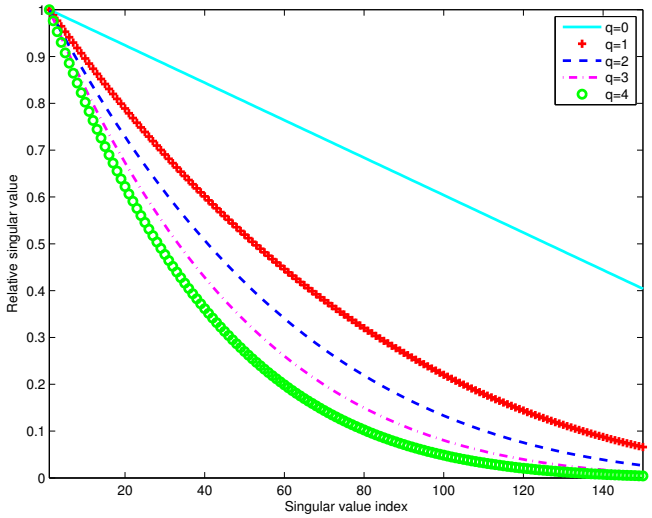


Figure 3.8: Extremely slow decay of singular values, power iteration isn't effective enough.

Chapter 4

Hierarchically Semi-Separable Matrix Representation

Hierarchically Semi-Separable Matrix Representation (HSS), defined in (Chandrasekaran et al. (2006)), provides a way of representing structured dense matrices in terms of lowrank sub-matrices, with the aim of reducing the cost of linear algebraic operations e.g. reducing the cost of matrix-vector products from $O(n^2)$ to $O(n)$. As part of the HSS representation, matrices are recursively repartitioned into high-rank diagonal and low-rank off-diagonal matrices. SVDs are carried on the off diagonals while the high-rank diagonal matrices are recursively partitioned to some user defined level, which depends on the desirable compression and accuracy. With this decomposition, HSS is able to carry out fast matrix operations. A 2×2 block partitioning of an input matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{1;1,1} & \mathbf{A}_{1;1,2} \\ \mathbf{A}_{1;2,1} & \mathbf{A}_{1;2,2} \end{pmatrix},$$

where the subscripts represent: the partition level, the row number, and the column number, respectively. The off diagonal submatricies are decomposed into their low-rank approximations, e.g by using SVD to give us the first level of HSS partitioning,i.e., we have

$$\mathbf{A} = \begin{pmatrix} \mathbf{D}_{1;1,1} & (\mathbf{USV}^*)_{1;1,2} \\ (\mathbf{USV}^*)_{1;2,1} & \mathbf{D}_{1;2,2} \end{pmatrix},$$

where $\mathbf{D}$ is a diagonal sub-matrix and the factorization $\mathbf{USV}^*$ correspond to the singular value decompositions of the off-diagonal submatrices. In the next HSS iteration, the high-rank matrices are partitioned using the same

HSS partitioning scheme,i.e., now we have

$$\mathbf{A} = \begin{pmatrix} \begin{pmatrix} \mathbf{D}_{2;1,1} & (\mathbf{USV}^*)_{2;1,2} \\ (\mathbf{USV}^*)_{2;2,1} & \mathbf{D}_{2;2,2} \end{pmatrix} & (\mathbf{USV}^*)_{1;1,2} \\ (\mathbf{USV}^*)_{1;2,1} & \begin{pmatrix} \mathbf{D}_{2;1,1} & (\mathbf{USV}^*)_{2;1,2} \\ (\mathbf{USV}^*)_{2;2,1} & \mathbf{D}_{2;2,2} \end{pmatrix} \end{pmatrix}.$$

HSS partitioning is based on based on a specific hierarchical partitioning of an index vector, $I = [1 \cdots n]$, where n is the number of columns in the input matrix. One form of this hierarchical partitioning is based on binary-trees, where each node in the tree can have at most 1 parent and a maximum of 2 children. An example of a three level binary tree, is show in Figure 4.1. Based on the presented tree, we can divide an input matrix e.g. $\widehat{\mathbf{A}}_{400\times400}$ into an HSS representation (Chandrasekaran et al. (2006)) as follows:

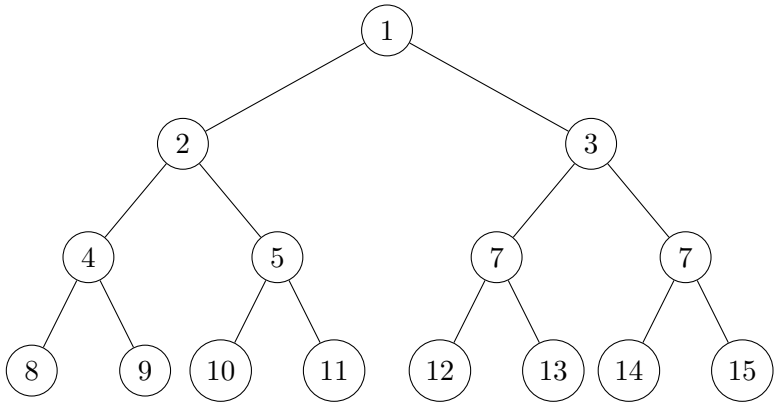


Figure 4.1: 3 level Binary Tree (adapted from Chandrasekaran et al. (2006))

- Level 0 : $I_1 = I = [1 \cdots 400]$
- Level 1 : $I_2 = [1 \cdots 200]$, $I_3 = [201 \cdots 400]$
- Level 2 : $I_4 = [1 \cdots 100]$, $I_5 = [101 \cdots 200]$, $I_6 = [201 \cdots 300]$, $I_7 = [301 \cdots 400]$
- Level 2 : $I_8 = [1 \cdots 50]$, $I_9 = [51 \cdots 100]$, $\cdots$, $14 = [301 \cdots 350]$, $I_15 = [351 \cdots 400]$

Where each diagonal sub-matrix is formed by taking the indices stored in the leaf nodes of the binary tree according to

$$\mathbf{D}_j = \mathbf{A}(I_j, I_j). \tag{4.1}$$

Here, j is the index of the leaf node. Off-diagonal blocks are defined using sibling pairs at each level of the binary tree as follows:

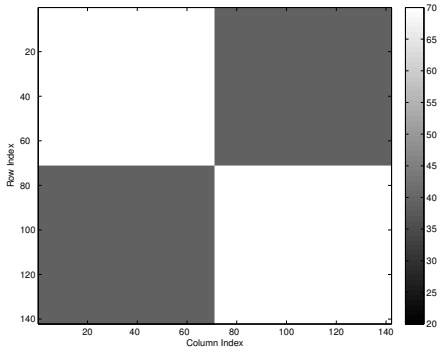
$$\mathbf{A}_{v1,v2} = \mathbf{A}(I_{v1}, I_{v2}) \;\; and \;\; \mathbf{A}_{v2,v1} = \mathbf{A}(I_{v2}, I_{v1}). \tag{4.2}$$

4.1 HSS matrices and randomized SVD

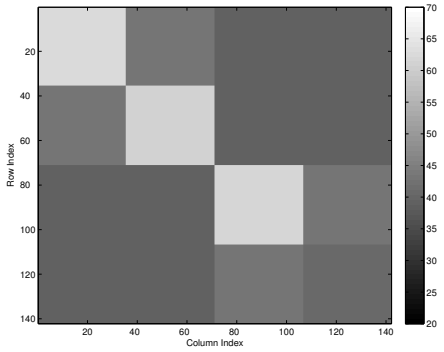
Within the HSS transformation scheme (defined in Chandrasekaran et al. (2006)) we incorporate the randomized power iteration SVD Algorithm 2, to compute the SVD of the off-diagonal blocks. In this case, the cost of probing the submatrices is cheap and requires only few passes over the data. Using the power iteration will help in increasing the decay of the singular values for the low-rank submatrices, hence providing a higher quality approximation. To determine the proper rank for the matrix probing, we use Algorithm 4.2 from Halko et al. (2011).

4.2 Examples

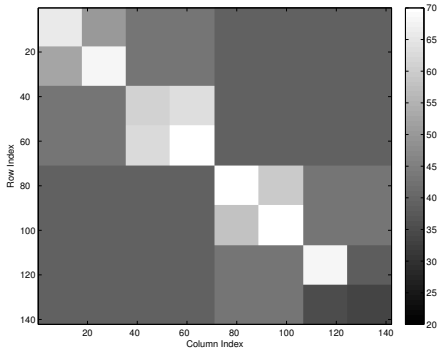
To demonstrate the effectiveness of the HSS factorization,we will consider a monochromatic frequency slice at a frequency of 100 Hz which we approximate using a three-level HSS representation including the adaptive rank selection. Before we apply this algorithm, we first verify the anticipated rank behavior of the HSS submatricies as shown in Figure 4.2, which shows that the off diagonal submatrices are indeed of lower-rank. his justifies the use of HSS on high-frequency data matrices. As we can see from Figure 4.3, the HSS-based factorization attains a better approximation over the randomized SVD, which is reflected in the SNR of the approximation.



(a)



(b)



(c)

Figure 4.2: HSS partitioning of a frequency slice at $\omega = 100$Hz, the colours corresponds to the rank of the sub matrix (white is high-rank, black is low-rank) (a) First level of HSS partitioning (b) second Level of HSS partitioning (c) third level of HSS partitioning.

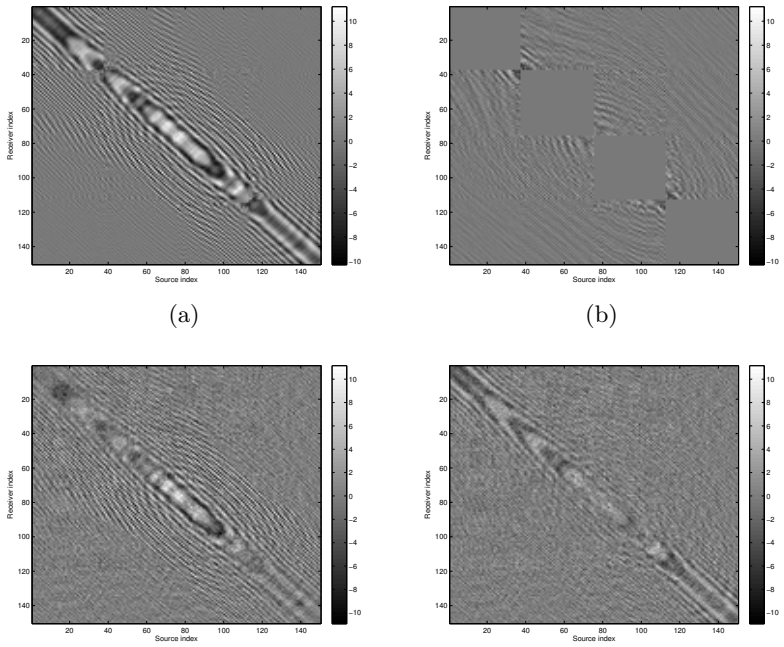


Figure 4.3: Approximation results of a monochromatic frequency slice at frequency of 100 Hz and $n_s = n_r = 150$. (a) HSS approximation with SNR = 11dB, (b) the difference between the HSS approximation and the original data matrix, (c) SVD approximation of the same frequency slice with SNR = 11dB, and (d) the difference of the SVD approximation with the original data matrix

Chapter 5

Results and Observations

5.1 Application to synthetic and real data

To establish the performance of our method, we compare the output of EPSI (Lin and Herrmann, 2011) with the output of EPSI using low-rank approximations for the data matrix. Depending on the on the ratio between the largest singular value of a particular data matrix and the largest singular amongst all data matrices, we either proceed by applying the randomized SVD or we first apply the HSS partitioning prior to the computation of the randomized SVDS on the off diagonals. We conduct two experiments and we fix the subsampling ratios to $\delta = [1/2, 1/5, 1/8, 1/12]$.

5.1.1 Synthetic data

In this example, we use data modeled from a velocity model that consists of a high-velocity layer, which represents salt, surrounded by sedimentary layers and a water bottom that is not completely flat (see Herrmann et al., 2007). Using an acoustic finite-difference modeling algorithm, 150 shots with 150 receivers are simulated on a fixed receiver spread. A shot record for the first 2 s with surface-related multiples is plotted in Figure 5.1. Refer to Figure 2.6, for the singular values for the data matrices associated with this synthetic data set.

Results of our adaptive rank selection on this data sets are plotted in Figure 5.2 and show that most of the available ranks are assigned to data matrices with frequencies that lie in the seismic band. Given these ranks, we compute low-rank approximations using our randomized SVD algorithm for $q = 2$ and with HSS for the matrices that have the highest spectral norm. With now solve the EPSI problem for $\delta = 1/12$ and the results are plotted in Figure 5.3. Comparison between the EPSI result obtained the full data matrix and its low-rank approximation shows that our method is able to get a satisfactory judged by the fact that the difference plot in Figure. 5.3(c) contains relatively little energy. Remember that this is a synthetic example, yielding a high-quality estimation for the surface-free Green's function. This

observation was also made by van Groenestijn and Verschuur (2009); Lin and Herrmann (2010).

For completeness, we included table 5.1 with SNRs for dimensionality-reduced EPSI experiments carried out for $\delta = [1/2, 1/5, 1/8, 1/12]$. As expected, the SNRs computed with respect to the output of EPSI carried out with the full data matrix, show a decrease in SNR for decreasing subsampling ratios.

5.1.2 Gulf of Suez

We carried out the same set of experiments for $q = 2$ on a Gulf of Suez dataset for the same subsampling ratios. A shot record for the first 2 s of this data set is plotted in Figure 5.4. The singular values and assigned ranks are included in Figure 5.5 and ranks assigned to a broader range of frequencies consistent with the more complex and more broad-band nature of this real data set. As before, we carry out EPSI for the complete and low-rank approximated data matrices and the results are shown in Figure 5.6 for $\delta = 1/12$. While both results are comparable, the difference plots contains some lost energy for areas in the shot record that have high curvature. This is expected because high-curvature events lead higher ranks. The SNRs in table 5.2 are not as high but still reasonable. Again, the SNRs decrease with decreasing subsampling ratios.

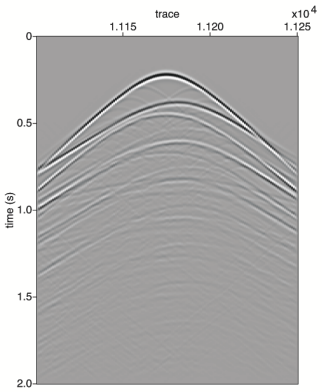


Figure 5.1: Shot gather from synthetic dataset including surface-related multiples.

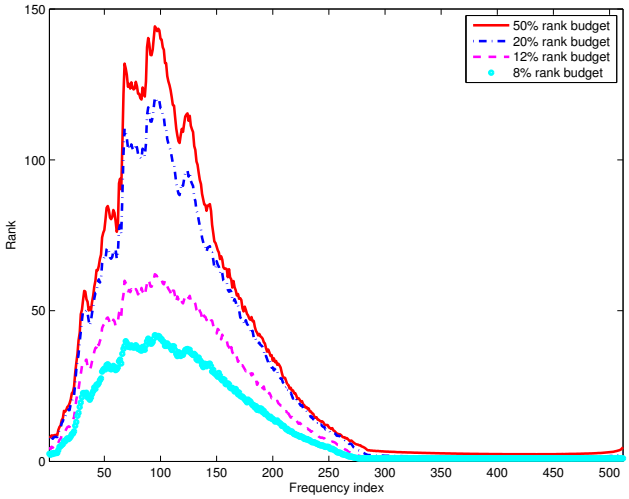


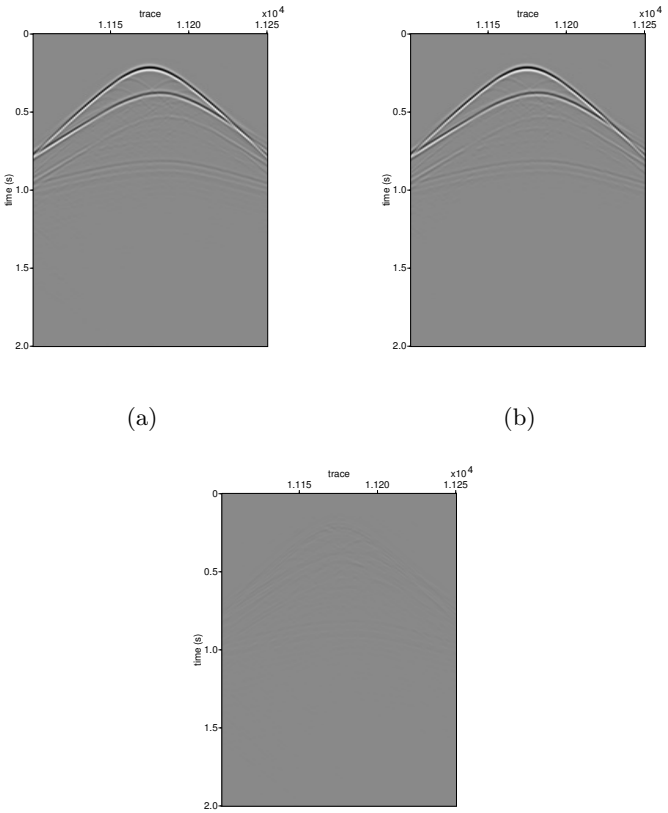
Figure 5.2: Approximation rank distributions for varying rank budgets, in the synthetic case.

Subsample ratio δ	1/2	1/5	1/8	1/12
Recovery error (dB)	44	30	18	13
Speed up ($\times$)	2	5	8	12

Table 5.1: Results of estimating the surface-free Green's function using the different subsampling ratios (synthetic dataset).

5.2 Discussion and outlook

The presented method is exciting for the following reasons. First, we reduce the storage and multiplication costs by roughly a factor of δ at a small up-front cost consisting of $2-3$ passes through all data, a QR factorization, and a singular-value decomposition on the dimensionality reduced system, all of which can be carried out in parallel. The resulting matrix factorization has low memory imprint, leads to fast matrix multiplies, and removes the requirement of passing through all data for each application of the data matrix during the inversion. This feature potentially removes one of the major challenges of extending estimation of primaries by sparse inversion to 3D. Note, however,that this extension needs a low-rank factorization of tensors representing wavefields in more than two dimensions (see e.g. Kolda



(a) (b)

(c)

Figure 5.3: EPSI results from the synthetic dataset : (a) Estimated Green's function using the full dataset, (b) estimated Green's function using $\delta = 1/12$ and (c) difference between the two estimates.

Subsample ratio δ	1/2	1/5	1/8	1/12
Recovery error (dB)	30	27	17	12
Speed up ($\times$)	1.6	2	3.5	5.7

Table 5.2: Summarizing results of estimating the surface-free Green's function using the different sub-sampling ratios.

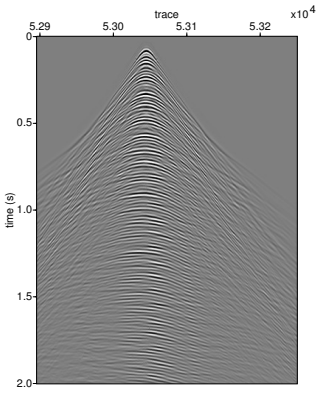
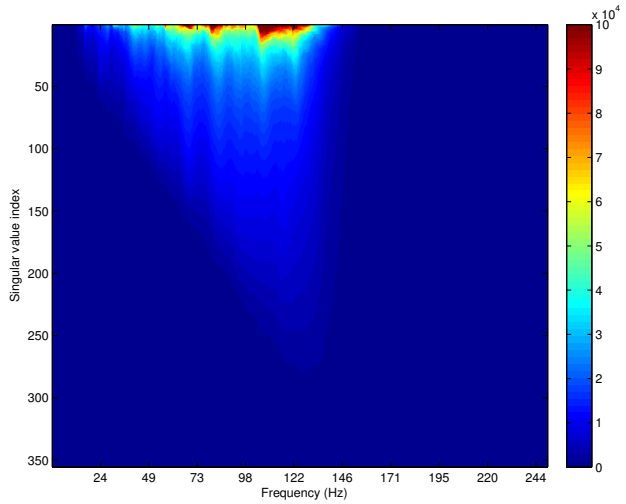
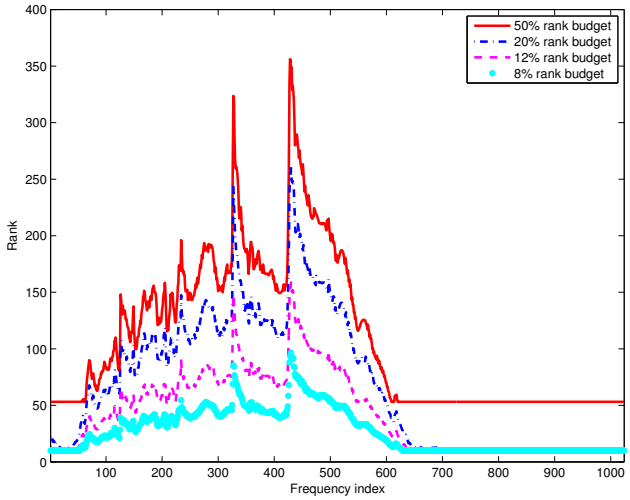


Figure 5.4: Shot gather from the Gulf of Suez dataset including surface-related multiples.

and Bader, 2009, for recent developments on this topic). Second, there are connections between matrix probing and simultaneous sourcing during acquisition of marine data (Wason et al., 2011) or during imaging and full-waveform inversion (Herrmann et al., 2009). This opens the possibility to further speed up our algorithms (see e.g. Herrmann, 2010; Tristan van Leeuwen and J.Herrmann) or to work directly with simultaneously acquired data. Third, because the singular vectors of the data matrix are incoherent with the Dirac basis, we can limit the need of interpolating the data to only once as part of the second stage during which the singular-value decomposition is conducted. In case the locations of the missing shots are sufficiently randomly distributed, we showed that it is no longer necessary to interpolate the data as part of the matrix probing. Instead, we can consider data with randomly missing shots as the result of the matrix probing. Needless to say, this could lead to significant cost savings. Fourth, the proposed algorithm



(a)



(b)

Figure 5.5: Gulf of Suez data. (a) Singular values of the gulf of Suez dataset, (b) distribution of the approximation rank budgets for varying sub-sampling ratios δ.

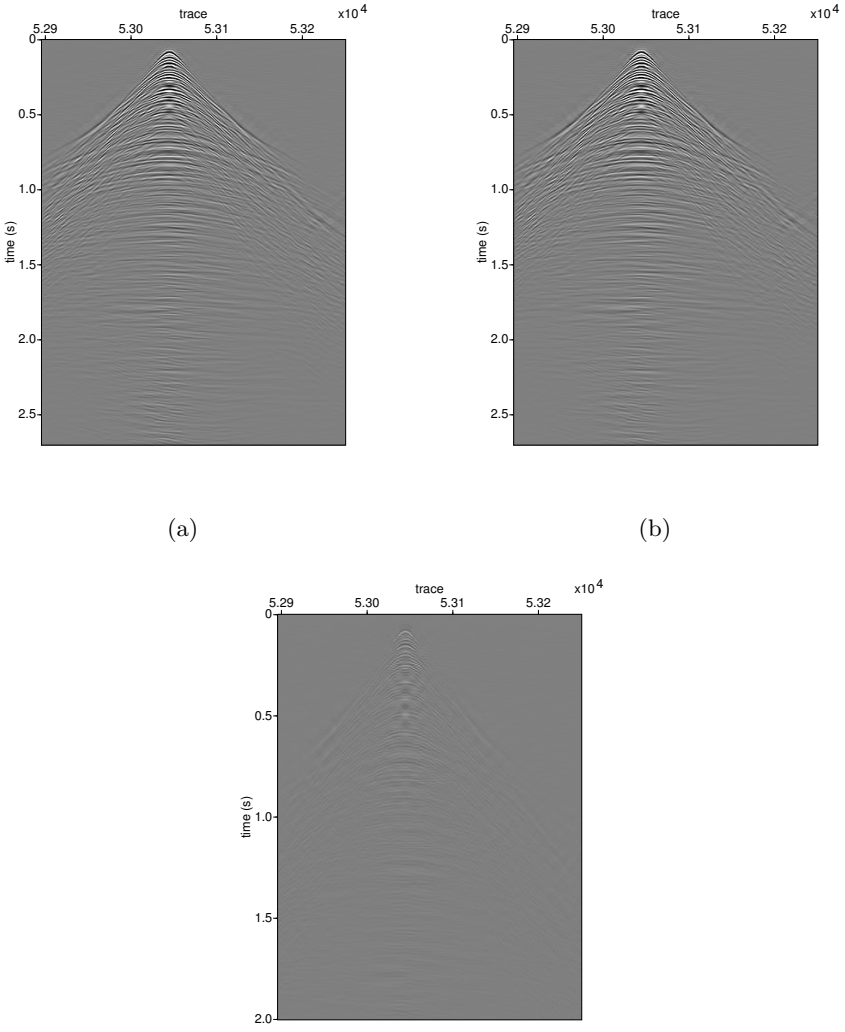


Figure 5.6: EPSI results from the gulf of Suez dataset : (a) Estimated Green's function using the full dataset, (b) Estimated Green's function using $\delta = 1/12$ and (c) difference between the two estimates.

is relatively simple and requires matrix-free application (read 'black-box' implementations of SRME (Verschuur et al., 1992; Berkhout and Verschuur, 1997; Weglein et al., 1997))) of the data matrix only. Fourth, our low-rank approximations of the data matrix allow us to leverage recent extensions of compressive sensing to matrix-completion problems (Candes and Recht, 2009; Gandy et al., 2011) where matrices or tensors are reconstructed from incomplete data (read data with missing traces). In these formulations, data is regularized solving the following optimization problem

$$\widetilde{\mathbf{X}} = \arg\min_{\mathbf{X}} \|\mathbf{X}\|_* \quad \text{subject to} \quad \|\mathcal{A}(\mathbf{X}) - \mathbf{b}\|_2 \leq \sigma, \tag{5.1}$$

with $\|\cdot\|_* = \sum |\lambda_i|$ the nuclear norm summing the magnitudes of the singular values (λ) of the matrix $\mathbf{X}$. Here, $\mathcal{A}(\cdot)$ a linear operator that samples the data matrix. It can be shown that this program is a convex relaxation of finding the matrix $\mathbf{X}$ with the smallest rank given incomplete data. Finally, low-rank approximations for tensors were recently proposed by Oropeza and Sacchi (2010) for seismic denoising. Fifth, the singular vectors of our low-rank approximation can also be used in imaging or full-waveform inversion Habashy et al. (2010).

5.3 Conclusions

Data-driven methods—such as the estimation of primaries by sparse inversion—suffer from the 'curse of dimensionality' because these methods require repeated applications of the data matrix whose size grows exponentially with the dimension. In this paper, we leverage recent developments in dimensionality reduction that allow us to approximate the action of the data matrix via a low-rank matrix factorization based on the randomized singular-value decomposition. Combination of this method with hierarchical semi-separable matrix representations enabled us to efficiently factor high-frequency data matrices that have high ranks. The resulting low-rank factorizations of the data matrices lead to significant reductions in storage and matrix multiplication costs. The reduction in costs for the low-rank approximations themselves are, by virtue of the randomization, cheap and only require a limited number of applications of the full data matrix to random vectors. This operation can easily be carried out in parallel using existing code bases for surface-related multiple prediction and can lead to significant speedups and reductions in memory use. Because the singular vectors of the data matrices are incoherent with the Dirac basis, matrix probing by Gaussian vectors that require on-the-fly interpolations can be replaced by matrix probings

consisting of data with missing shots. As a consequence, the number of interpolations is reduced to only one and this could give rise to a significant improvement in the performance of the inversion, which typically requires several applications of the data matrix.

5.4 Acknowledgments

We are grateful to Charles Jones from BG for providing us with the BG Compass velocity model. We would also like to thank the authors of SPOT (`http://www.cs.ubc.ca/labs/scl/spot/`), SPGℓ_1 (`http://www.cs.ubc.ca/labs/scl/spgl1/`), and CurveLab (`curvelet.org`). This paper was prepared with Madagascar (`rsf.sf.net`) and was in part financially supported by by the CRD Grant DNOISE 334810-05. The industrial sponsors of the Seismic Laboratory for Imaging and Modelling (SLIM) BG Group, BP, BGP, BP, Chevron, ConocoPhillips, Petrobras, PGS, Total SA., and WesternGeco are also gratefully acknowledged.

Bibliography

A. J. Berkhout and D. J. Verschuur. Estimation of multiple scattering by iterative inversion, part I: theoretical considerations. *Geophysics*, 62(5): 1586–1595, 1997.

E. Candes and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2009.

S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals. A fast solver for hss representations via sparse matrices. *SIAM J. Matrix Analysis Applications*, 29(1):67–81, 2006. URL `http://dblp.uni-trier.de/db/journals/siammax/siammax29.html#ChandrasekaranDGLP06`.

J. Chiu and L. Demanet. Sublinear randomized algorithms for skeleton decompositions. 2012.

E. Dedem. *3D surface-related multiple prediction*. PhD thesis, 2002.

L. Demanet, P.-D. Letourneau, N. Boumal, H. Calandra, J. Chiu, and S. Snelson. Matrix probing: a randomized preconditioner for the wave-equation hessian. *Journal of Apllied Computational Harmonic Analysis*, 32:155–168, 2012.

S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011. URL `http://stacks.iop.org/0266-5611/27/i=2/a=025010`.

T. M. Habashy, A. Abubakar, G. Pan, and A. Belani. Full-waveform seismic inversion using the source-receiver compression approach. In *SEG Technical Program Expanded Abstracts*, volume 29, pages 1023–1028. SEG, 2010. doi: 10.1190/1.3513021. URL `http://link.aip.org/link/?SGA/29/1023/1`.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53:217–288, May 2011. ISSN 0036-1445.

doi: http://dx.doi.org/10.1137/090771806. URL http://dx.doi.org/10.1137/090771806.

F. J. Herrmann. Randomized sampling and sparsity: Getting more information from fewer samples. *Geophysics*, 75(6):WB173–WB187, 2010. ISSN 6. doi: 10.1190/1.3506147. URL http://link.aip.org/link/?GPYSA7/75/WB173/1.

F. J. Herrmann and G. Hennenfent. Non-parametric seismic data recovery with curvelet frames. *Geophysical Journal International*, 2008.

F. J. Herrmann, U. Boeniger, and D. J. Verschuur. Non-linear primary-multiple separation with directional curvelet frames. *Geophysical Journal International*, 170(2):781–799, 2007. doi: 10.1111/j.1365-246X.2007.03360.x. URL https://www.slim.eos.ubc.ca/Publications/Public/Journals/GeophysicalJournalInternational/2007/herrmann07nlp/herrmann07nlp.pdf.

F. J. Herrmann, Y. A. Erlangga, and T. Lin. Compressive simultaneous full-waveform simulation. *Geophysics*, 74:A35, 2009.

F. J. Herrmann, X. Li, A. Y. Aravkin, and T. van Leeuwen. A modified, sparsity-promoting, Gauss-Newton algorithm for seismic waveform inversion. *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 8138, Sept. 2011. doi: 10.1117/12.893861.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009. doi: 10.1137/07070111X.

X. Li, A. Aravkin, T. van Leeuwen, and F. J. Herrmann. Fast randomized full-waveform inversion with compressive sensing. Tech. rep., University of British Columbia, Vancouver, University of British Columbia, Vancouver, September 2011.

L. Lin, J. Lu, and L. Ying. Fast construction of hierarchical matrix representation from matrixvector multiplication. *Journal of Computational Physics*, 230(10):4071 – 4087, 2011. ISSN 0021-9991. doi: 10.1016/j.jcp.2011.02.033. URL http://www.sciencedirect.com/science/article/pii/S0021999111001227.

T. T. Lin and F. J. Herrmann. Stabilization of estimation of primaries via sparse inversion. In *EAGE Technical Program Expanded Abstracts*. EAGE, EAGE, 2010. URL https://www.slim.eos.ubc.ca/Publications/Public/Conferences/EAGE/2010/lin10EAGEseo/lin10EAGEseo.pdf.

T. T. Lin and F. J. Herrmann. Estimating primaries by sparse inversion in a curvelet-like representation domain. In *Estimating primaries by sparse inversion in a curvelet-like representation domain*. EAGE, EAGE Technical Program Expanded Abstracts, 2011.

T. T. Y. Lin and F. J. Herrmann. Robust estimation of primaries by sparse inversion via ℓ_1 minimization. 2012.

V. E. Oropeza and M. D. Sacchi. A randomized svd for multichannel singular spectrum analysis (mssa) noise attenuation. *SEG, Expanded Abstracts*, 29(1):3539–3544, 2010. ISSN 1. doi: 10.1190/1.3513584. URL `http://link.aip.org/link/?SEGEAB/29/3539/1`.

A. A. Tristan van Leeuwen and F. J.Herrmann. Seismic waveform inversion by stochastic optimization. *International Journal of Geophysics*, 2011.

G. J. A. van Groenestijn and D. J. Verschuur. Estimating primaries by sparse inversion and application to near-offset data reconstruction. *Geophysics*, 74(3):A23–A28, 2009. doi: 10.1190/1.3111115. URL `http://link.aip.org/link/?GPY/74/A23/1`.

T. van Leeuwen, M. Schmidt, M. Friedlander, and F. Herrmann. A hybrid stochastic-deterministic optimization method for waveform inversion. In *A hybrid stochastic-deterministic optimization method for waveform inversion*. EAGE, EAGE Technical Program Expanded Abstracts, 2011.

D. J. Verschuur, A. J. Berkhout, and C. P. A. Wapenaar. Adaptive surface-related multiple elimination. *Geophysics*, 57(9):1166–1177, 1992.

H. Wason, F. J. Herrmann, and T. T. Lin. Sparsity-promoting recovery from simultaneous data: a compressive sensing approach. In *SEG Technical Program Expanded Abstracts*, pages 6–10. SEG, SEG, 04/2011 2011. URL `https://www.slim.eos.ubc.ca/Publications/Public/Conferences/SEG/2011/wason11SEGsprsd/wason11SEGsprsd.pdf`.

A. B. Weglein, F. A. Carvalho, and P. M. Stolt. An iverse scattering series method for attenuating multiples in seismic reflection data. *Geophysics*, 62:1975–1989, 1997.