An open-source framework for the implementation of large-scale integral operators with flexible, modern HPC solutions - Enabling 3D Marchenko imaging by least squares inversion

GEOPHYSICS®

An open-source framework for the implementation of large-scale integral operators with flexible, modern HPC solutions - Enabling 3D Marchenko imaging by least squares inversion

Journal:	*Geophysics*
Manuscript ID	GEO-2020-0796.R2
Manuscript Type:	Advances in seismic multiple reflection processing
Keywords:	3D, distributed systems, extrapolation, least squares, wave propagation
Manuscript Focus Area:	Seismic Migration, Seismic Modeling and Wave Propagation
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
tables.tex	

DOI:10.1190/geo2020-0796.1

DOI: 10.1190/geo2020-0786.1

Matteo Ravasi and Ivan Vasconcelos

An open-source framework for the implementation of large-scale integral operators with flexible, modern HPC solutions - Enabling 3D Marchenko imaging by least squares inversion

(April 30, 2021)

Running head: **Integral operators on HPC**

ABSTRACT

Numerical integral operators of convolution type form the basis of most wave-equation-based methods for processing and imaging of seismic data. As several of these methods require the solution of an inverse problem, multiple forward and adjoint passes of the modeling operator are generally required to converge to a satisfactory solution. This work highlights the memory requirements and computational challenges that arise when implementing such operators on 3D seismic datasets and their usage for solving large systems of integral equations. A Python framework is presented that leverages libraries for distributed storage and computing, and provides a high-level symbolic representation of linear operators. A driving goal for our work is not only to offer a widely deployable, ready-to-use high-performance computing (HPC) framework, but to demonstrate that it enables addressing research questions that are otherwise difficult to tackle. To this end, the first example of 3D full-wavefield target-oriented imaging, which comprises of two subsequent steps of seismic redatuming, is presented. The redatumed fields are estimated by means of gradient-based inversion using the full dataset as well as spatially decimated versions of the dataset as a way to investi-

Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
DOI: 10.1190/geo2020-0786.1

gate the robustness of both inverse problems to spatial aliasing in the input dataset. Our numerical example shows that when one spatial direction is finely sampled, satisfactory redatuming and imaging can be accomplished also when the sampling in other direction is coarser than a quarter of the dominant wavelength. While aliasing introduces noise in the redatumed fields, they are less sensitive to well-known spurious artefacts compared to cheaper, adjoint-based redatuming techniques. These observations are shown to hold for a relatively simple geologic structure, and while further testing is needed for more complex scenarios, we expect them to be generally valid while possibly breaking down for extreme cases (e.g., highly heterogeneous/scattering media).

INTRODUCTION

Multi-dimensional integral operators are ubiquitous in almost every form of wave-equation based processing of multi-channel seismic data; they appear in, e.g. demultiple algorithms, seismic interferometry, and are a key component of many imaging techniques. Their numerical implementation has been investigated in the past, mostly with the aim of applying surface-related multiple elimination (SRME) to 2D (Verschuur, 1992; Dragoset and MacKay, 1993) and 3D datasets (Pica et al., 2005; Baumstein, 2008; Moore and Dragoset, 2008; Dragoset et al., 2010), and more recently in the context of Marchenko redatuming (Thorbecke et al., 2017; Lomas and Curtis, 2019; Brackenhoff et al., 2020). During the development of SRME, two alternative approaches emerged to solve the underlying forward modeling operator described in Verschuur (1992). The most common approach can be seen as the adaptive summation of the first two (or more) terms of a Neumann series (Verschuur, 1992). The other approach involves the inversion of the full modeling operator per frequency component or per frequency-wavenumber component (Dragoset and MacKay, 1993), which bears similarity with the up-down deconvolution of Amundsen (2001) when multi-component data are available in an ocean-bottom acquisition scenario. In both cases the problem is fully separable in the frequency-domain: in the inversion-based approach, this means that each modeling matrix can be explicitly created and inverted independently from those at different frequencies. More recent research has explored the possibility to jointly estimate the multiple-free data as well as the source wavelet and solving the underlying problem by means of a gradient-based iterative solver with additional sparsity constraints (van Groenestijn and Verschuur, 2009; Lopez and Verschuur, 2015b); in practice, the main advantage of the latter approach with respect to the original SRME is that adaptive subtraction is no longer required (nor prior knowledge of the source wavelet is needed) and that

gaps in the acquisition geometries can be also handled within the inversion. On the other hand, the computational cost of the overall process is dramatically increased as the problem loses the frequency decoupling of the original SRME algorithms. The most computationally expensive operation, being multi-dimensional convolution (MDC) between the input dataset and the current estimate of the primary-only data, has to be performed multiple times as part of the optimization process.

In the last decade, a variety of other algorithms that experience similar computational challenges have been developed. All of them contain a modeling operator with one or more MDC operations and require an inverse problem to be solved. Examples of such algorithms, categorized based on the task they aim to solve, are:

- Inversion-based free-surface demultiple: estimation of primaries by sparse inversion (EPSI - van Groenestijn and Verschuur (2009); Lin and Herrmann (2013)) and Closed-loop SRME (Lopez and Verschuur, 2015b,a);
- Deconvolution-based free-surface demultiple (Amundsen, 2001; Ravasi et al., 2015; Boiero and Bagaini, 2020);
- Marchenko-based processing and imaging – among others, redatuming via Marchenko equations (van der Neut et al., 2015; Dukalski and de Vos, 2018), joint redatuming and wavelet estimation via Marchenko equations (Becker et al., 2018), Rayleigh-Marchenko equations (Ravasi, 2017), scattering-Marchenko equations (Vasconcelos and Sripanich, 2019), scattering Rayleigh-Marchenko equations (Vargas et al., 2021), internal multiple attenuation in the data domain (Zhang et al., 2019), and time-lapse redatuming via the Marchenko equations (Haindl et al., 2018, 2021);
- Interferometric redatuming: interferometric redatuming by multi-dimensional decon-

DOI: 10.1190/geo2020-0786.1

volution (MDD – Wapenaar and van der Neut (2010); Wapenaar et al. (2011); van der Neut and Herrmann (2013); Li et al. (2021)) and target-oriented redatuming (van der Neut et al., 2017; Vasconcelos et al., 2017);

- Full-wavefield modeling and inversion (Berkhout and Verschuur, 2011; Davydenko and Verschuur, 2016), joint-migration-inversion (Staal and Verschuur, 2013), and generalized internal multiple imaging (Zuberi and Alkhalifah, 2014; Alkhalifah and Guo, 2019).

As an iterative scheme is required to solve the underlying inverse problem, the adjoint of the modeling operator must also be implemented and special care has to be given to its structure and implementation efficiency. This calls for possible modifications of some of the ideas originally developed for the efficient implementation of 3D SRME to be applicable to the adjoint operators. For example, Lopez and Verschuur (2015a) show that to be able to use the 3D general surface multiple prediction (GSMP) method of Bisley et al. (2005) in closed-loop SRME, a generalization of this algorithm to accommodate for a correlation-type modeling is required. Moreover, whilst adaptive SRME relies on a final step of adaptive summation that can compensate for inaccuracies arising during the prediction step, even more care must be taken when implementing these inversion-based methods such that inaccuracies in the forward and adjoint passes do not lead to instabilities in the inverse process. Trading computational efficiency for accuracy may be required in these scenarios, and on-the-fly (nearest neighbour) interpolation during the evaluation of the multi-dimensional integrals of convolution- (and correlation-) type as currently done in GSMP should be evaluated for applications baed on numerical inversions.

In this paper, we analyze the computational challenges that arise when implementing

DOI: 10.1190/geo2020-0786.1

large multi-dimensional convolutional operators on 3D seismic datasets and present a framework that allows for their evaluation over distributed computer systems. We further discuss how various open-source libraries in the Python ecosystem can be combined to create an efficient end-to-end computational framework that is easy to modify and use for the implementation of a variety of different algorithms. The effectiveness of our implementation is evaluated by applying the forward and adjoint MDC to kernels of increasing size. We show that both operations scale with respect to computational resources and can be deployed over a large variety of HPC architectures. Finally, the proposed computational framework is used to perform full-wavefield target-oriented imaging of a 3D synthetic dataset which contains up to 9801 co-located sources and receivers. First, the Marchenko equations (Wapenaar et al., 2014; van der Neut et al., 2015) are solved to redatum receivers from the acquisition level to a certain subsurface datum. Second, we perform source-side redatuming by means of multi-dimensional deconvolution of the up- and down-going Green's functions estimated by the previous algorithm. Both the full dataset as well as three spatially decimated versions of the dataset are used to illustrate the robustness of these redatuming methods to spatial aliasing in the input dataset for a simple geological model. Complementing the work of Jia et al. (2018b), Lomas and Curtis (2020), and Brackenhoff et al. (2020), our study shows that the Marchenko equations can also be directly inverted by means of an iterative optimization algorithm (e.g., conjugate greadient least-squares (CGLS) - Shewchuk (1994)) in three dimensions (and not only via Neumann iterations). Moreover, similar to Brackenhoff et al. (2019), our example shows how the estimated Green's functions can be further used for target-oriented imaging purposes also in cases where the sampling requirements are mildly violated. Similar to previously published studies performed in two dimensions, we show that downsampling of the input dataset along one of the spatial directions does

introduce coherent noise in the estimated Green's functions and reflectivities; nevertheless, the accuracy of the redatuming processes when it comes to the handling of the complex propagation in the overburden is not fully compromised even in cases where the source and receiver spacing are one and a half larger than a quarter of the dominant wavelength of the input focusing functions.

THEORY

In this work, we are concerned with integral operators of the following kind:

$$g(t, \mathbf{x}_B, \mathbf{x}_A) = \int \int_{\mathbb{R}} K(t - \tau, \mathbf{x}_B, \mathbf{x}_R) f(\tau, \mathbf{x}_R, \mathbf{x}_A) d\mathbf{x}_R d\tau, \tag{1}$$

where $K(t, \mathbf{x}_B, \mathbf{x}_R)$, $f(t, \mathbf{x}_R, \mathbf{x}_A)$ and $g(t, \mathbf{x}_B, \mathbf{x}_A)$ are the *integral kernel operator*, input, and output functions in time-space domain, respectively. Here t is used to indicate time axis, and $\mathbf{x}_B$, $\mathbf{x}_A$, and $\mathbf{x}_R$ refer to the spatial coordinates of a point in 3D, with the latter ($\mathbf{x}_R$) corresponding to the integration support $\mathbb{R}$ of the spatial integral.

As generally neither K nor f have compact support in the time domain (i.e., filters with finite time response), equation 1 can be equivalently, and more efficiently, implemented by using its frequency-domain representation:

$$g(t, \mathbf{x}_B, \mathbf{x}_A) = \mathcal{F}^{-1}_{\omega_{max}} \left(\int_{\mathbb{R}} K(\omega, \mathbf{x}_B, \mathbf{x}_R) \mathcal{F}_{\omega_{max}} \left(f(t, \mathbf{x}_R, \mathbf{x}_A) \right) d\mathbf{x}_R \right), \tag{2}$$

where $K(\omega, \mathbf{x}_B, \mathbf{x}_R)$ is the kernel of the integral operator in the frequency-space domain. In our notation $\mathcal{F}$ and $\mathcal{F}^{-1}$ represent the forward and inverse Fourier transforms, ω is the angular frequency and ω_{max} is used to indicate that the output of the forward Fourier transform is truncated to contain only frequencies where the signal spectrum resides. Depending on the problem at hand, this integral operator is equivalent to the entire modeling

operator (e.g., MDD) or constitutes part of the modeling operator (e.g., Marchenko redatuming). At this point we note that the requirement to move back and forth between time and frequency domains arises under two circumstances: (1) the modeling operator performs any operation to the output function in time domain (e.g., windowing) as in the case of the Marchenko equations discussed in the Application section, (2) a regularization (or preconditioning) term, used to stabilize the inversion, is required to act on the input function in the time domain (e.g., enforcement of time causality as used by van der Neut et al. (2017) in target-oriented imaging applications or sparsifying transform in EPSI – van Groenestijn and Verschuur (2009); Lin and Herrmann (2013)). In the case neither of the above is necessary, equation 2 can be further simplified and $g(\omega, \mathbf{x}_B, \mathbf{x}_A)$ can be calculated entirely in the frequency-domain.

Numerical implementation of equation 2 requires a discretization of the integration domain; to this end, K becomes a 3D tensor of size $(n_{\omega_{max}} \times n_{\mathbf{x}_B} \times n_{\mathbf{x}_R})$, f and g are tensors of size $(n_t \times n_{\mathbf{x}_R} \times n_{\mathbf{x}_A})$ and $(n_t \times n_{\mathbf{x}_B} \times n_{\mathbf{x}_A})$, respectively. Because each frequency can be treated independently, the integral reduces to a matrix-matrix multiplication (or matrix-vector multiplication in the special case of $n_{\mathbf{x}_A} = 1$) repeated for each frequency slice. In this work we will refer to this operation as *batch matrix multiplication*, with one batch being the frequency component. Moreover, the integration step ($d\mathbf{x}_R$) has to be considered: this is especially important for the case of an irregularly sampled domain $\mathbb{R}$, where a receiver-dependant scaling must be applied to the kernel K. In this work we perform a Voronoi tessellation of the receiver geometry, which is used to identify a scaling factor for each receiver that is equivalent to the the areal extent of the corresponding Voronoi cell. The integration step is then accounted for by simply multiplying the columns of K with the corresponding scalar prior to applying batch multiplication (see Figure 1).

The forward and adjoint operations of equation 2 can be written in a compact notation as:

$$\mathbf{g} = \mathbf{F}^H \mathbf{I}^H_{\omega_{max}} \hat{\mathbf{K}} \mathbf{I}_{\omega_{max}} \mathbf{F}\mathbf{f} := \mathbf{C}\mathbf{f} \qquad \mathbf{f} = \mathbf{F}^H \mathbf{I}^H_{\omega_{max}} \hat{\mathbf{K}}^H \mathbf{I}_{\omega_{max}} \mathbf{F}\mathbf{g} := \mathbf{C}^H \mathbf{g}, \tag{3}$$

where $\mathbf{F}$ and $\mathbf{F}^H$ represent the operators performing forward and inverse Fast Fourier Transforms along the time/frequency axis, $\hat{\mathbf{K}}$ is the operator performing batch matrix multiplication with the already scaled kernel, and $\mathbf{f}$ and $\mathbf{g}$ contain the input and output functions rearranged into vectors of size $(n_t n_{\mathbf{x}_R} n_{\mathbf{x}_A} \times 1)$ and $(n_t n_{\mathbf{x}_B} n_{\mathbf{x}_A} \times 1)$, respectively. $\mathbf{C}$ and $\mathbf{C}^H$ are introduced to compactly write the overall forward and adjoint operators. Given the symmetry of the Fourier operators, the overall forward and adjoint operators are both implemented by first applying the forward Fourier transform ($\mathbf{F}$) to the input vector, truncating the frequencies up to $n_{\omega_{max}}$ ($\mathbf{I}_{\omega_{max}}$), performing a batch matrix multiplication with either the kernel or its transpose and complex conjugate ($\hat{\mathbf{K}}$ or $\hat{\mathbf{K}^H}$), padding the output to the number of frequencies required by the Fourier transform ($\mathbf{I}^H_{\omega_{max}}$ — noting that the adjoint of a truncation operator is a zero-padding operator) and finally applying the inverse Fourier transform ($\mathbf{F}^H$).

From a mathematical point of view, the batch matrix multiplication operator can be equivalently seen as a block-diagonal matrix with each frequency slice of the kernel K representing a block along its main diagonal, i.e. $\mathbf{K} = diag\{\mathbf{K}(\omega_1), \mathbf{K}(\omega_2), ..., \mathbf{K}(\omega_{N_\omega})\}$ where each frequency slice is a dense matrix of size $N_B \times N_R$ laid out as follows:

$$\mathbf{K}(\omega) = \begin{bmatrix} K(\omega, \mathbf{x}_{B_1}, \mathbf{x}_{R_1}) & K(\omega, \mathbf{x}_{B_1}, \mathbf{x}_{R_2}) & ... & K(\omega, \mathbf{x}_{B_1}, \mathbf{x}_{R_{N_R}}) \\ ... & ... & ... & ... \\ K(\omega, \mathbf{x}_{B_{N_B}}, \mathbf{x}_{R_1}) & K(\omega, \mathbf{x}_{B_{N_B}}, \mathbf{x}_{R_2}) & ... & K(\omega, \mathbf{x}_{B_{N_B}}, \mathbf{x}_{R_{N_R}}) \end{bmatrix}. \tag{4}$$

In this paper we refer to this operation as batch matrix multiplication as it more clearly

highlights the independency of the different frequency components of the kernel in the application of both the forward and adjoint operations. Finally, note that from an implementation point of view, this requires to arrange the kernel in memory such that the frequency becomes the largest (non-contiguous) step, the $\mathbf{x}_B$ dimension is the second largest (non-contiguous) step, whilst the $\mathbf{x}_R$ dimension is contiguous in memory.

IMPLEMENTING THE BATCH MATRIX MULTIPLICATION OPERATOR

Batch matrix multiplication represents the most expensive operator in the chain of operations in equation 3, as it requires acting on the kernel K that is generally a large 3D tensor in the applications discussed in this paper. The kernel may in fact contain an entire seismic surface dataset in applications such as EPSI or Marchenko redatuming or a set of surface-to-subsurface responses in MDD applications. When devising an implementation strategy for such an operator, we must consider the following *constraints*:

C1: The kernel K may not fit into a single computer's main memory at one time. As such, an *out-of-core* implementation is required, and the kernel is preferably distributed across multiple compute nodes to minimize I/O operations;

C2: Repeated evaluations of the forward and adjoint operations are required to solve an inverse problem, and;

C3: $n_{\mathbf{x}_B} \geq n_{\mathbf{x}_R} >> n_{\mathbf{x}_A}$ and $n_{\mathbf{x}_A} \geq 1$ - i.e., K is a 3D array while f and g can be either 2D or 3D, but they are both generally smaller than the kernel. In other words, multiple independent problems can be solved for each element (or group of elements) in the $\mathbf{x}_A$ dimension, as discussed in more details below.

Figure 1 presents a schematic representation of the pre-processing and forward pass of the operator in equation 3. In seismic applications, the kernel K is generally available in the time-space domain and originally stored in multiple files containing one or more shot (or receiver) gathers. For the sake of simplicity, in Figure 1 we assume that the kernel is originally divided in a set $S = \{S_1, S_2, ..., S_N\}$ of N separate files. As solving an inverse problem requires access to the input dataset multiple times, a pre-processing step is usually required to transform the kernel into frequency-space domain. Two alternative strategies can be followed: the input data are read by a distributed system of compute nodes and Fourier transformed to the frequency domain. The different nodes exchange with each other different frequency batches such that ultimately every node has access to a set of frequencies for all sources and receivers in the input dataset. Alternatively, in a pre-processing step, the same group of compute nodes reads a set of source (or receiver) gathers in time domain, converts them to frequency domain and saves them back into a new set of files partitioned along the frequency axis. While the latter approach does obviously lead to an overhead in terms of storage, it is the preferred solution as it guarantees that the time-to-frequency transformation of the entire input dataset is performed only once upfront. The former approach, although more appealing from a storage point of view, comes with the risk of losing the frequency domain representation of the input data in case of hardware failure. This also requires performing the same transformation multiple times if several inverse problems are solved independently from each other as further discussed in the Discussion section.

The choice of partitioning the input dataset over the frequency axis is justified by the fact that we wish to compute both the forward and adjoint passes of the MDC operator (C2). As the core computation for each frequency slice is a matrix-matrix multiplication,

chunking the kernel along its row space would only be favourable during the forward pass where each chunk is responsible for computing a group of values in the output vector (Figure 2b). This is however not the case for the adjoint pass, because each chunk can only be used to partially compute the elements of the output vector and communication across compute nodes is required to sum their partial outputs. The opposite scenario occurs when chunking is performed along the column space (i.e., integration axis) as shown in Figure 2c. On the other hand, we can take advantage of the independency of different frequencies to avoid any data transfer during the batch matrix multiplication step (Figure 2a). By performing the chunking along the frequency direction, no data transfer is required for applications performed fully in the frequency domain and only at the end of each batch matrix multiplication for applications where the output vector needs to be converted back to time domain. This way, the kernel is distributed only once at the beginning of the computations and different chunks never leave the compute node to which they have been initially assigned. A more detailed comparison of the three possible chunking strategies is offered in Appendix A.

[Figure 1 about here.]

[Figure 2 about here.]

Finally, in geophysical applications the kernel in time-space domain is generally a real-valued function. The orthogonal Fast Fourier transform (FFT) for real-valued inputs should be used to implement the Fourier operator $\mathbf{F}$. In this context orthogonal simply means that both forward and inverse transforms are scaled by $1/\sqrt{n_t}$. Doing so, the adjoint of the FFT operator is equivalent to the inverse FFT. Moreover, by using the FFT for real-valued

signals, only positive frequencies must be computed and stored, because values at negative frequencies are simply the complex-conjugate of those at corresponding positive frequencies.

OPEN-SOURCE SOFTWARE STACK

In this paper we aim to propose an efficient and scalable computational framework for the implementation of numerical integrals of convolution (and correlation) type. At the same time, we wish to provide users with an easy-to-use programming interface that resembles as closely as possible the underlying mathematical formulation of the equations to be solved. To achieve this, the following three ingredients are required: (1) an efficient storage format for large N-dimensional regular arrays, (2) the possibility to distribute computations over a cluster of compute nodes, and (3) the ability to define modeling operators and solve inverse problems writing high-level code with little to no modification when moving from small scale, 2D examples running on a single general-purpose computer to large, 3D field scale applications for research purposes.

In this work we opt for the expressivity of the Python programming language and the availability of a large variety of open-source libraries in the Python ecosystem. The *Zarr*[*] file format is chosen to store the kernel of the multi-dimensional convolution operator. This storage format is chosen because it allows chunked, compressed, N-dimensional arrays to be written and read concurrently from multiple threads or processes. Concurrent reading and writing is achieved by storing the entire N-dimensional array in multiple files, containing N-dimensional chunks of the entire data, alongside with a metadata file enabling correct interpretation of the stored data. Moreover, Zarr provides classes and functions for working with N-dimensional arrays that behave like NumPy arrays but whose data are divided into

[*]https://zarr.readthedocs.io/

chunks and each chunk is compressed.

The *Dask*[†] library is used to enable distributed computations. Parallelism in Dask is achieved by means of a task scheduler that builds dependency graphs and schedules tasks to a pool of workers. While Dask allows parallelism at different levels – from threads/processes on a single compute machine to distributed computing over a group of machines – our application targets the latter level of parallelism as it ultimately allows the kernels of our integral operators to reside in memory at any time during computations, split over different compute nodes. Being able to limit I/O operations to a minimum provides obvious benefits in any HPC system, and this is of particular importance for cloud computing environments – more details are provided in the Discussion section. Additionally, Dask provides distributed data structures with APIs similar to NumPy arrays (or Pandas Dataframes) and eases the loading and distribution of large input dataset in a variety of formats including Zarr).

Lastly, the *PyLops* framework (Ravasi and Vasconcelos, 2020) is used to provide a high-level symbolic representation of linear operators, easing the setup and solution of inverse problems. While initially developed for in-memory computations, it has been possible to easily adapt the PyLops framework to out-of-core operations by leveraging the ability of Dask to transparently handle distributed N-dimensional arrays as if they were in-memory Numpy arrays. Dask arrays and its API are used as a drop-in replacement for Numpy routines and arrays. In this work operators and solvers from the `pylops-distributed`[‡] library are therefore used to solve large systems of equations like those discussed in the Application section. More information about our software package and implementation details are provided in Appendix B.

[†]http://dask.org/

[‡]https://pylops-distributed.readthedocs.io/en/latest/

Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
DOI: 10.1190/geo2020-0786.1

BENCHMARKING THE MULTI-DIMENSIONAL CONVOLUTION OPERATOR

We consider the application of forward and adjoint operations in equation 3, using a seismic reflection response as the kernel of the integral operator. The constant velocity ($c = 2400$ m/s), variable density model in Figure 3 is used to generate the input dataset. This model is an extension of the 2D model used in several prior publications (Meles et al., 2015; Ravasi, 2017; Lomas and Curtis, 2020) which, despite its simplicity, contains a high density syncline inclusion that creates strong reverberations in the surface data. Nevertheless, for the 2D case both Marchenko redatuming and MDD have been shown to provide excellent cancellation of spurious artefact in the redatumed wavefield and local reflectivities. This motivates the use of a similar model to benchmark the correctness of our 3D implementation. The acquisition geometry consists of a regular grid of 9801 co-located sources and receivers which span an area of 1.8×1.2 km^2. The dataset is generated using a staggered-grid finite-difference modeling scheme, transformed to the frequency domain, truncated to contain the first $n_{\omega_{max}} = 300$ frequencies ($f \leq 73$ Hz), and stored in the Zarr file format. Additionally, we create three subsampled versions of the original dataset, whose source (and receiver) geometries are shown in Figure 4. The input vector $\mathbf{f}$ in equation 2 is instead created by numerical modeling of a single event from a subsurface point $\mathbf{x}_A = (580, 620, 650)$ m, convolved with a Ricker wavelet ($f_{dom} = 20$ Hz). Forward and adjoint modeling are performed for the following combinations of data and compute resources (Table 1).

The first benchmark is performed on compute nodes equipped with two six-core Intel(R) Xeon(R) CPUs @ 2.90GHz, and 128GB of DRAM each. Compute nodes communicate with

each other via SSH using the `SSHCluster` functionality of Dask. Moreover, the Anaconda Python distribution is used to take advantage of distributions of the NumPy and SciPy libraries with the Intel Math Kernel Library (MKL). This setup is used to create Figure 5a and 5b as well as the black line in Figure 5c. Two additional benchmarks are performed using different cluster configurations as shown in Figure 5c, namely:

- An HPC cluster with Portable Batch System (PBS) job scheduling using the `PBSCluster` Dask interface;
- A Kubernetes cluster on Microsoft Azure Cloud with *Standard_E16_v3* machines.

In both cases, the number of CPU and RAM visible to Dask is limited to 12 threads and 128GB, respectively. This ensures that the compute nodes have as similar as possible specifications to those used in the first benchmark. In our experience, the best performing approach under this framework requires creating one worker per node and dividing the kernel operator in $n_{workers}$ chunks with similar number of frequencies (where $n_{workers}$ is the number of available workers). Memory usage, CPU usage, and network communication have been monitored using both the Dask interactive dashboard and the `performance_report` functionality, which can be used save key performance indicators shown in the interactive dashboard as a static HTML file.

Each computation is repeated multiple times and the reported compute time is obtained as the average of all runs. Figure 5a shows that both forward and adjoint computations have a *quasi-linear* scaling with respect to increasing compute resources. This is especially the case for the full dataset where each node performs a large enough amount of computations to saturate its hardware usage (reducing the overall overhead of scheduling and communication). Moreover, similar compute times for forward and adjoint passes prove that the

selected strategy is optimal for both operations. In Figure 5b, the number of subsurface points for the input vector (i.e., $n_{\mathbf{x}_A}$) is gradually increased. By doing so, matrix-matrix multiplications are performed when applying the kernel K instead of matrix-vector multiplications when using a single subsurface point. The compute time observed when increasing the number of points is smaller compared to the total time of running forward (or adjoint) modeling for each point separately. This result shows the importance of choosing the most suitable linear algebra operations which can take as much as possible advantage of the available memory of each worker node. Such a strategy will be used in the Application section to solve our two inverse problems for multiple virtual points at the same time. Finally, Figure 5c shows that similar performance can be achieved on different HPC systems, including the Microsoft Azure cloud environment.

[Figure 3 about here.]

[Figure 4 about here.]

[Table 1 about here.]

[Figure 5 about here.]

APPLICATIONS

Receiver-side redatuming via Marchenko equations

The first step of redatuming aims at moving receivers from the acquisition level to a subsurface datum of interest. By doing so, we wish to be able to handle complex propagation in the overburden and accurately estimate surface-to-subsurface Green's functions. While

conventional single-scattering redatuming (Berryhill, 1984) produces inaccurate wavefields that contain spurious events arising from the incorrect handling of internal multiples in the overburden, the solution of the Marchenko equations (Wapenaar et al., 2014) provides a means to handle all internal multiples in the redatuming process. This is accomplished by solving the following system of equations (van der Neut et al., 2015):

$$\begin{bmatrix} \Theta\mathbf{R}\mathbf{f}_{\mathbf{d}}^{+} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\Theta\mathbf{R} \\ -\Theta\mathbf{R}^{*} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}^{-} \\ \mathbf{f}_{\mathbf{m}}^{+} \end{bmatrix} \rightarrow \mathbf{d} = (\mathbf{I} - \mathbf{M})\mathbf{f}, \tag{5}$$

where $\mathbf{R}$ and $\mathbf{R}^{*}$ are convolution and correlation integral operators, Θ is a time-space window, $\mathbf{I}$ is the identity operator, $*$ is used to indicate complex conjugation of the kernel in frequency domain, and $\mathbf{f}^{-}$ and $\mathbf{f}^{+} = \mathbf{f}_{d}^{+} + \mathbf{f}_{m}^{+}$ are the up- and down-going focusing functions to invert for. Following the theoretical requirements of the Marchenko equations, each frequency slice of the kernel is a $N_S \times N_R$ matrix, whose ij-th element is equal to $\hat{v_z}(\omega, \mathbf{x}_{S_i}, \mathbf{x}_{R,j}) = v_z(\omega, \mathbf{x}_{S_i}, \mathbf{x}_{R,j})/S(\omega)$. Each element represents the particle velocity recording at the j-th receiver from the i-th monopole source (v_z), deconvolved by the source signature $S(\omega)$. Note that in the literature $R(\omega, \mathbf{x}_S, \mathbf{x}_R)$ is generally defined as the pressure response at $\mathbf{x}_R$ from a vertical dipole source at $\mathbf{x}_S$, with the integration in equation 2 being carried out over sources. Using reciprocity, we have alternatively expressed the reflection response to be the vertical particle velocity recording from a monopole source; the integration in this case is carried out over the receiver axis. Mathematically speaking, the two definitions are equivalent; however, given the wider availability of seismic recordings with multi-component receivers, we model our data using the latter convention.

Once focusing functions are estimated, the up- and down-going subsurface fields $\mathbf{g}^{-}$ and

$\mathbf{g}^+$ respectively, can be obtained by direct evaluation of the following equation:

$$\begin{bmatrix} -\mathbf{g}^- \\ \mathbf{g}^{+*} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{R} \\ -\mathbf{R}^* & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{f}^- \\ \mathbf{f}^+ \end{bmatrix}. \tag{6}$$

Two approaches can be taken for the solution of equation 5: a Neumann-type iterative scheme or an inversion by means of an iterative gradient-based method. The first approach is cheaper as it generally requires fewer evaluations of the costly forward integral operator to converge; moreover, as it does not require the adjoint operator, its numerical implementation can be fully tailored to the forward operation. However, as discussed in Dukalski and de Vos (2018) and Peng and Vasconcelos (2021), convergence is guaranteed only when the spectral radius of the operator $\mathbf{M}$ satisfies the following condition $\rho(\mathbf{M}) = \lambda_{max}(\mathbf{M}) = \lim_{n\to\infty} ||\mathbf{M}^n||^{1/n} < 1$. Experience has shown that this condition is often not met in the case of highly scattering overburdens with strong contrasts, as well as when free-surface multiple reflections are included in the reflection response. In these cases we must resolve to one of the gradient-based iteration solvers to stably solve the Marchenko equations. Althoughh more demanding from a computational point of view, this approach is also beneficial in cases where the Neumann-type iterative scheme is also applicable. For example, this allows handling of missing sources in the acquisition geometry (Haindl et al., 2018, 2021) and a joint estimation of the focusing functions and source signature can be performed by using an alternating optimization approach (Becker et al., 2018). Moreover, when dealing with ocean-bottom acquisition systems (or any system where sources and receivers are placed at different depths levels), the so-called Rayleigh-Markenko equations (Ravasi, 2019, 2017; Slob and Wapenaar, 2017; Vasconcelos and Sripanich, 2019; Vargas et al., 2021) can be used to estimate surface-to-subsurface fields with minimal data pre-processing. As their forward operator does not show a structure suitable for Neumann series expansion, inversion by

means of an iterative scheme becomes the only viable solution to solve such systems of equations.

The solution of the Marchenko equations for 3D datasets has so far been limited to the Neumann-type iterative scheme (Ravasi, 2018; Jia et al., 2018b; Lomas and Curtis, 2019; Brackenhoff et al., 2020; Staring and Wapenaar, 2020). In this paper we present the first 3D example of Marchenko redatuming where the solution is obtained using a gradient-based solver. Subsurface wavefields are successfully estimated after ten iterations of the CGLS solver (Figures 6-8) for both the *Full*, *Sub2*, *Sub4*, and *Sub8* data sets. Little to no aliasing effects are visible in the retrieved Green's functions and spurious events arising from incorrect handling of overburden propagation in single-scattering redatuming are successfully suppressed (Figure 8c and 8d). Note that the source/receiver spacing for the *Sub4* dataset is greater than a quarter of the dominant wavelength ($\lambda_{dom}/4 = c/(4 * f_{dom}) =$ 30 m) in the coarser direction; this result suggests that the ability to remove artefacts from the up-going wavefield may be less affected by coarse spatial sampling than in its 2D counterpart (Brackenhoff et al., 2019; Peng and Vasconcelos, 2019; Lomas and Curtis, 2020; Peng and Vasconcelos, 2021). Strong aliasing does however arise when solving the Marchenko equations with an even coarser acquisition geometry (i.e., *Sub8* dataset in Figure 8e) showing that strict acquisition requirements do still apply to the 3D case as well.

[Figure 6 about here.]

[Figure 7 about here.]

[Figure 8 about here.]

DOI: 10.1190/geo2020-0786.1

Source-side redatuming via multi-dimensional deconvolution (MDD)

As observed in Figures 6 and 8, the retrieved Green's functions from the *Sub2* and *Sub4* data sets show minimal degradation in terms of quality and aliasing effects. On the other hand, the responses retrieved using the *Sub8* dataset show a stronger degree of noise due to poor sampling of the spatial integration axis in the convolutional integral. Nevertheless, our ability to correctly solve the Marchenko equations and remove spurious events present in the standard redatumed up-going Green's function (top panel in Figure 8) does not seem to be greatly affected. To assess the validity of this observation, a second step of redatuming is required, i.e., the redatuming on the source side.

By moving sources to the same depth level as the previously redatumed receivers, we create the so-called local reflectivity of the subsurface in the target area below the focusing datum. Further to that, we can assess the strength of spurious reflectors in both virtual common-shot gathers and zero-offset sections. More specifically, provided the availability of full-wavefield up- ($\mathbf{g}^-$) and down-going ($\mathbf{g}^+$) band-limited Green's functions, the local reflectivity $\mathbf{r}(t, \mathbf{x}_F, \mathbf{x}_{F'})$ can be estimated by solving the so-called MDD equations (Wapenaar et al., 2011; van der Neut and Herrmann, 2013):

$$\mathbf{g}^- = \mathbf{G}^+\mathbf{r}. \tag{7}$$

This equation represents a Fredholm integral of the first kind that cannot be solved via Neumann series expansion. The same iterative solver employed in the previous example, however, is used instead.

In the context of seismic interferometry, the adjoint step of equation 7 – $\mathbf{r}_{adj} = (\mathbf{G}^+)^H\mathbf{g}^-$ – is equivalent to the well-known interferometric-based redatuming method of Bakulin and Calvert (2006). Similarly, in the context of seismic imaging, this is equivalent to the

correlation-based imaging condition, commonly used in wave-equation based imaging algorithms such as reverse-time migration (Sava and Vasconcelos, 2009). As a consequence of not solving the underlying inverse problem, source-side propagation in the overburden is not properly handled and artefacts appear in the estimated reflectivity response. Even when the input Green's functions do not contain any spurious arrivals from the receiver-side redatuming step, artefacts appear due to cross-talk between unrelated events in the up- and down-going components.

An alternative approach that can alleviate this problem has been recently proposed by Staring et al. (2018). The so-called double-focusing method replaces the full up- and down-going Green's functions by the first two terms of the Neumann series expansion of the Marchenko equations. These terms are then adaptively summed leading to the cancellation of all purely overburden-related artefacts. While not requiring any inversion and being robust because of its adaptive nature, this method does not use the entire knowledge of the coda in the downgoing wavefield. As a result, the amplitude-variation-with-offset behaviour of the different events in the retrieved local reflectivity can still affected by illumination unbalancing which can only be corrected by fully solving equation 7 (Ravasi, 2017; Alfaraj et al., 2020).

In this example, we estimate up- and down-going Green's functions by solving the Marchenko equations for a grid of points at depth of $z_F = 650$ m. The grid is composed of 2911 points, sampled from $x_{F,in} = 200$ m to $x_{F,end} = 1600$ m with spacing $dx_F = 20$ m along the x axis, and from $y_{F,in} = 200$ m to $y_{F,end} = 1000$ m with spacing $dy_F = 20$ m along the y axis. Equation 7 is then repeatedly solved for batches of 20 virtual sources $\mathbf{x}_{F'}$ at a time. While solving the entire problem at once (i.e., with all virtual sources) brings some additional benefits, such as the possibility to add reciprocity constraints (van der Neut

et al., 2017), it becomes prohibitive for problems of this size. The model ($\mathbf{r}$) and data ($\mathbf{g}^-$) vectors would in fact dramatically increase in size, violating our third assumption (constraint C3) in the implementation of the batch matrix multiplication operator. This does in turn lead to a high communication overhead at every pass of MDC (and its adjoint) and poor performance.

Figure 9 shows a portion of the retrieved local reflectivity for a virtual source in the redatuming grid (yellow dot in Figure 3) and several lines of virtual receivers (cyan lines in Figure 3). The top two panels have been created by applying the adjoint and inverse of equation 7 to the Green's functions estimated via single-scattering redatuming (i.e., by inserting the initial focusing function $\mathbf{f}_d^+$ into equation 6). The Green's functions estimated by solving the Marchenko equations are instead used to produce the reflectivities in the two bottom panels. Inversion of equation 7 using Green's functions estimated via single-scattering redatuming is not able to correct for errors in the input datasets and leads to strong artefacts as visible in Figure 9b. Similarly, the use of Marchenko-retrieved Green's functions in combination with the adjoint of equation 7 produces strong artefacts in the retrieved local reflectivity response (Figure 9c). Note that in both cases artefacts are not only represented by inverted hyperbolic events (i.e., smiles), which could be easily attenuated via frequency-wavenumber filtering, but also interfering events with similar curvature to the real reflection events. On the other hand, by using accurate Green's functions and inverting equation 7, we can produce satisfactory estimates of the local reflectivity (Figure 9d). Such reflectivity contains three main events arising from the corresponding discontinuities in the density model below the focusing datum. The lack of full illumination at the edges of the grid leads to degradation of the estimated local reflectivity as distance from the virtual source increases. This amplitude behaviour is also partially due to the fact that the

local reflectivity is a particle velocity response from a monopole source; it does thus carry the directional radiation pattern of a velocity field. Finally, a clear difference is observed in the frequency content of the reflectivities obtained from the adjoint and inverse operators; this is due to the fact that wavelet deconvolution is also achieved when solving equation 7 via inversion.

Figure 10 shows the corresponding zero-offset sections along the middle cyan line in Figure 3, which can be interpreted as local *time* images of the subsurface beneath the focusing datum. Similarly to the local reflectivities, artefacts are partially removed in Figure 10c and further suppressed in Figure 10d, which is obtained from the inverted local reflectivity using the Marchenko Green's functions as input.

[Figure 9 about here.]

[Figure 10 about here.]

As a final remark, we note that to take full advantage of all the available offsets, the estimated local reflectivity could be used as input to target-oriented imaging and inversion algorithms: examples of such a kind are localized imaging via RTM (Wapenaar et al., 2014; Ravasi, 2017), localized FWI (Cui et al., 2018) or localized 1D inversion with multiply scattered waves (Gisolf et al., 2017). In all of the above scenarios, the ability to redatum an entire acquisition level closer to a target of interest without compromising on the data quality can potentially allow for the introduction of more complex physics (e.g., elastic wave equation) as well as for increasing the resolution of the estimated model parameters, making those products more suitable for subsequent reservoir characterization tasks.

DOI: 10.1190/geo2020-0786.1

DISCUSSION

In this section, we highlight some of the outstanding challenges in the application of MDC-based inversion workflows to 3D seismic datasets. More specifically, the first part focuses on practical aspects arising from the difference in the theoretical requirements and real life acquisition constraints. On the other hand, the second part discusses the numerical and computational aspects associated with these kind of inverse problems and provides insights from our experience in scaling the associated algorithms to 3D datasets.

Geophysical aspects of MDC-based inversion workflows

Historically, multi-dimensional deconvolution problems like those in equation 7 have been solved in the frequency domain. Because each frequency component can be computed and inverted individually, this approach is computationally attractive and it also allows for the adoption of very simple parallelization strategies with no exchange of data between each subproblem. However, the ill-posed nature of such an inverse problem calls for the inclusion of prior information (i.e., regularization and/or preconditioning) to improve the stability of the overall inverse process. As frequencies show different levels of ill-posedness, with low frequencies being generally more stable then high frequencies, different regularizations have to be carefully selected when solving each subproblem independently. Failing to do so results in incorrectly balanced frequency components in the estimated solution, which may ultimately lead to additional noise in its time domain counterpart (Minato et al., 2013).

van der Neut and Herrmann (2013) and van der Neut et al. (2017) suggested to tackle the MDD problem directly in the time-space domain. While this alternative approach requires to solve a single, much larger, system of equations instead of multiple independent

Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
10.1190/geo2020-0786.1

subproblems, it carries some important advantages. First, it relieves from the need to defining frequency-dependent regularization parameters. Second, additional time-domain (e.g., causal time windowing) or frequency-wavenumber preconditioners can be added to aid the inversion. In this paper, we have shown that by solving equation 7 with an iterative solver without any additional regularization still leads to a stable solution even when the input wavefields the solution of another inverse problem, rather than being directly recorded. We conjecture that the use of an iterative solver and a stopping criterion based on the residual norm acts as a natural regularizer to the problem as it prevents for entirely fitting noise in the data term. This further explains the value of using time-domain MDC operators also in problems where a solution in the frequency-domain is theoretically possible.

While solving MDD in time domain can also partially alleviate the strict acquisition requirements in terms of spatial sampling of the receiver axis, the quality of reconstruction of seismic wavefields in both Marchenko redatuming and MDD is still expected to heavily rely on the ability to accurately evaluate the spatial integral in equation 2. To better understand the impact of spatial sampling on the reconstructed wavefield, we first analyze the effect of regular subsampling on the source-side redatuming step. The up- and down-going Green's functions, estimated by solving the Marchenko equation at a constant depth level ($z_F = 650$ m), are now used as an input to equation 7 and local reflectivities are retrieved by means of least-squares inversion with the same number of iterations used to invert the *Full* dataset. As the results for the *Sub2* dataset are practically identical to those from the *Full* dataset, we avoid showing them and focus on the other two subsampled versions of the original dataset. In Figure 11 the local reflectivities from a virtual source in the middle of the redatuming grid and zero-offset images along a single crossline are shown for both datasets. It is possible to notice how by subsampling both the receiver

DOI: 10.1190/geo2020-0786.1

and source arrays by a factor of four, no severe degradation in the source-side redatuming step is observed apart from the introduction of some small coherent noise in the far offsets. This is possibly due to a poor sampling of some of the stationary contributions in the spatial integral in equation 7. On the other hand, when applying a subsampling factor of eight, the retrieved reflectivities are of noticeably poorer quality. This is especially the case when we attempt to invert equation 7, while correlation-based redatuming shows less sensitivity to a coarsening of the spatial integration axis. Similarly, the images for the *Sub4* dataset are of very similar quality to those of the *Full* dataset, both for correlation-based and inversion-based source-side redatuming. On the other hand, whilst the key reflectors are still visible in the images from the *Sub8* dataset, we can conclude that by heavily subsampling the acquisition geometry, the resulting images are affected by strong coherent noise, which becomes even more prominent when attempting to image the up- and down-going Green's functions by means of inversion. Although the correlation-based imaging condition appears to be less sensitive to poor spatial sampling, migration artefacts due to cross-talk between unrelated events in the up- and down-going Green's function do however show up as expected from our previous imaging experiments.

Finally, we also consider the case of irregular sampling. The acquisition geometry is now created by randomly selecting half of the original set of sources and receivers, which were regularly sampled over a rectangular grid with spatial sampling of 15 both in the x and y axes. Note that the number of traces in this dataset are thus the same as those in the Sub2 dataset. A pragmatic approach to handle such a spatial irregularity consists of performing Voronoi tessellation of the available receivers to determine the contribution that each receiver has in the spatial integration in the convolutional integral in equation 2, which is proportional to the areal extent of the corresponding voxel (Figure 12a). Such

10.1190/geo2020-0786.1

receiver-dependant scaling (Figure 12b) can be applied to each shot gather as part of the pre-processing of the reflection response. The inverse problem in equation 5 is then solved using CGLS and the same number of iterations as in the other examples and the retrieved Green's functions are shown in Figure 12c and 12d. Apart from the jittering effect in the seismic events in the retrieved Green's functions, arising from the fact that sources are irregularly sampled, this example shows that by carefully handling the contribution of each receiver in the spatial integration step in 2, the Marchenko equations can be satisfactorily inverted also in the presence of completely irregular geometries. This would have not been the case if we have very sparse sources (and receivers) as this would ultimately lead to aliasing effects when performing spatial integration similar to those observed for the *Sub8* dataset. Moreover, this example still assumes co-location between sources and receivers in the irregular grid, which is never the case in real life applications. In the case when the availability of sources if limited to only a portion of the receiver locations, the modified formulation of Haindl et al. (2018, 2021) could be used to deal with missing sources. Although they show results from 2D data only, its extension to 3D is trivial. Alternatively, when receivers are heavily subsampled leading to a coarse and irregular spatial integration support, the resulting fields will be a blurred version of the equivalent fields obtained from a finely sampled grid; a modification of the Neumann-like iterative scheme has been recently proposed by Wapenaar and van Ijsseldijk (2020) to handle such a scenario.

[Figure 11 about here.]

[Figure 12 about here.]

These results raise questions about the applicability of such redatuming algorithms to 3D field datasets. Methods for interpolation of coarsely sampled seismic data to fine, regu-

lar grids (Hennenfent and Herrmann, 2010; Vassallo et al., 2010; Ozbek et al., 2010; Mosher et al., 2014; Kumar et al., 2020) could however be used in a pre-processing step to provide a satisfactory sampling for both streamer and OBC acquisition systems. Data regularization, however, further increases the size of the kernel operator and, as a consequence, the memory requirements to perform multi-dimensional convolution in a distributed fashion. Nevertheless, we foresee that techniques for dimensionality reduction, such as randomized SVD (Halko et al., 2011) or Tucker tensor decomposition (Tucker, 1966), could be employed to identify redundancy and reduce the size of the kernel operator without compromising on the quality of the redatumed wavefields (Jumah and Herrmann, 2012). Alternatively, on-the-fly interpolation as conventionally applied in SRME (Moore and Dragoset, 2008; Dragoset et al., 2010) and closed-loop SRME (Lopez and Verschuur, 2015a) is also expected to provide accurate results.

Finally, other challenges specific to the solution of the Marchenko equations such as knowledge and deconvolution of the source wavelet and removal of free-surface multiples are inherently similar to both 2D and 3D applications. Several solutions have been proposed in previous field data applications, both in terms of data pre-processing (Ravasi, 2019; Jia et al., 2018a; Staring et al., 2018; Mildner et al., 2019; Staring and Wapenaar, 2020), as well as modification of the underlying redatuming scheme (Ravasi, 2017; Slob and Wapenaar, 2017; Vargas et al., 2021) which could be directly applied in a 3D context. Perhaps more poorly understood right now is the impact of attenuation on the solution of the Marchenko equations (Slob, 2016): whilst simple amplitude-based data compensations have proven sufficient so far in 2D field data application, the effect of attenuation on the solution of the 3D Marchenko equations requires additional studies and will be subject of future research. Alternatively, in scenarios where this may be attributed to complex propagation phenomena

such as internal scattering within fine layers, the augumented Marchenko method (Dukalski et al., 2019; Elison et al., 2020) may represent a more robust way to handle this problem than the above mentioned pre-processing.

Computational aspects of MDC-based inversion workflows

As a last point of discussion, we carry out an analysis of the overall computational cost involved in the solution of the two inverse problems presented in the Application section. In order to keep the section concise, we assume that the cost of solving such inverse problems is dominated by the number of convolutional integrals that are performed throughout the iterations of the chosen solver (in our case CGLS).

Starting from equation 5, the overall modeling operator for the Marchenko equations is a 2×2 block matrix which contains a multi-dimensional convolution and correlation operators, namely $\mathbf{R}$ and $\mathbf{R}^*$. The cost of performing the forward and adjoint steps is therefore quantified as 2 MDC operations. As each iteration of CGLS requires the application of one forward and adjoint pass, the overall cost of one iteration of CGLS is 4 MDC operations. Following our numerical example where that acceptable convergence is reached after $N_{iter} = 10$ iterations of CGLS, the overall cost of the inverse problem is 40 MDC operations. Noting that an additional MDC operation is required to create the data vector in equation 5 and that two additional MDC operations are required to compute the Green's functions (equation 6), the overall cost of estimating subsurface-to-surface fields is in the order of $4 * N_{iter} + 3 = 43$ MDC operations. A similar analysis can be carried out for the multi-dimensional deconvolution problem in equation 7. In this case the data vector is already available and the modeling operator is directly represented by a multi-dimensional

DOI: 10.1190/geo2020-0796.1

convolution operator with the downgoing Green's function. In the solution of the inverse problem, it is therefore only required to perform N_{iter} MDC operations.

Moreover, as the MDC operator can be applied to multiple virtual points at the same time (Figure 5b), we can also solve equations 5 and 7 for a certain number of focusing points (N_F) instead of solving N_F independent system of equations in parallel. This number should be chosen as large as possible whilst ensuring that the communication overhead associated to every MDC step does not become too large and dominates over the advantage of performing one matrix-matrix multiplication over many matrix-vector multiplications. However, as the number of focusing points for which we are interested to perform redatuming is generally of the same order of magnitude (or possibly even larger) than the number of sources and receivers, multiple groups of focusing points are generally created and inverted separately. Notice that by carefully selecting the groups of focusing points, additional constrains such as smoothness of the wavefield across virtual points (Boiero and Bagaini, 2020) or reciprocity (van der Neut et al., 2017) could be still enforced to further stabilize the inversion.

The overall redatuming process can therefore take advantage of three levels of parallelism:

- Embarrassingly parallel parallelism: the entire redatuming job is divided into a number of independent tasks, each of them comprising of an inversion for a group of focusing points;
- Distributed parallelism: each independent task is performed using a group of Dask workers (alongside a Dask scheduler and client) which have enough memory to load the entire convolutional kernel in memory. Each task solves multiple inverse problems sequentially;

- Multi-core parallelism: each worker performs several operations that can take advantage of NumPy routines like *np.matmul* for batched matrix multiplication that can leverage highly optimized Intel MKL or AMD Blis/Flame libraries, allowing also multithreading within the compute node.

Table 2 summarizes the timings related to the loading of the input dataset and setting up of the inverse problem, as well as those related to the solution of the inverse problem itself for the different subsampled data.

The ability of such algorithms to naturally leverage different levels of parallelism allows for the use of both traditional HPC solutions and Cloud computing. In our implementation, this is facilitated by the use of Dask, which supports a large variety of queuing systems used by HPC clusters. Figure 5c shows that we can achieve similar performance independently on the HPC solution adopted. Nevertheless, the ability to perform parallelism at different levels makes *cloud computing* a cost effective alternative to on-premise computing for such type of workloads. This comes with the obvious benefit of providing theoretically unlimited scalability without any up-front cost. A particularly appealing feature of most cloud computing environments is represented by the so-called *spot pricing*, which provides access to unused compute capacity at deep discount. However, as the underlying compute resource can be evicted at any time if the cloud provider needs capacity, only workloads that can sustain and/or recover from interruptions are suited to run on spot instances. This feature is of particular interest for both the Marchenko redatuming and MDD workflows as we generally solve the same inverse problem many times for a single (or groups of) independent subsurface points. While an upfront time is required for new resources to boot-strap and start running computations, each inverse problem is then completely independent from the

others, meaning that the only wasted computation if a node is evicted before the inverse process is terminated is that of the initial setup and the current inversion that was running. Our experience shows that eviction does not happen very often, and we can thus obtain a great cost reduction when running these workflows with spot instances compared to guaranteed resources (Table 3).

By leveraging the orchestration capabilities offered by Kubernetes (K8S), limited effort is required to adapt codes that were developed for classic HPC environments to be run on cloud environments. More specifically, starting from the freely available Dask Helm charts[§], we have created a new chart that contains a master pod running a K8S *Job*. This pod is responsible for setting up the inverse problem of interest for a number of subsurface points and keeping track of the number of inversions that have been carried out to completion. Another K8S pod, physically living on the same node (or another), is instantiated for the Dask scheduler and a number of other Pods for the Dask workers. Any time a node is evicted and the corresponding pod(s) fail, K8S is in charge of automatically re-instatiating the required resources. If any of the Worker is permanently deleted, Dask will handle this event by redistributing the lost portion of the kernel to the other Worker nodes and the main job is preserved. On the other hand, if either the Scheduler or Master pods are deleted, an entire new job is restarted from the subsurface point at which failure was experienced. As K8S is responsible to ensure that a job is successfully terminated, our responsibility is limited to keeping track of the evolution of our inversion and ensure that the new Job restarts the computation from where it had stopped.

[Table 2 about here.]

[§]https://github.com/dask/helm-chart

DOI: 10.1190/geo2020-0796.1

[Table 3 about here.]

CONCLUSION

In this work we have presented a framework for the computation of out-of-core multi-dimensional convolution integral operators and discussed how they can be efficiently implemented on distributed computer systems. The proposed solution is shown to handle kernel operators whose size exceeds hundreds of Gigabytes and to scale with available compute resources. Furthermore, by providing implementations of the forward and adjoint passes with similar performances, it allows for solving inverse problems that require repeated application of such operators. To illustrate its applicability, an example of full-wavefield redatuming for a simple synthetic 3D seismic dataset is presented. Being able to achieve such redatumings opens the door to a large variety of novel target-oriented imaging and inversion applications, provided they can be made applicable to 3D datasets acquired in real life scenarios.

ACKNOWLEDGEMENTS

The authors thank their respective institutions for allowing to present this work. MR thanks Haithem Jarraya and Stian Øvrevåge for the support with K8S. The different components of the computational framework presented in the work are freely available and can be installed using standard distribution systems such as PyPI and conda-forge. We refer the reader to the following Github repository (https://github.com/DIG-Kaust/MDC-HPC), which contains a set of Jupyter notebooks and Python scripts used to produce the figures in this manuscript as well as the Helm charts for running our workflows on a Kubernetes cluster.

Appendix A: A comparison of strategies for the batch matrix multiplication step

In this Appendix, we compare two possible chunking strategies for the kernel K: namely, partitioning over the frequency and row axes (Figures 2a and 2b). Such a comparison focuses on the implications in terms of computations and network communications with respect to the application of the forward and adjoint passes of the MDC operator in equation 3. Note that similar observations to those for row partitioning apply for partitioning over the column axis where the forward and adjoint passes are swapped with respect to each other.

Let's assume for simplicity that $n_{\mathbf{x}_B} = n_{\mathbf{x}_R}$, $n_{\mathbf{x}_A} = 1$, and that a certain number of workers ($n_{workers}$) are used for the computations. Starting from frequency partitioning, both forward and adjoint computations require $n_{\omega_{max}} n_{\mathbf{x}_B} n_{\mathbf{x}_R}$ multiplications and summations in the batch multiplication step. The transfer of $2(n_{\omega_{max}} - n_{\omega_{max}}/n_{workers})n_{\mathbf{x}_R}$ elements of the input vector from the master node to all workers node is also required prior to performing batch matrix multiplication, followed by the same amount of communication from worker nodes to the master node after the computation is executed. Overall, $4(n_{\omega_{max}} - n_{\omega_{max}}/n_{workers})n_{\mathbf{x}_R}$ elements are thus exchanged between workers during a pair of forward and adjoint computations. Notice, the master node is required to collect the entire frequency axis from the other workers in order to compute the forward and inverse FFTs in equation 3.

Considering now chunking over the rows of the kernel: no network communication and the same amount of computations as for the frequency partitioning are required in the forward pass. Each worker is in fact responsible for computing a portion of the output vector given access to a portion of the kernel and the entire input vector. On the other

hand, during the adjoint computation, each worker produces a partial result for the entire output vector. After that, all workers must exchange their partial results with the master node that sums them together to produce the final output vector. Finally, since during the forward computation each node needs to access the entire input vector, the master node must scatter the input vector to the workers, either straight after the adjoint pass or just before the forward pass. This amounts to an overall $2(n_{workers}-1)n_{\omega_{max}}n_{\mathbf{x}_R}$ elements being transferred across the network and an additional $n_{workers}n_{\omega_{max}}n_{\mathbf{x}_R}$ summations being performed by the master node in the adjoint pass. By comparing the overall number of elements being transferred during a pair of forward and adjoint MDC computations, we observe that for $n_{workers} > 2$ the frequency partitioning approach outperforms the other two approaches.

Our analysis shows that for the batch multiplication operation, partitioning over frequencies seems to be more efficient both in terms of computation and network traffic. However, it is worth noting that the forward and inverse FFTs required by the MDC operator in equation 3 are performed by the master node whilst all other workers are idle. On the other hand, when the kernel is chunked over rows (or columns), these operations can be also performed in parallel by all available worker nodes. More investigation is required to understand whether this benefit outweighs the extra cost in both computations and network traffic described above; this is likely to be dependant on multiple parameters such as $n_{workers}$, $n_{\omega_{max}}$, and the triplet $n_{\mathbf{x}_A}, n_{\mathbf{x}_B}, n_{\mathbf{x}_R}$. A more thorough comparison will be reported in future publications.

Appendix B: A brief introduction to the software implementation

The key elements of our software package are presented in this Appendix alongside with a high-level description of the steps required to configure Dask for the different distributed computational environments discussed in this paper.

As our computations are based on the Dask computational framework, all Jupyter notebooks and Python scripts provided alongside this paper start by initializing a Dask environment. This is a two steps process where a Dask cluster is first created, followed by the instantiation of the Dask client. The creation of the Dask cluster is highly dependant on HPC system used, however its setup is generally composed of a minimal number of steps. For example, for managed networks of machines which communicate via SSH, a single shell command is run on the machine that acts as the client:

```
dask-ssh --hostfile hostfile.txt
```

where the IP addresses of the scheduler and workers can simply be placed in the `hostfile.txt` file. For HPC environments managed via queuing systems such as PBS, Slurm, LSF, etc., a common interface is provided by the `dask-jobqueue` library. In this case, the instantiation of the Dask cluster and request of resources (e.g., number of worker nodes to be used) can be made directly within the Python code. An example for the PBS queuing system is:

```
from dask_jobqueue import PBSCluster
cluster = PBSCluster(cores=16,
                     memory='128GB')
cluster.scale(jobs=10)
```

where the number the resources in terms of cores and memory requested for each worker is defined in the first command. Once the cluster object has been created, the `scale` method is invoked to request a certain number of workers (here 10), triggering Dask to

create and submit a corresponding number of jobs to the queuing system in order to obtain the requested computational resources. Refer to the `dask-jobqueue documentation` for a full list of input parameters of the `PBSCluster` method. Finally, whilst different alternative solutions exist to deploy Dask on a cloud environment, we have used the combination of Kubernetes and Helm to create the Dask cluster in this scenario. A set of Helm charts for both interactive and batch processing are provided in our repository: we refer the reader to the accompanying documentation for a detailed set of instructions to create and configure a Kubernetes cluster on cloud environments.

Once a Dask cluster has been configured, the Dask client can easily be initialized as follows:

```
client = Client(cluster) # or Client()
```

where the variable cluster must be set equal to the IP address of the scheduler for the SSH configuration, the cluster object previously created in the case of HPC systems, or can be left empty for the Kubernetes scenario. To further ease the setup of resources, we also provide the following method `pylops_distributed.utils.backend.dask`, which directly instantiates both the Dask cluster and client as described above.

At this point, we are ready to define and solve the inverse problems discussed in this paper, whose computations will naturally take advantage of the distributed computational resources requested by the user. To start, we consider the multi-dimensional convolution operator in equation 3, which represents the key operator for the applications discussed in this paper. This operator can be initialized as follows:

```
from pylops_distributed.waveeqprocessing.mdd import MDC
# boilerplate code...
```

DOI: 10.1190/geo2020-0786.1

```
Cop = MDC(K, nt=2*nt-1, nv=1, dt=dt, dr=darea,
          twosided=True, saveGt=False, conj=False,
          prescaled=True, todask=(True, True),
          chunks=(None, None), compute=(False, False),
          dtype=np.float32)
```

and returns an object `Cop` that can be used to perform the forward and adjoint computations, respectively:

```
g = Cop.matvec(f.ravel())
f = Cop.rmatvec(g.ravel())
```

or equivalently:

```
g = Cop * f.ravel()
f = Cop.H * g.ravel()
```

where the latter commands take advantage of operator overloading for the `*` symbol and allow writing code that closely resembles the underlying mathematical formulation of the problem. This is the common usage pattern for any linear operator provided by the various libraries in the PyLops framework; for more details on the implementation of such class-based entities we refer the reader to Ravasi and Vasconcelos (2020).

The different parameters of the `MDC` class, which are also described in the self-documentation of the class, are explained in the following. `K` is the kernel of the multi-dimensional convolution operator in frequency-domain and must be of `dask.array.core.Array` type. Assuming that this kernel is stored in a Zarr file, the following lines of code can be used to read in and prepare it as input to the `MDC` operator:

DOI: 10.1190/geo2020-0786.1

```
K = scaling * da.from_zarr(zarrfile)
K = K.rechunk((max(nfmax // (nworkers + 1), 1), ns, nr))
K = client.persist(K)
```

where the Zarr file is initially read into a Dask array and possibly multiplied by a scalar (`scaling`) according to the requirement of the problem to be solved. This is followed by a chunking operation over the frequency axis (in case the input data has been saved with a different chunking pattern or different number of chunks along the frequency axis compared to the available number of workers – `nworkers`). Finally the Dask command `persist` is called to physically run the operations defined in the lines above and persist the outcome of these computations into the physical memory of each worker. `nt` and `nv` refer to the number of time samples and vitual sources expected by the operator from the input vector `f`, which correspond to n_t and $n_{\mathbf{x}_A}$ in our equations. `dt` and `dr` represent the time and spatial sampling. `twosided` is an optional parameter used to define whether the first axis of the kernel `K` contained the negative time axis followed by the positive time axis or viceversa, prior to being transformed to the frequency-domain. `saveGt` is an optional parameter that when set to `True` saves both the kernel and its transpose as contiguous arrays in memory; this parameter is however generally set to `False` when dealing with large datasets to limit the amount of data to be kept in memory of the distributed system of compute nodes.

The `conj` parameter is used to define whether to perform multi-dimensional convolution with the kernel or its complex conjugate (as, for example, required by the bottom-left element of the operator in equation 5). In the latter case, given that the kernel is generally much bigger than the input and output vectors, the forward pass is effectively performed by complex-conjugating the input and output vectors instead of the kernel itself (i.e., $K^H * f =$

$(K * f^H)^H$). Note that a similar strategy is also used when computing the adjoint pass of the MDC operator. The `prescaled` parameter is used to define whether the kernel has been pre-scaled or not. As mentioned above, it is highly recommended to pre-scale the kernel by $\sqrt{2n_t - 1} * d_t * d_r$ (with an additional factor of 2 in the Marchenko redatuming case) prior to persisting the kernel in memory. This avoids the same scaling operation to be performed every time the forward and adjoint passes of the MDC operator are evaluated. `dtype` is used to provide the expected type of the elements of the input vector **f**; if not provided, this is internally inferred from the kernel. Finally, `chunks`, `todask`, `compute` are optional parameters that are used in every `pylops-distributed` operator; they allow to re-chunk and force the input arrays to be Dask arrays (if not already) prior to applying the operator, and to trigger computations for both the forward and adjoint passes, respectively.

Considering now the MDD problem, once the forward operator has been defined, solving the inverse problem in equation 7 simply amounts to:

```
from pylops_distributed.optimization.cg import cgls
r = cgls(Cop, gminus.ravel(), niter=10,
         tol=1e-10, client=client)[0]
r = r.compute()
```

where the CGLS solver is called by passing the MDC kernel `Cop`, the input dataset `gminus`, and the number of iterations and tolerance of the solver. Moreover, by passing the Dask client, a compute graph that contains all the computations required for a single iteration of the solver is first computed and then persisted. In this context, a *compute graph* is a graph whose nodes identify individual computations that must be executed in a specific order to produce the desired output. The same procedure is repeated `niter` times. In

our experience, this represents the best strategy as it creates a large enough computation graph, therefore making the overhead of creating the graph insignificant when compared to the actual computations, whilst still creating a graph of manageable size. On the other hand, creating a single compute graph for the entire set of operations required by the CGLS solver seems to outperform the previous approach for small datasets but shows poor performance as the size of the data increases. Finally, by invoking the Dask command `compute`, the inverted vector `r` is returned to the master node, which can save it to file or use it for subsequent computations.

Moving onto the Marchenko equations, the following lines of code show how our framework allows to very easily combine multiple operators and create more complex operators like the one in equation 5:

```
from pylops_distributed.basicoperators import \
    Diagonal, Identity, Roll, Block, BlockDiag
from pylops_distributed.waveeqprocessing.mdd import MDC


Rop = MDC(K, nt=2*nt-1, nv=1,
          dt=dt, dr=darea, twosided=True,
          saveGt=False, prescaled=True,
          dtype=np.float32)
R1op = MDC(K, nt=2*nt-1, nv=1,
           dt=dt, dr=darea, twosided=True,
           saveGt=False, conj=True,
           prescaled=True, dtype=np.float32)
Rollop = Roll((2 * nt - 1) * nr,
```

```
                dims=(2 * nt - 1, nr),
                dir=0, shift=-1)
Wop = Diagonal(w.ravel())
Iop = Identity(nr * (2 * nt - 1))
Mop = Block([[Iop, -1 * Wop * Rop],
             [-1 * Wop * Rollop * R1op, Iop]]) * \
            BlockDiag([Wop, Wop])
```

We can finally solve the associated inverse problem in equation 5 in a similar fashion to the MDD case:

```
from pylops.waveeqprocessing.marchenko import directwave
# Create direct wave
Gd = directwave(wav, td, nt, dt, nfft=nfft)

# Create data
fd_plus = np.concatenate((np.flipud(Gd),
                          np.zeros((nt - 1, nr),
                          dtype=np.float32)))
fd_plus = da.from_array(fd_plus)
d = Wop * Rop * fd_plus.ravel()

# Invert
f1_inv = cgls(Mop, d.ravel(), niter=10, client=client)[0]
f1_inv = f1_inv.compute()
```

where `Gd` is the direct arrival that can be for example computed using the `directwave` method given a wavelet `wav` and a traveltime grid from the virtual source to the receiver array `td`. Alternative strategies for the computation of the direct arrival, such as FD modelling, are also allowed provided that the user can transform the direct arrival into a `numpy` array prior to creating the data `d`. Similarly `w` is the window that can easily be created given knowledge of the traveltime grid – we refer the reader to our code implementation for details.

To conclude, this Appendix serves as a basis for the reader to understand how by combining the MDC operator with the extensive set of other linear operators available in `pylops-distributed`, some of the applications discussed in this paper can be easily implemented. Given its modularity and high-level interface, we also argue that extension to other problems like those mentioned in the Introduction can be achieved with minimal extra coding effort.

REFERENCES

Alfaraj, H., J. Brackenhoff, and K. Wapenaar, 2020, Obtaining Angle-Dependent Reflectivity using the Marchenko Redatuming Method: 82nd Conference and Exhibition, EAGE, Extended Abstracts. (doi: 10.3997/2214-4609.202012138).

Alkhalifah, T., and Q. Guo, 2019, Subsurface wavefields based on the generalized internal multiple imaging: Geophysical Journal International, **219**, 1212–1224. (doi: 10.1093/gji/ggz340).

Amundsen, L., 2001, Elimination of free-surface related multiples without need of a source wavelet: Geophysics, **66**, 327–341. (doi: 10.1190/1.1444912).

Bakulin, A., and R. Calvert, 2006, The virtual source method: Theory and case study: Geophysics, **71**, no. 4, SI139–SI150. (doi: 10.1190/1.2216190).

Baumstein, A., 2008, An upside-down approach to efficient surface-related and interbed multiple prediction: SEG Technical Program Expanded Abstracts, 2466–2470. (doi: 10.1190/1.3063856).

Becker, T. S., M. Ravasi, D. van Manen, F. Broggini, and J. Robertsson, 2018, Sparse inversion of the coupled Marchenko equations for simultaneous source wavelet and focusing functions estimation: 80th Conference and Exhibition, EAGE, Extended Abstracts. (doi: 10.3997/2214-4609.201801661).

Berkhout, A., and D. Verschuur, 2011, Full wavefield migration, utilizing surface and internal multiple scattering: 81st Annual International Meeting, SEG, Expanded Abstracts, 3212–3216. (doi: 10.1190/1.3627863).

Berryhill, J. R., 1984, Wave-equation datuming before stack: Geophysics, **49**, 2064–2066. (doi: 10.1190/1.1441620).

Bisley, R., I. Moore, and H. W. Dragoset, 2005, Generalized 3D surface multiple prediction.

(WO 2005/10376).

Boiero, D., and C. Bagaini, 2020, Up-down deconvolution in complex geological scenarios: Online EAGE Conference and Exhibition, Extended Abstracts. (doi: 10.3997/2214-4609.2020611021).

Brackenhoff, J., J. Thorbecke, V. Koehne, D. Barrera, and K. Wapenaar, 2020, Implementation of the 3D Marchenko method: arXiv:2004.00896.

Brackenhoff, J., J. Thorbecke, and K. Wapenaar, 2019, Virtual sources and receivers in the real Earth: Considerations for practical applications: Journal of Geophysical Research - Solid Earth, **124**, no. 11, 802–821. (doi: 10.1029/2019JB018485,).

Cui, T., J. E. Rickett, and I. Vasconcelos, 2018, Target-oriented full-waveform inversion using marchenko redatumed wavefields: The Journal of the Acoustical Society of America, **144**, 1792. (doi: 10.1093/gji/ggaa333a,).

Davydenko, M., and E. Verschuur, 2016, Full-wavefield migration: using surface and internal multiples in imaging: Geophysical Prospecting, **65**, 7–21. (doi: 10.1111/1365-2478.12360,).

Dragoset, B., and S. MacKay, 1993, Surface multiple attenuation and subsalt imaging: SEG Technical Program Expanded Abstracts, 1099–1102. (doi: 10.1190/1.1822305).

Dragoset, B., E. Verschuur, I. Moore, and R. Bisley, 2010, A perspective on 3D surface-related multiple elimination: Geophysics, **75**, no. 5, 75A245–75A261. (doi: 10.1190/1.3475413,).

Dukalski, M., and K. de Vos, 2018, Marchenko inversion in a strong scattering regime including surface-related multiples: Geophysical Journal International, **212**, 760–776. (doi: 10.1093/gji/ggx434).

Dukalski, M., E. Mariani, and K. de Vos, 2019, Handling short-period scattering using

Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
DOI: 10.1190/geo2020-0786.1

augmented Marchenko autofocusing: Geophysical Journal International, **216**, 2129–2133. (doi: 10.1093/gji/ggy544).

Elison, P., M. S. Dukalski, K. De Vos, D.-J. van Manen, and J. O. Robertsson, 2020, Data-driven control over short-period internal multiples in media with a horizontally layered overburden: Geophysical Journal International, **221**, 769–787. (doi: 10.1093/gji/ggaa020).

Gisolf, A., P. Haffinger, and P. Doulgeris, 2017, Reservoir-Oriented Wave-Equation Based Seismic AVO Inversion: Interpretation, **5**, no. 3, 1–60. (doi: 10.1190/INT-2016-0157.1).

Haindl, C., F. Broggini, M. Ravasi, and D. van Manen, 2018, Using sparsity to improve the accuracy of Marchenko imaging given imperfect acquisition geometries: 80th Conference and Exhibition, EAGE, Extended Abstracts. (doi: 10.3997/2214-4609.201801660).

Haindl, C., M. Ravasi, and F. Broggini, 2021, Handling gaps in acquisition geometries - improving Marchenko-based imaging using sparsity-promoting inversion and joint inversion of time-lapse data: Geophysics, **86**, no. 2, S143–S154. (doi: 10.1190/geo2020-0036.1).

Halko, N., P. G. Martinsson, and J. A. Tropp, 2011, Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions: SIAM Review, **53**, 217–288. (doi: 10.1137/090771806).

Hennenfent, G., and F. Herrmann, 2010, Simply denoise: Wavefield reconstruction via jittered undersampling: Geophysics, **73**, no. 3, V19–V28. (doi: 10.1190/1.2841038).

Jia, X., A. Guitton, and R. Snieder, 2018a, A practical implementation of subsalt Marchenko imaging with a Gulf of Mexico data set: Geophysics, **83**, no. 5, S409–S419. (doi: 10.1190/geo2017-0646.1).

Jia, X., Y. Zhao, and R. Snieder, 2018b, Data interpolation for 3D Marchenko Green's function retrieval: SEG Technical Program Expanded Abstracts, 4588–4592. (doi:

10.1190/segam2019-3216672.1).

Jumah, B., and F. J. Herrmann, 2012, Dimensionality-reduced estimation of primaries by sparse inversion: SEG Technical Program Extended Abstracts, 3520–3525. (doi: 10.1111/1365-2478.12113).

Kumar, R., Y. Amin, S. Sonika, M. Vassallo, and N. Allouche, 2020, Dense ocean-bottom node interpolation using priors derived from reciprocity: SEG Technical Program Extended Abstracts, 3144–3148. (doi: 10.1190/segam2020-3426844.1).

Li, X., T. Becker, M. Ravasi, J. Robertsson, and D.-J. van Manen, 2021, Closed-aperture unbounded acoustics experimentation using multidimensional deconvolution: The Journal of the Acoustical Society of America, **149**, 1813–1828. (doi: 10.1121/10.0003706).

Lin, T. Y., and F. J. Herrmann, 2013, Robust signature deconvolution and the estimation of primaries by sparse inversion: Geophysics, **78**, no. 3, R133–R150. (doi: 10.1190/1.3628116).

Lomas, A., and A. Curtis, 2019, An introduction to Marchenko methods for imaging: Geophysics, **84**, no. 2, F35–F45. (doi: 10.1190/geo2018-0068.1).

———, 2020, Marchenko methods in a 3-D world: Geophysical Journal International, **220**, no. 1, 296–307. (doi: 10.1093/gji/ggz408).

Lopez, G. A., and D. Verschuur, 2015a, 3d focal closed-loop SRME for shallow water: Geophysical Journal International, **203**, 792–813. (doi: 10.1190/segam2015-5921009.1).

———, 2015b, Closed-loop surface-related multiple elimination and its application to simultaneous data reconstruction: Geophysics, **80**, no. 6, V189–V199. (doi: 10.1190/geo2015-0287.1).

Meles, G., K. Loer, M. Ravasi, A. Curtis, and C. A. da Costa Filho, 2015, Internal multiple prediction and removal using Marchenko autofocusing and seismic interferometry:

Geophysics, **80**, no. 1, A7–A11. (doi: 10.1190/geo2014-0408.1).

Mildner, C., F. Broggini, C. A. da Costa Filho, and J. O. Robertsson, 2019, Source wavelet correction for practical Marchenko imaging: A sub-salt field-data example from the Gulf of Mexico: Geophysical Prospecting, **67**, no. 8, 2085–2103. (doi: 10.1111/1365-2478.12822).

Minato, S., T. Matsuoka, T. Matsuoka, and T. Tsuji, 2013, Singular-value decomposition analysis of source illumination in seismic interferometry by multidimensional deconvolution: Geophysics, **78**, no. 3, Q25–Q34. (doi: 10.1190/geo2012-0245.1).

Moore, I., and B. Dragoset, 2008, General surface multiple prediction: a flexible 3D SRME algorithm: First Break, **26**, no. 9. (doi: 10.3997/1365-2397.26.1292.28603).

Mosher, C., C. Li, L. Morley, Y. Ji, F. Janiszewski, R. Olson, and J. Brewer, 2014, Increasing the efficiency of seismic data acquisition via compressive sensing: The Leading Edge, **33**, no. 4, 386–391. (doi: 10.1190/tle33040386.1).

Ozbek, A., M. Vassallo, K. Ozdemir, D.-J. van Manen, and K. Eggenberger, 2010, Crossline wavefield reconstruction from multicomponent streamer data: Part 2 — Joint interpolation and 3D up/down separation by generalized matching pursuit: Geophysics, **75**, no. 6, WB69–WB85. (doi: 10.1190/1.3497316).

Peng, H., and I. Vasconcelos, 2019, A study of acquisition-related sub-sampling and aperture effects on Marchenko focusing and redatuming: SEG Technical Program Expanded Abstracts, 248–252. (doi: 10.1190/segam2019-3214965.1).

———, 2021, A systematic analysis of acquisition sampling effects on Marchenko focusing, redatuming, and primary estimation: Geophysics, **Submitted**.

Pica, A. L., G. Poulain, B. David, M. Magesan, S. Baldock, T. Weisser, P. Hugonnet, and P. H. Herrmann, 2005, 3D surface-related multiple modeling, principles and re-

sults: 75th Annual International Meeting, SEG, Expanded Abstracts, 2080–2083. (doi: 10.1190/1.2148121).

Ravasi, M., 2017, Rayleigh-Marchenko redatuming for target-oriented, true-amplitude imaging: Geophysics, **82**, no. 6, S439–S452. (doi: 10.1190/geo2017-0262.1).

———, 2018, An overview of Marchenko-based redatuming: past, present, (and future): 80th Conference and Exhibition, EAGE, Extended Abstracts. (doi: 10.3997/2214-4609.201801966).

———, 2019, A method of redatuming geophysical data. (US Patent App. 16/084,660).

Ravasi, M., and I. Vasconcelos, 2020, PyLops - A Linear-Operator Python Library for scalable algebra and optimization: SoftwareX, **11**, 100361. (doi: 10.1016/j.softx.2019.100361).

Ravasi, M., I. Vasconcelos, A. Curtis, and A. Kritski, 2015, Multi-dimensional free-surface multiple elimination and source deblending of Volve OBC data: 77th Conference and Exhibition, EAGE, Extended Abstracts. (doi: 10.3997/2214-4609.201413355).

Sava, P., and I. Vasconcelos, 2009, Extended imaging conditions for wave-equation migration: Geophysical Prospecting, **59**, 35–55. (doi: 10.1111/j.1365-2478.2010.00888.x).

Shewchuk, J. R., 1994, An introduction to the conjugate gradient method without the agonizing pain: Carnegie-Mellon University.

Slob, E., 2016, Green's function retrieval and Marchenko imaging in a dissipative acoustic medium: Physical review letters, **116**, no. 16. (doi: 10.1103/PhysRevLett.116.164301).

Slob, E., and K. Wapenaar, 2017, Theory for Marchenko imaging of marine seismic data with free surface multiple elimination: 79th Conference and Exhibition, EAGE, Extended Abstracts. (doi: 10.1190/geo2015-0646.1).

Staal, X., and D. Verschuur, 2013, Joint Migration Inversion, Imaging Including All Multi-

ples with Automatic Velocity Update: 75th Conference and Exhibition, EAGE, Extended Abstracts. (doi: 10.3997/2214-4609.20130375).

Staring, M., R. Pereira, H. Douma, J. van der Neut, and K. Wapenaar, 2018, Source-receiver Marchenko redatuming on field data using an adaptive double-focusing method: Geophysics, **83**, no. 6, S579–S590. (doi: 10.1190/geo2017-0796.1).

Staring, M., and K. Wapenaar, 2020, Three-dimensional Marchenko internal multiple attenuation on narrow azimuth streamer data of the Santos Basin, Brazil: Geophysical Prospecting, **68**, 1864–1877. (doi: 10.1111/1365-2478.12964).

Thorbecke, J., E. Slob, J. Brackenhoff, J. van der Neut, and K. Wapenaar, 2017, Implementation of the Marchenko method: Geophysics, **82**, no. 6, WB29–WB45. (doi: 10.1190/geo2017-0108.1).

Tucker, L. R., 1966, Some mathematical notes on three-mode factor analysis: Psychometrika Review, **31**, 279–311. (doi: 10.1007/BF02289464).

van der Neut, J., and F. Herrmann, 2013, Interferometric redatuming by sparse inversion: Geophysical Journal International, **192**, 666–670. (doi: 10.1093/gji/ggs052).

van der Neut, J., M. Ravasi, Y. Liu, and I. Vasconcelos, 2017, Target-enclosed seismic imaging: Geophysics, **82**, no. 6, Q53–Q66. (doi: 10.1190/geo2017-0166.1).

van der Neut, J., I. Vasconcelos, and K. Wapenaar, 2015, On Green's function retrieval by iterative substitution of the coupled Marchenko equations: Geophysical Journal International, **203**, 792–813. (doi: 10.1093/gji/ggv330).

van Groenestijn, G. J., and D. J. Verschuur, 2009, Estimating primaries by sparse inversion and application to near-offset data reconstruction: Geophysics, **74**, no. 3, 1MJ–Z54. (doi: 10.1190/1.3111115).

Vargas, D., I. Vasconcelos, M. Ravasi, and Y. Sripanich, 2021, Scattering-based focusing for

imaging in highly-complex media from band-limited, multi-component data: Geophysics, **Submitted**.

Vasconcelos, I., M. Ravasi, A. Kritski, J. van der Neut, and T. Cui, 2017, Local, reservoir-only reflection and transmission responses by target-enclosing extended imaging: SEG Technical Program Expanded Abstracts, 5289–5293. (doi: 10.1190/segam2017-17730961.1).

Vasconcelos, I., and Y. Sripanich, 2019, Scattering-Based Marchenko for Subsurface Focusing and Redatuming in Highly Complex Media: 81st Conference and Exhibition, EAGE, Extended Abstracts. (doi: 10.3997/2214-4609.201900829).

Vassallo, M., A. Ozbek, K. Ozdemir, and K. Eggenberger, 2010, Crossline wavefield reconstruction from multicomponent streamer data: Part 1 — Multichannel interpolation by matching pursuit (MIMAP) using pressure and its crossline gradient: Geophysics, **75**, no. 6, WB53–WB67. (doi: 10.1190/1.3496958).

Verschuur, D. J., 1992, Surface-related multiple elimination in terms of Huygens sources: Journal of Seismic Exploration, **1**, 49–59.

Wapenaar, K., J. Thorbecke, J. van der Neut, F. Broggini, E. Slob, and R. Snieder, 2014, Marchenko imaging: Geophysics, **79**, no. 3, WA39–WA57. (doi: 10.1190/geo2013-0302.1).

Wapenaar, K., and J. van der Neut, 2010, A representation for Green's function retrieval by multidimensional deconvolution: The Journal of the Acoustical Society of America, **128**, EL366–EL371. (doi: 10.1121/1.3509797).

Wapenaar, K., J. van der Neut, E. Ruigrok, D. Draganov, J. Hunziker, E. Slob, J. Thorbecke, and R. Snieder, 2011, Seismic interferometry by crosscorrelation and by multidimensional deconvolution: A systematic comparison: Geophysical Journal International, **185**, 1335–1364. (doi: 10.1111/j.1365-246X.2011.05007.x).

Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
Downloaded 07/11/21 to 109.171.133.54. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/page/policies/terms
DOI: 10.1190/geo2019-0786.1

Wapenaar, K., and J. van Ijsseldijk, 2020, Discrete representations for Marchenko imaging of imperfectly sampled data: Geophysics, **85**, no. 2, A1–A5. (doi: 10.1190/geo2019-0407.1).

Zhang, L., J. Thorbecke, K. Wapenaar, and E. Slob, 2019, Transmission compensated primary reflection retrieval in the data domain and consequences for imaging: Geophysics, **84**, no. 4, Q27–Q36. (doi: 10.1190/geo2018-0340.1).

Zuberi, M. A. H., and T. Alkhalifah, 2014, Generalized internal multiple imaging (GIMI) using Feynman-like diagrams: Geophysical Journal International, **197**, 1582–1592. (doi: 10.1093/gji/ggt527).

DOI: 10.1190/geo2020-0786.1

LIST OF FIGURES

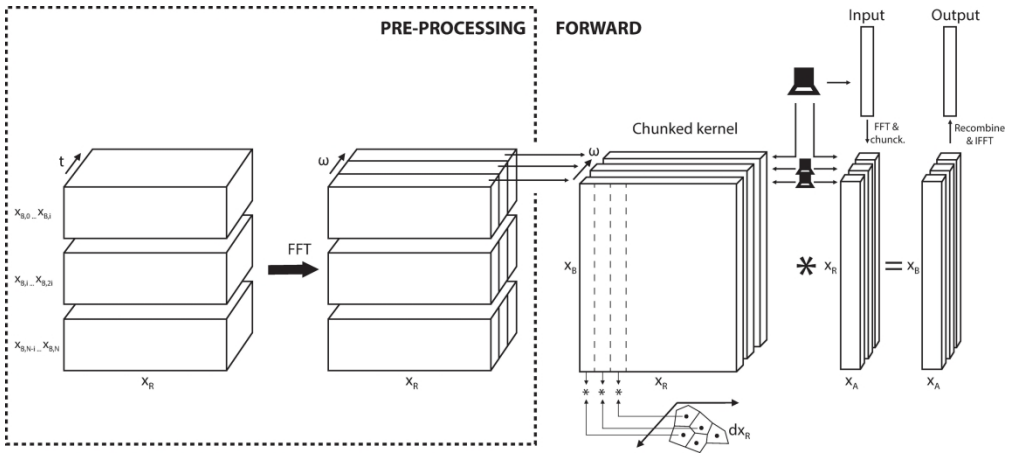

Figure 1. Schematic representation of the pre-processing of the kernel operator and forward pass.

263x117mm (300 x 300 DPI)

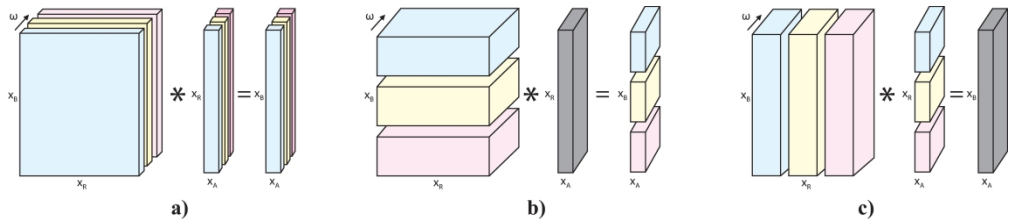

Figure 2. Schematic representations of the possible memory layouts of the 3D kernel operator, input and output vectors where chunking is performed over a) frequency slices, b) rows, and c) columns. Blue, yellow and red correspond to different compute nodes where part of the kernel resides, while grey is used to indicate that the input (or output) vector needs to be available to all compute nodes.

347x74mm (300 x 300 DPI)

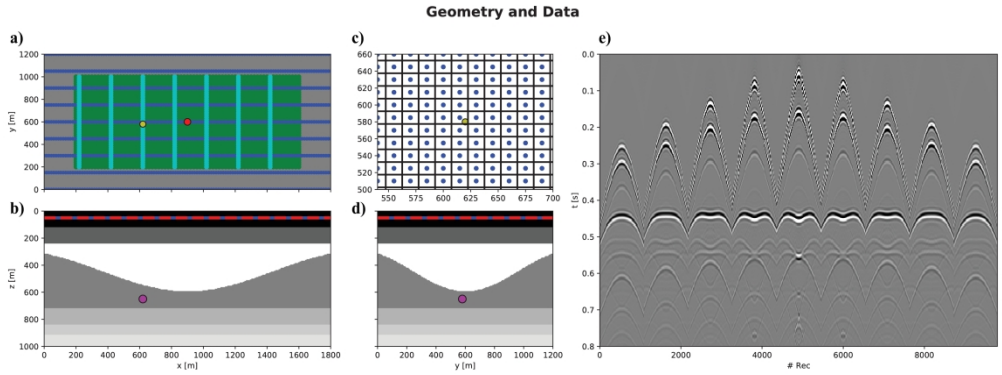

Figure 3. Model and data used in the numerical example. a-b-d Slices of the density model along z, y, and x axes}, respectively. In panel a, the red dot refers to the source used in the visualization of the data in panel e. Blue dots represent the receivers used for the visualization of the data and wavefields in the rest of the paper. Green dots represent to the overall areal coverage of the virtual receivers, whilst yellow and cyan dots refer to the virtual source and virtual receivers used in the visualization of local reflectivities, respectively. In panels b and d, red and blue lines identify the source and receiver lines along the specific cross-sections, whilst a purple dot is used to indicated the subsurface point used in subsequent visualizations of redatumed wavefields. c) Close-up of full acquisition geometry around the chosen subsurface point and Voronoi tessellation used to define the areal extent $\textbf{x}_R=900 m^2$ for the MDC integrals. e) Reflection data for source in the middle of the model and a subset of receivers (blue dots in panel a).

454x173mm (300 x 300 DPI)

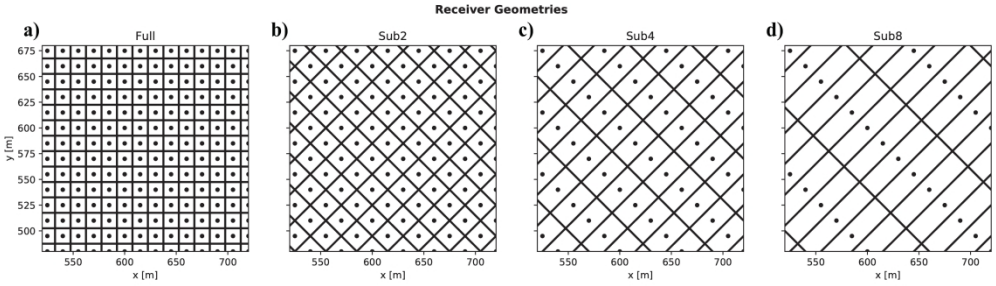

Figure 4. Close-up of the source (and receiver) grid for a) Full, b) Sub2, c) Sub4, and d) Sub8 datasets.

373x107mm (300 x 300 DPI)

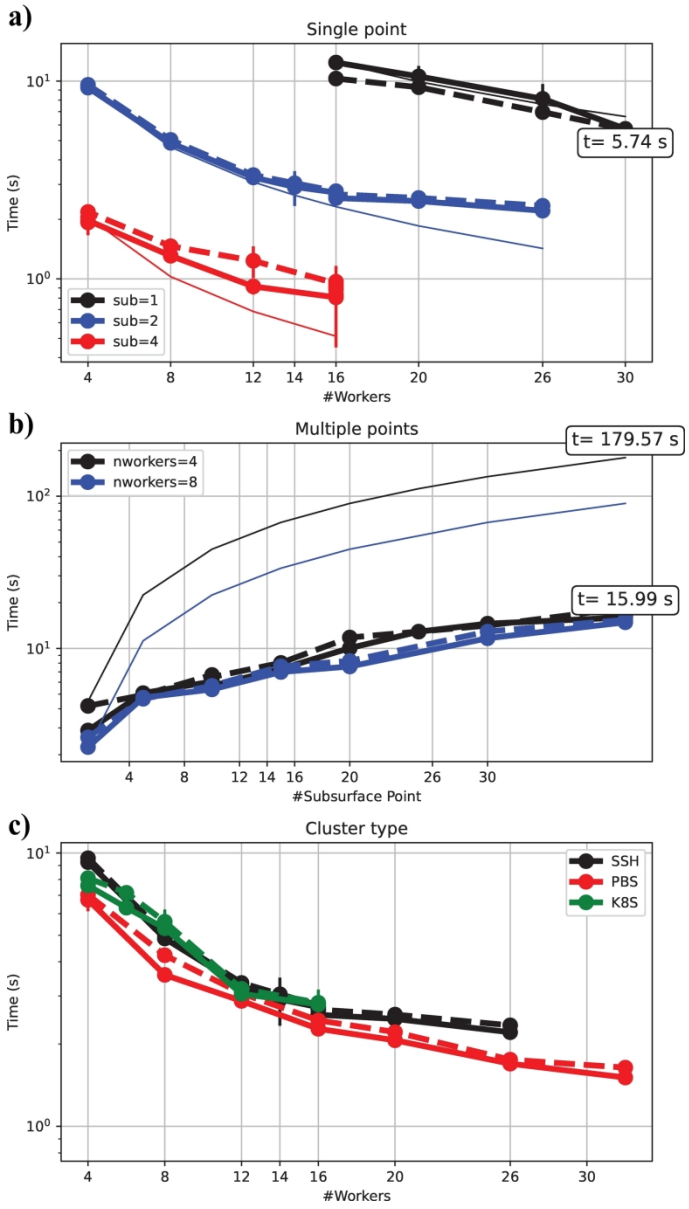

Figure 5. Benchmarking of the MDC forward (thick solid lines) and adjoint (thick dashed lines) operators with respect to a) increasing compute resources (thin lines represent the theoretical times assuming that doubling in compute resources leads to halving in compute time), b) increasing number of subsurface points (thin lines represent the time of running MDC multiple times for each subsurface point independently), c) computer cluster configurations.

170x301mm (300 x 300 DPI)

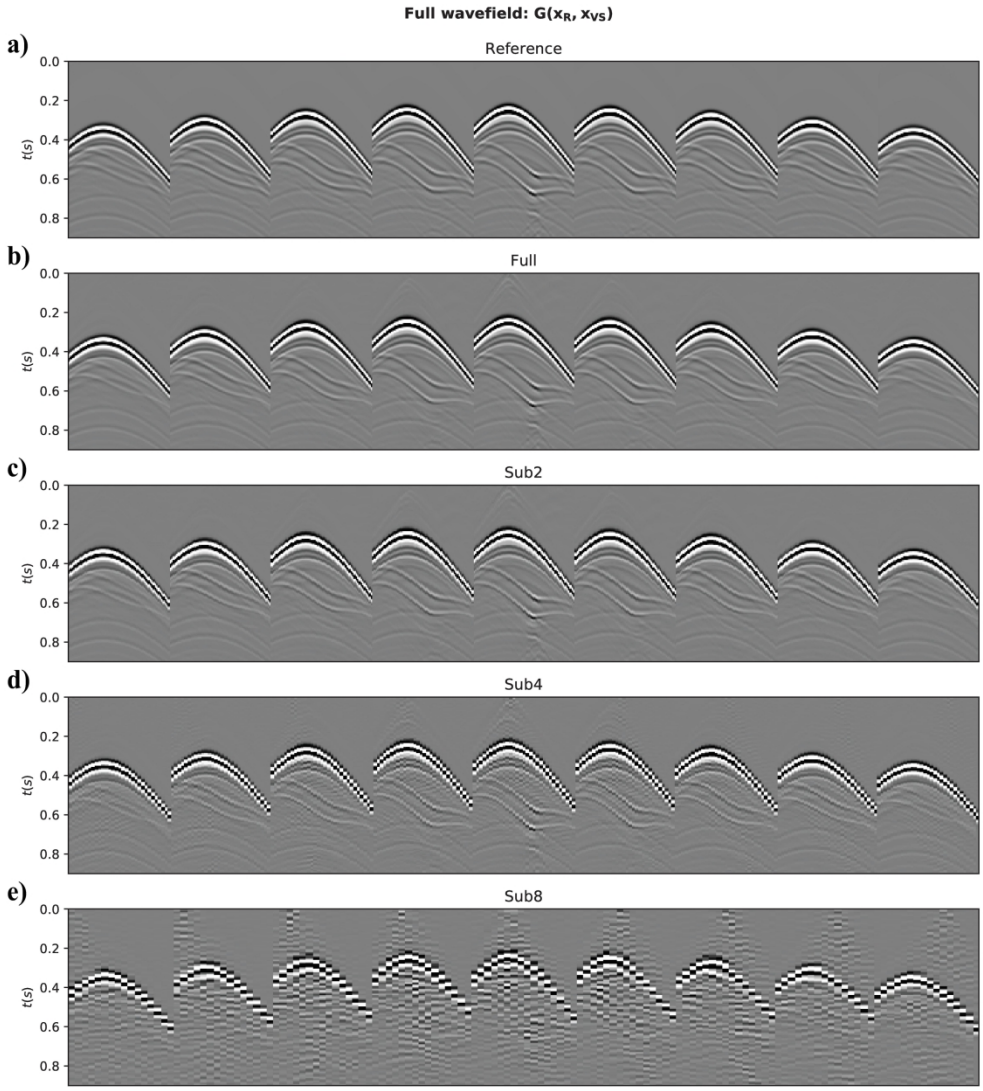

Figure 6. Green's function estimates. a) Reference from finite-difference modeling, and estimated by solving the Marchenko equations for b) Full data, c) Sub2 data, c) Sub4 data, and e) Sub8 data.

281x309mm (300 x 300 DPI)

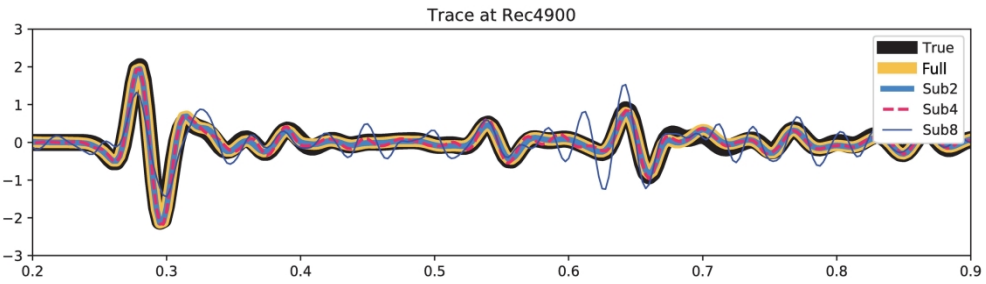

Figure 7. Trace comparison of the full wavefield for Receiver 4900 (e^{3t} amplitude gain applied to traces).

251x73mm (300 x 300 DPI)

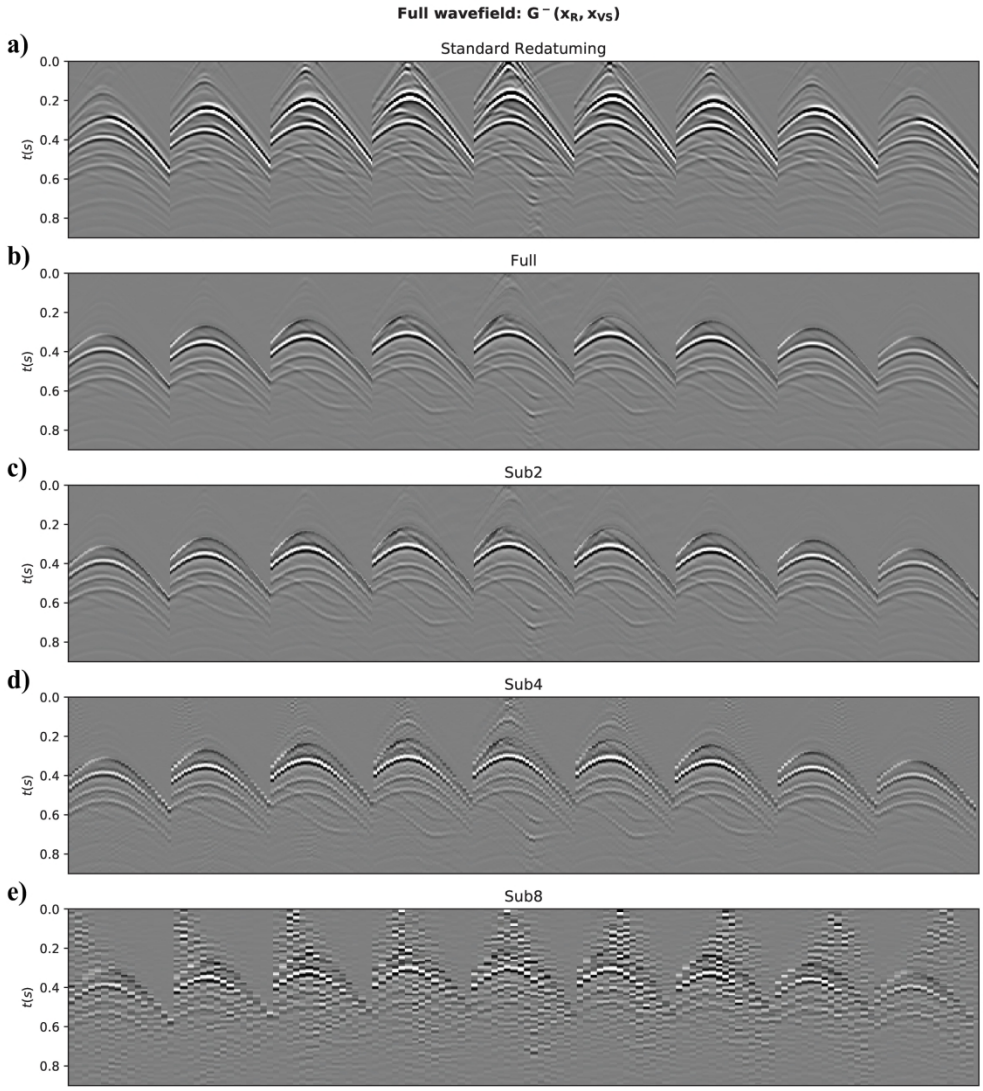

Figure 8. Up-going Green's function estimates. a) Single-scattering redatuming (i.e., using only the direct component of the downgoing focusing function in equation ef{eq:greens}), and estimated by solving the Marchenko equations for b) Full data, c) Sub2 data, c) Sub4 data, and e) Sub8 data.

281x309mm (300 x 300 DPI)

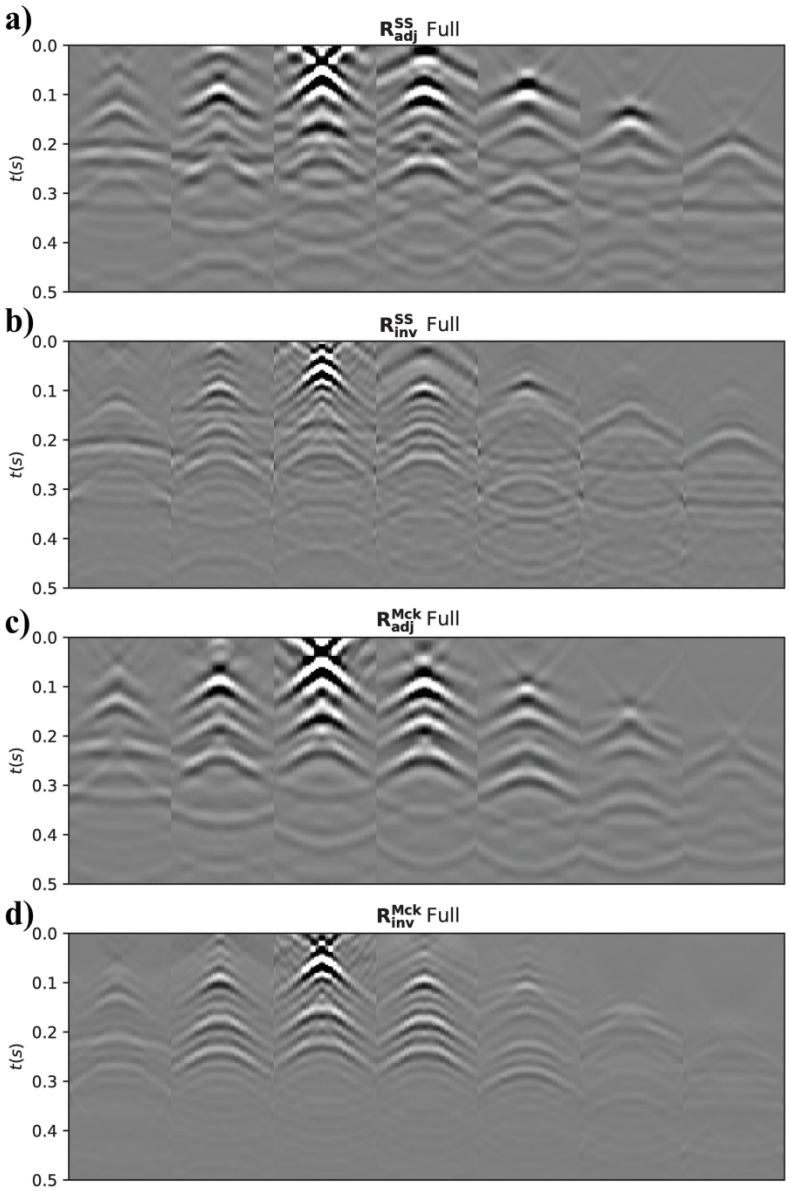

Figure 9. Local reflectivities. Reflectivities are estimated using a) single-scattering redatumed fields and the adjoint of equation 7, b) single-scattering redatumed fields and the inverse of equation 7, c) Marchenko redatumed fields and the adjoint of equation 7, d) Marchenko redatumed fields and the inverse of equation 7.

175x263mm (300 x 300 DPI)

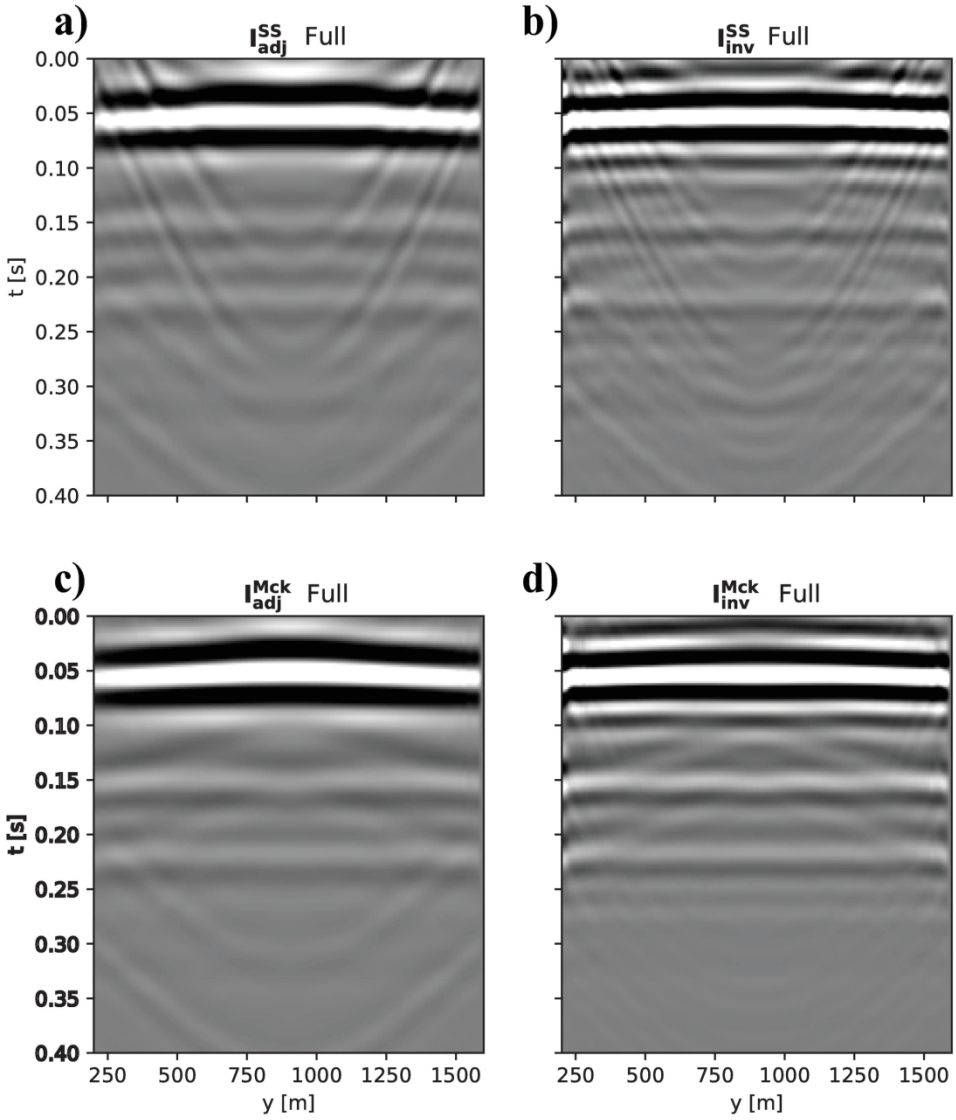

Figure 10. Zero-offset sections extracted from the local reflectivities in Figure 9, ordered in the same fashion as in the previous figure.

175x201mm (300 x 300 DPI)

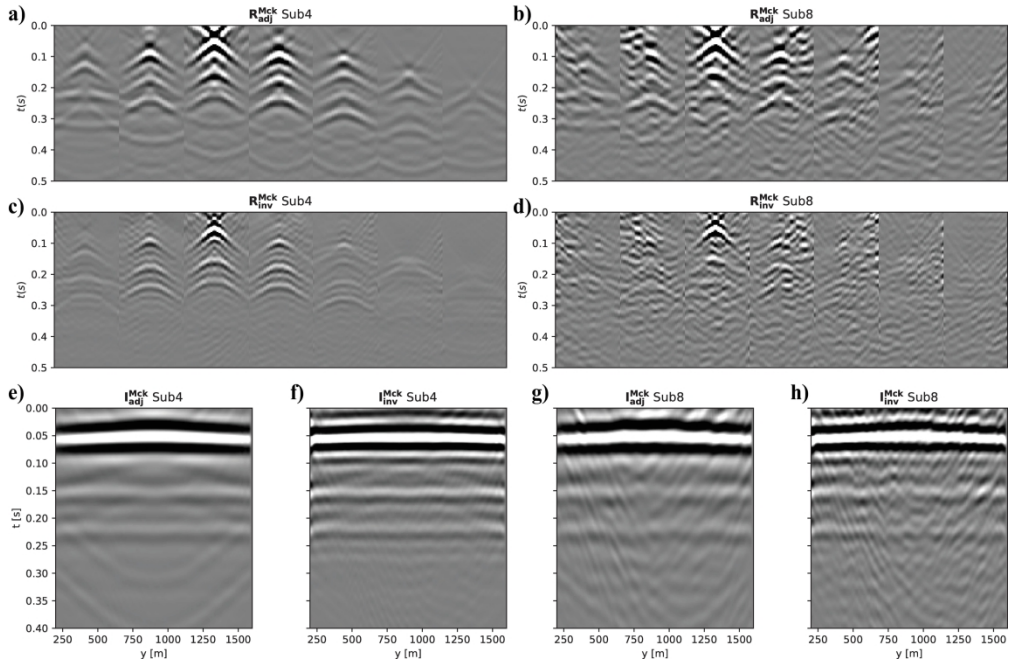

Figure 11. Local reflectivities from the Marchenko redatumed fields using a-b) the adjoint and c-d) the inverse of equation 7 for the \textit{Sub4} and \textit{Sub8}, respectively. e-f-g-h) Corresponding zero-offset sections.

348x228mm (300 x 300 DPI)

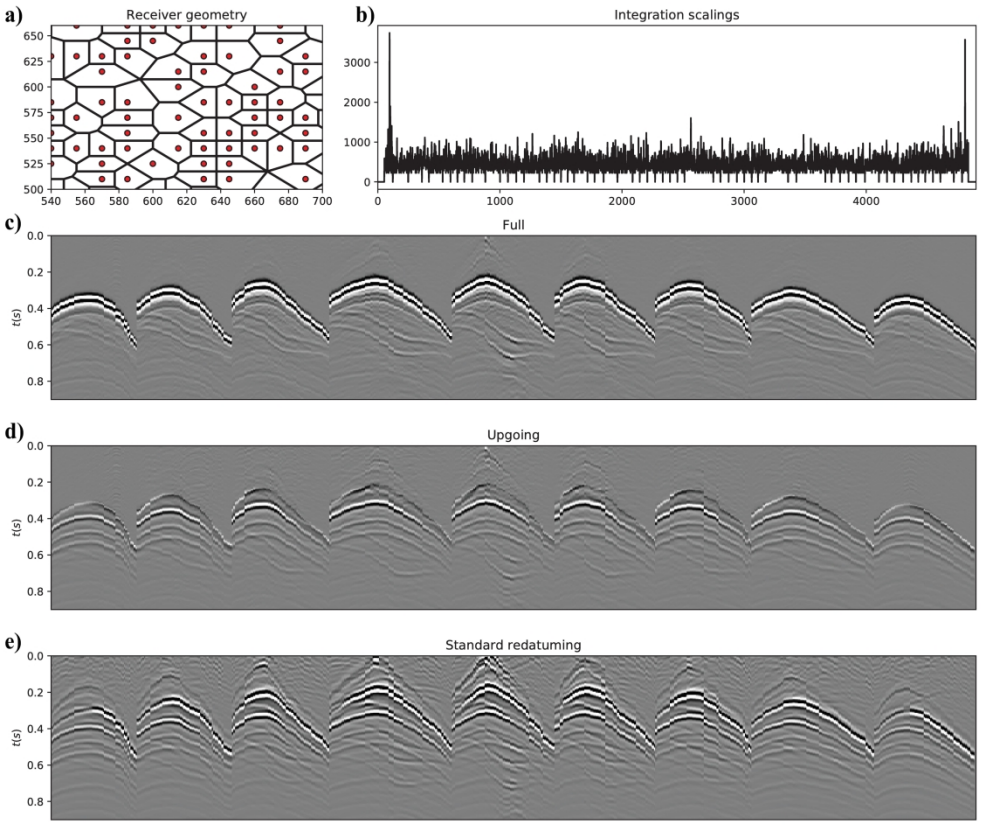

Figure 12. Marchenko redatuming for a irregularly sampled acquisition geometry. a) Close-up of receiver (and source) distribution with Voronoi tesselation. b) Integration scaling factors used for each receiver based of the area of its Voronoi cell. c) Full, b) up-going, and c) single-scattering wavefields obtained by solving the Marchenko equations.

333x276mm (300 x 300 DPI)

[Table 1 about here.]

[Table 2 about here.]

[Table 3 about here.]

LIST OF TABLES

Geometry	Spat. sampling	Data/Kernel dimensions	Data/Kernel Size (GB)	Number of nodes
Full	15 x 15	1201/300 x 9801 x 9801	461/230	16, 20, 26, 30
Sub2	21 x 21	1201/300 x 4901 x 4901	115/57	4, 8, 12, 14, 16, 20, 26
Sub4	21 x 42	1201/300 x 2451 x 2451	29/14	4, 8, 12, 16
Sub8	21 x 84	1201/300 x 1126 x 1126	6/3	-

Table 1: Combinations of data and compute resources in the evaluation of MDC forward operator. Time domain data is of float32 type, whilst frequency domain data is of complex64 type.

Geometry	Nodes	Preparation (s)	Inversion of 1 point (s)	Inversion of 20 points (s)
Full	16	180	260	1760
Sub2	8	55	115	930
Sub4	8	30	44	460

Table 2: Timings of the different steps involved in the solution of the Marchenko equations as an inverse problem for one or multiple focusing points. Intel(R) Xeon(R) CPUs @ 2.90GHz with 12 threads and 128GB of DRAM each are used for these computations.

	On-demand	Spot ($\sim 3h$ eviction)	Spot ($\sim 6h$)	Spot ($\sim 12h$)
Pod(s) creation (min)	~ 3	160	80	40
Setup (min)	3	240	120	60
N. of total evictions	0	80	40	20
Proc. time (h)	242.58	307.27	274.93	258.76
Total time (h)	242.67	313.93	278.26	260.42
Projected Cost ($)	3913.72	1199.50	1063.18	995.02

Table 3: Costs associated with the entire receiver-side redatuming process for the *Sub1* dataset ($N_F = 2911$ subsurface points) in a cloud environment using both on-demand and spot resources. These costs are computed using the hourly price of a *Standard_E16_v3* instance at the time of writing (1,008 $/h) as well as the hourly price that we were able to obtain for the same instance in a SPOT configuration (0,2388 $/h)

DATA AND MATERIALS AVAILABILITY

Data associated with this research are available and can be obtained by contacting the corresponding author.

DOI:10.1190/geo2020-0796.1