Applied Numerical Mathematics 148 (2020) 28–44

Contents lists available at ScienceDirect

Applied Numerical Mathematics

www.elsevier.com/locate/apnum

A curvelet method for numerical solution of partial differential equations

Deepika Sharma [a], Kavita Goyal [a,*], Rohit Kumar Singla [b]

[a] School of Mathematics, Thapar Institute of Engineering and Technology, Patiala, India
[b] Mechanical Engineering Department, Thapar Institute of Engineering and Technology, Patiala, India

ARTICLE INFO

Article history:
Received 4 May 2018
Received in revised form 25 June 2019
Accepted 29 August 2019
Available online 4 September 2019

Keywords:
Wavelet
Partial differential equations
Curvelet
Curvelet based numerical methods

ABSTRACT

This paper proposes a fast curvelet based finite difference method for numerical solutions of partial differential equations (PDEs). The method uses finite difference approximations for differential operators involved in the PDEs. After the approximation, the curvelet is used for the compression of the finite difference matrices and subsequently for computing the dyadic powers of these matrices required for solving the PDE in a fast and efficient manner. As a prerequisite, compression and reconstruction errors for the curvelet have been tested against different parameters. The developed method has been applied on five test problems of different nature. For each test problem the convergence of the method is examined. Moreover, to measure the performance of the proposed method the computational time taken by the proposed method is compared to that of the finite difference method. It is observed that the proposed method is computationally very efficient.

1. Introduction

Partial differential equations (PDEs) are extensively appearing for modeling real life situations associated with a wide range of fields such as financial markets, biological sciences, climate prediction etcetera. Wavelet bases were introduced as an alternative to the Fourier bases. The former has obvious and significant advantages over the later, such as absence of Gibb's effect, having a compact support etc. [27]. After looking at the usefulness of wavelets in the areas of signal and image processing [25,26], people stared using wavelets and other multi-scale representations for numerically solving PDEs [11,12, 21,29,32,31,18–20]. The most advantageous property of wavelet is that, it is possible to represent a large class of functions with fewer wavelet coefficients. This property leads to the popularity of wavelets in PDEs community. Apart from the above mentioned property, various other mathematical properties such as compact support, existence of fast wavelet transform, vanishing moments, algorithms for compression and regularization of differential operators, worked as icing on the cake [3,2]. For a large category of PDEs, wavelet methods are already designed. One can refer to the following papers for details: advection diffusion problems [28], Laplace/Poisson equations [1], reaction-diffusion equations [30], Burger's equation [23,33] and Stokes equation [16]. It can be noted that because of the advantages of wavelet bases over Fourier bases, wavelet based methods are better than Fourier methods such as FFT method for solving PDEs. Wavelet based numerical algorithms have the following advantages:

* Corresponding author.
E-mail addresses: deepika.sharma@thapar.edu (D. Sharma), kavita@thapar.edu (K. Goyal), rohit.kumar@thapar.edu (R.K. Singla).

https://doi.org/10.1016/j.apnum.2019.08.029
0168-9274/

1. The localization of wavelets, both in space and scale, governs an efficient sparse representation of differential operators (and its inverses) and functions by taking non-linear thresholding of wavelet coefficients.
2. The best marvelous characteristic of study of wavelet for numerically solving PDEs is their capability to evaluate the local regularity of the result, as by local mesh refinement, it accepts self-adaptive discretizations. Moreover, the evaluation of function spaces with respect to coefficients of wavelet function and the associated standard norm equivalence [13,22] approves preconditioning of diagonal operators in wavelet space.
3. The differential operator can be directly computed in a wavelet domain with high speed and accuracy by setting threshold values in the domain of wavelet.
4. The presence of the FWT methods yield calculations with ideal linear complexity.

Despite of the many advantages of wavelets, they have some limitations such as poor orientation sensitivity. To mitigate these limitations, curvelet had been introduced by Demanet and Candes [5]. Curvelets are attractive, because they effectively describe essential problems in which wavelet ideas are not upto mark.

1. Because of the bad orientation selectivity of wavelets, they do not present higher-dimensional irregularities efficiently. This makes curvelets interesting as they can yield an ideally adapted numerical construction to represent functions that display smooth punctuated curve.
2. Wavelets are not sufficient to detect, classify or present an intermediate dimensional arrangement with a compressed description. Curvelet is the better product because it produces better-adapted choices by consolidating concepts from geometry with ideas from classical multiscale analysis.
3. Wavelets are not optimal for solving PDEs on complex geometries. Several constructions of wavelets on complex geometries exist. In [14,10] wavelet bases are developed in light of certain form of geometries that could be presented as separate union of smooth parametric images of a unit cube. It has numerous drawbacks from a practical viewpoint as its formation depends entirely on smooth parametrization of the standard cube. This issue is settled in [15], where finite element supported wavelet bases respecting an arbitrary initial triangularization are developed. In [17] wavelets are constructed on a sphere. In spite of vast literature available on arbitrary manifolds, the subject of wavelet based methods for numerically solving PDEs on complex geometries is yet in its emerging phase. Beauty of curvelet is that it can be designed on arbitrary manifolds.
4. Wavelets are optimal for solving elliptical PDEs, but not for hyperbolic PDEs. For hyperbolic PDEs, curvelets provide better results. The hyperbolic solution operator's curvelet representation is efficient as well as ideally sparse.

Curvelets are the basis elements which are extremely sensitive to direction and are highly anisotropic. The application of a curvelet for solving PDEs is one of the recommended future scopes in [9], the locus of this work. The most significant characteristic of the curvelet is that it can be designed on complex geometries. Therefore, it can be used for finding solution of PDEs on any type of manifolds. Recently, Ying et al. [34,9] has extended the curvelet transform to three dimensions. J. Ma is dealing with curvelet based finite difference techniques for seismic wave equation with his students [24]. The objective is to develop a fast adaptive technique for numerically figure out the wave propagation. The article [4] is the first in a projected series to show how the curvelet transform structure could be exploited.

In this work, we are marching ahead a little in the direction of use of curvelets in PDEs. We have proposed fast curvelet based finite difference method which exploits the compression property of the curvelet for sparse representation of the finite difference matrices corresponding to the differential operators involved in PDEs. The proposed method is tested on five test problems and the run time of the proposed method (obtained from **cputime** command of Matlab) is compared with that of the finite difference method (FDM). It is found that the proposed method is computationally efficient.

The paper is divided into various sections such that section 2 briefly discusses curvelets. In Section 3, behavior of the compression error with respect to different variables involved in curvelet transform has been tested for two test functions. Section 4 gives information about how curvelets are used to solve PDEs. In section 5, the numerical results of proposed method on five test problems of different nature are discussed. Section 6 gives the conclusion and a few possible future directions.

2. A short explanation of curvelet

2.1. Drawbacks of classical multiscale approaches

Our aim in this work is to rose curvelets for solving PDEs. In that area, we endeavor a sparse representation of the finite difference matrices associated with PDE. Over the past two decades, multiscale approaches such as multi-grid, fast multi-polar techniques in wavelets, finite element techniques with or without adaptive refining have been widespread. All these descriptions mean references to approximately isotropic elements in all positions as well as in scales; the template is re-scaled and handled essentially in the same way in all areas. Isotropic scaling might be useful if the function under examination has no particular highlights along the orientations chosen. Tools from hereditary multi-scale analysis such as wavelets are insufficient to detect, establish or present a compact presentation of the intermediate dimensional arrangement. Curvelet is the better product because it produces better-adapted choices by consolidating concepts from geometry with

ideas from classical multiscale analysis. Curvelets can produce a geometrical framework that is ideally suited to describe functions that demonstrate punctuated smoothness of the curve.

2.2. Definition of curvelets

Curvelets are waveforms that have anisotropic behavior in fine layers, with an efficient support that accepts the parabolic postulate, length $\approx$ width^2 [4]. As with wavelets, there is a discrete as well as continuous transformation of the curvelet. A curvelet is classified by three quantities particularly, a parameter of orientation, say, $\theta, \theta \in [-\frac{\pi}{2}, \frac{\pi}{2})$; a scaling parameter $j, 0 < j < 1$ and a position parameter k where $k \in R^2$. At level a, the class of curvelet is produced by dilate, translate, revolution of a fundamental component φ_j

$$\varphi_{j,k,\theta}(x) = \varphi_j(R_\theta(x-k)),$$

where $\varphi_j(x)$ is any type of spatial wavelet with spatial breadth $\sim j$ and spatial height $\sim \sqrt{j}$, (the symbol $\sim$ stands for proportional) among the minor axis facing in the horizontal position, which is described as

$$\varphi_j(x) \approx \varphi(D_j x),$$

here $D_j = \begin{pmatrix} 1/j & 0 \\ 0 & 1/\sqrt{j} \end{pmatrix}$ is the parabolic-scaling matrix and $R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is a 2×2 rotation matrix with θ angle.

A significant feature of curvelets is that it obeys the fundamental law of harmonic analysis, which states that evaluating and restoring an arbitrarily function $f(x_1, x_2)$ as superpositions of that models is reasonable. It is acceptable to create stiff curvelet-frames as well as an arbitrarily function $f(x_1, x_2)$ can certainly be extended as an orthonormal set of curvelets. There is a discretization of angle/location/scale proceeding at a straightforward level of analysis, that approximately goes like $j_a = 2^{-a}, a = 0, 1, ..., \theta_{a,l} = 2\pi l.2^{-\lfloor a/2 \rfloor}, l = 0, 1, ..., 2^{\lfloor a/2 \rfloor} - 1$, and $b_k^{(a,l)} = R_{\theta_{a,l}}(k_1 2^{-j}, k_2 2^{-a/2})$, $k_1, k_2 \in Z$, the set φ_μ is a tight frame, here μ indexing the triplets $(j_a, \theta_{a,l}, k_b^{(a,l)})$. It implies

$$f = \sum_\mu < f, \varphi_\mu > \varphi_\mu, \qquad \|f\|_2^2 = \sum_\mu | < f, \varphi_\mu > |^2. \tag{1}$$

When a scale is given, curvelets φ_μ are achieved by implementing translations and orientations of the "mother" curvelet $\varphi_{a,0,0}$. In the frequency region,

$$\hat{\varphi}_{a,0,0}(\xi) = 2^{-3a/4} W(2^{-a}|\xi|) V(2^{\lfloor a/2 \rfloor}\theta).$$

In this case W, V is of compact support with continuous windows at interval $[1, 2]$ and $[-1/2, 1/2]$ respectively. Curvelets exists in the frequency region adjacent to an oriented rectangle R of width 2^{-a} and length $2^{-a/2}$ while in the spatial region, they are positioned in a parabolic drive of width $2^{a/2}$ and length 2^a with an orthogonal orientation to R. The combined localization in frequency as well as in space permits us to study curves such as the use of a "Heisenberg cell" in phase space and parabolic scales in these two realms.

2.3. Importance of curvelets over wavelets

The curvelets give optimum sparseness for "curve-punctuated continuous" function, wherever the function is continuous excluding discontinuities together with C^2 curves [6,7]. The degree of decline in the m-term estimate (reconstructing the function by employing m no. of coefficients) of the data is estimated to be sparse. A sparse description, along with better compression and additional sparseness for the remodeling of the denoising performance, increases the number of smooth functional areas. The curvelet transform is designed in such a fashion that maximum energy of the function is limited in only a few coefficients. This is measurable. There is clearly no basis for a function's coefficients with an uncertain curve of singularity to decline more quickly than in a frame of curvelet. This decline estimate is fast as compared to any other known system, included wavelets. The improved decline in the coefficient provides an optimally sparse representation, which is useful for solving the PDE.

2.4. Continuous-time curvelet transform

We begin with the set of $W(r)$ and $V(t)$ windows, which we refer to as the "radial window" and the "angular window". These two windows are non-negative, continuous and real-valued, with W getting non-negative real arguments along with carried out on $r \in [1/2, 2]$ and V using real arguments along with carried out on $t \in [-1, 1]$. The both windows will meet the requirements for admissibility, *i.e.*,

$$\sum_{j=-\infty}^{j=\infty} W^2(2^j r) = 1,\ r > 0; \qquad \sum_{l=-\infty}^{l=\infty} V^2(t-l) = 1,\ t \in R.$$

Presently, for every $j \geq j_0$, we start with the U_j frequency window, described in the Fourier domain as

$$U_j(r,\theta) = 2^{-3j/4} W(2^{-j}r) V\left(\frac{2^{\lfloor j/2 \rfloor}\theta}{2\pi}\right), \tag{2}$$

here $\lfloor \frac{j}{2} \rfloor$ is the integer portion of $\frac{j}{2}$. Therefore, the U_j support is a polar "wedge" described as the W and V support, the angular and the radial-window, used in all directions with scale-dependent window widths.

Set the waveform $\varphi_j(x)$ with the aid of its Fourier transformation $\hat{\varphi}_j(w) = U_j(w)$. We can consider φ_j as a "mother" curvelet on scale 2^{-j} means that, all curvelets on this scale, are achieved by translations as well as rotations of φ_j. Define,

- the equi-spaced series of rotated angles $\theta_l = 2\pi .2^{-\lfloor j/2 \rfloor}.l$ with $l = 0, 1, \ldots\ldots$ means that $0 \leq \theta_l < 2\pi$.
- the sequence of the $k = (k_1, k_2)$ translation parameters of the Z^2.

We define curvelets with these notations

$$\varphi_{j,l,k} = \varphi_j(R_{-\theta_{j,l}}(x - b_k^{(j,l)})), \tag{3}$$

here R_θ^{-1}, the inverse of R_θ is described as,

$$R_\theta^{-1} = R_\theta^T = R_{-\theta}.$$

The coefficient of a curvelet is merely the inner product between a curvelet $\varphi_{j,l,k}$ and an element $f \in L^2(R^2)$,

$$c(j,l,k) := < f, \varphi_{j,l,k} > = \int_{R^2} f(x)\overline{\varphi_{j,l,k}}(x)dx. \tag{4}$$

The curvelet coefficients, by using the theorem of Plancherel, can be represented as

$$c(j,l,k) = \frac{1}{(2\pi)^2}\int \hat{f}(w)\overline{\hat{\varphi}_{j,l,k}}(w)dw = \frac{1}{(2\pi)^2}\int \hat{f}(w)U_j(R_{\theta_l}w)e^{i<x_k^{(j,l)}, w>}dw.$$

We have coarse-scale elements too, as in wavelet theory. The low-pass window W_0 introduced as a function that follows the following equation

$$|W_0(r)|^2 + \sum_{j\geq 0} |W(2^{-j}r)|^2 = 1,$$

and for $k_1, k_2 \in Z$, coarse scale curvelets are these described by

$$\Phi_{j_0,k}(x) = \Phi_{j_0}(x - 2^{-j_0}k), \quad \hat{\Phi}_{j_0}(\xi) = 2^{-j_0}W_0(2^{-j_0}|\xi|). \tag{5}$$

The complete curvelet transformation consist of the directional finest scale elements $(\varphi_{j,l,k})_{j\geq j_0,l,k}$ along with the isotropic coarsest scale father wavelets $(\Phi_{j_0,k})$.

2.5. Fast discrete curvelet transform

Candes et al. developed two fast discrete curvelet transformations (FDCT) in 2006. One relies on the unevenly spaced fast-Fourier transformation (USFFT) [9] and the another one depends on the binding of specifically chosen Fourier-samplings (FDCT WRAPPING) [9]. We have used FDCT wrapping in this work, as it is the rapid curvelet transformation. After curvelet transformation, various groups of coefficients of curvelets at distinct angles and scales are constructed. The coefficients of curvelets at angle l and at scale j are expressed as a matrix $c_{j,l}$ and scale j is from finer level to coarser level and angle l begins at the upper-left edge and raises clockwise.

Assume that $f(t_1, t_2), 1 \leq t_1 \leq N_1,\ 1 \leq t_2 \leq N_2$ refers to the original function and $\hat{f}[n_1, n_2]$ defines $2D$ discrete Fourier transformation.

The FDCT WRAPPING implementation is as follows:

Step 1. $2D$ fast Fourier transform (FFT) is implemented on $f(t_1, t_2)$ to achieve $\hat{f}[n_1, n_2]$ Fourier coefficients.

Step 2. The new sampling function is provided by re-sampling $\hat{f}[n_1, n_2]$ at every combination of direction and scale l, j into the frequency-region as:

$$\hat{f}[n_1, n_2 - n_1 \tan\theta_l], \quad (n_1, n_2) \in P_j, \tag{6}$$

here $P_j = \{(n_1, n_2), n_{1,0} \leq n_1 < n_{1,0} + L_{1,j}, n_{2,0} \leq n_2 < n_{2,0} + L_{2,j}\}$ and $n_{1,0}, n_{2,0}$ both are the original locations of window function $\widetilde{U}_{j,l}[n_1, n_2]$. $L_{1,j}$, $L_{2,j}$ respectively are significant constants of 2^j and $2^{j/2}$, and are constituents of the length and width of the support interval of the window function.

Step 3. Multiply the new sampling function $\hat{f}[n_1, n_2 - n_1 \tan_{\theta_l}]$ with a window function $\widetilde{U}_{j,l}[n_1, n_2]$

$$\widetilde{f}_{j,l}[n_1, n_2] = \hat{f}[n_1, n_2 - n_1 \tan_{\theta_l}]\widetilde{U}_{j,l}[n_1, n_2], \tag{7}$$

where

$$\widetilde{U}_{j,l}[n_1, n_2] = W_j(w_1, w_2)V_j\big(S_{\theta_l}.\frac{2^{\lfloor j/2 \rfloor}w_2}{w_1}\big),$$
$$W_j(w_1, w_2) = \sqrt{\Phi^2_{j+1}(w) - \Phi^2_j(w)},$$
$$\Phi_j(w_1, w_2) = \phi(2^{-j}w_1)\phi(2^{-j}w_2),$$
$$S_{\theta_l} = \begin{pmatrix} 1 & 0 \\ -\tan\theta_l & 1 \end{pmatrix},$$
$$\tan\theta_l = l \times 2^{\lfloor -j/2 \rfloor}, \ \ l = -2^{\lfloor -j/2 \rfloor}, \cdots, 2^{\lfloor -j/2 \rfloor} - 1.$$

Step 4. To each $\widetilde{f}_{j,l}$, apply the inverse 2-D FFT and therefore we obtain the discrete coefficients $c_{j,l}$.

The FDCT Inverse Wrapping algorithm is as follows:

- For each array of the curvelet coefficients.
  a. Apply FFT on the array.
  b. Uncover the rectangular support to the initial orientation state.
  c. Translate to the original location.
- Add all the translated curvelet arrays.
- To reconstruct the function, take the inverse FFT.

It can be noted that the above discussed curvelets fall in the category of second generation curvelets. Prior to these constructions, exist the first generation curvelets developed by same authors [6]. For all our numerical experiments, we have used the second generation curvelets and the Matlab codes given in the toolbox Curvelab Toolbox [8] are utilized for performing the curvelet and inverse curvelet transforms. It can also be noted that no analytical formula for mother curvelet exists in general. By imposing certain conditions on abstractly assumed mother curvelet, the algorithm for the forward and inverse curvelet transform is designed as discussed above.

3. Reconstruction and compression error

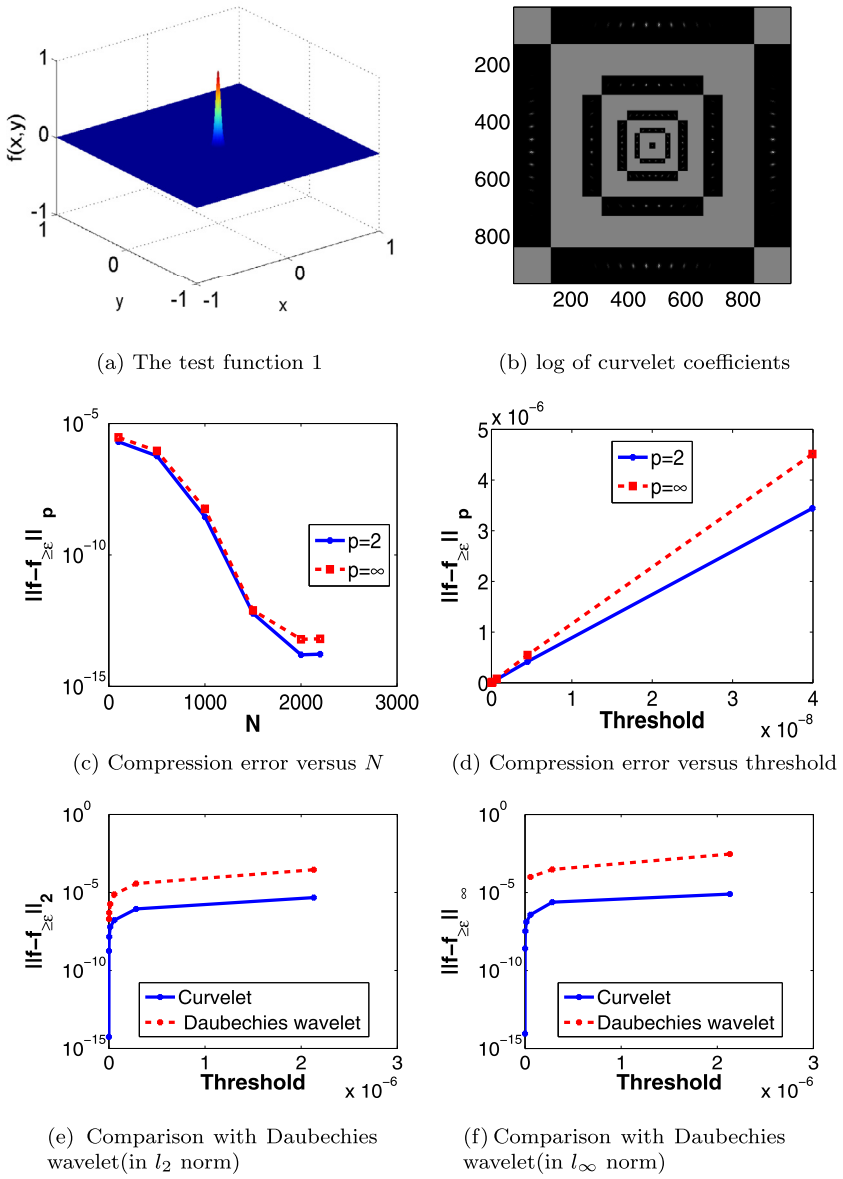
For a given function $f(x)$ and a threshold value ϵ, the curvelet expansion of function f can be decomposed into two parts as $f(x) = f_{\geq\epsilon}(x) + f_{<\epsilon}(x)$, where $f_{\geq\epsilon} = \sum_{|<f,\varphi_n>|\geq\epsilon} < f, \varphi_n > \varphi_n$ and $f_{<\epsilon} = \sum_{|<f,\varphi_n>|<\epsilon} < f, \varphi_n > \varphi_n$. The term $\|f - f_{\geq\epsilon}\|_p$ is named as the compression error. It should be noted that, $\epsilon = 0$ means no coefficients are being discarded and in that case the quantity $\|f - f_{\geq\epsilon}\|_p$ is named as the reconstruction error.

3.1. Algorithm for compression by using curvelet transform

1. Calculate the curvelet coefficients using FDCT explained in section 2, let the vector of curvelet coefficients be called as C.
2. If the threshold ϵ is given, then in the vector C, put zeros wherever the magnitude of the element is less than ϵ. Call the new vector as C_1.
3. If instead of ϵ, a compression ratio CPR is given, then the process is as follows:
   - Arrange the curvelet coefficients in descending order, let this vector be called as C_2.
   - Find out the threshold as follows: M = CPR*length of the function, and then threshold ϵ = magnitude of the Mth element of the vector C_2.
   - In the vector C, put zeros wherever the magnitude of the element is less than ϵ. Call the new vector as C_1.
4. On C_1 inverse FDCT is applied to get the compressed function.

The following two test functions are being considered for numerical experiments:

Test function 1: $f(x, y) = \exp(-1000(x^2 + y^2))$ on the domain $[-1, 1] \times [1, 1]$ is chosen as our first test function. It is shown in Fig. 1(*a*) and the corresponding curvelet coefficients (after taking log) are shown in Fig. 1(*b*). Following observations are made based on our numerical results.



(a) The test function 1

(b) log of curvelet coefficients

(c) Compression error versus N

(d) Compression error versus threshold

(e) Comparison with Daubechies wavelet(in l_2 norm)

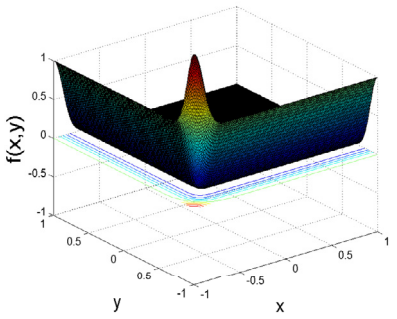
(f) Comparison with Daubechies wavelet(in l_∞ norm)

Fig. 1. Results for the test function 1. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

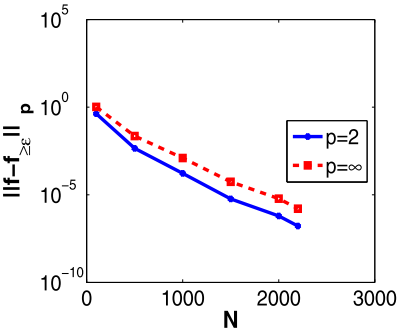
- Order of the reconstruction error is 10^{-16}.
- In Fig. 1(c), the variation of the compression error versus N is shown. One can observe a good compression.
- Fig. 1(d) plots compression error versus ϵ. An increase in the compression error with increase in values of threshold, can easily be seen. The reason for this trend is the fact that with increase in the value of threshold, more of the curvelet coefficients will be discarded.
- Fig. 1(e) and 1(f) compares the compression error for curvelet and Daubechies wavelet. It can be observed that compression in case of curvelets is much higher as compared to wavelets and hence one can conclude that curvelets are efficient than wavelets.

Test function 2: Function $f(x, y) = \exp(-50(x+1)^2 + \exp(-50(y+1)^2))$ is chosen as our second test function on the domain $[-1, 1] \times [1, 1]$ which is shown in Fig. 2(a). The following numerical results are being observed.

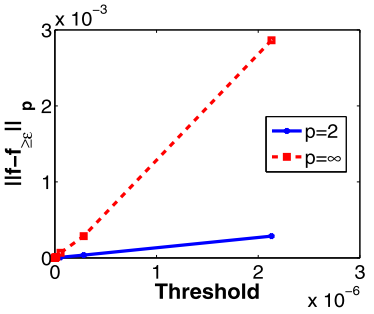
- The order of the reconstruction error is 10^{-15}.
- Fig. 2(b) plots the compression error versus N, a substantial compression can be observed. Fig. 2(c) plots compression error as a function of threshold ϵ. Here again we see that compression error increases on increasing the value of threshold.



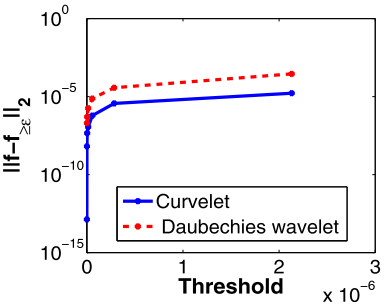
(a) The test function 2



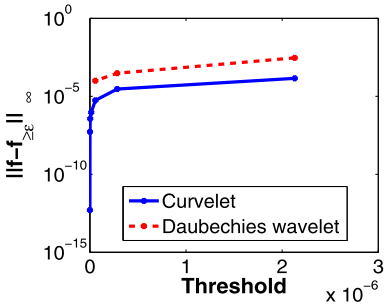
(b) Compression error versus N



(c) Compression error versus threshold



(d) Comparison with Daubechies wavelet(in l_2 norm)



(e) Comparison with Daubechies wavelet(in l_∞ norm)

Fig. 2. Results for the test function 2.

- Fig. 2(d) and 2(e) curvelets and wavelets in terms of compression error and here again curvelets perform better.

4. Solving the PDEs using curvelets

In the following text the fast curvelet based finite difference method (FCFD) to solve PDEs is explained. The factor which motivates the development of curvelet based numerical techniques for solving PDEs is that curvelet description of the finite difference matrices associated with PDEs is well organized and sparse. The main characteristic of the proposed method is to represent the differential operator in a curvelet basis φ_μ of $L^2(R^m)$ or in other words apply curvelet transform on the finite difference matrices approximating the differential operators. Having applied the curvelet transform, one can discard all the curvelet coefficients which are having magnitude less than a pre decided threshold ϵ. After that, all the computations are performed using the sparse matrices and at the end the inverse FDCT is applied to get the solution at the final time.

To explain the method in detail, a 2-D diffusion equation given as follows is considered

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(x, y); 0 \leq x, y \leq 1, \tag{8}$$

with an initial profile $u(x, y, t = 0) = u_0(x, y)$ and with appropriate boundaries such as Periodic, Neumann, Dirichlet or Robin's. On applying time discretization scheme, the Eq. (8) takes the following form

$$\mathcal{A}u^n = \mathcal{C}u^{n-1} + \Delta t f, \;\; u^0 = u_0,$$

where u^n approximated u at time $t = n\Delta t$. $\mathcal{A}$ and $\mathcal{C}$ are differential operators associated with the time discretization method. For example $\mathcal{A} = I$, $\mathcal{C} = \left(I + \Delta t\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\right)$ for the explicit forward Euler's scheme and $\mathcal{A} = \left(I - \Delta t\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\right)$, $\mathcal{C} = I$ for the implicit backward Euler's scheme.

After spatial discretization, we get

$$Au^n = Cu^{n-1} + \Delta t f, \;\; u^0 = u_0, \tag{9}$$

where u^n is the vector of all unknowns $u^n_{i,j}$, $i = 1, \cdots, N$; $j = 1, \cdots, M$ at time $t = n\Delta t$. Eq. (9) can be written as

$$u^n = A^{-1}(Cu^{n-1} + \Delta t f). \tag{10}$$

By using the Eq. (10) recursively, we obtain

$$u^n = (A^{-1})^n C^n u^0 + \sum_{k=0}^{n-1} (A^{-1})^k C^k \Delta t A^{-1} f. \tag{11}$$

When $A = I$, the above equation takes the following form

$$u^n = C^n u^0 + \sum_{k=0}^{n-1} C^k \Delta t f. \tag{12}$$

Now if we choose, $n = 2^m$ for some integer m which actually means that we are computing the numerical solution at times $2\Delta t, 4\Delta t, 8\Delta t, \cdots$, then

$$\begin{aligned}
\sum_{k=0}^{n-1} C^k \Delta f &= \sum_{k=0}^{2^m-1} C^k \Delta t f, \\
&= (I + C + C^2 + \cdots + C^{2^m-1})\Delta t f, \\
&= \left((I + C) + C^2(I + C) + C^4(I + C) + C^6(I + C) + \cdots C^{2^m-2}(I + C)\right)\Delta t f, \\
&= (I + C)(I + C^2 + C^4 + C^6 + \cdots C^{2^m-2})\Delta t f, \\
&= (I + C)(I + C^2)(I + C^4 + C^8 + \cdots C^{2^m-4})\Delta t f, \\
&\;\;\vdots \\
&= \prod_{k=0}^{m-1} (I + (C)^{2^k})\Delta t f.
\end{aligned} \tag{13}$$

Using Eq. (13), Eq. (12) can be written as

$$u^{2^m} = C^{2^m} u^0 + \prod_{k=0}^{m-1} (I + C^{2^k})\Delta t f. \tag{14}$$

It is clear from the Eq. (14), that the computation of the solution at time $t = 2^m \Delta t$ involves only the repeated squaring of the matrix C which leads to the following algorithm for computing the solution by compressing the dyadic powers of C using curvelets:

Algorithm for FCFD

- **Step 1:** Apply curvelet transform (FDCT) on the matrix C. It gives us a matrix of curvelet coefficients of C. Call this matrix as D.
- **Step 2:** Let $E = I$ (identity matrix).
  Iterate the following two steps Step 3 and Step 4 m times
- **Step 3:** Replace E with the new matrix which is created by applying threshold ϵ on $E + DE$ (it means discard all the elements which are having magnitude less than ϵ).

Table 1
The performance of FCFD for problem 1.

Threshold ϵ	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.2657	0.4469	0.5469
Θ	2.234	1.32	1.085

Table 2
Comparison of performance of FCFD and FEM.

Final time T (in seconds)	0.4848	3.8784	4.12
CPU time taken by FCFD (in seconds)	0.2341	0.583	0.9793
CPU time taken by FEM (in seconds)	0.87	0.92	1.32

Table 3
Order of accuracy for the test problem 1.

Number of grid points N	225	324	441	529	676
Order of accuracy α	-	1.5959	1.7728	2.4842	2.822

- **Step 4:** Replace D with a new matrix which is created by applying threshold ϵ on $D * D$.
  Note that after **Step 4** we get the curvelet coefficients of C^{2^m} and $\prod_{k=0}^{m-1}(I + (C)^{2^k})$ respectively in the form of D and E.
- **Step 5:** Apply inverse FDCT on D and E and call the new matrices as D_1 and E_1.
- **Step 6:** Solution at time $2^m \Delta t$ is then given by $u^{2^m} = D_1 u^0 + E_1(\Delta t) f$.

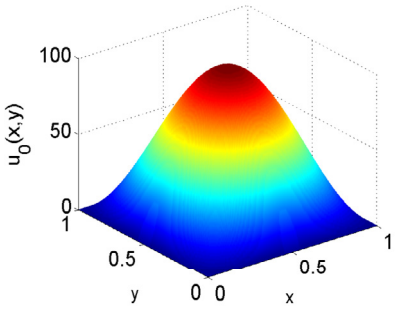
It should be noted that, if we do not choose $n = 2^m$, then an algorithm similar to the above can not be designed. Moreover, given a final time T at which we have to compute the solution of the PDE, we can always find an integer m such that $2^m \Delta t$ is approximately equal to T.

We applied the above discussed FCFD on five test problems of different nature. In each of the five test problems, we have discussed about the performance/efficiency of the method in terms of a **time compression coefficient** Θ which is described as $\Theta = \frac{CPU(threshold=0)}{CPU(threshold)}$, where $CPU(threshold = 0)$ is the run time of the FDM code and $CPU(threshold)$ is the run time of the FCFD method code. It can be noted that higher values of Θ indicate that there is a significant decrease in the CPU time of FCFD and hence the efficiency/performance of FCFD is higher.

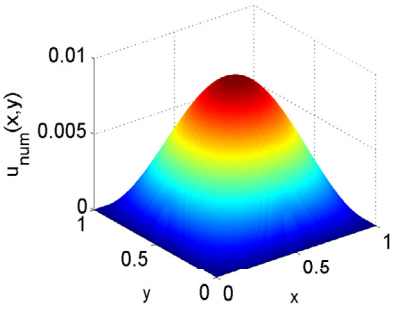
We have also computed the numerical order of accuracy, α, defined as follows: For a numerical scheme with step size h, let $\|u - u_h\| \leq kh^\alpha$, where u is the analytical solution of the problem, u_h is the numerical solution of the problem, k is a constant independent of h, then α is called the order of accuracy. Note that for the problems without analytical solutions, a reference solution with very small h can be chosen as u. Numerically, $\alpha = \frac{\log\left(\frac{\|u-u_{N1}\|}{\|u-u_{N2}\|}\right)}{\log\left(\frac{N_2}{N_1}\right)}$ where u_{N1} and u_{N2} are the numerical solutions with number of grid points equal to N_1 and N_2 respectively.

5. Results and discussions

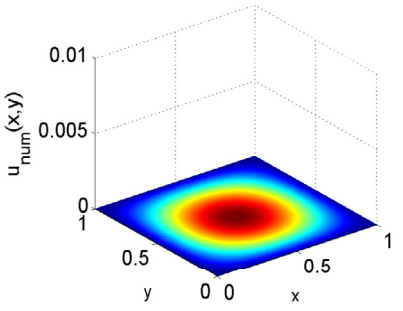
Test problem 1**:** Take $f(x, y) = 0$ in Eq. (8) along with $u(x, y, t = 0) = u_0(x, y) = 100 \sin(\pi x) \sin(\pi y)$ and the Dirichlet boundaries $u(x = 0, y, t) = u(x = 1, y, t) = u(x, y = 0, t) = u(x, y = 1, t) = 0$ are to be imposed. Fig. 3(*a*), 3(*b*), 3(*c*) show numerical solutions at time $t = 0, 0.4848, 3.8784$ respectively. We can see that with time the solution diffuses as expected. In order to have a clear view, Fig. 3(*d*) presents the numerical solution of the problem at different times by fixing $y = 0.5$. Fig. 3(*e*) shows the variation of the error (*i.e.*, $\|u_{num} - u_{ana}\|_p$) at $t = 0.4848$ versus N and the graph reports a decrease in error with an increase in value of N. Fig. 3(*f*) reports that the error increases with increase in the threshold parameter ϵ. Fig. 3(*g*) consists of two lines, the red line indicates the computational time taken by FCFD and the blue line indicates the computational time taken by FDM for computing the solution at time $t = 2^m \Delta t$. It can be observed that at all times, the red line is always below the blue line and hence the time taken by FCFD is much less. Fig. 3(*h*) shows log of curvelet coefficients of the numerical solution of the problem. Table 1 gives the variation of CPU time taken and of the time compression coefficient Θ with threshold ϵ. It can be observed that Θ decreases as the threshold decreases and hence FCFD becomes less efficient for low value of threshold. From the above analysis, one may infer that a high value of Θ can be chosen for more efficiency, but Fig. 3(*f*) should be kept in mind which indicates an increase in error with threshold ϵ. Therefore, we can conclude that the value of ϵ should be chosen wisely keeping in mind both points of view. A comparison of the run time consumed by the finite element method (FEM) with square finite elements and quadratic basis functions with that of the run time of FCFD is given in the Table 2. In this table the CPU time of FCFD and FEM is given for computing the numerical solution at the final time $T = 0.4848, 3.8784$ and 4.12 seconds. It can be observed that FCFD is taking less time. Table 3 gives the α of the proposed method.



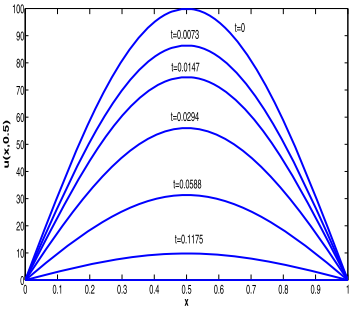
(a) The solution at time $t = 0$



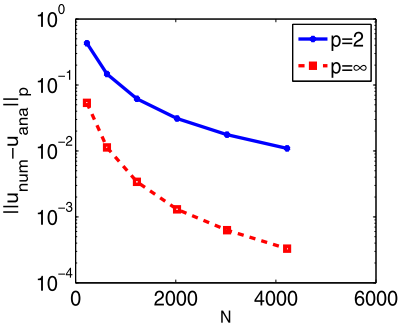
(b) The solution at time $t = 0.4848$



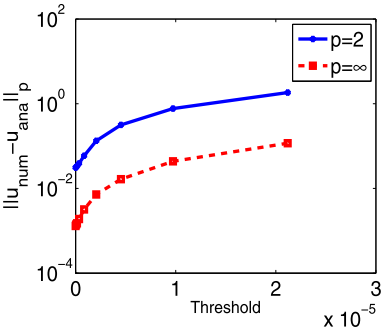
(c) The solution at time $t = 3.8784$



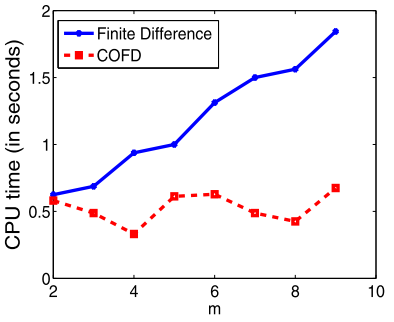
(d) $u(x, 0.5)$



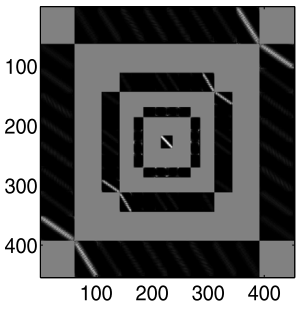
(e) $\|u_{num} - u_{ana}\|_p$ versus N



(f) $\|u_{num} - u_{ana}\|_p$ versus threshold

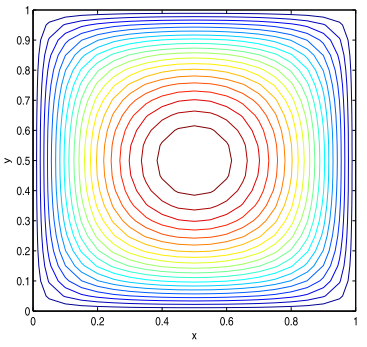


(g) CPU time for computing the solution at the time $t = 2^m \Delta t$

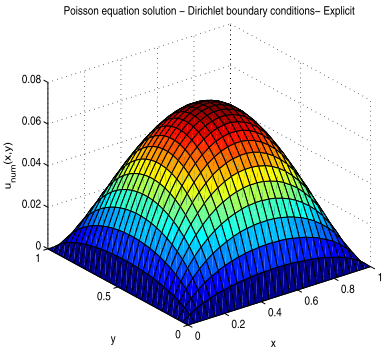


(h) log of curvelet coefficients

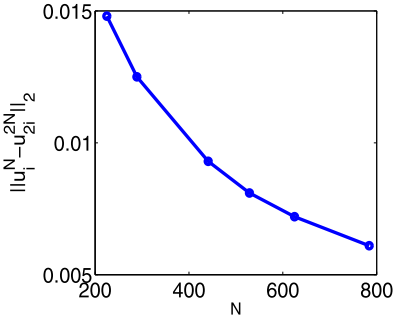
Fig. 3. Results for the test problem 1.



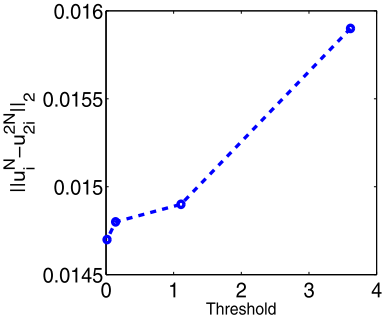
(a) Contour plot of solution



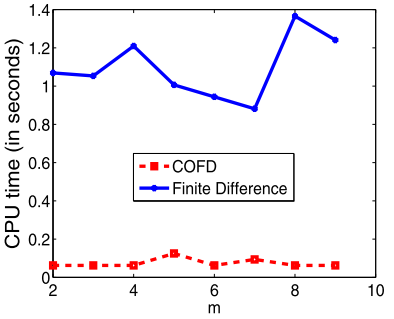
(b) Poisson equation solution



(c) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N



(d) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold



(e) CPU time for computing the solution at the time $t = 2^m \Delta t$

Fig. 4. Results for the test problem 2.

Test problem 2**:** The following 2-dimensional Poisson equation is considered

$$-\Delta u = 1, 0 \leq x, y \leq 1,$$

with $u(x = 0, y) = u(x, y = 0) = u(x = 1, y) = u(x, y = 1) = 0$ as the Dirichlet boundaries. On discretizing the domain with 625×625 points, Fig. 4(a) represent the contour plot of the solution. The closeness of contour lines indicates steepness. We have observed that, the contour lines which are evenly spaced and closed together indicates a uniform, steep slope. Fig. 4(b) represents the numerical solution of the problem. Fig. 4(c) represents the graph between the error *i.e.*, $\|(u_i)^M - (u_{2i})^{2M}\|_2$ where $i = 1, 2, \cdots, M$ and number of grid points. Here error is calculated according to double mesh principle. The procedure of double mesh principle is to double the no. of mesh points and then error is calculated as above, where $(u_{2i})^{2M}$ is the solution obtained on a mesh containing the same mesh points which are used in the previous mesh. We have observed from the graph that error decreases on increasing the N. Fig. 4(d) represents the graph between $\|(u_i)^M - (u_{2i})^{2M}\|_2$ and threshold (chosen by the user). It shows that error increases as we increase threshold. Fig. 4(e) compares the computational time taken by FDM and FCFD method for computing the solution at time $t = 2^m \Delta t$. It is shown that CPU time taken by

Table 4
The efficiency of FCFD for problem 2.

Threshold ϵ	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.0103	0.0198	0.0254
Θ	3.038	1.581	1.232

Table 5
Order of accuracy for the test problem 2.

Number of grid points N	400	625	841	1089
Order of accuracy α	-	1.8171	2.5079	2.5226

Table 6
The efficiency of FCFD for problem 3.

Threshold	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.4853	0.5478	0.5573
Θ	1.197	1.061	1.052

Table 7
Order of accuracy for the test problem 3.

Number of grid points N	441	676	900	1089
Order of accuracy α	-	1.7740	2.4254	2.6200

FCFD method is less than FDM. Table 4 displays the Θ values for different values of threshold. It shows the efficiency of our algorithm. Table 5 shows the order of accuracy FCFD for the second test problem.

Test problem 3**:** Consider the following equation

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}; 0 \leq x, y \leq 1, t > 0, \tag{15}$$

along with $u(x, y, 0) = x(x-1)y(y-1)$ and $\frac{\partial w}{\partial t}(x, y, 0) = 0$ as the initial profiles and the Dirichlet zero boundaries. Figs. 5(a), 5(b), 5(c) show the solution of the test problem 3 at time $t = 0$, $t = 0.5$, $t = 2$ respectively. Fig. 5(d) plots the error $\|u_i^M - u_{2i}^{2M}\|_2$, where $i = 1, 2, \cdots, M$ at the final time 0.99 as a function of N. We have observed that error decreases on increasing the number of grid points. Fig. 5(e) represents the graph between $\|(u_i)^M - (u_{2i})^{2M}\|_2$ and threshold. It shows that error increases on increasing threshold. Fig. 5(f) compares the CPU time consumed by FCFD and FDM to compute the solution at time $t = 2^m \Delta t$. It shows that CPU time taken by FCFD is less than FDM. Table 6 shows that as we decrease threshold, the value of Θ decreases which reveals the efficiency of our method. Table 7 shows α of the developed method.

Test problem 4**:** The next PDE is given as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}; 0 \leq x, y, z \leq 1,$$

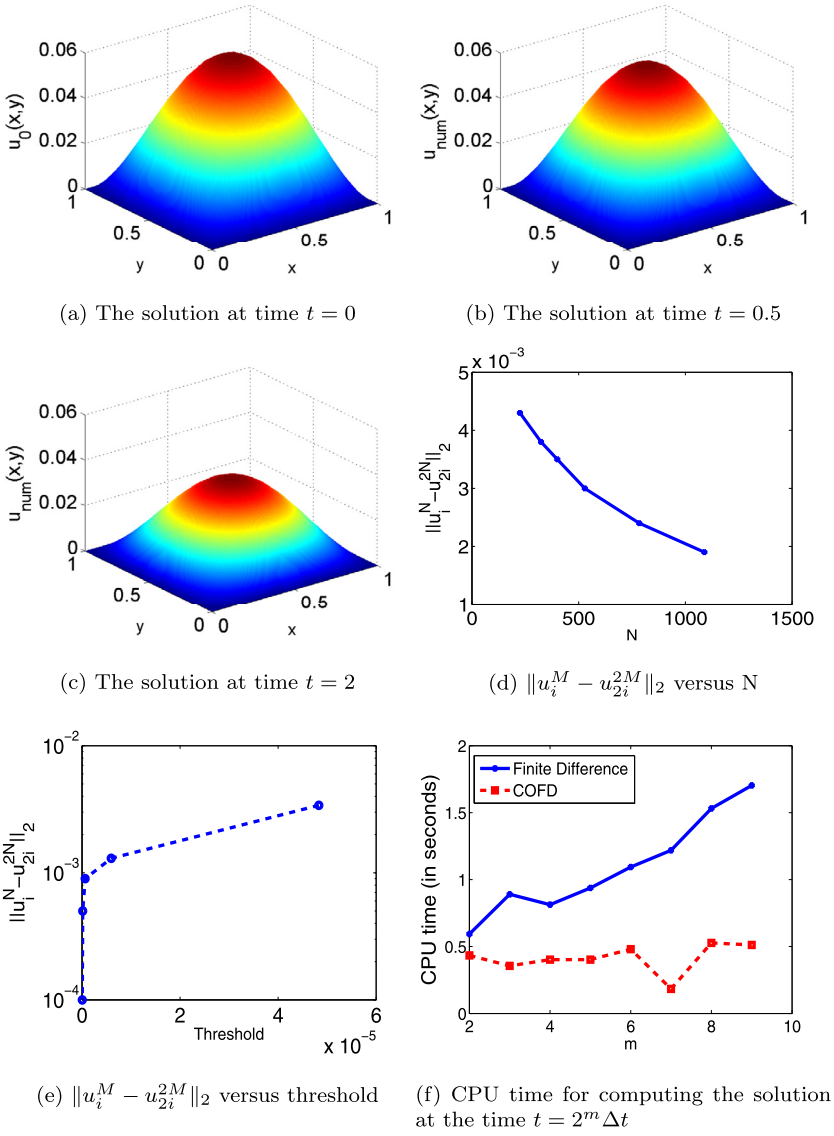
where an initial profile is $u(x, y, z, t = 0) = 100 \sin(\pi x) \sin(\pi y) \sin(\pi z)$ and Dirichlet zero boundary conditions. Figs. 6(a), 6(b), 6(c) show the solution profile for a fixed z coordinate as $z = 0.2$ at times $0, 0.1316$ and 4.213 respectively. For a better view of the solution profile, we draw the solution of the problem by making the slices at $x = \{0.3, 0.6\}$, $y = 0.5$ and $z = 0.5$. Figs. 6(d), 6(e) show such solutions at time 0 and 0.1316 respectively. Fig. 6(f) plots the graph of error versus N. Fig. 6(g) plots $\|u_{num} - u_{ana}\|_p$ as a function of threshold. We can here note that the increase in the error with respect to ϵ is insignificant. Fig. 6(h) compares the run time consumed by FCFD and the FDM to compute the solution at time $t = 2^m \Delta t$. It shows that CPU time taken by FCFD is less than FDM (Table 8). Table 9 shows the α of the developed method.

Test problem 5**:** Now we consider two dimensional Lotka-Volterra predator-prey model

$$u_t = c_{11} u_{xx} + c_{12} u_{yy} + a_1 u - r_1 u v,$$
$$v_t = c_{21} v_{xx} + c_{22} v_{yy} + a_2 v - r_2 u v, \quad 0 \leq x, y \leq 1$$

where $v(t, x, y)$ and $u(t, x, y)$ respectively represent the predator and prey population density at time t and space co-ordinates (x, y). Following parameters are chosen: $c_{12} = c_{11} = 0.1$ and $c_{22} = c_{21} = 0.01$, $\{a_1, a_2\} = \{0.47, 0.76\}$, $\{r_1, r_2\} = \{0.024, 0.023\}$. At the boundary, we assume $\overrightarrow{n}.\nabla u = 0$ and $\overrightarrow{n}.\nabla v = 0$. The initial profile is chosen as follows:

$$u(0, x, y) = \begin{cases} 10, & (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 \leq 16, \\ 0, & otherwise. \end{cases} \tag{16}$$



(a) The solution at time $t = 0$

(b) The solution at time $t = 0.5$

(c) The solution at time $t = 2$

(d) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N

(e) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold

(f) CPU time for computing the solution at the time $t = 2^m \Delta t$

Fig. 5. Results for the test problem 3.

Table 8
The efficiency of FCFD for problem 4.

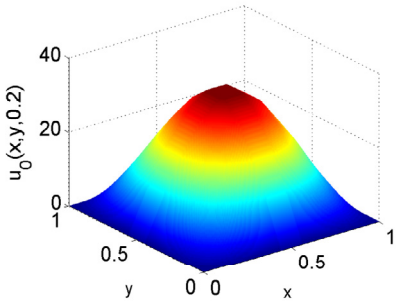
Threshold	10^{-3}	10^{-5}	10^{-6}
CPU time taken (in seconds)	39.9063	41.0938	41.5313
Θ	1.048	1.018	1.007

Table 9
Order of accuracy for the test problem 4.

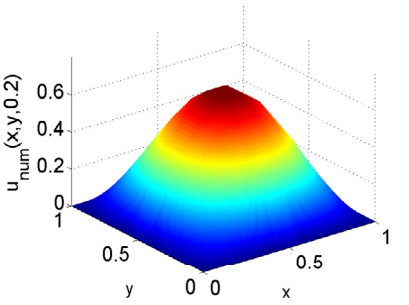
Number of grid points N	216	343	512	1000
Order of accuracy α	-	2.4822	2.3903	2.7824

$$v(0,x,y) = \begin{cases} 10, & (x-\frac{1}{2})^2 + (y-\frac{1}{2})^2 \geq 4, \\ 0, & otherwise. \end{cases} \tag{17}$$

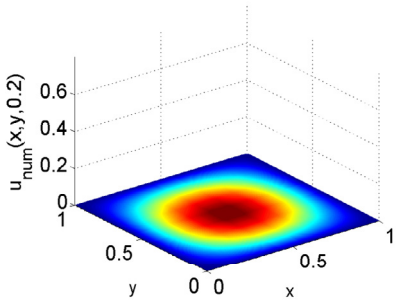
Fig. 7(*a*), 7(*b*) shows the numerical solution of prey population density at time $t = 1$ and $t = 2$ respectively. Fig. 7(*c*), 7(*d*) shows the numerical solution of predator population density at time $t = 1$ and $t = 2$ respectively. Fig. 7(*e*) shows the error (*i.e.*, $\|u_i^M - u_{2i}^{2M}\|_2$) with respect to number of grid points. Fig. 7(*f*) plots $\|u_i^M - u_{2i}^{2M}\|_2$ as a function of threshold. Fig. 7(*g*)



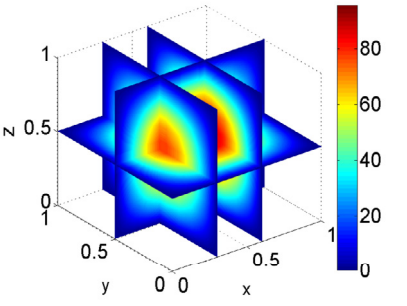
(a) The solution $u(x, y, 0.2)$ at time $t = 0$



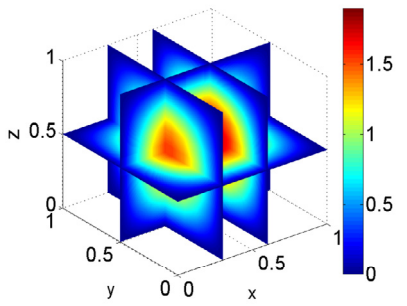
(b) The solution $u(x, y, 0.2)$ at time $t = 0.1316$



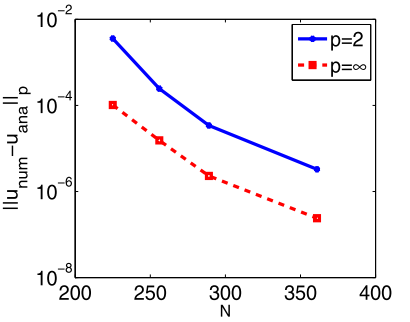
(c) The solution $u(x, y, 0.2)$ at time $t = 4.213$



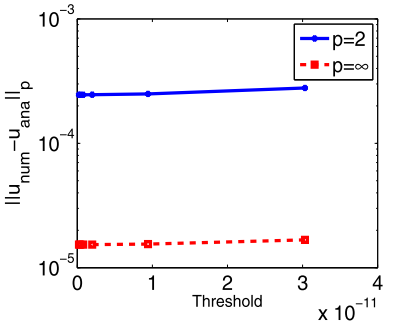
(d) The solution $u(x, y, z)$ at time $t = 0$



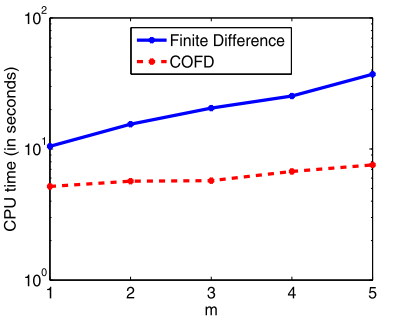
(e) The solution $u(x, y, z)$ at time $t = 0.1316$



(f) $\|u_{num} - u_{ana}\|_p$ versus N

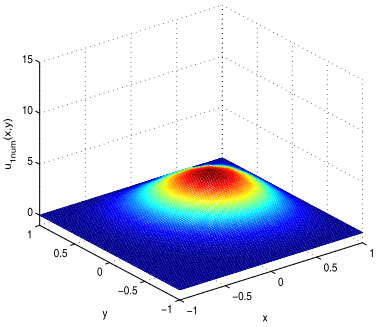


(g) $\|u_{num} - u_{ana}\|_p$ versus threshold

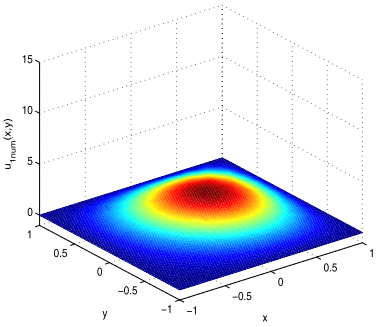


(h) CPU time for computing the solution at the time $t = 2^m \Delta t$

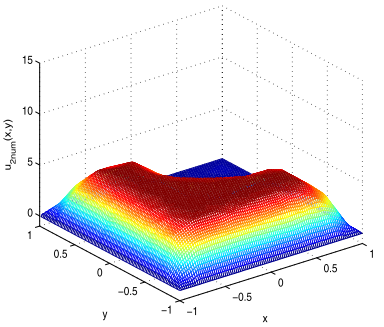
Fig. 6. Results for the test problem 4.



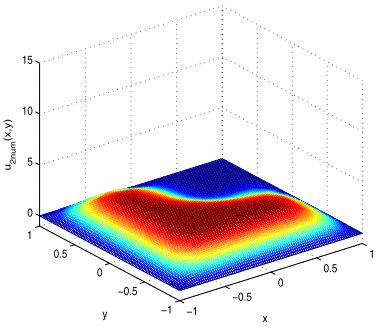
(a) The solution $u(x, y, t)$ at time $t = 1$



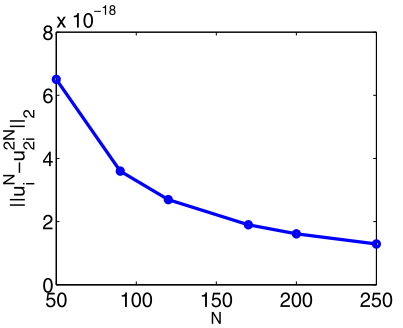
(b) The solution $u(x, y, t)$ at time $t = 2$



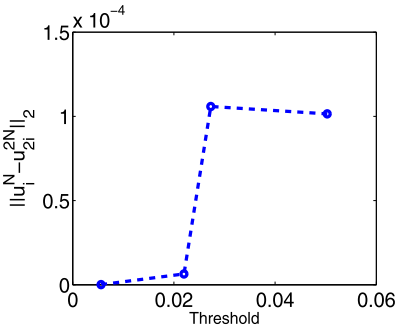
(c) The solution $v(x, y, t)$ at time $t = 1$



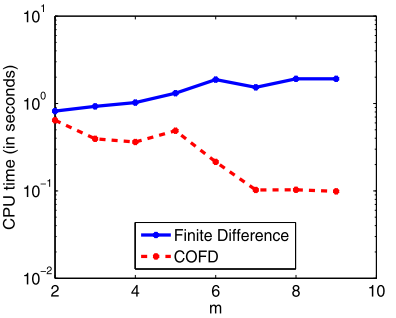
(d) The solution $v(x, y, t)$ at time $t = 2$



(e) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N



(f) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold



(g) CPU time for computing the solution at the time $t = 2^m \Delta t$

Fig. 7. Results for the test problem 5.

Table 10
The performance of FCFD for test problem 5.

Threshold	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.3667	0.5379	0.6379
Θ	1.894	1.301	1.089

Table 11
Order of accuracy for the test problem 5.

Number of grid points N	150	200	250	300
Order of accuracy α	-	1.6338	1.8215	2.5900

compares the CPU time taken FCFD and FDM. Table 10 gives the variation of computational time taken with Θ. Table 11 shows the α of the developed method.

6. Conclusion and future work

A fast curvelet based FDM has been devised for finding the numerical solutions of PDEs. The differential operators are approximated using finite difference matrices and curvelets are used for compressing these matrices. The method is applied on five test problems and it is found that the developed method is computationally very efficient. In future, the proposed method can be applied for solving PDEs on complex manifolds.

Acknowledgements

The first author would like to thank Council of Scientific and Industrial Research, New Delhi, India for providing financial support under Senior Research Fellowship scheme with File No. 09/677(0038)/2019-EMR-I. The second author is grateful to Science and Engineering Research Board (DST) for MTR/2017/000619 grant in support of this research work and TIET for Seed grant.

References

[1] A. Barinka, T. Barsch, P. Charton, A. Cohen, S. Dahlke, W. Dahmen, K. Urban, Adaptive wavelet schemes for elliptic problems - implementation and numerical experiments, SIAM J. Sci. Comput. 23 (1999) 910–939.
[2] S. Bertoluzza, G. Naldi, J.C. Ravel, Wavelet methods for numerical solution of boundary value problems on the interval, in: C. Chui, L. Montefusco, L. Puccio (Eds.), Wavelets: Theory, Algorithms and Applications, Academic Press, New York, 1994, pp. 425–448.
[3] G. Beylkin, R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms, Commun. Pure Appl. Math. 44 (1991) 141–183.
[4] E.J. Candes, L. Demanet, The curvelet representation of wave propagators is optimally sparse, Commun. Pure Appl. Math. 58 (2005) 1472–1528.
[5] E.J. Candes, D.L. Donoho, Curvelets - a surprisingly effective nonadaptive representation for objects with edges, in: Curves and Surfaces, Vanderbilt University Press, Nashville, TN, 2000, pp. 105–120.
[6] E.J. Candes, D.L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities, Commun. Pure Appl. Math. 57 (2004) 219–266.
[7] E.J. Candes, D.L. Donoho, Continuous curvelet transform: i. Resolution of the wavefront set, Appl. Comput. Harmon. Anal. 19 (2005) 162–197.
[8] E.J. Candes, L. Demanet, D.L. Donoho, L. Ying, Curvelab Toolbox, Version 2.1.3, CIT, 2005.
[9] E.J. Candes, L. Demanet, D.L. Donoho, L. Ying, Fast discrete curvelet transforms, Multiscale Model. Simul. 5 (2006) 861–899.
[10] A.C. Canuto, K. Urban, The wavelet element method. Part i: construction and analysis, Appl. Comput. Harmon. Anal. 6 (1999) 1–5.
[11] A. Cohen, W. Dahmen, R. DeVore, Adaptive wavelet methods for elliptic operator equations: convergence rates, Math. Comput. 70 (2001) 27–75.
[12] S. Dahlke, W. Dahmen, R. Hochmuth, R. Schneider, Stable multiscale bases and local error estimation for elliptic problems, Appl. Numer. Math. 23 (1997) 21–47.
[13] W. Dahmen, A. Kunoth, Multilevel preconditioning, Numer. Math. 63 (1992) 315–344.
[14] W. Dahmen, R. Schneider, Wavelets on manifolds i: construction and domain decomposition, SIAM J. Math. Anal. 31 (1999) 184–230.
[15] W. Dahmen, R. Stevenson, Element-by-element construction of wavelets satisfying stability and moment conditions, SIAM J. Numer. Anal. 37 (1999) 319–352.
[16] W. Dahmen, K. Urban, J. Vorloeper, Adaptive Wavelet Methods: Basic Concepts and Applications to the Stokes Problem, World Scientific, Singapore, 2002, pp. 39–80.
[17] W. Freeden, M. Schreiner, Orthogonal and non-orthogonal multiresolution analysis, scale discrete and exact fully discrete wavelet transform on the sphere, Constr. Approx. 14 (1997) 493–515.
[18] K. Goyal, M. Mehra, An adaptive meshfree diffusion wavelet method for partial differential equations on the sphere, J. Comput. Phys. 272 (2014) 747–771.
[19] K. Goyal, M. Mehra, Fast diffusion wavelet method for partial differential equations, Appl. Math. Model. 40 (2016) 5000–5025.
[20] K. Goyal, M. Mehra, An adaptive meshfree spectral graph wavelet method for partial differential equations, Appl. Numer. Math. 113 (2017) 168–185.
[21] M. Holmstrom, J. Walden, Adaptive wavelet methods for hyperbolic PDE's, J. Comput. Phys. 13 (1998) 19–49.
[22] S. Jaffard, Wavelet methods for fast resolution of elliptic problems, SIAM J. Numer. Anal. 29 (1992) 965–986.
[23] J. Liandrat, P. Tchamitchian, Resolution of the 1D Regularized Burgers Equation Using a Spatial Wavelet Approximation, NASA CR-187480, ICASE report No. 90-83, 1990.
[24] J. Ma, G. Tang, M.Y. Hussaini, A refining estimation for adaptive solution of wave equation based on curvelets, Proc. SPIE Wavelets XII 6701 (2007).
[25] S. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. Pattern Anal. Mach. Intell. 11 (1989) 674–693.
[26] S. Mallat, Multifrequency channel decompositions of images and wavelet models, IEEE Trans. Signal Process. 37 (1989) 2091–2110.
[27] S. Mallat, A Wavelet Tour of Signal Processing: The Sparse Way, Academic Press, Burlington, MA, 2009.

[28] M. Mehra, B.V.R. Kumar, Time-accurate solution of advection-diffusion problems by wavelet-Taylor-Galerkin method, Commun. Numer. Methods Eng. 21 (2005) 313–326.
[29] O. Roussel, K. Schneider, A. Tsigulin, H. Bockhorn, A conservative fully adaptive multiresolution algorithm for parabolic PDE's, J. Comput. Phys. 188 (2003) 493–523.
[30] K. Schneider, F. Chemie, T. Chemie, J. Frohlich, J. Frohlich, An adaptive wavelet-vaguelette algorithm for the solution of PDEs, J. Comput. Phys. 130 (1997) 90–174.
[31] D. Sharma, K. Goyal, Second-generation wavelet optimized finite difference method (SGWOFD) for solution of Burger's equation with different boundary conditions, Int. J. Wavelets Multiresolut. Inf. Process. 16 (2018) 1850032.
[32] D. Sharma, K. Goyal, Spectral graph wavelet optimized finite difference method for solution of Burger's equation with different boundary conditions, J. Differ. Equ. Appl. 25 (2019) 373–395.
[33] O.V. Vasilyev, C. Bowman, Second generation wavelet collocation method for the solution of partial differential equations, J. Comput. Phys. 165 (2000) 660–693.
[34] L. Ying, L. Demanet, E. Candes, 3D discrete curvelet transform, Proc. SPIE Wavelets XI 5914 (2005).

GEOPHYSICS, VOL. 83, NO. 6 (NOVEMBER-DECEMBER 2018); P. T333–T344, 12 FIGS., 4 TABLES.
10.1190/GEO2017-0710.1

3D numerical simulation of elastic waves with a frequency-domain iterative solver

Mikhail Belonosov[1], Victor Kostin[2], Dmitry Neklyudov[2], and Vladimir Tcheverda[2]

ABSTRACT

The efficiency of any inversion method for estimating the medium parameters from seismic data strongly depends on simulation of the wave propagation, i.e., forward modeling. The requirements are that it should be accurate, fast, and computationally efficient. When the inversion is carried out in the frequency domain (FD), e.g., FD full-waveform inversion, only a few monochromatic components are involved in the computations. In this situation, FD forward modeling is an appealing potential alternative to conventional time-domain solvers. Iterative FD solvers, based on a Krylov subspace iterative method, are of interest due to their moderate memory requirements compared with direct solvers. A huge issue preventing their successful use is a very slow convergence. We have developed an iterative solver for the elastic wave propagation in 3D isotropic heterogeneous land models. Its main ingredient is a novel preconditioner, which provides the convergence of the iteration. We have developed and justified a method to invert our preconditioner effectively on the base of the 2D fast Fourier transform and solving a system of linear algebraic equations with a banded matrix. In addition, we determine how to parallelize our solver using the conventional hybrid parallelization (MPI in conjunction with OpenMP) and demonstrate the good scalability for the widespread 3D SEG/EAGE overthrust model. We find that our method has a high potential for low-frequency simulations in land models with moderate lateral variations and arbitrary vertical variations.

INTRODUCTION

Numerical simulation of seismic wave propagation is the backbone of many technologies being developed in exploration geophysics to recover the reflectivity and the velocity model of the subsurface. For instance, applied several times at each iteration of full-waveform inversion (FWI) (Pratt, 1999; Symes, 2008; Virieux et al., 2009), it is greatly responsible for the total computational time and the accuracy of an output velocity.

Most of current successful solutions for 3D large-scale numerical simulation applied in the industry are time-domain methods (Sirgue et al., 2007; Etienne et al., 2014, Kostin et al., 2015, Lisitsa et al., 2016). Advances in computational technologies, known as the "supercomputing era," gave an impulse for development of an alternative approach — frequency-domain (FD) methods. For FWI applications, when only several frequencies are needed, they may be a good option for forward modeling. Discretizing the wave equation by finite-difference or finite-element approximations creates a system of linear algebraic equations (SLAE). One approach is to solve these equations by a direct method (Operto et al., 2007). For 3D problems, the main bottleneck is the hundreds of gigabytes of memory required, even for acoustic media, making it unrealistic to be applied in production. Attempts to resolve this issue include applying data compression techniques based on the block low-rank approach, hierarchically semiseparable formats of storing data, and low-rank approximation of matrices (Wang et al., 2012; Weisbecker et al., 2013; Kostin et al., 2017).

An alternative to the direct approach is an iterative approach based on a Krylov-type iterative method (Saad, 2003). Its memory requirements are much more modest. However, the indefiniteness of the coefficient matrix in seismic applications leads to slow convergence. As a remedy, an appropriate preconditioner might be

Manuscript received by the Editor 27 October 2017; revised manuscript received 19 June 2018; published ahead of production 30 August 2018; published online 23 October 2018.
[1]Aramco Overseas Company B.V., Aramco Research Center — Delft, Informaticalaan 6, Delft 2628 ZD, the Netherlands. E-mail: mikhail.belonosov@aramcooverseas.com.
[2]Trofimuk Institute of Petroleum Geology and Geophysics SB RAS, 3, Koptyug Ave, Novosibirsk 630090, Russia. E-mail: kostinvi@ipgg.sbras.ru; dmitn@mail.ru; cheverdava@ipgg.sbras.ru.

applied. For instance, for the acoustic wave equation, the shifted Laplace operator (Oosterlee et al., 2010) is an option.

Onshore seismic data contain all types of seismic waves, with P-waves often being the weakest part of the recorded signal. Inversion of P-waves requires careful suppression of other events in the data because they are considered as noise. Removing noise is a challenging task that can result in removal of the signal as well (Bakulin et al., 2015). This is one of the motivations to develop elastic FWI to use some of the other modes recorded, such as shear/surface waves to provide an accurate velocity model. This type of inversion requires elastic forward modeling.

A few 3D elastic iterative solvers have been developed so far (Darbas and Louer, 2013; Li et al., 2015; Rizzuti and Mulder, 2015). The method proposed in this paper extends the approach from Belonosov et al. (2017) developed for the 3D acoustic wave equation, in which the preconditioner is a complex damped Helmholtz operator with some 1D vertically heterogeneous background. This was developed for FWI applications for macrovelocity reconstruction and showed fast convergence at low frequencies.

In this paper, we build the preconditioner following a similar methodology with special modifications for the elastic operator. We show the effectiveness of our method for cases of moderate lateral velocity variations in the overburden in Saudi Arabian land seismic data. Vertical variations of any kind are acceptable.

To use the capabilities of modern high-performance computing (HPC) systems, we parallelize our method in a hybrid manner involving message passing interface (MPI) and open multiprocessing (OpenMP). The speed of any parallel algorithm is strongly dependent on how effectively computation can be allocated among the available hardware. In that regard, strong scaling (Colella et al., 2007) is an important characteristic of a parallel method reflecting the ability to decrease the runtime with the increasing number of MPI processes/OpenMP threads involved in the computations. We provide scalability and performance data for our method.

A brief structure of the paper is as follows: We begin with a mathematical statement of the problem and describe the algorithm with details provided in Appendices A and B. Then, we validate the method via examples of simulation using different models. Finally, we summarize and provide with possible directions for further development of the method.

BASICS OF THE METHOD

Statement of the problem

Consider the first-order system of 3D isotropic elasticity equations in the temporal-FD written in terms of compliance

$$\left[i\omega\mathbf{M}(x,y,z)-\mathbf{P}\frac{\partial}{\partial x}-\mathbf{Q}\frac{\partial}{\partial y}-\gamma(z)\mathbf{R}\frac{\partial}{\partial z}\right]\mathbf{u}=\mathbf{f}(x,y,z), \quad (1)$$

where $\mathbf{u}=(u_x,u_y,u_z,\sigma_{xx},\sigma_{yy},\sigma_{zz},\sigma_{yz},\sigma_{xz},\sigma_{xy})^T$ is a vector function of the particle velocity components (u_x,u_y,u_z) and components of the stress tensor $(\sigma_{xx},\sigma_{yy},\sigma_{zz},\sigma_{yz},\sigma_{xz},\sigma_{xy})$, ω is the real angular frequency

$$\mathbf{M}=\begin{pmatrix}\rho(x,y,z)\mathbf{I}_{3\times3} & 0\\ 0 & \mathbf{S}(x,y,z)\end{pmatrix}, \quad (2)$$

where ρ is the density, $\mathbf{I}_{3\times3}$ is the 3-by-3 identity matrix, and $\mathbf{S}(x,y,z)$ is the compliance 6-by-6 matrix

$$\mathbf{S}(x,y,z)=\begin{pmatrix} a & -b & -b & & & \\ -b & a & -b & & 0 & \\ -b & -b & a & & & \\ & & & c & 0 & 0\\ & 0 & & 0 & c & 0\\ & & & 0 & 0 & c\end{pmatrix},$$

$$a(x,y,z)=\frac{\lambda+\mu}{\mu(2\mu+3\lambda)},\quad b(x,y,z)=\frac{\lambda}{2\mu(2\mu+3\lambda)},$$

$$c(x,y,z)=\frac{1}{\mu}, \quad (3)$$

where λ and μ are the Lamé parameters. The $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$ are 9-by-9 constant matrices of the following structures:

$$\mathbf{P}=\begin{pmatrix}0 & \bar{\mathbf{P}}\\ \bar{\mathbf{P}}^{\mathrm{T}} & 0\end{pmatrix},\quad \bar{\mathbf{P}}=\begin{pmatrix}1&0&0&0&0&0\\0&0&0&0&0&1\\0&0&0&0&1&0\end{pmatrix},$$

$$\mathbf{Q}=\begin{pmatrix}0 & \bar{\mathbf{Q}}\\ \bar{\mathbf{Q}}^{\mathbf{T}} & 0\end{pmatrix},\quad \bar{\mathbf{Q}}=\begin{pmatrix}0&0&0&0&0&1\\0&1&0&0&0&0\\0&0&0&1&0&0\end{pmatrix},$$

$$\mathbf{R}=\begin{pmatrix}0 & \bar{\mathbf{R}}\\ \bar{\mathbf{R}}^{\mathrm{T}} & 0\end{pmatrix},\quad \bar{\mathbf{R}}=\begin{pmatrix}0&0&0&0&1&0\\0&0&0&1&0&0\\0&0&1&0&0&0\end{pmatrix} \quad (4)$$

where $\gamma(z)$ is a damping function responsible for the perfectly matched layers (PMLs) (Berenger, 1996) along the vertical direction. The term $\mathbf{f}(x,y,z)$ is the right side representing a seismic source.

Elastic equation 1 is solved in a cuboid domain $D=[0,X]\times[0,Y]\times[0,Z]$ filled with a heterogeneous medium. The top boundary of the domain is the free surface unless stated to be absorbing. That means $\sigma_{zz}=\sigma_{yz}=\sigma_{xz}=0$ for $z=0$ or, equivalently, $\mathbf{B}_0\mathbf{u}(x,y,0)=0$, where

$$\mathbf{B}_0=\begin{pmatrix}0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1&0\end{pmatrix}. \quad (5)$$

The computational domain is assumed to include the PML on the bottom; i.e., interval $[0,Z]$ consists of two subintervals, where $[0,Z_1]$ is the actual computational domain and $[Z_1,Z]$ is the PML with the boundary condition on boundary $z=Z$ (Collino and Tsogka, 1998): $u_z=\sigma_{yz}=\sigma_{xz}=0$. The latter condition could be rewritten in an equivalent form $\mathbf{B_Z}\mathbf{u}(x,y,Z)=0$, where

$$\mathbf{B_Z}=\begin{pmatrix}0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1&0\end{pmatrix}. \quad (6)$$

On the lateral boundaries, domain D includes specially constructed sponge zones with periodic boundary conditions (for details, refer to Belonosov et al., 2017)

$$\bar{\tilde{\mathbf{u}}}(0,y,z)=\bar{\tilde{\mathbf{u}}}(X,y,z),\quad \boldsymbol{\sigma}_x(0,y,z)=\boldsymbol{\sigma}_x(X,y,z),$$
$$\bar{\tilde{\mathbf{u}}}(x,0,z)=\bar{\tilde{\mathbf{u}}}(x,Y,z),\quad \boldsymbol{\sigma}_y(x,0,z)=\boldsymbol{\sigma}_y(x,Y,z), \quad (7)$$

with $\bar{\tilde{\mathbf{u}}}=(u_x,u_y,u_z)^T$, $\boldsymbol{\sigma}_x=(\sigma_{xx},\sigma_{xy},\sigma_{xz})^T$, and $\boldsymbol{\sigma}_y=(\sigma_{xy},\sigma_{yy},\sigma_{yz})^T$. These conditions arise from the 2D fast Fourier transform (FFT) by the lateral directions being applied later. Also, it is worth explaining that its application does not allow using the PML in the x- and y-directions. Schematically, the structure of computational domain D is illustrated in Figure 1.

To numerically solve the boundary value problem for equation 1, we use the Krylov-type iterative method. In most cases, its straightforward application does not guarantee convergence. To handle this issue, a strategy based on a specially designed preconditioner is applied.

Preconditioner

Denote the left side of elastic equation 1 by $\mathcal{L}$

$$\mathcal{L}=i\omega\mathbf{M}(x,y,z)-\mathbf{P}\frac{\partial}{\partial x}-\mathbf{Q}\frac{\partial}{\partial y}-\gamma(z)\mathbf{R}\frac{\partial}{\partial z}. \quad (8)$$

Note that we also use notation $\mathcal{L}$ for the operator defined by differential expression 8 subject to the boundary conditions described above.

Following the form of a preconditioner developed for the 2D elastic wave equation in (Neklyudov et al., 2011), consider an auxiliary, depth-dependent operator $\mathcal{L}_0$ subject to the same boundary conditions as operator $\mathcal{L}$

$$\mathcal{L}_0=i\omega\mathbf{M}_0(z)-\mathbf{P}\frac{\partial}{\partial x}-\mathbf{Q}\frac{\partial}{\partial y}-\gamma(z)\mathbf{R}\frac{\partial}{\partial z}, \quad (9)$$

with complex-valued matrix $\mathbf{M}_0$

$$\mathbf{M}_0(z)=\begin{pmatrix}\rho_0(z)\mathbf{I}_{3\times3} & 0\\ 0 & (1+i\beta)\mathbf{S}_0(z)\end{pmatrix}, \quad (10)$$

where

$$\mathbf{S}_0(z)=\begin{pmatrix} a_0(z) & -b_0(z) & -b_0(z) & & & \\ -b_0(z) & a_0(z) & -b_0(z) & & 0 & \\ -b_0(z) & -b_0(z) & a_0(z) & & & \\ & & & c_0(z) & 0 & 0\\ & 0 & & 0 & c_0(z) & 0\\ & & & 0 & 0 & c_0(z)\end{pmatrix}, \quad (11)$$

where β is a positive real number smaller than one and is responsible for $\mathbf{M}_0(z)$ complex shifting by analogy with the shifted Laplacian (Erlangga and Nabben, 2008a). At each depth level z, the values of functions $\rho_0(z)$, $a_0(z)$, $b_0(z)$, and $c_0(z)$ are defined as some averaging of functions $\rho(x,y,z)$, $a(x,y,z)$, $b(x,y,z)$, and $c(x,y,z)$ along the horizontal directions. In general, $\rho_0(z)$, $a_0(z)$, $b_0(z)$, and $c_0(z)$ might be functions of any kind, but satisfying two conditions, where $\rho_0(z)>0$ and $\mathbf{S}_0(z)$ is positive definite. We define them in a certain way to provide the proximity of operator $\mathcal{L}_0$ to the original operator $\mathcal{L}$ to ensure convergence of the Krylov-type iterative method. In fact, operator $\mathcal{L}_0(z)$ is of exactly the same structure as operator $\mathcal{L}(x,y,z)$, but with complex-valued coefficients that act in a 1D varying medium. In the next section, we illustrate a computationally efficient numerical algorithm to solve the boundary value problems for equation 1 with operator $\mathcal{L}_0$ used instead of $\mathcal{L}$, or, in other words — how to invert operator $\mathcal{L}_0$.

We use this operator as a preconditioner to solve the boundary value problem for equation 1 that leads to the equivalent equation

$$\mathcal{L}\mathcal{L}_0^{-1}\mathbf{v}=\mathbf{f}, \quad (12)$$

over unknown vector $\mathbf{v}$, such that

$$\mathbf{u}=\mathcal{L}_0^{-1}\mathbf{v}. \quad (13)$$

We expect operator $\mathcal{L}\mathcal{L}_0^{-1}$ to have better spectral properties than operator $\mathcal{L}$ and by this to improve convergence of the iterative method. An illustration on how a successful preconditioner works can be found in Belonosov et al. (2017, Figure 1).

Even though the Krylov-type iterative method converged fast, the second requirement for successful application would be to perform multiplication of operator $\mathcal{L}\mathcal{L}_0^{-1}$ by a vector at each iteration for a reasonable time. We represent operator $\mathcal{L}$ as a perturbation of operator $\mathcal{L}_0$ by operator δL

$$\mathcal{L}\mathbf{u}=\mathcal{L}_0\mathbf{u}-\delta\mathcal{L}\mathbf{u}=\left[i\omega\mathbf{M}_0(z)-\mathbf{P}\frac{\partial}{\partial x}-\mathbf{Q}\frac{\partial}{\partial y}-\gamma(z)\mathbf{R}\frac{\partial}{\partial z}\right]\mathbf{u}$$
$$-i\omega[\mathbf{M}_0(z)-\mathbf{M}(x,y,z)]\mathbf{u}. \quad (14)$$

Substitution of expansion 14 into preconditioned equation 12 transforms it into

$$(I-\delta\mathcal{L}\mathcal{L}_0^{-1})\mathbf{v}=\mathbf{f}, \quad (15)$$

where I is the unity operator.

To solve equation 15, we apply the biconjugate gradient stabilized method (BiCGSTAB) (Van der Vorst, 1992; Saad, 2003) because it is the one with the lowest memory requirements of the available

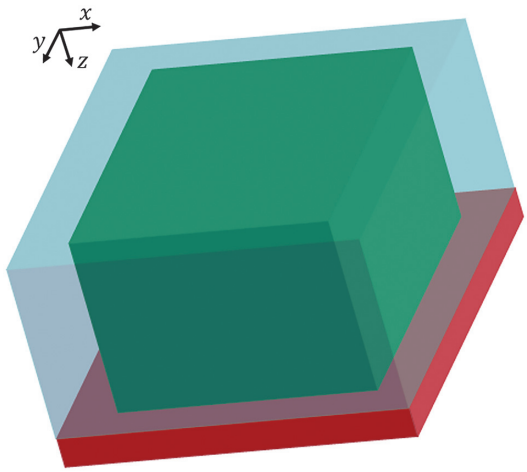


Figure 1. Computational domain D showing the physical domain (green area), sponge boundaries (blue), and PML (red).

methods. At each iteration, BiCGSTAB requires two matrix-by-vector products.

Computation of $\mathcal{L}_0^{-1}\mathbf{w}$

Computation of $\mathbf{u} = \mathcal{L}_0^{-1}\mathbf{w}$, where $\mathbf{w}$ is an arbitrary vector is equivalent to solving the boundary value problem for the equation

$$\left[i\omega\mathbf{M}_0(z) - \mathbf{P}\frac{\partial}{\partial x} - \mathbf{Q}\frac{\partial}{\partial y} - \gamma(z)\mathbf{R}\frac{\partial}{\partial z}\right]\mathbf{u}(x,y,z) = \mathbf{w}(x,y,z) \tag{16}$$

in domain D with the same boundary conditions as are set on the initial elastic equation 1.

Assume that function $\mathbf{w}(x,y,z)$ is expanded into the Fourier series with respect to the x- and y-coordinates and denote its coefficients by $\hat{\mathbf{w}}(k_x,k_y;z)$, where k_x and k_y are the respective spatial frequencies. Then, solving the boundary value problem for equation 16 could be broken down into two parts. First, we search for $\hat{\mathbf{u}}(k_x,k_y;z)$, which are coefficients of the Fourier series for function $\mathbf{u}(x,y,z)$. They are solutions to the ordinary differential equation (ODE) boundary value problems

$$\left[i\omega\mathbf{M}_0(z) - ik_x\mathbf{P} - ik_y\mathbf{Q} - \gamma(z)\mathbf{R}\frac{d}{dz}\right]\hat{\mathbf{u}}(k_x,k_y;z) = \hat{\mathbf{w}}(k_x,k_y;z),$$
$$\mathbf{B}_0\hat{\mathbf{u}}(k_x,k_y;0) = 0,$$
$$\mathbf{B}_Z\hat{\mathbf{u}}(k_x,k_y;Z) = 0, \tag{17}$$

on the interval $[0,Z]$. The existence problem is addressed in Appendix A. Second, function $\mathbf{u}(x,y,z)$ is reconstructed by computation of the Fourier series.

Numerically, for $\mathbf{w}$ being a grid vector-function defined on a uniform grid along the x- and y -axes, computation of $\hat{\mathbf{w}}(k_x,k_y;z)$ is the 2D discrete Fourier transform (DFT) of function $\mathbf{w}(x,y,z)$ over the lateral coordinates. The Fourier series for $\mathbf{u}(x,y,z)$ reduces to a finite sum that is the 2D inverse DFT of $\hat{\mathbf{u}}(k_x,k_y;z)$. To compute the DFTs, we apply a computationally efficient FFT from the highly optimized software library Intel MKL (Intel, 2017). The boundary value problems 17 are solved numerically by finite-difference approximation that results in SLAEs with small, banded matrices (for details, see Appendix B) solved using a banded matrix solver from the Intel MKL library.

To sum up, the algorithm for matrix-by-vector multiplication is as follows:

- forward 2D FFT over lateral coordinates to vector-function $\mathbf{w}$
- numerical solution of the ODE boundary value problems 17
- inverse 2D FFT to obtain the matrix-by-vector product.

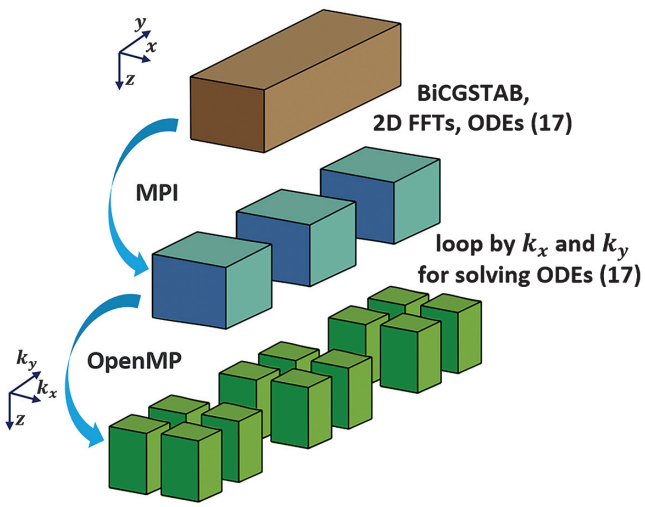


Figure 2. Hybrid parallelization scheme for the elastic iterative solver.

Hybrid parallelization

Numerical simulations of seismic waves in a 3D model were carried out using high-performance computing clusters. To optimize this process, different parallelization strategies, depending on architecture available might be applied. An important advantage of the proposed method is that it allows hybrid parallelization: MPI in conjunction with OpenMP. This allows more effective allocating parallel processes among available computational resources.

A parallelization strategy for our method is illustrated in Figure 2 and comprises two levels. The first level is MPI parallelization via spatial decomposition (the initial brown domain is decomposed into several smaller blue subdomains) of the BiCGSTAB, the 2D FFT and solving the boundary value problems 17. The 2D FFT version that we use supports parallelization along one of coordinates only. The second level is OpenMP parallelization within each blue subdomain: each OpenMP thread is responsible for its own set of k_x and k_y in boundary value problem 17.

According to our scheme, we expect that the main speed-up will be from MPI parallelization because it is a "global" one. OpenMP is used at each iteration, but "locally," only for a minor part of the code. It might be beneficial in cases in which involvement of new MPI processes no longer accelerates the computational process, but some CPU cores are still available.

NUMERICAL EXPERIMENTS

All results presented in the paper have been computed on a HPC cluster comprising nodes each having two Intel Xeon E5-2680v4 at 2400 MHz CPUs. The double precision floating point format has been used in the computations. As a stopping criterion for the BiCGSTAB, we used a 10^{-3} threshold for the relative residual of the L_2-norm providing enough accuracy for FWI applications. We use a vertical point force as the seismic source.

A proper nondimensionalization is necessary to be applied to equation 1. This is a common practice performed in applications when different variables are at very different scales. In our case, the difference between computed stresses and velocities is of order 10^6. To minimize this issue, we scale velocities by a factor of 10^3 and the stress components by a factor of 10^{-3}.

Accuracy analysis

At first, the wavefield computed via the solver in a homogeneous model with the PML at the top boundary was compared with the analytical solution (Aki and Richards, 1980). A domain of $12 \times 12 \times 4.5$ km is filled with a homogeneous elastic medium of 2600 m/s P-wave velocity, 1500 m/s S-wave velocity, and 2210 kg/m^3 den-

sity. For numerical computations, the domain was discretized using a uniform grid with lateral cell sizes of 60 m and vertical cells of 15 m. The source was excited in the middle of the model at 15 m depth. A 1D profile comparison of the 10 Hz monochromatic part of the vertical velocity component is presented in Figure 3. All traces are normalized to their maximum value. There is a very good agreement between the analytic and numerical modeling results (the root-mean-square [rms] deviation of the computed solution from the exact solution along the 1D profile in percent is 0.88%). Note that in this case, the lateral steps correspond to 2.5 points per minimum wavelength and the vertical cell size is 10 points.

For further verification, we have benchmarked the solver against a known time-domain solver based on explicit fourth-order finite differences for the 3D SEG/EAGE overthrust model (Aminzadeh et al., 1997) (Figure 4) with the free-surface top boundary. This model is discretized using a $660 \times 660 \times 155$ uniform grid with cell sizes of 30 m in all directions resulting in eight points per minimum wavelength at 5 Hz frequency. Synthetic wavefields excited by a source fired in the middle of the model at 15 m depth were simulated with both methods at a frequency of 5 Hz. In Figure 5, we present the vertical particle velocities along the x-axis as well as along the y-axis. The results are normalized to their maximum value

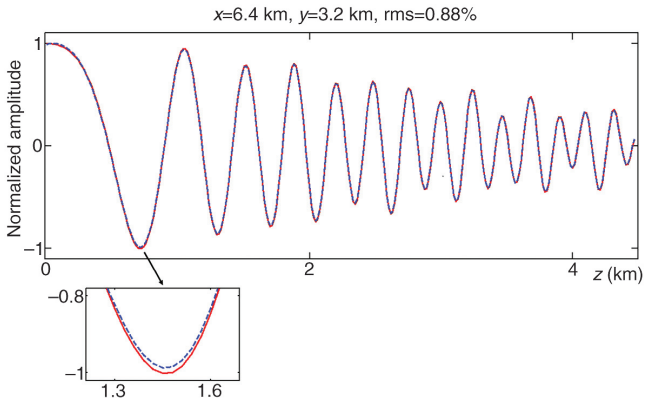


Figure 3. Vertical profile (real part) of the FD z-component of the displacement velocity at 10 Hz computed in the homogeneous model at location $x = 6400$ m, $y = 3200$ m. The exact solution is the dashed blue line and the iterative solver solution is the red line. The arrow points to a magnified part of the picture.

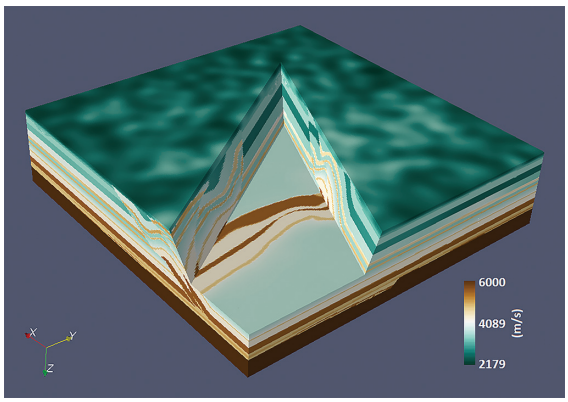


Figure 4. A 3D view of the V_P-velocity distribution in the 3D SEG/EAGE overthrust model ($19.8 \times 19.8 \times 4.65$ km).

and show close agreement (rms difference is 3.9%–4.7%). The total 3D view of the vertical velocity computed by the iterative solver is shown in Figure 6.

Strong scaling analysis

We estimate the strong scaling of our algorithm on an example of numerical simulation at a frequency of 5 Hz in the SEG/EAGE overthrust model (Figure 4). We measure t_1/t_m (see Figure 7), where t_m is the total computational time with m MPI processes. Results show the very good scalability of the algorithm up to 64 CPUs after which scalability worsens. This might be partly explained by the fact that the FFT shows strong scalability decay when the number of points in a subdomain becomes too small (Dmitruk et al., 2001).

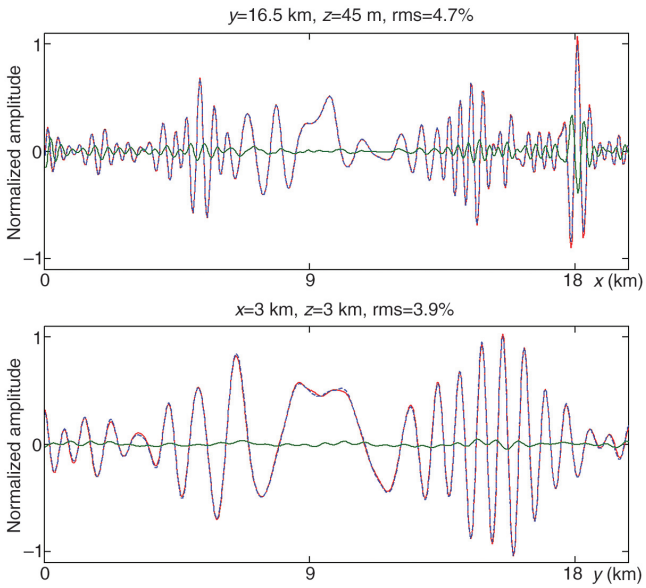


Figure 5. The horizontal profiles of the real part of the vertical component of the 5 Hz FD displacement velocities computed in the SEG/EAGE overthrust model at locations $y = 16{,}500$ m, $z = 45$ m (top) and $x = 3000$ m, $z = 3000$ m (bottom). The TD solver solution is the dashed blue line, the iterative solver solution is in red, and the difference multiplied by five is in green.

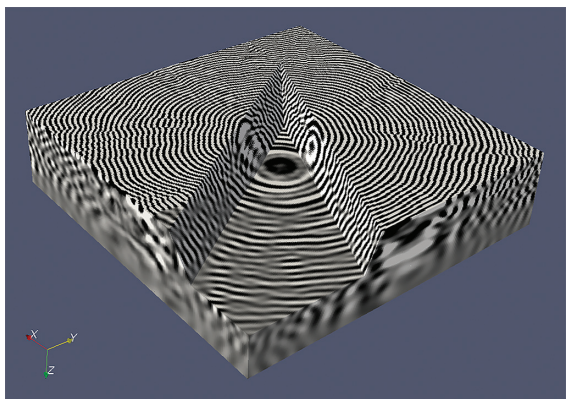


Figure 6. A 3D view of the real part of an FD wavefield of the vertical component of the particle velocity computed for the SEG/EAGE overthrust model.

For the case of OpenMP parallelization, the simulations were performed on a single CPU with 14 cores with hyperthreading switched off and without using MPI. Remember that multithreading is used at each iteration, but only to parallelize the loop over spatial frequencies solving the boundary value problems 17. The strong scaling data measured for that part separately as well as for solving the whole problem are presented in Figure 8 showing that the OpenMP part scales well. The OpenMP contribution to acceleration of the total runtime (the green line), has an upper limit close to two. This is in a good agreement with Amdahl's (1967) law because only a part taking approximately 50% of the execution time is parallelized via OpenMP. According to the results presented in Figures 7 and 8, the speed-up factors in case of pure MPI parallelization with 32 processes is $t_{32}^{\mathrm{MPI}} = 30.2$ and with pure OpenMP with six threads is $t_{6}^{\mathrm{OMP}} = 1.63$. Hybridization results in a speed-up of 46.8 that is very close to $t_{32}^{\mathrm{MPI}} \cdot t_{6}^{\mathrm{OMP}}$.

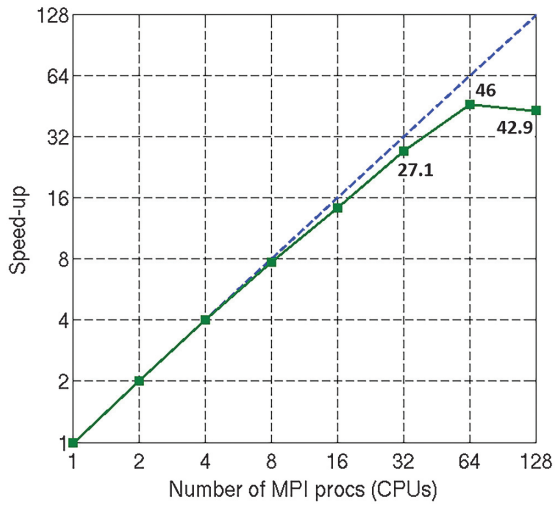


Figure 7. The MPI strong scalability analysis. The ideal speed-up is the dashed blue line, and the iterative solver scalability is in green.

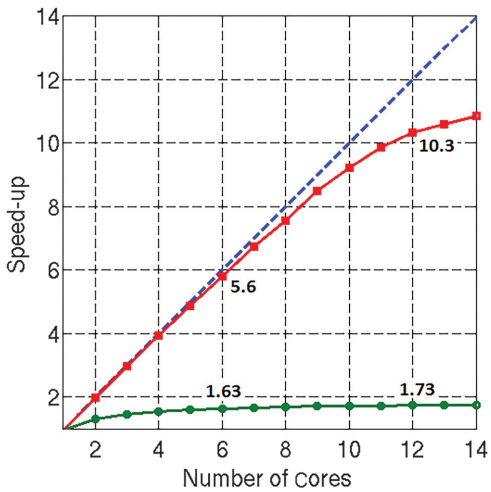


Figure 8. OpenMP strong scalability analysis. The ideal speed-up is the dashed blue line, the iterative solver multithread part speed-up is in red, and the total speed-up for the iterative solver is in green.

The further acceleration and increase of the OpenMP contribution of the total runtime might be achieved by hybrid parallelization of the BiCGSTAB process. Currently used function supports only MPI parallelization.

Comparison with a known direct solver

As previously mentioned, the main advantage of an iterative approach over its FD competitor — direct approach, is that it requires less RAM. For 3D simulations, when hundreds of gigabytes are required that factor may become crucial when choosing a solver. For an example illustrating this fact, we considered a conventional direct method based on the lower–upper (LU) factorization. To decrease the memory needed, we applied it to the second-order elastic equation, approximated by the second-order finite differences. The major RAM required by this method is being spent to store LU factors that we computed via the Intel MKL function — PARDISO (Intel, 2017). In the third column of Table 1, we present the necessary memory being consumed by the described direct solver; in the second column — by our solver. The results show the difference being of around an order of magnitude. Note that for this comparison, the absorbing layers are not included for either method.

Benchmarking against a known iterative solver

For performance analysis, we considered a known 3D elastic iterative solver — CARP-CG (Li et al., 2015). Following the paper, the same simulations were performed in a $14.4 \times 14.4 \times 14.4$ km homogeneous model of 5000 m/s P-wave velocity, 2886.75 m/s S-velocity, and the density was estimated via Gardner's relation. We used the same lateral steps of four points per minimal wavelength. In the vertical direction, we used a smaller step of 10 points per minimal wavelength due to fourth-order finite-difference approximation used in the current version of our solver. It is worth explaining that we considered only the homogeneous model because other 3D simulations in this paper were performed in the inhomogeneous model not available to us.

In Table 2, we demonstrate the number of BiCGSTAB iterations for our solver (second column), as well as the number of iterations for CARP-CG (third column) (these data are taken from Table 9 of Li et al., 2015). Note that the ratio between the elements of the third column and the second column is almost the same (approximately 38) for all frequencies. We do not provide computational times because in our opinion, this comparison may be a bit unfair due to the difference in hardware and software, the code optimizations

Table 1. RAM requirements for the iterative solver and the direct solver ($N_x \times N_y \times N_z$ — the size of the computational domain in points; RAM(It) — RAM consumption of the iterative solver; and RAM(D) — the memory necessary to store LU factors).

$N_x \times N_y \times N_z$	RAM (It)	RAM (D)
$50 \times 50 \times 50$	0.083 GB	13 GB
$75 \times 75 \times 75$	0.28 GB	72 GB
$100 \times 100 \times 100$	0.67 GB	285 GB
$115 \times 115 \times 115$	1 GB	467 GB

applied, and, possibly, some other factors. Having no access to the actual CARP-CG code, it is problematic to estimate its computational time on our hardware and with our settings.

Convergence analysis in a model with strong lateral variations

The convergence of our solver strongly depends on the lateral variations of the medium parameters. To illustrate that, we performed simulations at 5 Hz in the model with a vertical interface comprising two layers of equal size (see Figure 9) with constant parameters $V_P = V_{P_1}$, $V_S = V_{S_1}$, $\rho = \rho_1$ for $x \geq L$ and $V_P = V_{P_2}$, $V_S = V_{S_2}$, $\rho = \rho_2$ for $x < L$ (here, we assume that $V_S = V_P/\sqrt{3}$, ρ is estimated via Gardner's relation and $V_{P_2} > V_{P_1}$). It is discretized with a uniform grid of $400 \times 400 \times 200$ points. The steps correspond to 2.5 points per minimal wavelength in the horizontal directions and 10 points in the vertical direction. The source location is in the middle of the blue subdomain. In Table 3, we illustrate how the convergence depends on the ratio of V_{P_2} (for increasing V_{P_2}) and $V_{P_1} = 2000$ m/s. The ratio of two is an example when the solver diverges.

For further improvement of the convergence, an extra preconditioning might be an option. For example, a similar ideology of a second-level preconditioner developed for the acoustic wave equation (Belonosov et al., 2017) or a preconditioner based on deflation (Erlangga and Nabben, 2008b) might be considered.

Table 2. Our solver versus CARP-CG: N_{solver} — number of BiCGSTAB iterations for our solver and $N_{\text{CARP-CG}}$ — number of CARP-CG iterations.

Frequency	N_{solver}	$N_{\text{CARP-CG}}$
2.5 Hz	14	546
7.5 Hz	39	1402
12.5 Hz	58	2162
17.5 Hz	73	2817
22.5 Hz	87	3389

Table 3. Convergence of the iterative solver for different lateral contrasts in the model presented in Figure 11.

V_{p_2}/V_{p_1}	Number of iterations
1.25	17
1.5	42
1.75	1151
2	>10,000 (divergence)

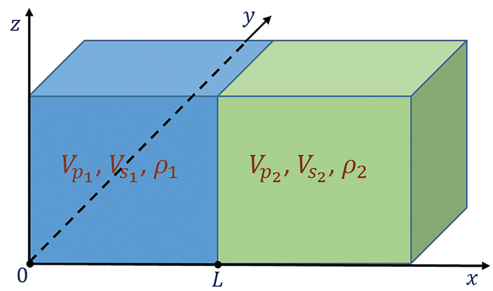


Figure 9. A model with a vertical interface.

Simulation in a realistic synthetic model

As an example of a numerical simulation using a realistic scenario, we considered a land model depicted in Figure 10 with the deep structure common to the eastern province of Saudi Arabia. Its dimensions correspond to one swath size typically used in the land

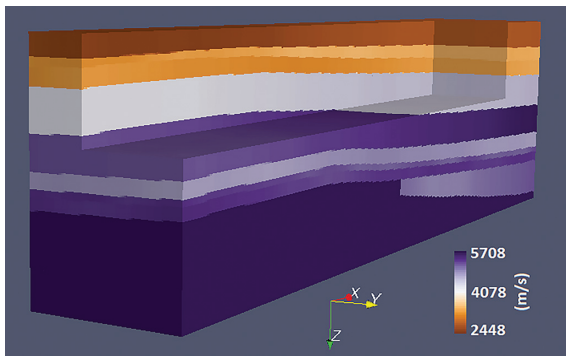


Figure 10. Compressional velocity distribution for a typical Saudi Arabian land model ($24 \times 4.2 \times 6.5$ km deep).

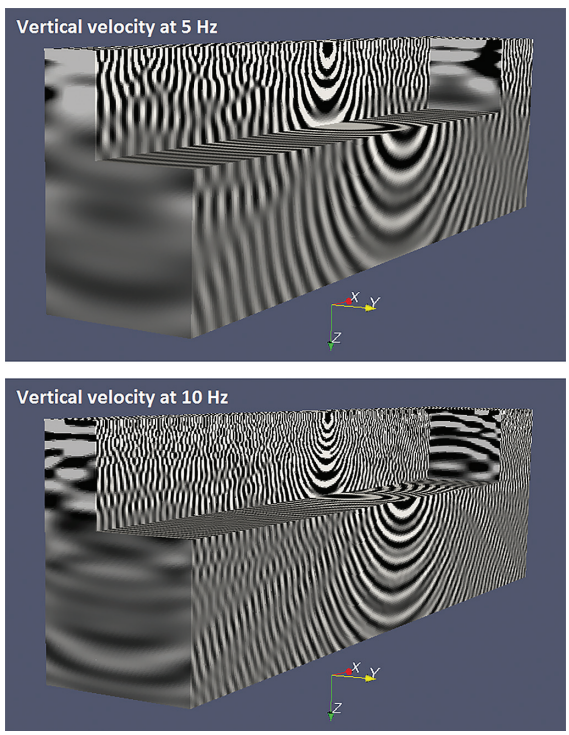


Figure 11. A 3D view of the real part of a FD wavefield of the vertical component of the particle velocity computed for the model depicted in Figure 10.

Table 4. Number of iterations versus frequency for numerical simulation with the elastic iterative solver for the model depicted in Figure 9.

Frequency	2 Hz	4 Hz	6 Hz	8 Hz	10 Hz	12 Hz	14 Hz	16 Hz	20 Hz
Iterations	10	27	58	92	158	229	311	450	1123

seismic acquisition. The model was discretized with a uniform grid of $957 \times 169 \times 651$ points with a lateral cell size of 25 m and a vertical cell size of 10 m. The source was placed in the middle of the area at 10 m depth. In Figure 11, we present a 3D view of the vertical velocity at 5 and 10 Hz computed with the iterative solver. To obtain these results, we used 18 computational nodes with four MPI processes per node and seven OpenMP threads per MPI process. The total computational times for the 5 and 10 Hz solutions are 32 and 108 min, respectively.

To understand how the number of iterations depends on the frequency, we carried out the previous experiment but over a range of frequencies from 2 to 20 Hz (Table 4). Here, we used the spatial lateral steps corresponding to 2.5 points per minimal wavelength and vertical — 10 points, so that the total number of grid points for each experiment is different. Our results suggest that when the frequency is low ($\leq$10 Hz), doubling of the frequency leads to the increase of the number of iterations by a factor of three. For higher frequencies, this coefficient increases, but up to 20 Hz, it is less than eight. It is worth mentioning that for models with more complicated lateral structures, the convergence rate deteriorates. For instance, for the SEG/EAGE overthrust model (Figure 4) at 5 Hz, it converges for 94 iterations, whereas it converges at 10 Hz for 904 iterations.

CONCLUSION

We presented an iterative method for parallel FD elastic wavefield simulation in a 3D land scenario. Its key component, a unique preconditioner, ensures fast convergence of the iterative process in models with moderate lateral variations at low frequencies needed for macrovelocity reconstruction with FWI. Having a limitation of planar free-surface topography, it could be incorporated as a forward modeling engine in an inversion process with data after static corrections or redatuming has been applied. For further improvement of the parallel implementation and, as the result, of the total computational time, one might consider incorporation of a parallel FFT based on 2D domain decomposition, which could increase the upper limit of MPI scalability.

In the future, our method might be extended to handle an arbitrary curved air-rock free surface using a hybrid approach, in which the simulation in the upper part of the model including that surface could be performed by a finite-element method.

Simulation for the marine case that is not covered by our paper is another challenge. Due to the S-wave velocity being zero in water, our method in its current state could not be applied to a combined acoustic/elastic model. That issue might be overcome by an appropriate perturbation of an imaginary part of the μ Lamé parameter in water. Preliminary numerical experiments show convergence of the method in the case of a planar seabed model. The case of a curved sea bottom requires further investigation.

AUTHOR CONTRIBUTIONS

The method presented in the paper has been proposed and theoretically justified by V. Kostin and V. Tcheverda. The numerical algorithm has been developed by M. Belonosov and D. Neklyudov. The software development and verification of the method have been carried out by M. Belonosov.

ACKNOWLEDGMENTS

We are grateful to M. Jervis for reviewing our manuscript. Special thanks go to A. Bakulin, V. Lisitsa, and M. Dmitriev for fruitful discussions and advice on this topic. Two of the authors (V. Kostin and V. Tcheverda) have been sponsored by the Russian Science Foundation grant 17-17-01128.

DATA AND MATERIALS AVAILABILITY

Data associated with this research are confidential and cannot be released.

APPENDIX A

EXISTENCE OF THE SOLUTION TO THE BOUNDARY VALUE PROBLEM 17

In the general case, the boundary value problem for a system of ODEs subject to some boundary conditions may not have a solution. That happens when there is a nonzero solution to the problem with a zero right side. Let us demonstrate that for the system of equations 17 this is not the case.

We restrict ourselves to the case in which $\gamma(z) = 1$, this is, equivalent to excluding the effect of PML on the bottom boundary. Let us introduce three relevant functional spaces:

- $L_2([0, Z])$: a Hilbert space of square-integrable 9C vector-functions $\mathbf{u} = (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9)^T$ with the inner product defined as

$$(\mathbf{u},\mathbf{v})_{L_2} = \int_0^Z (\mathbf{u}(z),\mathbf{v}(z))dz,$$
$$\text{for } \mathbf{u}(z) \text{ and } \mathbf{v}(z) \text{ from } L_2([0,Z]), \qquad \text{(A-1)}$$

  with the integrand being the inner product in vector space C^9 of vectors of nine complex valued components

$$(\mathbf{u}(z), \mathbf{v}(z)) = \sum_{j=1}^{9} u_j(z)\overline{v_j(z)}; \qquad \text{(A-2)}$$

- $H^1([0, Z])$: a subspace in $L_2([0, Z])$ of functions whose first derivatives belong to $L_2([0, Z])$;
- $\mathbb{H}$: a linear subspace in $H^1([0, Z])$ defined by boundary conditions

$$\begin{cases} \mathbf{B_0}\mathbf{u}(0) = 0 \\ \mathbf{B_Z}\mathbf{u}(Z) = 0 \end{cases} \qquad \text{(A-3)}$$

that is the domain of differential operator $\mathcal{L}_1$ defined by expression

$$\mathcal{L}_1\mathbf{u} = \left[ik_x\mathbf{P} + ik_y\mathbf{Q} + \mathbf{R}\frac{d}{dz}\right]\mathbf{u}, \tag{A-4}$$

where matrices $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{B}_0$, and $\mathbf{B}_\mathbf{Z}$ are defined in formulas 4–6.

Operator $\mathcal{L}_1$ maps its domain $\mathbb{H}$ into space $L_2([0,Z])$. Given the right side vector-function $\hat{\mathbf{w}}(k_x,k_y;z) \in L_2([0,Z])$, boundary value problem 17 can be represented as an operator equation:

$$\left[i\omega\begin{pmatrix}\rho_0(z)\mathbf{I}_{3\times3} & 0\\ 0 & (1+i\beta)\mathbf{S}_0(z)\end{pmatrix} - \mathcal{L}_1\right]\hat{\mathbf{u}}(k_x,k_y;z) = \hat{\mathbf{w}}(k_x,k_y;z) \tag{A-5}$$

with the respective homogeneous problem:

$$\left[i\omega\begin{pmatrix}\rho_0(z)\mathbf{I}_{3\times3} & 0\\ 0 & (1+i\beta)\mathbf{S}_0(z)\end{pmatrix} - \mathcal{L}_1\right]\mathbf{u} = 0 \tag{A-6}$$

and, consequenly, our task is to show that problem A-6 has only trivial solutions. That is equivalent to prove that the eigenvalue problem

$$\mathcal{L}_1\mathbf{u} = \lambda\begin{pmatrix}\rho_0\mathbf{I}_{3\times3} & 0\\ 0 & (1+i\beta)\mathbf{S}_0\end{pmatrix}\mathbf{u} \tag{A-7}$$

cannot have purely imaginary eigenvalues λ.

One can easily verify that operator $\mathcal{L}_1$ is skew-Hermitian; i.e., for any pair of vector-functions $\mathbf{u}$ and $\mathbf{v}$ from space $\mathbb{H}$ the following equality holds:

$$(\mathcal{L}_1\mathbf{u},\mathbf{v})_{L_2} = -(\mathbf{u},\mathcal{L}_1\mathbf{v})_{L_2}. \tag{A-8}$$

Hence, the quadratic form $(\mathcal{L}_1\mathbf{u},\mathbf{u})_{L_2}$ may have only purely imaginary values. For solution $\mathbf{u}$ to problem A-7, the respective λ can be expressed as the ratio

$$\lambda = (\mathcal{L}_1\mathbf{u},\mathbf{u})_{L_2} \Big/ \left(\begin{pmatrix}\rho_0\mathbf{I}_{3\times3} & 0\\ 0 & (1+i\beta)\mathbf{S}_0\end{pmatrix}\mathbf{u},\mathbf{u}\right)_{L_2}. \tag{A-9}$$

The numerator of the ratio in equality A-9 is purely imaginary; hence, there are only two possibilities for λ to be purely imaginary, i.e., $\lambda = i\omega$:

- $\lambda = 0$, which means $\omega = 0$. We do not consider this case because the angular frequency $\omega \neq 0$.
- $\left(\begin{pmatrix}0 & 0\\ 0 & \mathbf{S}_0\end{pmatrix}\mathbf{u},\mathbf{u}\right)_{L_2} = 0$. This implies that $u_4 = u_5 = u_6 = u_7 = u_8 = u_9 = 0$ due to $\mathbf{S}_0$ is a symmetric positive definite matrix. Using this, from equation A-3 one gets that $\lambda\rho_0(u_1,u_2,u_3)^T = 0$. Having $\lambda \neq 0$, we conclude that $(u_1,u_2,u_3)^T = 0$. Hence, $\mathbf{u}$ is zero.

Thus, we have just proved that in the case of $\gamma(z) = 1$ the boundary value problem 17 is resolvable. In the case of the PML in place, when $\gamma(z)$ is not a constant, we do not have a theoretical proof, but in our modeling tests, we did not encounter any issues.

APPENDIX B

NUMERICAL SOLUTION TO THE BOUNDARY VALUE PROBLEM 17

For unknown vector-function $\hat{\mathbf{u}}(k_x,k_y;z) = (\hat{u}_1,\hat{u}_2,\hat{u}_3,\hat{u}_4,\hat{u}_5,\hat{u}_6,\hat{u}_7,\hat{u}_8,\hat{u}_9)^T$ and right side vector-function $\hat{\mathbf{w}}(k_x,k_y;z) = (\hat{w}_1,\hat{w}_2,\hat{w}_3,\hat{w}_4,\hat{w}_5,\hat{w}_6,\hat{w}_7,\hat{w}_8,\hat{w}_9)^T$, the system in formula 17 can be reduced to the following system of six ODEs:

$$\frac{\gamma(z)}{i\omega}\frac{d}{dz}\begin{bmatrix}\hat{u}_1\\ \hat{u}_2\\ \hat{u}_6\\ \hat{u}_3\\ \hat{u}_7\\ \hat{u}_8\end{bmatrix} + \begin{pmatrix}0 & 0 & 0 & \frac{k_x}{\omega} & 0 & -c_0\\ 0 & 0 & 0 & \frac{k_y}{\omega} & -c_0 & 0\\ 0 & 0 & 0 & -\rho_0 & \frac{k_y}{\omega} & \frac{k_x}{\omega}\\ d_{11} & d_{12} & d_{13} & 0 & 0 & 0\\ d_{21} & d_{22} & d_{23} & 0 & 0 & 0\\ d_{31} & d_{32} & d_{33} & 0 & 0 & 0\end{pmatrix} \times \begin{bmatrix}\hat{u}_1\\ \hat{u}_2\\ \hat{u}_6\\ \hat{u}_3\\ \hat{u}_7\\ \hat{u}_8\end{bmatrix} = -\frac{1}{i\omega}\begin{bmatrix}\hat{w}_8\\ \hat{w}_7\\ \hat{w}_3\\ \bar{\bar{w}}_1\\ \bar{\bar{w}}_2\\ \bar{\bar{w}}_3\end{bmatrix}, \tag{B-1}$$

for six components $\hat{u}_1,\hat{u}_2,\hat{u}_3,\hat{u}_6,\hat{u}_7,\hat{u}_8$ subject to boundary conditions

$$\begin{aligned}\hat{u}_6(0) = \hat{u}_7(0) = \hat{u}_8(0) = 0,\\ \hat{u}_3(Z) = \hat{u}_7(Z) = \hat{u}_8(Z) = 0.\end{aligned} \tag{B-2}$$

Three remaining components are evaluated by explicit formulas

$$\begin{aligned}\hat{u}_4 &= \frac{k_x a_0\hat{u}_1 + k_y b_0\hat{u}_2}{\omega(1+i\beta)(a_0^2-b_0^2)} + \frac{b_0\hat{u}_6}{a_0-b_0} + \frac{a_0\hat{w}_4 + b_0\hat{w}_5}{i\omega(1+i\beta)(a_0^2-b_0^2)},\\ \hat{u}_5 &= \frac{k_x b_0\hat{u}_1 + k_y a_0\hat{u}_2}{\omega(1+i\beta)(a_0^2-b_0^2)} + \frac{b_0\hat{u}_6}{a_0-b_0} + \frac{b_0\hat{w}_4 + a_0\hat{w}_5}{i\omega(1+i\beta)(a_0^2-b_0^2)},\\ \hat{u}_9 &= \frac{1}{\omega(1+i\beta)c_0}[k_y\hat{u}_1 + k_x\hat{u}_2 - \omega\hat{w}_9].\end{aligned} \tag{B-3}$$

In formulas B-2 and B-3, we use the following notations:

$$
\begin{aligned}
&d_{11}=\frac{k_x b_0}{\omega(a_0-b_0)},\ d_{12}=\frac{k_y b_0}{\omega(a_0-b_0)},\\
&d_{13}=\frac{2(1+i\beta)b_0^2}{a_0-b_0}-(1+i\beta)a_0,\ d_{21}=\frac{k_x k_y}{\omega^2(1+i\beta)}\left[\frac{b_0}{a_0^2-b_0^2}+\frac{1}{c_0}\right],\\
&d_{22}=\frac{k_x^2}{\omega^2(1+i\beta)c_0}+\frac{k_y^2 a_0}{\omega^2(1+i\beta)(a_0^2-b_0^2)}-\rho_0,\ d_{23}=\frac{k_y b_0}{\omega(a_0-b_0)},\\
&d_{31}=\frac{k_x^2 a_0}{\omega^2(1+i\beta)(a_0^2-b_0^2)}+\frac{k_y^2}{\omega^2(1+i\beta)c_0}-\rho_0,\\
&d_{32}=\frac{k_x k_y}{\omega^2(1+i\beta)}\left[\frac{b_0}{a_0^2-b_0^2}+\frac{1}{c_0}\right],\\
&d_{33}=\frac{k_x b_0}{\omega(a_0-b_0)},\ \bar{\hat{w}}_1=\hat{w}_6+\frac{b_0(\hat{w}_4+\hat{w}_5)}{(a_0-b_0)},\\
&\bar{\hat{w}}_2=\hat{w}_2+\frac{k_x}{\omega(1+i\beta)c_0}\hat{w}_9+\frac{k_y(b_0\hat{w}_4+a_0\hat{w}_5)}{\omega(1+i\beta)(a_0^2-b_0^2)},\\
&\bar{\hat{w}}_3=\hat{w}_1+\frac{k_y}{\omega(1+i\beta)c_0}\hat{w}_9+\frac{k_x(a_0\hat{w}_4+b_0\hat{w}_5)}{\omega(1+i\beta)(a_0^2-b_0^2)}.
\end{aligned}
\tag{B-4}
$$

Let us emphasize that these manipulations do not affect the conclusion made in Appendix A regarding solvability of the boundary value problem 17; i.e., the boundary value problem B-1 subject to B-2 is solvable.

To numerically solve boundary value problem B-1, given integer N — the number of grid points in the z-direction, interval $[0, Z]$ is covered by a uniform grid of step $h_z = Z/(N-1)$ composed of N nodes with integer indices $z_i = i \cdot h_z (i = 0, \ldots, N-1)$, so that $z_0 = 0$ and $z_{N-1} = Z$, and $N-1$ nodes with half-integer indices $z_{i+\frac{1}{2}} = (i+1/2) \cdot h_z$ $(i = 0,1, \ldots, N-2)$. For the sake of brevity, the grid points with integer indices we call integer grid points and the points with half-integer indices we call half-integer grid points.

To approximate a certain function $v(z)$ at integer grid point z_i or half-integer grid point $z_{i+\frac{1}{2}}$, we take its value at this point and denote by $(v)_{z_i}$ or $(v)_{z_{i+\frac{1}{2}}}$, respectively. Its first derivative is approximated by standard staggered-grid (Richtmyer and Morton, 1957) finite-difference approximation with a stencil depicted in the top image of Figure B-1:

$$
\begin{aligned}
\left.\frac{dv}{dz}\right|_{z_i} &= \frac{9}{8}\frac{(v)_{z_{i+\frac{1}{2}}}-(v)_{z_{i-\frac{1}{2}}}}{h_z}-\frac{(v)_{z_{i+\frac{3}{2}}}-(v)_{z_{i-\frac{3}{2}}}}{24h_z}+O(h_z^4),\\
\left.\frac{dv}{dz}\right|_{z_{i+\frac{1}{2}}} &= \frac{9}{8}\frac{(v)_{z_i}-(v)_{z_{i+1}}}{h_z}-\frac{(v)_{z_{i+2}}-(v)_{z_{i-1}}}{24h_z}+O(h_z^4).
\end{aligned}
\tag{B-5}
$$

Applying these rules to approximate the first three ODEs of system B-1 at integer grid points z_i for $2 \le i \le N-3$ and to its other three equations at half-integer grid point $z_{i+\frac{1}{2}}$ for $1 \le i \le N-3$, we arrive at the following linear algebraic equations:

$$
\begin{cases}
\frac{\gamma(z_i)}{i\omega}\left[\frac{9}{8}\frac{(\hat{u}_1)_{z_{i+\frac{1}{2}}}-(\hat{u}_1)_{z_{i-\frac{1}{2}}}}{h_z}-\frac{(\hat{u}_1)_{z_{i+\frac{3}{2}}}-(\hat{u}_1)_{z_{i-\frac{3}{2}}}}{24h_z}\right]+\frac{k_x}{\omega}(\hat{u}_3)_{z_i}\\
-c_0(z_i)(\hat{u}_8)_{z_i}=-\frac{\hat{w}_8(z_i)}{i\omega},\\
\frac{\gamma(z_i)}{i\omega}\left[\frac{9}{8}\frac{(\hat{u}_2)_{z_{i+\frac{1}{2}}}-(\hat{u}_2)_{z_{i-\frac{1}{2}}}}{h_z}-\frac{(\hat{u}_2)_{z_{i+\frac{3}{2}}}-(\hat{u}_2)_{z_{i-\frac{3}{2}}}}{24h_z}\right]+\frac{k_y}{\omega}(\hat{u}_3)_{z_i}\\
-c_0(z_i)(\hat{u}_7)_{z_i}=-\frac{\hat{w}_7(z_i)}{i\omega},\\
\frac{\gamma(z_i)}{i\omega}\left[\frac{9}{8}\frac{(\hat{u}_6)_{z_{i+\frac{1}{2}}}-(\hat{u}_6)_{z_{i-\frac{1}{2}}}}{h_z}-\frac{(\hat{u}_6)_{z_{i+\frac{3}{2}}}-(\hat{u}_6)_{z_{i-\frac{3}{2}}}}{24h_z}\right]-\rho_0(z_i)(\hat{u}_3)_{z_i}+\frac{k_y}{\omega}(\hat{u}_7)_{z_i}\\
+\frac{k_x}{\omega}(\hat{u}_8)_{z_i}=-\frac{\hat{w}_3(z_i)}{i\omega},\\
\frac{\gamma(z_{i+\frac{1}{2}})}{i\omega}\left[\frac{9}{8}\frac{(\hat{u}_3)_{z_{i+1}}-(\hat{u}_3)_{z_i}}{h_z}-\frac{(\hat{u}_3)_{z_{i+2}}-(\hat{u}_3)_{z_{i-1}}}{24h_z}\right]+d_{11}(z_{i+\frac{1}{2}})(\hat{u}_1)_{z_{i+\frac{1}{2}}}\\
+d_{12}(z_{i+\frac{1}{2}})(\hat{u}_2)_{z_{i+\frac{1}{2}}}+d_{13}(z_{i+\frac{1}{2}})(\hat{u}_6)_{z_{i+\frac{1}{2}}}=-\frac{\bar{\hat{w}}_1(z_{i+\frac{1}{2}})}{i\omega},\\
\frac{\gamma(z_{i+\frac{1}{2}})}{i\omega}\left[\frac{9}{8}\frac{(\hat{u}_7)_{z_{i+1}}-(\hat{u}_7)_{z_i}}{h_z}-\frac{(\hat{u}_7)_{z_{i+2}}-(\hat{u}_7)_{z_{i-1}}}{24h_z}\right]+d_{21}(z_{i+\frac{1}{2}})(\hat{u}_1)_{z_{i+\frac{1}{2}}}\\
+d_{22}(z_{i+\frac{1}{2}})(\hat{u}_2)_{z_{i+\frac{1}{2}}}+d_{23}(z_{i+\frac{1}{2}})(\hat{u}_6)_{z_{i+\frac{1}{2}}}=-\frac{\bar{\hat{w}}_2(z_{i+\frac{1}{2}})}{i\omega},\\
\frac{\gamma(z_{i+\frac{1}{2}})}{i\omega}\left[\frac{9}{8}\frac{(\hat{u}_8)_{z_{i+1}}-(\hat{u}_8)_{z_i}}{h_z}-\frac{(\hat{u}_8)_{z_{i+2}}-(\hat{u}_8)_{z_{i-1}}}{24h_z}\right]+d_{31}(z_{i+\frac{1}{2}})(\hat{u}_1)_{z_{i+\frac{1}{2}}}\\
+d_{32}(z_{i+\frac{1}{2}})(\hat{u}_2)_{z_{i+\frac{1}{2}}}+d_{33}(z_{i+\frac{1}{2}})(\hat{u}_6)_{z_{i+\frac{1}{2}}}=-\frac{\bar{\hat{w}}_3(z_{i+\frac{1}{2}})}{i\omega}.
\end{cases}
\tag{B-6}
$$

To approximate the first derivative of a certain function $v(z)$ at integer points z_i $(i = 1, N-2)$ or at half-integer grid points $z_{i+\frac{1}{2}}$ $(i = 0, N-2)$, we use another staggered-grid scheme with a stencil presented in the middle image of Figure B-1:

$$
\begin{aligned}
\left.\frac{dv}{dz}\right|_{z_i} &= \frac{(v)_{z_{i+\frac{1}{2}}}-(v)_{z_{i-\frac{1}{2}}}}{h_z}+O(h_z^2),\\
\left.\frac{dv}{dz}\right|_{z_{i+\frac{1}{2}}} &= \frac{(v)_{z_{i+1}}-(v)_{z_i}}{h_z}+O(h_z^2).
\end{aligned}
\tag{B-7}
$$

Using approximations B-6, the first three ODEs of system B-1 at grid points z_1, z_{N-2}, and its other three equations at points $z_{\frac{1}{2}}$, $z_{N-\frac{3}{2}}$ are approximated by equations

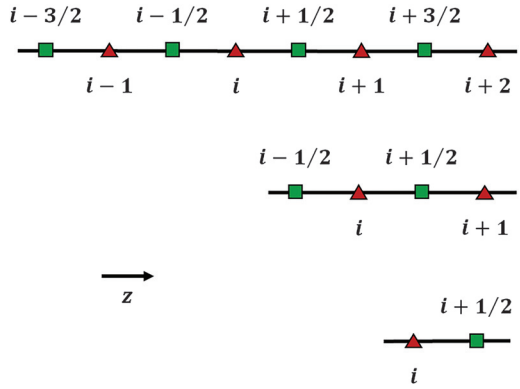


Figure B-1. Stencils used for a finite-difference approximation: the top — in formula B-4, the middle — in formula B-6, and the bottom — in formula B-8.

$$
\begin{cases}
\frac{\gamma(z_i)}{i\omega}\frac{(\hat{u}_1)_{z_{i+\frac{1}{2}}}-(\hat{u}_1)_{z_{i-\frac{1}{2}}}}{h_z}+\frac{k_x}{\omega}(\hat{u}_3)_{z_i}-c_0(z_i)(\hat{u}_8)_{z_i}\\
=-\frac{\hat{w}_8(z_i)}{i\omega},\\
\frac{\gamma(z_i)}{i\omega}\frac{(\hat{u}_2)_{z_{i+\frac{1}{2}}}-(\hat{u}_2)_{z_{i-\frac{1}{2}}}}{h_z}+\frac{k_y}{\omega}(\hat{u}_3)_{z_i}-c_0(z_i)(\hat{u}_7)_{z_i}\\
=-\frac{\hat{w}_7(z_i)}{i\omega},\\
\frac{\gamma(z_i)}{i\omega}\frac{(\hat{u}_6)_{z_{i+\frac{1}{2}}}-(\hat{u}_6)_{z_{i-\frac{1}{2}}}}{h_z}-\rho_0(z_i)(\hat{u}_3)_{z_i}+\frac{k_y}{\omega}(\hat{u}_7)_{z_i}\\
+\frac{k_x}{\omega}(\hat{u}_8)_{z_i}=-\frac{\hat{w}_3(z_i)}{i\omega},\\
\frac{\gamma(z_{i+\frac{1}{2}})}{i\omega}\frac{(\hat{u}_3)_{z_{i+1}}-(\hat{u}_3)_{z_i}}{h_z}+d_{11}(z_{i+\frac{1}{2}})(\hat{u}_1)_{z_{i+\frac{1}{2}}}+d_{12}(z_{i+\frac{1}{2}})(\hat{u}_2)_{z_{i+\frac{1}{2}}}\\
+d_{13}(z_{i+\frac{1}{2}})(\hat{u}_6)_{z_{i+\frac{1}{2}}}=-\frac{\bar{\hat{w}}_1(z_{i+\frac{1}{2}})}{i\omega},\\
\frac{\gamma(z_{i+\frac{1}{2}})}{i\omega}\frac{(\hat{u}_7)_{z_{i+1}}-(\hat{u}_7)_{z_i}}{h_z}+d_{21}(z_{i+\frac{1}{2}})(\hat{u}_1)_{z_{i+\frac{1}{2}}}+d_{22}(z_{i+\frac{1}{2}})(\hat{u}_2)_{z_{i+\frac{1}{2}}}\\
+d_{23}(z_{i+\frac{1}{2}})(\hat{u}_6)_{z_{i+\frac{1}{2}}}=-\frac{\bar{\hat{w}}_2(z_{i+\frac{1}{2}})}{i\omega},\\
\frac{\gamma(z_{i+\frac{1}{2}})}{i\omega}\frac{(\hat{u}_8)_{z_{i+1}}-(\hat{u}_8)_{z_i}}{h_z}+d_{31}(z_{i+\frac{1}{2}})(\hat{u}_1)_{z_{i+\frac{1}{2}}}+d_{32}(z_{i+\frac{1}{2}})(\hat{u}_2)_{z_{i+\frac{1}{2}}}\\
+d_{33}(z_{i+\frac{1}{2}})(\hat{u}_6)_{z_{i+\frac{1}{2}}}=-\frac{\bar{\hat{w}}_3(z_{i+\frac{1}{2}})}{i\omega}.
\end{cases}
\tag{B-8}
$$

Let us approximate the first derivative of a certain function $v(z)$ at point z_i by the following formula with a stencil presented in the bottom image of Figure B-1:

$$
\left.\frac{dv}{dz}\right|_{z_i}=\frac{(v)_{z_{i+\frac{1}{2}}}-(v)_{z_i}}{h_z/2}+O(h_z). \tag{B-9}
$$

Approximation B-8 is used to replace $d\hat{u}_6/dz$ at point z_0 in the third equation of system B-1 to get

$$
(\hat{u}_3)_{z_0}=\frac{1}{i\omega\rho_0(z_0)}\left[2\gamma(z_0)\frac{(\hat{u}_6)_{z_{\frac{1}{2}}}}{h_z}+\hat{w}_3(z_0)\right]. \tag{B-10}
$$

Together, formulas B-5, B-7, B-9, and boundary conditions B-2 build the finite-difference approximation of boundary value problem B-1. This results in a SLAE with a square, banded matrix of size $(6N-9)\times(6N-9)$. Due to approximation of the boundary conditions with the first and second order, this scheme is not of fourth order. It can be improved to the fourth order by an appropriate approximation of the boundary conditions, but we skip this step in this paper for the sake of brevity.

It is worth mentioning that if for initial equation 1 we assumed the top boundary of the computational domain to be absorbing (for example by including a PML), then formula B-9 would be replaced by $(\hat{u}_3)_{z_0}=0$. The rest of the scheme would be the same.

REFERENCES

Aki, K., and P. G. Richards, 1980, Quantitative seismology, theory and methods: W. H. Freeman and Co 1.

Amdahl, G. M., 1967, Validity of the single processor approach to achieving large-scale computing capabilities: AFIPS Conference Proceedings, 483–485.

Aminzadeh, F., J. Brac, and T. Kuntz, 1997, 3-D salt and overthrust models: SEG.

Bakulin, A., M. Dmitriev, P. Golikov, and D. Neklyudov, 2015, Enhancing 3D broadband land seismic data with smart super groups for processing and FWI: 77th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201413435.

Belonosov, M., M. Dmitriev, V. Kostin, D. Neklyudov, and V. Tcheverda, 2017, An iterative solver for the 3D Helmholtz equation: Journal of Computational Physics, **345**, 330–344, doi: 10.1016/j.jcp.2017.05.026.

Berenger, J. P., 1996, Three-dimensional perfectly matched layer for the absorption of electromagnetic waves: Journal of Computational Physics, **127**, 363–379, doi: 10.1006/jcph.1996.0181.

Colella, P., J. Bell, N. Keen, T. Ligocki, M. Lijewski, and B. van Straalen, 2007, Performance and scaling of locally-structured grid methods for partial differential equations: Journal of Physics: Conference Series, **78**, doi: 10.1088/1742-6596/78/1/012013/meta.

Collino, F., and C. Tsogka, 1998, Application of the PML absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media: Technical report RR-3471, INRIA.

Darbas, M., and F. Louer, 2013, Analytic preconditioners for the iterative solution of elastic scattering problems: HAL, hal-00839653, 1–32.

Dmitruk, P., L. P. Wang, W. H. Matthaeus, R. Zhang, and D. Seckel, 2001, Scalable parallel FFT for spectral simulations on a Beowulf cluster: Parallel Computing, **27**, 1921–1936, doi: 10.1016/S0167-8191(01)00120-X.

Erlangga, Y. A., and R. Nabben, 2008a, On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian: Electronic Transactions on Numerical Analysis, **31**, 403–424.

Erlangga, Y. A., and R. Nabben, 2008b, Deflation and balancing preconditioners for Krylov subspace methods applied to nonsymmetric matrices: SIAM Journal on Matrix Analysis and Applications, **30**, 684–699, doi: 10.1137/060678257.

Etienne, V., T. Tonellot, P. Thierry, V. Berthoumieux, and C. Andreolli, 2014, Optimization of the seismic modeling with the time-domain finite-difference method: 84th Annual International Meeting, SEG, Expanded Abstracts, 3536–3540, doi: 10.1190/segam2014-0176.1.

Intel, 2017, Intel®Math Kernel Library (Intel®MKL), https://software.intel.com/en-us/intel-mkl, accessed 01 May 2017.

Kostin, V., V. Lisitsa, G. Reshetova, and V. Tcheverda, 2015, Local time-space mesh refinement for simulation of elastic wave propagation in multi-scale media: Journal of Computational Physics, **281**, 669–689, doi: 10.1016/j.jcp.2014.10.047.

Kostin, V., S. Solovev, H. Liu, and A. Bakulin, 2017, HSS cluster-based direct solver for acoustic wave equation: 87th Annual International Meeting, SEG, Expanded Abstracts, 4017–4021, doi: 10.1190/segam2017-17443086.1.

Li, Y., L. Métivier, R Brossier, B. Han, and J. Virieux, 2015, 2D and 3D frequency-domain elastic wave modeling in complex media with a parallel iterative solver: Geophysics, **80**, no. 3, T101–T118, doi: 10.1190/geo2014-0480.1.

Lisitsa, V., V. Tcheverda, and C. Botter, 2016, Combination of discontinuous Galerkin method with finite differences for simulation of elastic wave: Journal of Computational Physics, **311**, 142–157, doi: 10.1016/j.jcp.2016.02.005.

Neklyudov, D., V. Tcheverda, and I. Silvestrov, 2011, Frequency domain iterative solver for elasticity with semi-analytical preconditioner: 81st Annual International Meeting, SEG, Expanded Abstracts, 2931–2935.

Oosterlee, C. W., C. Vuik, W. A. Mulder, and R.-E. Plessix, 2010, Shifted-Laplacian preconditioners for heterogeneous Helmholtz problems, *in* B. Koren and K. Vuik, eds., Advanced Computational Methods in Science and Engineering: Springer, 21–46.

Operto, S., J. Virieux, P. Amestoy, J. L'Excellent, L. Giraud, and H. Hadj, 2007, 3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study: Geophysics, **72**, no. 5, SM195–SM211, doi: 10.1190/1.2759835.

Pratt, R. G., 1999, Seismic waveform inversion in the frequency domain. Part I: Theory and verification in a physical scale model: Geophysics, **64**, 888–901, doi: 10.1190/1.1444597.

Richtmyer, R., and K. Morton, 1957, Difference methods for initial value problems, 2nd ed.: Interscience Publishers.

Rizzuti, G., and W. A. Mulder, 2015, A multigrid-based iterative solver for the frequency-domain elastic wave equation: 77th Annual International Conference and Exhibition, EAGE, Extended Abstracts, doi: 10.3997/2214-4609.201413295.

Saad, Y., 2003, Iterative methods for sparse linear systems, 2nd ed.: SIAM.

Sirgue, L., J. Etgen, U. Albertin, and S. Brandsberg-Dahl, 2007, System and method for 3D frequency domain waveform inversion based on 3D time-domain forward modeling: U.S. Patent, 11/756, 384.

Symes, W. W., 2008, Migration velocity analysis and waveform inversion: Geophysical Prospecting, **56**, 765–790, doi: 10.1111/j.1365-2478.2008.00698.x.

Van der Vorst, H. A., 1992, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems: SIAM Journal on Scientific Computing, **13**, 631–644.

Virieux, J., S. Operto, H. Ben-Hadj-Ali, R. Brosssier, V. Etienne, F. Sourber, L. Giraud, and A. Haidar, 2009, Seismic wave modeling for seismic imaging: The Leading Edge, **28**, 538–544, doi: 10.1190/1.3124928.

Wang, S., M. V. de Hoop, J. Xia, and X. S. Li, 2012, Massively parallel structured multifrontal solver for time-harmonic elastic waves in 3-D anisotropic media: Geophysical Journal International, **191**, 346–366, doi: 10.1111/j.1365-246X.2012.05634.x.

Weisbecker, C., P. Amestoy, O. Boiteau, R. Brossier, A. Buttari, J.-Y. L'Excellent, S. Operto, and J. Virieux, 2013, 3D frequency-domain seismic modeling with a block low-rank algebraic multifrontal direct solver: 83rd Annual International Meeting, SEG, Expanded Abstracts, 3411–3416, doi: 10.1190/segam2013-0603.1.

Journal of Computational Physics 345 (2017) 330–344

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

An iterative solver for the 3D Helmholtz equation

Mikhail Belonosov [a], Maxim Dmitriev [b], Victor Kostin [c,*], Dmitry Neklyudov [c], Vladimir Tcheverda [c,d]

[a] EXPEC ARC GRC Delft, Aramco Overseas Company B.V., Informaticalaan 6, 2628 ZD, Delft, The Netherlands
[b] EXPEC Advanced Research Center, Saudi Aramco, Building 137, Dhahran 31311, Saudi Arabia
[c] Trofimuk Institute of Petroleum Geology and Geophysics SB RAS, 3, Koptyug ave, 630090, Novosibirsk, Russia
[d] Kazakh–British Technical University, 59, Tole bi st, 480091, Almaty, Kazakhstan

ARTICLE INFO

Article history:
Received 12 December 2016
Received in revised form 11 May 2017
Accepted 13 May 2017
Available online 17 May 2017

Keywords:
Acoustics
Modeling
Helmholtz equation
Iterative methods
Preconditioning

ABSTRACT

We develop a frequency–domain iterative solver for numerical simulation of acoustic waves in 3D heterogeneous media. It is based on the application of a unique preconditioner to the Helmholtz equation that ensures convergence for Krylov subspace iteration methods. Effective inversion of the preconditioner involves the Fast Fourier Transform (FFT) and numerical solution of a series of boundary value problems for ordinary differential equations. Matrix-by-vector multiplication for iterative inversion of the preconditioned matrix involves inversion of the preconditioner and pointwise multiplication of grid functions. Our solver has been verified by benchmarking against exact solutions and a time-domain solver.

1. Introduction

Numerical simulation of acoustic wavefields is an important part of many algorithms developed to solve problems arising in exploration geophysics. Usually, simulation needs to be performed many times to reach the final result and consumes the main part of the computational time. In particular, it serves as an engine for acoustic frequency–domain full waveform inversion (FD FWI) ([1–3] and references cited therein). For macro velocity reconstruction such simulation is performed several times for a number of low frequencies (up to 10 Hz) at each iteration of this process. Therefore, the efficiency of the whole process strongly depends on how fast and accurately the simulation is performed.

In the frequency domain, the acoustic wave propagation in R^3 is governed by the Helmholtz equation supplied with some condition at infinity [4]. The numerical complexity of the problem rises due to the sheer size of the system of linear equations to solve and the unfavorable spectral properties of the coefficient matrix of this system. Time-domain schemes are commonly used [5,6]. Frequency–domain iterative solvers with efficient preconditioners present advantages when one deals with FD FWI and uses only a few frequencies per inversion [7,8]. Their memory and computational complexities are $O(N^3)$, where N is the number of grid points in one direction.

A common approach to solve the Helmholtz equation is using Krylov-type iterative techniques for the system of linear algebraic equations that arises in finite-difference or finite-element approximations of the respective boundary value

* Corresponding author.
E-mail addresses: Mikhail.Belonosov@aramcooverseas.com (M. Belonosov), maxim.dmitriev@aramco.com (M. Dmitriev), KostinVI@ipgg.sbras.ru (V. Kostin), dmitn@mail.ru (D. Neklyudov), CheverdaVA@ipgg.sbras.ru (V. Tcheverda).

http://dx.doi.org/10.1016/j.jcp.2017.05.026
0021-9991/

problems (see, for example, [2,9,10]). The indefiniteness of the coefficient matrix leads to its poor conditionality and slow convergence of the Krylov subspace iteration method [11].

This work follows the approach proposed in [12]. We construct a right preconditioner as a complex damped Helmholtz operator (shifted Laplace operator, [13–17]) with some 1D vertically heterogeneous background velocity. This operator can be inverted effectively. We represent the actual Helmholtz operator as a perturbation of the preconditioner. As a result, the operator of the preconditioned system becomes some perturbation of the unity operator. The norm of this perturbation is proportional to the magnitude of the velocity perturbation and frequency. For cases where the perturbation is small enough, the convergence of the iterative method is guaranteed. Similar approach with 1D vertically inhomogeneous background velocity involved was presented in [18] for the Maxwell equations in the 3D half-space.

Matrix-by-vector multiplication of the preconditioned system is decomposed to a series of 2D Fast Fourier Transforms (FFTs) in horizontal directions, solving a set of pairwise independent boundary value problems for ordinary differential equations in the vertical direction followed by a series of inverse 2D FFTs and pointwise multiplication by a scalar function. The numerical complexity of such an algorithm is close to optimal. Effective parallelization techniques can be applied at all steps of the algorithm. This enables us to perform simulation of an acoustic wavefield for realistic scenarios within a reasonable computational time.

This paper is structured as follows. In Section 2, the problem is considered using a set of models that vary only in the vertical direction. We introduce Perfectly Matched Layers (PMLs) to attenuate reflections from the boundary of a computational domain and set a respective boundary value problem. We derive a method to solve the problem that consists of expansion of the unknown function in the Fourier series, solving boundary value problems for ordinary differential equations for the coefficients of the Fourier series and summing up the Fourier series. In section 3, we discuss the details of numerical implementation of the algorithm and estimate the numerical complexity. In Section 4, we develop an algorithm to solve the problem in a 3D heterogeneous model that is the main goal of this paper. The Helmholtz equation is preconditioned by the Helmholtz operator in a vertically inhomogeneous medium, and the Krylov-type iteration method is applied. Some other ingredients to improve the convergence rate are introduced. In Section 5, the feasibility of the method is verified by several numerical experiments. We end with the conclusions in Section 6.

2. Solution to the Helmholtz equation in a 3D layered medium

In the algorithm we describe in this paper, to solve the Helmholtz equation in a 3D heterogeneous bounded medium, the main step is solving a boundary value problem in a layered bounded medium. Later on, we will describe how the layered medium arises naturally in the solution process. In this section we develop an algorithm to solve the problem in a layered medium and explain step by step how we shift from the statement of the problem in a 3D infinite layered medium to a bounded medium.

2.1. 3D vertically heterogeneous infinite layer

Consider the 3D Helmholtz equation in R^3 filled with an acoustic medium having sound velocity (or compressional wave propagation velocity) $c(z)$ that depends only on depth (z coordinate)

$$\Delta u(x, y, z) + \frac{\omega^2}{c^2(z)} u(x, y, z) = f(x, y, z). \tag{1}$$

Here $u(x, y, z)$ is the pressure at point (x, y, z), $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ is the Laplace operator, ω is the real angular frequency and $f(x, y, z)$ is the source function.

It is assumed that the unknown function satisfies the radiation condition at infinity (see [4]). Note that everywhere below, this condition is assumed implicitly when we consider equation (1) in an unbounded domain.

Suppose that function $c(z)$ is a constant function outside some interval, i.e.,

$$c(z) = c_0 \text{ for } |z| \geq Z_0 > 0,$$

and, consequently, the upgoing and the downgoing waves being described by equation (1) do not reflect for $|z| > Z_0$. If we are interested in the solution in a bounded outside interval $|z| \leq Z_0$ domain (our case), PMLs (see e.g. [19–21]) could be used to reduce reflections from the boundary.

To consider this case, let D be an acoustic layer

$$D = \{(x, y, z) : (x, y) \in R^2, -Z < z < Z\},$$

where $Z > Z_0$. This domain includes sublayers $\{(x, y, z) : (x, y) \in R^2, Z_0 < z < Z\}$ and $\{(x, y, z) : (x, y) \in R^2, -Z < z < -Z_0\}$, where the sound velocity is constant. Layer D is extended to

$$D_{PML} = \{(x, y, z) : (x, y) \in R^2, -Z_1 < z < Z_1\} \supset D,$$

where $Z_1 > Z$. Unknown function $u(x, y, z)$ is defined in D_{PML} and satisfies differential equation (1) in D, and in extra layers $Z < z < Z_1$ and $-Z_1 < z < -Z$ it satisfies modified equations. In layer $Z < z < Z_1$, the modified equation is

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial \tilde{z}^2}\right) u(x, y, z) + \frac{\omega^2}{c^2(z)} u(x, y, z) = f(x, y, z), \tag{2}$$

where source function $f(x, y, z)$ is extended into D_{PML} with zeros. In this expression, a new variable

$$\tilde{z}(z) = z - \frac{i}{\omega} \int_0^z d(s) ds$$

is introduced using some smooth damping function $d(s)$ [21], which is zero for $0 \le z \le Z$ and positive for $Z < s < Z_1$. So, $\tilde{z}$ coincides with z for $0 < z \le Z$ and becomes complex valued with a negative imaginary part for $Z < z < Z_1$. On boundary $z = Z_1$, the Dirichlet homogeneous condition $u = 0$ is imposed. The damping function assures attenuation of waves within layer $Z < z < Z_1$, in particular the reflection from boundary $z = Z_1$ due to the Dirichlet condition. We do not consider here the questions of how to choose the damping function and the width of the PML. These and other important practical questions are considered in the numerous papers on PMLs. Construction of the PML for sublayer $-Z_1 < z < -Z$ is similar.

Equation (2) can be used in the whole domain D_{PML} with a note that it coincides with equation (1) within D. Solutions to equation (2) are complex valued functions, though only real parts of these solutions have physical meaning.

Along with equation (1), in domain D, we consider the modified Helmholtz equation

$$\Delta u(x, y, z) + (1 - i\beta) \frac{\omega^2}{c^2(z)} u(x, y, z) = f(x, y, z) \tag{3}$$

with some constant parameter $\beta > 0$. This parameter ensures existence and uniqueness of the solution to equation (3), which tends to zero for $\sqrt{x^2 + y^2} \to \infty$. The operator of equation (3), called the shifted Laplace operator, has been proved to be a good preconditioner for the Helmholtz equation (see [14,15]).

The PML for equation (3) is constructed in a similar way as that for the original Helmholtz equation (1) by considering equation

$$\tilde{\Delta} u(x, y, z) + (1 - i\beta) \frac{\omega^2}{c^2(z)} u(x, y, z) = f(x, y, z) \tag{4}$$

in domain D_{PML} with $\tilde{\Delta} = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial \tilde{z}^2}\right)$ and homogeneous Dirichlet boundary conditions posed on outer boundaries.

So far, we have dealt with R^3, but similarly we could consider equation (1) in a half-space $\{(x, y, z) : (x, y) \in R^2, z > 0\}$ with the free-surface boundary condition $u = 0$ on plane $z = 0$. In this case, only one PML is needed and domains D_{PML} and D are

$$\begin{aligned} D_{PML} &= \{(x, y, z) : (x, y) \in R^2, 0 < z < Z_1\}, \\ D &= \{(x, y, z) : (x, y) \in R^2, 0 < z < Z\}. \end{aligned} \tag{5}$$

Below, we consider only this case but it is worth mentioning that all the results can be easily rewritten for the case of R^3.

Equation (4) is considered in D_{PML} with boundary conditions

$$\begin{aligned} u(x, y, 0) &= 0, \\ u(x, y, Z_1) &= 0. \end{aligned} \tag{6}$$

Applying a 2D Fourier transform over x and y in equation (4) results in a set of linear ordinary differential equations of second order

$$\frac{d^2}{d\tilde{z}^2} \hat{u}(k_x, k_y; z) + \left(k^2(z) - k_x^2 - k_y^2\right) \hat{u}(k_x, k_y; z) = \hat{f}(k_x, k_y; z), \tag{7}$$

where k_x, k_y are spatial frequencies (Fourier variables dual to x and y respectively) and $k^2(z) = (1 - i\beta)\omega^2 / c^2(z)$. The caret symbol $\hat{}$ above functions in equation (7) and below denotes respective Fourier images:

$$\begin{aligned} \hat{u}(k_x, k_y; z) &= \int_{R^2} u(x, y, z) e^{-i(k_x x + k_y y)} dx dy, \\ \hat{f}(k_x, k_y; z) &= \int_{R^2} f(x, y, z) e^{-i(k_x x + k_y y)} dx dy. \end{aligned} \tag{8}$$

Equation (7) can be written in terms of differentiation with respect to variable z

$$\left(\mu(z)\frac{d}{dz}\mu(z)\frac{d}{dz} + k^2(z) - k_x^2 - k_y^2\right)\hat{u}(k_x, k_y; z) = \hat{f}(k_x, k_y; z), \tag{9}$$

where

$$\mu(z) = \frac{i\omega}{d(z) + i\omega}.$$

Differential equations (9) (or equivalently, (7)) are fully independent of each other for different values of k_x, k_y. Boundary conditions (6), posed for solutions of equation (4), imply the following boundary conditions for solutions of equation (9)

$$\begin{aligned} \hat{u}(k_x, k_y; 0) &= 0, \\ \hat{u}(k_x, k_y; Z_1) &= 0. \end{aligned} \tag{10}$$

As a result, we get a set of independent boundary value problems on interval $[0, Z_1]$ for functions $\hat{u}(k_x, k_y; z)$.

The correctness (i.e. existence and uniqueness of solutions) of the above boundary value problems for equations (9) is guaranteed by parameter β being positive. The solutions can be represented in the form

$$\hat{u}(k_x, k_y; z) = \int_0^{Z_1} G(z, \zeta; k_x, k_y)\hat{f}(k_x, k_y, \zeta)d\zeta, \tag{11}$$

where $G(z, \zeta; k_x, k_y)$ are the respective Green's functions.

Having found $\hat{u}(k_x, k_y; z)$, the solution $u(x, y, z)$ of equation (4) is computed using the inverse Fourier transform

$$u(x, y, z) = \frac{1}{4\pi^2}\int_{R^2} \hat{u}(k_x, k_y; z)e^{i(k_x x + k_y y)}dk_x dk_y. \tag{12}$$

Let L_0 be a differential operator defined by the left-hand side of equation (4) for functions subject to boundary conditions posed on boundaries $z = 0$ and $z = Z_1$, as described above. The domain of operator L_0 is defined as a linear subspace in Sobolev functional space $H^1(D_{PML})$ (quadratically integrable over D_{PML} functions with quadratically integrable weak partial derivatives of first order) subject to its elements satisfying the boundary conditions.

The problem we solve can be written in the operator form as

$$L_0 u = \tilde{\Delta} u + (1 - i\beta)\frac{\omega^2}{c^2(z)}u = f, \tag{13}$$

where f is the right-hand side of the differential equation (4) and is supposed to belong to the functional subspace $H^{-1}(D_{PML})$. The solution to this problem is formally represented as

$$u = L_0^{-1} f.$$

Above, we described an algorithm to compute $L_0^{-1} f$. The algorithm comprises three steps:

1. Computing the forward Fourier transform of source function f using formula (8).
2. Solving boundary value problems (9) and (10) for the ordinary differential equations (see formula (11), which formally represents the solutions).
3. Computing the inverse Fourier transform (see (12)).

The simplicity of the algorithm for inverting L_0 makes this operator suitable to be used as a preconditioner. In the next section we modify the operator and the inversion algorithm to be applicable for bounded parallelepipedal domains. Note, that this modification does not complicate the algorithm. Keep in mind, that existence of a fast inversion method is a necessary requirement for an operator to be used as a preconditioner.

2.2. 3D layered bounded medium

In practical applications, for instance when we deal with seismic waves, bounded domains are considered. In this subsection, we modify the aforementioned algorithm to be able to work in a cuboid domain

$$D_{bound} = \{(x, y, z) : 0 \le x \le X, 0 \le y \le Y, 0 \le z \le Z_1\} \tag{14}$$

of size $X \times Y \times Z_1$. Equation (4) is considered in D_{bound} with PML included.

Formal replacement of integration over R^2 in (8) to integrals over rectangle $\Pi = [0, X] \times [0, Y]$ leads to

$$\begin{aligned} f_{k_x,k_y}(z) &= \frac{1}{XY}\int\limits_{\Pi} f(x,y,z)e^{-2\pi i\left(\frac{k_x x}{X}+\frac{k_y y}{Y}\right)}dxdy, \\ u_{k_x,k_y}(z) &= \frac{1}{XY}\int\limits_{\Pi} u(x,y,z)e^{-2\pi i\left(\frac{k_x x}{X}+\frac{k_y y}{Y}\right)}dxdy, \end{aligned} \tag{15}$$

where k_x, k_y are integer indices. These are Fourier series coefficients (with respect to x and y) for functions $u(x, y, z)$ and $f(x, y, z)$.

Application of transform (15) to equation (4) in D_{bound} results in ordinary differential equations for Fourier series coefficients $u_{k_x,k_y}(z)$

$$\left(\mu(z)\frac{d}{dz}\mu(z)\frac{d}{dz} + k^2(z) - k_x^2 - k_y^2\right) u_{k_x,k_y}(z) = f_{k_x,k_y}(z) \tag{16}$$

with boundary conditions

$$\begin{aligned} u_{k_x,k_y}(0) &= 0, \\ u_{k_x,k_y}(Z_1) &= 0. \end{aligned} \tag{17}$$

These boundary value problems differ from (9), (10) only by notations.

Having found solutions $u_{k_x,k_y}(z)$ to these boundary value problems, function $u(x, y, z)$ is restored using the summation formula

$$u(x,y,z) = \sum_{k_y=-\infty}^{\infty}\sum_{k_x=-\infty}^{\infty} u_{k_x,k_y}(z)e^{2\pi i\left(\frac{k_x x}{X}+\frac{k_y y}{Y}\right)}. \tag{18}$$

This formula replaces the inverse Fourier transform (12) in the third step above. Note that $u(x, y, z)$, obtained by formula (18), is a periodic function with respect to x and y with periods X and Y respectively.

Let us remember that these modifications are performed to develop an algorithm to invert operator L_0 that acts on functions defined in D_{bound}. Given source function $f \in H^{-1}(D_{bound})$ Function $u = L_0^{-1} f$ belongs to the functional space $H^1(D_{bound})$ and satisfies boundary conditions (6). It also satisfies periodic boundary conditions with respect to x and y:

$$\begin{aligned} u(0,y,z) &= u(X,y,z), \\ u(x,0,z) &= u(x,Y,z). \end{aligned} \tag{19}$$

In other words, application of operator L_0^{-1} to any function $f \in H^{-1}(D_{bound})$ results in a function, which is periodic with respect to x and y.

To sum up, the modified algorithm of inverting L_0 is as follows:

1. Computing Fourier series coefficients $f_{k_x,k_y}(z)$ of source function f using formula (15).
2. Solving boundary value problems (16) and (17).
3. Restoring the solution by summation formula (18).

3. Computational complexity to invert L_0

In the algorithm we describe, solving boundary value problems $L_0 v = g$ is to be performed many times for different right-hand sides $g(x, y, z)$ defined in D_{bound}. Assume this domain is covered by a uniform grid of $N_x \times N_y \times N_z$ points that we use to approximate the operator equation $L_0 v = g$. For the respective discrete problem, we use the same notation

$$L_0 v = g,$$

where g and v are grid functions for the right-hand side and the unknown function respectively (we hope that using the same notations for functions and their grid counterparts will not confuse the reader).

Computing Fourier series coefficients $g_{k_x,k_y}(z)$ of function $g(x, y, z)$ reduces to a numerical approximation of the integral over rectangle $\{0 \le x \le X, 0 \le y \le Y\}$ and can be carried out by use of the FFT algorithm [22]. On the equidistant grid of $N_x \times N_y$ points the FFT needs $O(N_x N_y \log(N_x N_y))$ floating point operations resulting in $N_x \times N_y$ values of the Fourier coefficients. The FFT should be applied to every (x, y)-plane. In total, the FFT applied to the right-hand side g requires $O(N_x N_y N_z \log(N_x N_y))$ floating point operations and results in $N_x N_y N_z$ Fourier coefficients.

The second step of the algorithm is solving boundary value problems (16) and (17) on interval $[0, Z_1]$. Each differential equation should be approximated on the grid of N_z points. Along with the boundary conditions, this results in a system of N_z linear equations with a banded coefficient matrix. The bandwidth depends on the method chosen for approximation

of the differential equation. In our implementation, we use a 6th order finite difference scheme resulting in a seven-diagonal coefficient matrix.[1] The computational complexity to solve a system of linear equations with banded coefficient matrix of size N_z with fixed bandwidth is $O(N_z)$. The systems have to be solved for every pair of spatial frequencies k_x, k_y. As a result, the total complexity of the second step is $O(N_x N_y N_z)$.

The last step is the summation formula (18), which is carried out by the inverse FFT with the same computational complexity as the first step. Summing up all estimates, we conclude that numerical complexity of the described algorithm is $O(N_x N_y N_z \log(N_x N_y))$.

It is possible to reduce the number of floating point operations while inverting L_0. Solutions to equations (16) vanish rapidly for $k_x^2 + k_y^2 > \frac{\omega^2}{c_{\min}^2}$, where $c_{\min} = \min_{0 \le z \le Z} c(z)$. So, in practice, for a given accuracy level there is no need to consider equations for the spatial frequencies lying outside the circle $k_x^2 + k_y^2 \le \gamma \frac{\omega^2}{c_{\min}^2}$ (for example, to achieve 1% accuracy one can take $\gamma = 2.5$).

4. Preconditioning the Helmholtz equation in a 3D inhomogeneous medium

So far, we have dealt with vertically inhomogeneous media. Now we proceed to a general case of the Helmholtz equation

$$Lu = \Delta u(x, y, z) + \frac{\omega^2}{c^2(x, y, z)} u(x, y, z) = f(x, y, z) \tag{20}$$

in cuboid

$$D' = \{(x, y, z) : X_l < x < X_r, Y_l < y < Y_r, 0 < z < Z_b\}.$$

We embed D' in some bigger cuboid D_{bound} defined by (14). As before, boundary $z = 0$ is treated as a free surface.

We expand equation (20) from D' to domain D_{bound}. Outside of D' function f is set to zero. In subdomain $\{(x, y, z) : 0 < x < X, 0 < y < Y, Z_b < z < Z_1\}$ we build a PML assuming the sound velocity function in this subdomain is extended by a constant as described in section 2. The lateral velocity extension outside of D' will be explained later in this section. As a result, we come to equation[2]

$$Lu = \tilde{\Delta} u(x, y, z) + \frac{\omega^2}{c^2(x, y, z)} u(x, y, z) = f(x, y, z) \tag{21}$$

in domain D_{bound} that includes the PML and where the unknown function satisfies boundary conditions

$$\begin{aligned} u(x, y, 0) &= 0, \\ u(x, y, Z_1) &= 0. \end{aligned} \tag{22}$$

On vertical boundaries of D_{bound} we set periodic boundary conditions (cf. (19)) with respect to x and y

$$\begin{aligned} u(0, y, z) &= u(X, y, z), \\ u(x, 0, z) &= u(x, Y, z). \end{aligned} \tag{23}$$

This way, the boundary value problem is completely defined and the operator L, which is introduced in (21), is defined as well. Its domain is the functional space $H^1(D_{bound})$ of functions u that satisfy boundary conditions (22) and (23). Note that by using periodic boundary conditions we ensure the domains of operators L and L_0 (subsection 2.2) are the same. This is crucial for applicability of the algorithm.

Let operator L_0 be defined by a modified Helmholtz equation in a layered medium (cf. equation (13))

$$L_0 u = \tilde{\Delta} u(x, y, z) + (1 - i\beta) \frac{\omega^2}{c_0^2(z)} u(x, y, z) \tag{24}$$

with the same boundary conditions as for operator L. Given $c_0(z)$ this operator acts on functions defined in D_{bound} and has the same domain as operator L.

[1] In a particular case of piecewise constant function $c(z)$, it is possible to get the solution in a closed form of linear combinations of exponential functions (the up-going and the down-going waves) on each interval where $c(z)$ is constant [12]. For this approach, the coefficient matrix of the system of linear equations is tridiagonal.

[2] Note we use the same notation L for the operator.

Operator L can be represented as a perturbation of operator L_0:

$$\begin{aligned} Lu &= \Delta u + \frac{\omega^2}{c^2(x,y,z)}u \\ &= \Delta u + (1-i\beta)\frac{\omega^2}{c_0^2(z)}u + \left(\frac{\omega^2}{c^2(x,y,z)} - (1-i\beta)\frac{\omega^2}{c_0^2(z)}\right)u \\ &= L_0 u - \delta L u, \end{aligned} \tag{25}$$

where the action of operator δL on function u is multiplication of u by function

$$\gamma(x,y,z;\omega,\beta) = -\omega^2\left(\frac{1}{c^2(x,y,z)} - \frac{(1-i\beta)}{c_0^2(z)}\right).$$

Let us explain how the velocity function $c(x,y,z)$ is defined outside of horizontal cross-sections of D'. Assume $c_0(z)$ is already defined. To avoid reflections from the interfaces between D' and D_{bound}, we require $c(x,y,z)$ to be continuous across these interfaces. Additionally, function $\gamma(x,y,z;\omega,\beta)$ that defines operator δL is required to be zero on outer boundaries of D_{bound}. One may notice that, due to these conditions, $c(x,y,z)$ becomes complex valued in the relative complement of D' with respect to D_{bound}. Using a complex valued sound velocity is equivalent to introducing absorbing layers (see [23]) and helps to reduce the influence of periodic boundary conditions on the solution.

Taking into account the invertibility of L_0 (section 2.2) we can factor L as a product of two operators

$$L = L_0 - \delta L = \left(I - \delta L \cdot L_0^{-1}\right) L_0. \tag{26}$$

Factorization (26) can be used to split the solution of equation $Lu = f$ in two steps. First, we can solve equation

$$\left(I - \delta L \cdot L_0^{-1}\right) v = f, \tag{27}$$

and after that find u from equation $L_0 u = v$.

Factorization (26) can be treated as right preconditioning of operator L by operator L_0^{-1}. The preconditioned operator (the operator of the equation (27)) is the unity operator I perturbed by bounded operator $\delta L \cdot L_0^{-1}$. To reduce the perturbation, function $c_0(z)$ should be close to $c(x,y,z)$ and can be defined as some average of velocity $c(x,y,z)$ over horizontal cross-sections of D_{bound}. For example, we can use the following formula:

$$\frac{1}{c_0^2(z)} = \frac{1}{XY}\int_0^X\int_0^Y \frac{1}{c^2(x,y,z)}dxdy.$$

Another way to define $c_0(z)$ is

$$c_0(z) = \frac{1}{2}\left(\max_{(x,y)} c(x,y,z) + \min_{(x,y)} c(x,y,z)\right).$$

For the last choice, the C-norm of the difference $\left(\frac{1}{c^2(x,y,z)} - \frac{1}{c_0^2(z)}\right)$ would be minimized, which might help to minimize the norm of operator δL (for the case $\beta = 0$ only). Let us note that it is not the norm, but the spectrum of perturbation $\delta L \cdot L_0^{-1}$ that influences the convergence of the Krylov subspace iteration method used to solve equation $\left(I - \delta L \cdot L_0^{-1}\right) v = g$. Therefore, we would suggest experimenting with different ways of averaging $c(x,y,z)$.

In order to solve equation $\left(I - \delta L \cdot L_0^{-1}\right) v = g$ we apply the Induced Dimension Reduction algorithm (IDR [24–26]), which has moderate memory requirements. As with any Krylov subspace iteration type algorithm, IDR requires an effective implementation of the matrix-by-vector multiplication, which in our case is evaluating expression $\left(I - \delta L \cdot L_0^{-1}\right) w$ for a given vector w. For w being a 3D grid function, this operation consists of applying the inverse of operator L_0 described in section 2, pointwise multiplication of the result by scalar function $\gamma(x,y,z;\omega,\beta)$ (once again, a 3D grid function) and subtracting the last result from w. The IDR itself uses a few BLAS (Basic Linear Algebra Subprograms) level 1 [27] operations with vectors. All needed functionality (including the FFT) is available in the highly optimized software library Intel® MKL [28] we use in our solver.

Our numerical experiments show that preconditioner L_0 is effective for models of moderate complexity. In situations of strong lateral variations, high contrasts or high frequency ω, convergence rate of the iterative method may be slow or even may fail to converge. To handle this issue we suggest a second-level "$\delta L \cdot L_0^{-1}$-based" preconditioner [29]. The reason relies on the fact that the cost of evaluation of $\delta L \cdot L_0^{-1}$ is small enough, especially if it is allowed to sacrifice accuracy for the sake of faster computations.

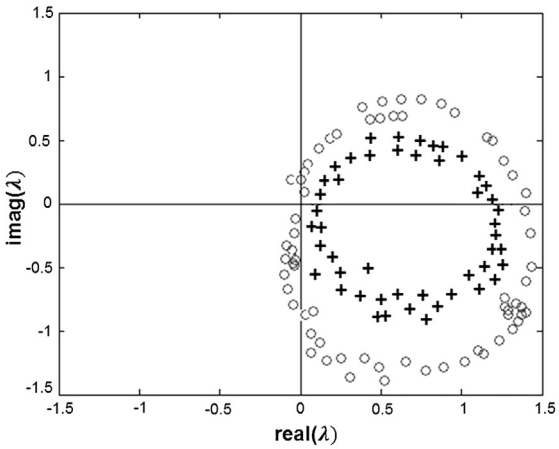


Fig. 1. Eigenvalues of two preconditioned operators: circles correspond to a single-stage preconditioner; pluses correspond to a two-stage preconditioner.

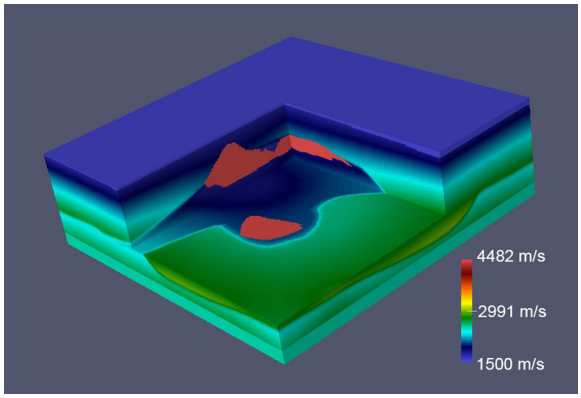


Fig. 2. A 3D view of the SEG/EAGE salt P-wave velocity model (13.5 km × 13.5 km × 4.2 km).

We introduce a second-level preconditioner for equation (27)

$$\left(I-\delta L\cdot L_0^{-1}\right)\left(I-\alpha\cdot\delta L\cdot L_0^{-1}\right)^{-1}\tilde{v}=g, \tag{28}$$

where $0<\alpha<1$ is some parameter, $\tilde{v}=\left(I-\alpha\cdot\delta L\cdot L_0^{-1}\right)L_0u$. Using a truncated Neumann series decomposition of the second preconditioner in equation (28) one obtains an expression for the numerical implementation:

$$\left(I-\delta L\cdot L_0^{-1}\right)\left(I+\alpha\delta L\cdot L_0^{-1}+\cdots+\left(\alpha\delta L\cdot L_0^{-1}\right)^N\right)\tilde{v}=g, \tag{29}$$

where N is some predefined parameter.

The impact of the second-level preconditioner is briefly illustrated in Fig. 1 where we compare the eigenvalues of operator $\left(I-\delta L\cdot L_0^{-1}\right)$ and a two-stage preconditioner given by the left-hand side operator in equation (29). This experiment has been performed using the SEG/EAGE salt model [30] (Fig. 2) at a frequency of 8 Hz. In Fig. 1 we present so-called extremal eigenvalues, i.e., eigenvalues with a maximum magnitude or a maximum/minimum real/imaginary part. These eigenvalues have been estimated by the Arnoldi [31] method. Eigenvalues corresponding to the two-stage preconditioner are better clustered around unity on the complex plane and better separated from the origin. This leads to a better convergence for Krylov-type iterative methods (Fig. 3). For more information regarding the influence of the two-level preconditioner refer to Appendix A.

As one might notice, in particular from Fig. 3, the usage of the second-level preconditioner is not free, namely, it increases the computational time of each iteration. However, cost can be decreased if inversion of L_0 in (29) is replaced with a less costly operation. For instance, we can decrease the approximation order of the finite difference schemes with respect to variable z used to solve boundary value problems (16), (17) – instead of the 6th order finite difference scheme use the

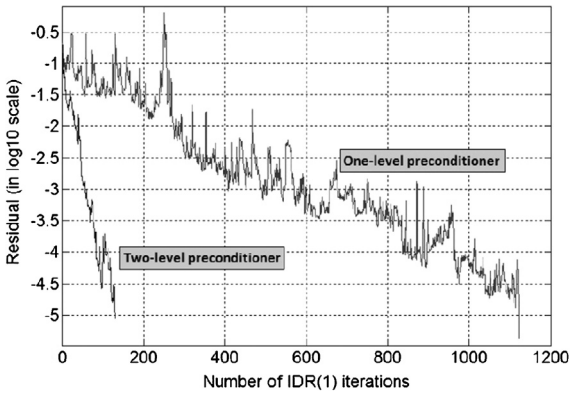


Fig. 3. Convergence curves of the iterative solver using a one-level and a two-level preconditioners. Note that computation time is 5.5 times longer for the two-level preconditioner. The two-level preconditioner (29) was used with parameters $\alpha = 0.75$ and $N = 12$.

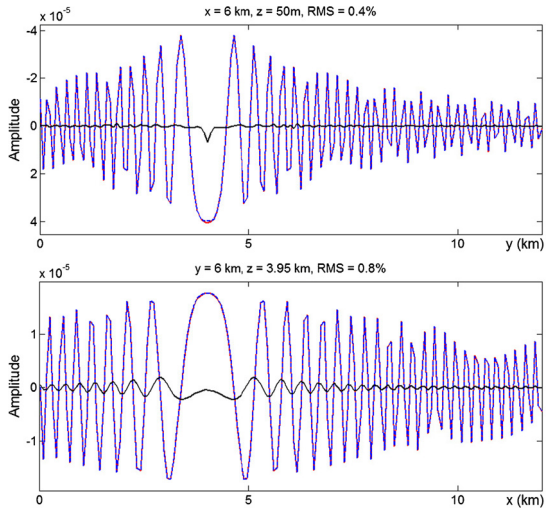


Fig. 4. 1D horizontal profiles (real part) of solutions for a homogeneous model at locations $x = 6000$ m, $z = 50$ m (top) and $y = 6000$ m, $z = 3950$ m (bottom). Exact solutions are dashed blue lines, our solver solutions – red lines and residuals multiplied by 10 – black lines.

2nd order scheme. One more optimization opportunity is decreasing the number of the spatial frequencies to be taken into account (see the last paragraph of Section 3) while inverting L_0 to build the second-level preconditioner.

5. Numerical experiments

In this section we show the results of several numerical experiments that have been performed using the solver. Computations were run on one cluster node with an Intel® Xeon® CPU E5-2670 v2 @ 2.50 GHz. As a stopping criterion for an IDR we use a 10^{-5} threshold for the relative residual of the L_2-norm.

5.1. Homogeneous medium

The first example comprises a domain of size 12 km × 12 km × 4 km filled with an acoustic medium with homogeneous velocity 1000 m/s discretized over a 151 × 151 × 161 uniform grid with steps $h_x = h_y = 80$ m, $h_z = 25$ m. The wave is excited by a point source located at the point $x = 4000$ m, $y = 4000$ m, $z = 25$ m. For this experiment we used PMLs on both top and bottom boundaries of width 20 grid points. The width of the absorbing zones in lateral directions are 30 grid points.

The solution at frequency $\nu = 5$ Hz obtained by our solver is compared to the exact solution $\frac{-e^{ikr}}{4\pi r}$, where r is the distance from the observation point to the source position, $k = \frac{2\pi\nu}{c} = 0.01\pi$ m^{-1}. Note that horizontal grid steps of 80 m correspond to 2.5 grid steps per wavelength at this frequency and velocity. In Fig. 4, 1D profiles of these solutions along a few lines are given which show the good agreement with the exact solution.

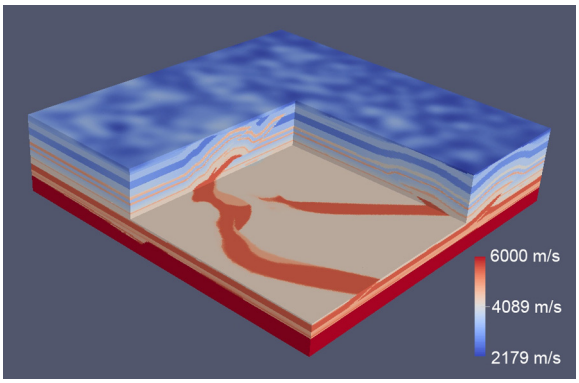


Fig. 5. A 3D view of the SEG/EAGE overthrust P-wave velocity model (19.8 km $\times$ 19.8 km $\times$ 4.65 km).

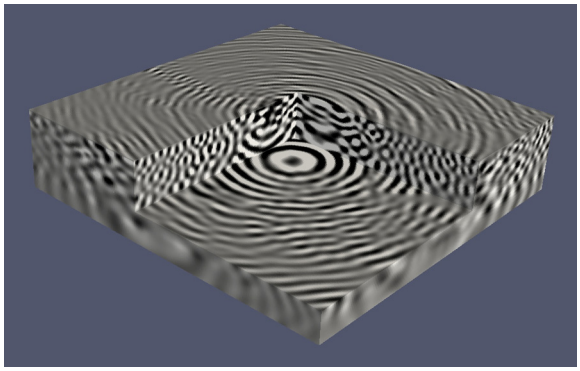


Fig. 6. A 3D view of the real part of the frequency–domain wavefield for the SEG/EAGE overthrust model.

5.2. 3D SEG/EAGE overthrust model

In this test the 3D SEG/EAGE overthrust model [30] is used. P-wave velocity is represented over a $660 \times 660 \times 155$ grid (Fig. 5) with cell sizes of 30 m in all dimensions resulting in 10 grid points per minimal wavelength. At the top boundary a free surface boundary condition was used. The width of the PML at the bottom boundary is 30 grid points and the width of the absorbing zones in the lateral directions is 60 grid points. The solution was computed for the frequency of 7 Hz with source coordinates $x = 9900$ m, $y = 9900$ m and $z = 30$ m.

The simulated wavefield is depicted in Fig. 6. To verify the solution obtained with our solver we compare it to the solution obtained with a time-domain finite-difference (TDFD) scheme. Snapshots and 1D profiles along the x-axis of both solutions are presented in Figs. 7 and 8, respectively. Their close agreement is clearly seen (RMS difference is 2.8–3.2%). The total computation time for the iterative solver is 21.6 min.

5.3. Convergence rate of the iterative solver in a complex model

To estimate the convergence rate and the computational time in the media with strong lateral contrasts and the impact of the second-level preconditioner consider the model presented in Fig. 2. We discretize this model with a uniform grid of $540 \times 540 \times 168$ points and cell sizes of 25 m. At the top boundary a free-surface boundary condition was used. The width of the PML at the bottom boundary is 30 grid points and the width of the absorbing zones in the lateral directions is 60 grid points. The source is placed in the middle of the target area at a depth of 25 m. We have run two types of experiments: with preconditioner L_0 only and with the second-level preconditioner (29) with parameters $\alpha = 0.75$ and $N = 12$. As one can see from the Table 1 if we use a basic preconditioner L_0 poor convergence is an issue, particularly at higher frequencies. Use of the second-level preconditioner provides a reasonable convergence rate and improves the total performance.

5.4. Benchmarking against a known iterative solver

For purposes of performance comparison, we considered a solver developed by Plessix (see [9]). For the sake of brevity, everywhere below that solver is referred to as PS. Following the paper, the same simulations were performed in the parts of the SEG/EAGE overthrust and salt models. In Table 2, the computation times (row t_{os}) and the number of iterations (row n_{iter}) are presented as well as the data mentioned in [9] for PS (see rows t_{PS} and $n_{\text{iter,PS}}$). By the number of iterations we

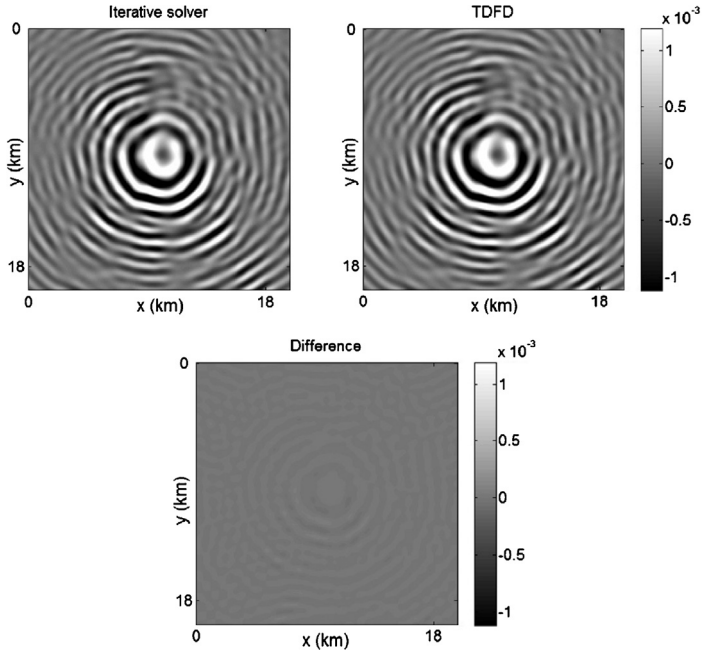


Fig. 7. Snapshots of the solution for the SEG/EAGE overthrust model at $z = 1500$ m; the top left picture computed by our solver, the top right picture computed with TDFD and the bottom one is their difference.

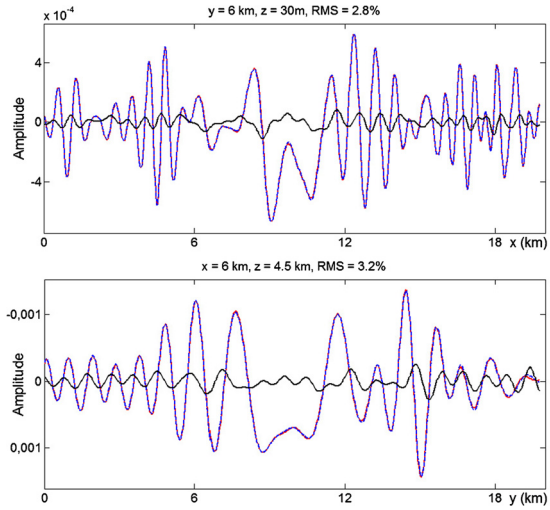


Fig. 8. 1D horizontal profiles (real part) of solutions for the SEG/EAGE overthrust model at locations $y = 6000$ m, $z = 30$ m (top) and $x = 6000$ m, $z = 4500$ m (bottom). Exact solutions are dashed blue lines, our solver solutions – red lines and residuals multiplied by 5 – black lines.

Table 1
Convergence of the algorithm under different conditions of the source frequency and preconditioner.

		8 Hz	15 Hz
L_0 preconditioner	Number of iterations	1715	No convergence
	Computation time	106 min	No convergence
Second-level preconditioner	Number of iterations	162	624
	Computation time	54 min	207 min

Table 2
Our solver vs PS: model sizes are in grid points; h (m) stands for grid steps (the same in all directions) used to approximate the models; t_{os} (min) – computation time (in minutes) for our solver; t_{PS} (min) – computation time for PS. See more details in the text.

	SEG overthrust model		SEG salt model	
ν (Hz)	5	10	5	10
model size	261 × 261 × 110	471 × 471 × 160	295 × 295 × 140	520 × 520 × 210
h (m)	100	50	60	30
t_{os} (min)	3	127	7	293
n_{iter}	24	131	47	225
t_{PS} (min)	126	1380	154	1111
$n_{\text{iter,PS}}$	125	252	125	210

mean the number of iterations of IDR(1) only and do not take into account the second-level preconditioner iterations. Note that PS is based on BiCGSTAB algorithm [32].

We tested our method on Intel® Core™ i7-3770K @3.5 GHz with all parallelization options switched off, whereas PS was run on AMD Opteron™ @2.2 GHz.

Definitely, data in Table 2 are not applicable for 'apple-to-apple' comparison due to the difference in hardware and software, code optimizations applied and, possibly, some other factors. Having no access to the actual PS code, it is problematic to estimate its computation time on our hardware and with our settings. However, one might infer that PS is less sensitive to the frequency increase then our solver: two times increase of the frequency results in a factor of 5 increase of iterations for our solver and a factor of 2 increase for PS. One of the reasons for that is different preconditioners applied in these solvers. Both of them are shifted Laplace operators but built in a different way. In our case a shifted Laplace operator is constructed as some 1D approximation of an original 3D sound velocity. This greatly benefits us while inverting the preconditioner and this way we satisfy the requirement for a preconditioner to be 'easily' inverted. For PS a preconditioner is a shifted Laplace operator for the original sound velocity which is approximately inverted by applying of one step of the multigrid V-cycle. Different preconditioners lead to different spectral properties of the preconditioned operator involved in computations.

Another fact that Table 2 reflects is that for higher frequencies the price per iteration increases. But this is in a good agreement with respective increase of vectors' size.

6. Conclusions

We developed a frequency–domain iterative solver for 3D acoustic wave propagation problems. The boundary value problem for the Helmholtz equation in an unbounded domain is approximated by posing the problem in a bounded domain surrounded with PMLs and absorbing zones. The operator of the Helmholtz equation is considered as a perturbation of a shifted Laplace operator for a layered model. After preconditioning by the shifted Laplace operator, the operator of the Helmholtz equation becomes a perturbation of the unity operator and the respective equation can be effectively solved by modern Krylov-type subspace iteration methods. The shifted Laplace operator for a layered model can be inverted using FFTs and solving the resulting systems of linear equations with banded coefficient matrices. Matrix-by-vector multiplication in iterative inversion of the preconditioned operator involves inversion of the preconditioner and pointwise multiplication of grid functions.

The solver has been verified by simulation in several models, and the results have been benchmarked against known solutions for a homogeneous velocity model with the fundamental solution of the Helmholtz equation in R^3, and for the SEG/EAGE overthrust model against a solution obtained by using TDFD. The accuracy of the proposed method with relatively reasonable computational time justifies using the solver in FWI problems.

In the future, we plan to extend the approach to the elastic wave modeling.

Acknowledgements

We greatly appreciate the hard work of Vincent Etienne, Michael Jervis, and Robert Smith who reviewed our manuscript and suggested numerous ideas to improve it. Special thanks to Jean Virieux and an anonymous reviewer for their insightful notes and interesting references. We thank Saudi Aramco for permission to publish this paper. Three of the authors (Victor Kostin, Dmitry Neklyudov, and Vladimir Tcheverda) have been supported by RFBR (Grant Nos. 15-35-20022, 15-05-01310, 15-55-20004, 16-05-00800 and 17-01-00416). Vladimir Tcheverda was partially sponsored by the Ministry of Education and Science of the Republic of Kazakhstan Grant No. 1771/GF4. We are grateful to the Siberian Supercomputing Center and Moscow State University Supercomputing Center for providing us with computational resources.

Appendix A. Spectrum of a double-preconditioned operator

In this section we describe the impact of second-level preconditioner (28) while solving equation (27). In particular, for different values of parameter α we derive the area of possible locations of eigenvalues of matrix $\delta L \cdot L_0^{-1}$ for the second-level preconditioner to be reasonable to apply.

For the sake of convenience we denote matrix $\delta L \cdot L_0^{-1}$ by B. Then equation (27) takes the form

$$(I - B)v = f. \tag{A.1}$$

The Neumann series expansion of the inverse to the left-hand side operator in (A.1) gives the representation for the unknown function

$$v = f + Bf + B^2 f + \cdots. \tag{A.2}$$

This series converges if all eigenvalues λ of matrix B belong to unit disc $|\lambda| < 1$. If this condition is not satisfied one might precondition equation (A.1) a second time by operator $(I - \alpha B)^{-1}$ and come to equation

$$(I - B)(I - \alpha B)^{-1} w = f \tag{A.3}$$

with left-hand side operator $C = (I - B)(I - \alpha B)^{-1}$ being a function of matrix B. Its eigenvalues μ are related to eigenvalues λ of matrix B in the following way

$$\mu = \frac{1 - \lambda}{1 - \alpha\lambda}. \tag{A.4}$$

Similarly to formula (A.2), we might represent the unknown function w in (A.3) as

$$w = f + (I - C)f + (I - C)^2 f + \cdots.$$

The convergence criterion for the series in the right-hand side is that all eigenvalues μ of matrix C have to be inside disc $|\mu - 1| < 1$. Applying the second-level preconditioner we aim to get a new operator, i.e. operator C, that satisfies this criterion. In other words, the second-level preconditioner is beneficial if operator B is such that transform (A.4) maps its eigenvalues λ onto disc $|\mu - 1| < 1$. Below we present the domain that is mapped onto this disc depending on the different values of parameter α.

- For $\alpha \in (0.5, 1)$ this is an exterior of disc

  $$|\lambda - \frac{\alpha}{2\alpha - 1}| < \left|\frac{1 - \alpha}{2\alpha - 1}\right| \tag{A.5}$$

  (for $\alpha = 0.75$ this disc is a dark gray {3} area on the left image in Fig. A.1). Inversion of operator $(I - \alpha B)$ in (A.3) by the Neumann series expansion

  $$(I - \alpha B)^{-1} = I + \alpha B + (\alpha B)^2 + \cdots \tag{A.6}$$

  implies the necessary condition

  $$|\alpha\lambda| < 1 \tag{A.7}$$

  to be satisfied. Hence, the area of interest is an intersection of exterior of disc (A.5) with disc $|\lambda| < 1/\alpha$ (for $\alpha = 0.75$ this is a union of bright grey {2} and grey {1} domains on the left image in Fig. A.1). Note, that if all eigenvalues λ are in disc $|\lambda| < 1$ (a bright grey {2} domain on the left image in Fig. A.1) then the second-level preconditioner is not needed at all, since the convergence criterion for the series in (A.2) is already satisfied.
- For $\alpha \in [1/3, 1/2)$ this is an interior of disc (A.5) (for $\alpha = 0.35$ this disc is a bright grey {2} plus grey {1} domain on the right image in Fig. A.1). Taking into account the necessary condition (A.7) the area of interest is an intersection of disc (A.5) with disc (A.7) (for $\alpha = 0.35$ this area is depicted by bright grey {2} color on the right image in Fig. A.1).
- For $\alpha \in (0, 1/3)$ the domain we search for is defined in the same way as in the previous case, but disc (A.5) is completely located inside disc (A.7) and that is why it is the whole disc (A.5).
- For $\alpha = 1/2$ formula (A.5) is not defined. In this case the half-plane $\mathrm{Re}(\lambda - 1) < 0$ stands instead of disc (A.5). Intersection of this half-plane with disc (A.7) is the area we search for.

A union of the aforementioned domains gives an area of possible locations for eigenvalues λ such that the presented second-level preconditioning technique (see formula (A.3)) is beneficial. Its boundary is a cardioid-like line $\lambda(\alpha) = x(\alpha) + iy(\alpha)$ parametrized in the following way

$$x(\alpha) = \frac{1}{2\alpha} + \frac{1}{\alpha^2} - \frac{1}{2\alpha^3}, \quad y(\alpha) = \pm\sqrt{\frac{1}{\alpha^2} - x^2(\alpha)}, \qquad \alpha \in [\frac{1}{3}, 1]. \tag{A.8}$$

It represents the intersection points of boundaries of discs (A.5) and (A.7) (see the bold black line in Fig. A.2).

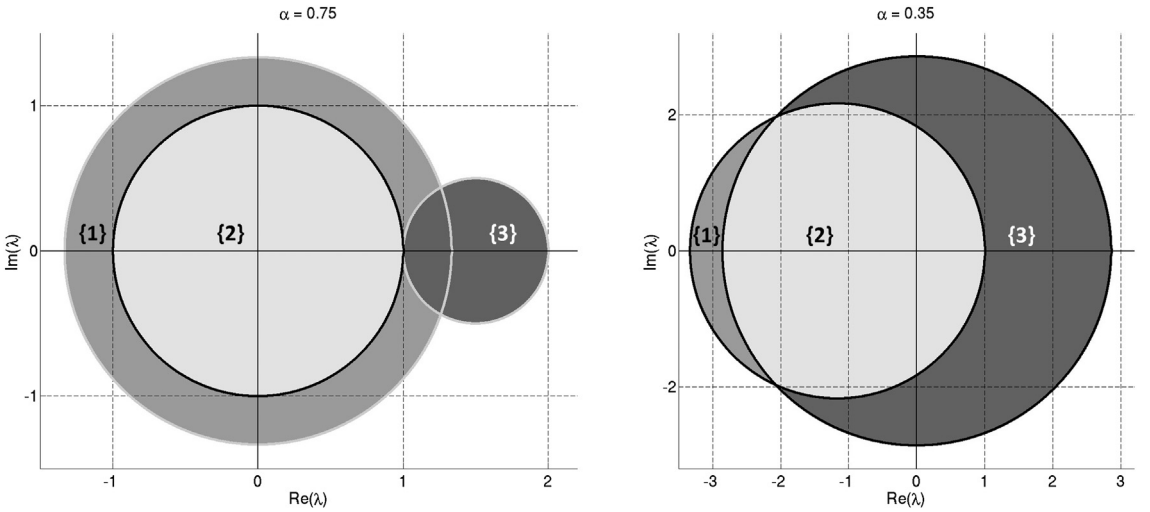


Fig. A.1. The left image: the domain depicted in bright grey {2} and grey {1} reflects the possible locations for eigenvalues of operator $\delta L \cdot L_0^{-1}$ in order to the second-level preconditioner (see formula (A.3)) be beneficial with $\alpha = 0.75$. The right image: the domain depicted in bright grey {2} reflects the possible locations for eigenvalues of operator $\delta L \cdot L_0^{-1}$ in order to the second-level preconditioner be beneficial with $\alpha = 0.35$.

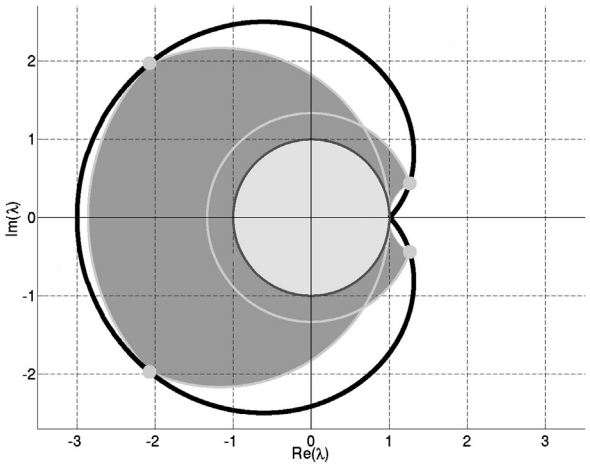


Fig. A.2. Cardioid-like bold black line encompasses the area of all possible eigenvalues of operator $\delta L \cdot L_0^{-1}$ for which the second-level preconditioner defined by formula (A.3) is reasonable to be applied. This line passes through intersections of the respective circles (see Fig. A.1), in particular ones depicted by bold bright grey dots.

It is worth mentioning that when solving the problem numerically we consider only the finite number of items in Neumann series (A.6) and the domain surrounded by the cardioid-like line (A.8) will be replaced by its subdomain containing the unit disc (see the bright grey domain in Fig. A.2). It will increase with the increasing number of items in the Neumann series remaining its subdomain.

References

[1] W. Mulder, R.-E. Plessix, How to choose a subset of frequencies in frequency–domain finite-difference migration, Geophys. J. Int. 158 (2004) 801–812, http://dx.doi.org/10.1111/j.1365-246X.2004.02336.x.
[2] J. Virieux, S. Operto, H. Ben-Hadj-Ali, R. Brosssier, V. Etienne, F. Sourber, L. Giraud, A. Haidar, Seismic wave modeling for seismic imaging, Lead. Edge 28 (2009) 538–544, http://dx.doi.org/10.1190/1.3124928.
[3] C. Shin, Y.H. Cha, Waveform inversion in the Laplace domain, Geophys. J. Int. 173 (2008) 922–931, http://dx.doi.org/10.1111/j.1365-246X.2008.03768.x.
[4] B.R. Vainberg, Principles of radiation, vanishing attenuation and limit amplitude in the general theory of partial differential equations, Usp. Mat. Nauk 21 (1966) 115–194 (in Russian).
[5] K.T. Nihei, X. Li, Frequency response modelling of seismic waves using finite difference time domain with phase sensitive detection (TD-PSD), Geophys. J. Int. 169 (2007) 1069–1078, http://dx.doi.org/10.1111/j.1365-246X.2006.03262.x.
[6] V. Etienne, T. Tonellot, P. Thierry, V. Berthoumieux, C. Andreolli, Optimization of the seismic modeling with the time-domain finite-difference method, in: 84th SEG Annual Meeting Expanded Abstracts, 2014, pp. 3536–3540.

[7] R.E. Plessix, Three-dimensional frequency–domain full-waveform inversion with an iterative solver, Geophysics 74 (2009) WCC149–WCC157, http://dx.doi.org/10.1190/1.3211198.
[8] Y.A. Erlangga, F.J. Herrmann, Full-waveform inversion with Gauss–Newton–Krylov method, in: 79th SEG Annual Meeting Expanded Abstracts, SEG, 2009, pp. 2357–2361.
[9] R.E. Plessix, A Helmholtz iterative solver for 3D seismic-imaging problems, Geophysics 72 (2007) SM185–SM194, http://dx.doi.org/10.1190/1.2738849.
[10] Y.A. Erlangga, F.J. Herrmann, An iterative multilevel method for computing wavefields in frequency–domain seismic inversion, in: 78th SEG Annual Meeting Expanded Abstracts, SEG, 2008, pp. 1957–1960.
[11] L. Trefethen, D. Bau, Numerical Linear Algebra, SIAM, Philadelphia, PA, 1997.
[12] D. Neklyudov, V. Tcheverda, A Helmholtz iterative solver without of finite-difference approximations, in: 72nd EAGE Conference and Exhibition Expanded Abstracts, 2010, p. G006.
[13] Y.A. Erlangga, C. Vuik, C.W. Oosterlee, On a class of preconditioners for solving the Helmholtz equation, Appl. Numer. Math. 50 (2004) 409–425, http://dx.doi.org/10.1016/j.apnum.2004.01.009.
[14] C.W. Oosterlee, C. Vuik, W.A. Mulder, R.-E. Plessix, Shifted-Laplacian preconditioners for heterogeneous Helmholtz problems, in: Advanced Computational Methods in Science and Engineering, Springer, 2010, pp. 21–46.
[15] T. Airaksinen, S. Mönkölä, Comparison between the shifted-Laplacian preconditioning and the controllability methods for computational acoustics, J. Comput. Appl. Math. 234 (2010) 1796–1802, http://dx.doi.org/10.1016/j.cam.2009.08.030.
[16] Y.A. Erlangga, R. Nabben, On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian, Electron. Trans. Numer. Anal. 31 (2008) 403–424.
[17] A. Sheikh, D. Lahaye, L.G. Ramos, R. Nabben, C. Vuik, Accelerating the shifted Laplace preconditioner for the Helmholtz equation by multilevel deflation, J. Comput. Phys. 322 (2016) 473–490, http://dx.doi.org/10.1016/j.jcp.2016.06.025.
[18] O. Pankratov, D. Avdeyev, A. Kuvshinov, Electromagnetic field scattering in a heterogeneous earth: a solution to the forward problem, Phys. Solid Earth 31 (3) (1995) 201–209.
[19] J.-P. Berenger, Three-dimensional perfectly matched layer for the absorption of electromagnetic waves, J. Comput. Phys. 127 (1996) 363–379, http://dx.doi.org/10.1006/jcph.1996.0181.
[20] I. Singer, E. Turkel, A perfectly matched layer for the Helmholtz equation in a semi-infinite strip, J. Comput. Phys. 201 (2004) 439–465, http://dx.doi.org/10.1016/j.jcp.2004.06.010.
[21] F. Collino, C. Tsogka, Application of the PML absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media, Tech. rep. RR-3471, INRIA, 1998.
[22] K.R. Rao, D.N. Kim, J.J. Hwang, Fast Fourier Transform: Algorithms and Applications, Springer, Netherlands, 2010.
[23] P.G. Petropoulos, Reflectionless sponge layers as absorbing boundary conditions for the numerical solution of Maxwell equations in rectangular, cylindrical, and spherical coordinates, SIAM J. Appl. Math. 60 (2000) 1037–1058, http://dx.doi.org/10.1137/S0036139998334688.
[24] P. Sonneveld, M.B. van Gijzen, IDR(s): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations, SIAM J. Sci. Comput. 31 (2008) 1035–1062, http://dx.doi.org/10.1137/070685804.
[25] M.B. van Gijzen, P. Sonneveld, Algorithm 913: an elegant IDR (s) variant that efficiently exploits biorthogonality properties, ACM Trans. Math. Softw. 38 (1) (2011) 5, http://dx.doi.org/10.1145/2049662.2049667.
[26] M.B. van Gijzen, G.L. Sleijpen, J.P.M. Zemke, Flexible and multi-shift induced dimension reduction algorithms for solving large sparse linear systems, Numer. Linear Algebra Appl. 22 (1) (2015) 1–25, http://dx.doi.org/10.1002/nla.1935.
[27] BLAS (Basic Linear Algebra Subprograms), http://www.netlib.org/blas/.
[28] Intel® Math Kernel Library (Intel® MKL), https://software.intel.com/en-us/intel-mkl.
[29] D. Neklyudov, M. Dmitriev, M. Belonosov, V. Tcheverda, Frequency–domain iterative solver for 3D acoustic wave equation with two-stage semi-analytical preconditioner, in: 76th EAGE Conference and Exhibition Expanded Abstracts, 2014, Tu G105 09.
[30] F. Aminzadeh, J. Brac, T. Kuntz, 3-D salt and overthrust models, in: SEG/EAGE Modelling Series, vol. 1, SEG, Tulsa, OK, 1997.
[31] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia, PA, 2003.
[32] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comp. 13 (2) (1992) 631–644, http://dx.doi.org/10.1137/0913035.

A Review of Curvelets and Recent Applications

Jianwei Ma[1,2] and Gerlind Plonka[3]

[1] School of Aerospace, Tsinghua University, Beijing 100084, China
[2] Centre de Geosciences, Ecole des Mines de Paris, 77305 Fontainebleau Cedex, France
[3] Department of Mathematics, University of Duisburg-Essen, 47048 Duisburg, Germany

Abstract

Multiresolution methods are deeply related to image processing, biological and computer vision, scientific computing, etc. The curvelet transform is a multiscale directional transform, which allows an almost optimal nonadaptive sparse representation of objects with edges. It has generated increasing interest in the community of applied mathematics and signal processing over the past years. In this paper, we present a review on the curvelet transform, including its history beginning from wavelets, its logical relationship to other multiresolution multidirectional methods like contourlets and shearlets, its basic theory and discrete algorithm. Further, we consider recent applications in image/video processing, seismic exploration, fluid mechanics, simulation of partial different equations, and compressed sensing.

Keywords: curvelets, wavelets, multiscale geometric analysis, image processing, biological and computer vision, seismic exploration, turbulence, surface metrology, compressed sensing

1 Introduction

Digital images are two-dimensional matrices in image processing. One important task is to adjust the values of these matrices in order to get clear features in images. The adjusting of values obeys a certain mathematical model. The main challenge is how to build suitable mathematical models for practical requirements. Taking image denoising as an example, many mathematical models are based on a frequency partition of the image, where components with high frequency are interpreted as noise that have to be removed while those with low frequency are seen as features to be remained. Curvelets, which are going to be reviewed in this paper, can be seen as an effective model that not only considers a multiscale time-frequency local partition but also makes use of the direction of features.

Most natural images/signals exhibit line-like edges, i.e., discontinuities across curves (so-called line or curve singularities). Although applications of wavelets have become increasingly popular in scientific and engineering fields, traditional wavelets perform well only at representing point singularities, since they ignore the geometric properties of structures and do not exploit the regularity of edges. Therefore, wavelet-based compression, denoising, or structure extraction become computationally inefficient for geometric features with line and surface singularities. For example, when we download compressed images or videos, we often find a mosaic phenomenon (i.e., block artifacts along edges of the images). This mosaic phenomenon mainly results from the poor ability of wavelets to handle line singularities. In fluid mechanics, discrete wavelet thresholding often leads to oscillations along edges of the coherent eddies, and consequently, to the deterioration of the vortex tube structures, which in turn can cause an unphysical leak of energy into neighboring scales producing an artificial "cascade" of energy.

Multiscale methods are also deeply related with biological and computer vision. Since, Olshausen and Field's work in Nature (1996) [62], researchers in biological vision have re-iterated the similarity between vision and multiscale image processing. It has been recognized that the receptive fields of simple cells in a mammalian primary visual cortex can be characterized as being spatially localized, oriented and bandpass (selective to structure at different spatial scales). Therefore, they can be well represented by wavelet transforms. One approach to understand the response properties of visual neurons has been to consider their relationship to the statistical structure of natural images in terms of efficient coding. A coding strategy that maximizes sparseness is sufficient to account for the similar properties of simple-cell receptive fields [62, 63]. The wavelet transform yields sparse image representations, and hence provides an efficient way to understand the localized, oriented, bandpass receptive files, similar to those found in the primary visual cortex. However, wavelets do not supply a good direction selectivity, which is also an important response property of simple cells and neurons at stages of the visual pathway. Therefore, a directional multiscale sparse coding is desirable in this field.

One of the primary tasks in computer vision is to extract features from an image or a sequence of images [57]. The features can be points, lines, edges, and textures. A given feature is characterized by position, direction, scale and other property parameters. The most common technique, used in early vision for extraction of such features, is linear filtering, which is also reflected in models used in biological visual systems, i.e., human visual motion sensing. Objects at different scales can arise from distinct physical processes. This leads to the use of scale space filtering and multiresolution wavelet transform in this field. An important motivation for computer vision is to obtain directional representations which capture anisotropic lines and edges while providing sparse decompositions.

A multiresolution geometric analysis (MGA), named curvelet transform, was proposed [8, 10, 11, 12, 13] in order to overcome the drawbacks of conventional two-dimensional discrete wavelet transforms. In the two-dimensional (2D) case, the curvelet transform allows an almost optimal sparse representation of objects with C^2-singularities. For a smooth object f with discontinuities along C^2-continuous curves, the best m-term approximation $\tilde{f}_m$ by curvelet thresholding obeys $\|f - \tilde{f}_m\|_2^2 \leq Cm^{-2} (\log m)^3$, while for wavelets the decay rate is only m^{-1}. Combined with other methods, excellent performance of the curvelet transform has been shown in image processing, see e.g. [47, 55, 50, 69, 70]. Unlike the isotropic elements of wavelets, the needle-shaped elements of this transform possess very high directional sensitivity and anisotropy (see Fig. 1 for the $2D$ case). Such elements are very efficient in representing line-like edges. Recently, the curvelet transform has been extended to three dimensions by Ying et al. [8, 77].

Let us roughly compare the curvelet system with the conventional Fourier and wavelet analysis. The short-time Fourier transform uses a shape-fixed rectangle in Fourier domain, and conventional wavelets use shape-changing (dilated) but area-fixed windows. By contrast, the curvelet transform uses angled polar wedges or angled trapezoid windows in frequency domain in order to resolve also directional features.

The theoretic concept of curvelets is easy to understand, but how to achieve the discrete algorithms for practical applications is challenging. In the following, we first address a brief history of curvelets starting from classical wavelets. We also mention some other wavelet constructions that aim to improve the representation of oriented features towards visual reception and image processing. Then we will have a closer look at the definition and the properties of the continuous curvelet transform. We derive the discrete curvelet frame and the corresponding fast algorithm for the discrete curvelet transform in the two- and three-dimensional case. In

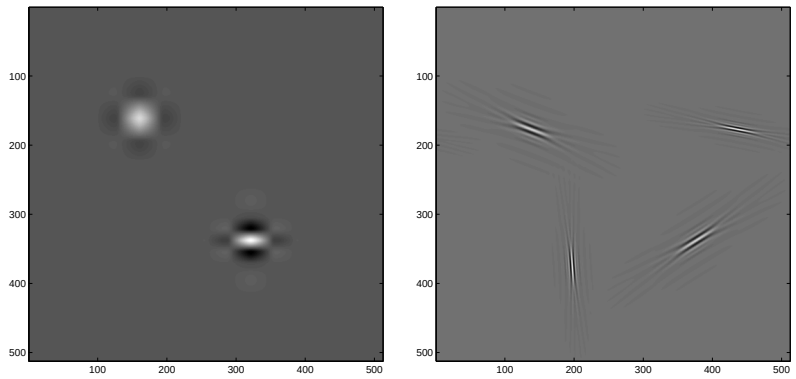


Figure 1: The elements of wavelets (left) and curvelets on various scales, directions and translations in the spatial domain (right). Note that the tensor-product 2D wavelets are not strictly isotropic but have directional selectivity.

particular, we present the construction of a curvelet system and its discretization by means of a typical example of "second generation curvelets". Finally, we show some recent applications of the curvelet transform in image and seismic processing, fluid mechanics, numerical treatment of partial differential equations, and compressed sensing.

2 From classical wavelets to curvelets

As outlined in the introduction, although the discrete wavelet transform (DWT) has established an impressive reputation as a tool for mathematical analysis and signal processing, it has the disadvantage of poor directionality, which has undermined its usage in many applications. Significant progress in the development of directional wavelets has been made in recent years. The complex wavelet transform is one way to improve directional selectivity and only requires $\mathcal{O}(N)$ computational cost. However, the complex wavelet transform has not been widely used in the past, since it is difficult to design complex wavelets with perfect reconstruction properties and good filter characteristics [33, 60]. One popular technique is the dual-tree complex wavelet transform (DT CWT) proposed by Kingsbury [41, 42], which added perfect reconstruction to the other attractive properties of complex wavelets, including approximate shift invariance, six directional selectivities, limited redundancy and efficient $\mathcal{O}(N)$ computation.

The 2D complex wavelets are essentially constructed by using tensor-product 1D wavelets. The directional selectivity provided by complex wavelets (six directions) is much better than that obtained by the classical DWT (three directions), but is still limited.

In 1999, an anisotropic geometric wavelet transform, named ridgelet transform, was proposed by Candès and Donoho [5, 9]. The ridgelet transform is optimal at representing straight-line singularities. This transform with arbitrary directional selectivity provides a key to the analysis of higher dimensional singularities. Unfortunately, the ridgelet transform is only applicable to objects with global straight-line singularities, which are rarely observed in real applications [52]. In order to analyze local line or curve singularities, a natural idea is to consider a partition for the image, and then to apply the ridgelet transform to the obtained sub-images. This block ridgelet based transform, which is named curvelet transform, was first proposed by Candès and Donoho in 2000, see [10]. Apart from the blocking effects, however, the application of this so-called

first-generation curvelet transform is limited because the geometry of ridgelets is itself unclear, as they are not true ridge functions in digital images. Later, a considerably simpler second-generation curvelet transform based on a frequency partition technique was proposed by the same authors, see [11, 12, 13]. Recently, a variant of the second-generation curvelet transform was proposed to handle image boundaries by mirror extension (ME) [22]. Previous versions of the transform treated image boundaries by periodization. Here, the main modifications are to tile the discrete cosine domain instead of the discrete Fourier domain, and to adequately reorganize the data. The obtained algorithm has the same computational complexity as the standard curvelet transform.

The second-generation curvelet transform [11, 12, 13] has been shown to be a very efficient tool for many different applications in image processing, seismic data exploration, fluid mechanics, and solving PDEs (partial different equations). In this survey, we will focus on this successful approach, and show its theoretical and numerical aspects as well as the different applications of curvelets. From the mathematical point of view, the strength of the curvelet approach is their ability to formulate strong theorems in approximation and operator theory. The discrete curvelet transform is very efficient in representing curve-like edges. But the current curvelet systems still have two main drawbacks: 1) they are not optimal for sparse approximation of curve features beyond C^2-singularities; 2) the discrete curvelet transform is highly redundant. The currently available implementations of the discrete curvelet transform (see www.curvelet.org) aim to reduce the redundancy smartly. However, independently from the good theoretical results on N-term approximation by curvelets, the discrete curvelet transform is not appropriate for image compression. The question of how to construct an orthogonal curvelet-like transform is still open.

3 Relationship of curvelets to other directional wavelets

There have been several other developments of directional wavelet systems in recent years with the same goal, namely a better analysis and an optimal representation of directional features of signals in higher dimensions. Non of these approaches has reached the same publicity as the curvelet transform. However, we want to mention shortly some of these developments and also describe their relationship to curvelets.

Steerable wavelets [32, 67], Gabor wavelets [44], wedgelets [26], beamlets [27], bandlets [58, 61], contourlets [24], shearlets [43, 35], wave atoms [23], platelets [76], surfacelets [46], have been proposed independently to identify and restore geometric features. These geometric wavelets or directional wavelets are uniformly called X-lets.

The steerable wavelets [32, 67] and Gabor wavelets [44] can be seen as early directional wavelets. The steerable wavelets were built based on directional derivative operators (i.e., the second derivative of a Gaussian), while the Gabor wavelets were produced by a Gabor kernel that is a product of an elliptical Gaussian and a complex plane wave. In comparison to separable orthonormal wavelets, the steerable wavelets provide translation-invariant and rotation-invariant representations of the position and the orientation of considered image structures. Applications of Gabor wavelets focused on image classification and texture analysis. Gabor wavelets have also been used for modeling the receptive field profiles of cortical simple cells. Applications of Gabor wavelets suggested that the precision in resolution achieved through redundancy may be a more relevant issue in brain modeling, and that orientation plays a key role in the primary

visual cortex. The main differences between steerable wavelets/Gabor wavelets and other X-lets is that the early methods do not allow for a different number of directions at each scale while achieving nearly critical sampling.

Contourlets, as proposed by Do and Vetterli [24], form a discrete filter bank structure that can deal effectively with piecewise smooth images with smooth contours. This discrete transform can be connected to curvelet-like structures in the continuous domain. Hence, the contourlet transform [24] can be seen as a discrete form of a particular curvelet transform. Curvelet constructions require a rotation operation and correspond to a partition of the 2D frequency plane based on polar coordinates. This property makes the curvelet idea simple in the continuous domain but causes problems in the implementation for discrete images. In particular, approaching critical sampling seems difficult in discretized constructions of curvelets. For contourlets, it is easy to implement the critically sampling. There exists an orthogonal version of contourlet transform that is faster than current discrete curvelet algorithms [8]. The directional filter bank, as a key component of contourlets, has a convenient tree-structure, where aliasing is allowed to exist and will be canceled by carefully designed filters. Thus, the key difference between contourlets and curvelets is that the contourlet transform is directly defined on digital-friendly discrete rectangular grids. Unfortunately, contourlet functions have less clear directional geometry/features than curvelets (i.e., more oscillations along the needle-like elements) leading to artifacts in denoising and compression.

Surfacelets [46] are 3D extensions of the 2D contourlets that are obtained by a higher-dimensional directional filter bank and a multiscale pyramid. They can be used to efficiently capture and represent surface-like singularities in multidimensional volumetric data involving biomedical imaging, seismic imaging, video processing and computer vision. Surfacelets and the 3D curvelets that will be addressed in Section 6 aim at the same frequency partitioning, but the two transforms achieve this goal with two very different approaches as we described above in the 2D case. The surfacelet transform is less redundant than the 3D curvelet transform, and this advantage is payed by a certain loss of directional features.

Unlike curvelets, the shearlets [43, 35] form an affine system with a single generating mother shearlet function parameterized by a scaling, a shear, and a translation parameter, where the shear parameter captures the direction of singularities. It has been shown that both the curvelet and shearlet transforms have the same decay rates [13, 35]. Indeed, using the fast curvelet transform based on transition to Cartesian arrays, described in Section 5.2, the discrete implementations of the two transforms are very similar [8, 29].

The bandlet transform [58, 61] is based on adaptive techniques and has a good performance for images with textures beyond C^2-singularities, but it has to pay much higher computational cost for its adaptation.

In this paper we are not able to give a more detailed overview on all these approaches and refer to the given references for further information.

4 The continuous curvelet transform (CCT) in $\mathbb{R}^2$

In this section we describe the CCT developed in [11, 12]. We present the construction of a second generation curvelet system. In particular, we illustrate the necessary steps to achieve a complete curvelet frame.

We work in $\mathbb{R}^2$ and we denote with $x = (x_1, x_2)^T$ the spatial variable, and with $\xi = (\xi_1, \xi_2)^T$

the variable in frequency domain. Further, let $r = \sqrt{\xi_1^2 + \xi_2^2}$, $\omega = \arctan \frac{\xi_1}{\xi_2}$ be the polar coordinates in frequency domain.

How can the elements of the continuous curvelet transform be constructed? What properties do they have in time and in frequency domain? Why is the continuous curvelet transform of interest? Let us consider these questions more closely.

4.1 Window functions

For constructing the curvelet functions we first need to define special window functions that satisfy certain admissibility conditions. We present an explicit example that is representative for all possible choices of window functions being the fundament of the curvelet construction. For this purpose, let us consider the scaled Meyer windows (see [21], p. 137)

$$V(t) = \begin{cases} 1 & |t| \leq 1/3, \\ \cos[\frac{\pi}{2}\nu(3|t|-1)] & 1/3 \leq |t| \leq 2/3, \\ 0 & \text{else}, \end{cases}$$

$$W(r) = \begin{cases} \cos[\frac{\pi}{2}\nu(5-6r)] & 2/3 \leq r \leq 5/6, \\ 1 & 5/6 \leq r \leq 4/3, \\ \cos[\frac{\pi}{2}\nu(3r-4)] & 4/3 \leq r \leq 5/3, \\ 0 & \text{else}, \end{cases}$$

where ν is a smooth function satisfying

$$\nu(x) = \begin{cases} 0 & x \leq 0, \\ 1 & x \geq 1, \end{cases} \qquad \nu(x) + \nu(1-x) = 1, \quad x \in \mathbb{R}.$$

For the simple case $\nu(x) = x$ in $[0,1]$, the window functions $V(t)$ and $W(r)$ are plotted in Figure 2. In order to obtain smoother functions W and V, we need to take smoother functions ν. We may use the polynomials $\nu(x) = 3x^2 - 2x^3$ or $\nu(x) = 5x^3 - 5x^4 + x^5$ in $[0,1]$, such that ν is in $C^1(\mathbb{R})$ or in $C^2(\mathbb{R})$. As we will see later, the curvelet elements will be obtained as the inverse Fourier transform of a suitable product of the above windows. Therefore, the smoothness of V and W will ensure a faster decay of the curvelet elements in time domain. An example of an arbitrarily smooth window ν is given by

$$\nu(x) = \begin{cases} 0 & x \leq 0, \\ \frac{s(x-1)}{s(x-1)+s(x)} & 0 < x < 1, \\ 1 & x \geq 1, \end{cases} \qquad \text{with} \quad s(x) = e^{-\left(\frac{1}{(1+x)^2} + \frac{1}{(1-x)^2}\right)}.$$

The above two functions $V(t)$ and $W(r)$ satisfy the conditions

$$\sum_{l=-\infty}^{\infty} V^2(t-l) = 1, \quad t \in \mathbb{R}, \tag{1}$$

$$\sum_{j=-\infty}^{\infty} W^2(2^j r) = 1, \quad r > 0. \tag{2}$$

For $V(t)$, this can be simply observed. Since supp $V \subset [-1,1]$, for a fixed $t \in \mathbb{R}$ the above sum has only two nonvanishing terms, and for $t \in [1/3, 2/3]$ we find with the substitution

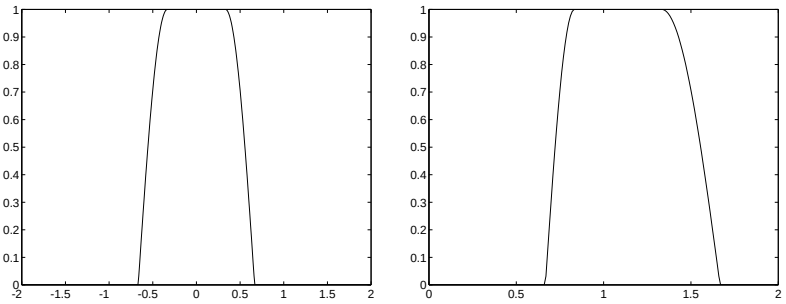


Figure 2: Plot of the windows $V(t)$ (left) and $W(r)$ (right).

$s = 3t - 1$

$$\begin{aligned}\sum_{l=-\infty}^{\infty} V^2(t-l) &= V^2(t) + V^2(t-1) = \cos^2[\frac{\pi}{2}\nu(3t-1)] + \cos^2[\frac{\pi}{2}\nu(3|t-1|-1)] \\ &= \cos^2[\frac{\pi}{2}\nu(s)] + \cos^2[\frac{\pi}{2}\nu(1-s)] = \cos^2[\frac{\pi}{2}\nu(s)] + \cos^2[\frac{\pi}{2}(1-\nu(s))] \\ &= \cos^2[\frac{\pi}{2}\nu(s)] + \sin^2[\frac{\pi}{2}\nu(s)] = 1.\end{aligned}$$

Similarly, formula (2) can be shown for W. We have supp $W(2^j\cdot) \subset [2^{-j-1}, 2^{-j+1}]$, and for a fixed $r \in [1/2, 1]$, it follows that $\sum_{j=-\infty}^{\infty} W^2(2^{-j}r) = W^2(r) + W^2(2r)$. Applying the definition of W we find in the interval $[1/2, 1]$

$$W^2(r) + W^2(2r) = \begin{cases} 1 & 1/2 \le r \le 2/3, \\ \cos^2[\frac{\pi}{2}(\nu(6r-4)] + \cos^2[\frac{\pi}{2}(\nu(5-6r)] & 2/3 \le r \le 5/6, \\ 1 & 5/6 \le r \le 1, \end{cases}$$

where we can show $\cos^2[\frac{\pi}{2}(\nu(6r-4)] + \cos^2[\frac{\pi}{2}(\nu(5-6r)] = 1$, similarly as before. Moreover, the above two windows also satisfy the normalization conditions

$$\int_0^{\infty} W^2(r)\,\frac{dr}{r} = \ln 2, \tag{3}$$

$$\int_{-1}^{1} V^2(t)\,dt = 1. \tag{4}$$

Indeed, we observe that

$$1 = \sum_{l=-\infty}^{\infty} V^2(t-l) = \int_0^1 \sum_{l=-\infty}^{\infty} V^2(t-l)\,dt = \int_{-\infty}^{\infty} V^2(t)\,dt$$

and substituting $r \in (0, \infty)$ by 2^t with $t \in (-\infty, \infty)$ we find

$$\begin{aligned}1 = \sum_{j=-\infty}^{\infty} W^2(2^j r) &= \sum_{j=-\infty}^{\infty} W^2(2^{j+t}) = \int_0^1 \sum_{j=-\infty}^{\infty} W^2(2^{j+t})\,dt \\ &= \sum_{j=-\infty}^{\infty} \frac{1}{\ln 2} \int_{2^j}^{2^{j+1}} W^2(s)\frac{ds}{s} = \frac{1}{\ln 2}\int_0^{\infty} W^2(s)\,\frac{ds}{s},\end{aligned}$$

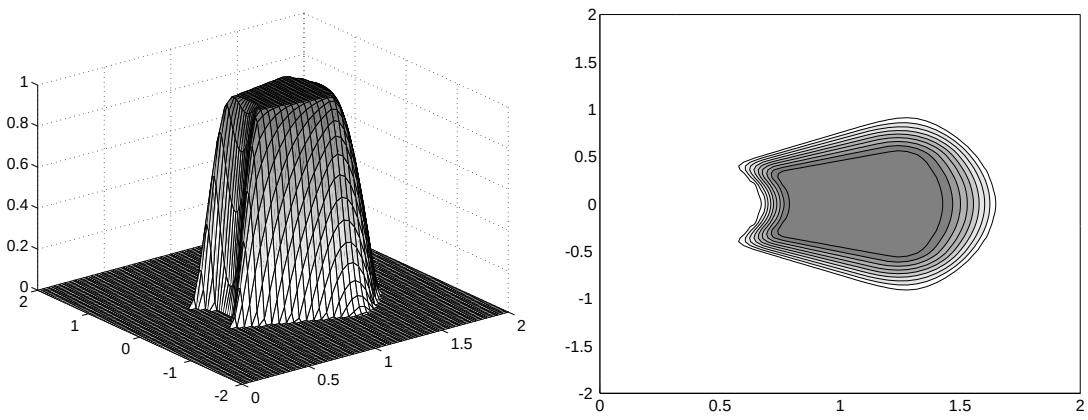


Figure 3: Window $U_1(\xi)$ (left) and its support (right).

Hence, the window functions V and W satisfy the conditions (3) and (4) that are called *admissibility conditions* for V and W for the continuous curvelet transform. Analogously, the conditions (1) and (2) are necessary admissibility conditions for the discrete curvelet transform.

4.2 System of curvelet functions

Assume now that the two window functions $V(t)$ and $W(r)$ satisfy the admissibility conditions (1)-(4). These windows will be used to construct a family of complex-valued waveforms with three parameters,

the *scale* $a \in (0, 1]$,

the *location* $b \in \mathbb{R}^2$,

and the *orientation* $\theta \in [0, 2\pi)$.

Let the Fourier transform for a function $f \in L^2(\mathbb{R}^2)$ be defined by

$$\widehat{f}(\xi) := \frac{1}{2\pi} \int_{\mathbb{R}^2} f(x)\, e^{-i\langle x,\xi\rangle}\, dx.$$

Using the polar coordinates (r, ω) in frequency domain, we now define the a-scaled window

$$U_a(r, w) := a^{3/4} W(ar)\, V\Big(\frac{\omega}{\sqrt{a}}\Big)$$

for some a with $0 < a \leq 1$. The support of U_a is a polar wedge depending on the supports of W and V, see Figure 3 for $a = 1$, and Figure 4 for $a = 1/2$ and $a = 1/8$. Comparing these supports in Figure 4, we nicely see the effect of scaling. While supp $W(a\cdot) \subset [1/2a, 2/a]$ is growing for decreasing $a \in (0, 1]$, the support $[-2\sqrt{a}/3, 2\sqrt{a}/3]$ of $V(\cdot/\sqrt{a})$ gets smaller, such that the wedges U_a become longer and thinner for decreasing a.

The window U_a is now applied for building curvelet functions as follows. Let $\varphi_{a,0,0} \in L^2(\mathbb{R}^2)$ be given by its Fourier transform

$$\widehat{\varphi}_{a,0,0}(\xi) := U_a(\xi),$$

and let the curvelet family be generated by translation and rotation of the basic element $\varphi_{a,0,0}$,

$$\varphi_{a,b,\theta}(x) := \varphi_{a,0,0}(R_\theta(x-b)), \tag{5}$$

with the translation $b \in \mathbb{R}^2$, and where $R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is the 2×2 rotation matrix with angle θ.

Observe that the rotation in spatial domain with an angle θ corresponds to a rotation in frequency domain with θ since

$$\widehat{\varphi}_{a,b,\theta}(\xi) = e^{-i\langle b,\xi\rangle}\, \widehat{\varphi}_{a,0,0}(R_\theta\xi) = e^{-i\langle b,\xi\rangle}\, U_a(R_\theta\xi).$$

Let us have a closer look at the properties of the functions $\varphi_{a,b,\theta}$.

Support in frequency domain

From the above equation we observe $\operatorname{supp}\widehat{\varphi}_{a,b,\theta} = \operatorname{supp} U_a(R_\theta\xi)$. In particular, the support of $\widehat{\varphi}_{a,b,\theta}$ does not at all depend on the translation parameter b. For $\theta = 0$, $\operatorname{supp}\widehat{\varphi}_{a,b,0}$ can be seen in Figures 3 and 4 for different a. For $\theta \in [0,\, 2\pi)$, this support is rotated by θ (clockwise). The curvelet functions $\varphi_{a,b,\theta}$ are complex functions. One is able to construct a real counterpart by replacing the function $W(ar)$ in the window $U_a(r,w)$ by $W(r)+W(-r)$; in this case $\widehat{\varphi}_{a,b,\theta}$ is supported on two polar wedges being symmetric with respect to zero.

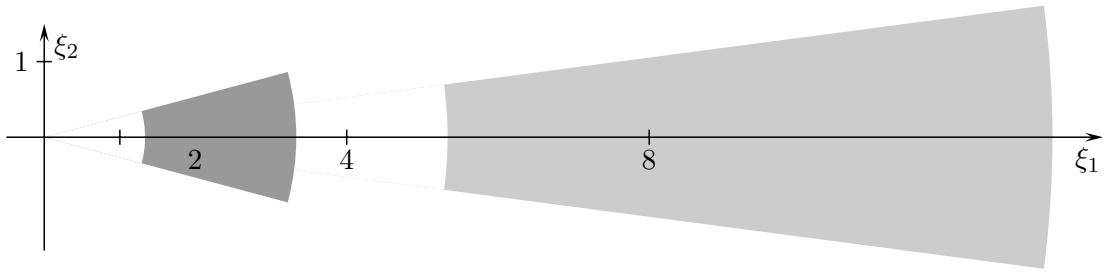


Figure 4: Supports of the windows $U_{1/2}(\xi)$ (grey) $U_{1/8}(\xi)$ (light grey).

Support in time domain and oscillation properties

In time domain, things are more involved. Since $\widehat{\varphi}_{a,b,\theta}$ has compact support, the curvelet function $\varphi_{a,b,\theta}$ cannot have compact support in time domain. From Fourier analysis, one knows that the decay of $\varphi_{a,b,\theta}(x)$ for large $|x|$ depends on the smoothness of $\widehat{\varphi}_{a,b,\theta}$ in frequency domain. The smoother $\widehat{\varphi}_{a,b,\theta}$, the faster the decay.

By definition, $\widehat{\varphi}_{a,0,0}$ is supported away from the vertical axis $\xi_1 = 0$ but near the horizontal axis $\xi_2 = 0$, see Figure 4. Hence, for small $a \in (0, 1]$ the function $\varphi_{a,0,0}$ is less oscillatory in x_2-direction (with frequency about $\sqrt{a}$) and very oscillatory in x_1-direction (containing frequencies of about $1/2a$). The *essential support* of the amplitude spectrum of $\varphi_{a,0,0}$ is a rectangle of size $[-\pi/2a, \pi/2a] \times [-\pi/\sqrt{a}, \pi/\sqrt{a}]$, and the decay of $\varphi_{a,0,0}$ away from this rectangle essentially depends on the smoothness of U_a resp. the function ν used in the windows V and W. Now, from (5) we simply observe that the essential support of $\varphi_{a,b,\theta}$ is the rectangle rotated by the angle θ and translated by $R_\theta b$.

Vanishing moments

Observe that $\varphi_{a,b,\theta}$ are C^∞ complex functions. Since the compact support of $\widehat{\varphi}_{a,b,\theta}$ is away from $(0,0)$, the functions $\varphi_{a,b,\theta}$ have mean value zero. Moreover, $\widehat{\varphi}_{a,b,\theta}$ has infinitely many directional moments, i.e., for all bivariate polynomials $p(x)$ with $x \in \mathbb{R}^2$ of arbitrary degree and for all angles $\tilde{\theta}$ we find

$$\int_{\mathbb{R}^2} p(R_{\tilde{\theta}} x)\, \varphi_{a,b,\theta}(x)\, dx = 0,$$

where again $R_{\tilde{\theta}}$ denotes the rotation matrix with the angle $\tilde{\theta}$. This observation is a direct consequence of the smoothness of $\varphi_{a,b,\theta}$, see e.g. [21], page 153, for the univariate case.

4.3 Definition of the continuous curvelet transform

Applying the family of high frequency elements

$$\{\varphi_{a,b,\theta} : a \in (0,1],\, b \in \mathbb{R}^2,\, \theta \in [0,\, 2\pi)\}, \tag{6}$$

the *continuous curvelet transform* Γ_f of $f \in L^2(\mathbb{R}^2)$ is given as

$$\Gamma_f(a,b,\theta) := \langle \varphi_{a,b,\theta},\, f\rangle = \int_{\mathbb{R}^2} \varphi_{a,b,\theta}(x)\, \overline{f(x)}\, dx,$$

i.e., one needs to compute the L^2 scalar products of a given function f with each curvelet element $\varphi_{a,b,\theta}$. Observe that we have bounded the scale a from above by $a = 1$, that means, low frequency functions are not contained in the system (6).

For functions $f \in L^2(\mathbb{R}^2)$ possessing a Fourier transform that vanishes for $|\xi| < 2$, the curvelet coefficients $\langle \varphi_{a,b,\theta},\, f\rangle$ contain the complete information about f, i.e., the continuous curvelet transform is invertible for these functions f, and there exists a reproducing formula to compute f from $\langle \varphi_{a,b,\theta},\, f\rangle$, $a \in (0,1], b \in \mathbb{R}^2$, $\theta \in [0, 2\pi)$, see [12]. One can extend the transform to low frequencies as follows. We consider

$$\widehat{\Psi}^2(\xi) := \frac{1}{\ln 2} \int_0^{|\xi|} |W(r)|^2\, \frac{dr}{r}.$$

By supp $W = [2/3, 5/3]$ and $\int_0^\infty W^2(r)/r\, dr = \ln 2$, we obviously have $\widehat{\Psi}^2(\xi) = 1$ for $|\xi| \geq 5/3$ and $\widehat{\Psi}^2(\xi) = 0$ for $|\xi| < 2/3$. Let now the function $\Phi \in L^2(\mathbb{R}^2)$ be given by its Fourier transform

$$\widehat{\Phi}(\xi) := \begin{cases} 1 & |\xi| \leq 1/2, \\ (1 - \widehat{\Psi}^2(\xi))^{1/2} & 1/2 < |\xi| < 2, \\ 0 & |\xi| \geq 2, \end{cases}$$

and let $\Phi_b(x) := \Phi(x - b)$, $b \in \mathbb{R}^2$ be the *father wavelets*.

Indeed, one can show that for all $f \in L^2(\mathbb{R}^2)$ it follows now the reproducing formula

$$f(x) = \int_{\mathbb{R}^2} \langle \Phi_b, f\rangle \Phi_b(x)\, db + \frac{1}{(\ln 2)} \int_0^{2\pi} \int_{\mathbb{R}^2} \int_0^1 \langle \varphi_{a,b,\theta},\, f\rangle\, \varphi_{a,b,\theta}(x)\, \frac{da}{a^{3/2}}\, \frac{db}{a^{1/2}}\, \frac{d\xi}{a},$$

i.e., the CCT is now invertible for all $\langle \varphi_{a,b,\theta},\, f\rangle$, see [12].

Since the CCT involves an infinite number of curvelet coefficients $\langle \varphi_{a,b,\theta},\, f\rangle$ (with the cardinal number of the continuum), it is not directly suitable for practical purposes. It has been successively applied to analyze different singularities of two-dimensional functions along lines and along smooth curves, see [11, 13]. For further applications we refer to Section 7. The CCT is closely related to the continuous transforms used by Hart Smith [68], and to the FBI and the Wave Packets transform, see [4, 11, 20].

5 The fast curvelet transform

For numerical computations we need to discretize the continuous curvelet transform, since we usually work with discrete data sets in applications. As shown in [12], a discrete version of the continuous curvelet transform can be derived by a suitable sampling at the range of scales, orientations and locations.

5.1 The discrete curvelet transform

We choose

- the scales $a_j := 2^{-j}$, $j \geq 0$;
- the equidistant sequence of rotation angles $\theta_{j,l}$,

$$\theta_{j,l} := \frac{\pi l\, 2^{-\lceil j/2\rceil}}{2} \qquad \text{with} \quad l = 0, 1, \ldots, 4 \cdot 2^{\lceil j/2\rceil} - 1;$$

  (Here $\lceil x\rceil$ denotes the smallest integer being greater than or equal to x.)
- the positions $b_k^{j,l} = b_{k_1,k_2}^{j,l} := R_{\theta_{j,l}}^{-1}\,(\frac{k_1}{2^j},\, \frac{k_2}{2^{j/2}})^T$, with $k_1, k_2 \in \mathbb{Z}$, and where R_θ denotes the rotation matrix with angle θ.

This choice will lead to a discrete curvelet system that forms a tight frame, i.e., every function $f \in L^2(\mathbb{R})$ will be representable by a curvelet series, and hence the discrete curvelet transform will be invertible.

For example, for $j = 0$ we consider the angles $\theta_{0,l} = \pi l/2$, $l = 0, 1, 2, 3$ and the positions $\{b_k^{0,l}\}_{k\in\mathbb{Z},l=0,1,2,3} = \mathbb{Z}^2$. For $j = 4$, the angles $\theta_{4,l} = \pi l/8$, $l = 0, \ldots, 15$ occur, and, depending on the angles $\theta_{4,l}$, eight different grids for translation are considered, where rectangles of size $1/16 \times 1/4$ are rotated by $\theta_{4,l}$, $l = 0, \ldots, 7$, see Figure 5. In particular, the choice of positions yields a parabolic scaling of the grids with the relationship length $\approx 2^{-j/2}$ and width $\approx 2^{-j}$.

As for the continuous transform, we define now the scaled windows in polar coordinates

$$U_j(r,\omega) := 2^{-3j/4}\, W(2^{-j}r)\, V\left(\frac{2\cdot 2^{\lceil j/2\rceil}\omega}{\pi}\right) = 2^{-3j/4}\, W(2^{-j}r)\, V\left(\frac{\omega}{\theta_{j,1}}\right), \qquad j \in \mathbb{N}_0. \tag{7}$$

The basic curvelet is defined by

$$\widehat{\phi}_{j,0,0}(\xi) := U_j(\xi),$$

and the family of curvelet functions is given by

$$\phi_{j,k,l}(x) := \phi_{j,0,0}(R_{\theta_{j,l}}(x - b_k^{j,l})). \tag{8}$$

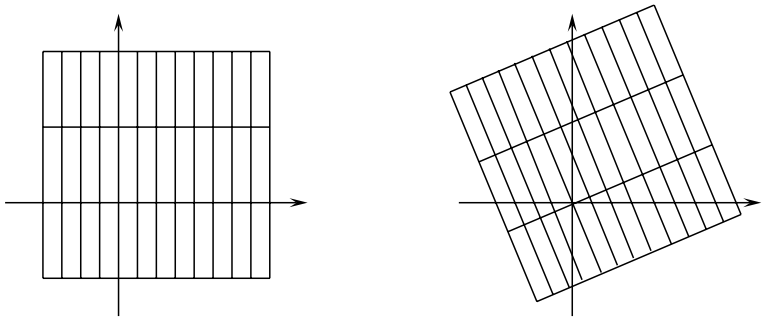


Figure 5: Grid for $\theta_{4,0} = 0$ and for $\theta_{4,1} = \pi/8$.

In frequency domain, the curvelet functions

$$\widehat{\phi}_{j,k,l}(\xi) = e^{-i\langle b_k^{j,l},\xi\rangle}\, U_j(R_{\theta_{j,l}}\xi) = e^{-i\langle b_k^{j,l},\xi\rangle}\, 2^{-3j/4}\, W(2^{-j}r) V\Big(\frac{\omega+\theta_{j,l}}{\theta_{j,1}}\Big)$$

are supported inside the polar wedge with radius $2^{j-1} \le r \le 2^{j+1}$ and angle $\frac{2^{-\lceil j/2\rceil}\pi(-1-l)}{2} < \omega < \frac{2^{-\lceil j/2\rceil}\pi(1-l)}{2}$. The support of $\widehat{\phi}_{j,k,l}$ does not depend on the position $b_k^{j,l}$. For example, $\widehat{\phi}_{2,k,l}(r,\omega)$ is supported inside the wedge with $2 \le r \le 8$ and $\frac{(-1-l)\pi}{4} \le \omega \le \frac{(1-l)\pi}{4}$, $l = 0, \ldots, 7$, see Figure 6. (Here we have used supp $V \subset [-1,1]$ and supp $W \subset [1/2,2]$.)

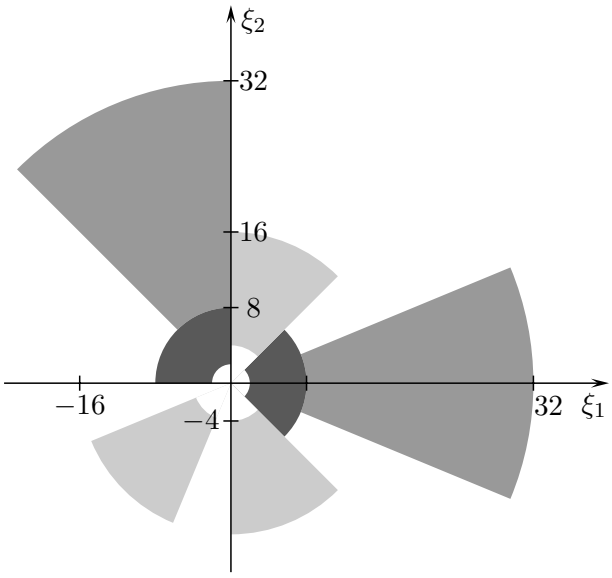


Figure 6: Maximal supports of $\widehat{\phi}_{2,k,0}$ and $\widehat{\phi}_{2,k,5}$(dark grey); of $\widehat{\phi}_{3,k,3}$, $\widehat{\phi}_{3,k,6}$ and $\widehat{\phi}_{3,k,13}$ (light grey); and of $\widehat{\phi}_{4,k,0}$ and $\widehat{\phi}_{4,k,11}$ (grey).

We again need some coarse scale curvelet elements for low frequencies and take here

$$\phi_{-1,k,0}(x) := \phi_{-1}(x-k), \qquad k \in \mathbb{Z}^2,$$

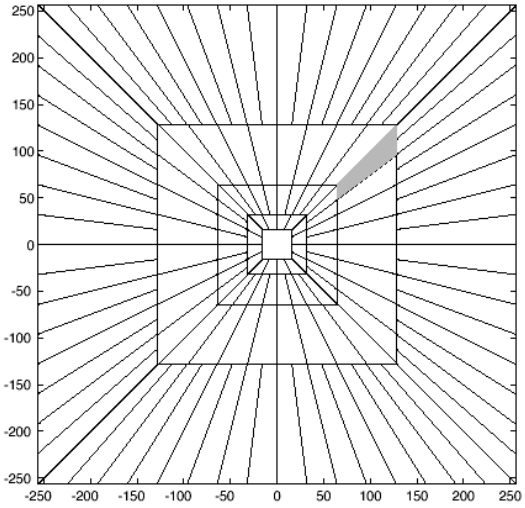


Figure 7: Discrete curvelet tiling with parabolic pseudo-polar support in the frequency plane.

where

$$\widehat{\phi}_{-1}(\xi) := W_0(|\xi|) \quad \text{with} \quad W_0(r)^2 := 1 - \sum_{j\geq 0} W(2^{-j}r)^2.$$

The system of curvelets

$$\{\phi_{-1,k,0} : k \in \mathbb{Z}^2\} \cup \{\phi_{j,k,l} : j \in \mathbb{N}_0,\, l = 0, \ldots, 4 \cdot 2^{\lceil j/2 \rceil} - 1,\, k = (k_1, k_2)^T \in \mathbb{Z}^2\}$$

satisfies a **tight frame property**. Every function $f \in L^2(\mathbb{R}^2)$ can be represented as a curvelet series, that means, the discrete curvelet transform is invertible. We have

$$f = \sum_{j,k,l} \langle f, \phi_{j,k,l} \rangle \, \phi_{j,k,l}, \tag{9}$$

and the Parseval identity

$$\sum_{j,k,l} |\langle f, \phi_{j,k,l} \rangle|^2 = \|f\|^2_{L^2(\mathbb{R}^2)}, \quad \forall f \in L^2(\mathbb{R}^2)$$

holds. For a proof we refer to [12]. The terms $c_{j,k,l}(f) := \langle f, \phi_{j,k,l} \rangle$ are called **curvelet coefficients**. In particular, we obtain by Plancherel's Theorem for $j \geq 0$

$$\begin{aligned} c_{j,k,l}(f) &:= \int_{\mathbb{R}^2} f(x)\, \overline{\phi_{j,k,l}(x)}\, dx = \int_{\mathbb{R}^2} \widehat{f}(\xi)\, \overline{\widehat{\phi}_{j,k,l}(\xi)}\, d\xi \\ &= \int_{\mathbb{R}^2} \widehat{f}(\xi)\, U_j(R_{\theta_{j,l}}\xi)\, e^{i\langle b^{j,l}_k, \xi\rangle}\, d\xi. \end{aligned} \tag{10}$$

5.2 Transition to Cartesian arrays

In practical implementations one would like to have Cartesian arrays instead of the polar tiling of the frequency plane. Cartesian coronae are based on concentric squares (instead of circles) and shears, see Figure 7. Therefore, a construction of window functions on trapezoids instead

of polar wedges is desirable. Hence, we need to adapt the discrete curvelet system in Subsection 5.1 suitably. Let us remark that the frequency tiling into shears, as given in Figure 7, has been similarly used for the construction of contourlets [24] by a pyramidal directional filter bank. However, the tiling for the contourlet transform is slightly more flexible by allowing that the number of directions need not to be doubled at each scale, see [24].

We consider now the window function

$$\gamma(r) := \begin{cases} 1 & |r| \le 1/2, \\ \cos\frac{(2|r|-1)\pi}{2} & 1/2 < |r| \le 1, \\ 0 & \text{else}, \end{cases}$$

such that supp $\gamma = [-1, 1]$. As in [8], we replace the window $W_j(r) := W(2^{-j}r)$ in Subsection 4.1 by a window of the form

$$\widetilde{W}_j(r) := \chi_{[0,\infty)}(r) \cdot \sqrt{\gamma^2(2^{-j-1}r) - \gamma^2(2^{-j}r)}, \qquad j \ge 0,\, r \ge 0.$$

Hence supp $\widetilde{W}_j = [2^{j-1}, 2^{j+1}]$. Further, for $\xi_1 > 0$, let $V_j(\xi) := V(2^{\lfloor j/2 \rfloor}\, \xi_2/\xi_1)$ with V in Subsection 4.1, such that the support of V_j is inside the cone

$$K_1 := \{(\xi_1, \xi_2) : \xi_1 > 0,\, \xi_2 \in [-2\xi_1/3, 2\xi_1/3]\}.$$

Here, $\lfloor x \rfloor$ denotes the largest integer being smaller than or equal to x. Now, the Cartesian window,

$$\widetilde{U}_j(\xi) := 2^{-3j/4}\, \widetilde{W}_j(\xi_1)\, V_j(\xi) = 2^{-3j/4} \widetilde{W}_j(\xi_1)\, V\left(\frac{2^{\lfloor j/2 \rfloor}\xi_2}{\xi_1}\right)$$

can be defined, being analogous to U_j in (7) and determining the frequencies in the trapezoid

$$\{(\xi_1, \xi_2) : 2^{j-1} \le \xi_1 \le 2^{j+1},\ -2^{-\lfloor j/2 \rfloor} \cdot \frac{2}{3} \le \xi_2/\xi_1 \le 2^{-\lfloor j/2 \rfloor} \cdot \frac{2}{3}\}.$$

The window $\widetilde{U}_0$ is presented in Figure 8. It is the Cartesian equivalent of U_0 ($= U_a$ with $a = 1$ in the notation of Subsection 4.1) in Figure 3.

Next, instead of equidistant angles, we define a set of equispaced slopes in the eastern cone $K = \{(\xi_1, \xi_2)^T : \xi_1 > 0,\, -\xi_1 < \xi_2 \le \xi_1\}$,

$$\tan\theta_{j,l} := l\, 2^{-\lfloor j/2 \rfloor}, \qquad l = -2^{\lfloor j/2 \rfloor} + 1, \ldots, 2^{\lfloor j/2 \rfloor} - 1.$$

Observe that the angles $\theta_{j,l}$, which range between $-\pi/4$ and $\pi/4$, are not equispaced here, while the slopes are.

Now, let the curvelet-like functions be given by

$$\begin{aligned} \widehat{\widetilde{\phi}}_{j,0,0} &:= \widetilde{U}_j(\xi), \\ \widetilde{\phi}_{j,k,l}(x) &:= \widetilde{\phi}_{j,0,0}\Big(S^T_{\theta_{j,l}}(x - \widetilde{b}^{j,l}_k)\Big), \end{aligned} \tag{11}$$

being the Cartesian counterpart of $\phi_{j,k,l}$ in (8), with the shear matrix

$$S_\theta = \begin{pmatrix} 1 & 0 \\ -\tan\theta & 1 \end{pmatrix},$$

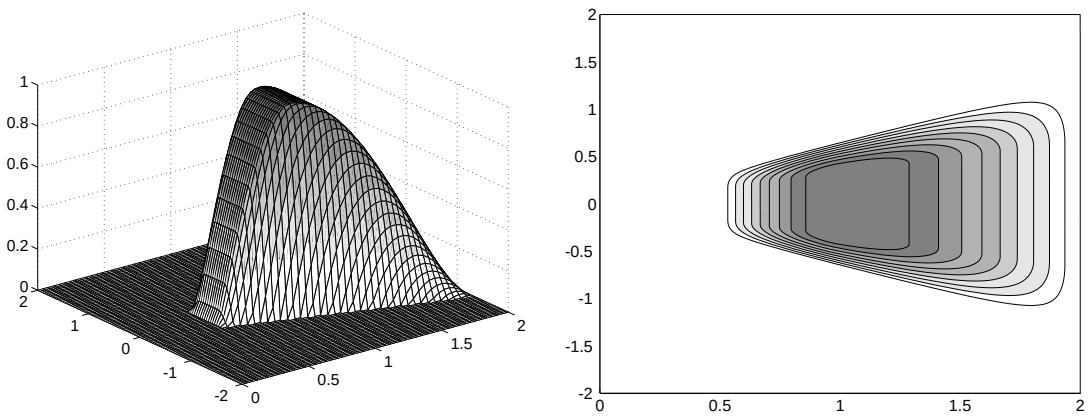


Figure 8: Window $\tilde{U}_0(\xi)$ (left) and its support (right).

and where $\widetilde{b}_k^{j,l} := S_{\theta_{j,l}}^{-T}(k_1\,2^{-j},\, k_2\,2^{-\lfloor j/2\rfloor}) =: S_{\theta_{j,l}}^{-T}\,k_j$. Let us have a closer look at the functions $\widetilde{\phi}_{j,k,l}$. The Fourier transform gives by $S_{\theta_{j,l}}^{-1}\xi = (\xi_1, \xi_1\,\tan\theta_{j,l} + \xi_2)^T$

$$\begin{aligned}\widehat{\widetilde{\phi}}_{j,k,l}(\xi) &= e^{-i\langle \widetilde{b}_k^{j,l},\,\xi\rangle}\,\widehat{\widetilde{\phi}}_{j,0,0}(S_{\theta_{j,l}}^{-1}\xi) = e^{-i\langle \widetilde{b}_k^{j,l},\,\xi\rangle}\,\widetilde{U}_j(S_{\theta_{j,l}}^{-1}\xi)\\ &= e^{-i\langle \widetilde{b}_k^{j,l},\,\xi\rangle}\,2^{-3j/4}\widetilde{W}_j(\xi_1)\,V_j(S_{\theta_{j,l}}^{-1}\xi) = e^{-i\langle \widetilde{b}_k^{j,l},\,\xi\rangle}\,2^{-3j/4}\widetilde{W}_j(\xi_1)\,V(2^{\lfloor j/2\rfloor}\xi_2/\xi_1 + l).\end{aligned}$$

Hence, $\widehat{\widetilde{\phi}}_{j,k,l}$ is compactly supported on sheared trapezoids.

Let us for example examine $\widehat{\widetilde{\phi}}_{4,k,l}$. For $j = 4$, we consider the angles $\tan\theta_{4,l} = l/4$, $l = -3, \ldots, 3$. The support of $\widehat{\widetilde{\phi}}_{j,k,0}$ is symmetric with respect to the ξ_1 axis, and for $j = 4$ we have

$$\operatorname{supp}\widehat{\widetilde{\phi}}_{4,k,0} = \{(\xi_1,\xi_2)^T :\, 8 \le \xi_1 \le 32,\, -\frac{1}{6} \le \frac{\xi_1}{\xi_2} \le \frac{1}{6}\}.$$

The supports of $\widehat{\widetilde{\phi}}_{4,k,l}$ with $l = -3, \ldots, 3$ are now sheared versions of this trapezoid, see Figure 9.

The set of curvelets $\widetilde{\phi}_{j,k,l}$ in (11) needs to be completed by symmetry and by rotation by $\pm\pi/2$ radians in order to obtain the whole family. Moreover, as we can also see in Figure 9, we need suitable "corner elements" connecting the four cones (north, west, south, east). In [8], it is suggested to take a corner element as the sum of two half-part sheared curvelet functions of neighboring cones as indicated in Figure 9 (left). More precisely, we take

$$\widehat{\widetilde{\phi}}_{j,0,-2^{\lfloor j/2\rfloor}}(\xi) := 2^{-3j/4}\,W_j(\xi_1)\,V^h\Big(2^{\lfloor j/2\rfloor}\Big(1 + \frac{\xi_2}{\xi_1}\Big)\Big) + 2^{-3j/4}\,W_j(\xi_2)\,V^h\Big(2^{\lfloor j/2\rfloor}\Big(1 - \frac{\xi_1}{\xi_2}\Big)\Big),$$

where the window function $V^h(t) := V(t)\,\chi_{[-1,0]}(t)$ is the left part of the window V and has only the half support $[-2/3,\, 0]$.

Finally, the coarse curvelet elements for low frequencies are needed, and we take here

$$\widetilde{\phi}_{-1,k,0}(x) := \widetilde{\phi}_{-1}(x - k), \qquad k \in \mathbb{Z}^2$$

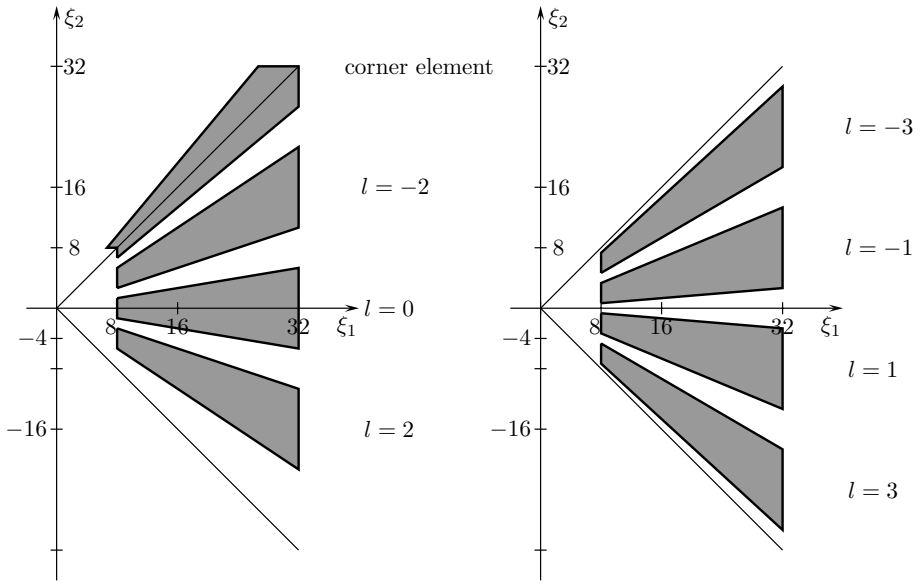


Figure 9: Supports of the functions $\widehat{\widetilde{\phi}}_{4,k,l}$ for $l = -3, \ldots, 3$, and one corner element.

with $\widehat{\widetilde{\phi}}_{-1}(\xi) := \gamma(\xi_1)\,\gamma(\xi_2)$. For this construction of curvelet-like elements one can show that

$$\widehat{\widetilde{\phi}}_{-1}(\xi) + \sum_{j=0}^{\infty} \sum_{l=-2^{\lfloor j/2 \rfloor}}^{2^{\lfloor j/2 \rfloor}} 2^{3j/4}\, \widehat{\widetilde{\phi}}_{j,0,l}(\xi) = 1$$

for all ξ in the eastern cone $K = \{(\xi_1, \xi_2)^T : \xi_1 > 0,\ \xi_2 \in [-\xi_1, \xi_1]\}$, where we have taken also the two corner elements in the sum. Similarly, this assertion is true for the rotated functions in the other three cones.

5.3 The algorithm

We find the Cartesian counterpart of the coefficients in (10) by

$$\begin{aligned} \widetilde{c}_{j,k,l}(f) &= \langle f, \widetilde{\phi}_{j,k,l} \rangle = \int_{\mathbb{R}^2} \widehat{f}(\xi)\, \widetilde{U}_j(S^{-1}_{\theta_{j,l}}\xi)\, e^{i\langle \tilde{b}^{j,l}_k, \xi \rangle} d\xi \\ &= \int_{\mathbb{R}^2} \widehat{f}(S_{\theta_{j,l}}\xi)\, \widetilde{U}_j(\xi)\, e^{i\langle k_j, \xi \rangle} d\xi \end{aligned} \tag{12}$$

with $k_j = (k_1 2^{-j},\, k_2 2^{-\lfloor j/2 \rfloor})^T$, $(k_1, k_2)^T \in \mathbb{Z}^2$.

The forward and the inverse Fast Discrete Curvelet Transform as presented in [8] and in CurveLab have a computational cost of $\mathcal{O}(N^2 \log N)$ for an $(N \times N)$ image, see http:curvelab.org with a collection of Matlab and C++ programs. The redundancy of that curvelet transform implementation is about 2.8 when wavelets are chosen at the finest scale, and 7.2 otherwise (see e.g. [8]). Using formula (12), the forward algorithm has roughly the following form.

Algorithm

1. **Compute the Fourier transform of f by means of a $2D$ FFT.**
   Let f be given by its samples $f(\frac{n_1}{N}, \frac{n_2}{N})$, $n_1, n_2 = 0, \ldots, N-1$, where N is of the form $N = 2^J$, $J \in \mathbb{N}$. Suppose, that f can be approximated by a linear combination of bivariate hat functions. Let $\tilde{s}(x) = s(x_1)\, s(x_2)$ with $s(x_1) := (1 - |x_1|)\, \chi_{[-1,1]}(x_1)$ and

$$f(x) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} f\Big(\frac{n_1}{N}, \frac{n_2}{N}\Big)\, \tilde{s}(Nx_1 - n_1,\, Nx_2 - n_2).$$

   With $\widehat{\tilde{s}}(\xi) = (\operatorname{sinc} \xi_1/2)^2\, (\operatorname{sinc} \xi_2/2)^2$ it follows that

$$\widehat{f}(\xi) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} f\Big(\frac{n_1}{N}, \frac{n_2}{N}\Big)\, e^{-i(n_1\xi_1 + n_2\xi_2)/N}\, \widehat{\tilde{s}}\Big(\frac{\xi}{N}\Big),$$

   and the 2D FFT of length N gives us the samples $\widehat{f}(2\pi n_1, 2\pi n_2)$, $n_1, n_2 = -\frac{N}{2}, \ldots, \frac{N}{2} - 1$.

2. **Compute $\widehat{f}(S_{\theta_{j,l}}\xi)$ by interpolation.**
   Fix the scales to be considered, say $j_0 \le j \le J$. The support of $\widetilde{U}_j$ is contained in the rectangle $R_j = [2^{j-1}, 2^{j+1}] \times [-2^{\lfloor j/2 \rfloor}, 2^{\lfloor j/2 \rfloor}]$. For each pair (j, l) compute now $\widehat{f}(2\pi n_1, 2\pi n_2 - 2\pi n_1 \tan\theta_{j,l})$ for $2\pi(n_1, n_2) \in R_j$.

3. **Compute the product $\widehat{f}(S_{\theta_{j,l}}\xi)\, \widetilde{U}_j(\xi)$.**
   For each pair (j, l) compute the product $\widehat{f}(2\pi n_1, 2\pi n_2 - 2\pi n_1 \tan\theta_{j,l})\, \widetilde{U}_j(2\pi n_1, 2\pi n_2)$.

4. **Apply the inverse $2D$ FFT** in order to obtain the discrete coefficients $\widetilde{c}^D_{j,k,l}(f)$ that are an approximation of the coefficients in (12).

For the inverse curvelet transform, one applies the algorithm in each step in reversed order. Observe that in the second step a suitable approximation scheme has to be applied in the forward transform and in the inverse transform.

6 3D Curvelet transform

For three-dimensional data, a generalization to three-dimensional multiscale geometric methods is of great interest. So far, only a few papers have been concerned with applications of the three-dimensional curvelet transform to 3D turbulence [2, 53] and 3D seismic processing [59].

In this section, we develop the idea of the three-dimensional curvelet transform on Cartesian arrays analogously as done in Section 5.2 for the two-dimensional case. Using polar coordinates, we consider curvelet functions being supported on sheared truncated pyramids instead of sheared trapezoids. In contrast to [77], we introduce three-dimensional shear matrices. The three-dimensional curvelet functions depend on four indices instead of three; the scale, the position and two angles.

In $\mathbb{R}^3$ we denote with $x = (x_1, x_2, x_3)^T$ the spatial variable and with $\xi = (\xi_1, \xi_2, \xi_3)^T$ the variable in frequency domain. Again, we use polar coordinates $(r,\, \omega_1,\, \omega_2)^T$ with $r = \sqrt{\xi_1^2 + \xi_2^2 + \xi_3^2}$, $\sin\omega_1 = \xi_2/\sqrt{\xi_1^2 + \xi_2^2}$, $\omega_1 \in [0, 2\pi)$, and $\cos\omega_2 = \xi_3/r$, $\omega_2 \in [0, \pi]$.

As in the two-dimensional case, we define a pair of smooth, nonnegative, real-valued window functions. We consider the same window $\widetilde{W}_j(r)$ as in Section 5.2,

$$\widetilde{W}_j(r) := \chi_{[0,\infty)}(r) \cdot \sqrt{\gamma^2(2^{-j-1}r) - \gamma^2(2^{-j}r)}, \qquad j \geq 0,\, r \geq 0,$$

such that supp $\widetilde{W}_j = [2^{j-1}, 2^{j+1}]$. Further, let

$$\widetilde{V}_j(\xi) := V\Big(2^{\lfloor j/2 \rfloor}\frac{\xi_2}{\xi_1}\Big)\, V\Big(2^{\lfloor j/2 \rfloor}\frac{\xi_3}{\xi_1}\Big)$$

with the window V as in Subsection 4.1. Obviously, $\widetilde{V}_j$ is supported in the cone $\{(\xi_1,\, \xi_2,\, \xi_3) : \xi_1 > 0,\, \xi_2 \in [-\frac{2}{3}\xi_1, \frac{2}{3}\xi_1],\, \xi_3 \in [-\frac{2}{3}\xi_1, \frac{2}{3}\xi_1]\}$. Disregarding the normalization constant, let the Cartesian window $\widetilde{U}_j$ be defined by

$$\widetilde{U}_j(\xi) = \widetilde{U}_j(\xi_1, \xi_2, \xi_3) = \widetilde{W}_j(\xi_1)\, \widetilde{V}_j(\xi) = \widetilde{W}_j(\xi_1)\, V\Big(2^{\lfloor j/2 \rfloor}\frac{\xi_2}{\xi_1}\Big)\, V\Big(2^{\lfloor j/2 \rfloor}\frac{\xi_3}{\xi_1}\Big),$$

determining the frequencies in the truncated pyramid

$$\{(\xi_1, \xi_2, \xi_3)^T : 2^{j-1} \leq \xi_1 \leq 2^{j+1},\, -\tfrac{2}{3}2^{-\lfloor j/2 \rfloor} \leq \tfrac{\xi_2}{\xi_1} \leq \tfrac{2}{3}2^{-\lfloor j/2 \rfloor},\, -\tfrac{2}{3}2^{-\lfloor j/2 \rfloor} \leq \tfrac{\xi_3}{\xi_1} \leq \tfrac{2}{3}2^{-\lfloor j/2 \rfloor}\}.$$

Every Cartesian corona has six components, one for each face of the unit cube. Let us consider only the cone

$$K = \left\{(\xi_1, \xi_2, \xi_3) :\, \xi_1 > 0,\, -1 \leq \frac{\xi_2}{\xi_1} < 1,\, -1 \leq \frac{\xi_3}{\xi_1} < 1\right\}.$$

With the angles

$$\begin{aligned} \tan\theta_{j,l} &:= l\, 2^{-\lfloor j/2 \rfloor}, \quad l = -2^{\lfloor j/2 \rfloor} + 1, \ldots, 2^{\lfloor j/2 \rfloor} - 1, \\ \tan\vartheta_{j,m} &:= m\, 2^{-\lfloor j/2 \rfloor}, \quad m = -2^{\lfloor j/2 \rfloor} + 1, \ldots, 2^{\lfloor j/2 \rfloor} - 1, \end{aligned}$$

we define the three-dimensional shear matrix

$$S_{\theta_{j,l}, \vartheta_{j,m}} := \begin{pmatrix} 1 & 0 & 0 \\ -\tan\theta_{j,l} & 1 & 0 \\ -\tan\vartheta_{j,m} & 0 & 1 \end{pmatrix}$$

and the positions

$$\widetilde{b}_k^{j,l,m} := S^{-T}_{\theta_{j,l}, \vartheta_{j,m}}(k_1 2^{-j},\, k_2 2^{-\lfloor j/2 \rfloor}, k_3 2^{-\lfloor j/2 \rfloor})^T = S^{-T}_{\theta_{j,l}, \vartheta_{j,m}}\, k_j,$$

where $(k_1, k_2, k_3)^T \in \mathbb{Z}^3$. Now in the cone K the curvelet functions are given by

$$\begin{aligned} \widehat{\widetilde{\phi}}_{j,0,0,0}(\xi) &:= \widetilde{U}_j(\xi), \\ \widetilde{\phi}_{j,k,l,m} &:= \widetilde{\phi}_{j,0,0,0}(S^T_{\theta_{j,l}, \vartheta_{j,m}}(x - \widetilde{b}_k^{j,l,m})). \end{aligned}$$

This definition can be seen as a direct generalization of (11). The Fourier transform gives

$$\begin{aligned}\widehat{\widetilde{\phi}}_{j,k,l,m}(\xi) &= e^{-i\langle \widetilde{b}_k^{j,l,m},\xi\rangle}\,\widehat{\widetilde{\phi}}_{j,0,0,0}(\xi)\\ &= e^{-i\langle \widetilde{b}_k^{j,l,m},\xi\rangle}\,\widetilde{W}_j(\xi_1)\,V\Big(2^{\lfloor j/2\rfloor}\frac{\xi_2}{\xi_1}+l\Big)\,V\Big(2^{\lfloor j/2\rfloor}\frac{\xi_3}{\xi_1}+m\Big),\end{aligned}$$

i.e., the functions $\widehat{\widetilde{\phi}}_{j,k,l,m}$ are compactly supported on sheared truncated pyramids (see e.g. Figure 3 in [77]). As in the two-dimensional case one needs to take care for special boundary elements where two (or three) different cones touch each other.

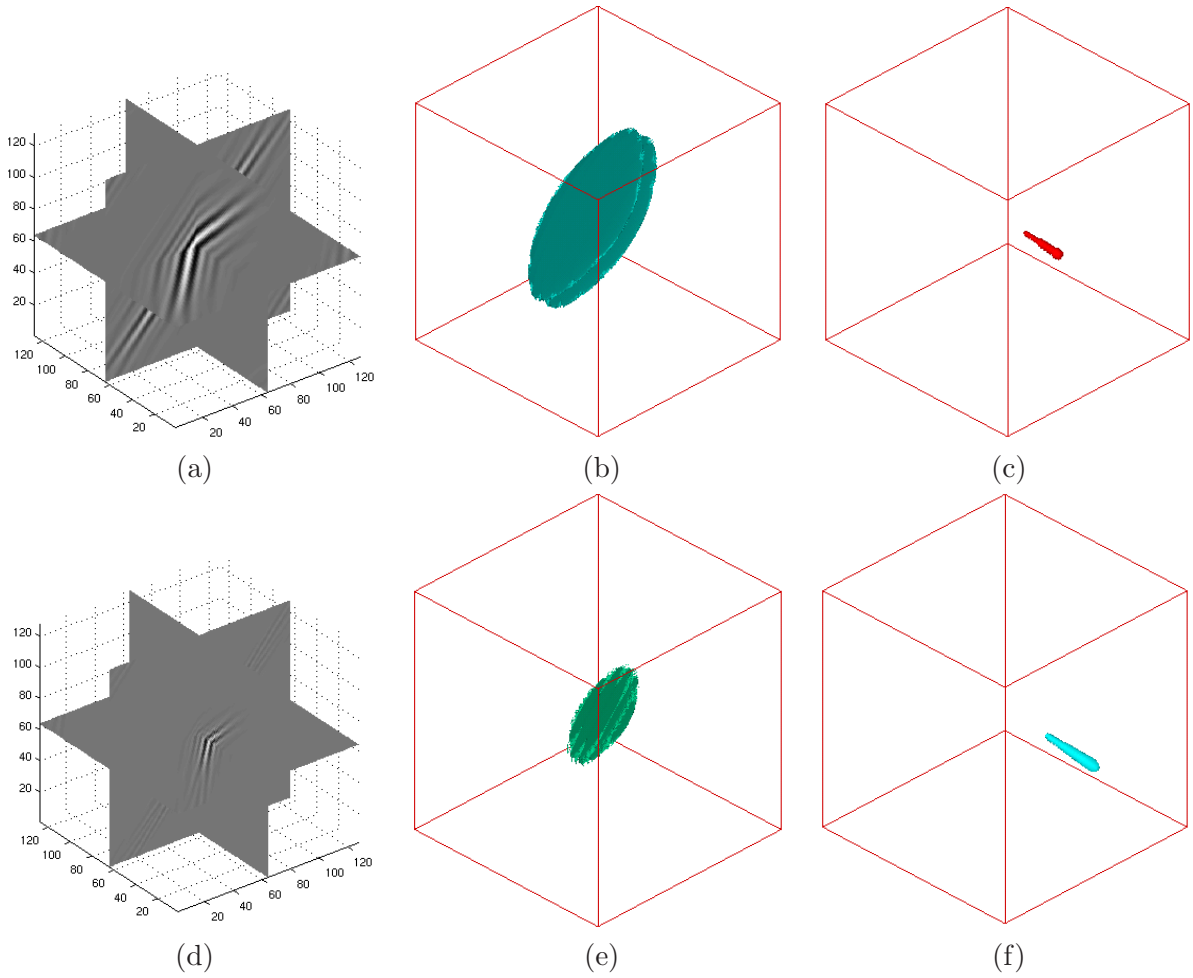


Figure 10: An element of 3D curvelets at a coarser scale (upper row) and finer scale (lower row) is shown in three cross-sections (left column) and isosurface (middle column). The right column shows their frequency support. It can be clearly seen that the element with high resolution in the space domain has low resolution in the frequency domain.

Figure 10 shows some 3D curvelet elements. Observe that in the spatial domain, $\phi_{j,k,l,m}$ is of plate-like shape, which rapidly decays away from a 2^{-j} by $2^{-j/2}$ cross-section rectangle with center $\widetilde{b}_k^{(j,l,m)}$ and orientations $\theta_{j,l}$ (with respect to the horizontal axis in x) and $\vartheta_{j,m}$ (with respect to the vertical axis in x). The element is smooth within the plate but exhibits oscillating

decay in the normal direction of the plate. It obeys a parabolic scaling law between the thickness and length (thickness $\approx$ length^2) and directional sensitivity (Orientations $= 1/\sqrt{\text{scale}}$).

As always, for the coarsest scale we need a special construction, as e.g.

$$\widetilde{\phi}_{-1,k}(x) := \widetilde{\phi}_{-1}(x-k), \qquad k \in \mathbb{Z}^3$$

with $\widehat{\widetilde{\phi}}_{-1}(\xi) := \gamma(\xi_1)\,\gamma(\xi_2)\,\gamma(\xi_3)$, where γ is defined in Subsection 5.2.

Analogously as in (12), the curvelet coefficients are given by

$$\begin{aligned} \widetilde{c}_{j,k,l,m}(f) &= \langle f, \widetilde{\phi}_{j,k,l,m}\rangle = \int_{\mathbb{R}^3} \widehat{f}(\xi)\,\widetilde{U}_j(S^{-1}_{\theta_{j,l},\vartheta_{j,m}}\xi)\, e^{i\langle \widetilde{b}_k^{j,l,m},\xi\rangle}\, d\xi \\ &= \int_{\mathbb{R}^3} \widehat{f}(S_{\theta_{j,l},\vartheta_{j,m}}\xi)\,\widetilde{U}_j(\xi)\, e^{i\langle k_j,\xi\rangle} d\xi. \end{aligned}$$

An algorithm can now be derived similarly as in Subsection 5.3 for the two-dimensional case. The computational complexity of the three-dimensional discrete curvelet transform based on FFT algorithms is $\mathcal{O}(n^3 \log n)$ flops for $n \times n \times n$ data [8]. For further details we refer to [8] and [77].

7 Recent applications

In this section, we shall review applications of the curvelets in image processing, seismic exploration, fluid mechanics, solving of PDEs, and compressed sensing, to show their potential as an alternative to wavelet transforms in some scenarios.

7.1 Image processing

In 2002, the first-generation curvelet transform was applied for the first time to image denoising by Starck et al. [69], and by Candès and Guo [14]. The applications of the first-generation curvelets were extended to image contrast enhancement [71] and astronomical image representation [70] in 2003, and to fusion of satellite images [19] in 2005. After the effective second-generation curvelet transform [13] had been proposed in 2004, the applications of curvelets increased very fast in many fields involving image/video presentation, denoising and classification. For instance, Ma et al. applied the second-generation curvelets for motion estimation and video tracking of geophysical flows [50], surface characterization [47] and deblurring [48]. Ma and Plonka presented two different models for image denoising by combining the discrete curvelet transform with nonlinear diffusion schemes. In the first model [55], a curvelet shrinkage is applied to the noisy data, and the result is further processed by a projected total variation diffusion in order suppress pseudo-Gibbs artifacts. In the second model [64], a nonlinear reaction-diffusion equation is applied, where curvelet shrinkage is used for regularization of the diffusion process. Starck et al. [72, 3] applied curvelets for morphological component analysis. Recently, B. Zhang et al. [78] used curvelets for Poisson noise removal in comparison with wavelets and ridgelets. In [79], C. Zhang et al. successively applied the multiscale curvelet transform to multipurpose watermarking for content authentication and copyright verification. Jiang et al. [40] considered structure and texture image inpainting with the help of an iterative curvelet thresholding method. Tessens et al. [75] proposed a new context adaptive image denoising by modeling of curvelet domain statistics. By performing an inter-sub-band statistical analysis of

curvelet coefficients, one can distinguish between two classes of coefficients: those that represent useful image content, and those dominated by noise. Using a prior model based on marginal statistics, an appropriate local spatial activity indicator for curvelets has been developed that is found to be very useful for image denoising, see [75]. Geback et al. [34] applied the curvelets for edge detection in microscopy images.

Interestingly, the pure discrete curvelet transform is less suitable for image compression and for image denoising. The reason may be the redundancy of the curvelet frame. Most successful approaches related with the discrete curvelet transform are hybrid methods, where curvelets are combined with another technique for image processing. These methods usually can exploit the ability of the curvelet transform to represent curve-like features.

Let us give one example of image denoising [55], where curvelet shrinkage is combined with nonlinear anisotropic diffusion. Figure 11 (a) shows a part of noisy Barbara image. Figures 11 (b)-(f) present the denoising results by using tensor-product Daubechies's DB4 wavelets, TV diffusion, contourlets, curvelets, and TV-combined curvelet transform [55], respectively. The curvelet-based methods preserve the edges and textures well.

7.2 Seismic processing

Seismic data records the amplitudes of transient/reflecting waves during receiving time. The amplitude function of time is called seismic trace. A seismic data or profile is the collection of these traces. All the traces together provide a spatio-temporal sampling of the reflected wave field containing different arrivals that respond to different interactions of the incident wave field with inhomogeneities in the Earth's subsurface. Common denominators among these arrivals are wave fronts (as shown in Fig. 12 (a) for a real seismic profile), which display anisotropic line-like features, as edges and textures in images. They basically show behaviors of C^2-continuous curves. The main characteristic of the wave fronts is their relative smoothness in the direction along the fronts and their oscillatory behavior in the normal direction. A crucial problem in seismic processing is to preserve the smoothness along the wave fronts when one aims to remove noise.

From a geophysical point of view, curvelets can be seen as local plane waves. They are optimal to sparsely represent the local seismic events and can be effectively used for wave front-preserving seismic processing. Therefore, the curvelet decomposition is an appropriate tool for seismic data processing.

Fig. 12 shows a denoising of a real seismic data set by curvelets, in comparison to wavelets. Five decomposing levels are used in both transforms. Fig. 13 shows the comparison of subband reconstruction in the first three levels from coarse scale to fine scale. It can be seen clearly that the curvelets perform much better than wavelets to preserve the wave fronts/textures in multiscale decomposition and denoising. We also observe that the curvelet transform can achieve an almost complete data reconstruction if used without any thresholding for coefficients (reconstructed SNR = 310.47 and error = 2.9770e-010).

So far, curvelets have been applied successfully in seismic processing. Hennenfent and Herrmann [36] suggested a nonuniformly sampled curvelet transform for seismic denoising. Neelamani et al. [59] proposed a 3D curvelet-based effective approach to attenuate random and coherent noise in a 3D data set from a carbonate environment. Comparisons of wavelets, contourlets, curvelets, and their combination for denoising of random noise have been also investigated in [66]. Douma and de Hoop [28] presented a leading-order seismic imaging by curvelets

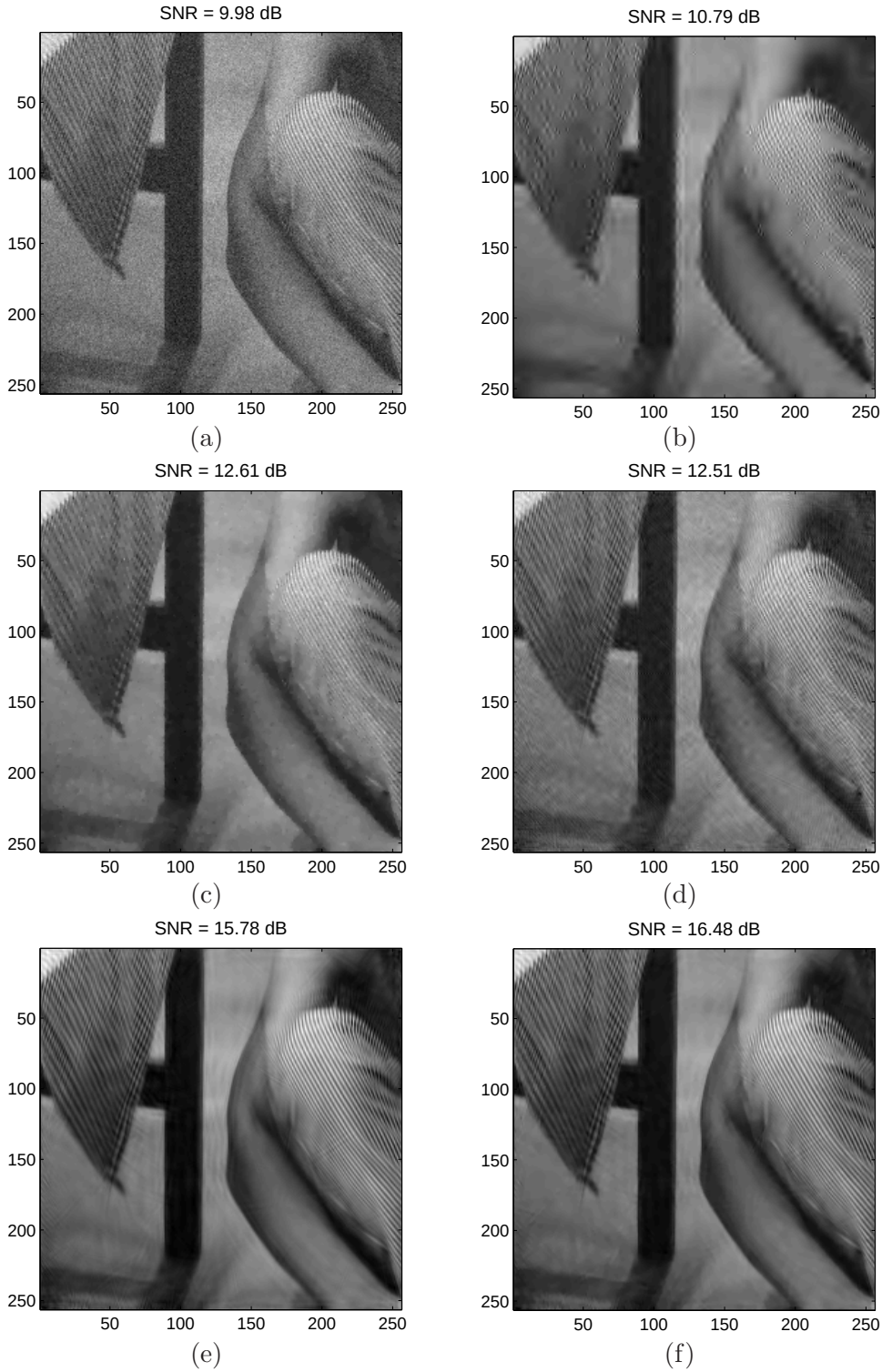
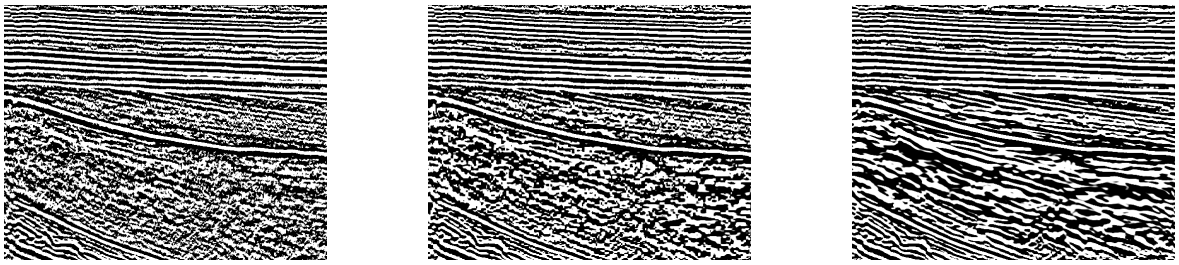


Figure 11: Image denoising. (a) Noisy image, (b) wavelet denoising, (c) TV-diffusion denoising, (d) contourlet denoising, (e) curvelet denoising, (f) TV-combined curvelet denoising.



(a) (b) (c)

Figure 12: Comparison of seismic denoising. (a) Original data, (b) wavelet denoising, (c) curvelet denoising.

They show that using curvelets as building blocks of seismic data, the Kirchhoff diffraction stack can, to leading order in angular frequency, horizontal wavenumber, and migrated location, be rewritten as a map migration of coordinates of the curvelets in the data, combined with an amplitude correction. This map migration uses the local slopes provided by the curvelet decomposition of the data. Chauris and Nguyen [17], and Chauris and Ma [18] considered seismic demigration/migration in the curvelet domain. The migration consists of three steps: decomposition of the input seismic data (e.g., common offset sections) using the curvelet transform; independent migration of the curvelet coefficients; inverse curvelet transform to obtain the final depth migrated image. Currently, they concentrate on a ray-based type of prestack depth-migration (i.e., common-offset Kirchhoff depth migration) with respect to heterogeneous velocity models. It comes out that curvelets are almost invariant under the migration operations. The final objective is to be able to derive a formulation and build an efficient algorithm for the full waveform inversion in the curvelet domain.

In addition, curvelet-based primary-multiple separation [39], extrapolation [45], and seismic data recovery [38, 37, 74] have been also proposed by Herrmann et al..

7.3 Turbulence analysis in fluid mechanics

Turbulence has been a source of fascination for centuries, because most fluid flows occurring in nature, as well as in engineering applications, are turbulent. Fluid turbulence is a paradigm of multiscale phenomena, where the coherent structures evolve in an incoherent random background. Turbulence is difficult to approximate and analyze mathematically or to calculate numerically because of its range of spatial and temporal scales. The geometrical representation of flow structures might seem to be restricted to a well defined set of curves along which the data are singular. As a consequence, the efficient compression of a flow field with minimum loss of the geometric flow structures is a crucial problem in the simulation of turbulence. The development of appropriate tools to study vortex breakdown, vortex reconnection, and turbulent entrainment at laminar-turbulent interfaces, is imperative to enhance our understanding of turbulence. Such tools must capture the vortical structure and dynamics accurately to unravel the physical mechanisms underlying these phenomena.

Recently, the curvelets have been applied to study the non-local geometry of eddy structures and the extraction of the coherent vortex field in turbulent flows [2, 53, 54]. Curvelets start to influence the field of turbulence analysis and have the potential to upstage the wavelet representation of turbulent flows addressed in [30, 31]. The multiscale geometric property, implemented

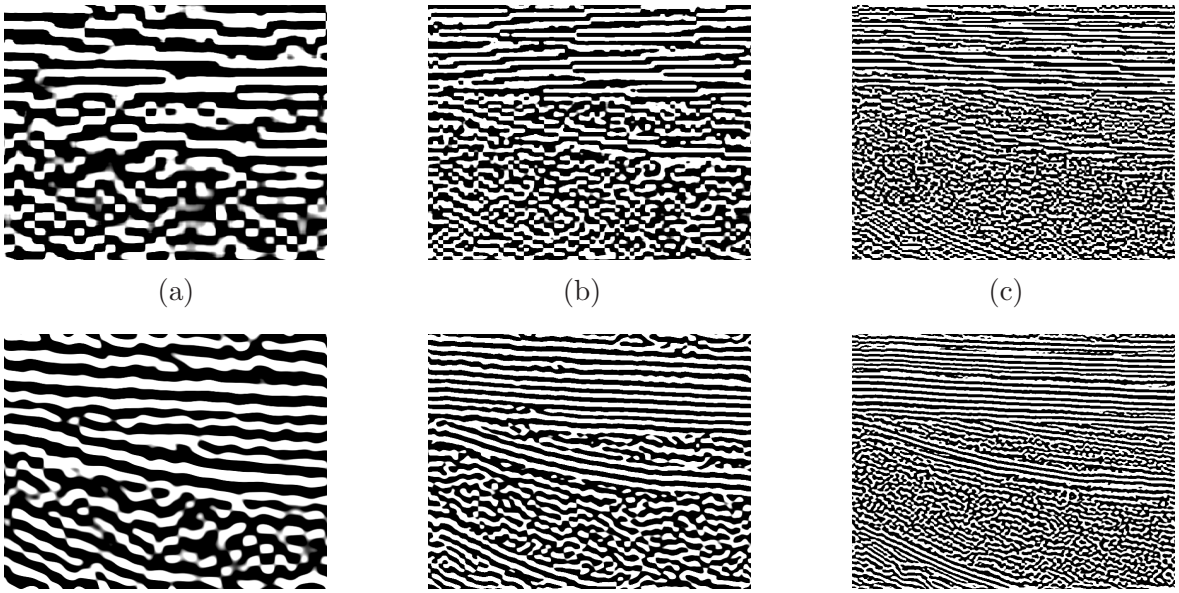


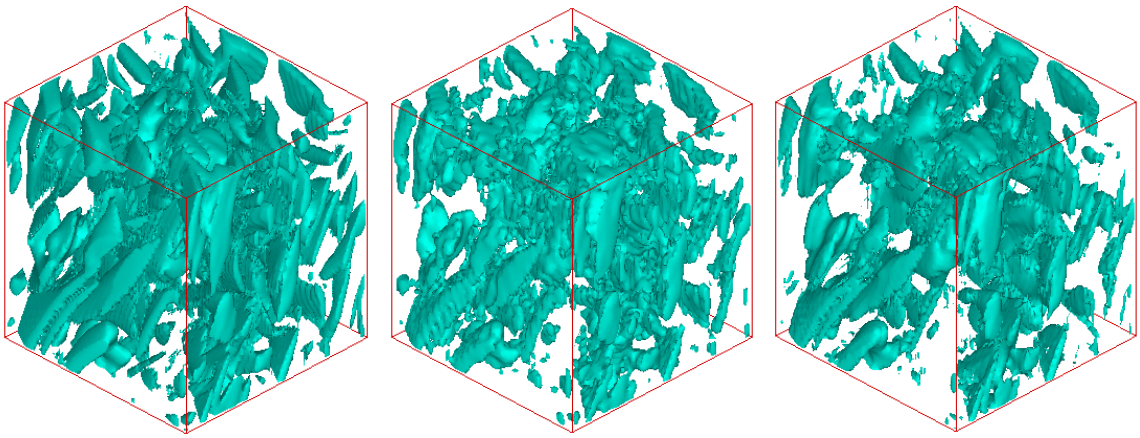
Figure 13: Comparisons of subband reconstruction in the first three levels from coarse scale to fine scale by wavelet transform (upper row) and curvelet transform (lower row).

by means of curvelets, provides the framework for studying the evolution of the structures associated to the main ranges of scales defined in Fourier space, while keeping the localization in physical space that enables a geometrical study of such structures. Such a geometrical characterization can provide a better understanding of cascade mechanics and dissipation-range dynamics. Moreover, curvelets have the potential to contribute to the development of structure-based models of turbulence fine scales, subgrid-scale models for large-eddy simulation, and simulation methods based on prior wavelet transforms [2].

Figure 14 gives an example for the extraction of coherent fields from turbulent flows. The curvelet method preserves the edges and structures better than wavelet methods. The results of multiscale turbulence analysis depend on the threshold or shrinkage. The question of how to find the optimal threshold to separate coherent fields and incoherent random fields still remains open.

7.4 Solving of PDEs

Candès and Demanet [6, 7] have shown that curvelets essentially provide optimally sparse representations of Fourier integral operators. While the wavelet transform is optimal for solving elliptical PDEs, the motivation to use the curvelet transform is that for a wide class of linear hyperbolic differential equations, the curvelet representation of the solution operator is both optimally sparse and well organized. Sparsity means that the matrix entries decay nearly exponentially fast, and they are well organized in the sense that very few nonnegligible entries occur near a few shifted diagonals. Wave fronts of solutions can be also sparsely represented in curvelet domain [7]. Some updated results for hyperbolic evolution equations with limited smoothness have been obtained by Andersson et al. [1]. The key idea of the existing methods is first to



(a) (b) (c)

Figure 14: Extraction of coherent fields from turbulent flows. (a) Original flow, (b) coherent components by wavelets, and curvelets.

decompose the initial fields by the curvelet transform, and then to compute the rigid motions of the significant curvelet coefficients along Hamiltonian ray flows at each scale. Finally, one needs to reconstruct the evolution coefficients at all scales by an inverse curvelet transform and obtains an approximate wave field $u(x,t)$ at a given time t. The theory is quite elegant but still far away from practical applications. The papers cited above show the potential of curvelets for solving of PDEs from the point of view of mathematical analysis and raise the hope to achieve fast algorithms for the solution of hyperbolic PDEs using curvelets.

Let us consider a wave equation with the associated Cauchy initial value problem,

$$\frac{\partial^2 u}{\partial t^2}(x,t) = v^2\,\Delta u(x,t) \quad u(x,0) = u_0(x), \quad \frac{\partial u}{\partial t}(x,0) = u_1(x). \tag{13}$$

For simplicity, assume that v is a constant wave speed, and $\Delta u(x,t) = \frac{\partial^2}{x_1^2}u(x,t) + \frac{\partial^2}{x_2^2}u(x,t)$ denotes the usual Laplace operator. Its solution can be written as $u(x,t) = F(x,t)u_0(x) + G(x,t)u_1(x)$, with suitable solution operators $F(x,t)$ and $G(x,t)$ (involving Green's functions) that can be sparsely represented in curvelet domain.

J. Ma and his students are working on curvelet-based finite difference schemes for seismic wave equations [56]. The goal is to construct a fast adaptive scheme for numerical modeling of wave propagation. Similarly as with prior wavelet-based finite difference schemes, one crucial problem is to explore how the differential operator Δ (or ∂_x) can be computed by the curvelet transform in an efficient way. The 2D wave field u can be transformed into curvelet domain by $u(x_1,x_2,t) = \sum_\mu c_\mu(t)\phi_\mu(x_1,x_2)$. Here, we have used the tight frame property (9) with the short notation $\mu = (j,k,l)$, and $c_\mu(t)$ denotes the μth curvelet coefficient of u at time t. A possible way to compute the curvelet coefficients of Δu is

$$c_\mu^\triangle := c_\mu(\Delta u) := \int \Delta u(x,t)\,\overline{\phi_\mu(x)}\,dx = \int \widehat{\Delta u}(\xi,t)\,\overline{\widehat{\phi_\mu}(\xi)}\,d\xi = \int (-\xi_1^2-\xi_2^2)\,\widehat{u}(\xi,t)\,\overline{\widehat{\phi_\mu}(\xi)}\,d\xi.$$

Using the definition of the curvelet coefficients in (12) we obtain with $S_{\theta_{j,l}}\xi = (\xi_1, -\xi_1\tan\theta_{j,l} +$

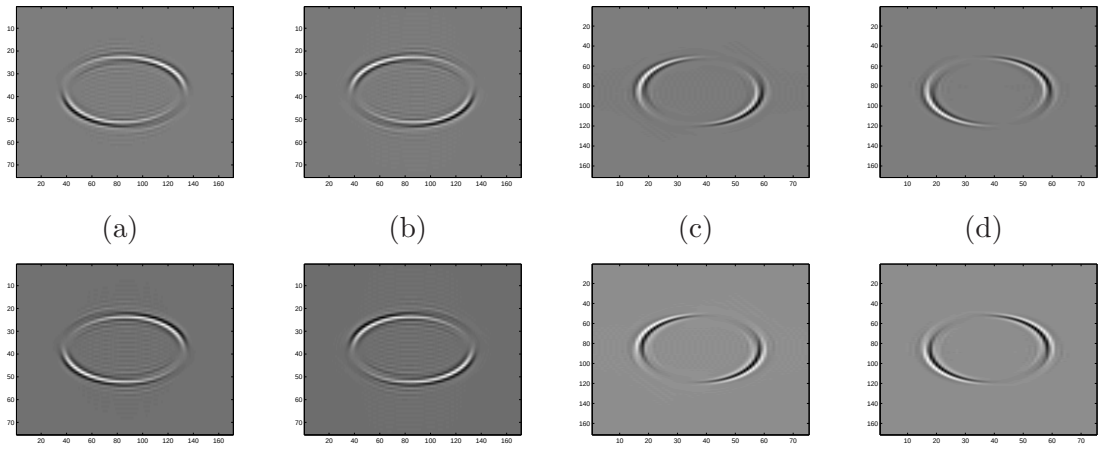


Figure 15: Curvelet coefficients of an instant wave field at the coarsest curvelet detail scale. (a)-(h) denotes eight different directional subbands in this curvelet scale.

$\xi_2)^T$

$$\begin{aligned} c_\mu^{\triangle} &= \int [-(1+\tan^2\theta_{j,l})\,\xi_1^2 - \xi_2^2 + 2(\tan\theta_{j,l})\,\xi_1\xi_2]\,\widehat{u}(S_{\theta_{j,l}}\xi)\,\widetilde{U}_j(\xi)\,e^{i\langle k_j,\xi\rangle}d\xi \\ &= 4^j(1+\tan^2\theta_{j,l})\frac{\partial^2 c_\mu}{\partial k_1^2} + 4^{\lfloor j/2\rfloor}\frac{\partial^2 c_\mu}{\partial k_2^2} - 2^{j+1}\,2^{\lfloor j/2\rfloor}\,\tan\theta_{j,l}\,\frac{\partial^2 c_\mu}{\partial k_1 \partial k_2}. \end{aligned}$$

Here we recall that $k = (k_1, k_2)^T \in \mathbb{Z}^2$ and $k_j = (k_1/2^j,\, k_2/2^{\lfloor j/2\rfloor})^T$. That means, we can obtain the curvelet coefficients of Δu by using the coefficients of the instant wave field u. Thus, we can rewrite the wave equation in coefficient domain by

$$\frac{\partial^2 c_\mu}{\partial t^2} = \upsilon^2\Big(4^j(1+\tan^2\theta_{j,l})\frac{\partial^2 c_\mu}{\partial k_1^2} + 4^{\lfloor j/2\rfloor}\frac{\partial^2 c_\mu}{\partial k_2^2} - 2^{j+1}\,2^{\lfloor j/2\rfloor}\tan\theta_{j,l}\frac{\partial^2 c_\mu}{\partial k_1\partial k_2}\Big). \tag{14}$$

Figure 15 shows an example of curvelet coefficients of an instant wave field at the coarsest curvelet detail scale, by implementing the computation in curvelet domain as given in (14). For details of this approach we refer to [73]. Using a suitable thresholding, one can implement a fast adaptive computation for the wave propagation. Unfortunately, due to the redundancy of the current discrete curvelet algorithm, the curvelets have not performed at the level that we expected. The matrices are not as sparse as the estimates promise. The efficient numerical treatment of PDEs using curvelets is still a challenging problem.

7.5 Compressed sensing

Finally, we mention a new direction of applications of the curvelet transform to the so-called compressed sensing or compressive sampling (CS), an inverse problem with highly incomplete measurements. CS [15, 16, 25] is a novel sampling paradigm, which carries imaging and compression simultaneously. The CS theory says that a compressible unknown signal can be recovered by a small number of random measurements using sparsity-promoting nonlinear recovery algorithms. The number of necessary measurements is considerably smaller than the number of

needed traditional measurements that satisfy the Shannon/Nyquist sampling theorem, where the sampling rate has to be at least twice as large as the maximum frequency of the signal. The CS based data acquisition depends on its sparsity rather than its bandwidth. CS might have an important impact for designing of measurement devices in various engineering fields such as medical magnetic resonance (MRI) imaging and remote sensing, especially for cases involving incomplete and inaccurate measurements limited by physical constraints, or very expensive data acquisition.

Mathematically, we handle the fundamental problem of recovering a signal $x \in \mathbb{R}^N$ from a small set of measurements $y \in \mathbb{R}^K$. Let $A \in \mathbb{R}^{K\times N}$ be the so-called CS measurement matrix, where $K \ll N$, i.e., there are much fewer rows in the matrix than columns. The measurements can be described as [15]

$$y = Ax + \epsilon. \tag{15}$$

Here ϵ denotes possible measurement errors or noise. It seems to be hopeless to solve this ill-posed underdetermined linear system since the number of equations is much smaller than the number of unknown variables. However, if the x is compressible by a transform, as e.g. $x = T^{-1}c$, where T denotes the discrete curvelet transform, and the sequence of discrete curvelet coefficients $c = (c_\mu)$ is sparse, then we have $y = AT^{-1}c + \epsilon = \tilde{A}c + \epsilon$. If the measurement matrix A is not correlated with T, the sparse sequence of curvelet coefficients c can be recovered by a sparsity-constraint l_1-minimization [15],

$$\min_c \|y - \tilde{A}c\|_{l_2} + \lambda\|c\|_{l_1}.$$

The second term is a regularization term that represents the a-priori information of sparsity. To solve the minimization, an iterative curvelet thresholding (ICT) can be used, based on the Landweber descent method (see e.g. [37]),

$$c_{p+1} = S_\tau(c_p + \tilde{A}^T(y - \tilde{A}c_p)),$$

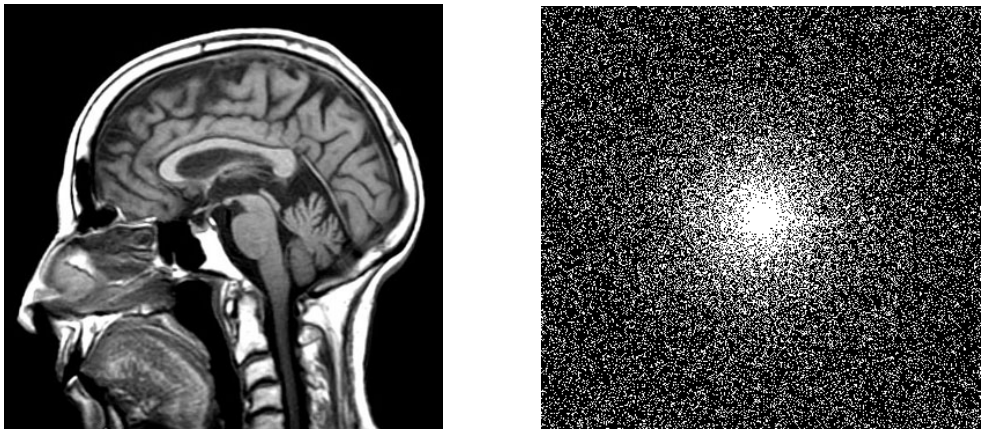
until $\|c_{p+1} - c_p\| < \varepsilon$, for a given error ε. Here the (soft) threshold function S_τ, given by

$$S_\tau(x) = \begin{cases} x - \tau, & x \geq \tau, \\ x + \tau, & x \leq -\tau, \\ 0, & |x| < \tau, \end{cases}$$

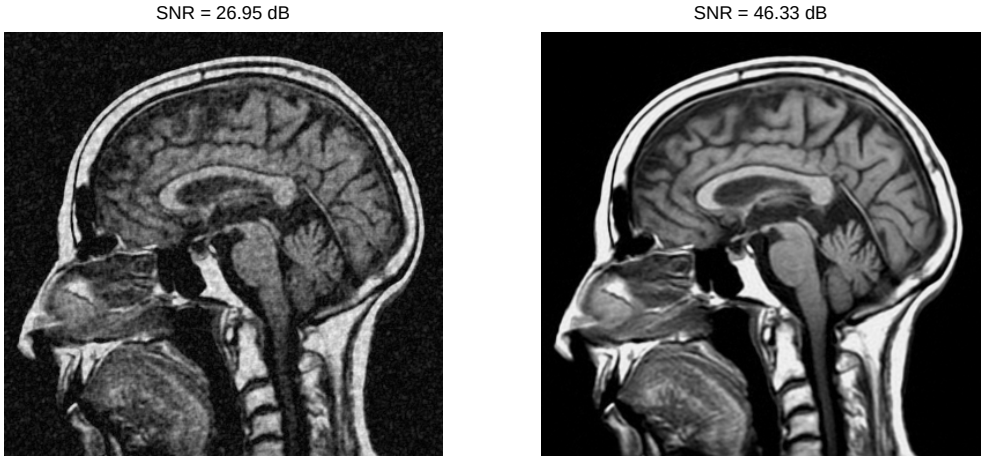
is taken componentwisely, i.e., for a sequence $a = (a_\mu)$ we have $S_\tau(a) = (S_\tau a_\mu)$.

Figure 16 shows an example of compressed sensing with 25 percent Fourier measurements. Here the operator A is obtained by a random subsampling of the Fourier matrix. Figure 16 (b) shows the 25% samples in Fourier domain, Figure 16 (c) is the recovering result by zero-filling reconstruction, and Figure 16 (d) is the result found by iterative curvelet thresholding. Figures 16 (e) and (f) denote the changes of the signal-to-noise ratio (SNR) and errors of the recovered images as the number of iterations increases. The unknown MRI image can be obtained by using highly incomplete measurements, which can reduce the on-line measurement time and thus lessen the pain of a patient.

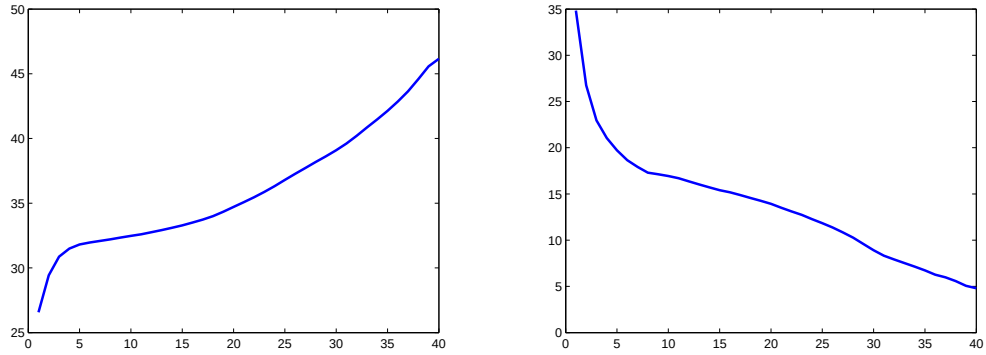
The motivation of applying the curvelet thresholding method is that most natural images are compressible by the curvelet transform. Currently, a few researchers have applied the ICT method to compressed sensing in seismic data recovery [37, 38, 74], and remote sensing [49, 51]. Variant ICT methods (see e.g. [65]) have been also proposed for compressed sensing.



(a) (b)



(c) (d)



(e) (f)

Figure 16: Compressed sensing in Fourier domain for medical imaging. (a) Original MRI image, (b) pseudo-random Fourier sampling, (c) recovery by zero-filling reconstruction, (d) recovery by ICT, (e) SNR (in dB) of the recovered image vs number of iterations for the ICT, (f) recovery error vs number of iterations for the ICT.

Figure 17 shows an example for the curvelet-based compressed sensing in remote sensing [49, 51]. It can be seen that the curvelet method is superior to the wavelet method to recover the edges.

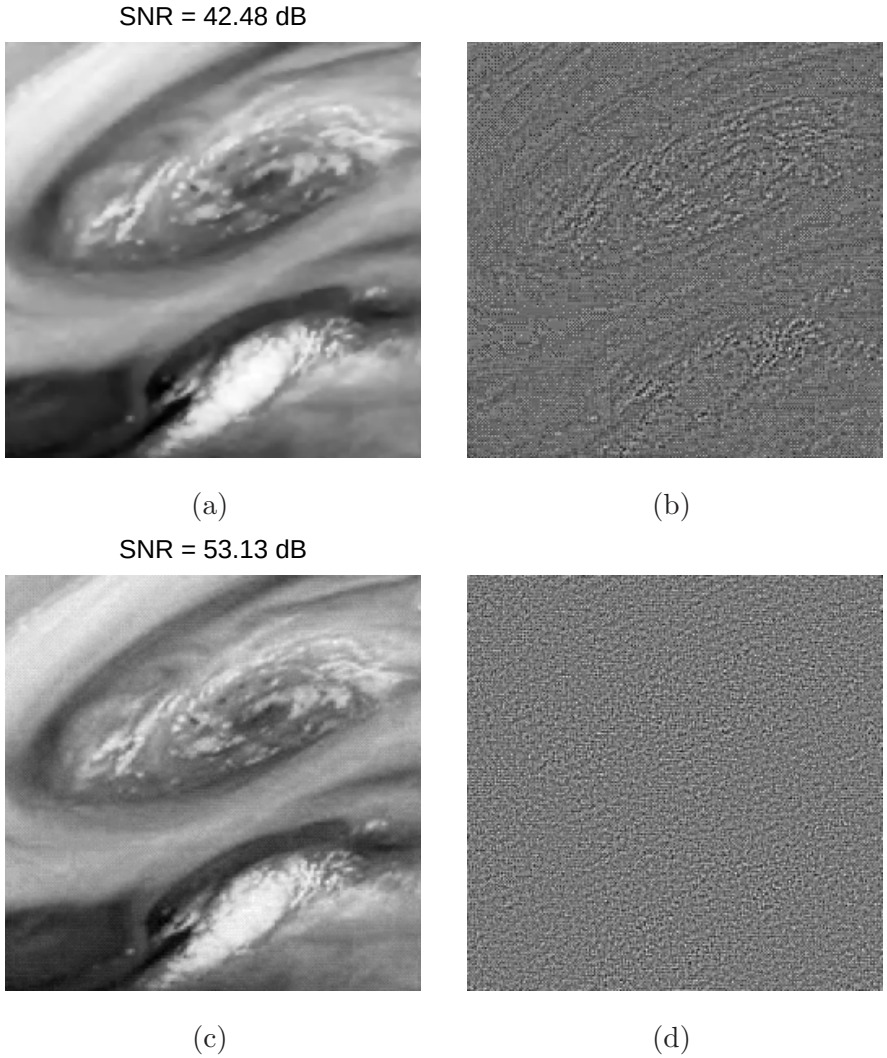


Figure 17: Compressed sensing in remote sensing. (a) Recovery by iterative wavelet thresholding, (b) recovery error by the wavelet method, (c) recovery by iterative curvelet thresholding, (d) recovery error by the curvelet method.

8 Future work

The multiresolution geometric analysis technique with curvelets as basis functions is verified as being effective in many fields. However, there are some challenging problems for future work.

1) The computational cost of the curvelet transform is higher than that of wavelets, especially in terms of 3D problems. However, the theory and application of the 3D curvelets are burgeoning areas of research, and it is possible that more efficient curvelet-like transforms will be developed

in the near future. Currently, a fast message passing interface-based parallel implementation can somewhat reduce the cost [77]. How to build a fast orthogonal curvelet transform is still open.

2) How to explore suitable thresholding functions that incorporate and exploit the special characteristics of the curvelet transform? This issue is very important for curvelet applications involving edge detection, denoising, and numerical simulation.

9 Acknowledgements

J. Ma has been supported by the NSFC under Grant No. 40704019 and 40674061, TBRF (JC2007030), PetroChina Innovation Fund (060511-1-1). G. Plonka has been supported by the German Research Foundation within the projects PL 170/11-2 and PL 170/13-1. These are gratefully acknowledged.

References

[1] F. Andersson, M. de Hoop, H. Smith, G. Uhlmann, A multi-scale approach to hyperbolic evolution equations with limited smoothness, Comm. Partial Differential Equations, **33**, 988-1017 (2008).

[2] I. Bermejo-Moreno, D. Pullin, On the non-local geometry of turbulence, J. Fluid Mech., **603**, 101-135 (2008).

[3] J. Bobin, J. Starck J. Fadili, Y. Moudden, D. Donoho, Morphological component analysis: an adaptative thresholding strategy, IEEE Trans. Image Process., **16** (11), 2675-2681 (2007).

[4] J. Bros, D. Iagolnitzer, Support essentiel et structure analytique des distributions. Séminare Goulaouic-Lions-Schwartz, exp. no. **19** (1975-1976).

[5] E. Candès, Harmonic analysis of neural networks, Appl. Comput. Harmon. Anal., **6**, 197-218 (1999).

[6] E. Candès, L. Demanet, Curvelets and Fourier integral operators, C. R. Math. Acad. Sci. Paris, **336** (5), 395-398 (2003).

[7] E. Candès, L. Demanet, The curvelet representation of wave propagators is optimally sparse, Commun. Pure Appl. Math., **58** (11), 1472-1528 (2005).

[8] E. Candès, L. Demanet, D. Donoho, L. Ying, Fast discrete curvelet transforms, Multiscale Model. Simul., **5** (3), 861-899 (2006).

[9] E. Candès, D. Donoho, Ridgelets: A key to higher-dimensional intermittency?, R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci., **357**, 2495-2509 (1999).

[10] E. Candès, D. Donoho, Curvelets - a surprisingly effective nonadaptive representation for objects with edges, In Curves and Surface Fitting: Saint-Malo 1999, A. Cohen, C. Rabut, L. Schumaker (Eds.), Vanderbilt Univ. Press, Nashville, 2000, 105-120.

[11] E. Candès, D. Donoho, Continuous curvelet transform: I. Resolution of the wavefront set, Appl. Comput. Harmon. Anal., **19**, 162-197 (2003).

[12] E. Candès, D. Donoho, Continuous curvelet transform: II. Discretization and frames, Appl. Comput. Harmon. Anal., **19**, 198-222 (2003).

[13] E. Candès, D. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise singularities, Comm. Pure Appl. Math., **57**, 219-266 (2004).

[14] E. Candès, F. Guo, New multiscale transforms, minimum total variation synthesis: applications to edge-preserving image reconstruction, Signal Process., **82** (11), 1519-1543 (2002).
[15] E. Candès, J. Romberg, T. Tao, Stable signal recovery from incomplete and inaccurate information, Commun. Pure Appl. Math., **59**, 1207-1233 (2005).
[16] E. Candès, T. Tao, Decoding by linear programming, IEEE Trans. Inform. Theory, **51**, 4203-4215 (2005).
[17] H. Chauris, T. Nguyen, Seismic demigration/migration in the curvelet domain, Geophysics, **73** (2), S35-S46 (2008).
[18] H. Chauris, J. Ma, Seismic imaging in the curvelet domain: achievements and perspectives, 71st EAGE conference and Exhibition, Amsterdam, Netherlands, 8-11 June 2009.
[19] M. Choi, R. Kim, M. Nam, H. Kim, Fusion of multispectral and panchromatic satellite images using the curvelet transform, IEEE Geosci. Remote Sensing Lett., **2** (2), 136-140 (2005).
[20] A. Cordoba, C. Fefferman, Wave packets and Fourier integral operators, Comm. Partial Differential Equations, **3**, 979-1005 (1978).
[21] I. Daubechies, Ten lectures on wavelets, Philadelphia, PA: SIAM, 1992.
[22] L. Demanet, L. Ying, Curvelets and wave atoms for mirror-extend images, Proc. SPIE Wavelets XII, San Diego, August 2007.
[23] L. Demanet, L. Ying, Wave Atoms and Sparsity of Oscillatory Patterns, Appl. Comput. Harmon. Anal., **23** (3), 368-387 (2007).
[24] M. Do, M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, IEEE Trans. Image Process., **14** (12), 2091-2106 (2005).
[25] D. Donoho, Compressed sensing, IEEE Trans. Inform. Theory, **52** (4), 1289-1306 (2006).
[26] D. Donoho, Wedgelets: nearly minimax estimation of edges, Ann. Statistics, **27** (3), 859-897 (1999).
[27] D. Donoho, X. Huo, Beamlets and multiscale image analysis, in: Multiscale and Multiresolution Methods, Springer Lecture Notes in Comput. Sci. Eng., T. Barth et al. (Eds.), vol. 20, pp. 149-196, 2002.
[28] H. Douma, M. de Hoop, Leading-order seismic imaging using curvelets, Geophysics, **72** (6), S231-S248 (2007).
[29] G. Easley, D. Labate, F. Colonna, Shearlet based total variation for denoising, IEEE Trans. Image Process., **18** (2), 260-268 (2009).
[30] M. Farge, N. Kevlahan, V. Perrier, E. Goirand, Wavelets and turbulence, Proceedings of the IEEE, **84** (4), 639-669 (1996).
[31] M. Farge, G. Pellegrino, K. Schneider, Coherent vortex extraction in 3D turbulent flows using orthogonal wavelets, Phys. Rev. Lett., **87** (5), 45011-45014 (2001).
[32] W. Freeman, E. Adelson, The design and use of steerable filters, IEEE Trans. Pattern Analysis and Machine Intelligence, **13** (9), 891-906 (1991).
[33] X. Gao, T. Nguyen, G. Strang, A study of two-channel complex-valued filter banks and wavelets with orthogonality and symmetry properties, IEEE Trans. Signal Process., **50** (4), 824-833 (2002).
[34] T. Geback, P. Koumoutsakos, Edge detection in microscopy images using curvelets, BMC Bioinformatics, **10** (75),1471-2105-10-75 (2009).
[35] K. Guo, D. Labate, Optimally sparse multidimensional representation using shearlets, SIAM J. Math. Anal., **39**, 298-318 (2007).
[36] G. Hennenfent, F. Herrmann, Seismic denoising with nonuniformly sampled curvelets, Comput. Sci. Eng., **8** (3), 16-25 (2006).

[37] F. Herrmann, G. Hennenfent, Non-parametric seismic data recovery with curvelet frames, Geophysical J. Int., **173** (1), 233-248 (2008).
[38] F. Herrmann, P. Moghaddam, C. Stolk, Sparsity- and continuity-promoting seismic image recovery with curvelet frames, Appl. Comput. Harmon. Anal., **24** (2), 150-173 (2008).
[39] F. Herrmann, D. Wang, D. Verschuur, Adaptive curvelet-domain primary-multiple separation, Geophysics, **73** (3), A17-A21 (2008).
[40] L. Jiang, X. Feng, H. Yin, Structure and texture image inpainting using sparse representations and an iterative curvelet thresholding approach, Int. J. Wavelets, Multiresolution and Inform. Process., **6** (5), 691-705 (2008).
[41] N. Kingsbury, Image processing with complex wavelets, Phil. Trans. R. Soc. Lond. A, **357**, 2543-2560 (1999).
[42] N. Kingsbury, Complex wavelets for shift invariant analysis and filtering of signals, Appl. Comput. Harmon. Anal., **10** (3), 234-253 (2001).
[43] D. Labate, W-Q. Lim, G. Kutyniok, and G. Weiss. Sparse multidimensional representation using shearlets, SPIE Proc. Wavelets XI, vol. 5914, pp. 254-262, San Diego, CA, 2005.
[44] T. Lee, Image representation using 2D Gabor wavelets, IEEE Trans. Pattern Analysis and Machine Intelligence, **18** (10), 1-13 (2008).
[45] T. Lin, F. Herrmann, Compressed extrapolation, Geophysics, **72** (5), SM77-SM93 (2007).
[46] Y. Lu, M. N. Do, Multidimensional directional filter banks and surfacelets, IEEE Trans. Image Process., **16** (4), 918-931 (2007).
[47] J. Ma, Curvelets for surface characterization, Appl. Phys. Lett., **90**, 054109 (2007).
[48] J. Ma, Deblurring using singular integrals and curvelet shrinkage, Physics Letters A, **368**, 245-250 (2007).
[49] J. Ma, Single-pixel remote sensing, IEEE Geosci. Remote Sensing Lett., **6** (2), 199-203 (2009).
[50] J. Ma, A. Antoniadis, F.-X. Le Dimet, Curvelet-based multiscale detection and tracking for geophysical fluids, IEEE Trans. Geosci. Remote Sensing, **44** (12), 3626-3637 (2006).
[51] J. Ma, F.-X. Le Dimet, Deblurring from highly incomplete measurements for remote sensing, IEEE Trans. Geosci. Remote Sensing, **47** (3), 792-802 (2009).
[52] J. Ma, M. Fenn, Combined complex ridgelet shrinkage and total variation minimization, SIAM J. Sci. Comput., **28** (3), 984-1000 (2006).
[53] J. Ma, M. Hussaini, Three-dimensional curvelets for coherent vortex analysis of turbulence, Appl. Phys. Lett., **91**, 184101 (2007).
[54] J. Ma, M. Hussaini, O. Vasilyev, F.-X. Le Dimet, Multiscale geometric analysis of turbulence by curvelets, Physics of Fluids, **21**, 075104 (2009).
[55] J. Ma, G. Plonka, Combined curvelet shrinkage and nonlinear anisotropic diffusion, IEEE Trans. Image Process., **16** (9), 2198-2206 (2007).
[56] J. Ma, G. Tang, M. Y. Hussaini, A refining estimation for adaptive solution of wave equation based on curvelets, in Wavelets XII, D. Van De Ville, V. K. Goyal, M. Papadakis (Eds.), Proc. of SPIE, Vol. 6701, pp. 67012J, San Diego, CA, 2007.
[57] S. Mallat, Wavelets for a vision, Proceedings of The IEEE, **84** (4), 604-614 (1996).
[58] S. Mallat, G. Peyré, A review of bandlet methods for geometrical image representation, Numer. Algorithms, **44** (3), 205-234 (2007).
[59] R. Neelamani, A. Baumstein, D. Gillard, M. Hadidi, W. Soroka, Coherent and random noise attenuation using the curvelet transform, the leading edge, **27** (2), 240-248 (2008).

[60] J. Neumann, G. Steidl, Dual–tree complex wavelet transform in the frequency domain and an application to signal classification, Int. J. Wavelets, Multiresolution and Inform. Process., **3** (1), 43-66 (2005).

[61] E. Le Pennec, S. Mallat, Sparse geometrical image approximation with bandlets, IEEE Trans. Image Process., **14** (4), 423-438 (2005).

[62] B. Olshausen, D. Field, Emergence of simple-cell receptive filed properties by learning a sparse code for natural images, Nature, **381**, 607-609 (1996).

[63] B. Olshausen, D. Field, Sparse coding of sensory inputs, Current Opinion in Neurobiology, **14**, 481-487 (2004).

[64] G. Plonka, J. Ma, Nonlinear regularized reaction-diffusion filters for denoising of images with textures, IEEE Trans. Image Process., **17** (8), 1283-1294 (2008).

[65] G. Plonka, J. Ma, Curvelet-wavelet regularized split Bregman method for compressed sensing, preprint, 2009.

[66] H. Shan, J. Ma, H. Yang, Comparisons of wavelets, contourlets, and curvelets for seismic denoising, J. Applied Geophysics, 2009, to appear.

[67] E. Simoncelli, W. Freeman, E. Adelson, D. Heeger, Shiftable multiscale transforms, IEEE Trans. Inform. Theory, **38** (2), 587-607 (1992).

[68] H. F. Smith, A Hardy space for Fourier integral operators, J. Geom. Anal., **8**, 629-653 (1998).

[69] J. Starck, E. Candès, D. Donoho, The curvelet transform for image denoising, IEEE Trans. Image Process., **11**, 670-684 (2002).

[70] J. Starck, E. Candès, D. Donoho, Astronomical image representation by the curvelet transform, Astronomy and Astrophysics, **398**, 785-800 (2003).

[71] J. Starck, F. Murtagh, E. Candès, F. Murtagh, D. Donoho, Gray and color image contrast enhancement by the curvelet transform, IEEE Trans. Image Process., **12** (6), 706-717 (2003).

[72] J. Starck, M. Elad, D. Donoho, Image decomposition via the combination of sparse representation and a variational approach, IEEE Trans. Image Process., **14** (10), 1570-1582 (2005).

[73] B. Sun, J. Ma, H. Chauris, H. Yang, Solving the wave equation in the curvelet domain: a mulitscale and multidirectional approach, J. Seismic Exploration, 2009, to appear.

[74] W. Tan, J. Ma, F. Herrmann, Improved compressed sensing for curvelet-based seismic data reconstruction, Preprint, 2009.

[75] L. Tessens, A. Pizurica, A. Alecu, A. Munteanu, W. Philips, Context adaptive image denoising through modeling of curvelet domain statistics, J. Electronic Imaging, **17** (3), 03021:1-17 (2008).

[76] R. Willett, K. Nowak, Platelets: A multiscale approach for recovering edges and surfaces in photon-limited medical imaging, IEEE Trans. Med. Imaging, **22** (3), 332-350 (2003).

[77] L. Ying, L. Demanet, E. Candès, 3D Discrete Curvelet Transform, Proc. of SPIE Wavelets XI, Vol. 5914, pp. 591413, San Diego, CA, 2005.

[78] B. Zhang, J. Fadili, J. Starck, Wavelets, ridgelets, and curvelets for Poisson noise removal, IEEE Trans. Image Process., **17** (7), 1093-1108 (2008).

[79] C. Zhang, L. Cheng, Z. Qiu, L. Cheng, Multipurpose watermarking based on multiscale curvelet transform, IEEE Trans. Inform. Forensics and Security, **3** (4), 611-619 (2008).

Jianwei Ma received the Ph.D. degree in solid mechanics from Tsinghua University in 2002. He has been a Visiting Scholar, Postdoctoral Research Fellow, and Guest Professor with the University of Cambridge, University of Oxford, University of Huddersfield, University of Grenoble (INRIA), and University of Duisburg-Essen. Since 2006, he has been an Assistant Professor with the School of Aerospace, and a Research Director with the Institute of Seismic Exploration, Tsinghua University. From June to August 2006, he was a Visiting Professor in Swiss Federal Institute of Technology (EPFL). From June to September 2007, he was a Visiting Professor in Florida State University and University of Colorado at Boulder. In May 2008 and from March to September 2009, he was an Invited Professor in INRIA-Grenoble and Ecole des Mines de Paris. His research interests include wavelets, curvelets, image processing, compressed sensing, remote sensing and seismic exploration. He is an Editor of Int. J. Artificial Intelligence. Contact him at jma@tsinghua.edu.cn.

Gerlind Plonka received the Diploma degree in mathematics from the University of Rostock (Germany), in 1990, and the Ph.D. degree in mathematics as well as the Habilitation from the University of Rostock, in 1993 and 1996, respectively. Since 1998, she is Associate Professor for Applied Analysis at the University of Duisburg-Essen. Her current research interests include wavelet and frame theory, Fourier analysis, nonlinear diffusion and applications in signal and image processing. Contact her at gerlind.plonka@uni-due.de.

GEOPHYSICS, VOL. 86, NO. 4 (JULY-AUGUST 2021); P. S299–S310, 13 FIGS.
10.1190/GEO2019-0307.1

True amplitude depth migration using curvelets

Hamideh Sanavi[1], Peyman P. Moghaddam[1], and Felix J. Herrmann[2]

ABSTRACT

We have developed a true amplitude solution to the seismic imaging problem. We derive a diagonal scaling approach for the normal operator approximation in the curvelet domain. This is based on the theorem that states that curvelets remain approximately invariant under the action of the normal operator. We use curvelets as essential tools for approximation and inversion. We also exploit the theorem that states that the curvelet-domain approximation should be smooth in phase space by enforcing the smoothness of curvelet coefficients in the angle and space domains. We analyze our method using a reverse time migration-demigration code, simulating the acoustic wave equation on different synthetic models. Our method produces a good resolution with reflecting dips, reproduces the true amplitude reflectors, and compensates for incomplete illumination in seismic images.

INTRODUCTION

Modern depth migration methods can accurately position reflecting events in the earth's subsurface. However, these methods often do not correctly resolve the amplitudes, which is true even when accurate knowledge of the velocity model is available. This comes from the fact that the migration operator is the adjoint but not the inverse of the modeling operator, with the modeling operator acting as the operator that maps the subsurface reflectivities to the recorded seismic data (see, e.g., Tarantola, 1984; Claerbout, 1992; Gray, 1997; Chauris and Cocher, 2017). Researchers have proposed three main approaches to derive true-amplitude migration schemes.

The first approach is the in-the-migration approach for correcting amplitudes. In this scheme, corrections are made during migration by modifying the imaging condition that corrects the reflector amplitudes (see, e.g., Zhang et al., 2007; Chattopadhyay and McMechan, 2008; Schleicher et al., 2008; Costa et al., 2009; da Silva Neto et al., 2011; Yang et al., 2015). Some researchers propose to define an approximate inverse operator for reverse time migration (RTM) (Hou and Symes, 2015; Chauris and Cocher, 2017; Li and Chauris, 2018), with the asymptotic approximation of the inverse operator using a pure wave-equation-based expression.

Second, the fact that the migration operator is the adjoint but not the (pseudo-)inverse of the scattering operator (see, e.g., Claerbout, 1992; Gray, 1997; Guitton, 2004; Symes, 2007; Herrmann and Li., 2012) leads to a second category of methods. In these methods, a variety of algorithms for true-amplitude migration have been introduced, which iteratively (using Lanczos or other linear algebra methods) update the migrated image (see, e.g., Tarantola, 1984; Kuel and Sacchi, 2003; Mulder and Plessix, 2004; Lu et al., 2017).

Third, there are the so-called image-to-image scaling methods, in which a scaling method is proposed for approximation of the Hessian (normal operator). The scaling is done by comparing the migrated image (with some possible pre-/postprocessing) with a remigrated image (i.e., applying the normal operator on the migrated image).

Typically, scaling is performed in some transform domains. The aim is to choose the scaling domain that can capture most important properties of the normal operator, namely, scattering point location invariance, compact support in the Fourier domain, invariance under the curvelet transform, etc. (see, e.g., Rickett, 2003; Guitton, 2004; Plessix and Mulder, 2004; Herrmann et al., 2008; Symes, 2008). Likewise, Liu and Peter (2018) propose a method to define the matching filter in the data domain for image-to-image scaling

Manuscript received by the Editor 31 May 2019; revised manuscript received 26 February 2021; published ahead of production 17 April 2021; published online 12 July 2021.
[1]Ferdowsi University of Mashhad, Faculty of Sciences, Razavi Khorasan 9177948974, Iran. E-mail: hamidehsanavi@gmail.com; ppm1407@hotmail.com (corresponding author).
[2]University of British Columbia, Seismic Laboratory for Imaging and Modeling, Department of Earth and Ocean Sciences, British Columbia V6T 1Z4, Canada and Georgia Institute of Technology, Earth and Atmospheric Sciences, Atlanta, Georgia 30332, USA. E-mail: felix.herrmann@gatech.edu.

methods. Our method falls into this category of using curvelets as transform domain scaling.

Recently, there has been some attention toward curvelet-domain scaling methods by Wang et al. (2016, 2017). One of the major differences between this study and previous work in this area is using the result of the theorem that states that curvelets are almost invariant under the action of the normal operator (Herrmann et al., 2008). Therefore, curvelets can be considered as the eigenvector for the normal operator, and we can represent it in a quasieigenvalue and eigenvector form with curvelets as the eigenvectors (Herrmann et al., 2008).

Another difference between this approximation and the curvelet-domain approximation used by Wang et al. (2016, 2017) is that the match is performed from the reference image to the approximate image in the physical domain rather than in the curvelet domain. Matching in the physical domain is more preferable than that in the curvelet domain because the latter generally leaves some artifacts behind due to curvelet coefficient truncation. In general, a large curvelet coefficient will leave a curvelet-like artifact in the physical domain.

The main contributions of this paper compared to the earlier work of Herrmann et al. (2008) are threefold. First, we replace the linear least-squares formulation for estimation of the curvelet domain coefficients (Herrmann et al., 2008) by a nonlinear least-squares formulation, which imposes positivity on the scaling coefficients. This comes from the fact that the normal operator is positive semidefinite. Second, we impose smoothness constraints on the scaling curvelet coefficients in physical and phase spaces to exploit the theorem (Herrmann et al., 2008), which states that symbols of the normal operator (in this case, curvelet coefficients) are smooth. Third, we remove the necessity for a second optimization (namely, stable amplitude recovery with the L1 norm) because we estimate the inverse of the normal operator directly and do not need to invert it anymore.

We start with a short review explaining the theoretical underpinning of image-to-image scaling methods. Then, we present our formulation, the diagonal estimation for the normal operator in the curvelet domain. Following that, we summarize our algorithm. Finally, we analyze the merit of our method on a series of examples with various complexity including the BP synthetic data set (Billette and Brandsberg-Dahl, 2005).

METHODOLOGY

Ideally, the data used before the migration are primary only (no multiples), which is mathematically described as

$$\mathbf{d} = \mathbf{K}\mathbf{m}, \tag{1}$$

where $\mathbf{d}$ is the data; $\mathbf{K}$ is the linearized Born scattering operator, which is also called the demigration operator; and $\mathbf{m}$ is the reflectivity. Theoretically, migration is the adjoint of the scattering operator so that when migration is applied to data we have

$$\mathbf{m}_{\text{mig}} = \mathbf{K}^T\mathbf{d} = \mathbf{K}^T\mathbf{K}\mathbf{m}, \tag{2}$$

where $\mathbf{K}^T$ is the transpose of $\mathbf{K}$ and the superscript T means transpose. The term $\mathbf{K}^T\mathbf{K}$ is the normal operator, which is also called the Hessian matrix in seismic imaging. To estimate $\mathbf{m}$, we need to invert this operator. In image-to-image-based methods (Rickett, 2003; Guitton, 2004; Plessix and Mulder, 2004; Herrmann et al., 2008; Symes, 2008), the migrated image is once again remigrated, which means demigration, followed by migration (i.e., $\mathbf{m}_{\text{remig}} = \mathbf{K}^T\mathbf{K}\mathbf{m}_{\text{mig}}$). Subsequently, using pairs of migrated and remigrated images, an approximation for the inverse of the normal operator can be found. Mathematically, the estimation for the inverse of the normal operator can be written as

$$\mathbf{m}_{\text{mig}} = (\mathbf{K}^T\mathbf{K})f^{-1}\mathbf{m}_{\text{remig}}, \tag{3}$$

$$= \mathbf{S}^\dagger\mathbf{W}\mathbf{S}\mathbf{m}_{\text{remig}}, \tag{4}$$

with $\mathbf{m}_{\text{mig}}$ being the migrated image, $\mathbf{m}_{\text{remig}}$ the remigrated image (i.e., $\mathbf{m}_{\text{remig}} = \mathbf{K}^T\mathbf{K}\mathbf{m}_{\text{mig}}$), $\mathbf{K}$ the demigration operator, and $\mathbf{K}^T$ the migration operator. The approximation consists of transforming (by $\mathbf{S}$, a linear transformation) the remigrated image, followed by a scaling with a positive diagonal matrix ($\mathbf{W}$), an inverse transform to map it back to the physical domain through the pseudoinverse of $\mathbf{S}$ (denoted by $\mathbf{S}^\dagger$). The performance of the scaling methods depends on two components: first is the choice of the appropriate domain for the scaling (i.e., the matrix $\mathbf{S}$) and the second is the calculation of the filter coefficients, i.e., the diagonal of $\mathbf{W}$, from the migrated image and remigrated image.

Rickett (2003) uses the scaling method first proposed by Claerbout and Nichols (1994), applies the identity basis ($\mathbf{S} = \mathbf{Id}$ with $\mathbf{Id}$ as the identity matrix), and calculates the filter coefficients by element-wise division of images. Aside from possible instability in this division, Rickett (2003) does not apply the appropriate preconditioning to the migration operator to be sure that operator is of zero order (see, e.g., Herrmann et al., 2008; Symes, 2008). This may lead to inaccuracies in the region of the image where there are phase changes due to the nature of the operator (see, e.g., Herrmann et al., 2008). Guitton (2004) improves this technique by applying scaling in a domain spanned by a filter bank. This approach allows him to treat dips locally and to impose smoothness of the filter coefficients. However, this approach does not correct for the order of the migration operator. More recently, Symes (2008) introduces a scaling approach that corrects for the order of the normal operator by filtering the migrated and demigrated images with a fractional inverse Laplacian, followed by estimation of the physical-domain scaling using bicubic splines. He also imposes smoothness of the scaling coefficients. However, this approach assumes the existence of a well-defined dip in the migrated image because splines do not capture any dip information in the approximation. In the high-frequency limit, the scattering operator and the normal operator can, under certain conditions on the medium and ray geometry, be considered as Fourier integral operators (FIOs) (Ten-Kroode et al., 1998). In two dimensions $(d = 2)$, the scattering operator $\mathbf{K}$ and its adjoint, the migration operator $\mathbf{K}^T$, can be considered as FIOs of order $1/4$, whereas the leading behavior for their composition, the normal operator, corresponds to an order-one invertible elliptic pseudodifferential operator (PsDO). To make this PsDO amenable to an approximation by curvelets, these operators should be made zero order by composing the data side with a $1/2$-order fractional integration along the time coordinate, i.e., $d- > (\partial./\partial t)^{-\frac{1}{2}}d$. Alternatively, we can half-integrate the source wavelet along the time coordinate.

CURVELET-DOMAIN DIAGONAL APPROXIMATION OF THE NORMAL OPERATOR

A 2D curvelet ϕ_μ is defined by its index $\mu = (j, k, \theta)$ with $j = 0, 1, 2, \ldots$ being the scale index, $\theta = 0, 1, \ldots, 2^{\lfloor j/2 \rfloor} - 1$ the orientation index ($\lfloor x \rfloor$ is the lower integer part of x), and the location index $k = (x, z)$. Figure 1a and 1b shows an example of different curvelets with different scaling, orientation, and locations and their corresponding Fourier spectra, respectively. An important property of curvelets is that the action of the normal operator (i.e., $\mathbf{\Psi} = \mathbf{K}^T\mathbf{K}$) on a curvelet ϕ_μ is approximately mapped into the same curvelet within a scalar. Mathematically, we can derive the following approximation:

$$\mathbf{\Psi}\phi_\mu \approx d_\mu \phi_\mu, \tag{5}$$

with d_μ being a positive scalar multiplier dependent on the index $\mu \in M$, with M being the index set. The $\approx$ symbol denotes "approximated by." In Herrmann et al. (2008), we theoretically and empirically show this behavior by studying the behavior of curvelets at different scales, angles, and locations after applying the normal operator. In addition, we prove a theorem that states that the above approximation (see equation 5) becomes more accurate at the finer scales (i.e., at higher spatial frequencies).

Approximation formulation

Equation 5 suggests that curvelets are similar to eigenvectors of the normal operator. Therefore, we suggest curvelets as quasieigenvectors and form a quasieigenvalue decomposition for the normal operator. This decomposition is similar to wavelet-vaguelette decomposition proposed by Donoho (1995) and Herrmann et al. (2008),

$$\mathbf{\Psi}\phi_\mu \approx \mathbf{C}^T\mathbf{P}_\mathbf{\Psi}\mathbf{C}\phi_\mu, \tag{6}$$

where $\mathbf{C}$ and $\mathbf{C}^T$ are the curvelet transform and its adjoint (Candes et al., 2006) and $\mathbf{P}_\mathbf{\Psi}$ is a diagonal scaling matrix with elements of d_μ in equation 5. Because curvelets are tight frames $\mathbf{C}^T\mathbf{C} = \mathbf{I}$, with $\mathbf{I}$ as the identity or unit matrix, assuming near unitary we can safely assume that a similar approximation holds for the inverse of the normal operator with

$$\mathbf{\Psi}^{-1}\phi_\mu \approx \mathbf{C}^T\mathbf{P}_\mathbf{\Psi}^{-1}\mathbf{C}\phi_\mu. \tag{7}$$

As we mentioned, our method is based on an image-to-image fitting scheme. Comparing equation 7 with equation 3 results in $\mathbf{W} = \mathbf{P}_\mathbf{\Psi}^{-1}$; hence, we have the following form:

$$\mathbf{m}_{\text{mig}} \approx \mathbf{C}^T\mathbf{W}\mathbf{C}\mathbf{m}_{\text{remig}} \mapsto \mathbf{m}_{\text{mig}} \approx \mathbf{C}^T\text{diag}(\mathbf{v})\mathbf{w}, \tag{8}$$

with $\mathbf{v} = \mathbf{C}\mathbf{m}_{\text{remig}}$ and $\mathbf{w}$ representing the diagonal elements of $\mathbf{W}$ (i.e., $\mathbf{W} = \text{diag}(\mathbf{w})$). This method aims to find a curvelet-domain diagonal scaling approximation (i.e., estimating $\mathbf{w}$ in equation 8) that resembles the action of the inverse of the normal operator on the reference vector. Because the curvelet transform is redundant, equation 8 is an underdetermined system of equations yielding nonunique solutions. To find a unique solution to this diagonal approximation problem, we exploit additional properties of the normal operator. First, the normal operator is positive definite yielding positive entries for the scaling vector $\mathbf{w}$. Second, the symbols of the PsDO, i.e., space and spatial frequency-dependent multipliers (in this case, curvelet coefficients) are known to be smooth for the smooth background velocity model (Stein, 1993; Stolk and DeHoop, 2006; Herrmann et al., 2008). We use these properties to formulate the diagonal approximation problem using a nonlinear least-

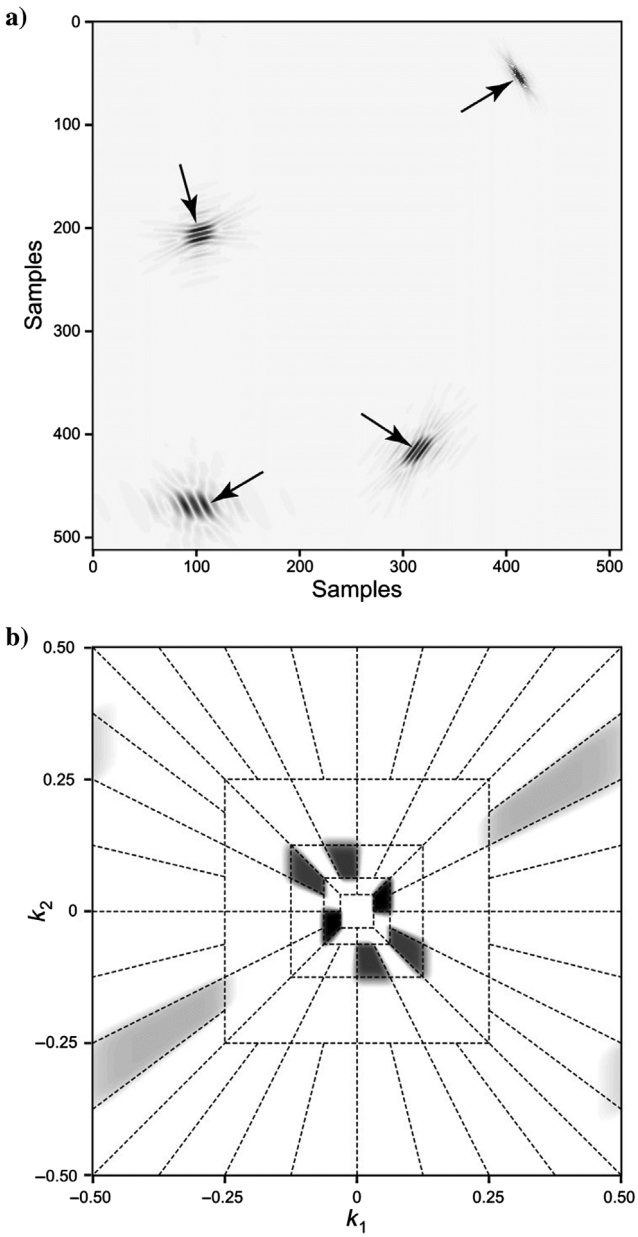


Figure 1. Spatial and frequency representation of the curvelets. (a) Four different curvelets in the spatial domain at three different scales. (b) Dyadic partitioning in the frequency domain, in which each wedge corresponds to the frequency support of a curvelet in the spatial domain. This figure illustrates the microlocal correspondence between the curvelets in the physical and Fourier domain. Curvelets are characterized by a rapid decay in the physical space and of compact support in the Fourier space. Notice the correspondence between the orientation of curvelets in the two domains. The 90° rotation is a property of the Fourier transform. Courtesy of Herrmann et al. (2008).

squares problem with enforcing positivity for the entries of the diagonal and smoothness among neighboring curvelet coefficients. The diagonal approximation reads

$$\mathbf{P_1}: \begin{cases} \hat{\mathbf{z}} = \arg\min_{\mathbf{z}} J(\mathbf{z}) = \frac{1}{2}\|\mathbf{m}_{\text{mig}} - \mathbf{C}^T \text{diag}(\mathbf{v}) e^{\mathbf{z}}\|_{\ell_2} + \frac{1}{2}\kappa \|\mathbf{L} e^{\mathbf{z}}\|_{\ell_2}, \\ \hat{\mathbf{w}} = e^{\hat{\mathbf{z}}}, \end{cases} \tag{9}$$

where $\mathbf{w}$ is replaced by $e^{\mathbf{z}}$ to ensure that the estimate of the diagonal elements of $\mathbf{W}$ are all positive. We also implement the smoothness constraint on $e^{\mathbf{z}}$. In equation 9, $\mathbf{L} = [\mathbf{D}_x \quad \mathbf{D}_z \quad \mathbf{D}_\theta]$ is a derivative operator in the curvelet domain, penalizing the fluctuations among neighboring coefficients in $e^{\mathbf{z}}$. Matrix $\mathbf{D}_{x,z}$ contains the first-order differences at each scale in the x, z-directions, and matrix $\mathbf{D}_\theta$ contains the first-order difference in the angle direction. These differences are scale-dependent because curvelet partitioning of the phase space is performed in different scales. The term κ is a smoothness control parameter. The well-behaved objective function in equation 9 can be minimized by an efficient numerical optimization method. We choose to use the limited-memory Broyden-Fletcher-Goldfarb-Shanno method (see, e.g., Nocedal and Wright, 1999), knowing the gradient of the minimization functional in equation 9 as

$$\begin{aligned} \nabla_{\mathbf{z}} J(\mathbf{z}) = {} & e^{\mathbf{z}} .* \text{diag}(\mathbf{v}) \mathbf{C} (\mathbf{C}^T \text{diag}(\mathbf{v}) e^{\mathbf{z}} - \mathbf{m}_{\text{mig}}) \\ & + \kappa e^{\mathbf{z}} .* \mathbf{L}^T \mathbf{L} e^{\mathbf{z}}, \end{aligned} \tag{10}$$

where $.*$ denotes the component-wise product of the vectors.

ALGORITHMIC DETAILS

As we show below, the algorithm requires access to matrix-free implementations of the Born modeling operator and its adjoint. For our 2D example, we use operators from reverse time migration-demigration. The recovery algorithm in this paper is designed to be part of a postprocessing procedure that enhances the amplitudes of the seismic image. The algorithm consists of two main steps, namely, the calculation of the curvelet scaling coefficients via an image-to-remigrated-image matched filtering and a subsequent recovery. Our algorithm consists of the following steps:

1) Perform the migration and the depth correction — i.e., $\mathbf{d} \mapsto \mathbf{D}\mathbf{K}^T\mathbf{d} = \tilde{\mathbf{m}}_{\text{mig}}$, with $\tilde{\mathbf{m}}_{\text{mig}}$ as the depth-corrected migrated image and $\mathbf{D} = \text{diag}(z)$, where $z = i\Delta z$, for $i = 1 : n_z$, Δz is the depth grid spacing, and n_z is the number of samples in depth. The reason for the depth correction is to boost the amplitudes of deep events in the migrated image. This ensures that after applying the normal operator on the migrated image, the deep events will be visible in the remigrated image and can contribute accordingly to the diagonal estimation.
2) Remigrate the migrated image — i.e., $\tilde{\mathbf{m}}_{\text{mig}} \mapsto \mathbf{K}^T\mathbf{K}\tilde{\mathbf{m}}_{\text{mig}} = \mathbf{m}_{\text{remig}}$.
3) Find the diagonal scaling $\mathbf{W}$ in the curvelet domain for which $\tilde{\mathbf{m}}_{\text{mig}} \approx \mathbf{C}^T\mathbf{W}\mathbf{C}\mathbf{m}_{\text{remig}}$ by solving optimization problem $\mathbf{P_1}$ (see equation 9).
4) Construct the original image by applying the approximation to migrated image (not depth corrected) $\hat{\mathbf{m}} = \mathbf{C}^T\mathbf{W}\mathbf{C}\mathbf{m}_{\text{mig}}$.

EXAMPLES

In this section, we first show the effect of the normal operator on a curvelet with different background velocity models. Then, we show our amplitude recovery method on a series of stylized examples. We use a constant-density, two-way acoustic wave equation RTM and demigration. We apply our method to the linearized Born and full synthetic data (i.e., the solution of the nonlinear wave equation) on a model with reflectors that have conflicting dips. We conclude by testing our algorithm on the large-scale BP velocity benchmark (Billette and Brandsberg-Dahl, 2005).

Curvelet invariance under the action of the normal operator

In this section, we provide an example to show how a curvelet is changed under the action of the normal operator. In this example, we apply the normal operator on a curvelet with different types of background velocity models, i.e., constant, smoothly varying, and nonsmooth background velocity models as shown in Figure 2. The velocity model for this example is a part of the North Sea velocity model with a salt body in the middle. Figure 2a shows a curvelet that has been used as the model reflectivity, Figure 2b shows the smoothly varying velocity model obtained by low-pass filtering the hard model, and Figure 2c shows the original model without any smoothness. Figure 2d–2f shows the effect of the normal operator on the curvelet shown in Figure 2a, and Figure 2g–2i shows the Fourier-domain support of Figure 2d–2f, respectively. As can be seen from these figures, the angle, orientation, and physical location of the curvelet remain invariant under the action of the normal operator with a slight support loss in the Fourier domain from left to right, which should be expected when the background model changes from the constant to the heterogeneous model. These figures confirm the theorem that the curvelet remain almost invariant under the action of the normal operator with a moderate degree of accuracy.

Conflicting-dip velocity model

We generate two sets of synthetic data. First, we use the linearized Born scattering operator (i.e., $\mathbf{d} = \mathbf{K}\mathbf{m}$ with $\mathbf{m}$ being the reflectivity and $\mathbf{K}$ being the scattering operator) and an acoustic wave propagator to generate the synthetic data set. Second, we produce data by solving the acoustic wave equation for the challenging model (Figure 3).

Linearized Born scattering data set

This example, which is shown in Figure 3a, uses a land acquisition with 32 shots between 0 and 2048 m at a 64 m spacing and 512 receivers per shot at a 4 m interval with offsets between 0 and 2048 m. To generate the linearized Born data set, we smooth this velocity model as shown in Figure 3b, and we use the difference between the smoothed and the original velocity as the reflectivity shown in Figure 3c. We choose this velocity model to demonstrate the ability of our algorithm to handle conflicting dips. As reported in the literature, situations involving no unique dips may result in erroneous amplitude correction (Symes, 2008).

Figure 4 summarizes our approximation of the normal operator with curvelet-domain scaling. We choose the reflectivity in Figure 3c and apply the scattering operator to generate the data set

(i.e., $\mathbf{d} = \mathbf{Km}$). We use the smoothed velocity model shown in Figure 3b as the background velocity for the scattering operator. The migrated result with the same background velocity model on linearized Born data is included in Figure 4a after applying the aforementioned depth correction. Notice the amplitude effects, which are more pronounced after the modeling and remigration of this image as shown in Figure 4b. These two images serve as the input to our curvelet scaling coefficient estimation. Applying curvelet-domain scaling to the reference image yields as good an approximation to the action of the normal operator as can be seen in Figure 4c.

To illustrate the importance of imposing the curvelet-domain smoothness during the scale coefficient estimation, we conduct two experiments, one without ($\kappa = 0$) and one with a smoothing regularization ($\kappa = 0.1$, which was chosen empirically and used to generate the example in Figure 4). From Figure 5, it is clear that the scale-coefficient estimation becomes biased toward the reference vector. This figure shows the curvelet transform of the reference vector (Figure 5a), a diagonal estimation without smoothness (Figure 5b) and with smoothness constraints (Figure 5c). As we can see, the curvelet coefficients for the reference image vary with scales that become finer when moving from the center outward. The angular wedges are roughly localized in positions that correspond to the angle of the curvelets. In other words, if this angle is close to the dip of the event, the curvelet coefficient will be large. This domain nicely decomposes the reflectivity into different scales and angles. This explains the success of curvelets in representing images with wavefronts and justifies our choice of this domain to do the scaling. However, as Figure 5a and 5b indicates, it is important to note how to estimate the scaling coefficients. Contrary to the reflectivity itself, with its wavefronts and hence localized features in the curvelet space, the scaling has to be smooth for smooth background velocity models for which the symbols of $\mathbf{\Psi}$ are smooth (Herrmann et al., 2008). Comparing Figure 5b and 5c, we see large coefficients in the top row that correspond to curvelets with the wrong angles in Figure 5b, whereas these coefficients do not exist in the diagonal estimation with smoothing (Figure 5c). The final result of our amplitude recovery for this diagonal is included in Figure 6. As one can see, the amplitudes are restored reasonably well even in the areas of conflicting dip in the reflectivity.

Synthetic data set

The velocity model in Figure 3a is also used to generate nonlinear synthetic data up to 60 Hz using a finite-difference constant-density acoustic code for the same land acquisition configuration. After applying a half-time integration and removing the direct waves from the data set, we follow the same process as mentioned in the previous section. As we can see from Figure 7, our recovery method also performs well on full synthetic data.

Large-scale synthetic example

The next example is based on the BP salt model (Billette and Brandsberg-Dahl, 2005). The BP model has been designed to

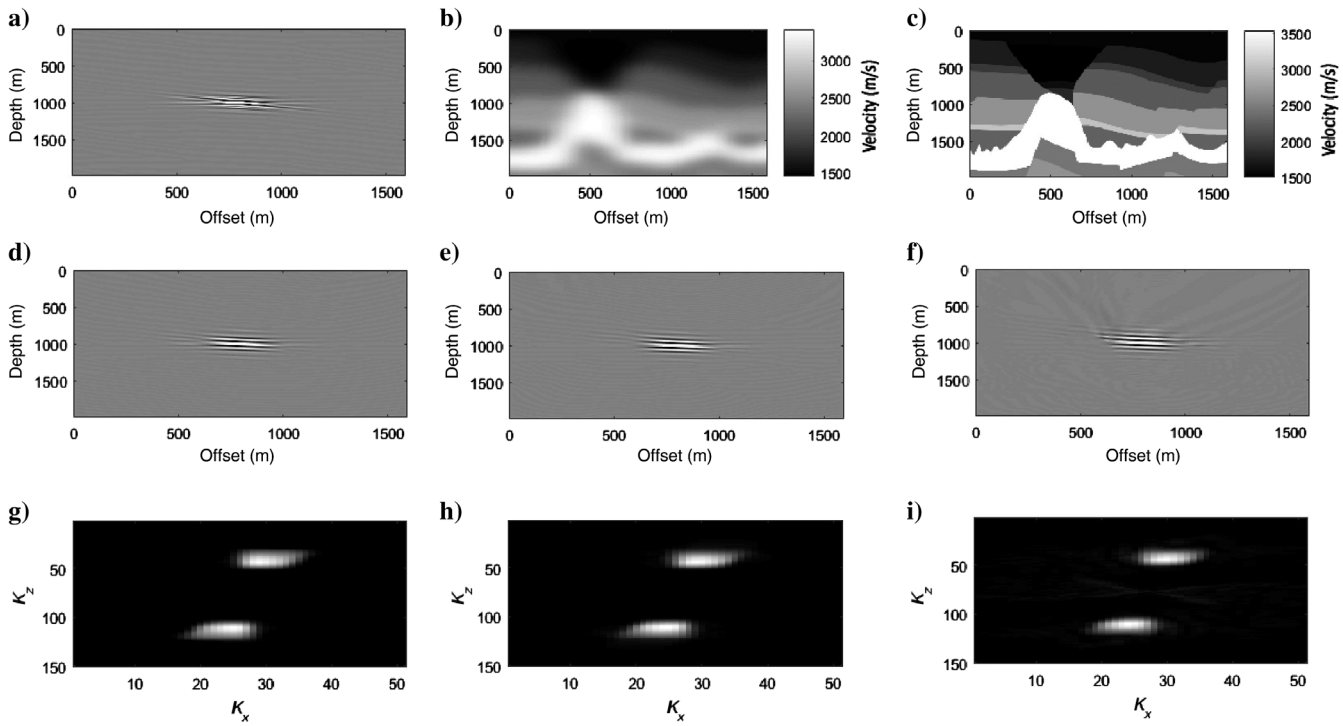


Figure 2. Action of the normal operator on a curvelet. (a) A curvelet as reflectivity, (b) smooth North Sea model, (c) hard North Sea model, (d) action of the normal operator on a curvelet with the constant velocity model (2500 m/s), (e) action of the normal operator on a curvelet with the smooth velocity model shown in (b), (f) action of the normal operator on a curvelet with the hard velocity model shown in (c), (g) Fourier domain representation of (d), (h) Fourier domain representation of (e), and (i) Fourier domain representation of (f).

exhibit illumination problems due to the complex salt shape, with a tooth-shaped salt top found in the area (Figure 8a). The data set was calculated using finite-difference modeling code with a free-surface boundry condition that generated free-surface multiples.

The data are generated with a streamer configuration, using a 15 km streamer with a 12.5 m group interval and a 50 m shot interval and 0 m minimum offset, and are recorded for 14 s with a 6 ms sampling interval. The dominant frequency is 27 Hz, and data can be whitened up to 54 Hz. The low-cut frequency is 3 Hz, and the wavelet is causal and has not been zero phased. In this example, a total of 1340 shots were generated, each with 1201 receivers.

The center part of the model (the tooth part) is exclusively extrasalt, and it is meant to represent a geologic setting with shallow gas and localized shallow anomalies (Figure 8b). However, because of the lensing effect, this part turned out to be extremely difficult to estimate for imaging methods. The geology in this part of the model is common in areas such as the Caspian Sea, offshore Trinidad, and in the North Sea. The velocity field has significant variations in the long-wavelength component and several low-velocity anomalies in the shallow section. The size, shape, and velocity of the anomalies

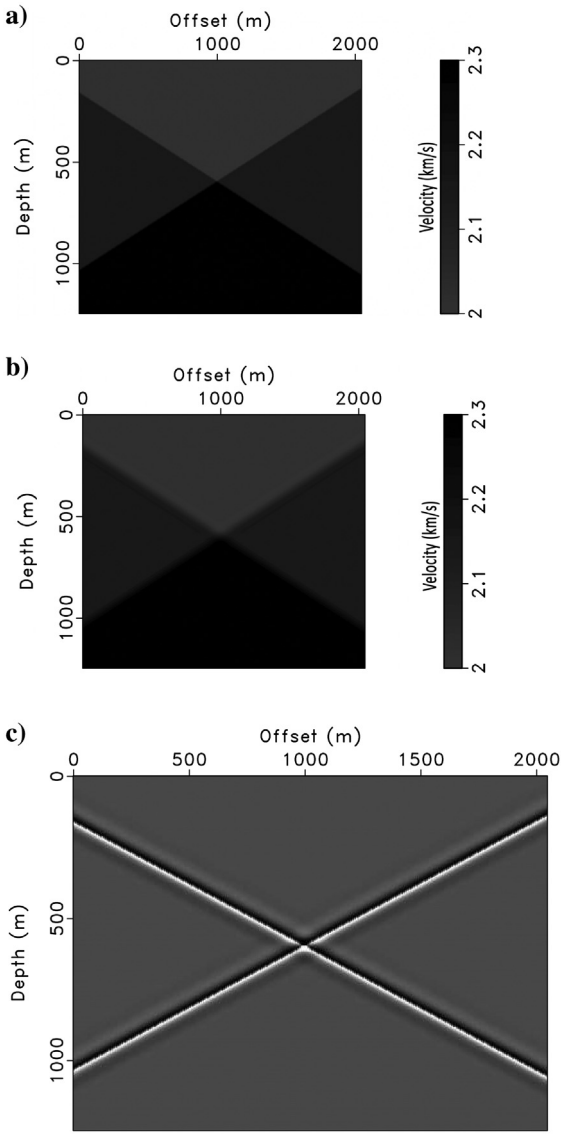


Figure 3. Conflicting dip velocity model and reflectivity. (a) Velocity model, (b) smooth velocity model used to generate our example, and (c) reflectivity generated by subtracting the smooth velocity model from the original one.

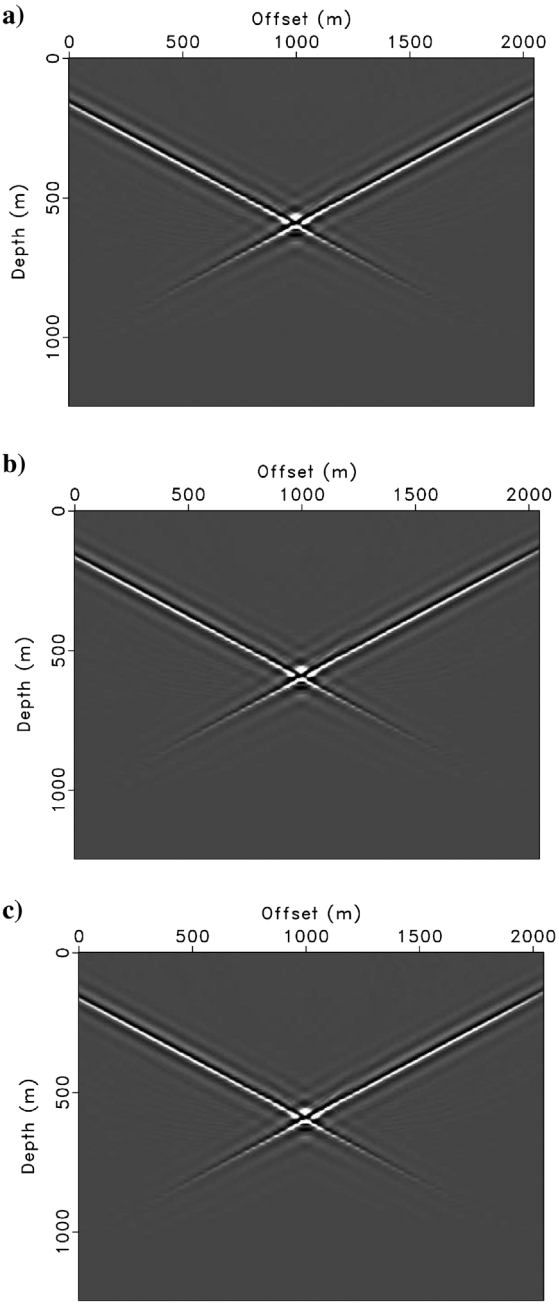


Figure 4. Example of amplitude recovery using the linearized Born modeling data set. (a) Reference image (depth-corrected migrated image), (b) the action of the normal operator on the reference image (i.e., $\mathbf{\Psi r}$), and (c) the approximate normal operator on the reference image (i.e., $\mathbf{C}^T\mathbf{D}_{\mathbf{\Psi}}\mathbf{Cr}$), with an approximation error of 3.71%.

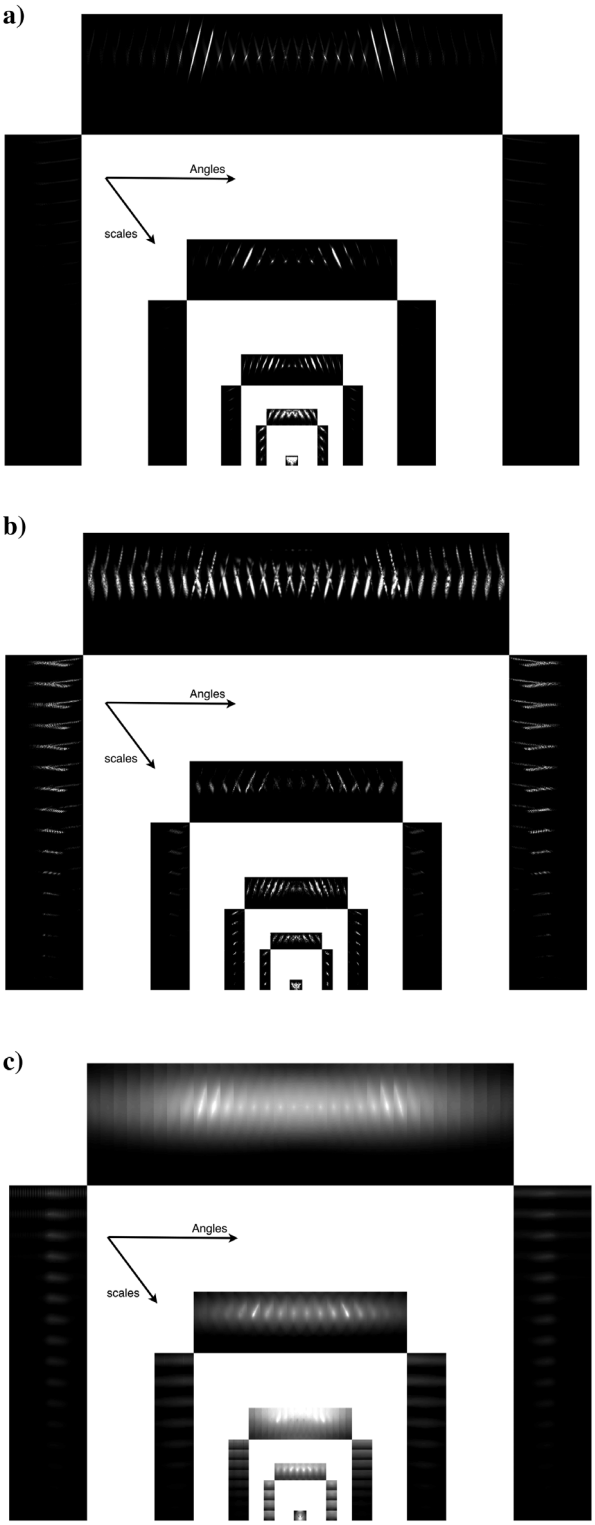


Figure 5. Curvelet-domain representation of the curvelet vector obtained from CurveLab. (a) Reference vector, (b) scaling coefficients without smoothing constraints, and (c) scaling coefficients with smoothing constraints. The different subimages represent the curvelet coefficients at different scales (coarsest in the center) and different angles.

are variable (Billette and Brandsberg-Dahl, 2005). We focus in that area of the model and show that our recovery method enhances this part of the model. We perform some pre-/postprocessing before/after migrating the data. For this purpose, the first step is to remove free surface multiples using the surface-related multiple elimination method Berkhout and Verschuur (1997) and mute the direct wave in the data set. The next step is to remove the dipping wave by band-pass filtering the migrated image and to correct the migrated image with an illumination map.

Figures 9, 10, 11, and 12 show the application of our method on this model. Figure 9a shows the migrated image. We use this image as the reference vector for the amplitude recovery method discussed in the "Algorithmic details" section. Figure 9b shows the enhanced image that is the result of our proposed amplitude recovery method. Note the improvement of the enhanced image compared to the migrated image especially in the locations marked by the arrowheads. As the arrowheads indicate, improvements are visible on the top of the salt with less ringing effect and more continuity along the top salt reflector, under the salt tooth with a boost in the reflector energy and better visibility under the salt tooth, and under the velocity lens in the middle of the model with better energy focusing and reflector visibility.

To show that our methodology is not simply a boost in the amplitude in the enhanced image, we set up the following experiment.

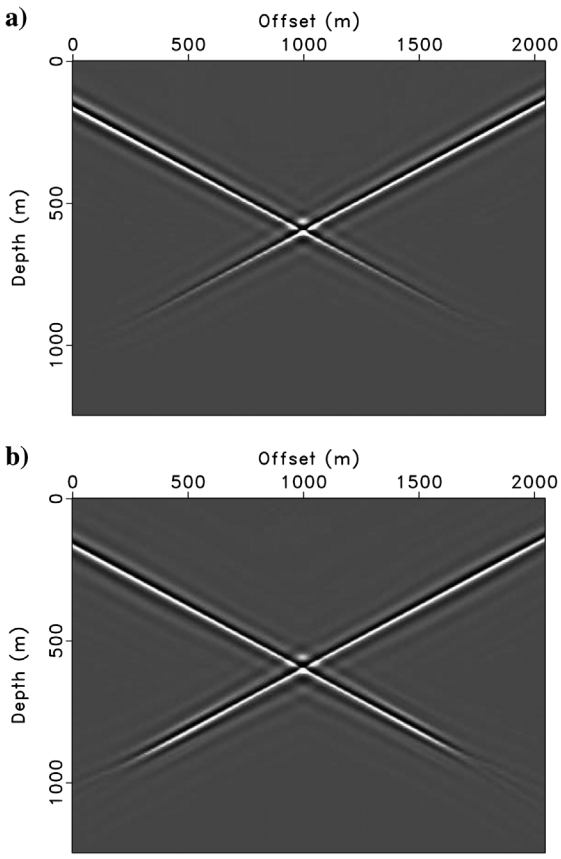


Figure 6. Example of amplitude recovery using the linearized Born modeling data set. (a) Reference image (the depth-corrected migrated image) and (b) enhanced image. Notice the enhancement along the deeper part of the reflectors.

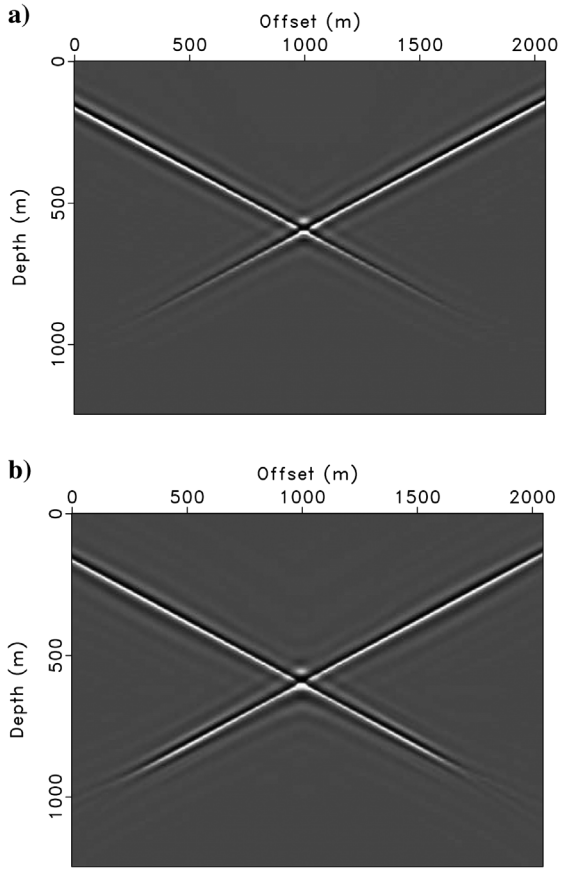


Figure 7. Example of amplitude recovery using acoustic wave propagation data set (not linearized). (a) Reference image (the depth-corrected migrated image) and (b) enhanced image.

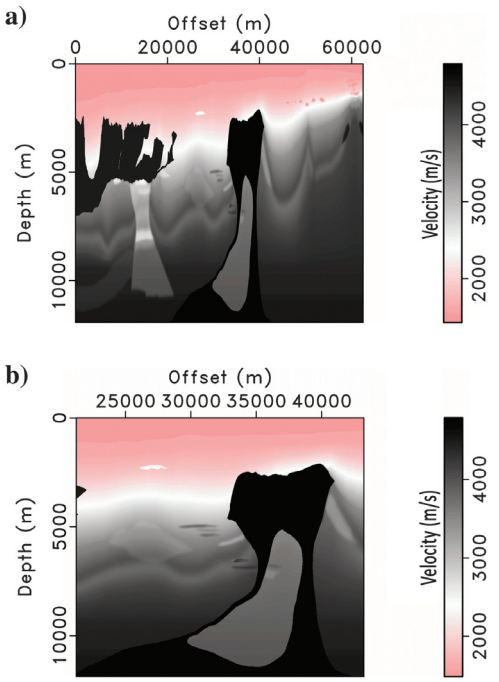


Figure 8. The BP velocity model. (a) The full model and (b) the tooth part of the BP model, area of interest.

Figure 10 shows the vertical derivative in the BP density model. This figure can be considered as the benchmark to compare how well our recovery method performs. Considering the fact that the seismic image is ideally a representative of the perturbation in the reflectivity (not the reflectivity itself) and in the BP model most

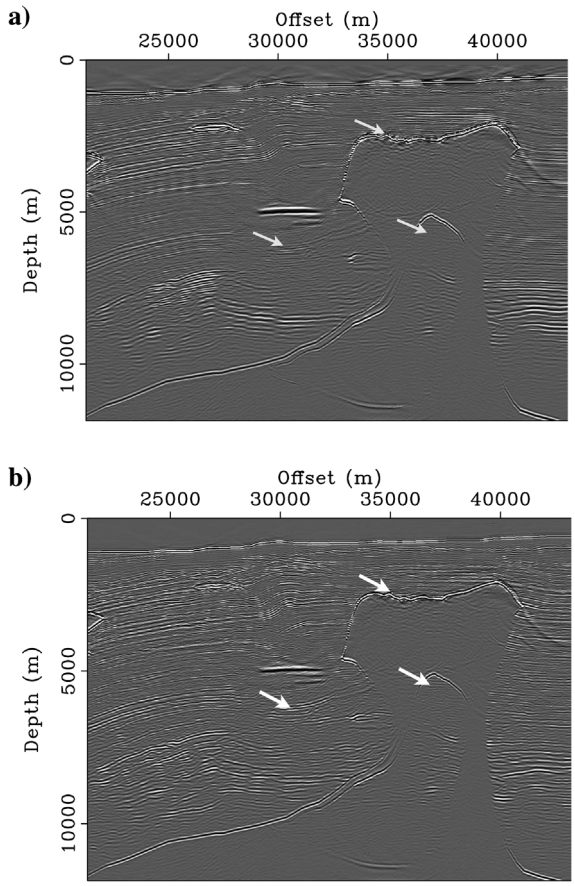


Figure 9. Our recovery result: (a) BP migrated image (illumination corrected) and (b) BP enhanced image.

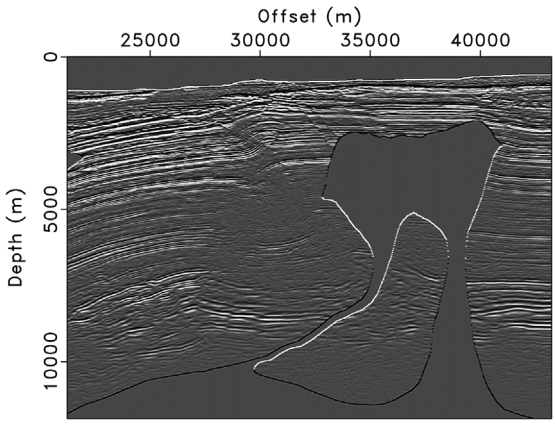


Figure 10. Differentiated density model.

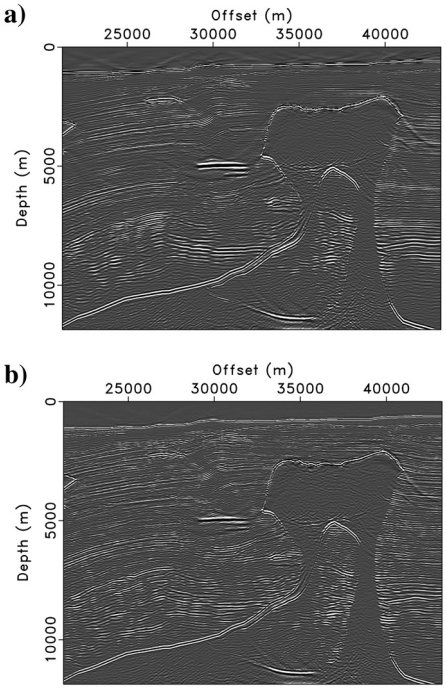


Figure 11. Result of AGC on the (a) migrated image and (b) enhanced image.

variation occurs in the density model instead of velocity (Billette and Brandsberg-Dahl, 2005), Figure 10 shows a good benchmark to compare with our final result. The closer the enhanced image is to the benchmark image means that our recovery method has a better performance.

To have an unbiased comparison between the migrated image and the enhanced image, we apply automatic gain control (AGC) to both images. Figure 11 shows the result of AGC on the migrated and the enhanced image.

Figures 12 and 13 display magnifications of two different areas shown in the images in Figure 11 to show how well our method performs. Figure 12a and 12b shows the migrated and the enhanced image around the potential reservoir zone, under the velocity lens, located at approximately (5000, 30,000) m, respectively. Comparing Figure 12b, the enhanced image, and Figure 12a, the migrated image, with the benchmark density model in Figure 12d, it is visible that the enhanced image could capture most density variations that exist in the density model with reasonably good amplitude preservation, whereas in the migrated image some important regions are faded away specially under the velocity lens.

Figure 13a and 13b shows the migrated and the enhanced image under the tooth-shape salt flank, respectively. Compared with the density model, shown in Figure 13d, we observe promising amplitude recovery results and noticable amplitude recovery in the enhanced image.

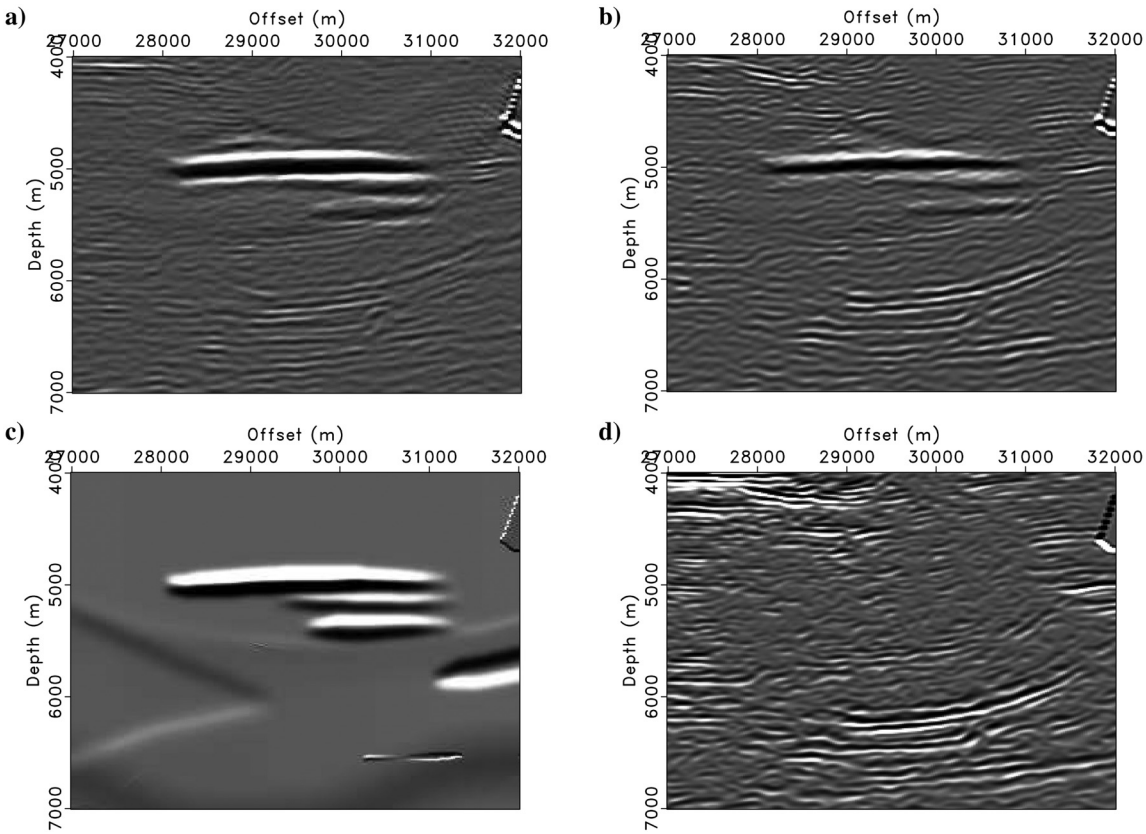


Figure 12. Comparison between the migrated and the enhanced image in potential reservior zones. (a) Migrated, (b) enhanced, (c) velocity differentiated along the vertical access, and (d) density differentiated along the vertical access.

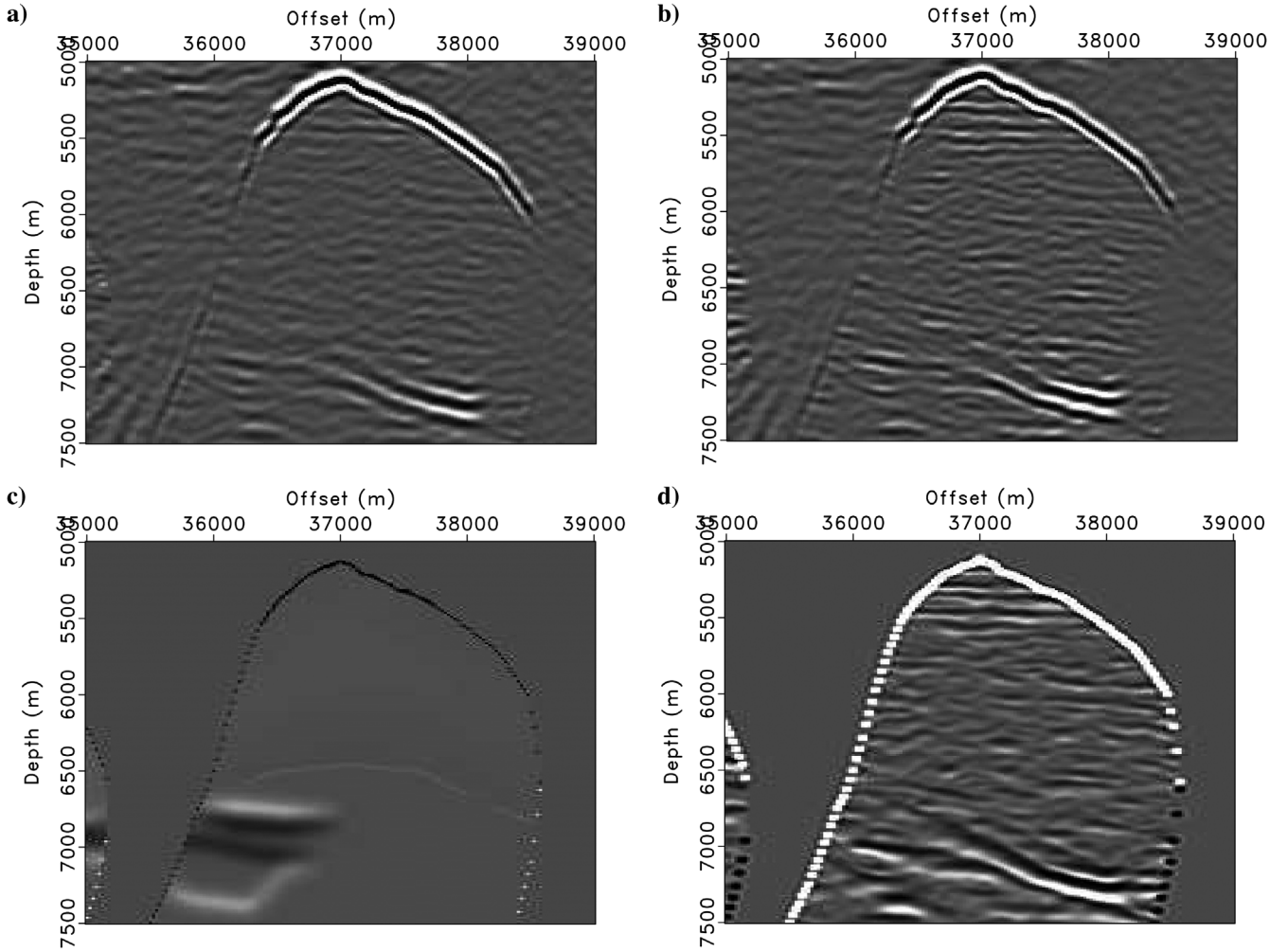


Figure 13. Comparison between the migrated and enhanced image in the undersalt zone. (a) Migrated, (b) enhanced, (c) velocity differentiated along the vertical access, and (d) density differentiated along the vertical access.

CONCLUSION

The migration section is an inaccurate image of the geologic reflectivity distribution if it is imaged incorrectly. With a coarsely sampled and narrow distribution of sources and receivers, the migration output introduces inaccuracy into the amplitude of the reflectors. Furthermore, migration is not the exact inverse of the forward-modeling operator, so the migration image is only a rough approximation of the reflectivity image.

To mitigate these effects, we present a fast and robust approach to approximation of the normal operator and use it to recover the amplitudes in the migrated image. We formulate the approximation of the normal operator as an eigenvalue-like decomposition with curvelets as eigenvectors. Our proposed method is faster than other methods because our method evaluates the normal operator only once. The method presented in this paper comprises the two important properties of the curvelet transform, namely, the smoothness of its coefficient and the invariance of its elements under the action of the normal operator. This combination allows us to formulate a stable recovery method for seismic amplitudes.

Compared with other approaches for migration amplitude recovery, some noteworthy improvements in this method are, first, speeding up the calculation of the normal operator by diagonalizing it in the curvelet domain, second, designing an efficient approximation that takes into account a laterally variant velocity model and steep reflectors, third, replacing the ad hoc or trial-and-error estimation of the migration amplitude with a method that has a more theoretical and practical background, and, fourth, bringing the amplitude correction problem within the context of stable recovery.

The results of applying our method on synthetic data suggest that our migration amplitude recovery can be useful in eliminating migration artifacts and in the seismic image. The recovered images show the partial elimination of noise, improved spatial resolution, and enhanced reflectivity amplitude.

DATA AND MATERIALS AVAILABILITY

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

APPENDIX A

REVERSE TIME WAVE EQUATION MIGRATION

Most migration operators are defined to be the adjoint of what is called the scattering operator. This assumption is also true for RTM.

Single scattering

The causal acoustic Green's function $G(\mathbf{x}, t; \mathbf{x}_s), \mathbf{x} \in \mathbf{R}^3$ for a point source at $\mathbf{x} = \mathbf{x}_s$ is the solution of

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 G}{\partial t^2}(\mathbf{x}, t; \mathbf{x}_s) - \nabla_x^2 G(\mathbf{x}, t; \mathbf{x}_s) = \delta(\mathbf{x} - \mathbf{x}_s)\delta(t), \quad \text{(A-1)}$$

with $G = 0$, $t < 0$, and v is the acoustic wave velocity field.

Denote by $m(\mathbf{x}) = \delta v(\mathbf{x})/v(\mathbf{x})$ a relative perturbation of the velocity field. Then, linearization of the wave equation yields for the corresponding perturbation of the Green's function

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 \delta G}{\partial t^2}(\mathbf{x}, t; \mathbf{x}_s) - \nabla_x^2 \delta G(\mathbf{x}, t; \mathbf{x}_s) = \frac{2m(\mathbf{x})}{v^2(\mathbf{x})}\frac{\partial^2}{\partial t^2} G(\mathbf{x}, t; \mathbf{x}_s), \quad \text{(A-2)}$$

whose solution has the integral representation, at the source and receiver points $\mathbf{x}_r, \mathbf{x}_s$, respectively, as

$$\delta G(\mathbf{x}_r, t; \mathbf{x}_s) = \frac{\partial^2}{\partial t^2}\int dx \int d\tau \frac{2m(\mathbf{x})}{v^2(\mathbf{x})} G(\mathbf{x}, t - \tau; \mathbf{x}_r) \times G(\mathbf{x}, \tau; \mathbf{x}_s). \quad \text{(A-3)}$$

Shot-geophone modeling and migration

The single-scattered wavefield is the time convolution of G with a source wavelet. The main concern of this paper is the kinematic relationships between the data and the image; thus, we ignore the filtering effect of the source functional and replace it with the delta function. This replacement of the source by an impulse does not violate any of our assumptions regarding the adjoint-state method, thus the Born modeling operator $K[v]$ is

$$K[v]m(\mathbf{x}) = \delta G(\mathbf{x}_r, t; \mathbf{x}_s). \quad \text{(A-4)}$$

The crux of our amplitude recovery method relies on the shot-geophone migration operator to be the adjoint of the shot-geophone modeling operator. The derivation of the adjoint reverse time implementation is a minor variation on the usual implementation of RTM (the "adjoint state method"; see, e.g., Lailly and Bednar, 1983; Whitmore, 1983; Tarantola, 1984; Yoon et al., 2003; Symes, 2007). The result is

$$K^*[v]d(\mathbf{x}_r, t; \mathbf{x}_s) = \hat{m}(\mathbf{x}) = -\int dx_s \int_0^T dt 2v(\mathbf{x}) \times q(\mathbf{x}, t; \mathbf{x}_s)\frac{\partial^2 G}{\partial t^2}(\mathbf{x}, t; \mathbf{x}_s), \quad \text{(A-5)}$$

where the adjoint state or backpropagated field $q(\mathbf{x}, t; \mathbf{x}_s)$ satisfies $q = 0, t > T$, and

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 q}{\partial t^2}(\mathbf{x}, t; \mathbf{x}_s) - \nabla_x^2 q(\mathbf{x}, t; \mathbf{x}_s) = \int dx_r d(\mathbf{x}_r, t; \mathbf{x}_s)\delta(\mathbf{x} - \mathbf{x}_r). \quad \text{(A-6)}$$

The migration operator defined by the above equations is the RTM operator. Symes et al. (2003) show that the migration operator, which is defined in equations A-5 and A-6, is the adjoint of the modeling operator defined in equation A-3. The RTM that is used for this work is the adjoint of the modeling operator and is properly tested in the discrete sense. By having the migration and modeling operators properly set, we can proceed with our amplitude-recovery method.

REFERENCES

Berkhout, A., and D. Verschuur, 1997, Adaptive surface-related multiple elimination: Geophysics, **62**, 1586–1595, doi: 10.1190/1.1444261.

Billette, F. J., and S. Brandsberg-Dahl, 2005, The 2004 BP velocity benchmark: 67th Annual International Conference and Exhibition, EAGE, Extended Abstracts, cp-1, doi: 10.3997/2214-4609-pdb.1.B035.

Candes, E., L. Demanet, D. Donoho, and L. Ying, 2006, Fast discrete curvelet transforms: SIAM Multiscale Modeling & Simulation, **5**, 861–899, doi: 10.1137/05064182X.

Chattopadhyay, S., and G. A. McMechan, 2008, Imaging conditions for prestack reverse-time migration: Geophysics, **73**, no. 3, S81–S89, doi: 10.1190/1.2903822.

Chauris, H., and E. Cocher, 2017, From migration to inversion velocity analysis: Geophysics, **82**, no. 3, S207–S223, doi: 10.1190/geo2016-0359.1.

Claerbout, J., 1992, Earth soundings analysis, processing versus inversion: Blackwell Scientific Publications 6.

Claerbout, J., and D. Nichols, 1994, Spectral preconditioning: Technical Report 82, Stanford Exploration Project, Stanford University, 82, 1–187.

Costa, J., F. S. Neto, M. Alcantara, J. Schleicher, and A. Novais, 2009, Obliquity-correction imaging condition for reverse time migration: Geophysics, **74**, no. 3, S57–S66, doi: 10.1190/1.3110589.

da Silva Neto, F. A., J. C. Costa, J. Schleicher, and A. Novais, 2011, 2.5-D reverse-time migration: Geophysics, **76**, no. 4, S143–S149, doi: 10.1190/1.3571272.

Donoho, D., 1995, Nonlinear solution of linear inverse problems by wavelet-vaguelette decomposition: Applied and Computational Harmonic Analysis, **2**, 101–126, doi: 10.1006/acha.1995.1008.

Gray, S., 1997, True amplitude seismic migration: A comparison of three approaches: Geophysics, **62**, 929–936, doi: 10.1190/1.1444200.

Guitton, A., 2004, Amplitude and kinematic corrections of migrated images for nonunitary imaging operators: Geophysics, **69**, 1017–1024, doi: 10.1190/1.1778244.

Herrmann, F., P. Moghaddam, and C. Stolk, 2008, Sparsity- and continuity-promoting seismic image recovery with curvelet frames: Applied and Computational Harmonic Analysis, **24**, 150–173, doi: 10.1016/j.acha.2007.06.007.

Herrmann, F. J., and X. Li, 2012, Efficient least-squares imaging with sparsity promotion and compressive sensing: Geophysical Prospecting, **60**, 696–712, doi: 10.1111/j.1365-2478.2011.01041.x.

Hou, J., and W. W. Symes, 2015, An approximate inverse to the extended born modeling operator: Geophysics, **80**, no. 6, R331–R349, doi: 10.1190/geo2014-0592.1.

Kuel, H., and M. Sacchi, 2003, Least-squares wave-equation migration for AVP/AVA inversion: Geophysics, **68**, 262–273, doi: 10.1190/1.1543212.

Lailly, P., and J. Bednar, 1983, The seismic inverse problem as a sequence of before stack migrations: Conference on Inverse Scattering: Theory and Application, 206–220.

Li, Y., and H. Chauris, 2018, Coupling direct inversion to common-shot image-domain velocity analysis: Geophysics, **83**, no. 5, R497–R514, doi: 10.1190/geo2017-0825.1.

Liu, Q., and D. Peter, 2018, One-step data-domain least-squares reverse time migration: Geophysics, **83**, no. 4, R361–R368, doi: 10.1190/geo2017-0622.1.

Lu, S., X. Li, A. Valenciano, N. Chemingui, and C. Cheng, 2017, Least-squares wave-equation migration for broadband imaging: 79th Annual International Conference and Exhibition, EAGE, Extended Abstracts, 1–5.

Mulder, W., and R.-E. Plessix, 2004, A comparison between one-way and two-way wave equation migration: Geophysics, **69**, 1491–1504, doi: 10.1190/1.1836822.

Nocedal, J., and W. Wright, 1999, Numerical optimization: Springer Verlag.

Plessix, R., and W. Mulder, 2004, Frequency-domain finite-difference amplitude-preserving migration: Geophysical Journal International, **157**, 975–987, doi: 10.1111/j.1365-246X.2004.02282.x.
Rickett, J., 2003, Illumination-based normalization for wave-equation depth migration: Geophysics, **68**, 1371–1379, doi: 10.1190/1.1598130.
Schleicher, J., J. C. Costa, and A. Novais, 2008, A comparison of imaging conditions for wave-equation shot-profile migration: Geophysics, **73**, no. 6, S219–S227, doi: 10.1190/1.2976776.
Stein, E., 1993, Harmonic analysis: Vanderbilt University Press.
Stolk, C., and M. De-Hoop, 2006, Seismic inverse scattering in the downward continuation approach: Wave Motion, **43**, 579–598, doi: 10.1016/j.wavemoti.2006.05.003.
Symes, W. W., 2007, Reverse time migration with optimal checkpointing: Geophysics, **72**, no. 5, SM213–SM221, doi: 10.1190/1.2742686.
Symes, W. W., 2008, Approximate linearized inversion by optimal scaling of prestack depth migration: Geophysics, **73**, no. 2, R23–R35, doi: 10.1190/1.2836323.
Symes, W. W., C. Stolk, B. Biondi, and F. Gao, 2003, Reverse time shot-geophone migration: SEP-50, 151-69.
Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, 1259–1266, doi: 10.1190/1.1441754.
Ten-Kroode, A., D. Smith, and A. Verdel, 1998, A microlocal analysis of migration: Wave Motion, **28**, 149–172, doi: 10.1016/S0165-2125(98)00004-3.
Wang, M., S. Huang, and P. Wang, 2017, Improved iterative least-squares migration using curvelet-domain Hessian filters: 87th Annual International Meeting, SEG, Expanded Abstracts, 4555–4560, doi: 10.1190/segam2017-17783350.1.
Wang, P., A. Gomes, Z. Zhang, and M. Wang, 2016, Least-squares RTM: Reality and possibilities for subsalt imaging: 86th Annual International Meeting, SEG, Expanded Abstracts, 4204–4209, doi: 10.1190/segam2016-13867926.1.
Whitmore, N., 1983, Iterative depth migration by backward time propagation: 53rd Annual International Meeting, SEG, Expanded Abstracts, 382–385, doi: 10.1190/1.1893867.
Yang, J.-D., J.-P. Huang, X. Wang, and Z.-C. Li, 2015, An amplitude-preserved adaptive focused beam seismic migration method: Petroleum Science, **12**, 417–427, doi: 10.1007/s12182-015-0044-7.
Yoon, K., C. Shin, A. Suh, S. Lines, and S. Hong, 2003, 3-D reverse-time migration using the acoustic wave equation: An experience with the SEG/EAGE data set: The Leading Edge, **22**, 38–41, doi: 10.1190/1.1542754.
Zhang, Y., S. Xu, N. Bleistein, and G. Zhang, 2007, True-amplitude, angle-domain, common-image gathers from one-way wave-equation migrations: Geophysics, **72**, no. 1, S49–S58, doi: 10.1190/1.2399371.

Hamideh Sanavi received an M.S. (2018) in geophysics from the Ferdowsi University of Mashhad, Mashhad, Iran. She has demonstrated excellence in researching, analyzing, and presenting information as a research assistant at the Ferdowsi University of Mashhad, focusing on the improvement of subsurface imaging technologies (focusing on prestack depth and prestack time migration techniques) since 2018.

Peyman P. Moghaddam received a B.S. (1995) and an M.S. (1998) from the Amirkabir University of Technology, Tehran, Iran, and a Ph.D. (2010) in geophysics from the University of British Columbia, Vancouver, Canada. After research positions at the University of British Columbia and Delft University of Technology in Delft, Netherlands, he worked for TGS in Houston, Texas, from 2013 till 2015. In 2015, he joined the Ferdowsi University of Mashhad, Faculty of Science, where he is now an assistant professor in geophysics. He is also a business development manager at BQI-FUM-Innovation Co. Ltd., a research-based company located in London, UK, which provides geoscience services. His research interests include seismic modeling, imaging and inversion, and high-performance computing methods for computationally intensive geoscientific applications.

Felix J. Herrmann graduated from the Delft University of Technology in 1992 and received a Ph.D. (1997) in engineering physics from that same institution. After research positions at Stanford University and the Massachusetts Institute of Technology, he became back in 2002 faculty at the University of British Columbia. In 2017, he joined the Georgia Institute of technology where he is now a Georgia Research Alliance Scholar Chair in Energy, cross-appointed between the Schools of Earth and Atmospheric Sciences, Computational Science and Engineering, and Electrical and Computer Engineering. His cross-disciplinary research program spans several areas of computational imaging including seismic, and more recently, medical imaging. Over his career, he has been responsible for several cost-saving innovations in industrial time-lapse seismic data acquisition and wave-equation based imaging. In 2019, he was the SEG Distinguished Lecturer and in 2020 the recipient of the SEG Reginald Fessenden Award. At Georgia Tech, he leads the Seismic Laboratory for imaging and modeling, and he is a cofounder/director of the Center for Machine Learning for Seismic (ML4Seismic).

Modelling Seismic Wave Propagation for Geophysical Imaging

J. Virieux, V. Etienne, V. Cruz-Atienza, R. Brossier, E. Chaljub, O. Coutant, S. Garambois, D. Mercerat, V. Prieux, S. Operto, et al.

13

Modelling Seismic Wave Propagation for Geophysical Imaging

Jean Virieux et al.[1]*, Vincent Etienne et al.[2]†and Victor Cruz-Atienza et al.[3]‡
[1]*ISTerre, Université Joseph Fourier, Grenoble*
[2]*GeoAzur, Centre National de la Recherche Scientifique, Institut de Recherche pour le développement*
[3]*Instituto de Geofisica, Departamento de Sismologia, Universidad Nacional Autónoma de México*
[1,2]*France*
[3]*Mexico*

1. Introduction

The Earth is an heterogeneous complex media from the mineral composition scale ($\simeq 10^{-6}m$) to the global scale ($\simeq 10^{6}m$). The reconstruction of its structure is a quite challenging problem because sampling methodologies are mainly indirect as potential methods (Günther et al., 2006; Rücker et al., 2006), diffusive methods (Cognon, 1971; Druskin & Knizhnerman, 1988; Goldman & Stover, 1983; Hohmann, 1988; Kuo & Cho, 1980; Oristaglio & Hohmann, 1984) or propagation methods (Alterman & Karal, 1968; Bolt & Smith, 1976; Dablain, 1986; Kelly et al., 1976; Levander, 1988; Marfurt, 1984; Virieux, 1986). Seismic waves belong to the last category. We shall concentrate in this chapter on the forward problem which will be at the heart of any inverse problem for imaging the Earth. The forward problem is dedicated to the estimation of seismic wavefields when one knows the medium properties while the inverse problem is devoted to the estimation of medium properties from recorded seismic wavefields.

The Earth is a translucid structure for seismic waves. As we mainly record seismic signals at the free surface, we need to consider effects of this free surface which may have a complex topography. High heterogeneities in the upper crust must be considered as well and essentially in the weathering layer zone which complicates dramatically the waveform and makes the focusing of the image more challenging.

Among the main methods for the computation of seismic wavefields, we shall describe some of them which are able to estime the entire seismic signal considering different approximations as acoustic or elastic, isotropic or anisotropic, and attenuating effects. Because we are interested in seismic imaging, one has to consider methods which should be efficient especially for the many-sources problem as thousands of sources are required for imaging. These sources could be active sources as explosions or earthquakes. We assume that their

*Romain Brossier, Emmanuel Chaljub, Olivier Coutant and Stéphane Garambois
†Diego Mercerat, Vincent Prieux, Stéphane Operto and Alessandra Ribodetti
‡Josué Tago

distribution are known spatially as punctual sources and that the source time function is the signal we need to reconstruct aside the medium properties.

Asymptotic methods based on the high frequency ansatz (see (Virieux & Lambaré, 2007) for references or textbooks (Červený, 2001; Chapman, 2004)) and spectral methods based on a spatial and time Fourier transformations (Aki & Richards, 2002; Cagniard, 1962; de Hoop, 1960; Wheeler & Sternberg, 1968) are efficient methods which are difficult to control: whispering galeries for flat layers are efficiently considered using spectral methods. These two methods may be used either for local interpretation of specific phases or as efficient alternatives when media is expected to be simple. They could be used as well for scattering inverse problems. In the general heterogeneous case, we have to deal with volumetric methods where the medium properties are described through a volume while seismic wave fields satisfy locally partial differential equations. Although one may consider boundaries as the free surface or the water/solid interface, we may consider that variations of the medium properties are continuous at the scale of the wavelength which we want to reconstruct: the best resolution we could expect is half the wavelength (Williamson & Worthington, 1993). Therefore a volumetric grid discretization is preferred where numerical expressions of boundary conditions should be mostly implicit through properties variations.

A quite popular method because of this apparent simplicity is the finite difference method where partial derivatives are transformed into finite difference expressions as soon as the medium has been discretized into nodes: discrete equations should be exactly verified. We shall consider first this method as it is an excellent introduction to numerical methods and related specific features. We will consider both time and frequency approaches as they have quite different behaviours when considering seismic imaging strategies.

Applications will enhance the different properties of this numerical tool and the caveats we must avoid for the various types of propagation we need.

Another well-known approach is the finite element method where partial differential equations are asked to be fulfilled in a average way (to be defined) inside elements paving the entire medium. We shall concentrate into the discontinuous Galerkin method as it allows to mix acoustic and elastic wave propagation into a same formalism: this particular method shares many features of finite element formalism when describing an element, but differs by the way these elements interact each other. We avoid the description of the continuous finite element method for compactness and differences will be pointed out when necessary. Again, we shall discuss both time-domain and frequency-domain approaches.

Applications will illustrate the different capabilities of this technique and we shall illustrate what are advantages and drawbacks compared to finite difference methods while specific features will be identified compared to continuous finite element methods.

We shall conclude on the strategy for seismic imaging when comparing precision of solutions and numerical efforts for both volumetric methods.

2. Equations of seismic wave propagation

In a heterogeneous continuum medium, seismic waves verify partial differential equations locally. Integral equations may provide an alternative for the evolution of seismic fields either

in the entire domain or at the scale of an elementary element of a given mesh describing the medium structure.

Fundamental laws of dynamics require the conservation of linear and angular momentum in a Galilean reference frame. In the continuum, a force applied across a surface, oriented by the unit normal $\mathbf{n}$ at a given position $\mathbf{x} = (x, y, z)$ in a Cartesian coordinate system (O, x, y, z), by one side of the material on the other side defines the traction vector $t_i = \sigma_{ij} n_j$ where the second-rank stress tensor σ has been introduced. The conservation of the angular momentum makes the stress tensor symmetrical $\sigma_{ij} = \sigma_{ji}$. We shall introduce as well a volumetric force per unit mass at the current position denoted as $\mathbf{f} = (f_x, f_y, f_z)$. The conservation of linear momentum allows one to write the acceleration of the displacement motion $\mathbf{u}(\mathbf{x})$ of a given particle at the current position as

$$\rho(\mathbf{x}) \frac{\partial^2 u_i}{\partial t^2} = \frac{\partial \sigma_{ik}}{\partial x_k} + \rho(\mathbf{x}) f_i, \tag{1}$$

where the density is denoted by $\rho(\mathbf{x})$.

Aside the translation and the rotation transformations preserving the distances inside the body we consider, the deformation of the continuum body is described by defining a strain tensor ϵ expressed as

$$\epsilon_{ij} = \frac{1}{2} \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right). \tag{2}$$

The symmetrical definition of the deformation ensures that no rigid-body rotations are included. The particle motion is decomposed into a translation, a rotation and a deformation: the two formers transformations preserve distances inside the solid body while the third one does not preserve distances, inducing stress variations inside the solid body. In the framework of linear elasticity, there is a general linear relation between the strain and stress tensors by introducing fourth-rank tensor c_{ijkl} defined as follows

$$\sigma_{ij} = c_{ijkl} \epsilon_{kl}. \tag{3}$$

Because of symmetry properties of stress and strain tensors, we have only 36 independent parameters among the 81 elastic coefficients while the positive strain energy leads to a further reduction to 21 independent parameters for a general anisotropic medium. For the particular case of isotropic media, we end up with two coefficients which can be the Lamé coefficients λ and μ. The second one is known also as the rigidity coefficient as it characterizes the mechanical shear mode of deformation. The following expression of elastic coefficients,

$$c_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}), \tag{4}$$

with the Kronecker convention for δ_{ij} gives the simplified expression linking the stress tensor to the deformation tensor for isotropic media as

$$\sigma_{ij} = \lambda \epsilon_{kk} \delta_{ij} + 2\mu \epsilon_{ij}. \tag{5}$$

One may prefer the inverse of the relation (5) where the deformation tensor is expressed from the stress tensor by the introduction of the Young modulus E. Still, we have two independent coefficients. By injecting the relation (5) into the fundamental relation of dynamics (1), we end up with the so-called elastic wave propagation system, which is an hyperbolic system of second order, where only the displacement u has to be found. This system can be written as

$$\begin{aligned}
\frac{\partial^2 u_x}{\partial t^2} &= \frac{1}{\rho}\left[(\lambda+2\mu)\frac{\partial^2 u_x}{\partial x^2} + (\lambda+\mu)\left(\frac{\partial^2 u_y}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial x \partial z}\right) + \right. \\
&\quad \left. +\mu\left(\frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2}\right)\right] \\
\frac{\partial^2 u_y}{\partial t^2} &= \frac{1}{\rho}\left[(\lambda+2\mu)\frac{\partial^2 u_y}{\partial y^2} + (\lambda+\mu)\left(\frac{\partial^2 u_x}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial y \partial z}\right) + \right. \\
&\quad \left. +\mu\left(\frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial z^2}\right)\right] \\
\frac{\partial^2 u_z}{\partial t^2} &= \frac{1}{\rho}\left[(\lambda+2\mu)\frac{\partial^2 u_z}{\partial z^2} + (\lambda+\mu)\left(\frac{\partial^2 u_x}{\partial x \partial z} + \frac{\partial^2 u_y}{\partial y \partial z}\right) + \right. \\
&\quad \left. +\mu\left(\frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial y^2}\right)\right],
\end{aligned} \tag{6}$$

where we have neglected spatial variations of Lamé coefficients. Therefore, we must reconstruct over time the three components of the displacement or equivalently of the velocity or the acceleration. Choosing the stress is a matter of mechanical behaviour in a similar way for seismic instruments which record one of these fields.

For heterogeneous media, spatial differential rules for Lamé coefficients have to be designed. We shall see how to avoid this definition in the continuum by first considering hyperbolic system of first-order equations, keeping stress field. More generally, any hyperbolic equation with n-order derivatives could be transformed in a hyperbolic system with only first derivatives by adding additional unknown fields. This mathematical transformation comes naturally for the elastodynamic case by selecting the velocity field v and the stress field σ as fields we want to reconstruct. In a compact form, this first-order system in particle velocity and stresses is the following

$$\rho\frac{\partial v_i}{\partial t} = \sigma_{ij,j} + \rho f_i \tag{7a}$$

$$\frac{\partial \sigma_{ij}}{\partial t} = \lambda v_{k,k}\delta_{ij} + \mu(v_{i,j} + v_{j,i}), \tag{7b}$$

with $i, j = x, y, z$. We may consider other dual quantities as (displacement, integrated stress) or (acceleration, stress rate) as long as the medium is at rest before the dynamic evolution. Let us underline that time partial derivatives are on the left-hand side and that spatial variations and derivations are on the right-and side.

Using simple linear algebra manipulations, alternative equivalent expressions may deserve investigation: the three components σ_{ii} could be linearly combined for three alternative components considering the trace $\sigma_1 = trace(\sigma)/3$, the x-coordinate deviatoric stress $\sigma_2 = (2\sigma_{xx} - \sigma_{yy} - \sigma_{zz})/3$ and the y-coordinate deviatoric stress $\sigma_3 = (-\sigma_{xx} + 2\sigma_{yy} - \sigma_{zz})/3$ which allows to separate partial spatial derivatives in the right hand side and material properties in the left hand side. The system (7) becomes

$$\begin{aligned}
\rho \frac{\partial v_i}{\partial t} &= \sigma_{ij,j} + \rho f_i \\
\frac{3}{3\lambda + 2\mu} \frac{\partial \sigma_1}{\partial t} &= v_{i,i} \\
\frac{3}{2\mu} \frac{\partial \sigma_2}{\partial t} &= \left(3 \frac{\partial v_x}{\partial x} - v_{i,i} \right) \\
\frac{3}{2\mu} \frac{\partial \sigma_3}{\partial t} &= \left(3 \frac{\partial v_y}{\partial y} - v_{i,i} \right) \\
\frac{1}{\mu} \frac{\partial \sigma_{ij}}{\partial t} &= v_{i,j} + v_{j,i}
\end{aligned} \tag{8}$$

which could be useful when we move from differential formulation to integral formulation over elementary volumes. Partial differential operators only in the right-hand side of the system (8) are separated from spatial variations of model parameters on the left-hand side as a diagonal matrix $\Lambda = (\frac{3}{3\lambda+2\mu}, \frac{3}{2\mu}, \frac{3}{2\mu}, \frac{1}{\mu}, \frac{1}{\mu}, \frac{1}{\mu})$. Similar strategies could be applied for 2D geometries.

Finally, for easing discussions on the numerical implementation, let us write both the 1D scalar second-order acoustic wave equation in the time domain as

$$\rho(x) \frac{\partial^2 u(x,t)}{\partial t^2} = \frac{\partial}{\partial x} E(x) \frac{\partial u(x,t)}{\partial x}, \tag{9}$$

or, in frequency domain,

$$\omega^2 \rho(x) u(x,\omega) + \frac{\partial}{\partial x} E(x) \frac{\partial u(x,\omega)}{\partial x} = 0, \tag{10}$$

away from sources where one can see the importance of the mixed operator $\partial_x E(x) \partial_x$. We have introduced the Young modulus E related to unidirectional compression/delation motion. The 1D vectorial first-order acoustic wave equation can be written as

$$\begin{aligned}
\rho(x) \frac{\partial v(x,t)}{\partial t} &= \frac{\partial \sigma(x,t)}{\partial x} \\
\frac{\partial \sigma(x,t)}{\partial t} &= E(x) \frac{\partial v(x,t)}{\partial x},
\end{aligned} \tag{11}$$

from which one can deduce immediatly the system of equations in the frequency domain

$$\begin{aligned} -i\omega\rho(x)v(x,\omega) &= \frac{\partial\sigma(x,\omega)}{\partial x} \\ -i\omega\sigma(x,\omega) &= E(x)\frac{\partial v(x,\omega)}{\partial x}. \end{aligned} \tag{12}$$

Please note that the mixed operator does not appear explicitly. By discretizing this system and by eliminating the stress discrete values, one can go back to an equation involving only the velocity: a natural and systematic procedure for discretizing the mixed operator as proposed by Luo & Schuster (1990).

For an isotropic medium, two types of waves - compressional and shear waves - are propagating at two different velocities v_p and v_s. These velocities can be expressed as

$$v_p = \sqrt{\frac{\lambda + 2\mu}{\rho}} \quad \text{and} \quad v_s = \sqrt{\frac{\mu}{\rho}}, \tag{13}$$

except for the 1D medium where only compression/dilatation motion could take place. The displacement induced by these two different waves is such that compressive waves u^P verify $\nabla \times u_i^P = 0$ and shear waves u^S verify $\nabla \cdot u_i^S = 0$. Applying these operators to the numerical displacement will separate it into these two wavefields.

2.1 Time-domain or frequency-domain approaches

These systems of equations could be solved numerically in the time domain or in the frequency domain depending on applications. For seismic imaging, the forward problem has to be solved for each source and at each iteration of the optimisation problem. The time approach has a computational complexity increasing linearly with the number of sources while precomputation could be achieved in the frequency domain before modelling the propagation of each source. Let us write a compact form in order to emphasize the time/frequency domains approaches. The elastodynamic equations are expressed as the following system of second-order hyperbolic equations,

$$\mathbf{M}(\mathbf{x})\frac{\partial^2 \mathbf{w}(\mathbf{x},t)}{\partial t^2} = \mathbf{S}(\mathbf{x})\mathbf{w}(\mathbf{x},t) + \mathbf{s}(\mathbf{x},t), \tag{14}$$

where $\mathbf{M}$ and $\mathbf{S}$ are the mass and the stiffness matrices (Marfurt, 1984). The source term is denoted by $\mathbf{s}$ and the seismic wavefield by $\mathbf{w}$. In the acoustic approximation, $\mathbf{w}$ generally represents pressure, while in the elastic case, $\mathbf{w}$ generally represents horizontal and vertical particle displacements. The time is denoted by t and the spatial coordinates by $\mathbf{x}$. Equation (14) is generally solved with an explicit time-marching algorithm: the value of the wavefield at a time step $(n+1)$ at a spatial position x is inferred from the value of the wavefields at previous time steps (Dablain, 1986; Tal-Ezer et al., 1990). Implicit time-marching algorithms are avoided as they require solving a linear system (Marfurt, 1984; Mufti, 1985). If both velocity and stress wavefields are helpful, the system of second-order equations can be recast as a first-order hyperbolic velocity-stress system by incorporating the necessary auxiliary variables (Virieux, 1986). The time-marching approach could gain in efficiency if one consider

local time steps related to the coarsity of the spatial grid (Titarev & Toro, 2002): this leads to a quite challenging load balancing program between processors when doing parallel programming as most processors are waiting for the one which is doing the maximum of number crunching as illustrated for the ADER scheme (Dumbser & Käser, 2006). Adapting the distribution of the number of nodes to each processor depending on the expected complexity of mathematical operations is still an open problem. Other integration schemes as the Runge-Kutta scheme or the Stormer/Verlet symplectic scheme (Hairer et al., 2002) could be used as well.

Seismic imaging requires the cross-correlation in time domain or the product in frequency domain of the incidents field of one source and the backpropagated residues from the receivers for this source. In order to do so, one has to save at each point of the medium the incident field from the source which could be a time series or one complex number. The storage when considering a time-domain approach could be an issue: a possible strategy is storing only few time snapshots for recomputing the incident field on the fly (Symes, 2007) at intermediate times. An additional advantage is that the attenuation effect could be introduced as well. in the time-domain approach, the complexity increases linearly with the number of sources.

In the frequency domain, the wave equation reduces to a system of linear equations, the right-hand side of which is the source, and the solution of which is the seismic wavefield. This system can be written compactly as

$$\mathbf{B}(\mathbf{x},\omega)\mathbf{w}(\mathbf{x},\omega) = \mathbf{s}(\mathbf{x},\omega), \tag{15}$$

where $\mathbf{B}$ is the so-called impedance matrix (Marfurt, 1984). The sparse complex-valued matrix $\mathbf{B}$ has a symmetric pattern, although is not symmetric because of absorbing boundary conditions (Hustedt et al., 2004; Operto et al., 2007). The fourier transform is defined with the following convention

$$f(\omega) = \int_{-\infty}^{+\infty} f(t) e^{i\omega t} dt.$$

Solving the system of equations (15) can be performed through a decomposition of the matrix $\mathbf{B}$, such as lower and upper (LU) triangular decomposition, which leads to the so-called direct-solver techniques. The advantage of the direct-solver approach is that, once the decomposition is performed, equation (15) is efficiently solved for multiple sources using forward and backward substitutions (Marfurt, 1984). This approach has been shown to be efficient for 2D forward problems (Hustedt et al., 2004; Jo et al., 1996; Stekl & Pratt, 1998). However, the time and memory complexities of the LU factorization, and its limited scalability on large-scale distributed memory platforms, prevents the use of the direct-solver approach for large-scale 3D problems (*i.e.*, problems involving more than ten millions of unknowns) (Operto et al., 2007).

Iterative solvers provide an alternative approach for solving the time-harmonic wave equation (Erlangga & Herrmann, 2008; Plessix, 2007; Riyanti et al., 2006; 2007). Iterative solvers are currently implemented with Krylov-subspace methods (Saad, 2003) that are preconditioned by the solution of the dampened time-harmonic wave equation. The solution of the dampened wave equation is computed with one cycle of a multigrid. The main advantage of the iterative approach is the low memory requirement, while the main drawback results from the difficulty to design an efficient preconditioner, because the impedance matrix is indefinite. To our

knowledge, the extension to elastic wave equations still needs to be investigated. As for the time-domain approach, the time complexity of the iterative approach increases linearly with the number of sources or, equivalently, of right-hand sides, in contrast to the direct-solver approach.

An intermediate approach between the direct and the iterative methods consists of a hybrid direct-iterative approach that is based on a domain decomposition method and the Schur complement system (Saad, 2003; Sourbier et al., 2011): the iterative solver is used to solve the reduced Schur complement system, the solution of which is the wavefield at interface nodes between subdomains. The direct solver is used to factorize local impedance matrices that are assembled on each subdomain. Briefly, the hybrid approach provides a compromise in terms of memory saving and multi-source-simulation efficiency between the direct and the iterative approaches.

The last possible approach to compute monochromatic wavefields is to perform the modeling in the time domain and extract the frequency-domain solution, either by discrete Fourier transform in the loop over the time steps (Sirgue et al., 2008) or by phase-sensitivity detection once the steady-state regime has been reached (Nihei & Li, 2007). An arbitrary number of frequencies can be extracted within the loop over time steps at a minimal extra cost. Time windowing can easily be applied, which is not the case when the modeling is performed in the frequency domain. Time windowing allows the extraction of specific arrivals (early arrivals, reflections, PS converted waves) for the full waveform inversion (FWI), which is often useful to mitigate the nonlinearity of the inversion by judicious data preconditioning (Brossier et al., 2009; Sears et al., 2008).

Among all of these possible approaches, the iterative-solver approach has theoretically the best time complexity (here, *complexity* denotes how the computational cost of an algorithm grows with the size of the computational domain) if the number of iterations is independent of the frequency (Erlangga & Herrmann, 2008). In practice, the number of iterations generally increases linearly with frequency. In this case, the time complexity of the time-domain approach and the iterative-solver approach are equivalent (Plessix, 2007).

For one-frequency modeling, the reader is referred to those articles (Plessix, 2007; 2009; Virieux et al., 2009) for more detailed complexity analysis of seismic modeling based on different numerical approaches. A discussion on the pros and cons of time-domain versus frequency-domain seismic modeling relating to what it is required for full waveform inversion is also provided in Vigh & Starr (2008) and Warner et al. (2008).

2.2 Boundary conditions

In seismic exploration, two boundary conditions are implemented for wave modeling: absorbing boundary conditions to mimic an infinite medium and free surface conditions on the top side of the computational domain to represent the air-solid or air-water interfaces which have the highest impedance contrast. For internal boundaries, we assume that effects are well described by variations of the physical properties of the medium: the so-called implicit formulation (Kelly et al., 1976; Kummer & Behle, 1982).

2.2.1 PML absorbing boundary conditions

For simulations in an infinite medium, an absorbing boundary condition needs to be applied at the edges of the numerical model. An efficient way to mimic such an infinite medium can be achieved with Perfectly-Matched Layers (PML), which has been initially developed by Berenger (1994) for electromagnetics, and adapted for elastodynamics by Chew & Liu (1996); Festa & Vilotte (2005). PMLs are anisotropic absorbing layers that are added at the periphery of the numerical model. The classical PML formulation is based on splitting of the elastodynamic equations. A new kind of PML, known as CPML, does not require split terms. The CPML originated from Roden & Gedney (2000) for electromagnetics was applied by Komatitsch & Martin (2007) and Drossaert & Giannopoulos (2007) to the elastodynamic system. CPML is based on an idea of Kuzuoglu & Mittra (1996), who has obtained a strictly causal form of PML by adding some parameters in the standard damping function of Berenger (1994), which enhanced the absorption of waves arriving at the boundaries of the model with grazing incidence angles.

In the frequency domain, the implementation of PMLs consists of expressing the wave equation in a new system of complex-valued coordinates $\tilde{x}$ defined by (e.g., Chew & Weedon, 1994):

$$\frac{\partial}{\partial \tilde{x}} = \frac{1.}{\xi_x(x)} \frac{\partial}{\partial x}. \tag{16}$$

In the PML layers, the damped 1D acoustic wave equation could be deduced from the equation (10) as

$$\left[\omega^2 \rho(x) + \frac{1.}{\xi_x(x)} \frac{\partial}{\partial x} \frac{E(x)}{\xi_x(x)} \frac{\partial}{\partial x} \right] u(x,\omega) = -s(x,\omega), \tag{17}$$

where $\xi_x(x) = 1 + i\gamma_x(x)/\omega$ and $\gamma_x(x)$ is a 1D damping function which defines the PML damping behavior in the PML layers. In the CPML layers, the damping function $\xi_x(x)$ becomes

$$\xi_x(x) = \kappa_x + \frac{d_x}{\alpha_x + i\omega}, \tag{18}$$

with angular frequency ω and coefficients $\kappa_x \geq 1$ and $\alpha_x \geq 0$. The damping profile d_x varies from 0 at the entrance of the layer, up to a maximum real value $d_{\theta max}$ at the end (Collino & Tsogka, 2001) such that

$$d_x = d_{x max} \Big(\frac{\delta_x}{L_{cpml}} \Big)^2, \tag{19}$$

and

$$d_{x max} = -3V_P \frac{\log(R_{coeff})}{2L_{cpml}}, \tag{20}$$

with δ_x as the depth of the element barycentre inside the CPML, L_{cpml} the thickness of the absorbing layer, and R_{coeff} the theoretical reflection coefficient. Suitable expressions for κ_x, d_x and α_x are discussed in Collino & Monk (1998); Collino & Tsogka (2001); Drossaert & Giannopoulos (2007); Komatitsch & Martin (2007); Kuzuoglu & Mittra (1996); Roden & Gedney (2000). We often choose $R_{coeff} = 0.1\%$ and the variation of the coefficient α_x goes

from a maximum value ($\alpha_{x_{max}} = \pi f_0$) at the entrance of the CPML, to zero at its end. If $\kappa_x = 1$ and $\alpha_x = 0$, the classical PML formulation is obtained.

One can use directly these frequency-dependent expressions when considering the frequency approach. The formulation in the time domain is slightly more involved. The spatial derivatives are replaced by

$$\partial_{\tilde{x}} \rightarrow \frac{1}{\kappa_x}\partial_x + \zeta_x * \partial_x, \tag{21}$$

with

$$\zeta_x(t) = -\frac{d_x}{\kappa_x^2} H(t) e^{-(d_x\kappa_x + \alpha_x)t}, \tag{22}$$

where *H(t)* denotes the Heaviside distribution. Roden & Gedney (2000) have demonstrated that the time convolution in equation (21) can be performed in a recursive way using memory variables defined by

$$\psi_x = \zeta_x * \partial_x. \tag{23}$$

The function ψ_x represents a memory variable in the sense that it is updated at each time step. Komatitsch & Martin (2007) have shown that the term κ_x has a negligible effect on the absorbing abilities, and it can be set to 1. If we take $\kappa_x = 1$, we derive the equation (23) using the equation (22) as

$$\partial_t \psi_x = -d_x \partial_x - (d_x + \alpha_x)\psi_x. \tag{24}$$

One equation is generated for each spatial derivative involved in the elastodynamic system, which can be a memory-demanding task. Once they are computed at each time step, we can introduce the memory variables into the initial elastodynamic system which requires two additional variables for the 1D equations (11) with the definition of $\psi_x(v)$ and $\psi_x(\sigma)$ leading to the following system

$$\begin{aligned}
\rho(x)\frac{\partial v}{\partial t} &= \frac{\partial \sigma}{\partial x} + \psi_x(\sigma) \\
\frac{\partial \sigma}{\partial t} &= E(x)\frac{\partial v}{\partial x} + \psi_x(v) \\
\frac{\partial \psi_x(v)}{\partial t} &= -d_x(x)\frac{\partial \psi(v)}{\partial x} - (d_x(x) + \alpha_x(x))\psi(v) \\
\frac{\partial \psi_x(\sigma)}{\partial t} &= -d_x(x)\frac{\partial \psi(\sigma)}{\partial x} - (d_x(x) + \alpha_x(x))\psi(\sigma)
\end{aligned} \tag{25}$$

At the outer edge of the PML zone, one could apply any conditions as simple absorbing conditions (Clayton & Engquist, 1977) or free surface conditions (Etienne et al., 2010) as fields go to zero nearby the outer edge.

We must underline that the extension to 2D and 3D geometries is straightforward both in the frequency domain (Brossier et al., 2010; 2008) and in the time domain (Etienne et al., 2010; Komatitsch & Martin, 2007).

2.2.2 Free surface

Planar free surface boundary conditions can be simply implemented using a strong formulation or a weak formulation.

In the first case which is often met in the finite-difference methods, one requires that the stress is zero at the free surface. The free surface matches the top side of the FD grid and the stress is forced to zero on the free surface (Gottschamer & Olsen, 2001). Alternatively, the method of image can be used to implement the free surface along a virtual plane located half a grid interval above the topside of the FD grid (Virieux, 1986). The stress is forced to vanish at the free surface by using a virtual plane located half a grid interval above the free surface where the stress is forced to have opposite values to that located just below the free surface. In case of more complex topographies, one strategy is to adapt the topography to the grid structure at the expense of numerical dispersion effect (Robertsson, 1996) or to deform the underlying meshing used in the numerical method to the topography (Hestholm, 1999; Hestholm & Ruud, 1998; Tessmer et al., 1992). In the first case, because of stair-case approximation, a local fine sampling is required (Hayashi et al., 2001).

Owing to the weak formulation used in finite-element methods, the free surface boundary condition are naturally implemented by zeroing test functions on these boundaries which follow edges of grid elements (Zienkiewicz & Taylor, 1967). This approach could be used as well for finite-difference methods through the summation-by-parts (SBP) operators based on energy minimization combined with Simultaneous Approximation Term (SAT) formulation based on a boundary value penalty method (Taflove & Hagness, 2000). In this case, boundaries on which stress should be zero are not requested to follow any grid discretisation.

Finally, one may consider an immersed boundary approach where the free surface boundary is not related to the discretisation of the medium as promoted by LeVeque (1986). Extensive applications have been proposed by Lombard & Piraux (2004); Lombard et al. (2008) where grid discretisation does not influence the application of boundary conditions. This approach might be seen as an extension of the method of images following extrapolation techniques above the free surface at any a priori order of precision.

2.3 Source implementation

There are different ways of exciting the numerical grid by the source term. The simplest one is the direct contribution of the source term in the discrete partial differential equations: for example, we may just increment by the source term after each time step or we may consider the right-hand side source term for solving the linear system in the frequency domain. Depending on the numerical approach, it is necessary to consider specific influences coming from the discretisation as we shall see for numerical methods we consider.

In order to avoid singularities of solutions nearby the source, on can use the injection technique as proposed in the pioneering work of Alterman & Karal (1968) where a specific box around the source is defined. Inside the box, only the scattering field is computed. The incident field is estimated at the edges of the box and it is substracted when propagations are estimated inside the box and added when propagations are estimated outside the box. A more general framework is proposed by Opršal et al. (2009) related to boundary integral approaches.

3. Finite-difference methods: solving the equations through a strong formulation

We shall first consider the discretization based on simple and intuitive approaches as finite-difference methods for solving these partial differential equations while focusing on techniques useful for seismic imaging which means a significant number of forward problems for many sources in the same medium. We shall identify features which might be interesting for seismic imaging. If these approaches are intuitive for solving differential equations, the numerical implementation of boundary conditions and source excitation is less obvious and requires specific strategies as we shall see.

3.1 Spatial-domain finite-difference approximations

Whatever is our strategy for the reconstruction of the wave field u, one has to discretize it. We may be very satisfied by considering a set of discrete values $(u_1, u_2, ..., u_{I-1}, u_I)$ along one direction at a given specific time t_n which can be discretized as well. Therefore, a simple way of solving this first-order differential system is by making finite difference approximations of spatial derivatives.

Still considering a 1D geometry, the partial operator $(\partial/\partial x)$ could be deduced from a Taylor expansion using Lagrange polynomial. A quite fashionable symmetrical estimation using a centered finite difference approximation is expressed as

$$\frac{\partial u_i^n}{\partial x} = \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} + O\left[\Delta x^2\right], \tag{26}$$

which is a three-nodes stencil as three nodes are involved: two for the derivative estimation and one for the updating. Let us mention that the discrete derivative is shifted with respect to discrete values of the field. Because of the very specific antisymmetrical structure of our first-order hyperbolic system where time evolution of velocity requires only stress derivatives (and vice versa), we may consider centered approximations both in space and in time. This will lead to a leap-frog structure or a red/black pattern. Of course, we have truncation errors expressed by the function $O(\Delta x^n)$ which depends on the power n of the spatial stepping and by the function $O(\Delta t^k)$ on the power k of the time stepping.

We may require a greater precision of the derivative operator by using more points for this partial derivative approximation and a very popular centered finite difference approximation of the first derivative is the following expression

$$\frac{\partial u_i^n}{\partial x} = \frac{c_1\left(u_{i+\frac{1}{2}}^n - u_{i-\frac{1}{2}}^n\right) + c_2\left(u_{i+\frac{3}{2}}^n - u_{i-\frac{3}{2}}^n\right)}{\Delta x} + O\left[\Delta x^4\right], \tag{27}$$

where $c_1 = 9/8$ et $c_2 = -1/24$ (Levander, 1988). This fourth-order stencil is compact enough (few discrete points inside the stencil) for numerical efficiency while having a small local truncation error. This stencil is a five-nodes stencil. Let us underline that centered approximations lead to have field quantities not at the same position in the numerical grid as derivative approximations (figure 1). In other words, stress and velocity components should be specified on different positions of the spatial grid. If we still consider a full grid where stress and velocities are known at the same position, this stencil could be recast as a seven-nodes

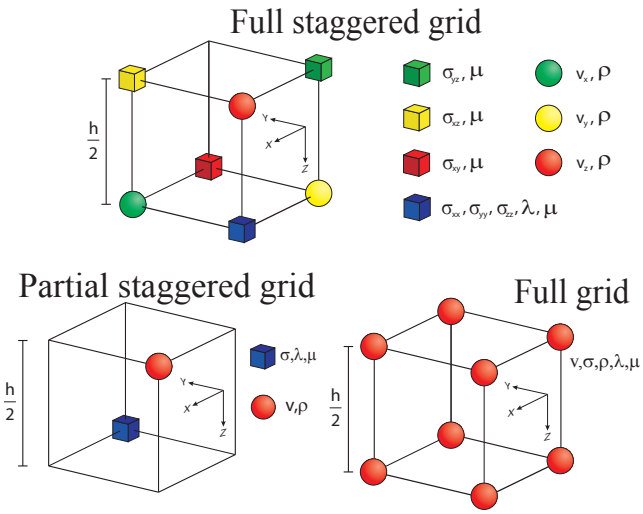


Fig. 1. Cell structure for the full staggered grid (top), the partial staggered grid (bottom left) and the full grid (bottom right, Stress tensor is denoted by σ, particle velocity by v and the density by ρ as well as Lamé coefficient by λ et μ. The regular grid step is denoted by h. Time stepping is missing

stencil given by

$$\frac{\partial u_i^n}{\partial x} = \frac{d_1\left(u_{i+1}^n - u_{i-1}^n\right) + d_2\left(u_{i+2}^n - u_{i-2}^n\right) + d_3\left(u_{i+3}^n - u_{i-3}^n\right)}{\Delta x'}, \tag{28}$$

where $\Delta x' = \Delta x/2$ and where we have following specific coefficients $d_1 = c_1$, $d_2 = 0$. and $d_3 = c_2$. The fourth-order scheme would require the following theoretical coefficients $d_1 = 15/20$, $d_2 = -3/20$ and $d_3 = 1/60$. For fourth-order stencils, the two sub-grids are not entirely decoupled and are weakly coupled leading to a dispersion behaviour as if the discretization is Δx. Let us remind that these sub-grids are completely decoupled when considering second-order stencils, leading to the staggered structure. Therefore, solving partial differential equations in the staggered grid structure has a less accurate resolution but improves significantly the efficiency of the method than solving equations in the full grid even with dispersion-relation-preserving stencils (Tam & Webb, 1993). The memory saving can be easily seen when comparing nodes for staggered grid and nodes for full grid (figure 1)

When dealing with 2D and 3D geometries, we may exploit the extra freedom and estimate derivatives along the direction x from nodes shifted by half the grid step in x but also by half the grid step in y (and eventually in z). This leads to another compact stencil as shown in the figure 1 where all components of the velocity are discretized in one location while all components of the stress field are discretized half the diagonal of the grid as proposed by Saenger et al. (2000). This grid is still partially staggered and could be named as a partial grid.

These standard and partial staggered structures are sub-grids of the full grid as shown in the figure 1 which is used in aeroacoustics (Tam & Webb, 1993).

These different discretizations related to various stencils may lead to preferential directions of propagation. Numerical anisotropy effect is observed even when considering isotropic wave propagation. The figure 2 shows error variations in velocities with respect to angles of propagation for the standard grid and the partial grid: one can see that the anisotropy behavior is completely different with a rotation shift of 45^0. In 2D, the two stencils provide the same anisotropic error while the partial grid has a slightly improved numerical anisotropic performance (percentage differences go from 3 % down to 2 % in 3D geometries). Of course, the spatial sampling is such that the error should be negligible and few percentages is considered to be acceptable except nearby the source.

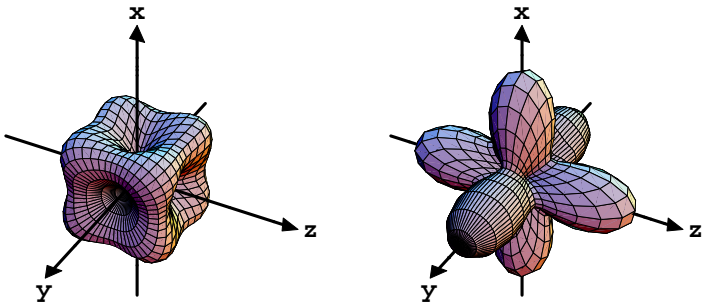


Fig. 2. Numerical anisotropic errors when considering finite difference stencils related to *partial staggered grid* (left panel) and *standard staggered grid* (right panel) Saenger et al. (2000).

Other spatial interpolations are possible. Previous discrete expressions are based on Lagrange interpolations while other interpolations are possible such as Chebychev or Laguerre polynomial or Fourier interpolations (Kosloff & Baysal, 1982; Kosloff et al., 1990; Mikhailenko et al., 2003). Interpolation basis could be local (Lagrange) or global(Fourier) ones based on equally spaced nodes or judiciously distributed nodes for keeping interpolation errors as small as possible: this will have a dramatic impact on the accuracy of the numerical estimation of the derivative and, therefore, on the resolution of partial differential equations. We should stress that local stencils should be preferred for seismic imaging for efficiency in the computation of the forward modeling.

3.2 Time-domain finite-difference approximations

Similarly, one may consider finite difference approximation for time derivatives which can be illustrated on the simple scalar wave equation. A widely used strategy is again the centered differences through the expression

$$\frac{\partial u_i^n}{\partial t} = \frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} + O\left[\Delta t^2\right]. \tag{29}$$

For understanding how the procedure of computing new values in time is performing, let us consider the simple 1D second-order scalar wave equation for displacement u. This equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \tag{30}$$

could be discretized through these finite difference approximations

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{(\Delta t)^2} \approx c^2 \left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \right]. \tag{31}$$

The next value at the discrete time $n+1$ comes from older values known at time n and time $n-1$ through the expression

$$u_i^{n+1} \approx (c\Delta t)^2 \left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \right] + 2u_i^n - u_i^{n-1}. \tag{32}$$

A more compact notation of this equation as

$$u_i^{n+1} = 2(1 - S^2)u_i^n + S^2(u_{i+1}^n + u_{i-1}^n) - u_i^{n-1} \tag{33}$$

shows the quantity

$$S = \frac{c\Delta t}{\Delta x},$$

known as the Courant number in the literature. This quantity is quite important for understanding the numerical dispersion and stability of finite difference schemes. The related stencil on the spatio-temporal grid as shown in the left panel of the figure 3 clearly illustrates that the value at time $n+1$ is explicitly computed from values at time $n-1$ and time n. In this explicit formulation, the selection of the time step Δt should verify that the Courant number is lower than 1 for any point of the medium.

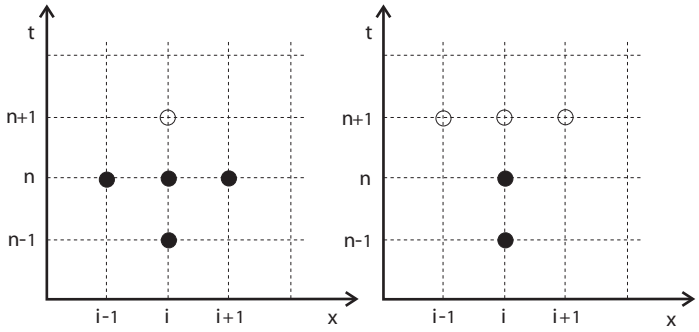


Fig. 3. Space/time finite difference stencil for an explicit scheme on the left and for an implicit scheme in a 1D configuration: black circles are known values from which the white circle is estimated.

On the contrary, we may consider spatial derivatives at time $n+1$. This leads us to an implicit scheme where more than one value at time $n+1$ is present in the discretisation. The equation is now

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{(\Delta t)^2} = c^2 \left[\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} \right] \tag{34}$$

which can be described by the Courant number S through the equation

$$(1+2S^2)u_i^{n+1} - S^2(u_{i+1}^{n+1} + u_{i-1}^{n+1}) = 2u_i^n - u_i^{n-1}. \tag{35}$$

The right panel of the figure 3 illustrates the structure of the stencil and that three unknowns have to be estimated through the single equation (35). By considering different spatial nodes, we may find these three unknowns by solving a linear system. The Courant number could take any value for time integration as long as discrete sampling is correctly performed.

Other implicit stencils might be designed by averaging the spatial derivatives over the three times $n-1$, n and $n+1$. We may as well average the time derivative over the three positions $i-1$, i and $i+1$. This lead to another linear system to be solved. These weighting strategies could be designed for reducing numerical noise as numerical dispersion and/or anisotropy: a road for further improvements.

As discretisation in space and time goes to zero, one expects the solution to be more precise but cumulative rounding errors should prevent to have too small values. In expressions (26) and (29), truncation error $O[\Delta x^2]$ goes to zero as the square of the discrete increment. We shall say that this is a second-order precision scheme both in space and in time. One consider often the stencil with the fourth-order precision in space and second-order precision in time, denoted as $O\left[\Delta x^4, \Delta t^2\right]$, as an optimal one for finite-differences simulations.

3.3 Frequency-domain finite-difference approximations

The second-order acoustic equation (10) provides a generalization of the Helmholtz equation. In exploration seismology, the source is generally a local point source corresponding to an explosion or a vertical force.

Attenuation effects of arbitrary complexity can be easily implemented in equations (10) and (12) using complex-valued wave speeds in the expression of the bulk modulus, thanks to the correspondence principle transforming time convolution into products in the frequency domain: in the frequency domain, one has to replace elastic coefficients by corresponding viscoelastic complex moduli for considering visco-elastic behaviors (Bland, 1960). For example, according to the Kolsky-Futterman model (Futterman, 1962; Kolsky, 1956), the complex wave speed $\bar{c}$ is given by

$$\bar{c} = c\left[\left(1 + \frac{1}{\pi Q}|log(\omega/\omega_r)|\right) + i\frac{sgn(\omega)}{2Q}\right]^{-1}, \tag{36}$$

where the P wave speed is denoted by $c = \sqrt{E/\rho}$, the attenuation factor by Q and a reference frequency by ω_r. The function sgn gives the sign of the function.

Since the relationship between the wavefields and the source terms is linear in the first-order and second-order wave equations, one can explicitely expressed the matrix structure of equations (10) and (12) through the compact expression,

$$[\mathbf{M} + \mathbf{S}]\,\mathbf{u} = \mathbf{B}\mathbf{u} = \mathbf{s}, \tag{37}$$

where **M** is the mass matrix, **S** is the complex stiffness/damping matrix. This expression holds as well in 2D and 3D geometries. The dimension of the square matrix **B** is the number of nodes in the computational domain multiplied by the number of wavefield components. System (37) could be solved using a sparse direct solver. A direct solver performs first a LU decomposition of **B** followed by forward and backward substitutions for the solutions (Duff et al., 1986) as shown by the following equations:

$$\mathbf{Bu} = (\mathbf{LU})\,\mathbf{u} = \mathbf{s} \tag{38}$$

$$\mathbf{Ly} = \mathbf{s}; \qquad \mathbf{Uu} = \mathbf{y} \tag{39}$$

Exploration seismology requires to perform seismic modeling for a large number of sources, typically, up to few thousands for 3D acquisition. The use of direct solver is the efficient computation of the solutions of the system (37) for multiple sources. Combining different stencils for constructing a compact and accurate stencil can follow strategies developped for acoustic and elastic wave propagation (Jo et al., 1996; Operto et al., 2007; Stekl & Pratt, 1998). The numerical anisotropy is dramatically reduced

The mass matrix **M** is a diagonal matrix although never explicitly constructed when considering explicit time integration. In the frequency domain formulation, we may spread out the distribution of mass matrix over neighboring nodes in order to increase the precision without increasing the computer cost as we have to solve a linear system in all cases. This strategy is opposite to the finite element approach where often the mass matrix is lumped into a diagonal matrix for explicit time integration (Marfurt, 1984). For a frequency formulation, considering the mass matrix as a non-diagonal matrix does not harm the solver. The weights of distribution are obtained through minimization of the phase velocity dispersion in an infinite homogeneous medium (Brossier et al., 2010; Jo et al., 1996): the numerical dispersion is dramatically reduced.

3.4 PML absorbing boundary condition implementation

Implementation of PML conditions in the frequency domain is straightforward using unsplit variables while, in the time domain, we need to introduce additional variables for handling the convolution through memory variables or to use split unphysical field variables (Cruz-Atienza, 2006). These additional variables are only necessary in the boundary layers following the figure 4

We first consider an infinite homogeneous medium which is embedded into a cubic box of a 16 km size and a grid stepping of $h = 100\ m$. The thickness of the PML layer is 1 km leading to $nsp = ten$ nodes inside the PML zone. The P-wave velocity is 4000 m/s while the S-wave velocity is 2300 m/s and the density 2500 kg/m^3. The figure 5 shows various time sections of the 3D volume for the vertical particle velocity where one can see that the explosive wavefront is entering the PML zone at the time 2.8 s. The last two snapshots shows the vanishing of the wavefront with completely negligible residues at the final time (the decrease of the elastic energy is better than 0.2 % for ten nodes and could reach 0.03 % for twenty nodes).

When we have discontinuous interfaces crossing the PML zone, we may expect difficulties coming from various angles of propagation waves (Chew & Liu, 1996; Festa & Nielsen, 2003; Marcinkovich & Olsen, 2003). Therefore, a simple heterogeneous medium is considered

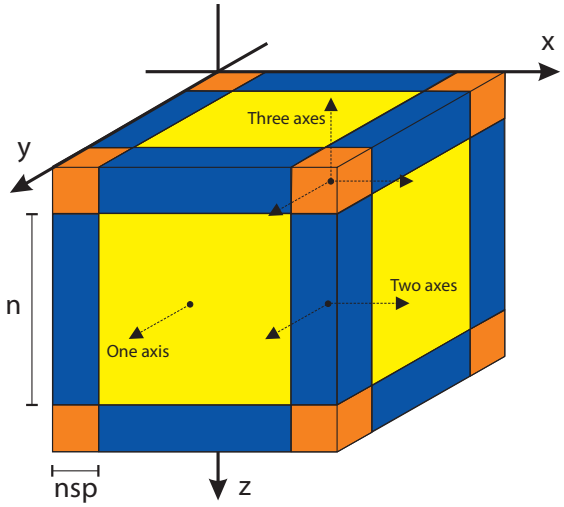


Fig. 4. Three kinds of PML boundary layers should be considered where only one coordinate is involved (yellow zone), two coordinates are involved (blue zone) and three coordinates are involved (red zone). Internally, standard elastodynamic equations are solved

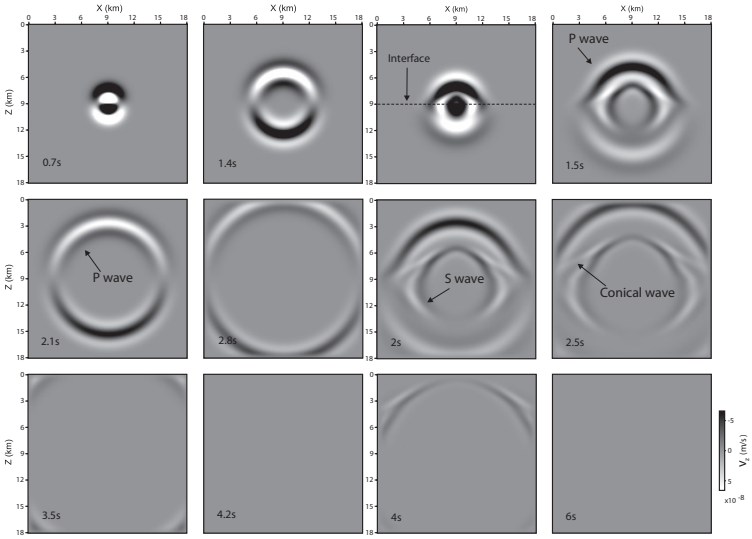


Fig. 5. Snapshots for y=0 of the vertical particle velocity at different times for an explosive source: on the left for an homogeneous infinite medium and on the right for an heterogeneous medium. Please note the vanishing of the seismic waves, thanks to the PML absorption

with two layers where physical parameters are (v_p, v_s, ρ)=(4330 m/s, 2500 m/s, 2156 kg/m^3) and (v_p, v_s, ρ)=(6000 m/s, 4330 m/s, 2690 kg/m^3). The figure 5 shows that, in spite of the complexities of waves generated at the horizontal flat interface, the PML layer succeeds to absorb seismic energy with a residual energy of 0.3 % in this case, still far better than standard paraxial absorbing boundary conditions (Clayton & Engquist, 1977).

3.5 Source and receiver implementation on coarse grids

Seismic imaging by full waveform inversion is initiated at an initial frequency as small as possible to mitigate the non linearity of the inverse problem resulting from the use of local optimization approach such as gradient methods. The starting frequency for modeling in exploration seismics can be as small as 2 Hz which can lead to grid intervals as large as 200 m. In this framework, accurate implementation of point source at arbitrary position in a coarse grid is critical. One method has been proposed by Hicks (2002) where the point source is approximated by a windowed Sinc function. The Sinc function is defined by

$$sinc(x) = \frac{sin(\pi x)}{\pi x}, \tag{40}$$

where $x = (x_g - x_s)$, x_g denotes the position of the grid nodes and x_s denotes the position of the source. The Sinc function is tapered with a Kaiser function to limit its spatial support (Hicks, 2002) . For multidimensional simulations, the interpolation function is built by tensor product construction of 1D windowed Sinc functions. If the source positions matches the position of one grid node, the Sinc function reduces to a Dirac function at the source position and no approximation is used for the source positioning. If the spatial support of the Sinc function intersects a free surface, part of the Sinc function located above the free surface is mirrored into the computational domain with a reverse sign following the method of image. Vertical force can be implemented in a straightforward way by replacing the Sinc function by its vertical derivative. The same interpolation function can be used for the extraction of the pressure wavefield at arbitrary receiver positions. The accuracy of the method of Hicks (2002) is illustrated in Figure 6a which shows a 3.75 Hz monochromatic wavefield computed in a homogeneous half space. The wave speed is 1500 m/s and the density is 1000 kg/m^3. The grid interval is 100 m. The free surface is half a grid interval above the top of the FD grid and the method of image is used to implement the free surface boundary condition. The source is in the middle of the FD cell at 2 km depth. The receiver line is oriented in the Y direction. Receivers are in the middle of the FD cell in the horizontal plane and at a depth of 6 m just below the free surface. Comparison between the numerical and the analytical solutions at the receiver positions are first shown when the source is positioned at the closest grid point and the numerical solutions are extracted at the closest grid point (Figure 6b). The amplitude of the numerical solution is strongly overestimated because the numerical solution is extracted at a depth of 50 m below free surface (where the pressure vanishes) instead of 6 m. Second, a significant phase shift between numerical and analytical solutions results from the approximate positioning of the sources and receivers. In contrast, a good agreement between the numerical and analytical solutions both in terms of amplitude and phase is shown in Figure 6c where the source and receiver positioning is implemented with the windowed Sinc interpolation.

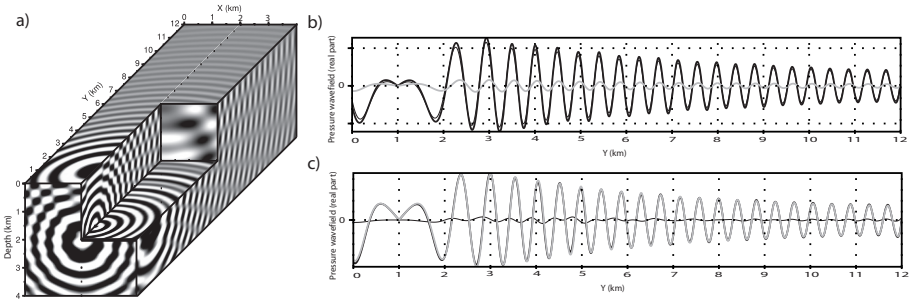


Fig. 6. a) Real part of a 3.75Hz monochromatic wavefield in a homogeneous half space. (b) Comparison between numerical (black) and analytical (gray) solutions at receiver positions when the closest grid point is used for both the source implementation and the extraction of the solution at the receiver positions on a coarse FD grid. (c) Same comparison between numerical (black) and analytical (gray) solutions at receiver positions when the Sinc interpolation with 4 coefficients is used for both the source implementation and the extraction of the solution at the receiver positions on a coarse FD grid.

4. Realistic examples for acoustic and elastic propagations using FD formulations

We shall provide two simple examples of seismic modeling using finite-differences methods both in the frequency and time approaches. The first example concerns seismic exploration problem where the acoustic approximation is often used while the second one is related to seismic risk mitigation where free surface effects including elastic propagation are quite important.

4.1 3D EAGE/SEG salt model

The salt model is a constant density acoustic model covering an area of 13.5 km $\times$ 13.5 km $\times$ 4.2 km (Aminzadeh et al., 1997)(Figure 7). The salt model is representative of a Gulf Coast salt structure which contains salt sill, different faults, sand bodies and lenses. The salt model is discretized with 20 m cubic cells, representing an uniform mesh of 676 x 676 x 210 nodes. The minimum and maximum velocities in the Salt model are 1500 m/s and 4482 m/s respectively. We performed a simulation for a frequency of 7.33 Hz and for one source located at $x = 3600\ m$, $y = 3.600\ m$ and $z = 100\ m$. The original model is resampled with a grid interval of 50 m corresponding to 4 grid points per minimum wavelength. The dimension of the resampled grid is 270 x 270 x 84 which represents 8.18 millions of unknowns after addition of the PML layers. Results of simulations performed with either in the frequency domain or in the time domain are compared in Figure 7. The time duration of the simulation is 15 s.

We obtain a good agreement between the two solutions (Figure 7d) although we show a small phase shift between the two solutions at offsets greater than 5000 m. This phase shift results from the propagation in the high-velocity salt body. The direct-solver modeling is performed on 48 MPI process using 2 threads and 15 Gbytes of memory per MPI process. The memory and the elapsed time for the LU decomposition were 402 Gbytes and 2863 s, respectively. The elapsed time for the solution step for one right-hand side (RHS) is 1.4 s when we process 16 RHS at a time during the solution step in MUMPS. The elapsed time for one time-domain simulation on 16 processors is 211 s. The frequency-domain approach is more than one order

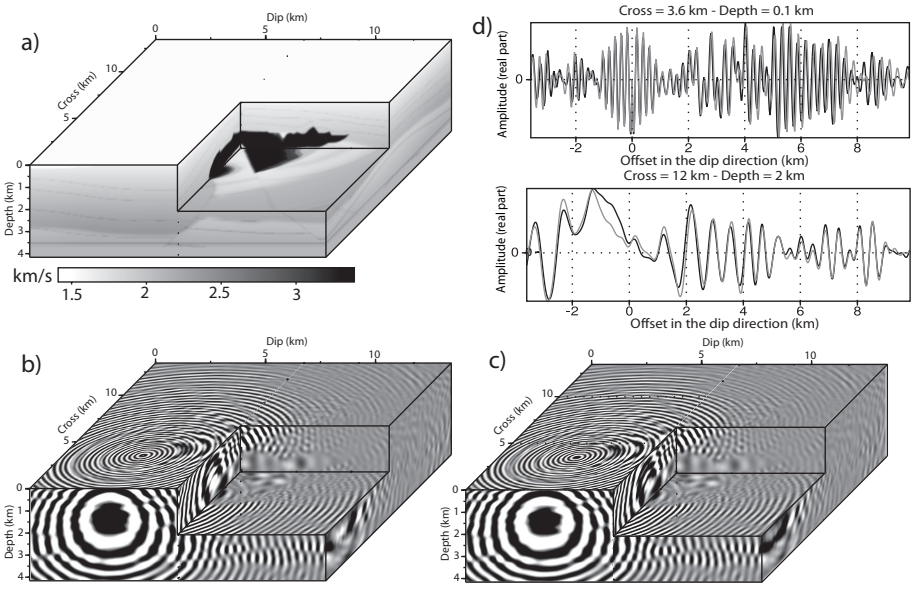


Fig. 7. (a) Salt velocity model. (b-c) 7.33-Hz monochromatic wavefield (real part) computed with a finite-different formulation in the frequency domain (b) and in the time domain (c). (d) Direct comparison between frequency-domain (gray) and time-domain (black) solutions. The receiver line in the dip direction is: (top) at 100 *m* depth and at 3600 *m* depth in the cross direction. The amplitudes are corrected for 3D geometrical spreading. (bottom) at 2500 *m* depth and at 15000 *m* in the cross direction.

of magnitude faster than the time-domain one when a large number of RHS members (2000) and a small number of processors (48) are used (Table 1). For a number of processors equal to the number of RHS members, the two approaches have the same cost. Of note, in the latter configuration (N_P=N_{rhs}), the cost of the two methods is almost equal in the case of the salt model (0.94 h versus 0.816 h).

Over the last decades, simulations of wave propagation in complex media have been efficiently tackled with finite-difference methods (FDMs) and applied with success to numerous physical problems (Graves, 1996; Moczo et al., 2007). Nevertheless, FDMs suffer from some critical issues that are inherent to the underlying Cartesian grid, such as parasite diffractions in cases where the boundaries have a complex topography. To reduce these artefacts, the discretisation should be fine enough to reduce the 'stair-case' effect at the free surface. For instance, a second-order rotated FDM requires up to 60 grid points per wavelength to compute an accurate seismic wavefield in elastic media with a complex topography (Bohlen & Saenger, 2006). Such constraints on the discretisation drastically restrict the possible field of realistic applications. Some interesting combinations of FDMs and finite-element methods (FEMs) might overcome these limitations (Galis et al., 2008). The idea is to use an unstructured FEM scheme to represent both the topography and the shallow part of the medium, and to adopt for the rest of the model a classical FDM regular grid. For the same reasons as the issues related to the topography, uniform grids are not suitable for highly

Model	Method	Pre. (hr)	Sol. (hr)	Total (hr)	Pre. (hr)	Sol. (hr)	Total (hr)
Salt	Time	0	39	39	0	0.94	0.94
Salt	Frequency	0.8	0.78	1.58	0.80	0.016	0.816

Table 1. Comparison between time-domain and frequency-domain modeling for 32 (left) and 2000 (right) processors. The number of sources is 2000. *Pre*. denotes the elapsed time for the source-independent task during seismic modeling (i.e., the LU factorization in the frequency-domain approach). *Sol*. denotes the elapsed time for multi-RHS solutions during seismic modeling (i.e., the substitutions in the frequency-domain approach).

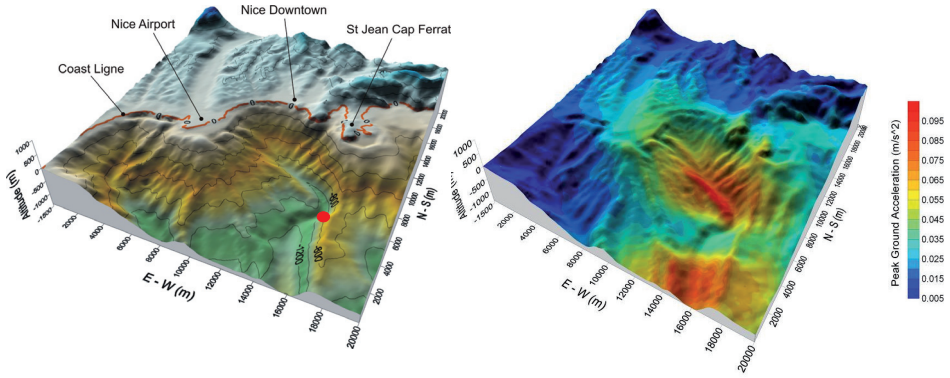


Fig. 8. On the left, the French Riviera medium with complex topography and bathymetry: an hypothetical earthquake of magnitude 4.5 is at a depth of 10 *km* below the epicenter shown by a red ball. The simulation medium is 20 *km* by 20 *km* by 15 *km*. On the right, the related peak ground acceleration (PGA). Please note that the acceleration is always lower than one tenth of the Earth acceleration g

heterogeneous media, since the grid size is determined by the shortest wavelength. Except in some circumstances, like mixing grids (Aoi & Fujiwara, 1999) or using non uniform Cartesian grids (Pitarka, 1999) in the case of a low velocity layer, it is almost impossible to locally adapt the grid size to the medium properties in the general case. From this point of view, FEMs are appealing, since they can use unstructured grids or meshes. Due to ever-increasing computational power, these kinds of methods have been the focus of a lot of interest and have been used intensively in seismology (Aagaard et al., 2001; Akcelik et al., 2003; Ichimura et al., 2007).

4.2 PGA estimation in the French Riviera

Peak ground acceleration (PGA) are estimated using empirical attenuation laws calibrated through databases of seismic records of various areas: these laws should be adapted to each area around the world and European moderate earthquakes require a specific calibration (Berge-Thierry et al., 2003). Aside these attenuation laws, numerical tools as finite-differences time-domain methods allows the deterministic estimation of the peak ground acceleration (PGA) in specific areas of interest once the medium is known and the source specified.

Small areas as the French Riviera where a complex topography and bathymetric makes the simulation difficult. We would like to illustrate the procedure of time-domain simulation on this specific example (Cruz-Atienza et al., 2007). The Figure 8 shows a very simple model surrounding the city of Nice: the box is 20 km by 20 km by 15 km in depth. The P-wave velocity is 5700 m/s while the S-wave velocity is 3300 m/s and the density 2600 km/m^3. The water is characterized by a P-wave velocity of 1530 m/s while the density is about 1030 km/m^3. The grid step is 50 m and the time integration step is 0.005 s.

The numerical simulation of an hypothetical earthquake of magnitude 4.5 at a depth of 10 km in the Mediterranean Sea provides us a deterministic estimation of the PGA as shown in the Figure 8. This small source is characterized upto a frequency of 3 Hz and we select a source time function with this expected spectral content.

Successful applications have been proposed in the Los Angeles basin and is improved as we increase our knowledge about the medium of propagation and about the source location and its characterization. The PGA is estimated everywhere and one can see that increase of the PGA is observed at the sea bottom and nearby the coast. One can show that the amplification of PGA is decreased when considering the water layer at the expense of a longer duration of the seismic signal.
Of course, various simulations should be performed using different models of the medium and for various source scenarii. These simulations could help to assess the variability of the acceleration for possible potential earthquakes and may be used for the mitigation of seismic risks. The importance of constraining the model structure should be emphasized and we can accumulate this knowledge through various and different initiatives performed for a more accurate reconstruction of the velocity structure (Rollet et al., 2002). One tool is the seismic imaging procedure we have underlined in this chapter.

5. Finite-elements discontinuous Galerkin methods: a weak formulation

Finite element methods, often more intensive in computer resources, introduce naturally boundary conditions in an explicit manner. Therefore, we expect improved accurate solutions with this numerical approach at the expense of computer requirements. The system of equations (14) in time has now a non-diagonal mass matrix while the system of equations (15) has a impedance matrix particularly ill-conditioned in 3D geometry taking into account its dimensionality. Therefore, for 2D geometries, the frequency formulation is still a quite feasible option while time domain approaches are there appealing when considering 3D geometries. Due to ever-increasing computational power, finite element methods using unstructured meshes have been the focus of increased interest and have been used extensively in seismology (Aagaard et al., 2001; Akcelik et al., 2003; Ichimura et al., 2007).

Usually, the approximation order remains low, due to the prohibitive computational cost related to a non-diagonal mass matrix. However, this high computational cost can be avoided by mass lumping, a standard technique that replaces the large linear system by a diagonal matrix (Chin-Joe-Kong et al., 1999; Marfurt, 1984) and leads to an explicit time integration. Another class of FEMs that relies on the Gauss-Lobatto-Legendre quadrature points has removed these limitations, and allows for spectral convergence with high approximation orders. This high-order FEM, called the spectral element method (SEM) (Komatitsch & Vilotte, 1998; Seriani & Priolo, 1994) has been applied to large-scale geological models up

to the global scale (Chaljub et al., 2007; Komatitsch et al., 2008). The major limitation of SEM is the exclusive use of hexahedral meshes, which makes the design of an optimal mesh cumbersome in contrast to the flexibility offered by tetrahedral meshes. With tetrahedral meshes (Frey & George, 2008), it is possible to fit almost perfectly complex topographies or geological discontinuities and the mesh width can be adapted locally to the medium properties (*h*-adaptivity). The extension of the SEM to tetrahedral elements represents ongoing work, while some studies have been done in two dimensions on triangular meshes (Mercerat et al., 2006; Pasquetti & Rapetti, 2006). On the other hand, another kind of FEM has been proven to give accurate results on tetrahedral meshes: the Discontinuous Galerkin finite-element method (DG-FEM) in combination with the arbitrary high-order derivatives (ADER) time integration (Dumbser & Käser, 2006). Originally, DG-FEM has been developed for the neutron transport equation (Reed & Hill, 1973). It has been applied to a wide range of applications such as electromagnetics (Cockburn et al., 2004), aeroacoustics (Toulopoulos & Ekaterinaris, 2006) and plasma physics (Jacobs & Hesthaven, 2006), just to cite a few examples. This method relies on the exchange of numerical fluxes between adjacent elements. Contrary to classical FEMs, no continuity of the basis functions is imposed between elements and, therefore, the method supports discontinuities in the seismic wavefield as in the case of a fluid/solid interface. In such cases, the DG-FEM allows the same equation to be used for both the elastic and the acoustic media, and it does not require any explicit conditions on the interface (Käser & Dumbser, 2008), which is, on the contrary, mandatory for continuous formulations, like the SEM (Chaljub et al., 2003). Moreover, the DG-FEM is completely local, which means that elements do not share their nodal values, contrary to conventional continuous FEM. Local operators make the method suitable for parallelisation and allow for the mixing of different approximation orders (*p*-adaptivity).

5.1 3D finite-element discontinuous Galerkin method in the time domain

Time domain approaches are quite attractive when considering explicit time integration. However, in most studies, the DG-FEM is generally used with high approximation orders. We present a low-order DG-FEM formulation with the convolutional perfectly matched layer (CPML) absorbing boundary condition (Komatitsch & Martin, 2007; Roden & Gedney, 2000) that is suitable for large-scale three-dimensional (3D) seismic wave simulations. In this context, the DG-FEM provides major benefits.

The *p*-adaptivity is crucial for efficient simulations, in order to mitigate the effects of the very small elements that are generally encountered in refined tetrahedral meshes. Indeed, the *p*-adaptivity allows an optimised time stepping to be achieved, by adapting the approximation order according to the size of the elements and the properties of the medium. The benefit of such a numerical scheme is particularly important with strongly heterogeneous media. Due to the mathematical formulation we consider, the medium properties are assumed to be constant per element. Therefore, meshes have to be designed in such a way that this assumption is compatible with the expected accuracy. The discretization must be able to represent the geological structures fairly well, without over-sampling, while the spatial resolution of the imaging process puts constraints on the coarsest parameterisation of the medium. If we consider full waveform inversion (FWI) applications, the expected imaging resolution reaches half a wavelength, as shown by Sirgue & Pratt (2004). Therefore, following the Shannon theorem, a minimum number of four points per wavelength is required to obtain

such accuracy. These reasons have motivated the development of DG-FEM with low orders. We focus on the quadratic interpolation, which yields a good compromise between accuracy, discretisation and computational cost.

5.1.1 3D time-domain elastodynamics

It is worth to provide notations for specific manipulation of equations for DG-FEM approaches. The first-order hyperbolic system (8) under the so-called pseudo-conservative form can be written following the approach of Ben Jemaa et al. (2007) as

$$\rho \partial_t \vec{v} = \sum_{\theta \in \{x,y,z\}} \partial_\theta (\mathcal{M}_\theta \vec{\sigma}) + \vec{f}$$

$$\Lambda \partial_t \vec{\sigma} = \sum_{\theta \in \{x,y,z\}} \partial_\theta (\mathcal{N}_\theta \vec{v}) + \Lambda \partial_t \vec{\sigma_0}, \tag{41}$$

with the definitions of the velocity and stress vectors as $\vec{v}^t = (v_x\ v_y\ v_z)^t$ and $\vec{\sigma} = (\sigma_1\ \sigma_2\ \sigma_3\ \sigma_{xy}\ \sigma_{xz}\ \sigma_{yz})^t$. Under this pseudo-conservative form, the RHS of (41) does not include any term that relates to the physical properties. The diagonal matrix Λ has been introduced in the system (8) and its inverse is required for the computation of the stress components (equation (41)). Matrices $\mathcal{M}_\theta$ and $\mathcal{N}_\theta$ are constant real matrices (Etienne et al., 2010). The extension of the pseudo-conservative form for the visco-elastic cases could be considered with the inclusion of memory variables while the anisotropic case should be further analysed since the change of variable may depend on the physical parameters. Finally, in the equation (41), the medium density is denoted by ρ, while $\vec{f}$ and $\vec{\sigma_0}$ are the external forces and the initial stresses, respectively.

5.1.2 Spatial discretisation

Following standard strategies of finite-element methods (Zienkiewicz et al., 2005), we want to approximate the solution of the equation (41) by means of polynomial basis functions defined in volume elements. The spatial discretisation is carried out with non-overlapping and conforming tetrahedra. We adopt the nodal form of the DG-FEM formulation (Hesthaven & Warburton, 2008), assuming that the stress and velocity vectors are approximated in the tetrahedral elements as follows

$$\widehat{\vec{v}}_i(\vec{x},t) = \sum_{j=1}^{d_i} \vec{v}_{ij}(\vec{x}_j,t)\ \varphi_{ij}(\vec{x})$$

$$\widehat{\vec{\sigma}}_i(\vec{x},t) = \sum_{j=1}^{d_i} \vec{\sigma}_{ij}(\vec{x}_j,t)\ \varphi_{ij}(\vec{x}), \tag{42}$$

where i is the index of the element, $\vec{x}$ is the spatial coordinates inside the element, and t is the time. d_i is the number of nodes or degrees of freedom (DOF) associated with the interpolating Lagrangian polynomial basis function φ_{ij} relative to the j-th node located at position $\vec{x}_j$. Vectors $\vec{v}_{ij}$ and $\vec{\sigma}_{ij}$ are the velocity and stress vectors, respectively, evaluated at

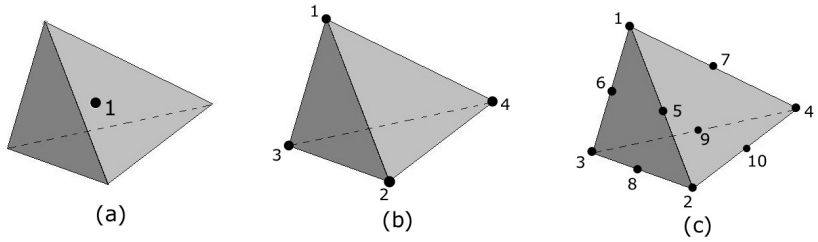


Fig. 9. (a) P_0 element with one unique DOF. (b) P_1 element with four DOF. (c) P_2 element with 10 DOF.

the j-th node of the element. Although it is not an intrinsic limitation, we have adopted here the same set of basis functions for the interpolation of the velocity and the stress components. In the following, the notation P_k refers to a spatial discretisation based on polynomial basis functions of degree k, and a P_k element is a tetrahedron in which a P_k scheme is applied. The number of DOF in a tetrahedral element is given by $d_i = (k+1)(k+2)(k+3)/6$. For instance, in a P_0 element (Figure 9.a), there is only one DOF (the stress and velocity are constant per element), while in a P_1 element (Figure 9.b), there are four DOF located at the four vertices of the tetrahedron (the stress and velocity are linearly interpolated). It is worth noting that the P_0 scheme corresponds to the case of the finite-volume method (Ben Jemaa et al., 2009; 2007; Brossier et al., 2008). For the quadratic approximation order P_2, one node is added at the middle of each edge of the tetrahedron, leading to a total of 10 DOF per element (Figure 9.c). The first step in the finite-element formulation is to obtain the weak form of the elastodynamic system. To do so, we multiply the equation (41) by a test function φ_{ir} and integrate the system over the volume of the element i. For the test function, we adopt the same kind of function as used for the approximation of the solution. This case corresponds to the standard Galerkin method and can be written as

$$\int_{V_i} \varphi_{ir}\, \rho \partial_t \vec{v}\, dV = \int_{V_i} \varphi_{ir} \sum_{\theta \in \{x,y,z\}} \partial_\theta (\mathcal{M}_\theta \vec{\sigma})\, dV$$

$$\int_{V_i} \varphi_{ir}\, \Lambda \partial_t \vec{\sigma}\, dV = \int_{V_i} \varphi_{ir} \sum_{\theta \in \{x,y,z\}} \partial_\theta (\mathcal{N}_\theta \vec{v})\, dV \qquad \forall r \in [1,\, d_i], \tag{43}$$

where the volume of the tetrahedral element i is denoted by V_i. For the purpose of clarity, we have omitted the external forces and stresses in the equation (43). Standard manipulations of finite-elements methods (integration by parts, Green theorem for fluxes along boundary surfaces) are performed as well as an evaluation of centered flux scheme for its non-dissipative property (Ben Jemaa et al., 2007; Delcourte et al., 2009; Remaki, 2000). Moreover, we assume constant physical properties per element. We define the tensorial product $\otimes$ as the Kronecker

product of two matrices **A** and **B** given by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1m}\mathbf{B} \\ . & . & . \\ . & . & . \\ . & . & . \\ a_{n1}\mathbf{B} & \dots & a_{nm}\mathbf{B} \end{bmatrix}, \tag{44}$$

where $(n \times m)$ denotes the dimensions of the matrix **A**. We obtain the expression

$$\begin{aligned} \rho_i(\mathcal{I}_3 \otimes \mathcal{K}_i)\partial_t \vec{v}_i &= -\sum_{\theta \in \{x,y,z\}} (\mathcal{M}_\theta \otimes \mathcal{E}_{i\theta})\vec{\sigma}_i + \frac{1}{2}\sum_{k \in N_i} \Big[(\mathcal{P}_{ik} \otimes \mathcal{F}_{ik})\vec{\sigma}_i + (\mathcal{P}_{ik} \otimes \mathcal{G}_{ik})\vec{\sigma}_k\Big] \\ (\Lambda_i \otimes \mathcal{K}_i)\partial_t \vec{\sigma}_i &= -\sum_{\theta \in \{x,y,z\}} (\mathcal{N}_\theta \otimes \mathcal{E}_{i\theta})\vec{v}_i + \frac{1}{2}\sum_{k \in N_i} \Big[(\mathcal{Q}_{ik} \otimes \mathcal{F}_{ik})\vec{v}_i + (\mathcal{Q}_{ik} \otimes \mathcal{G}_{ik})\vec{v}_k\Big], \end{aligned} \tag{45}$$

where $\mathcal{I}_3$ represents the identity matrix. In the system (45), the vectors $\vec{v}_i$ and $\vec{\sigma}_i$ should be red as the collection of all nodal values of the velocity and stress components in the element i. The system (45) indicates that the computations of the stress and velocity wavefields in one element require information from the directly neighbouring elements. This illustrates clearly the local nature of DG-FEM. The flux-related matrices $\mathcal{P}$ and $\mathcal{Q}$ are defined as follows

$$\begin{aligned} \mathcal{P}_{ik} &= \sum_{\theta \in \{x,y,z\}} n_{ik\theta}\, \mathcal{M}_\theta \\ \mathcal{Q}_{ik} &= \sum_{\theta \in \{x,y,z\}} n_{ik\theta}\, \mathcal{N}_\theta, \end{aligned}$$

where the component along the θ axis of the unit vector $\vec{n}_{ik}$ of the face S_{ik} that points from element i to element k is denoted by $n_{ik\theta}$, while we also introduce the mass matrix, the stiffness matrix and the flux matrices with $\theta \in \{x, y, z\}$ respectively,

$$\begin{aligned} (\mathcal{K}_i)_{rj} &= \int_{V_i} \varphi_{ir}\, \varphi_{ij}\, dV \qquad j, r \in [1,\, d_i], \\ (\mathcal{E}_{i\theta})_{rj} &= \int_{V_i} (\partial_\theta \varphi_{ir})\, \varphi_{ij}\, dV \qquad j, r \in [1,\, d_i], \\ (\mathcal{F}_{ik})_{rj} &= \int_{S_{ik}} \varphi_{ir}\, \varphi_{ij}\, dS \qquad j, r \in [1,\, d_i] \\ (\mathcal{G}_{ik})_{rj} &= \int_{S_{ik}} \varphi_{ir}\, \varphi_{kj}\, dS \qquad r \in [1,\, d_i] \qquad j \in [1,\, d_k]. \end{aligned} \tag{46}$$

It is worth noting that, in the last equation of the system (46), the DOF of elements i and k appear (d_i and d_k, respectively) indicating that the approximation orders are totally decoupled from one element to another. Therefore, the DG-FEM allows for varying approximation orders in the numerical scheme. This feature is referred to as p-adaptivity. Moreover, given an approximation order, these matrices are unique for all elements (with a normalisation according to the volume or surface of the elements) and they can be computed before hand with appropriate integration quadrature rules. The memory requirement is therefore low, since only a collection of small matrices is needed according to the possible combinations of

approximation orders. The maximum size of these matrices is $(d_{max} \times d_{max})$ where d_{max} is the maximum number of DOF per element and the number of matrices to store is given by the square of the number of approximation orders mixed in the numerical domain. The four matrices $\mathcal{K}_i$, $\mathcal{E}_i$, $\mathcal{F}_{ik}$ and $\mathcal{G}_{ik}$ are computed by numerical integration using Hammer quadrature (Hammer & Stroud, 1958) and explicit forms of these matrices could be found in Etienne et al. (2010) for P_0, P_1 and P_2 orders.

It should be mentioned that, in order to retrieve both the velocity and the stress components, the system (45) requires the computation of $\mathcal{K}_i^{-1}$, which can also be performed before hand. Note that, if we want to consider variations in the physical properties inside the elements, the pseudo-conservative form makes the computation of flux much easier and computationally more efficient than in the classical elastodynamic system. These properties come from the fact that, in the pseudo-conservative form, the physical properties are located in the left-hand side of the system (41). Therefore, no modification of the stiffness and flux matrices nor additional terms are needed in the system (45) to take into account the variation of properties. Only the mass matrix needs to be evaluated for each element and for each physical property according to the expression

$$(\mathcal{K}_i)_{rj} = \int_{V_i} \chi_i(\vec{x})\, \varphi_{ir}(\vec{x})\, \varphi_{ij}(\vec{x})\, dV \qquad j, r \in [1,\, d_i], \tag{47}$$

where $\chi_i(\vec{x})$ represents the physical property (ρ_i or one of the Λ_i components) varying inside the element.

5.1.3 Time discretisation

The time integration of the system (45) relies on the second-order explicit leap-frog scheme that allows to compute alternatively the velocity and the stress components between a half time step. The system (45) can be written as

$$\begin{aligned}
\rho_i(\mathcal{I}_3 \otimes \mathcal{K}_i)\frac{\vec{v}_i^{\,n+\frac{1}{2}} - \vec{v}_i^{\,n-\frac{1}{2}}}{\Delta t} &= -\sum_{\theta\in\{x,y,z\}} (\mathcal{M}_\theta \otimes \mathcal{E}_{i\theta})\vec{\sigma}_i^{\,n} + \frac{1}{2}\sum_{k\in N_i}\left[(\mathcal{P}_{ik}\otimes\mathcal{F}_{ik})\vec{\sigma}_i^{\,n} + (\mathcal{P}_{ik}\otimes\mathcal{G}_{ik})\vec{\sigma}_k^{\,n}\right] \\
(\Lambda_i \otimes \mathcal{K}_i)\frac{\vec{\sigma}_i^{\,n+1} - \vec{\sigma}_i^{\,n}}{\Delta t} &= -\sum_{\theta\in\{x,y,z\}} (\mathcal{N}_\theta \otimes \mathcal{E}_{i\theta})\vec{v}_i^{\,n+\frac{1}{2}} \\
&\quad + \frac{1}{2}\sum_{k\in N_i}\left[(\mathcal{Q}_{ik}\otimes\mathcal{F}_{ik})\vec{v}_i^{\,n+\frac{1}{2}} + (\mathcal{Q}_{ik}\otimes\mathcal{G}_{ik})\vec{v}_k^{\,n+\frac{1}{2}}\right],
\end{aligned} \tag{48}$$

where the superscript n indicates the time step. We chose to apply the definition of the time step as given by Käser et al. (2008), which links the mesh width and time step as follows

$$\Delta t < \min_i\left(\frac{1}{2k_i+1}\cdot\frac{2r_i}{V_{Pi}}\right), \tag{49}$$

where r_i is the radius of the sphere inscribed in the element indexed by i, V_{Pi} is the P-wave velocity in the element, and k_i is the polynomial degree used in the element. Equation (49) is a heuristic stability criterion that usually works well. However, there is no mathematical proof for unstructured meshes that guarantees numerical stability.

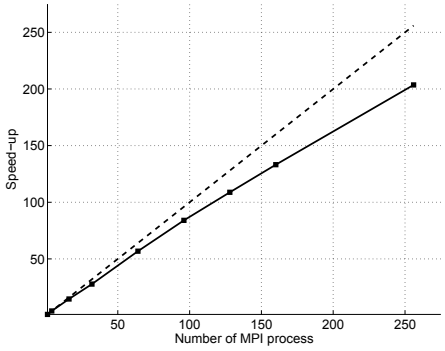


Fig. 10. Speed-up observed when the number of MPI processes is increased from 1 to 256 for modelling with a mesh of 1.8 million P_2 elements. The ideal speed-up is plotted with a dashed line, the observed speed-up with a continuous line. These values were observed on a computing platform with bi-processor quad core Opteron 2.3 GHz CPUs interconnected with Infiniband at 20 Gb/s.

5.1.4 Computational aspects

The DG-FEM is a local method, and therefore it is naturally suitable for parallel computing. In our implementation, the parallelism relies on a domain-partitioning strategy, assigning one subdomain to one CPU. This corresponds to the single program mutiple data (SPMD) architecture, which means that there is only one program and each CPU uses the same executable to work on different parts of the 3D mesh. Communication between the subdomains is performed with the message passing interface (MPI) parallel environment (Aoyama & Nakano, 1999), which allows for applications to run on distributed memory machines. For efficient load balancing among the CPUs, the mesh is divided with the partitioner METIS (Karypis & Kumar, 1998), to balance the number of elements in the subdomains, and to minimise the number of adjacent elements between the subdomains. These two criteria are crucial for the efficiency of the parallelism on large-scale numerical simulations. Figure 10 shows the observed speed-up (i.e. the ratio between the computation time with one CPU, and the computation time with N CPUs) when the number of MPI processes is increased from 1 to 256, for strong scaling calculations on a fixed mesh of 1.8 million P_2 elements. This figure shows good efficiency of the parallelism, of around 80%. In our formulation, another key point is the time step, which is common for all of the subdomains. The time step should satisfy the stability condition given in equation (49) for every element. Consequently, the element with the smallest time step imposes its time step on all of the subdomains. We should mention here a more elaborate approach with local time stepping (Dumbser et al., 2007) that allows for elements to have their own time step independent of the others. Nevertheless, the p-adaptivity offered by DG-FEM allows mitigation of the computational burden resulting from the common time step as we shall see.

5.1.5 Source excitation

We proceed with the addition of the excitation to incremental increase of each involved field component. The excitation of a point source is projected onto the nodes of the element that

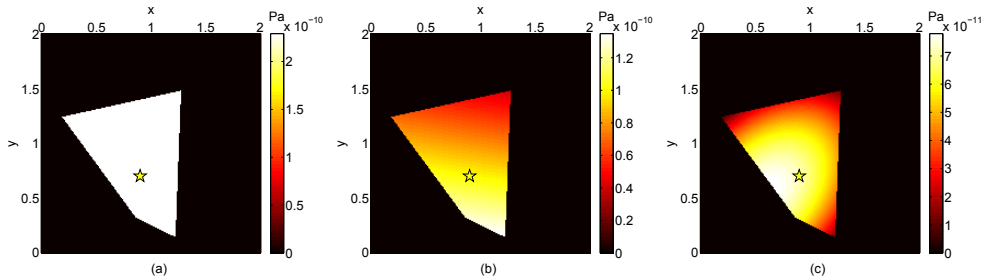


Fig. 11. (a) Cross-section of the mesh near the source position, indicated with a yellow star in the xy plane. This view represents the spatial support of the stress component in a P_0 element containing the point source. (b) Same as (a) with a P_1 element. (c) Same as (a) with a P_2 element.

contains the source as follows

$$\vec{s}_i^n = \frac{\vec{\varphi}_i(\vec{x}_s)}{\sum_{j=1}^{d_i} \varphi_{ij}(\vec{x}_s) \int_{V_i} \varphi_{ij}(\vec{x}) dV} s(t), \tag{50}$$

with $\vec{s}_i^n$ the nodal values vector associated to the excited component, $t = n\Delta t$, $\vec{x}_s$ the position of the point source and $s(t)$ the source function. Equation (50) gives the source term that should be added to the right-hand side of equation (48) for the required components. It should be noticed that this term is only applied to the element containing the source. Depending on the approximation order, the spatial support of the source varies. Figure 11.a shows that the support of a P_0 element is actually the whole volume of the element (represented on the cross-section with a homogeneous white area). In this case, no precise localisation of the source inside the element is possible due to the constant piece-wise interpolation approximation. On the other hand, in a P_1 element (Figure 11.b), the spatial support of the source is linear and allows for a rough localisation of the source. In a P_2 element (Figure 11.c), the quadratic spatial support tends to resemble the expected Dirac in space close to the source position. It should be noted that the limitations concerning source localisation also apply to the solution extraction at the receivers, according to the approximation order of the elements containing the receivers.

5.1.6 Free surface condition

Among the various approaches presented previously, we proceed by considering that the free surface follows the mesh elements. For the element faces located on the free surface, we use an explicit condition by changing the flux expression locally. This is carried out with the concept of virtual elements, which are exactly symmetric to the elements located on the free surface. Inside the virtual elements, we impose a velocity wavefield that is identical to the wavefield of the corresponding inner elements, and we impose an opposite stress wavefield on this virtual element. Thanks to the nodal formulation, the velocity is seen as continuous across the free surface, while the stress is equal to zero on the faces related to the free surface.

This is a quite natural approach similar to the one used in continuous finite-element methods where the test function is set to zero on the free surface boundary.

5.1.7 Absorbing boundary condition

We proceed through some simulations of wave propagation in a homogeneous, isotropic and purely elastic medium for an illustration of CPML conditions. The model size is 8 km $\times$ 8 km $\times$ 8 km, and the medium properties are: V_P = 4000 m/s, V_S = 2310 m/s and ρ = 2000 kg/m^3. An explosive source is placed at coordinates (x_s= 2000 m, y_s = 2000 m, z_s = 4000 m) and a line of receivers is located at coordinates (3000 m $\leq x_r \leq$ 6000 m, y_r = 2000 m, z_r = 4000 m) with 500 m between receivers. The conditions of the tests are particularly severe, since the source and the receivers are located close to the CPMLs (at a distance of 250 m), thus favouring grazing waves. The source signature is a Ricker wavelet with a dominant frequency of 3 Hz and a maximum frequency of about 7.5 Hz. Due to the explosive source, only P-wave is generated and the minimum wavelength is about 533 m. The mesh contains 945,477 tedrahedra with an average edge of 175 m, making a discretisation of about 3 elements per λ_{min}. Figures 12.c and 12.d show the results obtained with the P_2 interpolation and CPMLs of 10-elements width (L_{cpml} = 1750 m) at all edges of the model. With the standard scale, no reflection can be seen from the CPMLs. When the amplitude is magnified by a factor of 100, some spurious reflections are visible. This observation is in agreement with the theoretical reflection coefficient ($R_{coeff} = 0.1\%$) in equation (20).
As shown by Collino & Tsogka (2001), the thickness of the absorbing layer plays an important role in the absorption efficiency. In Figures 12.a and 12.b, the same test was performed with CPMLs of 5-elements width (L_{cpml} = 875 m) at all edges of the model. Compared to Figures 12.c and 12.d, the amplitude of the reflections have the same order of magnitude. Nevertheless, in the upper and left parts of the model, some areas with a strong amplitude appear close to the edges. These numerical instabilities arise at the outer edges of the CPMLs, and they expand over the complete model during the simulations.

Instabilities of PML in long time simulations have been studied in electromagnetics (Abarbanel et al., 2002; Bécache et al., 2004). For elastodynamics, remedies have been proposed by Meza-Fajardo & Papageorgiou (2008) for an isotropic medium with standard PML. These authors proposed the application of an additional damping in the PML, onto the directions parallel to the layer, leading to a multiaxial PML (M-PML) which does not follow strictly the matching property of PML in the continuum and which has a less efficient absorption power. Through various numerical tests, Etienne et al. (2010) has shown that instabilities could be delayed outside the time window of simulation when considering extended M-PML from CPML.

5.1.8 Saving computation time and memory

Table 2 gives the computation times for updating the velocity and stress wavefields in one element for one time step, for different approximation orders, without or with the update of the CPML memory variables (i.e. elements located outside or inside the CPMLs). These computation times illustrate the significant increase with respect to the approximation order, and they allow an evaluation of the additional costs of the CPML memory variables computation from 40% to 60%. The effects of this additionnal cost have to be analysed in

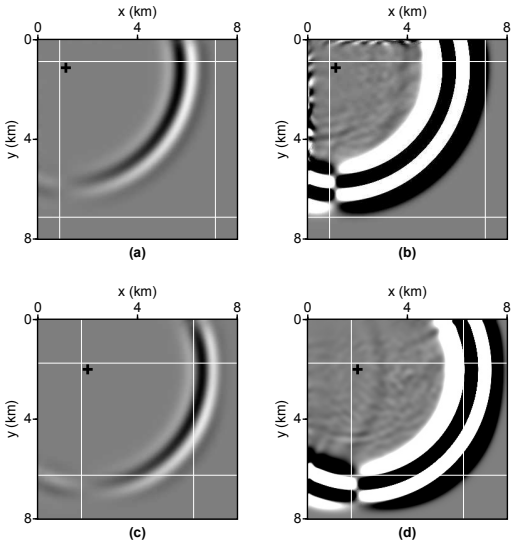


Fig. 12. Snapshots at 1.6 s of the velocity component v_x in the plane xy that contains the source location. CPMLs of 10-elements width are applied at all edges of the model. The modelling was carried out with P_2 interpolation. White lines, the limits of the CPMLs; black cross, the position of the source. (a) Real amplitude. (b) Amplitude magnified by a factor of 100. (c) & (d) Same as (a) & (b) with CPMLs of 5-elements width.

Approximation order	Element outside CPML	Element inside CPML
P_0	2.6 μs	3.6 μs
P_1	5.0 μs	8.3 μs
P_2	21.1 μs	29.9 μs

Table 2. Computation times for updating the velocity and stress wavefields in one element for one time step. These values correspond to average computation times for a computing platform with bi-processor quad core Opteron 2.3 GHz CPUs interconnected with Infiniband 20 at Gb/s.

the context of a domain-partitioning strategy. The mesh is divided into subdomains, using a partitioner. Figure 13.a shows the layout of the subdomains that were obtained with the partitioner METIS (Karypis & Kumar, 1998) along the xy plane used in the previous validation tests. The mesh was divided into 32 partitions, although only a few of these are visible on the cross-section in Figure 13.a. We used an unweighted partitioning, meaning that each partition contains approximately the same number of elements.

The subdomains, partially located in the CPMLs, contain different numbers of CPML elements. In large simulations, some subdomains are totally located inside the CPMLs, and some others outside the CPMLs. In such a case, the extra computation costs of the subdomains located in the absorbing layers penalise the whole simulation. Indeed, most of the subdomains spend 40% to 60% of the time just waiting for the subdomains located in the CPMLs to

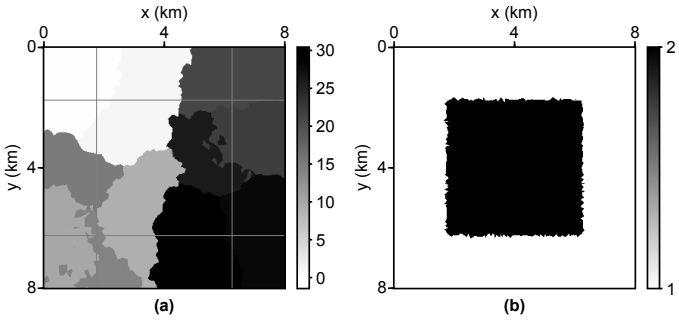


Fig. 13. (a) Layout of the subdomains obtained with the partitioner METIS (Karypis & Kumar, 1998) along the xy plane that contains the source location. Grey lines, the limits of the CPMLs. The mesh was divided into 32 partitions, although only a few of these are visible on this cross-section. (b) View of the approximation order per element along the same plane. Black, the P_2 elements; white, the P_1 elements.

complete the computations at each time step. For a better load balancing, we propose to benefit from the p-adaptivity of DG-FEM, using lower approximation orders in the CPMLs. Indeed, inside the absorbing layers, we do not need a specific accuracy, and consequently the approximation order can be decreased. Table 2 indicates that such a mixed numerical scheme is advantageous, since the computation time required for a P_0 or P_1 element located in the CPML is shorter than the computation time of a standard P_2 element. Figure 13.b shows the approximation order per element when P_1 is used in the CPMLs and P_2 in the rest of the medium. We should note here that the interface between these two areas is not strictly aligned to a cartesian axis, and has some irregularities due to the shape of the tetrahedra. Although it is possible to constrain the alignment of the element faces parallel to the CPML limits, we did not observe significant differences in the absorption efficiency whether the faces are aligned or not.

Figure 14.a shows the seismograms computed when the modelling was carried out with P_2 inside the medium and P_1 in the CPMLs. Absorbing layers of 10-elements width are applied at all edges of the model. For comparison, Figure 14.b shows the results obtained with P_2 inside the medium and P_0 in the CPMLs. In this case, the spurious reflections have significant amplitudes, preventing any use of these seismograms. On the other hand, the seismograms computed with the mixed scheme P_2/P_1 show weak artefacts, and are reasonably comparable with the seismograms obtained with complete P_2 modelling. Therefore, taking into account that the computation time and the memory consumption of the P_2/P_1 simulation are nearly half of those required with the full P_2 modelling, we can conclude that this mixed numerical scheme is of interest. It should be noticed that it is possible to adopt a weighted partitioning approach to overcome partly load balancing issues We should also stress that the saving in CPU time and memory provided with this kind of low-cost absorbing boundary condition is crucial for large 3D simulations, and this becomes a must in the context of 3D seismic imaging applications that require a lot of forward problems, such as FWI.

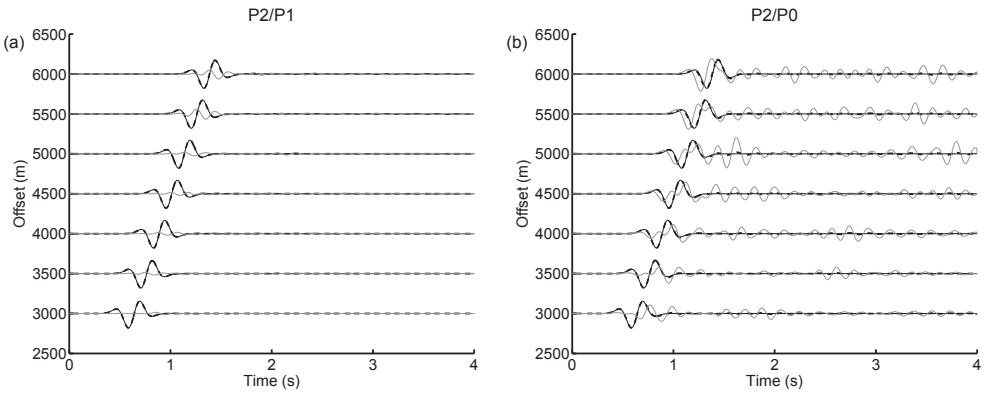


Fig. 14. (a) Seismograms of the velocity component v_x. The amplitude of each seismogram is normalised. The modelling is done with P_1 in the CPMLs and P_2 inside the medium. Black continuous line, numerical solution in large model without reflection in the time window; dashed line, numerical solution with 10-elements width CPMLs; grey line, residuals magnified by a factor of 10. (b) Same as (a) except the modelling is done with P_0 in the CPMLs and P_2 inside the medium.

5.1.9 Accuracy of DG-FEM with tetrahedral meshes

There are a variety of studies in the literature concerning the dispersive and dissipative properties of DG-FEM with reference to wave-propagation problems. Let us quote few examples: Ainsworth et al. (2006) provided a theoretical study for the 1D case; Basabe et al. (2008) analysed the effects of basis functions on 2D periodic and regular quadrilateral meshes; and Käser et al. (2008) discussed the convergence of the DG-FEM combined with ADER time integration and 3D tetrahedral meshes. More related to our particular concern here, Delcourte et al. (2009) provided a convergence analysis of the DG-FEM with a centred flux scheme and tetrahedral meshes for elastodynamics. They demonstrated the sensitivity of the DG-FEM to the mesh quality, and they proved that the convergence is limited by the second-order time integration we have used in the present study, despite the order of the basis function. Specific analysis of the convergence in the scheme we have presented could be found in Etienne et al. (2010).

5.2 2D finite-element discontinuous Galerkin method in the frequency domain

On land exploration seismology, there is a need to perform elastic wave modeling in area of complex topography such as foothills and thrust belts (Figure 15) in the frequency domain. Moreover, onshore targets often exhibit weathered layers with very low wave speeds in the near surface which require a locally-refined discretisation for accurate modeling. In shallow water environment, a mesh refinement is also often required near the sea floor for accurate modeling of guided and interface waves near the sea floor. Accurate modeling of acoustic and elastic waves in presence of complex boundaries of arbitrary shape and the local adaptation of the discretisation to local features such as weathered near surface layers or sea floor

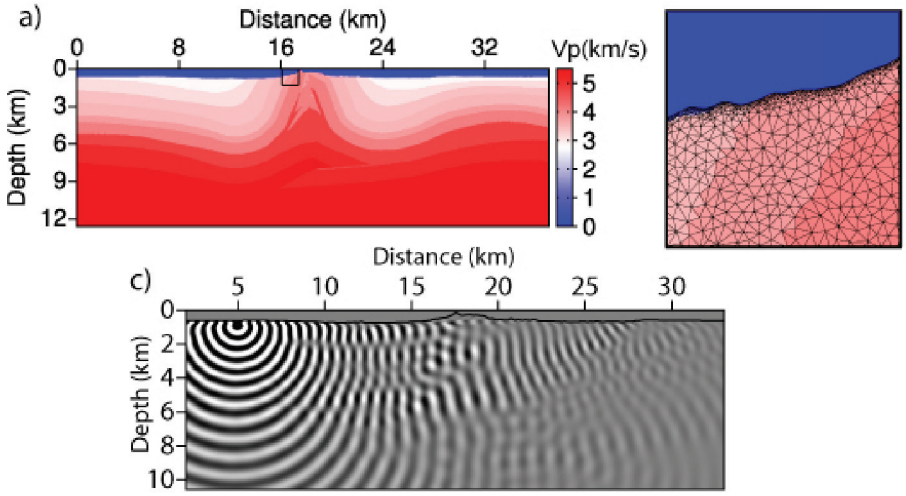


Fig. 15. Application of the DG method in seismic exploration. (a) Velocity model representative of a foothill area affected by a hilly relief and a weathered layer in the near surface. (b) Close-up of the unstructured triangular mesh locally refined near the surface. (c) Example of monochromatic pressure wavefield.

were two of our motivations behind the development of a discontinuous element method on unstructured meshes for acoustic and elastic wave modeling.

5.2.1 hp-adaptive discontinuous Galerkin discretisation

Similarly to the time formulation we adopt the nodal form of the DG formulation, assuming that the wavefield vector is approximated in triangular elements for 2D geometry which leads to the following expression,

$$\vec{u}_i(\omega, x, y, z) = \sum_{j=1}^{d_i} \vec{u}_{ij}(\omega, x_j, y_j, z_j)\varphi_{ij}(\omega, x, y, z), \tag{51}$$

where $\vec{u}$ is the wavefield vector of components such as the following vector $\vec{u} = (p, v_x, v_y, v_z)$ for acoustic propagation. The index of the element in an unstructured mesh is denoted by i. The expression $\vec{u}_i(\omega, x, y, z)$ denotes the wavefield vector in the element i and (x, y, z) are the coordinates inside the element i. In the framework of the nodal form of the DG method, φ_{ij} denotes Lagrange polynomial and d_i is the number of nodes in the element i. The position of the node j in the element i is denoted by the local coordinates (x_j, y_j, z_j).

In the frequency domain, the pseudo-conservative form (41) could be written in a 2D geometry as

$$\mathcal{M}\vec{u} = \sum_{\theta \in \{x,y,z\}} \partial_\theta \left(\mathcal{N}_\theta \vec{u}\right) + \vec{s}, \tag{52}$$

where $\mathcal{N}_\theta \vec{u}$ are linear fluxes and the source vector is denoted by $\vec{s}$. Expressions of matrices $\mathbf{M}$ and $\mathbf{N}$ could be found in Brossier et al. (2010).

The weak form of the system (52) is similar in the frequency domain and proceed by selecting a test function φ_{ir} and then an integration over the element volume V_i which gives

$$\int_{V_i} \varphi_{ir}\, \mathcal{M}_i \vec{u}_i \, dV = \int_{V_i} \varphi_{ir} \sum_{\theta \in \{x,y,z\}} \partial_\theta \left(\mathcal{N}_\theta \vec{u}_i \right) \, dV + \int_{V_i} \varphi_{ir} \vec{s}_i dV, \tag{53}$$

where the quantity $r \in [1, d_i]$. In the framework of Galerkin methods, we used the same function for the test function and the shape function. Similar procedures as for the 3D case and related to standard steps of the finite-element method lead to the discrete expression,

$$\left(\mathcal{M}_i \otimes \mathcal{K}_i \right) \vec{\vec{u}}_i = - \sum_{\theta \in \{x,y,z\}} \left(\mathcal{N}_\theta \otimes \mathcal{E}_{i\theta} \right) \vec{\vec{u}}_i + \frac{1}{2} \sum_{k \in N_i} \left[\left(\mathcal{Q}_{ik} \otimes \mathcal{F}_{ik} \right) \vec{\vec{u}}_i + \left(\mathcal{Q}_{ik} \otimes \mathcal{G}_{ik} \right) \vec{\vec{u}}_k \right] + \left(\mathcal{I} \otimes \mathcal{K}_i \right) \vec{\vec{s}}_i \tag{54}$$

where the mass matrix $\mathcal{K}_i$, the stiffness matrix $\mathcal{E}_i$ and the flux matrices $\mathcal{F}_i$ and $\mathcal{G}_i$ are similar to those defined for the 3D case (equation (46)). The matrix $\mathcal{Q}$ is also defined as for the 3D case (equation (46))

It is worth repeting that, in the equation (46), arbitrary polynomial order of the shape functions can be used in elements i and k indicating that the approximation orders are totally decoupled from one element to another. Therefore, the DG allows for varying approximation orders in the numerical scheme, leading to the p-adaptivity.

The equation (54) can be recast in matrix form as

$$\mathbf{B}\, \mathbf{u} = \mathbf{s}. \tag{55}$$

5.2.2 Which interpolation orders to choose?

For the shape and test functions, we used low-order Lagrangian polynomials of orders 0, 1 and 2, referred to as P_k, $k \in 0, 1, 2$ in the following (Brossier, 2009; Etienne et al., 2009). Let us remind that our motivation behind seismic modeling is to perform seismic imaging of the subsurface by full waveform inversion, the spatial resolution of which is half the propagated wavelength and that the physical properties of the medium are piecewise constant per element in our implementation of the DG method. The spatial resolution of the FWI and the piecewise constant representation of the medium direct us towards low-interpolation orders to achieve the best compromise between computational efficiency, solution accuracy and suitable discretisation of the computational domain. The P_0 interpolation (or finite volume scheme) was shown to provide sufficiently-accurate solution on 2D equilateral triangular mesh when ten cells per minimum propagated wavelength are used (Brossier et al., 2008), while 10 cells and 3 cells per propagated wavelengths provide sufficiently-accurate solutions on unstructured triangular meshes with the P_1 and the P_2 interpolation orders, respectively (Brossier, 2011). Of note, the P_0 scheme is not convergent on unstructured meshes when centered fluxes are used (Brossier et al., 2008). This prevents the use of the P_0 scheme in 3D medium where uniform tetrahedral meshes do not exist (Etienne et al., 2010). A second remark is that the finite volume scheme on square cells is equivalent to second-order accurate

	FD^{2D}	$DG^{2D}_{P_0}$	$DG^{2D}_{P_1}$	$DG^{2D}_{P_2}$
n_d	1	1	3	6
n_z	9	5-9	13-25	24-48

Table 3. Number of nodes per element (n_d) and number of non-zero coefficients per row of the impedance matrix (n_z) for the FD and DG methods. The number n_z depends on the number of wavefield components involved in the r.h.s of the first-order wave equation n_{der}.

FD stencil (Brossier et al., 2008) which is consistent with a discretisation criterion of 10 grid points per wavelength (Virieux, 1986). Use of interpolation orders greater than 2 would allow us to use coarser meshes for the same accuracy but these coarser meshes would lead to an undersampling of the subsurface model during imaging. On the other hand, use of high interpolation orders on mesh built using a criterion of 4 cells par wavelength would provide an unnecessary accuracy level for seismic imaging at the expense of the computational cost resulting from the dramatic increase of the number of unknowns in the equation (55).

The computational cost of the LU decomposition depends on the numerical bandwidth of the matrix, the dimension of the matrix (i.e., the number of rows/columns) and the number of non-zero coefficients per row (n_z). The dimension of the matrix depends in turn of the number of cell (n_{cell}), of the number of nodes per cell (n_d) and the number of wavefield components (n_{wave}) (ranging from 3 to 5 in 2D geometry). The number of nodes in a 2D triangular element is given by Hesthaven & Warburton (2008) and leads to the following expression $n_d = (k+1)(k+2)/2$ where k denotes the interpolation order similar to what is done in the 3D geometry.

The numerical bandwidth is not significantly impacted by the interpolation order. The dimension of the matrix and the number of non-zero elements per row of the impedance matrix are respectively given by $n_{wave} \times n_d \times n_{cell}$ and $(1 + n_{neigh}) \times n_d \times n_{der} + 1$, where n_{neigh} is the number of neighbor cell (3 in 2D geometry) and n_{der} is the number of wavefield components involved in the r.h.s of the velocity-pressure wave equation, equation (52). Table 3 outlines the number of non-zero coefficients per row for the mixed-grid FD and DG methods. Increasing the interpolation order will lead to an increase of the number of non-zero coefficients per row, a decrease of the number of cells in the mesh and an increase of the number of nodes in each element. The combined impact of the 3 parameters n_z, n_{cell}, n_d on the computational cost of the DG method makes difficult the definition of the optimal discretisation of the frequency-domain DG method. The medium properties should rather drive us towards the choice of a suitable discretisation.

One must underline that the *LU* factorization is quite demanding in computer memory and has also some drawbacks for scalability, suggesting that nodes with high memory should be preferred at the expense of the CPU numbers.

5.2.3 Boundary conditions and source implementation

Absorbing boundary conditions are implemented with unsplitted PML in the frequency-domain DG method (Brossier, 2011) following the same approach than for the FD method: one can see that the PML implementation in the frequency is straightforward. We have found that constraining the meshing to have edges of elements in the PML zone parallel to the direction of dissipation of the waves improves the efficiency.

Free surface boundary condition is implemented with the method of image. A virtual cell is considered above the free surface with the same velocity and the opposite pressure components to those below the free surface. This allows us to fulfill the zero pressure condition at the free surface while keeping the correct numerical estimation of the particle velocity at the free surface. Using these particle velocities and pressures in the virtual cell, the pressure flux across the free surface interface vanishes, while the velocity flux is twice the value that would have been obtained by neglecting the flux contribution above the free surface. As in the FD method, this boundary condition has been implemented by modifying the impedance matrix accordingly without introducing explicitely the virtual element in the mesh. The rigid boundary condition is implemented following the same principle except that the same pressure and the opposite velocity are considered in the virtual cell.

Concerning the source excitation, the point source at arbitrary positions in the mesh is implemented by means of the Lagrange interpolation polynomials for $k \geq 1$. This means that the source excitation is performed at the nodes of the cell containing the source with appropriate weights corresponding to the projection of the physical position of the source on the polynomial basis. When the source is located in the close vicinity of a node of a triangular cell, all the weights are almost zero except that located near the source. In the case of the P_2 interpolation, a source close to the vertex of the triangular cell is problematic because the integral of the P_2 basis function over the volume of the cell is zero for nodes located at the vertex of the triangle. In this case, no source excitation will be performed (see equation (54)). To overcome this problem specific to the P_2 interpolation, one can use locally a P_1 interpolation in the element containing the source at the expense of the accuracy or distribute the source excitation over several elements or express the solution in the form of local polynomials (i.e., the so-called modal form) rather than through nodes and interpolating Lagrange polynomials (i.e., the so-called nodal form).

Another issue is the implementation of the source in P_0 equilateral mesh. If the source is excited only within the element containing the source, a checker-board pattern is superimposed on the wavefield solution. This pattern results from the fact that one cell out of two is excited in the DG formulation because the DG stencil does not embed a staggered-grid structure (the unexcited grid is not stored in staggered-grid FD methods; see Hustedt et al. (2004) for an illustration). To overcome this problem, the source can be distributed over several elements of the mesh or P_1 interpolation can be used in the area containing the sources and the receivers, while keeping P_0 interpolation in the other parts of the model (Brossier et al., 2010).

Of note, use of unstructured meshes together with the source excitation at the different nodes of the element contribute to mitigate the checker-board pattern in the in P_1 and P_2 schemes. The same procedure as for the source is used to extract the wavefield solution at arbitrary receiver positions.

6. Realistic examples for highly contrasted and strongly heterogeneous media using finite-elements methods

We shall consider two examples for the illustration of the Discontinuous Galerkin approach. The first one is related to the problem of 3D wave propagation inside an active volcano using the time-domain approach while the second one deals with the problem of 2D wave propagation above a oil reservoir using the frequency-domain approach.

6.1 The volcano *La Soufrière*

6.1.1 Characteristics of the model

La Soufrière of Guadeloupe (France) is one of nine active volcanoes of Lesser Antilles. It belongs to a recent volcanic system situated in the south part of the *Basse-Terre*. A P-wave velocity model of the volcano has been obtained by first arrival time tomography (Coutant et al., 2010). Figure 16 is the reconstructed Vp velocity model that reveals the existence of a high velocity zone below the dome of *La Soufrière*. The dimensions of the model are 1400 $m \times 1400$ $m \times 1000$ m in xyz respectively. We consider a constant Poisson ratio of 0.25 to assess the S-wave velocity model from V_P. The velocity ranges from 660 m/s to 3800 m/s for V_P and 380 m/s to 2200 m/s for V_S. Considering a maximum frequency of 25 Hz, the minimum wavelength is about 15 m. In addition, we consider a constant density equal to 2000 kg/m^3. Absorbing layers of CPML type with a thickness of 300 m are added at each side edge of the model as well at the bottom edge. Therefore, the complete dimensions of the numerical model are 2000 $m \times 2000$ $m \times 1300$ m.

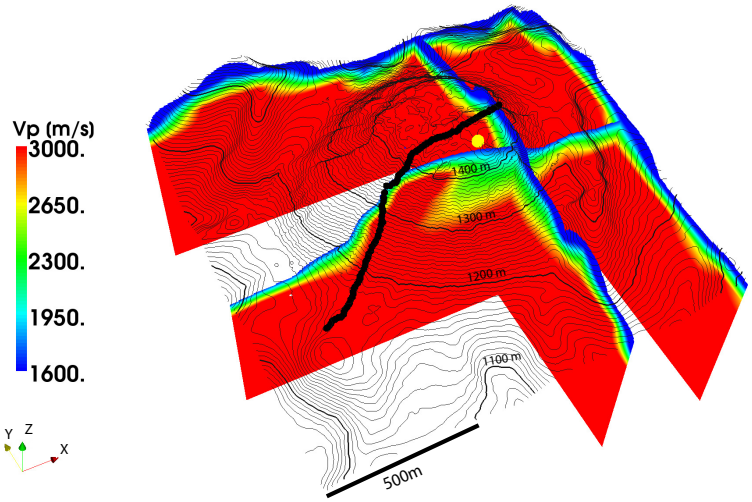


Fig. 16. Topography of the volcano *La Soufrière* with the underlying reconstructed Vp velocity structure. The position of the dynamite shot is indicated with a yellow circle and the receivers with black triangles.

6.1.2 Construction of the tetraedral mesh

The mesh has been built with the mesher TETGEN (Si, 2006) combined with an iterative h-refinement procedure to obtain a locally adapted mesh to the velocity field(with an average of 3 elements per minimum wavelength λ_{min}): a cross-section is shown in the left panel of the Figure 17. For building this mesh, we have started our iterative reconstruction with a uniform mesh shown in the right panel of the Figure 17. After the sixth refinement iteration, the discretization criteria are met. Areas of high velocities are correlated with the parts of the

Modelling time	5 s
Nb elements	4.6 million
Nb unknowns	414 million
Min/Max element edge	1.29 - 58.62 m
Nb time steps	37 787
Nb CPUs	512
Total memory	10 GB
Memory per CPU	14 - 23 MB
Computation time	6 h 45 min.
Time / unknown / time step	0.79 μs

Table 4. Statistics of the modelling for *La Soufrière* performed on an IBM Blue Gene machine.

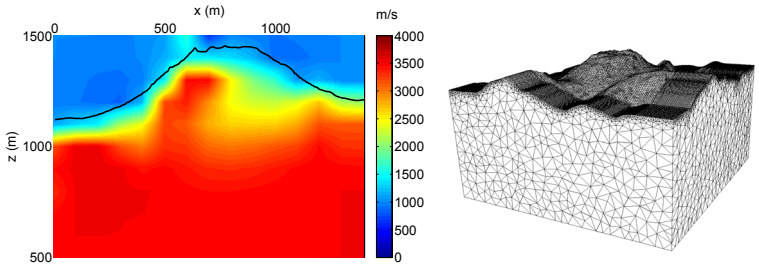


Fig. 17. On the left, cross-section of the P-wave velocity model in the plane xz in the middle of the volcano *La Soufrière*. The back line represents the topography. On the right, initial mesh of the volcano *La Soufrière* from which we deduce automatically the one used for modeling by adapting the mesh size to the local P-wave velocity. Absorbing layers of CPML type with a thickness of 300 m are added at each side edge of the model.

mesh where the elements are the largest ones. On the contrary, near the free surface, we find the finest elements.

6.1.3 Numerical result

We have performed 3D simulations with the Discontinuous Galerkin Finite-Element Method in the time domain. The computations have been performed on a Blue Gene machine with 512 processors. The statistics for these computations are given in Table 4.

The configuration of the seismic acquisition is given in Figure 16. This is a quasi-2D system with a profile according to the East-West direction, which includes 100 single-component receivers (v_z) with 10 m between receivers. The source is a shot of dynamite. For the numerical simulations, we used an explosive source with a Ricker function of dominant frequency of 10 Hz (maximum frequency 25 Hz). We present in Figure 18 a comparison between the observed and computed data. Despite significant uncertainties and approximations (source function, Poisson ratio, density, absence of attenuation, low signal to noise ratio), there are striking similarities in the data. In particular, the seismic traces exhibit well marked discontinuities related to the strong velocity contrasts and the complex topography of the volcano *La Soufrière*.

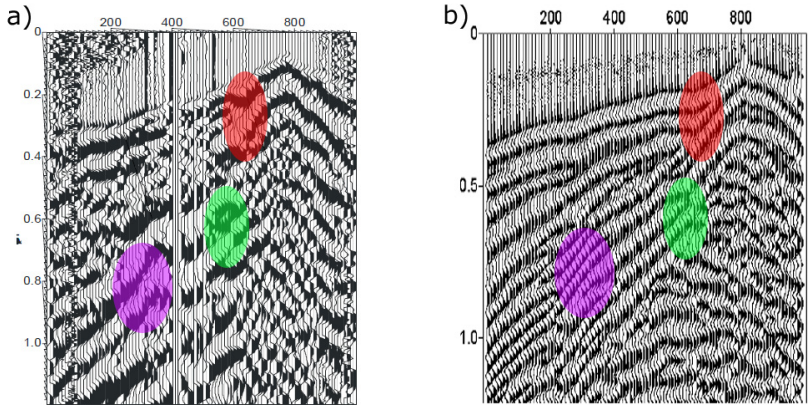


Fig. 18. (a) Recorded seismograms (component v_z). (b) Computed seismograms. Some similarities between both set of data are highlighted with color shapes.

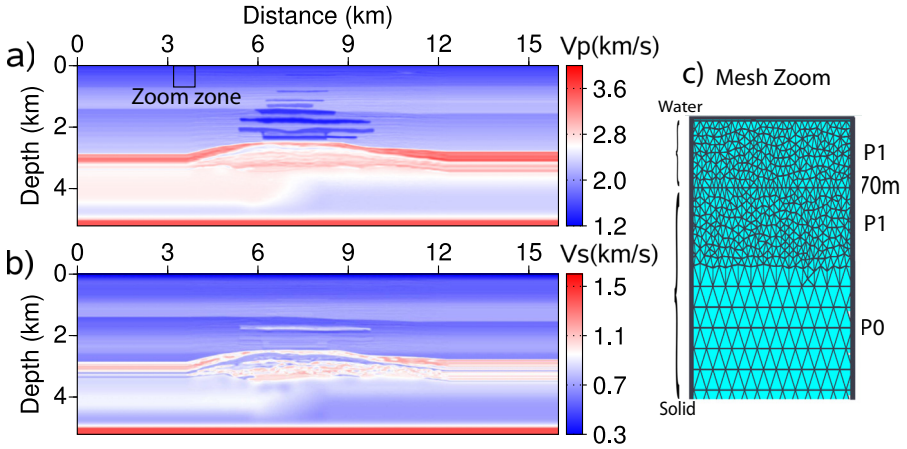


Fig. 19. The synthetic Valhall model for (a) P-wave and (b) S-wave velocities. Panel (c) represents a zoom of the shallow mesh.

6.2 Application 2D in the frequency-domain: the synthetic Valhall application

This 2D application is based on a synthetic representation of the Valhall zone in the North Sea, Norway. This model is representative of oil and gas fields in shallow water environments of the North Sea (Munns, 1985). The model is described as an heterogeneous P- and S- wave velocity model (Figure 19a-b). The water layer is only 70 m depth. The main targets are a gas cloud in the large sediment layer, and in a deeper part of the model, the trapped oil underneath the cap rock, which is formed of chalk. Gas clouds are easily identified by the low P-wave velocities, whereas their signature is much weaker in the V_S model, as gas does not affect S-waves propagation.

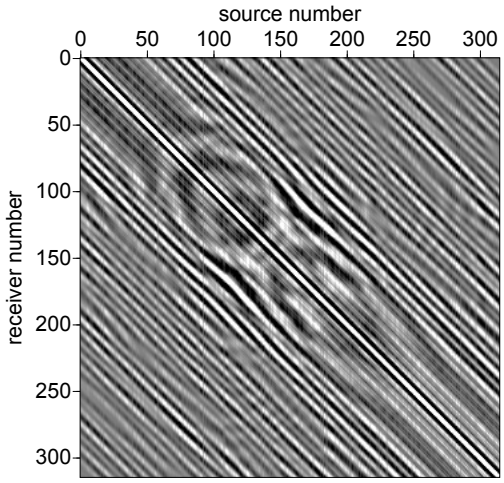


Fig. 20. Frequency-domain data for the hydrophone component at 4 Hz. The data (real-part) are plotted in the source/receiver domain.

In order to investigate seismic imaging in such environment, the selected acquisition mimics a four-component ocean-bottom cable survey (Kommedal et al., 2004), as is deployed on the field. A line of 315 explosive sources is positioned 5 m below water surface to simulated air-gun sources and a cable of 315 3-components sensors is located on the sea floor (1 hydrophone and 2 geophones). This geological setting, which is composed of a significant soft sea-bed with high Poisson'ratio due to soft and unconsolidated sediments leads to a particularly ill-posed problem for S-wave velocity reconstruction, due to the relatively small shear-wave velocity contrast at the sea bed, which prevents recording of significant P-to-S converted waves.

For the meshing of the model, the narrow velocity range in most parts of the model requires the use of a regular mesh as much as possible for computational efficiency. However, to correctly discretize the shallow-water layer and liquid-solid interface, a *p*-adapted mesh implemented with a mixed P_0-P_1 interpolation is chosen (Figure 19c): a refined unstructured P_1 layer of cells is used for the first 130 m of the subsurface for accurate modeling of the interface waves at the liquid/solid interface, and for accurate positioning of the sources, located 5 m below the surface, and of the receivers, located on the sea floor. Below this 130 m depth zone, a regular equilateral mesh is used in combination of a P_0 interpolation.

Figure 20 illustrates an example of frequency-domain data (real-part of the complex-value wavefield) at 4Hz. These data are plotted in the source/receiver domain for the full acquisition survey. The diagonal part of the figure represents the collocation of the source and the closest receiver, as the source moves in the acquisition. The Figure 21 illustrates a time-domain shot-gather for the 3 components of the sensors and a source located at position 4 km : the frequency-domain solutions at all the receiver positions have been computed for the single source at all the frequencies of the source spectrum, between 0 and

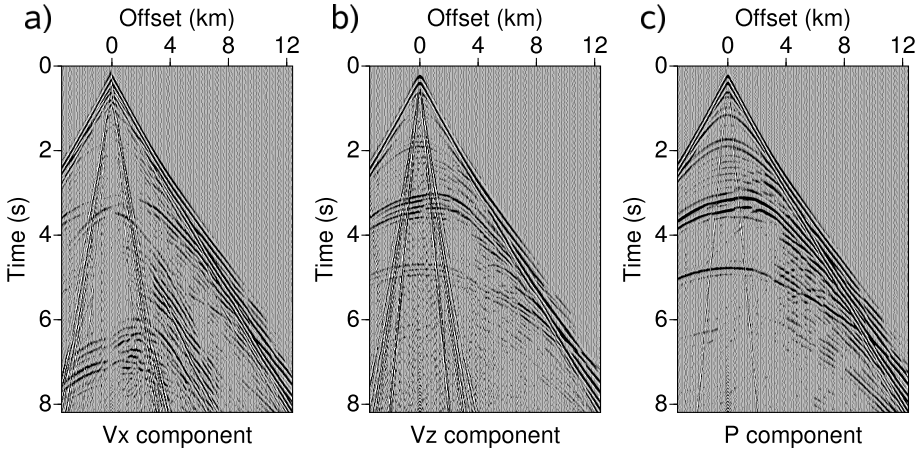


Fig. 21. Time-domain shot-gather for a source located at position 4 km. The data are computed from frequency-domain simulation and Fourier transformed in the time-domain for (a) the horizontal geophone, (b) the vertical geophone and (c) the hydrophone components.

13 Hz. These frequency-domain complex-values data have then been Fourier transform to the time-domain. The time-domain show specific properties of propagation in such environments : the hydrophone and the vertical geophones are mainly sensitive to P-wave arrivals that dominate the elastic propagation in soft-sediment zones. The horizontal geophone allows however to record some late P-to-S conversions, which could be used to image the V_S model from seismic imaging methods.

7. Conclusion

We have presented mainly two families of techniques for solving partial differential equations for elastodynamics: some finite-differences formulations in both time domain and frequency domain and some finite-element methods also in both time domain and frequency domain. Both approaches have appealing features, especially when considering seismic imaging where numerous forward problems should be performed. Such classification helps to understand the advantages and limitations of each particular method to model a specific physical phenomenon

The discretization of the strong formulation of the partial differential equations has been presented through finite-difference techniques. These approaches are easy to implement and quite flexible. They are currently the methods of choice for large-scale modelling and inversion in exploration geophysics, especially in the marine environment. They may however demand a very fine discretization when the earth model contains large contrasts; and accurately modelling the responses around a sharp interface is quite challenging. We have introduced various perspectives as summation-by-parts formulation or the immersed boundary approach as well as simple mesh deformation might broaden the use of finite-differences techniques by avoiding the stair-case approximation.

The weak formulation expressed in the finite-element methods has been considered under the specific family of Discontinuous Galerkin approaches. The use of test functions gives us more freedom and the integral form provides us flexibility in the meshing. However, they lead to numerical challenges: they are more difficult to implement than the finite-difference method, they are often more expensive in computational time and memory, and they are more complicated to use because the accuracy of the response depends on the quality of the meshing. Therefore, they are not intensively used for seismic imaging and are until now more oriented to seismic modeling in the final reconstructed model.

It should be noticed that attempts exist to combine the advantages of these methods in one approach for computing elastic fields, at least for specific applications. Even, one can think that decoupling the inverse problem procedure and the forward problem is possible: we can flip-flop between the two forward problem formulations inside iterations of the inverse problem.

When the modelling method serves as the kernel of an inversion algorithm, additional constrains generally appear because the gradient of the misfit functional needs to be evaluated. The choice of the modelling approach notably depends (1) on the needed accuracy, (2) the efficiency in evaluating the solution and the gradient of the misfit functional in an inversion algorithm, and (3) the simplicity of use.

Finally, the practical implementation shall probably be adapted to the data acquisition. Densely sampled acquisition in exploration geophysics with or without blending, or in lithospheric investigation with the recent deployment of sensors such as the USarray experiment challenges our modelling choice. This seems to indicate that development in modelling and associated inversion approaches remain crucial to improve our knowledge of the subsurface, notably by extracting more information from the, ever larger, recorded data sets.

8. Acknowledgements

We are gratefull to René-Édouard Plessix (SHELL) and Henri Calandra (TOTAL) for fruitful discussions. This work was partially performed using HPC resources from GENCI-[CINES/IDRIS] (Grant 2010-046091 & Grant 2010-082280). Facilities from mesocentres SIGAMM in Nice and CIMENT in Grenoble are also greatly acknowledged. Diego Mercerat received support from the European Research Council (ERC) Advanced grant 226837, and a Marie Curie Re-integration grant (project 223799).

9. References

Aagaard, B. T., Hall, J. F. & Heaton, T. H. (2001). Characterization of near-source ground motion with earthquake simulations, *Earthquake Spectra* 17: 177–207.

Abarbanel, S., Gottlieb, D. & Hesthaven, J. S. (2002). Long-time behavior of the perfectly matched layer equations in computational electromagnetics, *Journal of scientific Computing* 17: 405–422.

Ainsworth, M., Monk, P. & Muniz, W. (2006). Dispersive and Dissipative Properties of Discontinuous Galerkin Finite Element Methods for the Second-Order Wave Equation, *Journal of Scientific Computing* 27(1-3): 5–40.

Akcelik, V., Bielak, J., Biros, G., Epanomeritakis, I., Fernandez, A., Ghattas, O., Kim, E. J., Lopez, J., O'Hallaron, D., Tu, T. & Urbanic, J. (2003). High resolution forward and inverse earthquake modeling on terascale computers, *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, IEEE Computer Society, Washington, DC, USA, p. 52.

Aki, K. & Richards, P. G. (2002). *Quantitative seismology, theory and methods, second edition*, University Science Books, Sausalito,California.

Alterman, Z. & Karal, F. C. (1968). Propagation of elastic waves in layared media by finite-difference methods, *Bulletin of the Seismological Society of America* 58: 367–398.

Aminzadeh, F., Brac, J. & Kunz, T. (1997). *3-D Salt and Overthrust models*, SEG/EAGE 3-D Modeling Series No.1.

Aoi, S. & Fujiwara, H. (1999). 3D finite-difference method using discontinuous grids, *Bulletin of the Seismological Society of America* 89: 918–930.

Aoyama, Y. & Nakano, J. (1999). *RS/6000 SP: Practical MPI Programming*, Red Book edn, IBM Corporation, Texas.

Basabe, J. D., Sen, M. & Wheeler, M. (2008). The interior penalty discontinuous galerkin method for elastic wave propagation: grid dispersion, *Geophysical Journal International* 175: 83–93.

Bécache, E., Petropoulos, P. G. & Gedney, S. G. (2004). On the long-time behavior of unsplit perfectly matched layers, *IEEE Transactions on Antennas and Propagation* 52: 1335–1342.

Ben Jemaa, M., Glinsky-Olivier, N., Cruz-Atienza, V. M. & Virieux, J. (2009). 3D Dynamic rupture simulations by a finite volume method, *Geophysical Journal International* 178: 541–560.

Ben Jemaa, M., Glinsky-Olivier, N., Cruz-Atienza, V., Virieux, J. & Piperno, S. (2007). Dynamic non-planar crack rupture by a finite volume method, *Geophysical Journal International* 172(1): 271–285.

Berenger, J.-P. (1994). A perfectly matched layer for absorption of electromagnetic waves, *Journal of Computational Physics* 114: 185–200.

Berge-Thierry, C., Cotton, F., Scotti, O., Pommera, D. & Fukushima, Y. (2003). New empirical response spectral attenuation laws for moderate european earthquakes, *Journal of Earthquake Engineering* 7(2): 193–222.

Bland, D. (1960). *The theory of linear viscoelasticity*, Pergamon Press, Oxford.

Bohlen, T. & Saenger, E. H. (2006). Accuracy of heterogeneous staggered-grid finite-difference modeling of Rayleigh waves, *Geophysics* 71: 109–115.

Bolt, B. & Smith, W. (1976). Finite element computation of seismic anomalies from bodies of arbitrary shape, *Geophysics* 41: 145–150.

Brossier, R. (2009). *Imagerie sismique à deux dimensions des milieux visco-élastiques par inversion des formes d'onde: développements méthodologiques et applications.*, PhD thesis, Université de Nice-Sophia-Antipolis.

Brossier, R. (2011). Two-dimensional frequency-domain visco-elastic full waveform inversion: Parallel algorithms, optimization and performance, *Computers & Geosciences* 37(4): 444 – 455.

Brossier, R., Etienne, V., Operto, S. & Virieux, J. (2010). Frequency-domain numerical modelling of visco-acoustic waves based on finite-difference and finite-element discontinuous galerkin methods, *in* D. W. Dissanayake (ed.), *Acoustic Waves*, SCIYO, pp. 125–158.

Brossier, R., Operto, S. & Virieux, J. (2009). Seismic imaging of complex onshore structures by 2D elastic frequency-domain full-waveform inversion, *Geophysics* 74(6): WCC63–WCC76.

Brossier, R., Operto, S. & Virieux, J. (2010). Which data residual norm for robust elastic frequency-domain full waveform inversion?, *Geophysics* 75(3): R37–R46.

Brossier, R., Virieux, J. & Operto, S. (2008). Parsimonious finite-volume frequency-domain method for 2-D P-SV-wave modelling, *Geophysical Journal International* 175(2): 541–559.

Cagniard, L. (1962). *Reflection and refraction of progressive seismic waves*, McGraw-Hill (translated from Cagniard 1932).

Červený, V. (2001). *Seismic Ray Theory*, Cambridge University Press, Cambridge.

Chaljub, E., Capdeville, Y. & Vilotte, J.-P. (2003). Solving elastodynamics in a fluid-solid heterogeneous sphere: a parallel spectral element approximation on non-conforming grids, *Journal of Computational Physics* 187: 457–491.

Chaljub, E., Komatitsch, D., Vilotte, J.-P., Capdeville, Y., Valette, B. & Festa, G. (2007). Spectral element analysis in seismology, *in* R.-S. Wu & V. Maupin (eds), *Advances in Wave Propagation in Heterogeneous Earth*, Vol. 48 of *Advances in Geophysics*, Elsevier - Academic Press, London, pp. 365–419.

Chapman, C. (2004). *Fundamentals of seismic waves propagation*, Cambridge University Press, Cambridge, England.

Chew, W. C. & Liu, Q. H. (1996). Perfectly matched layers for elastodynamics: a new absorbing boundary condition, *Journal of Computational Acoustics* 4: 341–359.

Chew, W. C. & Weedon, W. H. (1994). A 3-D perfectly matched medium from modified Maxwell's equations with stretched coordinates, *Microwave and Optical Technology Letters* 7: 599–604.

Chin-Joe-Kong, M. J. S., Mulder, W. A. & Van Veldhuizen, M. (1999). Higher-order triangular and tetrahedral finite elements with mass lumping for solving the wave equation, *Journal of Engineering Mathematics* 35: 405–426.

Clayton, R. & Engquist, B. (1977). Absorbing boundary conditions for acoustic and elastic wave equations, *Bulletin of the Seismological Society of America* 67: 1529–1540.

Cockburn, B., Li, F. & Shu, C. W. (2004). Locally divergence-free discontinuous Galerkin methods for the Maxwell equations, *Journal of Computational Physics* 194: 588–610.

Cognon, J. H. (1971). Electromagnetic and electrical modelling by finite element methods, *Geophysics* 36: 132–155.

Collino, F. & Monk, P. (1998). Optimizing the perfectly matched layer, *Computer methods in Applied Mechanics and Engineering* 164: 157–171.

Collino, F. & Tsogka, C. (2001). Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media, *Geophysics* 66: 294–307.

Coutant, O., Doré, F., Nicollin, F., Beauducel, F. & Virieux, J. (2010). Seismic tomography of the Soufrière de Guadeloupe upper geothermal system, *Geophysical Research Abstracts*, Vol. 12, EGU.

Cruz-Atienza, V. (2006). *Rupture dynamique des faille non-planaires en différences finies*, PhD thesis, Université de Nice-Sophia Antipolis.

Cruz-Atienza, V., Virieux, J., Khors-Sansorny, C., Sardou, O., Gaffet, S. & Vallée, M. (2007). Estimation quantitative du PGA sur la Côte d'Azur, *Ecole Centrale Paris 1*, Association Française du Génie Parasismique (AFPS), p. 8.

Dablain, M. (1986). The application of high order differencing for the scalar wave equation, *Geophysics* 51: 54–66.

de Hoop, A. (1960). A modification of Cagniard's method for solving seismic pulse problems, *Applied Scientific Research* 8: 349–356.

Delcourte, S., Fezoui, L. & Glinsky-Olivier, N. (2009). A high-order discontinuous Galerkin method for the seismic wave propagation, *ESAIM: Proc.* 27: 70–89.
URL: *http://dx.doi.org/10.1051/proc/2009020*

Drossaert, F. H. & Giannopoulos, A. (2007). A nonsplit complex frequency-shifted PML based on recursive integration for FDTD modeling of elastic waves, *Geophysics* 72(2): T9–T17.

Druskin, V. & Knizhnerman, L. (1988). A spectral semi-discrete method for the numerical solution of 3-d non-stationary problems, *electrical prospecting: Izv. Acad. Sci. USSR, Physics of Solid Earth (Russian, translated into English)* 8: 63–74.

Duff, I. S., Erisman, A. M. & Reid, J. K. (1986). *Direct methods for sparse matrices*, Clarendon Press, Oxford, U. K.

Dumbser, M. & Käser, M. (2006). An Arbitrary High Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes II: The Three-Dimensional Isotropic Case, *Geophysical Journal International* 167(1): 319–336.

Dumbser, M., Käser, M. & Toro, E. (2007). An Arbitrary High Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes V: Local Time Stepping and p-Adaptivity, *Geophysical Journal International* 171(2): 695–717.

Erlangga, Y. A. & Herrmann, F. J. (2008). An iterative multilevel method for computing wavefields in frequency-domain seismic inversion, *Expanded Abstracts*, Soc. Expl. Geophys., pp. 1956–1960.

Etienne, V., Chaljub, E., Virieux, J. & Glinsky, N. (2010). An hp-adaptive discontinuous Galerkin finite-element method for 3D elastic wave modelling, *Geophysical Journal International* 183(2): 941–962.

Etienne, V., Virieux, J. & Operto, S. (2009). A massively parallel time domain discontinuous Galerkin method for 3D elastic wave modeling, *Expanded Abstracts, 79th Annual SEG Conference & Exhibition, Houston*, Society of Exploration Geophysics.

Festa, G. & Nielsen, S. (2003). PML absorbing boundaries, *Bulletin of Seismological Society of America* 93: 891–903.

Festa, G. & Vilotte, J.-P. (2005). The newmark scheme as a velocity-stress time stag- gering: An efficient PML for spectral element simulations of elastodynamics, *Geophysical Journal International* 161(3): 789–812.

Frey, P. & George, P. (2008). *Mesh Generation*, ISTE Ltd & John Wiley Sons Inc, London (UK) & Hoboken (USA).

Futterman, W. (1962). Dispersive body waves, *Journal of Geophysics Research* 67: 5279–5291.

Galis, M., Moczo, P. & Kristek, J. (2008). A 3-D hybrid finite-difference - finite-element viscoelastic modelling of seismic wave motion, *Geophysical Journal International* 175: 153–184.

Goldman, M. & Stover, C. (1983). Finite-difference calculations of the transient field of an axially symmetric earth for vertical magnetic dipole excitation, *Geophysics* 48: 953–963.

Gottschamer, E. & Olsen, K. B. (2001). Accuracy of the explicit planar free-surface boundary condition implemented in a fouth-order staggered-grid velocity-stress finite-difference scheme, *Bulletin of the Seismological Society of America* 91: 617–623.

Graves, R. (1996). Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences, *Bull. Seismol. Soc. Am.* 86: 1091–1106.

Günther, T., Rücker, C. & Spitzer, K. (2006). Three-dimensional modelling and inversion of dc resistivity data incorporating topography â€" II: Inversion, *Geophysical Journal International* 166: 506–517.

Hairer, E., Lubich, C. & Wanner, G. (2002). *Geometrical numerical integration: structure-preserving algorithms for ordinary differential equations*, Springer-Verlag.

Hammer, P. & Stroud, A. (1958). Numerical evaluation of multiple integrals, *Mathematical Tables Other Aids Computation* 12: 272–280.

Hayashi, K., Burns, D. R. & Toks´ oz, M. (2001). Discontinuous-grid finite-difference seismic modeling including surface topography, *Bulletin of the Seismological Society of America* 96(6): 1750–1764.

Hesthaven, J. & Warburton, T. (2008). *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, Springler.

Hestholm, S. (1999). 3-D finite-difference visocelastic wave modeling including surface topography, *Geophysical Journal International* pp. 852–878.

Hestholm, S. & Ruud, B. (1998). 3-D finite-difference elastic wave modeling including surface topography, *Geophysics* pp. 613–622.

Hicks, G. J. (2002). Arbitrary source and receiver positioning in finite-difference schemes using kaiser windowed sinc functions, *Geophysics* 67: 156–166.

Hohmann, G. W. (1988). Numerical modeling for electromagnetic methods of geophysics, *in* M. N. Nabighian (ed.), *Electromagnetic methods in applied geophysics*, Investigations in Geophysics, Society of Exploration Geophysics, pp. 313–363.

Hustedt, B., Operto, S. & Virieux, J. (2004). Mixed-grid and staggered-grid finite difference methods for frequency domain acoustic wave modelling, *Geophysical Journal International* 157: 1269–1296.

Ichimura, T., Hori, M. & Kuwamoto, H. (2007). Earthquake motion simulation with multi-scale finite element analysis on hybrid grid, *Bulletin of the Seismological Society of America* 97(4): 1133–1143.

Jacobs, G. & Hesthaven, J. S. (2006). High-order nodal discontinuous Galerkin particle-in-cell methods on unstructured grids, *Journal of Computational Physics* 214: 96–121.

Jo, C. H., Shin, C. & Suh, J. H. (1996). An optimal 9-point, finite-difference, frequency-space 2D scalar extrapolator, *Geophysics* 61: 529–537.

Karypis, G. & Kumar, V. (1998). *METIS - A software package for partitioning unstructured graphs, partitioning meshes and computing fill-reducing orderings of sparse matrices - Version 4.0*, University of Minnesota.

Käser, M. & Dumbser, M. (2008). A highly accurate discontinuous Galerkin method for complex interfaces between solids and moving fluids, *Geophysics* 73(3): 23–35.

Käser, M., Hermann, V. & de la Puente, J. (2008). Quantitative accuracy analysis of the discontinuous Galerkin method for seismic wave propagation, *Geophysical Journal International* 173(2): 990–999.

Kelly, K., Ward, R., Treitel, S. & Alford, R. (1976). Synthetic seismograms - a finite-difference approach, *Geophysics* 41: 2–27.

Kolsky, H. (1956). The propagation of stress pulses in viscoelastic solids, *Philosophical Magazine* 1: 693–710.

Komatitsch, D., Labarta, J. & Michéa, D. (2008). A simulation of seismic wave propagation at high resolution in the inner core of the Earth on 2166 processors of MareNostrum, *Lecture Notes in Computer Science* 5336: 364–377.

Komatitsch, D. & Martin, R. (2007). An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation, *Geophysics* 72(5): SM155–SM167.

Komatitsch, D. & Vilotte, J. P. (1998). The spectral element method: an efficient tool to simulate the seismic response of 2D and 3D geological structures, *Bulletin of the Seismological Society of America* 88: 368–392.

Kommedal, J. H., Barkved, O. I. & Howe, D. J. (2004). Initial experience operating a permanent 4C seabed array for reservoir monitoring at Valhall, *SEG Technical Program Expanded Abstracts* 23(1): 2239–2242.
URL: *http://link.aip.org/link/?SGA/23/2239/1*

Kosloff, D. & Baysal, E. (1982). Forward modeling by a Fourier method, *Geophysics* 47: 1402–1412.

Kosloff, D., Kessler, D., Filho, A., Tessmer, E., Behle, A. & Strahilevitz, R. (1990). Solution of the equations of dynamic elasticity by a Chebychev spectral method, *Geophysics* 55: 464–473.

Kummer, B. & Behle, A. (1982). Second-order finite-difference modelling of SH-wave propagation in laterally inhomogeneous media, *Bulletin of the Seismological Society of America* 72: 793–808.

Kuo, J. & Cho, D.-H. (1980). Transient time-domain electromagnetics, *Geophysics* 45: 271–291.

Kuzuoglu, M. & Mittra, R. (1996). Frequency dependence of the constitutive parameters of causal perfectly matched anisotropic absorbers, *IEEE Microwave and Guided Wave Letters* 6: 447–449.

Levander, A. R. (1988). Fourth-order finite-difference P-SV seismograms, *Geophysics* 53(11): 1425–1436.

LeVeque, R. J. (1986). Intermediate boundary conditions for time-split methods applied to hyperbolic partial differential equations, *Mathematical Computations* 47: 37–54.

Lombard, B. & Piraux, J. (2004). Numerical treatment of two-dimensional interfaces for acoustic and elastic waves, *Journal of Computional Physics* 195: 90–116.

Lombard, B., Piraux, J., Gelis, C. & Virieux, J. (2008). Free and smooth boundaries in 2-D finite-difference schemes for transient elastic waves, *Geophysical Journal International* 172: 252–261.

Luo, Y. & Schuster, G. T. (1990). Parsimonious staggered grid finite-differencing of the wave equation, *Geophysical Research Letters* 17(2): 155–158.

Madariaga, R. (1976). Dynamics of an expanding circular fault, *Bulletin of Seismological Society of America* 66: 639–666.

Marcinkovich, C. & Olsen, K. (2003). On the implementation of perfectly matched layers in a three-dimensional fourth-order velocity-stress finite difference scheme, *Journal of Geophysical Research* 108: doi:10.1029/2002GB002235.

Marfurt, K. (1984). Accuracy of finite-difference and finite-elements modeling of the scalar and elastic wave equation, *Geophysics* 49: 533–549.

Mercerat, E. D., Vilotte, J. P. & Sanchez-Sesma, F. J. (2006). Triangular spectral element simulation of two-dimensional elastic wave propagation using unstructured triangular grids, *Geophysical Journal International* 166: 679–698.

Meza-Fajardo, K. & Papageorgiou, A. (2008). A nonconvolutional, split-field, perfectly matched layer for wave propagation in isotropic and anisotropic elastic media: Stability analysis, *Bulletin of the Seismological Society of America* 98(4): 1811–1836.

Mikhailenko, B., Mikhailov, A. & Reshetova, G. (2003). Numerical viscoelastic modelling by the spectral laguerre method, *Geophysical Prospecting* 51: 37–48.

Moczo, P., Kristek, J., Galis, M., Pazak, P. & Balazovjech, M. (2007). The finite-difference and finite-element modeling of seismic wave propagation and earthquake motion, *Acta Physica Slovaca* 52(2): 177–406.

Mufti, I. R. (1985). Seismic modeling in the implicit mode, *Geophysical Prospecting* 33: 619–656.

Munns, J. W. (1985). The Valhall field: a geological overview, *Marine and Petroleum Geology* 2: 23–43.

Nihei, K. T. & Li, X. (2007). Frequency response modelling of seismic waves using finite difference time domain with phase sensitive detection (TD-PSD), *Geophysical Journal International* 169: 1069–1078.

Operto, S., Virieux, J., Amestoy, P., L'Éxcellent, J.-Y., Giraud, L. & Ben Hadj Ali, H. (2007). 3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study, *Geophysics* 72(5): SM195–SM211.

Opršal, I., Matyska, C. & Irikura, K. (2009). The source-box wave propagation hybrid methods: general formulation and implementation, *Geophysical Journal International* 176: 555–564.

Oristaglio, M. & Hohmann, G. W. (1984). Diffusion of electromagnetic fields into a two-dimensional earth: A finite-difference approach, *Geophysics* 49: 870–894.

Pasquetti, R. & Rapetti, F. (2006). Spectral element methods on unstructured meshes: Comparisons and recent advances, *Journal of Scientific Computing* 27: 377–387.

Pitarka, A. (1999). 3D elastic finite-difference modeling of seismic motion using staggered grids with nonuniform spacing, *Bulletin of the Seismological Society of America* 89(1): 54–68.

Plessix, R. E. (2007). A Helmholtz iterative solver for 3D seismic-imaging problems, *Geophysics* 72(5): SM185–SM194.

Plessix, R. E. (2009). Three-dimensional frequency-domain full-waveform inversion with an iterative solver, *Geophysics* 74(6): WCC53–WCC61.

Reed, W. & Hill, T. (1973). Triangular mesh methods for the neuron transport equation, *Technical Report LA-UR-73-479*, Los Alamos Scientific Laboratory.

Remaki, M. (2000). A new finite volume scheme for solving Maxwell's system, *COMPEL* 19(3): 913–931.

Riyanti, C. D., Erlangga, Y. A., Plessix, R. E., Mulder, W. A., Vuik, C. & Oosterlee, C. (2006). A new iterative solver for the time-harmonic wave equation, *Geophysics* 71(E): 57–63.

Riyanti, C. D., Kononov, A., Erlangga, Y. A., Vuik, C., Oosterlee, C., Plessix, R. E. & Mulder, W. A. (2007). A parallel multigrid-based preconditioner for the 3D heterogeneous high-frequency Helmholtz equation, *Journal of Computational physics* 224: 431–448.

Robertsson, J. O. A. (1996). A numerical free-surface condition for elastic/viscoelastic finite-difference modeling in the presence of topography, *Geophysics* 61: 1921–1934.

Roden, J. A. & Gedney, S. D. (2000). Convolution PML (CPML): An efficient FDTD implementation of the CFS-PML for arbitrary media, *Microwave and Optical Technology Letters* 27(5): 334–339.

Rollet, N., Deverchère, J., Beslier, M., Guennoc, P., Réhault, J., Sosson, M. & Truffert, C. (2002). Back arc extension, tectonic inheritance and volcanism in the ligurian sea, western mediterranean, *Tectonics* 21(3).

Rücker, C., Günther, T. & Spitzer, K. (2006). Three-dimensional modeling and inversion of DC resistivity data incorporating topography - Part I: Modeling, *Geophysical Journal International* 166: 495–505.

Saad, Y. (2003). *Iterative methods for sparse linear systems*, SIAM, Philadelphia.

Saenger, E. H., Gold, N. & Shapiro, S. A. (2000). Modeling the propagation of elastic waves using a modified finite-difference grid, *Wave motion* 31: 77–92.

Sears, T., Singh, S. & Barton, P. (2008). Elastic full waveform inversion of multi-component OBC seismic data, *Geophysical Prospecting* 56(6): 843–862.

Seriani, G. & Priolo, E. (1994). Spectral element method for acoustic wave simulation in heterogeneous media, *Finite elements in analysis and design* 16: 337–348.

Si, H. (2006). *TetGen - A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator - Version 1.4*, University of Berlin.

Sirgue, L., Etgen, J. T. & Albertin, U. (2008). 3D Frequency Domain Waveform Inversion using Time Domain Finite Difference Methods, *Proceedings 70th EAGE, Conference and Exhibition, Roma, Italy*, p. F022.

Sirgue, L. & Pratt, R. G. (2004). Efficient waveform inversion and imaging : a strategy for selecting temporal frequencies, *Geophysics* 69(1): 231–248.

Sourbier, F., Haiddar, A., Giraud, L., Ali, H. B. H., Operto, S. & Virieux, J. (2011). Three-dimensional parallel frequency-domain visco-acoustic wave modelling based on a hybrid direct-iterative solver, *Geophysical Prospecting* Special issue: Modelling methods for geophysical imaging, 59(5), 835-856.

Stekl, I. & Pratt, R. G. (1998). Accurate viscoelastic modeling by frequency-domain finite difference using rotated operators, *Geophysics* 63: 1779–1794.

Symes, W. W. (2007). Reverse time migration with optimal checkpointing, *Geophysics* 72(5): SM213–SM221.

Taflove, A. & Hagness, C. (2000). *Computational electrodynamics: the finite-difference time-domaine method*, Artech House, London, United Kindom.

Tal-Ezer, H., Carcione, J. & Kosloff, D. (1990). An accurate and efficient scheme for wave propagation in linear viscoelastic media, *Geophysics* 55: 1366–1379.

Tam, C. K. & Webb, J. C. (1993). Dispersion-relation-preserving finite difference schemes for computational acoustics, *Journal of Computational Physics* 107: 262–281.

Tessmer, E., Kosloff, D. & Behle, A. (1992). Elastic wave propagation simulation in the presence of surface topography, *Geophysical Journal International* 108: 621–632.

Titarev, V. & Toro, E. (2002). Ader: arbitrary high order godunov approach, *SIAM Journal Scientific Computing* 17: 609–618.

Toulopoulos, I. & Ekaterinaris, J. A. (2006). High-order discontinuous Galerkin discretizations for computational aeroacoustics in complex domains, *AIAA J.* 44: 502–511.

Vigh, D. & Starr, E. W. (2008). Comparisons for Waveform Inversion, Time domain or Frequency domain?, *Extended Abstracts*, pp. 1890–1894.

Virieux, J. (1986). P-SV wave propagation in heterogeneous media, velocity-stress finite difference method, *Geophysics* 51: 889–901.

Virieux, J. & Lambaré, G. (2007). Theory and observations - body waves: ray methods and finite frequency effects, *in* B. Romanovitz & A. Diewonski (eds), *Treatise of Geophysics, volume 1: Seismology and structure of the Earth*, Elsevier.

Virieux, J., Operto, S., Ben Hadj Ali, H., Brossier, R., Etienne, V., Sourbier, F., Giraud, L. & Haidar, A. (2009). Seismic wave modeling for seismic imaging, *The Leading Edge* 28(5): 538–544.

Warner, M., Stekl, I. & Umpleby, A. (2008). 3D wavefield tomography: synthetic and field data examples, *SEG Technical Program Expanded Abstracts* 27(1): 3330–3334.

Wheeler, L. & Sternberg, E. (1968). Some theorems in classical elastodynamics, *Archive for Rational Mechanics Analysis* 31: 51–90.

Williamson, P. R. & Worthington, M. H. (1993). Resolution limits in ray tomography due to wave behavior: Numerical experiments, *Geophysics* 58: 727–735.

Yee, K. S. (1966). Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, *IEEE Trans. Antennas and Propagation* 14: 302–307.

Zienkiewicz, O. C., Taylor, R. L. & Zhu, J. Z. (2005). *The Finite Element Method: Its Basis and Fundamentals*, Elsevier, London. 6th edition.

Zienkiewicz, O. & Taylor, R. (1967). *The Finite Element Method for Solid and Structural Mechanics*, McGraw Hill, New York.

ARCHITECTURE AND PERFORMANCE OF DEVITO, A SYSTEM FOR AUTOMATED STENCIL COMPUTATION *

FABIO LUPORINI†, MICHAEL LANGE‡, MATHIAS LOUBOUTIN§, NAVJOT KUKREJA†, JAN HÜCKELHEIM†, CHARLES YOUNT¶, PHILIPP WITTE‖, PAUL H. J. KELLY#, GERARD J. GORMAN†, AND FELIX J. HERRMANN§

Abstract. Stencil computations are a key part of many high-performance computing applications, such as image processing, convolutional neural networks, and finite-difference solvers for partial differential equations. Devito is a framework capable of generating highly-optimized code given symbolic equations expressed in *Python*, specialized in, but not limited to, affine (stencil) codes. The lowering process – from mathematical equations down to C++ code – is performed by the Devito compiler through a series of intermediate representations. Several performance optimizations are introduced, including advanced common sub-expressions elimination, tiling and parallelization. Some of these are obtained through well-established stencil optimizers, integrated in the back-end of the Devito compiler. The architecture of the Devito compiler, as well as the performance optimizations that are applied when generating code, are presented. The effectiveness of such performance optimizations is demonstrated using operators drawn from seismic imaging applications.

Key words. Stencil, finite difference method, symbolic processing, structured grid, compiler, performance optimization

AMS subject classifications. 65N06, 68N20

1. Introduction.

Developing software for high-performance computing requires a considerable interdisciplinary effort, as it often involves domain knowledge from numerous fields such as physics, numerical analysis, software engineering and low-level performance optimization. The result is typically a monolithic application where hardware-specific optimizations, numerical methods, and physical approximations are interwoven and dispersed throughout a large number of loops, functions, files and modules. This frequently leads to slow innovation, high maintenance costs, and code that is hard to debug and port onto new computer architectures. A powerful approach to alleviate this problem is to introduce a separation of concerns and to raise the level of abstraction by using domain-specific languages (DSLs). DSLs can be used to express numerical methods using a syntax that closely mirrors how they are expressed mathematically, while a stack of compilers and libraries is responsible for automatically

*Submitted to SIAM Journal on Scientific Computing on July 9, 2018.

Funding: This work was supported by the Engineering and Physical Sciences Research Council through grants EP/I00677X/1, EP/L000407/1, EP/I012036/1], by the Imperial College London Department of Computing, by the Imperial College London Intel Parallel Computing Centre (IPCC), and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics and Computer Science programs under contract number DE-AC02-06CH11357.

†Department of Earth Science and Engineering, Imperial College London, London, UK, (f.luporini12@imperial.ac.uk, n.kukreja@imperial.ac.uk, j.hueckelheim@imperial.ac.uk, g.gorman@imperial.ac.uk)

‡European Centre for Medium-Range Weather Forecasts, Reading, UK, (michael.lange@ecmwf.int)

§Georgia Institute of Technology, School of Computational Science and Engineering, Atlanta GA, USA, (mlouboutin3@gatech.edu, felix.herrmann@gatech.edu)

¶Intel Corporation, (chuck.yount@intel.com)

‖Seismic Laboratory for Imaging and Modeling (SLIM), The University of British Columbia, Vancouver BC, CANADA, (pwitte.slim@gmail.com)

#Department of Computing, Imperial College London, London, SW7 2AZ, UK, (p.kelly@imperial.ac.uk)

creating the optimized low-level implementation in a general purpose programming language such as C++. While the focus of this paper is on finite-difference (FD) based codes, the DSL approach has already had remarkable success in other numerical methods such as the finite-element (FE) and finite-volume (FV) method, as documented in Section 2.

This work describes the architecture of *Devito*, a system for automated stencil computations from a high-level mathematical syntax. Devito was developed with an emphasis on FD methods on structured grids. For this reason, Devito's underlying DSL has many features to simplify the specification of FD methods, as discussed in Section 3. The original motivation was to solve large-scale partial differential equations (PDEs) in the context of seismic inverse problems, where FD solvers are commonly used for solving wave equations as part of complex workflows (e.g., data inversion using adjoint-state methods and backpropogation). Devito is equally useful as a framework for other stencil computations in general; for example, computations where all array indices are affine functions of loop variables. The Devito compiler is also capable of generating arbitrarily nested, possibly irregular, loops. This key feature is needed to support many complex algorithms that are used in engineering and scientific practice, including applications from image processing, cellular automata, and machine-learning.

One of the design goals of Devito was to enable high-productivity, so it is fully written in *Python*, with easy access to solvers, optimizers, input and output, and the wide range of other libraries in the *Python* ecosystem. At the same time, Devito transforms high-level symbolic input into optimized C++ code, resulting in a performance that is competitive with hand-optimized implementations. While the examples presented in this paper focus on using Devito from a *Python* application, exploiting the full potential of on-the-fly code generation and just-in-time (JIT) compilation, a practical advantage of generating C++ as an intermediate step is that it can be also used to generate libraries for legacy software, thus enabling incremental code modernisation.

Compared to other DSL frameworks that are used in practice, Devito uses compiler technology, including several layers of intermediate representations, to perform optimizations in multiple passes. This allows Devito to perform more complex optimizations, and to better optimize the code for individual target platforms. The fact that these optimisations are performed programmatically facilitates performance portability across different computer architectures [28]. This is important, as industrial codes are often used on a variety of platforms, including clusters with multi-core CPUs, GPUs, and many-core chips spread across several compute nodes as well as various cloud platforms. Devito also performs high-level transformations for floating-point operation (FLOP) reduction based on symbolic manipulation, as well as loop-level optimizations as implemented in Devito's own optimizer, or using a third-party stencil compiler such as YASK [40]. The Devito compiler is presented in detail in Sections 4, 5 and 6.

After the presentation of the Devito compiler, we show test cases in Section 7 that are inspired by real-world seismic-imaging problems. The paper finishes with directions for future work and conclusions in Sections 8 and 9.

2. Related work. The objective of maximizing productivity and performance through frameworks based upon DSLs has long been pursued. In addition to well-known systems such as Mathematica® and Matlab®, which span broad mathematical areas, there are a number of tools specialized in numerical methods for PDEs, some

dating back to the 1970s [6, 34, 7, 35].

2.1. DSL-based frameworks for partial differential equations. One noteworthy contemporary framework centered on DSLs is FEniCS [22], which allows the specification of weak variational forms, via UFL [2], and finite-element methods, through a high-level syntax. Firedrake [30] implements the same languages as FEniCS, although it differs from it in a number of features and architectural choices. Devito is heavily influenced by these two successful projects, in particular by their philosophy and design. Since solving a PDE is often a small step of a larger workflow, the choice of *Python* to implement these software provides access to a wide ecosystem of scientific packages. Firedrake also follows the principle of graceful degradation, by providing a very simple lower-level API to escape the abstraction when non-standard calculations (i.e., unrelated to the finite-element formulation) are required. Likewise, Devito allows injecting arbitrary expressions into the finite-difference specification; this feature has been used in real-life cases, for example for interpolation in seismic imaging operators. On the other hand, a major difference is that Devito lacks a formal specification language such us UFL in FEniCS/Firedrake. This is partly because there is no systematic foundation underpinning FD, as opposed to FE which relies upon the theory of Hilbert spaces [5]. Yet another distinction is that, for performance reasons, Devito takes control of the time-stepping loop. Other examples of embedded DSLs are provided by the OpenFOAM project, with a language for FV [13], and by PyFR, which targets flux reconstruction methods [36].

2.2. High-level approaches to finite differences. Due to its simplicity, the FD method has been the subject of multiple research projects, chiefly targeting the design of effective software abstraction and/or the generation of high performance code [14, 3, 16, 21]. Devito distinguishes itself from previous work in a number of ways including: support for the principle of graceful degradation for when the DSL does not cover a feature required by an application; incorporation of a symbolic mathematics engine; using actual compiler technology rather than template-based code generation; adoption of a native *Python* interface that naturally allows composition into complex workflows such as optimisation and machine-learning frameworks.

At a lower level of abstraction there are a number of tools targeting "stencil" computation (FD codes belong to this class), whose major objective is the generation of efficient code. Some of them provide a DSL [40, 31, 43, 29], whereas others are compilers or user-driven code generation systems, often based upon a polyhedral model, such as [4, 18]. From the Devito standpoint, the aim is to harness these tools – for example by integrating them, to maximize performance portability. As a proof of concept, we shall discuss the integration of one such tool, namely YASK [40], with Devito.

2.3. Devito and seismic imaging. Devito is a general purpose system, not restricted to specific PDEs, so it can be used for any form of the wave equation. Thus, unlike software specialized in seismic exploration, like IWAVE [32] and Madagascar [12], it suffers neither from the restriction to a small set of wave equations and discretizations, nor from the lack of portability and composability typical of a pure C/Fortran environment.

2.4. Performance optimizations. The Devito compiler can introduce three types of performance optimizations: FLOPs reduction, data locality, and parallelism. Typical FLOPs reduction transformations are common sub-expressions elimination, factorization, and code motion. A thorough review is provided in [11]. To different

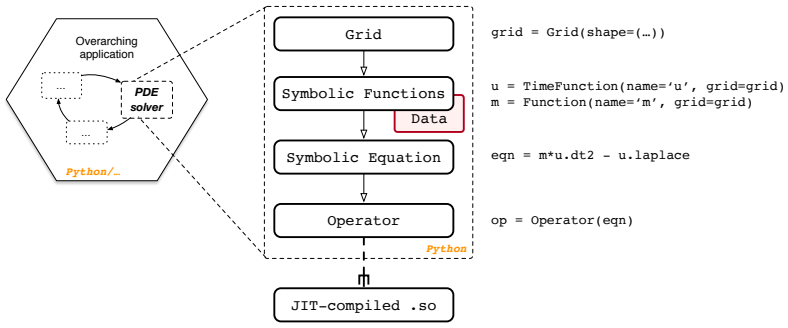


Fig. 1: The typical usage of Devito within a larger application.

extent, Devito applies all of these techniques (see Section 5.1). Particularly relevant for stencil computation is the search for redundancies across consecutive loop iterations [9, 10, 20]. This is at the core of the strategy described in Section 6, which essentially extends these ideas with optimizations for data locality. Typical loop transformations for parallelism and data locality [17] are also automatically introduced by the Devito compiler (e.g., loop blocking, vectorization); more details will be provided in Sections 5.2 and 5.3.

3. Specification of a finite-difference method with Devito. The Devito DSL allows concise expression of FD and general stencil operations using a mathematical notation. It uses *SymPy* [27] for the specification and manipulation of stencil expressions. In this section, we describe the use of Devito's DSL to build PDE solvers. Although the examples used here are for FD, the DSL can describe a large class of operations, such as convolutions or basic linear algebra operations (e.g., chained tensor multiplications).

3.1. Symbolic types. The key steps to implement a numerical kernel with Devito are shown in Figure 1. We describe this workflow, as well as fundamental features of the Devito API, using the acoustic wave equation, also known as d'Alembertian or Box operator. Its continuous form is given by:

$$\begin{aligned} m(x,y,z)\frac{d^2u(x,y,z,t)}{dt^2} - \nabla^2 u(x,y,z,t) &= q_s, \\ u(x,y,z,0) &= 0, \\ \frac{du(x,y,z,t)}{dt}|_{t=0} &= 0, \end{aligned} \tag{3.1}$$

where the variables of this expression are defined as follows:

- $m(x,y,z) = \frac{1}{c(x,y,z)^2}$, is the parametrization of the subsurface with $c(x,y,z)$ being the speed of sound as a function of the three space coordinates (x,y,z);
- $u(x,y,z,t)$, is the spatially varying acoustic wavefield, with the additional dimension of time t;
- q_s is the source term, which is a point source in this case.

The first step towards solving this equation is the definition of a discrete computational grid, on which the model parameters, wavefields and source are defined. The computational grid is defined as a `Grid(shape)` object, where `shape` is the number

of grid points in each spatial dimension. Optional arguments for instantiating a `Grid` are `extent`, which defines the extent in physical units, and `origin`, the origin of the coordinate system, with respect to which all other coordinates are defined.

The next step is the symbolic definition of the squared slowness, wavefield and source. For this, we introduce some fundamental types.

- `Function` represents a discrete spatially varying function, such as the velocity. A `Function` is instantiated for a defined `name` and a given `Grid`.
- `TimeFunction` represents a discrete function that is both spatially varying and time dependent, such as wavefields. Again, a `TimeFunction` object is defined on an existing `Grid` and is identified by its `name`.
- `SparseFunction` and `SparseTimeFunction` represent sparse functions, that is functions that are only defined over a subset of the grid, such as a seismic point source. The corresponding object is defined on a `Grid`, identified by a `name`, and also requires the `coordinates` defining the location of the sparse points.

Apart from the grid information, these objects carry their respective FD discretization information in space and time. They also have a `data` field that contains values of the respective function at the defined grid points. By default, `data` is initialized with zeros and therefore automatically satisfies the initial conditions from equation 3.1. The initialization of the fields to solve the wave equation over a one-dimensional grid is displayed in Listing 1.

Listing 1 Setup `Functions` to express and solve the acoustic wave equation.

```
>>> from devito import Grid, TimeFunction, Function, SparseTimeFunction
>>> g = Grid(shape=(nx,), origin=(ox,), extent=(sx,))
>>> u = TimeFunction(name="u", grid=g, space_order=2, time_order=2)  # Wavefield
>>> m = Function(name="m",  grid=g)  # Physical parameter
>>> q = SparseTimeFunction(name="q", grid=g, coordinates=coordinates)  # Source
```

3.2. Discretization. With symbolic objects that represent the discrete velocity model, wavefields and source function, we can now define the full discretized wave equation. As mentioned earlier, one of the main features of Devito is the possibility to formulate stencil computations as concise mathematical expressions. To do so, we provide shortcuts to classic FD stencils, as well as the functions to define arbitrary stencils. The shortcuts are accessed as object properties and are supported by `TimeFunction` and `Function` objects. For example, we can take spatial and temporal derivatives of the wavefield `u` via the shorthand expressions `u.dx` and `u.dt` (Listing 2).

Listing 2 Example of spatial and temporal FD stencil creation.

```
>>> u.dx
-u(t, x - h_x)/(2*h_x) + u(t, x + h_x)/(2*h_x)
>>> u.dt
-u(t - dt, x)/(2*dt) + u(t + dt, x)/(2*dt)
>>> u.dt2
-2*u(t, x)/dt**2 + u(t - dt, x)/dt**2 + u(t + dt, x)/dt**2
```

Furthermore, Devito provides shortcuts for common differential operations such as the Laplacian via `u.laplace`. The full discrete wave equation can then be implemented in a single line of *Python* (Listing 3).

To solve the time-dependent wave equation with an explicit time-stepping scheme, the symbolic expression representing our PDE has to be rearranged such that it yields an update rule for the wavefield u at the next time step: $u(t + dt) = f(u(t), u(t -$

Listing 3 Expressing the wave equation.

```
>>> wave_equation = m * u.dt2 - u.laplace
>>> wave_equation
(-2*u(t, x)/dt**2 + u(t - dt, x)/dt**2 + u(t + dt, x)/dt**2)*m(x) + 2*u(t, x)/h_x
    **2 - u(t, x - h_x)/h_x**2 - u(t, x + h_x)/h_x**2
```

dt))). Devito allows to rearrange the PDE expression automatically using the `solve` function, as shown in Listing 4.

Listing 4 Time-stepping scheme for the acoustic wave equation. `region=INTERIOR` ensures that the Dirichlet boundary conditions at the edges of the Grid are satisfied.

```
>>> from devito import Eq, INTERIOR, solve
>>> stencil = Eq(u.forward, solve(wave_equation, u.forward), region=INTERIOR)
>>> stencil
Eq(u(t + dt, x), -2*dt**2*u(t, x)/(h_x**2*m(x)) + dt**2*u(t, x - h_x)/(h_x**2*m(x)
    ) + dt**2*u(t, x + h_x)/(h_x**2*m(x)) + 2*u(t, x) - u(t - dt, x))
```

Note that the `stencil` expression in Listing 4 does not yet contain the point source q. This could be included as a regular `Function` which has zeros all over the grid except for a few points; this, however, would obviously be wasteful. Instead, `SparseFunction`s allow to perform operations, such as injecting a source or sampling the wavefield, at a subset of grid points determined by `coordinates`. In the case in which coordinates do not coincide with grid points, bilinear (for 2D) or trilinear (for 3D) interpolation are employed. To inject a point source into the `stencil` expression, we use the `inject` function of the `SparseTimeFunction` object that represents our seismic source (Listing 5).[1]

Listing 5 Expressing the injection of a source into a field.

```
>>> injection = q.inject(field=u.forward, expr=dt**2 * q / m)
>>> injection
[Eq(u[t + 1, INT(floor((-o_x + q_coords[p_q, 0])/h_x))], dt**2*(1 - FLOAT(-h_x*INT
    (floor((-o_x + q_coords[p_q, 0])/h_x)) - o_x + q_coords[p_q, 0])/h_x)*q[time,
    p_q]/m[INT(floor((-o_x + q_coords[p_q, 0])/h_x))] + u[t + 1, INT(floor((-o_x +
     q_coords[p_q, 0])/h_x))]),
 Eq(u[t + 1, INT(floor((-o_x + q_coords[p_q, 0])/h_x)) + 1], dt**2*FLOAT(-h_x*INT(
    floor((-o_x + q_coords[p_q, 0])/h_x)) - o_x + q_coords[p_q, 0])*q[time, p_q
    ]/(h_x*m[INT(floor((-o_x + q_coords[p_q, 0])/h_x)) + 1]) + u[t + 1, INT(floor
    ((-o_x + q_coords[p_q, 0])/h_x)) + 1])]
```

The `inject` function takes the field being updated as an input argument (in this case `u.forward`), while `expr=dt**2 * q / m` is the expression being injected. The result of the `inject` function is a list of symbolic expressions, similar to the `stencil` expression we defined earlier. As we shall see, these expressions are eventually joined together and used to create an `Operator` object – the solver of our PDE.

3.3. Boundary conditions. Simple boundary conditions (BCs), such as Dirichlet BCs, can be imposed on individual equations through special keywords (see Listing 4). For more exotic schemes, instead, the BCs need to be explicitly written (e.g., Higdon BCs [15]), just like any of the symbolic expressions defined in the Listings

[1] More complicated interpolation schemes can be defined by precomputing the grid points corresponding to each sparse point, and their respective coefficients. The result can then be used to create a `PrecomputedSparseFunction`, which behaves like a `SparseFunction` at the symbolic level.

above. For reasons of space, this aspect is not elaborated further; the interested reader may refer to [26].

3.4. Control flow. By default, the extent of a `TimeFunction` in the time dimension is limited by its time order. Hence, the shape of u in Listing 1 is $(time\_order + 1, nx) = (3, nx)$. The iterative method will then access u via modulo iteration, that is $u[t\%3, ...]$. In many scenarios, however, the entire time history, or at least periodic time slices, should be saved (e.g., for inversion algorithms). Listing 6 expands our running example with an equation that saves the content of u every 4 iterations, up to a maximum of $save = 100$ time slices.

Listing 6 Implementation of time sub-sampling.

```
>>> from devito import ConditionalDimension
>>> ts = ConditionalDimension('ts', parent=g.time_dim, factor=4)
>>> us = TimeFunction(name='us', grid=g, save=100, time_dim=ts)
>>> save = Eq(us, u)
```

In general, all equations that access `Function`s (or `TimeFunction`s) employing one or more `ConditionalDimension`s will be conditionally executed. The condition may be a number indicating how many iterations should pass between two executions of the same equation, or even an arbitrarily complex expression.

3.5. Domain, halo, and padding regions. A `Function` internally distinguishes between three regions of points.

Domain Represents the *computational domain* of the `Function` and is inferred from the input `Grid`. This includes any elements added to the *physical domain* purely for computational purposes, e.g. absorbing boundary layers.

Halo The grid points surrounding the domain region, i.e. "ghost" points that are accessed by the stencil when iterating in proximity of the domain boundary.

Padding The grid points surrounding the halo region, which are allocated for performance optimizations, such as data alignment. Normally this region should be of no interest to a user of Devito, except for precise measurement of memory allocated for each `Function`.

4. The Devito compiler. In Devito, an `Operator` carries out three fundamental tasks: generation of low-level code, JIT compilation, and execution. The `Operator` input consists of one or more symbolic equations. In the generated code, these equations are scheduled within loop nests of suitable depth and extent. The `Operator` also accepts substitution rules (to replace symbols with constant values) and optimization levels for the Devito Symbolic Engine (DSE) and the Devito Loop Engine (DLE). By default, all DSE and DLE optimizations that are known to unconditionally improve performance are automatically applied. The same `Operator` may be reused with different input data; JIT-compilation occurs only once, triggered by the first execution. Overall, this lowering process – from high-level equations to dynamically compiled and executable code – consists of multiple compiler passes, summarized in Figure 2 and discussed in the following sections (a minimal background in data dependence analysis is recommended; the unfamiliar reader may refer to a classic textbook such as [1]).

4.1. Equations lowering. In this pass, three main tasks are carried out: *indexification*, *substitution*, and *domain-alignment*.

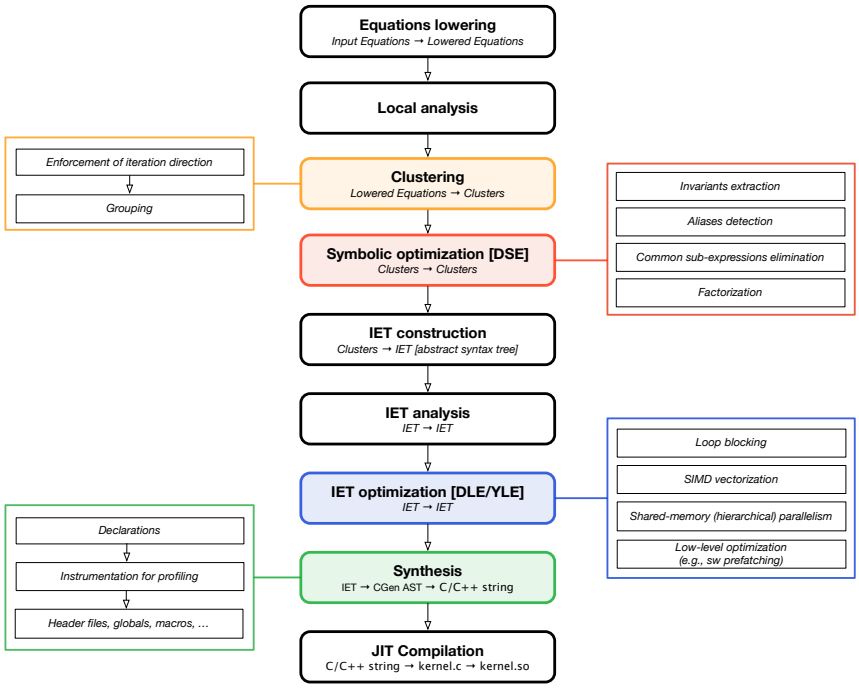


Fig. 2: Compiler passes to lower symbolic equations into shared objects through an `Operator`.

- As explained in Section 3, the input equations typically involve ore or more indexed `Function`s. The *indexification* consists of converting such objects into actual arrays. An array always keeps a reference to its originating `Function`. For instance, all accesses to u such as $u[t, x+1]$ and $u[t+1, x-2]$ would store a pointer to the same, user-defined `Function` $u(t, x)$. This metadata is exploited throughout the various compilation passes.
- During *substitution*, the user-provided substitution rules are applied. These may be given for any literal appearing in the input equations, such as the grid spacing symbols. Applying a substitution rule increases the chances of constant folding, but it makes the `Operator` less generic. The values of symbols for which no substitution rule is available are provided at execution time.
- The *domain-alignment* step shifts the array accesses deriving from `Function`s having non-empty halo and padding regions. Thus, the array accesses become logically aligned to the equation's natural domain. For instance, given the usual `Function` $u(t, x)$ having two points on each side of the x halo region, the array accesses $u[t, x]$ and $u[t, x+2]$ are transformed, respectively, into $u[t, x+2]$ and $u[t, x+4]$. When $x = 0$, therefore, the values $u[t, 2]$ and $u[t, 4]$ are fetched, representing the first and third points in the computational domain.

4.2. Local analysis. The lowered equations are analyzed to collect information relevant for the `Operator` construction and execution. In this pass, an equation is inspected "in isolation", ignoring its relationship with the rest of the input. The following metadata are retrieved and/or computed:

- input and output `Function`s;
- `Dimension`s, which are topologically ordered based on how they appear in the various array index functions; and
- two notable `Space`s: the iteration space, `ISpace`, and the data space, `DSpace`.

A `Space` is a collection of points given by the product of n compact intervals on $\mathbb{Z}$. With the notation $d[o_m, o_M]$ we indicate the compact interval $[d_m + o_m, d_M + o_M]$ over the `Dimension` d, in which d_m and d_M are parameters (specialized only at runtime), while o_m and o_M are known integers. For instance, $[x[0,0], y[-1,1]]$ describes a rectangular two-dimensional space over x and y, whose points are given by the Cartesian product $[x_m, x_M] \times [y_m - 1, y_M + 1]$. The `ISpace` and `DSpace` are two special types of `Space`. They usually span different sets of `Dimension`s. A `DSpace` may have `Dimension`s that do not appear in an `ISpace`, in particular those that are accessed only via integer indices. Likewise, an `ISpace` may have `Dimension`s that are not part of the `DSpace`, such as a reduction axis. Further, an `ISpace` also carries, for each `Dimension`, its iteration direction.

As an example, consider the equation *stencil* in Listing 4. Immediately we see that input $= [u, m]$, output $= [u]$, `Dimension`s $= [t, x]$. The compiler constructs the `ISpace` $[t[0,0]^+, x[0,0]^*]$. The first entry $t[0,0]^+$ indicates that, along t, the equation should run between $t_m + 0$ and $t_M + 0$ (extremes included) in the *forward* direction, as indicated by the symbol $+$. This is due to the fact that there is a flow dependence in t, so only a unitary positive stepping increment (i.e., $t = t + 1$) allows a correct propagation of information across consecutive iterations. The only difference along x is that the iteration direction is now arbitrary, as indicated by $*$. The `DSpace` is $[t[0,1], x[0,0]]$; intuitively, the entry $t[0,1]$ is used right before running an `Operator` to provide a default value for t_M – in particular, t_M will be set to the largest possible value that does not cause out-of-domain accesses (i.e., out-of-bounds array accesses).

4.3. Clustering. A `Cluster` is a sequence of equations having (i) same `ISpace`, (ii) same control flow (i.e., same `ConditionalDimension`s), and (iii) no dimension-carried "true" anti-dependences among them.

As an example, consider again the setup in Section 3. The equation *stencil* cannot be "clusterized" with the equations in the *injection* list as their `ISpace`s are different. On the other hand, the equations in *injection* can be grouped together in the same `Cluster` as (i) they have same `ISpace` $[t[0,0]^*, p_q[0,0]^*]$, (ii) same control flow, and (iii) there are no true anti-dependences among them (note that the second equation in *injection* does write to $u[t+1, ...]$, but as explained later this is in fact a reduction, that is a "false" anti-dependence).

4.3.1. Iteration direction. First, each equation is assigned a new `ISpace`, based upon a *global* analysis. Any of the iteration directions that had been marked as "arbitrary" ($*$) during local analysis may now be enforced to *forward* ($+$) or *backward* ($-$). This process exploits data dependence analysis.

For instance, consider the flow dependence between *stencil* and the *injection* equations. If we want u to be up-to-date when evaluating *injection*, then we eventually need all equations to be scheduled sequentially within the t loop. For this, the `ISpace`s of the *injection* equations are specialized by enforcing the direction *forward* along the `Dimension` t. The new `ISpace` is $[t[0,0]^+, p_q[0,0]^*]$.

Algorithm 1 illustrates how the enforcement of iteration directions is achieved in general. Whenever a clash is detected (i.e., two equations with `ISpace` $[d[0,0]^+, ...]$ and $[d[0,0]^-, ...]$), the original direction determined by the local analysis pass is kept (lines 11 and 13), which will eventually lead to generating different loops.

Algorithm 1: Clustering: enforcement of iteration directions (pseudocode).

Input: A sequence of equations $\mathcal{E}$.
Output: A sequence of equations $\mathcal{E}'$ with altered `ISpace`.

```
// Map each dimension to a set of expected iteration directions
mapper ← DETECT_FLOW_DIRECTIONS(ℰ);
for e in ℰ do
    for dim, directions in mapper do
        if len(directions) == 1 then
            // No ambiguity
            forced[dim] ← directions.pop();
        else if len(directions) == 2 then
            // No ambiguity as long as one of the two items is /Any/
            try
                directions.remove(Any);
                forced[dim] ← directions.pop();
            except
                forced[dim] ← e.directions[dim];
        else
            forced[dim] ← e.directions[dim];
        end if
    end for
    ℰ'.append(e._rebuild(directions=forced))
end for
return ℰ'
```

4.3.2. Grouping. This step performs the actual clustering, checking `ISpace`s and anti-dependences, as well as handling control flow. The procedure is shown in Algorithm 2; some explanations follow.

Algorithm 2: Clustering: grouping expressions into `Cluster`s (pseudocode)

Input: A sequence of equations $\mathcal{E}$.
Output: A sequence of clusters $\mathcal{C}$.

```
𝒞 ← ClusterGroup();
for e in ℰ do
    grouped ← false;
    for c in reversed(𝒞) do
        anti, flow ← GET_DEPENDENCES(c, e);
        if e.ispace == c.ispace and anti.carried is empty then
            c.add(e);
            grouped ← true;
            break;
        else if anti.carried is not empty then
            c.atomics.update(anti.carried.cause);
            break;
        else if flow.cause.intersection(c.atomics) then
            // cannot search across earlier clusters
            break;
    end for
    if not grouped then
        𝒞.append(Cluster(e));
    end if
end for
𝒞 ← CONTROL_FLOW(𝒞);
return 𝒞
```

- Robust data dependence analysis, capable of tracking flow-, anti-, and output-

dependencies at the level of array accesses, is necessary. In particular, it must be able to tell whether two generic *array accesses* induce a dependence or not. The data dependence analysis performed is conservative; that is, a dependence is always assumed when a test is inconclusive. Dependence testing is based on the standard Lamport test [1]. In Algorithm 2, data dependence analysis is carried out by the function GET_DEPENDENCES.

- If an anti-dependence is detected along a `Dimension` i, then i is marked as *atomic* – meaning that no further clustering can occur along i. This information is also exploited by later `Operator` passes (see Section 4.5).
- Reductions, and in particular increments, are treated specially. They represent a special form of anti-dependence, as they do not break clustering. GET_DEPENDENCES detects reductions and removes them from the set of anti-dependencies.
- Given the sequence of equations $[E_1, E_2, E_3]$, it is possible that E_3 can be grouped with E_1, but not with its immediate predecessor E_2 (e.g., due to a different `ISpace`). However, this can only happen when there are no flow or anti-dependences between E_2 and E_3; i.e. when the `if` commands at lines 10 and 13 are not entered, thus allowing the search to proceed with the next equation. This optimization was originally motivated by gradient operators in seismic imaging kernels.
- The routine CONTROL_FLOW, omitted for brevity, creates additional `Cluster`s if one or more `ConditionalDimension`s are encountered. These are tracked in a special `Cluster` field, *guards*, as also required by later passes (see Section 4.5).

4.4. Symbolic optimization. The DSE – Devito Symbolic Engine – is a macro-pass reducing the *arithmetic strength* of `Cluster`s (e.g., their operation count). It consists of a series of passes, ranging from standard common sub-expression elimination (CSE) to more advanced rewrite procedures, applied individually to each `Cluster`. The DSE output is a new ordered sequence of `Cluster`s: there may be more or fewer `Cluster`s than in the input, and both the overall number of equations as well as the sequence of arithmetic operations might differ. The DSE passes are discussed in Section 5.1. We remark that the DSE only operates on `Cluster`s (i.e., on collections of equations); there is no concept of "loop" at this stage yet. However, by altering `Cluster`s, the DSE has an indirect impact on the final loop-nest structure.

4.5. IET construction. In this pass, the intermediate representation is lowered to an Iteration/Expression Tree (IET). An IET is an abstract syntax tree in which `Iteration`s and `Expression`s – two special node types – are the main actors. Equations are wrapped within `Expression`s, while `Iteration`s represent loops. Loop nests embedding such `Expression`s are constructed by suitably nesting `Iteration`s. Each `Cluster` is eventually placed in its own loop (`Iteration`) nest, although some (outer) loops may be shared by multiple `Cluster`s.

Consider again our running acoustic wave equation example. There are three `Cluster`s in total: C_1 for *stencil*, C_2 for *save*, and C_3 for the equations in *injection*. We use Algorithm 3 – an excerpt of the actual cluster scheduling algorithm – to explain how this sequence of `Cluster`s is turned into an IET. Initially, the *schedule* list is empty, so when C_1 is handled two nested `Iteration`s are created (line 15), respectively for the `Dimension`s t and x. Subsequently, C_2's `ISpace` and the current *schedule* are compared (line 5). It turns out that t appears among C_2's guards, hence the for loop is exited at line 12 without inspecting the second and last iteration. Thus, $index = 1$, and the previously built `Iteration` over t is reused. Finally, when processing C_3, the for loop is exited at the second iteration due to line 6, since $p_q! = x$. Again, the t `Iteration` is reused, while a new `Iteration` is constructed for

Algorithm 3: An excerpt of the cluster scheduling algorithm, turning a list (of `Clusters`) into a tree (IET). Here, the fact that different `Clusters` may eventually share some outer `Iterations` is highlighted.

Input: A sequence of `Clusters` $\mathcal{C}$.
Output: An Iteration/Expression Tree.

```
schedule ← list();
for c in C do
    root ← None;
    index ← 0;
    for i0, i1 in zip(c.ispace, schedule) do
        if i0 != i1 or i0.dimension in c.atomics then
            break;
        end if
        root ← schedule[i1];
        index ← index + 1;
        if i0.dim in c.guards then
            break;
        end if
    end for
    ⟨build as many Iterations as Dimensions in c.ispace[index:] and nest them inside root⟩;
    ⟨update schedule⟩;
    ⟨...⟩
end for
```

the `Dimension` p_q. Eventually, the constructed IET is as in Listing 7.

Listing 7 Graphical representation of the IET produced by the cluster scheduling algorithm for the running example.

```
for t = t_m to t_M:
 |-- for x = x_m to x_M:
 |     |-- <Eq(u[t+1,x], ...)>
 |
 |-- if t % 4 == 0
 |     |-- for x = x_m to x_M:
 |           |-- <Eq(us[t/4, x], ...)>
 |
 |-- for p_q = p_q_m to p_q_M:
       |-- <Eq(u[t+1,f(p_q)], ...)>
       |-- <Eq(u[t+1,g(p_q)], ...)>
```

4.6. IET analysis. The newly constructed IET is analyzed to determine `Iteration` properties such as `sequential`, `parallel`, and `vectorizable`, which are then attached to the relevant nodes in the IET. These properties are used for loop optimization, but only by a later pass (see Section 4.7). To determine whether an `Iteration` is `parallel` or `sequential`, a fundamental result from compiler theory is used – the i-th `Iteration` in a nest comprising n `Iterations` is parallel if for all dependences D, expressed as distance vectors $D = (d_0, ..., d_{n-1})$, either $(d_1, ..., d_{i-1}) > 0$ or $(d_1, ..., d_i) = 0$ [1].

4.7. IET optimization. This macro-pass transforms the IET for performance optimization. Apart from runtime performance, this pass also optimizes for rapid JIT compilation with the underlying C compiler. A number of loop optimizations are introduced, including loop blocking, minimization of remainder loops, SIMD vectorization, shared-memory (hierarchical) parallelism via OpenMP, software prefetching. These will be detailed in Section 5. A *backend* (see Section 4.9) might provide its own

loop optimization engine.

4.8. Synthesis, dynamic compilation, and execution. Finally, the IET adds variable declarations and header files, as well as instrumentation for performance profiling, in particular, to collect execution times of specific code regions. Declarations are injected into the IET, ensuring they appear as close as possible to the scope in which the relative variables are used, while honoring the OpenMP semantics of private and shared variables. To generate C code, a suitable tree visitor inspects the IET and incrementally builds a *CGen* tree [19], which is ultimately translated into a string and written to a file. Such files are stored in a software cache of Devito-generated `Operator`s, JIT-compiled into a shared object, and eventually loaded into the *Python* environment. The compiled code has a default entry point (a special function), which is called directly from *Python* at `Operator` application time.

4.9. Operator specialization through backends. In Devito, a *backend* is a mechanism to specialize data types as well as `Operator` passes, while preserving software modularity (inspired by [25]).

One of the main objectives of the backend infrastructure is promoting software composability. As explained in Section 2, there exist a significant number of interesting tools for stencil optimization, which we may want to integrate with Devito. For example, one of the future goals is to support GPUs, and this might be achieved by writing a new backend implementing the interface between Devito and third-party software specialized for this particular architecture.

Currently, two backends exist:

`core` the default backend, which relies on the DLE for loop optimization.

`yask` an alternative backend using the YASK stencil compiler to generate optimized C++ code for Intel® Xeon® and Intel® Xeon Phi™ architectures [40]. Devito transforms the IET into a format suitable for YASK, and uses its API for data management, JIT-compilation, and execution. Loop optimization is performed by YASK through the YASK Loop Engine (YLE).

The *core* and *yask* backends share the compilation pipeline in Figure 2 until the loop optimization stage.

5. Automated performance optimizations. As discussed in Section 4, Devito performs symbolic optimizations to reduce the arithmetic strength of the expressions, as well as loop transformations for data locality and parallelism. The former are implemented as a series of compiler passes in the DSE, while for the latter there currently are two alternatives, namely the DLE and the YLE (depending on the chosen execution backend).

Devito abstracts away the single optimizations passes by providing users with a certain number of optimization levels, called "modes", which trigger pre-established sequences of optimizations – analogous to what general-purpose compilers do with, for example, `-O2` and `-O3`. In Sections 5.1, 5.2, and 5.3 we describe the individual passes provided by the DSE, DLE, and YLE respectively, while in Section 7.1 we explain how these are composed into modes.

5.1. DSE - Devito Symbolic Engine. The DSE passes attempt to reduce the arithmetic strength of the expressions through FLOP-reducing transformations [11]. They are illustrated in Listings 8-11, which derive from the running example used throughout the article. A detailed description follows.

- **Common sub-expression elimination (CSE).** Two implementations are available: one based upon *SymPy*'s `cse` routine and one built on top of more basic

SymPy routines, such as `xreplace`. The former is more powerful, being aware of key arithmetic properties such as associativity; hence it can discover more redundancies. The latter is simpler, but avoids a few critical issues: (i) it has a much quicker turnaround time; (ii) it does not capture integer index expressions (for increased quality of the generated code); and (iii) it tries not to break factorization opportunities. A generalized common sub-expressions elimination routine retaining the features and avoiding the drawbacks of both implementations is still under development. By default, the latter implementation is used when the CSE pass is selected.

Listing 8 An example of common sub-expressions elimination.

```
>>> 9.0*dt*dt*u[t, x + 1] - 18.0*dt*dt*u[t][x + 2] + 9.0*dt*dt*u[t, x + 3]
temp0 = dt*dt
9.0*temp0*u[t, x + 1] - 18.0*temp0*u[t][x + 2] + 9.0*temp0*u[t, x + 3]
```

- **Factorization.** This pass visits each expression tree and tries to factorize FD weights. Factorization is applied without altering the expression structure (e.g., without expanding products) and without performing any heuristic search across groups of expressions. This choice is based on the observation that a more aggressive approach is only rarely helpful (never in the test cases in Section 7), while the increase in symbolic processing time could otherwise be significant. The implementation exploits the *SymPy* `collect` routine. However, while `collect` only searches for common factors across the immediate children of a single node, the DSE implementation recursively applies `collect` to each `Add` node (i.e., an addition) in the expression tree, until the leaves are reached.

Listing 9 An example of FD weights factorization.

```
>>> 9.0*temp0*u[t, x + 1] - 18.0*temp0*u[t][x + 2] + 9.0*temp0*u[t, x + 3]
9.0*temp0*(u[t, x + 1] + u[t, x + 3]) - 18.0*temp0*u[t][x + 2]
```

- **Extraction.** The name stems from the fact that sub-expressions matching a certain condition are pulled out of a larger expression, and their values are stored into suitable scalar or tensor temporaries. For example, a condition could be "*extract all time-varying sub-expressions whose operation count is larger than a given threshold*". A tensor temporary may be preferred over a scalar temporary if the intention is to let the *IET construction* pass (see Section 4.5) place the pulled sub-expressions within an outer loop nest. Obviously, this comes at the price of additional storage. This peculiar effect – trading operations for memory – will be thoroughly analyzed in Sections 6 and 7.

Listing 10 An example of time-varying sub-expressions extraction. Only sub-expressions performing at least one floating-point operation are extracted.

```
>>> 9.0*temp0*(u[t, x + 1] + u[t, x + 3]) - 18.0*temp0*u[t][x + 2]
temp1[x] = u[t, x + 1] + u[t, x + 3]
9.0*temp0*temp1[x] - 18.0*temp0*u[t][x + 2]
```

- **Detection of aliases.** The Alias-Detection Algorithm implements the most advanced DSE pass. In essence, an alias is a sub-expression that is redundantly computed at multiple iteration points. Because of its key role in the Cross-Iteration

Redundancy-Elimination algorithm, the formalization of the Alias-Detection Algorithm is postponed until Section 6.

Listing 11 An example of alias detection.

```
>>> 9.0*temp0*u[t, x + 1] - 18.0*temp0*u[t][x + 2] + 9.0*temp0*u[t, x + 3]
temp[x] = 9.0*temp0*u[t, x]
temp[x + 1] - 18.0*temp0*u[t][x + 2] + temp[x + 3]
```

5.2. DLE - Devito Loop Engine. The DLE transforms the IET via classic loop optimizations for parallelism and data locality [17]. These are summarized below.

- **SIMD Vectorization.** Implemented by enforcing compiler auto-vectorization via special `pragma`s from the OpenMP 4.0 language. With this approach, the DLE aims to be performance-portable across different architectures. However, this strategy causes a significant fraction of vector loads/stores to be unaligned to cache boundaries, due to the stencil offsets. As we shall see, this is a primary cause of performance loss.
- **Loop Blocking.** Also known as "tiling", this technique implemented by replacing `Iteration` trees in the IET. The current implementation only supports blocking over fully-parallel `Iteration`s. Blocking over dimensions characterized by flow- or anti-dependences, such as the time dimension in typical explicit finite difference schemes, is instead work in progress (this would require a preliminary pass known as loop skewing; see Section 8 for more details). On the other hand, a feature of the present implementation is the capability of blocking across particular *sequences* of loop nests. This is exploited by the Cross-Iteration Redundancy-Elimination algorithm, as shown in Section 6.3. To determine an optimal block shape, an `Operator` resorts to empirical auto-tuning.
- **Parallelism.** Shared-memory parallelism is introduced by decorating `Iteration`s with suitable OpenMP `pragma`s. The OpenMP `static` scheduling is used. Normally, only the outermost fully-parallel `Iteration` is annotated with the parallel `pragma`. However, heuristically nested fully-parallel `Iteration`s are `collapsed` if the core count is greater than a certain threshold. This pass also ensures that all array temporaries allocated in the scope of the parallel `Iteration` are declared as `private` and that storage is allocated where appropriate (stack, heap).

Summarizing, the DLE applies a sequence of typical stencil optimizations, aiming to reach a minimum level of performance across different architectures. As we shall see, the effectiveness of this approach, based on simple transformations, deteriorates on architectures strongly conceived for hierarchical parallelism. This is one of the main reasons behind the development of the `yask` backend (see Section 4.9), described in the following section.

5.3. YLE - YASK Loop Engine. "YASK" (Yet Another Stencil Kernel) is an open-source C++ software framework for generating high-performance implementations of stencil codes for Intel® Xeon® and Intel® Xeon Phi™ processors. Previous publications on YASK have discussed its overall structure [40] and its application to the Intel® Xeon Phi™ x100 family (code-named Knights Corner) [37] and Intel® Xeon Phi™ x200 family (code-named Knights Landing) [38, 33] many-core CPUs. Unlike Devito, it does not expose a symbolic language to the programmer or create stencils from finite-difference approximations of differential equations. Rather, the programmer provides simple declarative descriptions of the stencil equations using a C++ or Python API. Thus, Devito operates at a level of abstraction higher than

that of YASK, while YASK provides performance portability across Intel architectures and is more focused on low-level optimizations. Following is a sample of some of the optimizations provided by YASK:[2]

- **Vector-folding.** In traditional SIMD vectorization, such as that provided by a vectorizing compiler, the vector elements are arranged sequentially along the unit-stride dimension of the grid, which must also be the dimension iterated over in the inner-most loop of the stencil application. Vector-folding is an alternative data-layout method whereby neighboring elements are arranged in small *multi-dimensional* tiles. Figure 3 illustrates three ways to pack eight double-precision floating-point values into a 512-bit SIMD register. Figure 3a shows a traditional 1D "in-line" layout, and 3b and 3c show alternative 2D and 3D "folded" layouts. Furthermore, these tiles may be ordered in memory in a dimension independent of the dimensions used in vectorization [37]. The combination of these two techniques can significantly increase overlap and reuse between successive stencil-application iterations, reducing the memory-bandwidth demand. For stencils that are bandwidth-bound, this can provide significant performance gains [37, 33].
- **Software prefetching.** Many high-order or staggered-grid stencils require many streams of data to be read from memory, which can overwhelm the hardware prefetchers. YASK can be directed to automatically generate software prefetch instructions to improve the cache hit rates, especially on Xeon Phi CPUs.
- **Hierarchical parallelism.** Dividing the spatial domain into tiles to increase temporal cache locality is a common stencil optimization as discussed earlier. When implementing this technique, sometimes called "cache-blocking", it is typical to assign each thread to one or more small rectilinear subsets of the domain in which to apply the stencil(s). However, if these threads share caches, one thread's data will often evict data needed later by another thread, reducing the effective capacity of the cache. YASK addresses this by employing two levels of OpenMP parallelization: the outer level of parallel loops are applied across the cache-blocks, and an inner level is applied across sub-blocks within those tiles. In the case of the Xeon Phi, the eight hyper-threads that share each L2 cache can now cooperate on filling and reusing the data in the cache, rather than evicting each other's data.

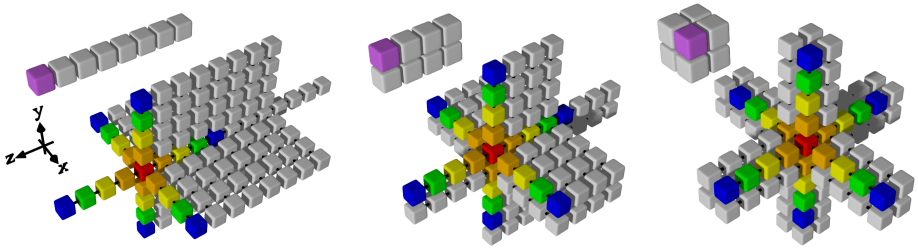
YASK also provides other optimizations, such as temporal wave-fron tiling, as well as MPI support. These features, however, are not exploited by Devito yet. The interested reader may refer to [38, 39].

To obtain the best of both tools, we have integrated the YASK framework into the Devito package. In essence, the Devito `yask` backend exploits the intermediate representation of an `Operator` to generate YASK kernels. This process is based upon sophisticated compiler technology. In *Devito v3.1*, roughly 70% of the Devito API is supported by the `yask` backend[3].

6. The Cross-Iteration Redundancy-Elimination Algorithm. Aliases, or "cross-iteration redundancies" (informally introduced in Section 5.1), in FD operators depend on the differential operators used in the PDE(s) and the chosen discretization scheme. From a performance viewpoint, the presence of aliases is a non-issue as long as the operator is memory-bound, while it becomes relevant in kernels with a high arithmetic intensity. In Devito, the Cross-Iteration Redundancy-Elimination (CIRE) algorithm attempts to remove aliases with the goal of reducing the operation count. As

[2] Not all YASK features are currently used by Devito.

[3] At the time of writing, reaching feature-completeness is one the major on-going development efforts



a. $1 \times 1 \times 8$ 1D fold b. $1 \times 2 \times 4$ 2D fold c. $2 \times 2 \times 2$ 3D fold

Fig. 3: Various folds of 8 elements [37]. The smaller diagram in the upper-left of each sub-figure illustrates a single SIMD layout, and the larger diagram shows the input values needed for a typical 25-point stencil, as from an 8th-order finite-difference approximation of an isotropic acoustic wave. Note that the $1 \times 1 \times 8$ 1D fold corresponds to the traditional in-line vectorization.

shown in Section 7, the CIRE algorithm has considerable impact in seismic imaging kernels. The algorithm is implemented through the orchestration of multiple DSE and DLE/YLE passes, namely *extraction of candidate expressions (DSE)*, *detection of aliases (DSE)*, *loop blocking (DLE/YLE)*.

6.1. Extraction of candidate expressions. The criteria for extraction of candidate sub-expressions are:

- Any *maximal time-invariant* whose operation count is greater than $Thr_0 = 10$ (floating point arithmetic only). The term "maximal" means that the expression is not embedded within a larger time-invariant. The default value $Thr_0 = 10$, determined empirically, provides systematic performance improvements in a series of seismic imaging kernels. Transcendental functions are given a weight in the order of tens of operations, again determined empirically.
- Any *maximal time-varying* whose operation count is greater than $Thr_1 = 10$. Such expressions often lead to aliases, since they typically result from taking spatial and time derivatives on `TimeFunction`s. In particular, cross-derivatives are a major cause of aliases.

This pass leverages the *extraction* routine described in Section 5.1.

6.2. Detection of aliases. To define the concept of aliasing expressions, we first need to formalize the notion of *translated operands*. Here, an operand is regarded as the arithmetic product of a scalar value (or "coefficient") and one or more indexed objects. An indexed object is characterized by a label (i.e., its name), a vector of n dimensions, and a vector of n displacements (one for each dimension). We say that an operand o_1 is translated with respect to an operand o_0 if o_0 and o_1 have same coefficient, label, and dimensions, and if their displacement vectors are such that one is the translation of the other (in the classic geometric sense). For example, the operand $2 * u[x, y, z]$ is translated with respect to the operand $2 * u[x + 1, y + 2, z + 3]$ since they have same coefficient (2), label (u), and dimensions ($[x, y, z]$), while the displacement vectors $[0, 0, 0]$ and $[1, 2, 3]$ are expressible by means of a translation.

Now consider two expressions e_0 and e_1 in fully-expanded form (i.e., a non-nested sum-of-operands). We say that e_0 is an alias of e_1 if the following conditions hold:

- the operands in e_0 (e_1) are expressible as a translation of the operands in e_1 (e_0);
- the same arithmetic operators are applied to the involved operands.

For example, consider $e = u[x] + v[x]$, having two operands $u[x]$ and $v[x]$; then:

- **u[x-1] + v[y-1]** is *not* an alias of e, due to a different dimension vector.
- **u[x] + w[x]** is *not* an alias of e, due to a different label.
- **u[x+2] + v[x]** is *not* an alias of e, since a translation cannot be determined.
- **u[x+2] + v[x+2]** is an alias of e, as the operands $u[x+2]$ and $v[x+2]$ can be expressed as a translation of $u[x]$ and $v[x]$ respectively, with $T(o_d) = o_d + \mathbf{2}$ and o_d representing the displacement vector of an operand.

The relation "e_0 is an alias of e_1" is an equivalence relation, as it is at the same time reflexive, symmetric, and transitive. Thanks to these properties, the turnaround times of the Alias-Detection Algorithm are extremely quick (less than 2 seconds running on an Intel® Xeon® E5-2620 v4 for the challenging `tti` test case with `so`=16, described in Section 7.2), despite the $O(n^2)$ computational complexity (with n representing the number of candidate expressions, see Section 6.1).

Algorithm 4 highlights the fundamental steps of the Alias-Detection Algorithm. In the worst case scenario, all pairs of candidate expressions are compared by applying the aliasing definition given above. Aggressive pruning, however, is applied to minimize the cost of the search. The algorithm uses some auxiliary functions: (i) CALCULATE_DISPLACEMENTS returns a mapper associating, to each candidate, its displacement vectors (one for each indexed object); (ii) COMPARE_OPS(e_1, e_2) evaluates to true if e_1 and e_2 perform the same operations on the same operands; (iii) IS_TRANSLATED(d_1, d_2) evaluates to true if the displacement vectors in d_2 are pairwise-translated with respect to the vectors in d_1 by the same factor. Together, (ii) and (iii) are used to establish whether two expressions alias each other (line 8).

Eventually, m sets of aliasing expressions are determined. For each of these sets $G_0, ..., G_{m-1}$, a *pivot* – a special aliasing expression – is constructed. This is the key for operation count reduction: the pivot p_i of $G_i = \{e_0, ..., e_{k-1}\}$ will be used in place of $e_0, ..., e_{k-1}$ (thus obtaining a reduction proportional to k). A simple example is illustrated in Listing 11.

Algorithm 4: The Alias-Detection Algorithm (pseudocode).

Input: A sequence of expressions $\mathcal{E}$.
Output: A sequence of Alias objects $\mathcal{A}$.

```
1  displacements ← CALCULATE_DISPLACEMENTS(E);
2  A ← list();
3  unseen ← list(E);
4  while unseen is not empty do
5  |   top ← unseen.pop();
6  |   G = Alias(top);
7  |   for e in unseen do
8  |   |   if COMPARE_OPS(top, e) and IS_TRANSLATED(displacements[top], displacements[e])
   |   |    then
9  |   |   |   G.append(e);
10 |   |   |   unseen.remove(e);
11 |   |   end if
12 |   end for
13 |   A.append(G)
14 end while
15 return A
```

Several optimizations for data locality, not shown in Algorithm 4, are also applied. The interested reader may refer to the documentation and the examples of *Devito v3.1* for more details; below, we only mention the underlying ideas.

- The pivot of G_i is *constructed*, rather than selected out of $e_0, ..., e_{k-1}$, so that it could coexist with as many other pivots as possible within the same `Cluster`. For example, consider again Listing 11: there are infinite possible pivots `temp[x + s] = 9.0*temp0*u[t, x + s]`, and the one with $s = 0$ is chosen. However, this choice is not random: the Alias-Detection Algorithm chooses pivots based on a global optimization strategy, which takes into account all of the m sets of aliasing expressions. The objective function consists of choosing s so that multiple pivots will have identical `ISpace`, and thus be scheduled to the same `Cluster` (and, eventually, to the same loop nest).
- Conservatively, the chosen pivots are assigned to array variables. A second optimization pass, called *index bumping and array contraction* in *Devito v3.1*, attempts to turn these arrays into scalar variables, thus reducing memory consumption. This pass is based on data dependence analysis, which essentially checks whether a given pivot is required only within its `Cluster` or by later `Cluster`s as well. In the former case, the optimization is applied.

6.3. Loop blocking for working-set minimization. In essence, the CIRE algorithm trades operation for memory – the (array) temporaries to store the aliases. From a run-time performance viewpoint, this is convenient only in arithmetic-intensive kernels. Unsurprisingly, we observed that storing temporary arrays spanning the entire grid rarely provides benefits (e.g., only when the operation count reductions are exceptionally high). We then considered the following options.

1. Capturing redundancies arising along the innermost dimension only. Thus, only scalar temporaries would be necessary. This approach presents three main issues, however: (i) only a small percentage of all redundancies are captured; (ii) the implementation is non-trivial, due to the need for circular buffers in the generated code; (iii) SIMD vectorization is affected, since inner loop iterations are practically serialised. Some previous articles followed this path [9, 10].
2. A generalization of the previous approach: using both scalar and array temporaries, without searching for redundancies across the outermost loop(s). This mitigates issue (i), although the memory pressure is still severely affected. Issue (iii) is also unsolved. This strategy was discussed in [20].
3. Using loop blocking. Redundancies are sought and captured along all available dimensions, although they are now assigned to array temporaries whose size is a function of the block shape. A first loop nest produces the array temporaries, while a subsequent loop nest consumes them, to compute the actual output values. The block shape should be chosen so that writes and reads to the temporary arrays do not cause high latency accesses to the DRAM. An illustrative example is shown in Listing 12.

The CIRE algorithm uses the third approach, based on cross-loop-nest blocking. This pass is carried out by the DLE, which can introduce blocking over sequences of loops (see Section 5.2).

7. Performance evaluation. We outline in Section 7.1 the compiler setup, computer architectures, and measurement procedure that we used for our performance experiments. Following that, we outline the physical model and numerical setup that define the problem being solved in Section 7.2. This leads to performance results, presented in Sections 7.3 and 7.4.

Listing 12 The loop nest produced by the CIRE algorithm for the example in Listing 11. Note that the block loop (line 2) wraps both the producer (line 3) and consumer (line 5) loops. For ease of read, unnecessary information are omitted.

```
1 for t = t_m to t_M:
2   for xb = x_m to x_M, xb += blocksize:
3     for x = xb to xb + blocksize + 3, x += 1
4       temp[x] = 9.0*temp0*u[t, x]
5     for x = xb to xb + blocksize; x += 1:
6       u[t+1,x,y] = ... + temp[x + 1] - 18.0*temp0*u[t][x + 2] + temp[x + 3] + ...
```

7.1. Compiler and system setup. We analyse the performance of generated code using enriched roofline plots. Since the DSE transformations may alter the operation count by allocating extra memory, only by looking at GFlops/s performance and runtime jointly can a quality measure of code syntheses be derived.

For the roofline plots, Stream TRIAD was used to determine the attainable memory bandwidth of the node. Two peaks for the maximum floating-point performance are shown: the ideal peak, calculated as

$$\#[\text{cores}] \cdot \#[\text{avx units}] \cdot \#[\text{vector lanes}] \cdot \#[\text{FMA ports}] \cdot [\text{ISA base frequency}]$$

and a more realistic one, given by the LINPACK benchmark. The reported runtimes are the minimum of three runs (the variance was negligible). The model used to calculate the operational intensity assumes that the time-invariant `Function`s are reloaded at each time iteration. This is a more realistic setting than a "compulsory-traffic-only" model (i.e., an infinite cache).

We had exclusive access to two architectures: an Intel® Xeon® Platinum 8180 (formerly code-named Skylake) and an Intel® Xeon Phi™ 7250 (formerly code-named Knights Landing), which will be referred to as `skl8180` and `knl7250`. Thread pinning was enabled with the program `numactl`. The Intel® compiler `icc version 18.0` was used to compile the generated code. The experiments were run with *Devito v3.1* [41]. The experimentation framework with instructions for reproducibility is available at [42]. All floating point operations are performed in single precision, which is typical for seismic imaging applications.

Any arbitrary sequence of DSE and DLE/YLE transformations is applicable to an `Operator`. Devito, provides three preset optimization sequences, or "modes", which vary in aggressiveness and affect code generation in three major ways:

- the time required by the Devito compiler to generate the code,
- the potential reduction in operation count, and
- the potential amount of additional memory that might be allocated to store (scalar, tensor) temporaries.

A more aggressive mode might obtain a better operation count reduction than a non-aggressive one, although this does not necessarily imply a better time to solution as the memory pressure might also increase. The three optimization modes – `basic`, `advanced`, and `aggressive`– apply the same sequence of DLE/YLE transformations, which includes OpenMP parallelism, SIMD vectorization, and loop blocking. However, they vary in the number, type, and order of DSE transformations. In particular,
`basic` enables common sub-expressions elimination only;
`advanced` enables `basic`, then factorization, extraction of time-invariant aliases;
`aggressive` enables `advanced`, then extraction of time-varying aliases.
Thus, `aggressive` triggers the full-fledged CIRE algorithm, while `advanced` uses only a relaxed version (based on time invariants). All runs used loop tiling with a

block shape that was determined individually for each case using auto-tuning. The auto-tuning phase, however, was not included in the measured experiment runtime. Likewise, the code generation phase is not included in the reported runtime.

7.2. Test case setup. In the following sections, we benchmark the performance of operators modeling the propagation of acoustic waves in two different models: isotropic and Tilted Transverse Isotropy (TTI, [44]), henceforth `isotropic` and `tti`, respectively. These operators were chosen for their relevance in seismic imaging techniques [44].

Acoustic `isotropic` modeling is the most commonly used technique for seismic inverse problems, due to the simplicity of its implementation, as well as the comparatively low computational cost in terms of FLOPs. The `tti` wave equation provides a more realistic simulation of wave propagation and accounts for local directional dependency of the wave speed, but comes with increased computational cost and mathematical complexity. For our numerical tests, we use the `tti` wave equation as defined in [44]. The full specification of the equation as well as the finite difference schemes and its implementation using Devito are provided in [24, 23]. Essentially, the `tti` wave equation consists of two coupled acoustic wave equations, in which the Laplacians are constructed from spatially rotated first derivative operators. As indicated by Figure 4, these spatially rotated Laplacians have a significantly larger number of stencil coefficients in comparison to its isotropic equivalent which comes with an increased operational intensity.

The `tti` and `isotropic` equations are discretized with second order in time and varying space orders of 4, 8, 12 and 16. For both test cases, we use zero initial conditions, Dirichlet boundary conditions and absorbing boundaries with a 10 point mask (Section 3.5). The waves are excited by injecting a time-dependent, but spatially-localized seismic source wavelet into the subsurface model, using Devito's sparse point interpolation and injection as described in Section 3.1. We carry out performance measurements for two velocity models of 512^3 and 768^3 grid points with a grid spacing of 20 m. Wave propagation is modeled for 1000 ms, resulting in 327 time steps for `isotropic`, and 415 time steps for `tti`. The time-stepping interval is chosen according to the Courant-Friedrichs-Lewy (CFL) condition [8], which guarantees stability of the explicit time-marching scheme and is determined by the highest velocity of the subsurface model and the grid spacing.

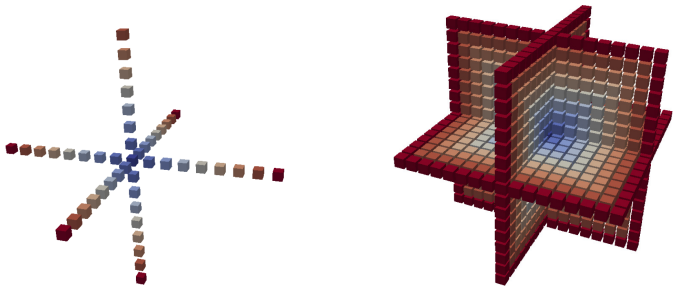


Fig. 4: Stencils of the acoustic Laplacian for the `isotropic` (left) and `tti` (right) wave equations and `so`=16. The anisotropic Laplacian corresponds to a spatially rotated version of the isotropic Laplacian. The color indicates the distance from the central coefficient.

7.3. Performance: acoustic wave in isotropic model. This section illustrates the performance of `isotropic` with the `core` and `yask` backends. To relieve the exposition, we show results for the DSE in `advanced` mode only; the `aggressive` has no impact on `isotropic`, due to the memory-bound nature of the code [23].

The performance of `core` on `skl8180`, illustrated in Figure 5a (`yask` uses slightly smaller grids than `core` due to a flaw in the API of *Devito v3.1*, which will be fixed in *Devito v3.2*), degrades as the space order (henceforth, `so`) increases. In particular, it drops from 59% of the attainable machine peak to 36% in the case of `so`=16. This is the result of multiple issues. As `so` increases, the number of streams of unaligned virtual addresses also increases, causing more pressure on the memory system. Intel® VTune™ revealed that the lack of split registers to efficiently handle split loads was a major source of performance degradation. Another major issue for `isotropic` on `core` concerns the quality of the generated SIMD code. The in-line vectorization performed by the auto-vectorizer produces a large number of pack/unpack instructions to move data between vector registers, which introduces substantial overhead. Intel® VTune™ also confirmed that, unsurprisingly, `isotropic` is a memory-bound kernel. Indeed, switching off the DSE basically did not impact the runtime, although it did increase the operational intensity of the four test cases.

The performance of `core` on `knl7250` is not as good as that on `skl8180`. Figure 5b shows an analogous trend to that on `skl8180`, with the attainable machine peak systematically dropping as `so` increases. The issue is that here the distance from the peak is even larger. This simply suggests that `core` is failing at exploiting the various levels of parallelism available on `knl7250`.

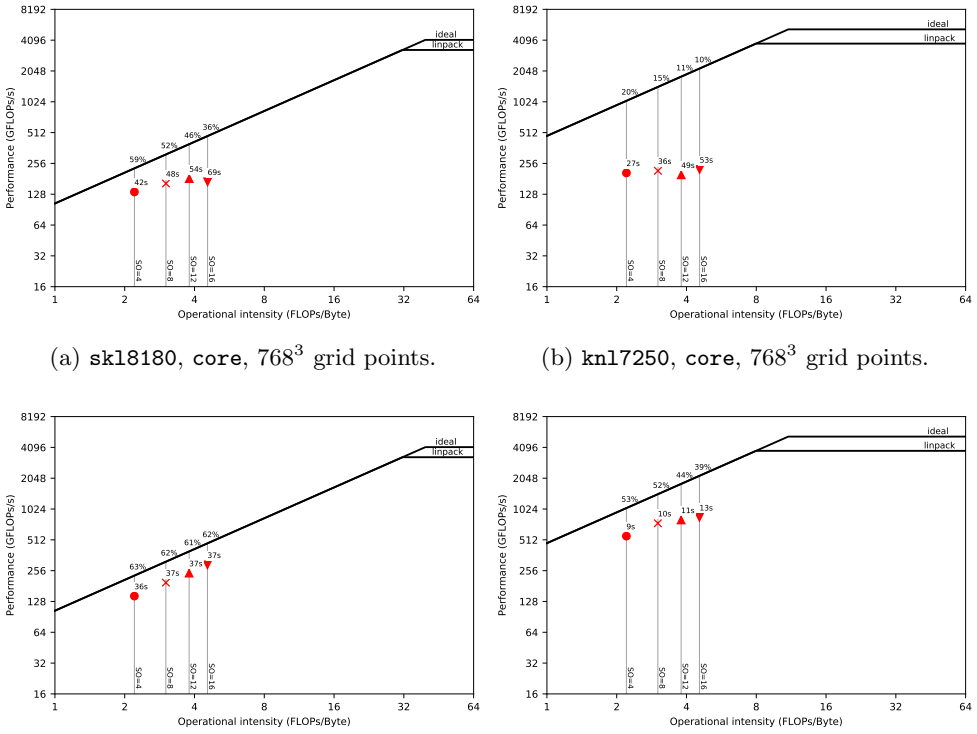
The `yask` backend overcomes all major limitations to which `core` is subjected. On both `skl8180` and `knl7250`, `yask` outperforms `core`, essentially since it does not suffer from the issues presented above. Vector folding minimizes unaligned accesses; software prefetching helps especially for larger values of `so`; hierarchical OpenMP parallelism is fundamental to leverage shared caches. The speed-up on `knl7250` is remarkable, since even in the best scenario for `core` (`so`=4), `yask` is roughly 3× faster, and more than 4× faster when `so`=12.

7.4. Performance: acoustic wave in tilted transverse isotropy model. This sections illustrates the performance of `tti` with the `core` backend. `tti` cannot be run on the `yask` backend in *Devito v3.1* as some fundamental features are still missing; this is part of our future work (more details in Section 8).

Unlike `isotropic`, `tti` significantly benefits from different levels of DSE optimizations, which play a key role in reducing the operation count as well as the register pressure. Figure 6 displays the performance of `tti` for the usual range of space orders on `skl8180` and `knl7250`, for two different cubic grids.

Generally, `tti` does not reach the same level of performance as `isotropic`. This is not surprising given the complexity of the PDEs (e.g., in terms of differential operators), which translates into code with much higher arithmetic intensity. In `tti`, the memory system is stressed by a considerably larger number of loads per loop iteration than in `isotropic`. On `skl8180`, we ran some profiling with Intel® VTune™. We determined that one of the major issues is the pressure on both L1 cache (lack of split registers, unavailability of "fill buffers" to handle requests to the other levels of the hierarchy) and DRAM (bandwidth and latency). Clearly, this is only a summary from some sample kernels – the actual situation varies depending on the DSE optimizations as well as the `so` employed.

It is remarkable that on both `skl8180` and `knl7250`, and on both grids, the



(a) `skl8180`, `core`, 768^3 grid points.

(b) `knl7250`, `core`, 768^3 grid points.

(c) `skl8180`, `yask`, 748^3 grid points.

(d) `knl7250`, `yask`, 748^3 grid points.

Fig. 5: Performance of `isotropic` on multiple Devito backends and architectures.

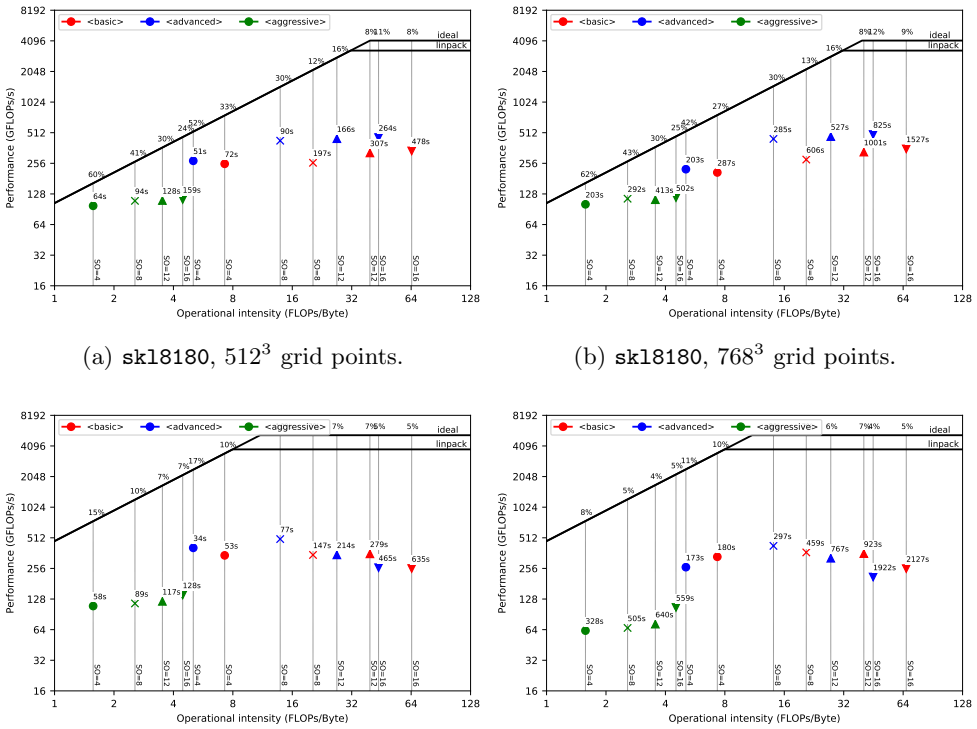
cutoff point beyond which `advanced` results in worse runtimes than `aggressive` is `so`=8. One issue with `aggressive` is that to avoid redundant computation, not only additional memory is required, but also more data communication may occur through caches, rather than through registers. In Figure 12, for example, we can easily deduce that `temp` is first stored, and then reloaded in the subsequent loop nest. This is an overhead that `advanced` does not pay, since temporaries are communicated through registers, for as much as possible. Beyond `so`=8, however, this overhead is overtaken by the reduction in operation count, which grows almost quadratically with `so`, as reported in Table 1.

Table 1: Operation counts for different DSE modes in `tti`

so	basic	advanced	aggressive
4	299	260	102
8	857	707	168
12	1703	1370	234
16	2837	2249	300

The performance on `knl7250` is overall disappointing. This is unfortunately caused by multiple factors – some of which already discussed in the previous sec-

tions. These results, and more in general, the need for performance portability across future (Intel® or non-Intel®) architectures, motivated the ongoing `yask` project. Here, the overarching issue is the inability to exploit the multiple levels of parallelism typical of architectures such as `knl7250`. Approximately 17% of the attainable peak is obtained when `so`=4 with `advanced` (best runtime out of the three DSE modes for the given space order). This occurs when using 512^3 points per grid, which allows the working set to completely fit in MCDRAM (our calculations estimated a size of roughly 7.5GB). With the larger grid size (Figure 6d), the working set increases up to 25.5GB, which exceeds the MCDRAM capacity. This partly accounts for the 5× slow down in runtime (from 34s to 173s) in spite of only a 3× increase in number of grid points computed per time iteration.



(a) `skl8180`, 512^3 grid points.

(b) `skl8180`, 768^3 grid points.

(c) `knl7250`, 512^3 grid points.

(d) `knl7250`, 768^3 grid points.

Fig. 6: Performance of `tti` on `core` for different architectures and grids.

8. Further work. While many simulation and inversion problems such as full-waveform inversion only require the solver to run on a single shared memory node, many other applications require support for distributed memory parallelism (typically via MPI) so that the solver can run across multiple compute nodes. The immediate plan is to leverage `yask`'s MPI support, and perhaps to include MPI support into `core` at a later stage. Another important feature is staggered grids, which are necessary for a wide range of FD discretization methods (e.g. modelling elastic wave propagation). Basic support for staggered grids is already included in *Devito v3.1*, but currently

only through a low-level API – the principle of graceful degradation in action. We plan to make the use of this feature more convenient.

As discussed in Section 7.4, the `yask` backend is not feature-complete yet; in particular, it cannot run the `tti` equations in the presence of array temporaries. As `tti` is among the most advanced models for wave propagation used in industry, extending Devito in this direction has high priority.

There also is a range of advanced performance optimization techniques that we want to implement, such as "time tiling" (i.e., loop blocking across the time dimension), on-the-fly data compression, and mixed-precision arithmetic exploiting application knowledge. Finally, there is an on-going effort towards adding an `ops` [31] backend, which will enable code generation for GPUs and also supports distributed memory parallelism via MPI.

9. Conclusions. Devito is a system to automate high-performance stencil computations. While Devito provides a *Python*-based syntax to easily express FD approximations of PDEs, it is not limited to finite differences. A Devito `Operator` can implement arbitrary loop nests, and can evaluate arbitrarily long sequences of heterogeneous expressions such as those arising in FD solvers, linear algebra, or interpolation. The compiler technology builds upon years of experience from other DSL-based systems such as FEniCS and Firedrake, and wherever possible Devito uses existing software components including *SymPy* and *NumPy*, and YASK. The experiments in this article show that Devito can generate production-level code with compelling performance on state-of-the-art architectures.

REFERENCES

[1] *Compilers: principles, techniques, and tools*, Pearson/Addison Wesley, Boston, MA, USA, second ed., 2007, http://www.loc.gov/catdir/toc/ecip0618/2006024333.html.

[2] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells, *Unified Form Language: a domain-specific language for weak formulations of partial differential equations*, ACM Transactions on Mathematical Software (TOMS), 40 (2014), p. 9.

[3] A. Arbona, B. Miñano, A. Rigo, C. Bona, C. Palenzuela, A. Artigues, C. Bona-Casas, and J. Massó, *Simflowny 2: An upgraded platform for scientific modeling and simulation*, arXiv preprint arXiv:1702.04715, (2017).

[4] U. Bondhugula, A. Hartono, J. Ramanujam, and P. Sadayappan, *A practical automatic polyhedral parallelizer and locality optimizer*, in Proceedings of the 2008 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '08, New York, NY, USA, 2008, ACM, pp. 101–113, https://doi.org/10.1145/1375581.1375595, http://doi.acm.org/10.1145/1375581.1375595.

[5] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, vol. 15, Springer New York, New York, NY, 2008, https://doi.org/10.1007/978-0-387-75934-0, http://dx.doi.org/10.1007/978-0-387-75934-0.

[6] A. F. Cárdenas and W. J. Karplus, *Pdel—a language for partial differential equations*, Communications of the ACM, 13 (1970), pp. 184–191.

[7] G. O. Cook Jr, *Alpal: A tool for the development of large-scale simulation codes*, tech. report, Lawrence Livermore National Lab., CA (USA), 1988.

[8] R. Courant, K. Friedrichs, and H. Lewy, *On the partial difference equations of mathematical physics*, International Business Machines (IBM) Journal of Research and Development, 11 (1967), pp. 215–234, https://doi.org/10.1147/rd.112.0215.

[9] K. Datta, S. Williams, V. Volkov, J. Carter, L. Oliker, J. Shalf, and K. Yelick, *Auto-tuning the 27-point stencil for multicore*, in In Proc. iWAPT2009: The Fourth International Workshop on Automatic Performance Tuning, 2009.

[10] S. J. Deitz, B. L. Chamberlain, and L. Snyder, *Eliminating redundancies in sum-of-product array computations*, in Proceedings of the 15th International Conference on Supercomputing, ICS '01, New York, NY, USA, 2001, ACM, pp. 65–77, https://doi.org/10.1145/377792.377807, http://doi.acm.org/10.1145/377792.377807.

[11] Y. Ding and X. Shen, *Glore: Generalized loop redundancy elimination upon ler-notation*, Proc. ACM Program. Lang., 1 (2017), pp. 74:1–74:28, https://doi.org/10.1145/3133898, http://doi.acm.org/10.1145/3133898.
[12] S. Fomel, P. Sava, I. Vlad, Y. Liu, and V. Bashkardin, *Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments*, Journal of Open Research Software, 1 (2013), p. e8, https://doi.org/http://dx.doi.org/10.5334/jors.ag.
[13] T. O. Foundation, *OpenFOAM v5 User Guide*, https://cfd.direct/openfoam/user-guide/.
[14] K. A. Hawick and D. P. Playne, *Simulation software generation using a domain-specific language for partial differential field equations*, in 11th International Conference on Software Engineering Research and Practice (SERP'13), no. CSTN-187, Las Vegas, USA, 22-25 July 2013, WorldComp, p. SER3829.
[15] R. L. Higdon, *Numerical absorbing boundary conditions for the wave equation*, Mathematics of Computation, 49 (1987), pp. 65–90, http://www.jstor.org/stable/2008250.
[16] C. T. Jacobs, S. P. Jammy, and N. D. Sandham, *Opensbli: A framework for the automated derivation and parallel execution of finite difference solvers on a range of computer architectures*, CoRR, abs/1609.01277 (2016), http://arxiv.org/abs/1609.01277.
[17] J. Jeffers and J. Reinders, *High Performance Parallelism Pearls Volume Two: Multicore and Many-core Programming Approaches*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st ed., 2015.
[18] A. Klöckner, *Loo.py: transformation-based code generation for GPUs and CPUs*, in Proceedings of ARRAY '14: ACM SIGPLAN Workshop on Libraries, Languages, and Compilers for Array Programming, Edinburgh, Scotland., 2014, Association for Computing Machinery, https://doi.org/{10.1145/2627373.2627387}.
[19] A. Klckner, *Cgen - c/c++ source generation from an ast.* https://github.com/inducer/cgen, 2016.
[20] S. Kronawitter, S. Kuckuk, and C. Lengauer, *Redundancy elimination in the exastencils code generator*, in ICA3PP Workshops, 2016.
[21] C. Lengauer, S. Apel, M. Bolten, A. Größlinger, F. Hannig, H. Köstler, U. Rüde, J. Teich, A. Grebhahn, S. Kronawitter, S. Kuckuk, H. Rittich, and C. Schmitt, *Exastencils: Advanced stencil-code engineering*, in Euro-Par 2014: Parallel Processing Workshops - Euro-Par 2014 International Workshops, Porto, Portugal, August 25-26, 2014, Revised Selected Papers, Part II, 2014, pp. 553–564, https://doi.org/10.1007/978-3-319-14313-2_47, https://doi.org/10.1007/978-3-319-14313-2_47.
[22] A. Logg, K.-A. Mardal, G. N. Wells, et al., *Automated Solution of Differential Equations by the Finite Element Method*, Springer, 2012, https://doi.org/10.1007/978-3-642-23099-8.
[23] M. Louboutin, M. Lange, F. J. Herrmann, N. Kukreja, and G. Gorman, *Performance prediction of finite-difference solvers for different computer architectures*, Computers & Geosciences, 105 (2017), pp. 148–157, https://doi.org/https://doi.org/10.1016/j.cageo.2017.04.014, https://www.slim.eos.ubc.ca/Publications/Public/Journals/ComputersAndGeosciences/2016/louboutin2016ppf/louboutin2016ppf.pdf. (Computers & Geosciences).
[24] M. Louboutin, M. Lange, F. Luporini, N. Kukreja, P. A. Witte, P. Velesko, G. Gorman, and F. J. Herrmann, *Devito: A portable and flexible mathematical api for geophysical applications.* 2018.
[25] G. R. Markall, F. Rathgeber, L. Mitchell, N. Loriant, C. Bertolli, D. A. Ham, and P. H. J. Kelly, *Performance-portable finite element assembly using pyop2 and fenics*, in 28th International Supercomputing Conference, ISC, Proceedings, J. M. Kunkel, T. Ludwig, and H. W. Meuer, eds., vol. 7905 of Lecture Notes in Computer Science, Springer, 2013, pp. 279–289, https://doi.org/10.1007/978-3-642-38750-0_21, http://dx.doi.org/10.1007/978-3-642-38750-0_21.
[26] Mathias Louboutin, Fabio Luporini, *Boundary conditions in Devito*, in preparation (2018).
[27] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, *Sympy: symbolic computing in python*, PeerJ Computer Science, 3 (2017), p. e103, https://doi.org/10.7717/peerj-cs.103, https://doi.org/10.7717/peerj-cs.103.
[28] S. J. Pennycook, J. Sewall, and V. Lee, *A metric for performance portability*, arXiv preprint arXiv:1611.07409, (2016).
[29] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe, *Halide: A language and compiler for optimizing parallelism, locality, and recomputation*

in image processing pipelines, in Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, New York, NY, USA, 2013, ACM, pp. 519–530, https://doi.org/10.1145/2491956.2462176, http://doi.acm.org/10.1145/2491956.2462176.

[30] F. Rathgeber, D. A. Ham, L. Mitchell, M. Lange, F. Luporini, A. T. T. Mcrae, G.-T. Bercea, G. R. Markall, and P. H. J. Kelly, *Firedrake: Automating the finite element method by composing abstractions*, ACM Trans. Math. Softw., 43 (2016), pp. 24:1–24:27, https://doi.org/10.1145/2998441, http://doi.acm.org/10.1145/2998441.

[31] I. Z. Reguly, G. R. Mudalige, M. B. Giles, D. Curran, and S. McIntosh-Smith, *The ops domain specific abstraction for multi-block structured grid computations*, in Proceedings of the Fourth International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing, WOLFHPC '14, Piscataway, NJ, USA, 2014, IEEE Press, pp. 58–67, https://doi.org/10.1109/WOLFHPC.2014.7, http://dx.doi.org/10.1109/WOLFHPC.2014.7.

[32] W. W. Symes, D. Sun, and M. Enriquez, *From modelling to inversion: designing a well-adapted simulator*, Geophysical Prospecting, 59 (2011), pp. 814–833, https://doi.org/10.1111/j.1365-2478.2011.00977.x, http://dx.doi.org/10.1111/j.1365-2478.2011.00977.x.

[33] J. Tobin, A. Breuer, A. Heinecke, C. Yount, and Y. Cui, *Accelerating seismic simulations using the intel xeon phi knights landing processor*, in Proceedings of ISC High Performance 2017 (ISC17), to appear 2017.

[34] Y. Umetani, *Deqsol a numerical simulation language for vector/parallel processors*, Proc. IFIP TC2/WG22, 1985, 5 (1985), pp. 147–164.

[35] R. Van Engelen, L. Wolters, and G. Cats, *Ctadel: A generator of multi-platform high performance codes for pde-based scientific applications*, in Proceedings of the 10th international conference on Supercomputing, ACM, 1996, pp. 86–93.

[36] F. Witherden, A. Farrington, and P. Vincent, *Pyfr: An open source framework for solving advectiondiffusion type problems on streaming architectures using the flux reconstruction approach*, Computer Physics Communications, 185 (2014), pp. 3028 – 3040, https://doi.org/https://doi.org/10.1016/j.cpc.2014.07.011, http://www.sciencedirect.com/science/article/pii/S0010465514002549.

[37] C. Yount, *Vector folding: Improving stencil performance via multi-dimensional simd-vector representation*, in Proceedings of the IEEE 17th International Conference on High Performance Computing and Communications (HPCC), Aug 2015, pp. 865–870, https://doi.org/10.1109/HPCC-CSS-ICESS.2015.27.

[38] C. Yount and A. Duran, *Effective use of large high-bandwidth memory caches in HPC stencil computation via temporal wave-front tiling*, in Proceedings of the 7th International Workshop in Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems held as part of ACM/IEEE Supercomputing 2016 (SC16), PMBS'16, Nov 2016.

[39] C. Yount, A. Duran, and J. Tobin, *Multi-level spatial and temporal tiling for efficient hpc stencil computation on many-core processors with large shared caches*, Future Generation Computer Systems, (2017), https://doi.org/https://doi.org/10.1016/j.future.2017.10.041, http://www.sciencedirect.com/science/article/pii/S0167739X17304648.

[40] C. Yount, J. Tobin, A. Breuer, and A. Duran, *Yask–yet another stencil kernel: a framework for hpc stencil code-generation and tuning*, in Proceedings of the 6th International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing held as part of ACM/IEEE Supercomputing 2016 (SC16), WOLFHPC'16, Nov 2016, https://doi.org/10.1109/WOLFHPC.2016.08.

[41] Zenodo/devito, *Devito v3.1*, October 2017, https://doi.org/10.5281/zenodo.836688.

[42] Zenodo/devito-performance, *Devito Experimentation Framework*, July 2018, https://doi.org/TODO.

[43] Y. Zhang and F. Mueller, *Auto-generation and auto-tuning of 3d stencil codes on gpu clusters*, in Proceedings of the Tenth International Symposium on Code Generation and Optimization, CGO '12, New York, NY, USA, 2012, ACM, pp. 155–164, https://doi.org/10.1145/2259016.2259037, http://doi.acm.org/10.1145/2259016.2259037.

[44] Y. Zhang, H. Zhang, and G. Zhang, *A stable tti reverse time migration and its implementation*, GEOPHYSICS, 76 (2011), pp. WA3–WA11, https://doi.org/10.1190/1.3554411, https://doi.org/10.1190/1.3554411, https://arxiv.org/abs/https://doi.org/10.1190/1.3554411.

SOLVING WAVE EQUATIONS IN THE CURVELET DOMAIN: A MULTI-SCALE AND MULTI-DIRECTIONAL APPROACH

BINGBING SUN[1], JIANWEI MA[1,2], HERVÉ CHAURIS[2] and HUIZHU YANG[1]

[1] *Institute of Seismic Exploration, School of Aerospace, Tsinghua University, P.R. China.*
[2] *Centre de Géosciences, Mines ParisTech, Paris, France. herve.chauris@mines-paristech.fr*

(Received April 15, 2009; revised version accepted June 15, 2009)

ABSTRACT

Sun, B., Ma, J., Chauris, H. and Yang, H., 2009. Solving wave equations in the curvelet domain: a multi-scale and multi-directional approach. *Journal of Seismic Exploration*, 18: 385-399.

Seismic imaging is a key step in seismic exploration to retrieve the earth properties from seismic measurements at the surface. One needs to properly model the response of the earth by solving the wave equation. We present how curvelets can be used in that respect. Curvelets can be seen from the geophysical point of view as the representation of local plane waves. The unknown pressure, solution of the wave equation, is decomposed in the curvelet domain. We derive the new associated equation for the curvelet coefficients and show how to solve it. In this paper, we focus on a simple homogeneous model to illustrate the feasibility of the curvelet-based method. This is a first step towards the modeling in more complex models. In particular, we express the derivative of the wave field in the curvelet domain. The simulation results show that our algorithm can give a multi-scale and multi-directional view of the wave propagation. A potential application is to model the wave motion in some specific directions. We also discuss the current limitations of this approach, in particular the extension to more complex models.

KEYWORDS: curvelets, wavelets, numerical simulation, wave equation, multi-scale, multi-directional, adaptive.

INTRODUCTION

In the context of seismic oil and gas exploration, numerical simulation of the wave equation is the key factor to establish the link between the earth properties and the observed data at the surface. Among several traditional methods for the wave field simulation, the finite difference method (Alford et al., 1974; Alterman and Karal, 1968; Kelly et al., 1976) is the most popular.

0963-0651/09/$5.00

It uses values on several grid points to estimate the derivative at a particular location. Its advantage relies on the relatively easy programming and high speed calculation. The pseudospectral method (Fornberg, 1989; Gazdag, 1981; Kosloff and Baysal, 1982) have also been proved to be an efficient method. Compared to low-order finite difference, the pseudospectral method provides more accurate result as it uses all grid point values to estimate the derivatives. Beyond these two methods, the finite element method (Lysmer and Draker, 1972; Meyer, 1992) is based on the variational principle while the boundary element method (Carrer et al., 2008; Funato and Fukui, 1999) is based on the integral principle. Both methods have good performance in stability and convergence, and can deal with complicated boundary conditions. However, these two methods are generally memory- and time-consuming.

In the past, the mentioned methods have been improved in terms of speed and accuracy. We focus here on a different aspect, related to multi-scale and multi-direction analysis. Wavelet schemes (Carrer et al., 2008; Kelly et al., 1976) provide a multi-scale solution where the basis elements are relatively well localized in the time and frequency domains, with a large number of applications in data processing (Antoine et al., 2004; Daubechies, 1992). Combined with other methods, wavelets can be used to solve the wave equation (Hong and Kennett, 2002a; Hong and Kennett, 2002b; Ma et al., 2001). The numerical algorithms have a number of advantages: the differential operator can be directly computed in the wavelet domain with high speed and accuracy; by setting threshold values in the wavelet domain, significant coefficients can be selected to reduce the calculations and the memory requirement; it is also possible to derive the wave field corresponding to certain scales without calculating the whole wave field.

Compared to wavelets, curvelets appears to be more suitable for solving hyperbolic problems (Candès and Demanet, 2005; Candès and Donoho, 2003). Curvelets (Candès and Donoho, 2000, 2004, 2005) were recently introduced in the field of applied harmonic analysis and have a number of applications on seismic processing (Lin and Herrmann, 2007; Herrmann et al., 2008; Ma et al., 2007; Ma and Plonka, 2009a, 2009b). They provide a multi-direction analysis and allow a sparse representation of smooth objects containing smooth discontinuities (i.e., twice continuously differentiable). Curvelets preserve the same time-frequency localization property as for wavelets and at the same time, with their elongated support in the Fourier domain, curvelets become directional.

Compared to the Fourier and wavelet analysis, the curvelet decomposition (Candès and Demanet, 2005; Candès and Donobo, 2003) theoretically provides an optimally sparse representation of the wave propagator. For example, Demanet (2006) uses curvelet-like wave atoms to solve the wave equation. His

algorithm is based on the sparsity of the matrix representation of Green's function. In his paper, the curvelets are used as basic functions and the derivatives are calculated through them. Because of lack of explicit expressions of the curvelets in the time domain, the derivative is computed numerically in the frequency domain, raising the computation aspect. Andersson et al. (2008) developed a method based on the Volterra equation (high frequency approximation) and considers the rigid motion of the curvelet along the ray. The wave operator is decomposed into individual scale using para-differential decomposition. The Hamilton flow at a given scale determines the motion of the curvelets. As for curvelet-based migration (Chauris and Nguyen, 2008; Douma and de Hoop, 2007), that focuses on the action of the migration operator on the curvelets, only the first-order approximation of the distorted curvelet is predicted. Moreover, the location of the discrete curvelets is not necessarily on the original grid, requiring some interpolation procedure (Chauris and Nguyen, 2008). Ma et al. (2007) introduced the curvelet transform into AMR (adaptive mesh refinement) to estimate the local error of the wave field so as to update the grid. This method suffers from the lack of analytical expression in space of the current curvelet transform.

In order to get a multi-scale and multi-direction analysis of the wave propagation, and for calculation efficiency, we solve the wave equation directly in the curvelet coefficient domain. In this paper, we show how to express the derivative of a function in the curvelet domain. The curvelet decomposition is coupled to the Finite Difference scheme, providing a multi-scale and multi-direction analysis of the wave field.

The outline of the paper is as follows. First, we explain the curvelet construction and the main properties of curvelets. Then, we show how to solve the wave equation in the curvelet domain for homogeneous models, including numerical examples. Finally, we discuss the current limitations of the approach.

CURVELETS

In this section, we give a brief introduction of the curvelet transform. First we define the continuous curvelet transform (Candes and Donoho, 2004; Candes et al., 2005).

In the two-dimensional space R^2, define the spatial variable $x = (x_1,x_2)$ and its counterpart $\omega = (\omega_1,\omega_2)$ in the frequency domain or alternatively the polar coordinates $r = \sqrt{(\omega_1^2 + \omega_2^2)}$ and $\theta = \arctan(\omega_2/\omega_1)$. We first start with a pair of window functions $W(r)$ and $V(t)$, called the "radial window" and "angular window". These window functions are both smooth, non-negative and real-valued, with W taking positive real arguments and supported on $r \in (1/2,2)$ and V taking real arguments and supported on $t \in [-1,1]$. Both obey

the admissibility conditions:

$$\sum_{j=-\infty}^{\infty} W^2(2^j r) = 1, r \in (3/4,3/2) \quad , \tag{1}$$

$$\sum_{l=-\infty}^{\infty} V^2(t - l) = 1, t \in (-1/2,1/2) \quad . \tag{2}$$

For $j \geq j_0$, we define the filter U_j in the frequency domain by

$$U_j(r,\theta) = 2^{-3j/4} W(2^{-j} r) V(2^{[j/2]}\theta/2\pi) \quad , \tag{3}$$

where [j/2] denotes the integer part of j/2. The support of U_j is a polar "wedge".

The waveform $\varphi_j(x)$ is defined by its Fourier transform $\hat{\varphi}_j(\omega) = U_j(\omega)$ where $U_j(\omega)$ is defined in the polar coordinate system by eq. (3). All the curvelets at scale 2^{-j} are obtained by rotations and translations of φ_j. Given the equispaced sequence of rotation angles $\theta_l = 2\pi \cdot 2^{-[j/2]} \cdot l$, with $l = 0,1,\ldots 2^{[j/2]}$, so $0 \leq \theta_l \leq 2\pi$ and the sequence of translation parameters $k = (k_1,k_2) \in Z^2$. Note the spacing between angles is scale-dependent. We define curvelets at scale 2^{-j}, orientation θ_l, and position $x_k^{(j,l)} = R^{-} (k_1 \cdot 2^{-j}, k_2 \cdot 2^{-j/2})$ [Let b denote $(k_1 \cdot 2^{-j}, k_2 \cdot 2^{-j/2})$].

$$\varphi_{j,l,k}(x) = \varphi_j[R_{\theta_l}(x - x_k^{(j,l)})] = \varphi_j(R_{\theta_l} x - b) \quad , \tag{4}$$

where R_{θ_l} is the rotation by θ_l radians and $R_{\theta_l}^{-}$ its inverse,

$$R_{\theta_l} = \begin{pmatrix} \cos\theta_l & \sin\theta_l \\ -\sin\theta_l & \cos\theta_l \end{pmatrix} \quad . \quad R_{\theta_l}^{-} = R_{\theta_l}^{T} = R_{-\theta_l} \tag{5}$$

The curvelet family forms a tight frames: we can expand a function $f \in L^2(R^2)$ as a series of curvelets:

$$f = \sum_{j,l,k} \langle f, \varphi_{j,l,k} \rangle \varphi_{j,l,k} \quad , \tag{6}$$

where $C_{j,l,k}$ denotes the curvelet coefficient or scalar product $\langle f, \varphi_{j,l,k} \rangle$

$$C_{j,l,k} = <f,\varphi_{j,l,k}> = \int_{R^2} f(x)\overline{\varphi_{j,l,k}(x)}dx \quad , \tag{7}$$

According to the Plancherel's theorem, we can express the inner product as the integral over the frequency domain.

$$C_{j,l,k} = [1/(2\pi)^2] \int \hat{f}(\omega)\overline{\hat{\varphi}_{j,l,k}(\omega)}d\omega$$

$$= [1/(2\pi)^2] \int \hat{f}(\omega)U_j(R_{\theta_l}\omega)e^{i<x_k^{(j,l)},\omega>} d\omega \quad . \tag{8}$$

The phase shift in the frequency domain corresponds to a shift in the spatial domain. We introduce the low-pass window W_0 obeying

$$|W_0|^2 + \sum_{j>j_0} |W(2^{-j}r)|^2 = 1 \quad , \tag{9}$$

and define coarse scale curvelets as

$$\varphi_{j_0,k}(x) = \varphi_{j_0}(x - 2^{-j_0}k) \quad , \tag{10}$$

$$\hat{\varphi}_{j_0}(\omega) = 2^{-j_0}(2^{-j_0}|\omega|) \quad . \tag{11}$$

The curvelets associated to the coarse scale curvelets are non directional. The curvelet transform consists of two parts: the fine scale directional elements $(\varphi_{j,l,k})_{j>j_0,l,k}$, the coarse scale isotropic curvelets $(\varphi_{j_0,k})_k$.

At least two existing codes are available (Candes et al., 2005; Ma et al., 2007) for practical applications, respectively based on the USFFT and wrapping. We introduce the USFFT. In the continuous-time definition, the window U_j is defined over the dyadic corona $\{2^j \leq r \leq 2^{j+1}\}$ and the angle $\{-\pi\cdot 2^{-j/2} \leq \theta \leq \pi\cdot 2^{-j/2}$. These are not especially well-suited for Cartesian arrays. We introduce the equivalents: "Cartesian coronae" based on concentric squares, instead of circles, and shears instead of rotations.

The Cartesian analog to the "radial window $(W_j)_{j\geq 0}$, would be a window of the form

$$\tilde{W}_j(\omega) = \sqrt{(\Phi_{j+1}^2 - \Phi_j^2)} \quad , \tag{12}$$

where Φ is defined as the product of low-pass one dimensional windows

$$\Phi_j(\omega) = \phi(2^{-j}\omega_1)\phi(2^{-j}\omega_2) \quad . \tag{13}$$

The function ϕ obeys $0 \leq \phi \leq 1$, might be equal to 1 on $[-1/2,1/2]$, and vanishes outside of $[-2,2]$. One can check that

$$\Phi_0(\omega)^2 + \sum_{j>0} \tilde{W}_j^2(\omega) = 1 \quad . \tag{14}$$

The V_j functions are defined as

$$V_j(\omega) = V(2^{[j/2]}\omega_2/\omega_1) \quad . \tag{15}$$

We can define the "Cartesian" frequency window:

$$\tilde{U}_j(\omega) = \tilde{W}_j(\omega)V_j(\omega) \quad . \tag{16}$$

It is clear that $\tilde{U}_j$ isolates frequency near the wedge $\{(\omega_1,\omega_2) : 2^j \leq \omega_1 \leq 2^{j+1}, -2^{-j/2} \leq \omega_2/\omega_1 \leq 2^{-j/2}\}$. We introduce the equispaced slopes $\tan\theta_l = l\cdot 2^{[-j/2]}$, $l = -2^{j/2},\ldots,2^{[j/2]} - 1,2^{[j/2]}$, and define

$$\tilde{U}_{j,l}(\omega) = \tilde{W}_j(\omega)V_j(S_{\theta_l}\omega) \quad , \tag{17}$$

where S_{θ_l} is the shear matrix,

$$S_{\theta_l} = \begin{pmatrix} 1 & 0 \\ -\tan\theta_l & 1 \end{pmatrix} \quad . \tag{18}$$

The angles θ_l are not equispaced here but the slopes are. After completion by symmetry around the origin and rotation by $\pm\pi/2$ radians, the $\tilde{U}_{j,l}$ define the Cartesian analog to the family $U_j(R_{\theta_l}\omega)$ as mentioned before. The digital coronization suggests Cartesion curvelets of the form

$$\tilde{\varphi}_{j,l,k}(x) = 2^{3j/4}\tilde{\varphi}_j[S_{\theta_l}(x - S_{\theta_l}^{-}b)] \quad ,$$

where b takes on the discrete values $b = (b_1,b_2) = (k_1\cdot 2^{-j},k_2\cdot 2^{-j/2})$. The new expression of the curvelet coefficient is

$$C_{j,l,k} = \int \hat{f}(\omega)\tilde{U}_j(S_{\theta_l}\omega)e^{i<S_{\theta_l}^{-}b,\omega>}d\omega \quad , \tag{19}$$

$$= \int \hat{f}(S_{\theta_l}\omega)\tilde{U}_j(\omega)e^{i<b,\omega>}d\omega \quad . \tag{20}$$

For a discretized version of f on a Cartesian array $f[t_1,t_2], 0 \le t_1,t_2 < n$, where $\hat{f}[n_1,n_2]$ denotes its 2D discrete Fourier transform

$$\hat{f}[n_1,n_2] = \sum_{t_1,t_2=0}^{n-1} f[t_1,t_2]e^{-i2\pi(n_1t_1 + n_2t_2)/n} \quad , \quad -n/2 \le n_1,n_2 < n/2 \tag{21}$$

where $\hat{f}[n_1,n_2] = \hat{f}(2\pi n_1, 2\pi n_2)$. Assume that $\tilde{U}_j[n_1,n_2]$ is supported on some rectangle of length $L_{1,j}$ and width $L_{2,j}$

$$\mathcal{P}_j = \{(n_1,n_2): n_{1,0} \le n_1 < n_{1,0} + L_{1,j}, n_{2,0} \le n_2 < n_{2,0} + L_{2,j}\} \quad , \tag{23}$$

where $(n_{1,0},n_{2,0})$ is the index of the pixel at the bottom-left of the rectangle. Because of the parabolic scaling, $L_{1,j}$ is bout 2^j and $L_{2,j}$ is about $2^{j/2}$. With these notations, the Fast Discrete Curvelet Transform (FDCT) via USFFT evaluates

$$C_{j,l,k} = \sum_{n_1,n_2 \in \mathcal{P}_j} \hat{f}[n_1, n_2 - n_1\tan\theta_l]\tilde{U}_j[n_1,n_2]e^{i2\pi(k_1n_1/L_{1,j} + k_2n_2/L_{2,j})} \quad , \tag{24}$$

We summarize the FDCT via USFFT as follows:

1. Apply the 2D FFT and obtain Fourier samples $\hat{f}[n_1,n_2]$, $-n/2 \le n_1,n_2 < n/2$.

2. For each scale and angle pair (j,l), resample (or interpolate) to obtain sampled values $\hat{f}[n_1, n_2 - n_1\tan\theta_l]$ for $n_1,n_2 \in \mathcal{P}_j$.

3. Multiply the interpolated object $\hat{f}$ with the parabolic window $\tilde{U}_j$, obtain $\hat{f}_{j,l}[n_1,n_2] = \hat{f}[n_1, n_2 - n_1\tan\theta_l]\tilde{U}_j[n_1,n_2])$.

4. Apply the inverse 2D FFT to each $\hat{f}_{j,l}$ to get the discrete coefficients $C_{j,l,k}$.

Fig. 1 gives the spatial and frequency view of a specific curvelet. Ma and Plonka (2007) presented a periodic curvelet transform. For more details on curvelets and recent applications, we refer to reviewal papers (Ma and Plonka, 2009a, 2009b).

SOLVING THE WAVE EQUATION IN THE CURVELET DOMAIN

The main objective is to solve the wave equation in the curvelet domain. The simplest form corresponds to the 2D constant density scalar wave equation.

$$\partial^2u(t,x)\partial/t^2 = a^2\Delta u(t,x), \quad u(t,x)\big|_{t=0} = u_1(x), \quad \partial u(t,x)/\partial t\big|_{t=0} = u_2(x), \tag{25}$$

where u(x,t) is the pressure field, a is the constant wave speed, x denotes (x_1,x_2), and Δ, the Laplace operator $(\partial^2/\partial x_1^2) + (\partial^2/\partial x_2^2)$. We give a detailed description of our new method. It is based on the transposition of the finite difference method in the curvelet domain. We first address how to estimate the spatial derivatives, specifically the Laplacian acting on the pressure u. Our ideas are inspired by the pseudospectral method that establishes a relationship between the Fourier transform of a function and its derivatives. We derive a similar relationship, leading to a new expression of the wave equation in the curvelet domain in the case of homogeneous models. Let us define the curvelet functions in the frequency domain $\hat{\varphi}_{j,l,k}(\omega)$ as

$$\hat{\varphi}_{j,l,k}(\omega) = \tilde{U}_j(S_{\theta_l}\omega)e^{i<S_{\theta_l}b,\omega>} \quad . \tag{26}$$

By construction, the curvelet coefficients associated to u or to its Laplacian are defined by

$$C_{j,l,k} = [1/(2\pi)^2] \int \hat{u}(\omega)\overline{\hat{\varphi}_{j,l,k}(\omega)}\, d\omega \quad , \tag{27}$$

$$C^{\Delta}_{j,l,k} = [1/(2\pi)^2] \int FT(\Delta u)\overline{\hat{\varphi}_{j,l,k}(\omega)}\, d\omega \quad . \tag{28}$$

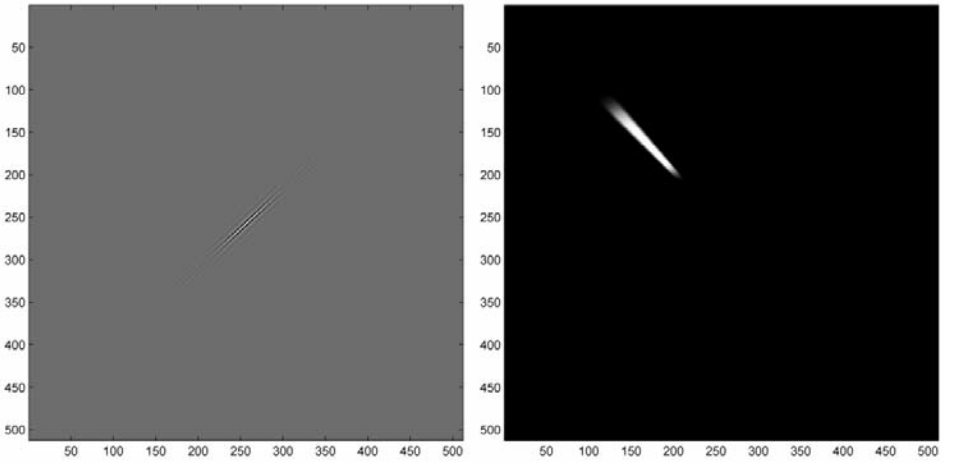


Fig. 1. Spatial and frequency view of a curvelet.

The basic properties of the Fourier Transform (FT) about the derivative give

$$FT(\partial^n f/\partial t^n) = (i\omega)^n \hat{f}(\omega) \quad , \tag{29}$$

so that

$$C^{\Delta}_{j,l,k} = -[1/(2\pi)^2] \int |\omega|^2 \hat{u}(\omega)\overline{\hat{\varphi}_{j,l,k}(\omega)}\, d\omega \quad . \tag{30}$$

We exploit the specific form for $\hat{\varphi}_{j,l,k}$ and perform a change of variables from S ω to ω, yielding to

$$C_{j,l,k} = [1/(2\pi)^2] \int S^{-}_{\theta_l} \hat{u}(\omega)\tilde{U}_j(\omega)e^{i<b,\omega>}\, d\omega \quad , \tag{31}$$

$$C^{\Delta}_{j,l,k} = -[1/(2\pi)^2] \int |S^{-}_{\theta_l}(\omega)|^2 \cdot S^{-}_{\theta_l} \hat{u}(\omega)\tilde{U}_j(\omega)e^{i<b,\omega>}\, d\omega \quad , \tag{32}$$

$$= -[1/(2\pi)^2] \int (\omega_1^2/\cos^2\theta_l + \omega_2^2 - 2\omega_1\omega_2 \tan_{\theta_l}) \cdot S^{-}_{\theta_l} \hat{u}(\omega)\tilde{U}_j(\omega)e^{i<b,\omega>} d\omega \, . \tag{33}$$

We observe that $C^{\Delta}_{j,l,k}$ is composed of three parts, $C^{\Delta}_{k_1}$, $C^{\Delta}_{k_2}$, $C^{\Delta}_{k_1k_2}$:

$$C^{\Delta}_{k_1} = [1/(2\pi\cos\theta_l)^2] \int -\omega_1^2 \tilde{u}(\omega)e^{i<b,\omega>} d\omega \quad , \tag{34}$$

$$C^{\Delta}_{k_2} = [1/(2\pi)^2] \int -\omega_2^2 \tilde{u}(\omega)e^{i<b,\omega>} d\omega \quad , \tag{35}$$

$$C^{\Delta}_{k_1k_2} = [2\tan\theta_l/(2\pi)^2] \int \omega_1\omega_2 \tilde{u}(\omega)e^{i<b,\omega>} d\omega \quad . \tag{36}$$

Consider eq. (34), the integral part can be seen as the inverse Fourier transform of $-\omega_1^2\tilde{u}(\omega)$. We thus obtain

$$C^{\Delta}_{k_1} = (1/\cos^2\theta_l)(\partial^2 C_{j,l,k}/\partial b_1^2) \quad , \tag{37}$$

$$C^{\Delta}_{k_2} = \partial^2 C_{j,l,k}/\partial b_2^2 \quad , \tag{38}$$

$$C^{\Delta}_{k_1k_2} = -2\tan\theta_l (\partial^2 C_{j,l,k}/\partial b_1 \partial b_2) \quad . \tag{39}$$

We now get an expression for the wave equation in the curvelet domain

$$\partial^2 C_{j,l,k}/\partial t^2 = a^2[(1/\cos^2\theta_l)(\partial^2 C_{j,l,k}/\partial b_1^2) + (\partial^2 C_{j,l,k}/\partial b_2^2) - 2\tan\theta_l(\partial^2 C_{j,l,k}/\partial b_1 \partial b_2)] \, . \tag{40}$$

We can use the central difference method or pseudo-spectrum method to

calculate the coefficients. Of course, any efficient methods for solving partial differential equations can be used theoretically.

NUMERICAL SIMULATION AND DISCUSSION

In this section, we present the numerical simulation in an homogeneous velocity model. We include a source term $p(x,t) = \delta(x - x_0)f(t)$ where f is a Ricker wavelet.

$$\partial^2 u/\partial t^2 = a^2[(\partial^2 u/\partial x_1^2) + (\partial^2 u/\partial x_2^2)] + p(x,t) \quad , \tag{41}$$

$$u(t,x)\big|_{t=0} = u_1(x), \quad \partial u(t,x)/\partial t\big|_{t=0} = u_2(x).$$

The complete form of the wave equation in the curvelet domain is given by

$$\partial^2 C_{j,l,k}/\partial t^2 = a^2[(1 + \tan^2\theta_l)(\partial^2 C_{j,l,k}/\partial b_1^2) + (\partial^2 C_{j,l,k}/\partial b_2^2) - 2\tan\theta_l(\partial^2/C_{j,l,k}/\partial b_1 \partial b_2)] + C^{\delta}_{j,l,k} f(t) \quad , \tag{42}$$

where $C^{\delta}_{j,l,k}$ denotes the curvelet transform of function $\delta(x - x_0)$.

We use explicit finite difference method and pseudospectral method to solve eq. (42). The calculation process is as follows:

1. Select the number of scales and directions used in curvelet transform and accordingly create the discrete grids in the curvelet domain. Note that the spatial and time steps are scale dependent.

2. Solve eq. (42) using numerical method in the curvelet domain. High accuracy numerical method is recommended for the estimation of the derivatives.

3. Apply the inverse curvelet transform to the final calculated curvelet coefficients $C_{i,j,k}$. We can also apply the curvelet transform to the data for some specific scales and directions to get a multi-scale and multi-direction view.

In our example, the grid size for u is 256 × 256, and the step size 10 m. The medium velocity is 1500 m/s and the main frequency of the ricker is 30 Hz. The curvelet transform is applied by the wrapping method and we have coarse scale, detailed scale and fine scale. For coarse scale, the time step is 0.1 ms. In the coarse domain, the equation is solved by central difference method and in the detailed and fine scale, pseudospectral method is used for high accuracy.

Fig. 2 displays the snapshots of the wave field at the time t = 2.0 s in the coarse and detail scale. Fig. 3 shows the snapshots of every direction in the detail scale. For individual snapshots, we only use the coefficients in one direction and then apply the inverse curvelet transform. As the figure shows, the total wavefield is composed of the waves in different directions.

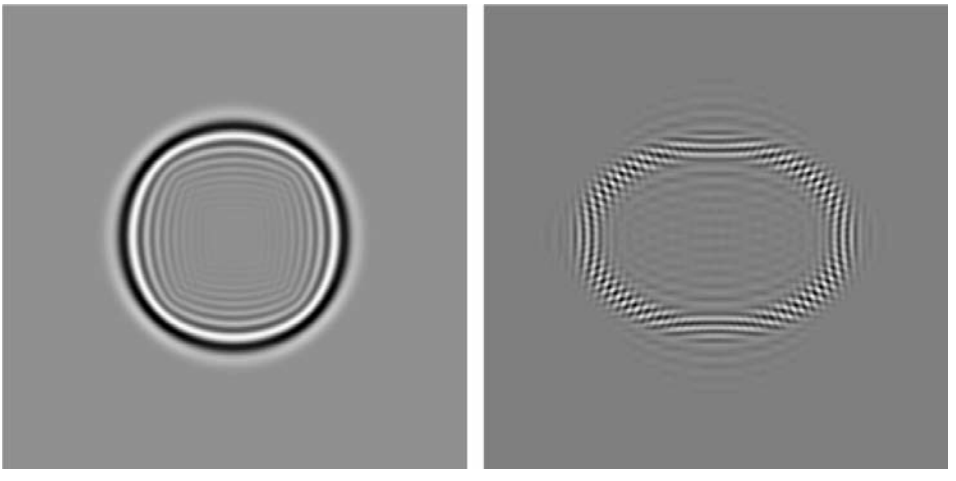


Fig. 2. Snapshot of the wave field in the coarse and detail scale.

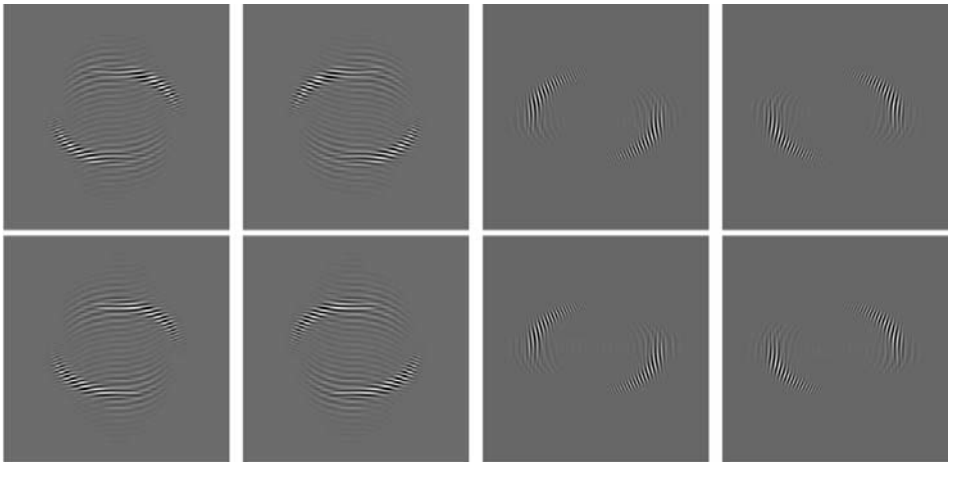


Fig. 3. Snapshot of the wave field in the detail scale for eight different directions.

Fig. 4 shows that the wave field evolves in certain scale and direction. From top-bottom and right to left, the corresponding time starts from 10 ms, every 10 ms until 160 ms. We can see how the curvelet evolves in the spatial domain. At first, the wave field is composed of several curvelets, and then such curvelets appears to split and gradually form the wave front.

Numerical results acclaim the advantages of our method: first,we can get a multi-scale analysis of the wave motion as for wavelets. By comparison, the wave field can propagate in the coarse, detailed and fine scale, it appears that the energy is much focused in the coarse scale. On the other hand, we can also obtain a multi-directional analysis of the wave motion. Using the curvelets, we

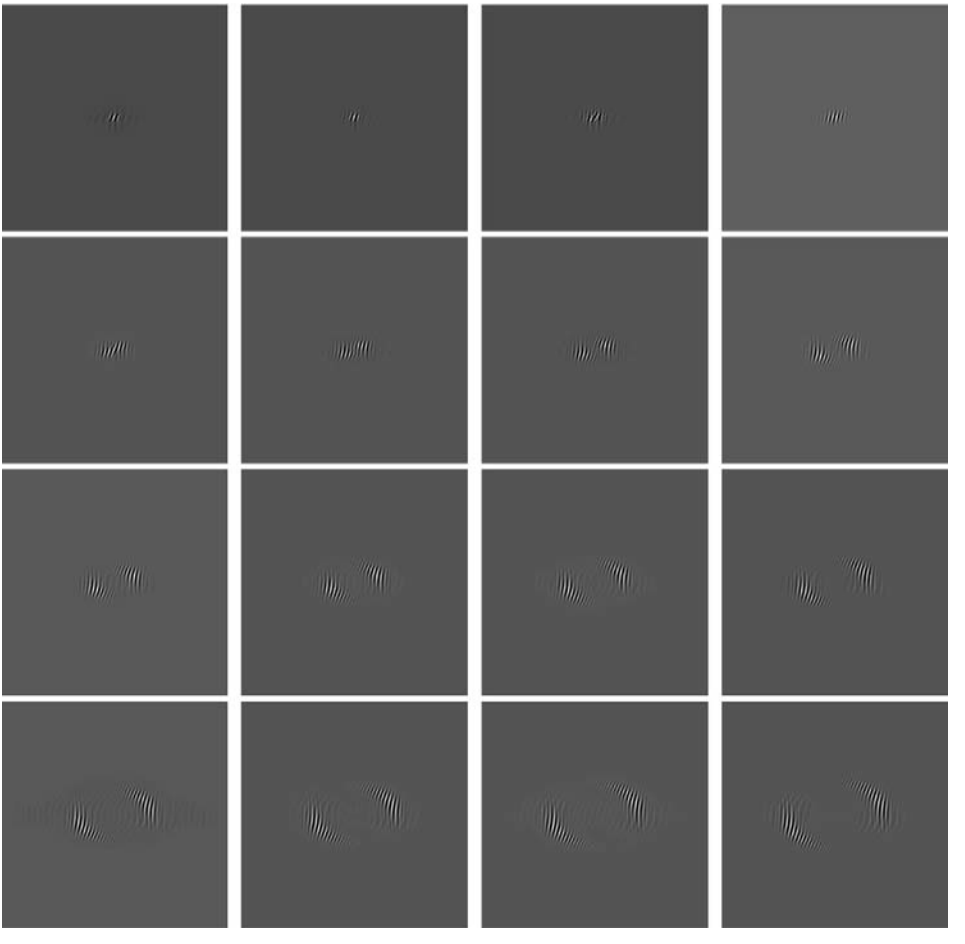


Fig. 4. Evolution of the wave field in the detail scale, from 10 ms every 10 ms until 160 ms.

have a multi-direction view of the wave propagation as the petal-like image in Fig. 3. This has a potential application for wave modeling in certain direction, because we can ignore unimportant directions and this will sufficiently reduce the computation time and storage space.

In the section on curvelets, we have presented the derivatives of arbitrary functions in the curvelet domain. We can easily extend the strategy to other linear partial differential equations with constant coefficients. For example, the 2D homogeneous elastic wave equation in the curvelet domain is:

$$\rho(\partial^2 C^u_{j,l,k}/\partial t^2 = (\lambda + \mu)[(\partial^2 C^u_{j,l,k}/\partial b_1^2) - \tan\theta_l(\partial^2 C^v_{j,l,k}/\partial b_1^2) + (\partial^2 C^v_{j,l,k}/\partial b_1 \partial b_2)]$$

$$+ \mu[(1 + \tan^2\theta_l)(\partial^2 C^u_{j,l,k}/\partial b_1^2) + (\partial^2 C^u_{j,l,k}/\partial b_2^2) - 2\tan\theta_l(\partial^2 C^u_{j,l,k}/\partial b_1 \partial b_2)] \quad , \quad (43)$$

$$\rho(\partial^2 C^v_{j,l,k}/\partial t^2 = (\lambda + \mu)[(\partial^2 C^v_{j,l,k}/\partial b_1^2) - \tan\theta_l(\partial^2 C^u_{j,l,k}/\partial b_1^2) + (\partial^2 C^u_{j,l,k}/\partial b_1 \partial b_2)]$$

$$+ \mu[(1 + \tan^2\theta_l)(\partial^2 C^v_{j,l,k}/\partial b_1^2) + (\partial^2 C^v_{j,l,k}/\partial b_2^2) - 2\tan\theta_l(\partial^2 C^v_{j,l,k}/\partial b_1 \partial b_2)] \quad , \quad (44)$$

where λ, μ are the Lamé constants, ρ the density. $C^u_{j,l,k}$, $C^v_{j,l,k}$ represent the curvelet transform of the displacement u and v.

For heterogeneous media, following the principle of section on curvelets, the Laplacian of the pressure field u can be decomposed into a sum of curvelets (3 terms). If a depends on x, then we must calculate the following coefficients $a_{j,l,k,m,n,q}$:

$$a_{j,l,k,m,n,q} = \int a^2(x)\varphi_{m,n,q}(x)\varphi_{j,l,k}(x)dx \quad , \quad (45)$$

and the wave equation in the curvelet domain changes to

$$\partial^2 C_{j,l,k}/\partial t^2 = \sum_{m,n,q} \{a_{j,l,k,m,n,q}[(1 + \tan^2\theta_l)(\partial^2 C_{j,l,k}/\partial b_1^2) + (\partial^2 C_{j,l,k}/\partial b_2^2)$$

$$- 2\tan\theta_l(\partial^2 C_{j,l,k}/\partial b_1 \partial b_2]\} \quad . \quad (46)$$

Calculation of eq. (45) would be a key issue in numerical simulation of heterogeneous model. This would be included in our future research.

CONCLUSION

This paper is a first attempt towards solving the wave equation in the curvelet domain. We have derived an expression for homogeneous velocity models. For the extension to more complex models, the expression obtained here of the spatial derivatives in the curvelet domain remains true. More work is however needed to establish the final formulation.

REFERENCES

Alford, R.M., Kelly, K.R. and Boore, D.M., 1974. Accuracy of finite difference modeling of the acoustic wave equation. Geophysics, 39: 834-842.

Alterman, Z. and Karal, F.C., 1968. Propagation of elastic waves in layered media by finite difference methods. Bull. Seismol. Soc. Am, 58: 367-398.

Andersson, F., de Hoop, M.V., Smith, H. and Uhlmann, G., 2003. A multi-scale approach to hyperbolic evolution equations with limited smoothness communications in partial differential equations. C.R. Acad. Sci. Paris, 33: 988-1017.

Antoine, J.P., Murenzi, R., Vandergheynst, P. and Ali, S.T., 2004. Two-dimensional wavelets and their relatives. Cambridge University Press, Cambridge.

Candès, E.J. and Demanet, L., 2005. The curvelet representation of wave propagators is optimally sparse. Comm. Pure Appl. Math., 58: 1472-1528.

Candès, E.J. and Donoho, D.L., 2000, Curvelet a surprisingly effective nonadaptive representation for objects with edges. Curves and Surfaces, Vanderbilt University Press: 105-120.

Candès, E.J. and Donoho, D.L., 2003. Curvelets and Fourier Integral Operators. C.R. Acad. Sci. Paris, 1: 395-398.

Candès, E.J. and Donoho, D.L., 2004. New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities. Comm. Pure Appl. Math., 57: 219-266.

Candès, E.J., Demanet, L., Donoho, D.L. and Ying, L., 2005. Fast discrete curvelet transforms. Multiscale. Model. Simul., 5: 861-899.

Carrer, J., Mansur, W. and Vanzuit, R., 2008. Scalar wave equation by the boundary element method: a D-BEM approach with non-homogeneous initial conditions. Computat. Mechan., 44: 31-44.

Chauris, H. and Nguyen, T.T., 2008. Seismic demigration/migration in the curvelet domain. Geophysics, 73: 35-46.

Daubechies, I., 1992. Ten Lectures on Wavelets. SIAM, Philadelphia.

Demanet, L., 2006. Curvelets, Wave Atoms and Wave Equations. Ph.D. Thesis, California Institute of Technology, Pasadena.

Douma, H. and de Hoop, M.V., 2007. Leading-order seismic imaging using curvelets. Geophysics, 72: 231-248.

Fornberg, B., 1989. Pseudospectral approximation of the elastic wave equation on staggered grid. Expanded Abstr., 59th Ann. Internat. SEG Mtg., Dallas, 8: 1047-1049.

Funato, K. and Fukui, T., 1999. Time domain boundary element method for wave propagation in biot material. Proc. Japan Nat. Symp. Boundary Element Meth., 16: 7-12.

Gazdag, J., 1981. Modeling of the acoustic wave equation with transform methods. Geophysics, 46: 854-859.

Herrmann, F.J., Wang, D., Hennenfent, G. and Moghaddam, P.P., 2008. Curvelet-based seismic data processing: A multiscale and nonlinear approach. Geophysics, 73: A1-A5.

Hong, T.K. and Kennett, B.L., 2002a, A wavelet based method for simulation of two-dimensional elastic wave propagation. Geophys. J. Internat., 150: 610-638.

Hong, T.K. and Kennett, B.L., 2002b. A wavelet based method for the numerical simulation of wave propagation. J. Comput. Phys., 183, 577-622.

Kelly, K., Ward, R. and Treitel, S., 1976. Synthetic seismograms: a finite-difference approach. Geophysics, 41: 2-27.

Kosloff, D. and Baysal, E., 1982. Forward modeling by a Fourier method. Geophysics, 47: 1402-1412.

Lin, T. and Herrmann, F.J., 2007. Compressed wavefield extrapolation. Geophysics, 72: SM77-SM93.

Lysmer, J. and Draker, L., 1972. A finite element method for seismology in methods in computational physics, Vol. 11. Academic Press, New York.

Ma, J., Gang, T. and Hussaini, M., 2007. A refining estimation for adaptive solution of wave equation based on curvelets. Proc. SPIE, San Diego: 67012J.

Ma, J. and Plonka, G., 2007. Combined curvelet shrinkage and nonlinear anisotropic diffusion. IEEE Trasnsact. Image Process., 16: 2198-2206.

Ma, J. and Plonka, G., 2009a. Computing with curvelets: from imaging processing to turbulent flows. IEEE J. Comput. Sci. Eng., 11: 72-80.

Ma, J. and Plonka, G., 2009b. A review of curvelets and recent applications. IEEE Signal Process. Magaz., in press.

Ma, J., Yang, H. and Zhu, Y., 2001. MRFD method for numerical solution of wave propagation in layered media with general boundary condition. Electron. Lett., 37: 1267-1268.

Meyer, Y., 1992. Wavelet and Operators. Cambridge University Press, Cambridge.

Mullen, R. and Belytschko, T., 1982. Dispersion analysis of finite element semidiscretizations of the two-dimensional wave equation. Internat. J. Numerical Methods in Engin., 18: 11-29.