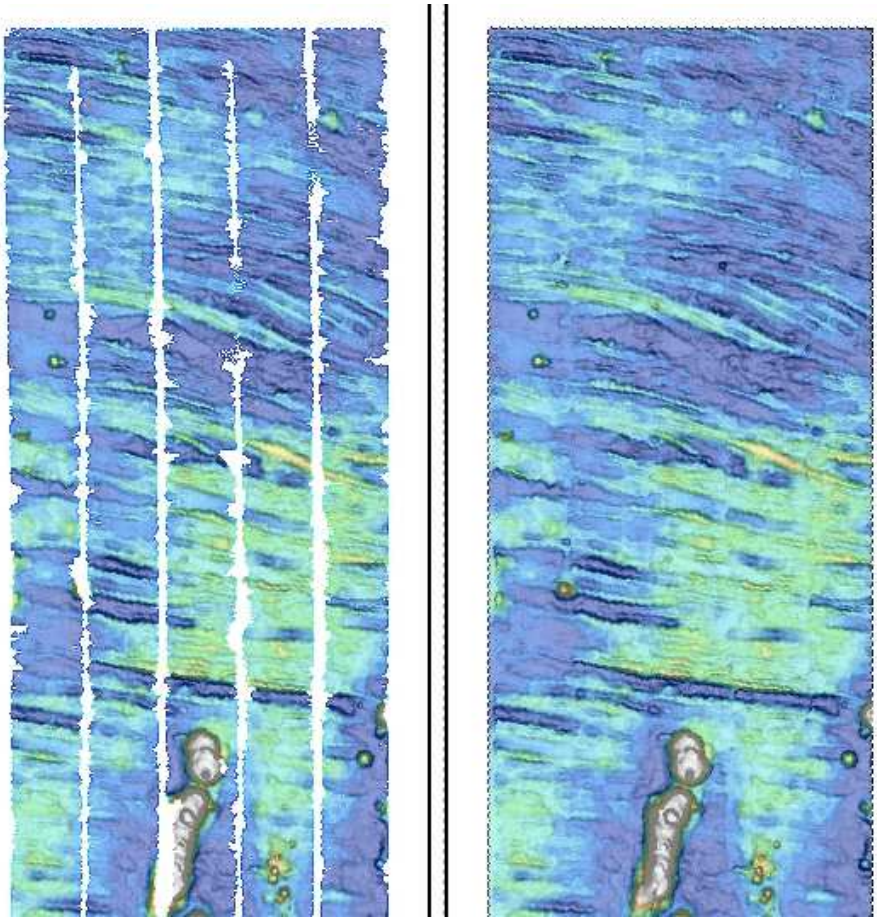
STANFORD EXPLORATION PROJECT

Rustam Akhmadiev, Biondo Biondi, Alejandro Cabrales-Vargas, Christopher Castillo, Robert Clapp, Taylor Dahlke, Huy Le, Stewart Levin, and Siyuan Yuan

Report Number 174, October 2018



Preface

The electronic version of this report[1] makes the included programs and applications available to the reader. The markings [ER], [CR], and [NR] are promises by the author about the reproducibility of each figure result. Reproducibility is a way of organizing computational research that allows both the author and the reader of a publication to verify the reported results. Reproducibility facilitates the transfer of knowledge within SEP and between SEP and its sponsors.

ER denotes Easily Reproducible and are the results of processing described in the paper. The author claims that you can reproduce such a figure from the programs, parameters, and makefiles included in the electronic document. The data must either be included in the electronic distribution, be easily available to all researchers (e.g., SEG-EAGE data sets), or be available in the SEP data library[2]. We assume you have a UNIX workstation with Fortran, Fortran90, C, C++, X-Windows system and the software downloadable from our website (SEP makerules, SEPlib, and the SEP latex package), or other free software such as SU. Before the publication of the electronic document, someone other than the author tests the author's claim by destroying and rebuilding all ER figures. Some ER figures may not be reproducible by outsiders because they depend on data sets that are too large to distribute, or data that we do not have permission to redistribute but are in the SEP data library.

CR denotes Conditional Reproducibility. The author certifies that the commands are in place to reproduce the figure if certain resources are available. The primary reasons for the CR designation is that the processing requires 20 minutes or more, MPI or CUDA based code, or commercial packages such as Matlab or Mathematica.

NR denotes Non-Reproducible figures. SEP discourages authors from flagging their figures as NR except for figures that are used solely for motivation, comparison, or illustration of the theory, such as: artist drawings, scannings, or figures taken from SEP reports not by the authors or from non-SEP publications.

Our testing is currently limited to LINUX 2.6 (using the Intel compiler), but the code should be portable to other architectures. Reader's suggestions are welcome. More information on reproducing SEP's electronic documents is available online[3].

[1] http://sepwww.stanford.edu/private/docs/sep174
[2] http://sepwww.stanford.edu/public/docs/sepdatalib/toc_html
[3] http://sepwww.stanford.edu/research/redoc/

SEP174 — TABLE OF CONTENTS

Waveform inversion

Signal and data processing

Computation

Instability of adjoint pseudo-acoustic anisotropic wave equations

Huy Le, Stewart A. Levin, and Biondo Biondi

ABSTRACT

We discover that the adjoint system of the second-order pseudo-acoustic anisotropic wave equations that we have implemented in previous work has unstable solutions and propose two possible alternative systems. The unstable solutions have magnitude that grows linearly with time and does not propagate spatially. This is not the kind of numerical instability that occurs when the propagation timestep does not satisfy the CFL condition. This instability is inherent to this particular type of wave equations. In fact, there is a family of unstable wave equations that all describe the same kinematics. The key to stability for this type of pseudo-acoustic anisotropic wave equations is not whether the system is self-adjoint but whether it reduces to the scalar wave equation in isotropic media.

INTRODUCTION

The pseudo-acoustic anisotropic wave equations were introduced by Alkhalifah (2000) to avoid having to process shear waves. This system of equations is derived by setting shear wave velocities along symmetry axes to zeros in the dispersion relation. The most computationally efficient form is the system of coupled second-order wave equations, which has been commonly used in reverse time migration (Fletcher et al., 2009; Duveneck and Bakker, 2011; Zhang et al., 2011). There are several properties to know about this kind of pseudo-acoustic anisotropic wave equations.

1. Even though shear wave velocity is set to zero, the solution space still encompasses residual shear waves. These residual shear waves manifest as dimond-shaped artifacts and can be easily reduced when the source injection location is in isotropic media, for example in marine environments (Alkhalifah, 2000). Complete suppression of this artifact requires projection onto a solution subspace (Le and Levin, 2014; Maharramov et al., 2015; Xu and Zhou, 2014).
2. There is actually a family of infinite number of equivalent systems of wave equations that all come from the same dispersion relation (Fowler et al., 2010). These systems similarly describe P-wave kinematics but behave differently in terms of amplitude.
3. All members of this family of pseudo-acoustic wave equations suffer from inherent weak instability (Bube et al., 2012). This instability is a result of setting shear velocity to zero combined with second-order time derivatives.

Bube et al. (2016) show that the instability of this type of second-order pseudo-acoustic wave equations can occur in practice when numerical computation is performed with low

presicion. They also develop a stable and self-adjoint system by reintroducing finite shear wave velocity and rewriting second-order derivatives as cascaded first-orders. We recently find that this instability problem also occurs in the adjoint systems even with high precision numerics. The unstable solutions grow linearly with time and are stationary (i.e. does not propagate spatially), as analyzed by Bube et al. (2012). In this report, we present two alternative systems that do not suffer from this instability. Compared to those proposed by Bube et al. (2016), our solutions are computationally more efficient and do not require medium parameters on different staggered grids.

NON-SELF-ADJOINT SYSTEM

The coupled second-order systems of pseudo-acoustic anisotropic wave equations are commonly used in reverse time migration and waveform inversion due to its computational efficiency and capability to acurately describe P-wave kinematics. One particular form of such system is

$$\begin{cases} \partial_t^2 \sigma_x = c_{11}(\partial_x^2 + \partial_y^2)\sigma_x + c_{13}\partial_z^2 \sigma_z, \\ \partial_t^2 \sigma_z = c_{13}(\partial_x^2 + \partial_y^2)\sigma_x + c_{33}\partial_z^2 \sigma_z, \end{cases} \tag{1}$$

which can be written in matrix form as

$$\partial_t^2 \sigma = CD\sigma, \tag{2}$$

where σ is the stresses, C is the density-normalized stiffness matrix, and D is the derivative matrix

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_z \end{bmatrix}, C = \begin{bmatrix} c_{11} & c_{13} \\ c_{13} & c_{33} \end{bmatrix}, D = \begin{bmatrix} \partial_x^2 + \partial_y^2 & \\ & \partial_z^2 \end{bmatrix}. \tag{3}$$

The elements of stiffness matrix C are related to vertical velocity, v, horizontal velocity, v_x, NMO velocity, v_n, and Thomsen parameters, ϵ and δ by

$$c_{11} = v^2(1 + 2\epsilon) = v_x^2, \tag{4}$$

$$c_{13} = v^2\sqrt{1 + 2\delta} = vv_n, \tag{5}$$

$$c_{33} = v^2. \tag{6}$$

Spatial derivatives on the right hand side of system 1 have form of $c\partial_i^2$, which is not self-adjoint (Appendix A). The adjoint system is

$$\partial_t^2 \lambda = DC\lambda, \tag{7}$$

or

$$\begin{cases} \partial_t^2 \lambda_x = (\partial_x^2 + \partial_y^2)(c_{11}\lambda_x + c_{13}\lambda_z), \\ \partial_t^2 \lambda_z = \partial_z^2(c_{13}\lambda_x + c_{33}\lambda_z), \end{cases} \tag{8}$$

where $\lambda = \begin{bmatrix} \lambda_x \\ \lambda_z \end{bmatrix}$ is the adjoint wave fields.

The gradients of the objective function χ with respect to c_{ij} are

$$\frac{\partial \chi}{\partial c_{11}} = \int_0^T \lambda_x(\partial_x^2 + \partial_y^2)\sigma_x dt, \tag{9}$$

$$\frac{\partial \chi}{\partial c_{13}} = \int_0^T \lambda_x \partial_z^2 \sigma_z + \lambda_z(\partial_x^2 + \partial_y^2)\sigma_x dt, \tag{10}$$

$$\frac{\partial \chi}{\partial c_{33}} = \int_0^T \lambda_z \partial_z^2 \sigma_z dt, \tag{11}$$

from which gradients with respect to (v, ϵ, δ) can be computed from chain rule

$$\frac{\partial \chi}{\partial v} = \frac{\partial \chi}{\partial c_{11}} 2v(1+2\epsilon) + \frac{\partial \chi}{\partial c_{13}} 2v\sqrt{1+2\delta} + \frac{\partial \chi}{\partial c_{33}} 2v, \tag{12}$$

$$\frac{\partial \chi}{\partial \epsilon} = \frac{\partial \chi}{\partial c_{11}} 2v^2, \tag{13}$$

$$\frac{\partial \chi}{\partial \delta} = \frac{\partial \chi}{\partial c_{13}} \frac{v^2}{\sqrt{1+2\delta}}. \tag{14}$$

Unfortunately, the above adjoint equations have solutions that grow linearly with time. The unstable solutions seem to have zero wave speed and does not propagate spatially. This is the kind of weak instability that is inherent to this type of pseudo-acoustic wave equations (Bube et al., 2012). Figure 1 shows the XZ palnes of two adjoint wavefields in a homogeneous isotropic medium. Notice the stationary instability points at injection location. Figure 2 shows the time evolutions of these adjoint wavefields at every second. Interestingly, the two adjoint wavefields have opposite linearly growing solutions. From equations 8, it is easy to see that, for isotropic media ($c_{11} = c_{13} = c_{33}$), if (λ_x, λ_z) is a solution, so is $(\lambda_x + t, \lambda_z - t)$. In fact, in such medium, the adjoint equations do not reduce to the well-known scalar acoustic isotropic wave equation. All simulations are performed in three dimensions.

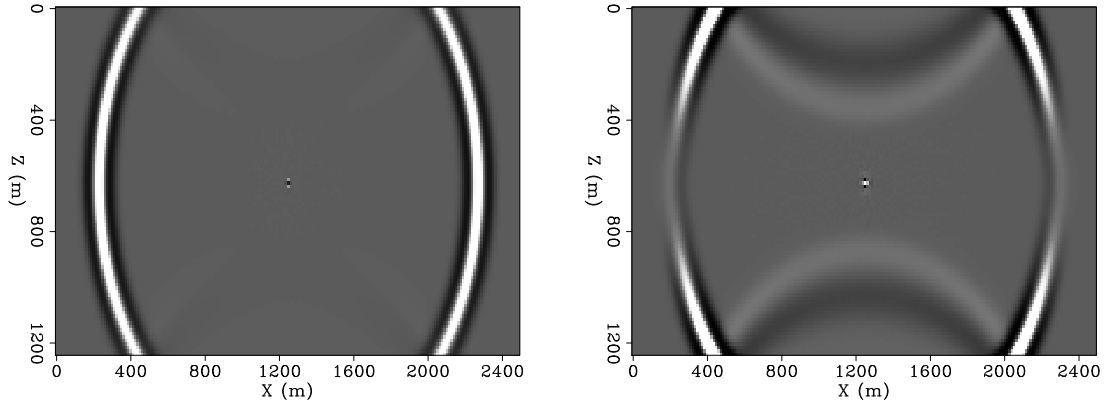


Figure 1: XZ planes of adjoint wavefields λ_x (a) and λ_z (b) at 0.8 seconds show stationary instability at the source locations. [**ER**] huyle/. lambdaX08s,lambdaZ08s

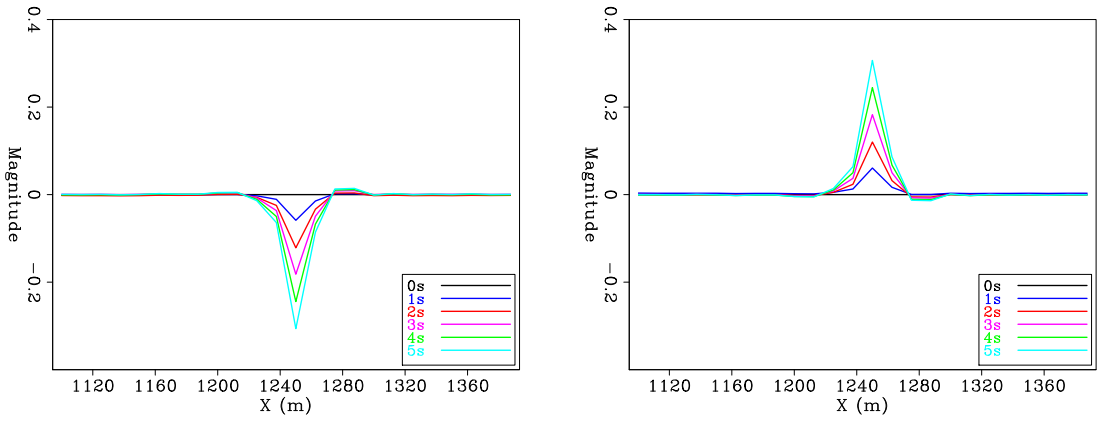


Figure 2: Time evolutions of adjoint wavefields λ_x (a) and λ_z (b) at every second show the linearly growing instability. [**ER**] huyle/. lambdaX,lambdaZ

A SELF-ADJOINT SYSTEM

Bube et al. (2016) modify the forward equations making it self-adjoint in order to achieve stability. One possible way to make the system self-adjoint is to define

$$R = \sqrt{C} = \begin{bmatrix} r_{11} & r_{13} \\ r_{13} & r_{33} \end{bmatrix}, \tag{15}$$

and rewrite the system as

$$\partial_t^2 \sigma = RDR\sigma, \tag{16}$$

or

$$\begin{cases} \partial_t^2 \sigma_x = r_{11}(\partial_x^2 + \partial_y^2)(r_{11}\sigma_x + r_{13}\sigma_z) + r_{13}\partial_z^2(r_{13}\sigma_x + r_{33}\sigma_z), \\ \partial_t^2 \sigma_z = r_{13}(\partial_x^2 + \partial_y^2)(r_{11}\sigma_x + r_{13}\sigma_z) + r_{33}\partial_z^2(r_{13}\sigma_x + r_{33}\sigma_z). \end{cases} \tag{17}$$

Because C is always symmetric and positive semi-definite when $\epsilon \geq \delta$, R exists and is also symmetric and positive semi-definite.

Equations 16 correctly captures the kinematics of the original equations 2. It can be shown that the two systems have the same dispersion relation

$$\det(C\hat{D} + \omega^2 I) = \det(R\hat{D}R + \omega^2 I), \tag{18}$$

where

$$\hat{D} = -\begin{bmatrix} k_x^2 + k_y^2 & \\ & k_z^2 \end{bmatrix}. \tag{19}$$

Moreover, if C is nonsingular, system 2 is equivalent to system 17 by a change of variable $\sigma \to R\sigma$. The adjoint system 8 is also equivalent to this newly derived system by a similar change of variable $\lambda \to R^{-1}\sigma$. Despite this equivalence, system 17 is stable. This is because both equations in this system reduce to the scalar acoustic wave equation in isotropic media. In fact, in isotropic media, the equivalence among these systems break because C is singular.

The gradients of the objective function with respect to r_{ij} are

$$\frac{\partial \chi}{\partial r_{11}} = \int_0^T \left[\lambda_x(\partial_x^2+\partial_y^2)(r_{11}\sigma_x + r_{13}\sigma_z) + \sigma_x(\partial_x^2+\partial_y^2)(r_{11}\lambda_x + r_{13}\lambda_z)\right] dt, \tag{20}$$

$$\begin{aligned}\frac{\partial \chi}{\partial r_{13}} = \int_0^T &\left[\lambda_x\partial_z^2(r_{13}\sigma_x + r_{33}\sigma_z) + \lambda_z(\partial_x^2+\partial_y^2)(r_{11}\sigma_x + r_{13}\sigma_z) + \right.\\ &\left.\sigma_x\partial_z^2(r_{13}\lambda_x + r_{33}\lambda_z) + \sigma_z(\partial_x^2+\partial_y^2)(r_{11}\lambda_x + r_{13}\lambda_z)\right] dt,\end{aligned} \tag{21}$$

$$\frac{\partial \chi}{\partial r_{33}} = \int_0^T \left[\lambda_z\partial_z^2(r_{13}\sigma_x + r_{33}\sigma_z) + \sigma_z\partial_z^2(r_{13}\lambda_x + r_{33}\lambda_z)\right] dt. \tag{22}$$

Elements of R are computed from trace and determinant of C

$$r_{11} = \frac{v}{t}(1+2\epsilon+s), \tag{23}$$

$$r_{13} = \frac{v}{t}\sqrt{1+2\delta}, \tag{24}$$

$$r_{33} = \frac{v}{t}(1+s), \tag{25}$$

where

$$s = \sqrt{2(\epsilon-\delta)}, t = \sqrt{2(\epsilon+1)+2s}. \tag{26}$$

Since $s = \sqrt{2(\epsilon-\delta)}$ can be zero, derivatives of s with respect to either ϵ or δ can become unbounded. As a result, one has to define new variables

$$\alpha = 1+s, \tag{27}$$

$$\beta = \sqrt{1+2\delta}. \tag{28}$$

so that

$$r_{11} = \frac{v}{t}(\beta^2+\alpha^2-\alpha), \tag{29}$$

$$r_{13} = \frac{v\beta}{t}, \tag{30}$$

$$r_{33} = \frac{v\alpha}{t}, \tag{31}$$

with

$$t = \sqrt{\alpha^2+\beta^2}. \tag{32}$$

Now the gradients with respect to (v,α,β) can be computed from change rule

$$\frac{\partial \chi}{\partial v} = \frac{\partial \chi}{\partial r_{11}}\frac{\beta^2+\alpha^2-\alpha}{t} + \frac{\partial \chi}{\partial r_{13}}\frac{\beta}{t} + \frac{\partial \chi}{\partial r_{33}}\frac{\alpha}{t}, \tag{33}$$

$$\frac{\partial \chi}{\partial \alpha} = \frac{\partial \chi}{\partial r_{11}}\frac{v(\alpha^3+\alpha\beta^2-\beta^3)}{t^3} - \frac{\partial \chi}{\partial r_{13}}\frac{v\alpha\beta}{t^3} + \frac{\partial \chi}{\partial r_{33}}\frac{v\beta^2}{t^3}, \tag{34}$$

$$\frac{\partial \chi}{\partial \beta} = \frac{\partial \chi}{\partial r_{11}}\frac{v\beta(\alpha^2+\alpha+\beta^2)}{t^3} + \frac{\partial \chi}{\partial r_{13}}\frac{v\alpha^2}{t^3} - \frac{\partial \chi}{\partial r_{33}}\frac{v\alpha\beta}{t^3}. \tag{35}$$

After the inversion, (ϵ,δ) can be obtained by

$$\delta = \frac{\beta^2-1}{2}, \tag{36}$$

$$\epsilon = \frac{(\alpha-1)^2}{2} + \delta. \tag{37}$$

Using system 17 for waveform inversion is computationally inefficient due to expensive forward solutions and cumbersome gradient expressions involving a change of variables. Next section seeks an alternative solution.

ANOTHER SELF-ADJOINT SYSTEM

Another way to make system 2 self-adjoint is to split the differential operator D instead of the medium matrix C. However, taking the square root of the horizontal derivative, $\partial_x^2 + \partial_y^2$, which results in a pseudo-differential operator, requires either Fourier transform or spectral factorization and helix coordinates (Claerbout and Fomel, 2014). Instead, we expand system 1 to three equations

$$\begin{cases} \partial_t^2 \sigma_x = c_{11}\partial_x^2 \sigma_x + c_{11}\partial_y^2 \sigma_y + c_{13}\partial_z^2 \sigma_z, \\ \partial_t^2 \sigma_y = c_{11}\partial_x^2 \sigma_x + c_{11}\partial_y^2 \sigma_y + c_{13}\partial_z^2 \sigma_z, \\ \partial_t^2 \sigma_z = c_{13}\partial_x^2 \sigma_x + c_{13}\partial_y^2 \sigma_y + c_{33}\partial_z^2 \sigma_z, \end{cases} \tag{38}$$

which has the same matrix form as system 2, but with

$$\sigma = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{bmatrix}, C = \begin{bmatrix} c_{11} & c_{11} & c_{13} \\ c_{11} & c_{11} & c_{13} \\ c_{13} & c_{13} & c_{33} \end{bmatrix}, D = \begin{bmatrix} \partial_x^2 & & \\ & \partial_y^2 & \\ & & \partial_z^2 \end{bmatrix}. \tag{39}$$

The adjoint equations of system 38 are

$$\begin{cases} \partial_t^2 \lambda_x = \partial_x^2 (c_{11}\lambda_x + c_{11}\lambda_y + c_{13}\lambda_z), \\ \partial_t^2 \lambda_y = \partial_y^2 (c_{11}\lambda_x + c_{11}\lambda_y + c_{13}\lambda_z), \\ \partial_t^2 \lambda_z = \partial_z^2 (c_{13}\lambda_x + c_{13}\lambda_y + c_{33}\lambda_z). \end{cases} \tag{40}$$

This adjoint system suffers the same instability as system 8.

Now to make it self-adjoint, define

$$D_1 = \sqrt{D} = \begin{bmatrix} \partial_x & & \\ & \partial_y & \\ & & \partial_z \end{bmatrix}, \tag{41}$$

and rewrite the system 38 as

$$\partial_t^2 \sigma = D_1 C D_1 \sigma, \tag{42}$$

or

$$\begin{cases} \partial_t^2 \sigma_x = \partial_x (c_{11}\partial_x \sigma_x + c_{11}\partial_y \sigma_y + c_{13}\partial_z \sigma_z), \\ \partial_t^2 \sigma_y = \partial_y (c_{11}\partial_x \sigma_x + c_{11}\partial_y \sigma_y + c_{13}\partial_z \sigma_z), \\ \partial_t^2 \sigma_z = \partial_z (c_{13}\partial_x \sigma_x + c_{13}\partial_y \sigma_y + c_{33}\partial_z \sigma_z). \end{cases} \tag{43}$$

It can be verified that systems 2 and 42 have the same dispersion relation, i.e. they describe the same kinematics

$$\det(C\hat{D} + \omega^2 I) = \det(\hat{D}_1 C \hat{D}_1 + \omega^2 I), \tag{44}$$

where

$$\hat{D} = -\begin{bmatrix} k_x^2 & & \\ & k_y^2 & \\ & & k_z^2 \end{bmatrix}, \hat{D}_1 = \begin{bmatrix} ik_x & & \\ & ik_y & \\ & & ik_z \end{bmatrix}. \tag{45}$$

Moreover, if one applies D_1 on both sides of system 38 and make a change of variables $\sigma' = D_1\sigma$, they would arrive at system 43. This means that if $(\sigma_x, \sigma_x, \sigma_z)$ is a smooth solution of system 38, $(\partial_x\sigma_x, \partial_y\sigma_x, \partial_z\sigma_z)$ is a solution of system 43.

The gradients of the objective function with respect to c_{ij} now are

$$\frac{\partial\chi}{\partial c_{11}} = \int_0^T (\partial_x\lambda_x + \partial_y\lambda_y)(\partial_x\sigma_x + \partial_y\sigma_y)dt, \tag{46}$$

$$\frac{\partial\chi}{\partial c_{13}} = \int_0^T (\partial_x\lambda_x + \partial_y\lambda_y)\partial_z\sigma_z + (\partial_x\sigma_x + \partial_y\sigma_y)\partial_z\lambda_z dt, \tag{47}$$

$$\frac{\partial\chi}{\partial c_{33}} = \int_0^T \partial_z\lambda_z\partial_z\sigma_z dt, \tag{48}$$

from which gradients with respect to (v, ϵ, δ) can be computed from chain rule as before (equations 12, 13, and 14).

Solving system 43 requires staggered grid finite difference methods. Figure 3 shows grid locations of wavefield variables and medium parameters. Advantages of this system include efficient forward solution, simple expressions for gradients, and co-location of medium parameters. Numerial experiments, however, show that even though system 43 is self-adjoint, it is unstable. Indeed, if one integrates system 40, which is unstable, one can prove that $(\sigma_x, \sigma_y, \sigma_z) = (\int_x \lambda_x, \int_y \lambda_y, \int_z \lambda_z)$ is also a solution of system 38.

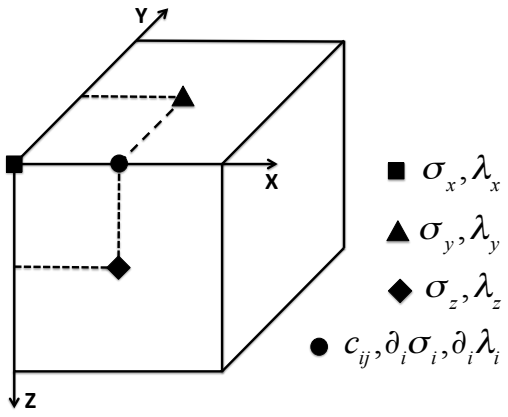


Figure 3: Staggered grids for system 38. [**NR**] huyle/. grid

NON-SELF-ADJOINT BUT STABLE SYSTEM

Fowler et al. (2010) show that system 1 is a member of a family of infinitely many kinematically equivalent wave equations. This section attempts to find a stable system in this family. The general form of this family of equations is

$$\begin{cases} \partial_t^2\sigma_x = \left[a_1(\partial_x^2 + \partial_y^2) + a_2\partial_z^2\right]\sigma_x + \left[b_1(\partial_x^2 + \partial_y^2) + b_2\partial_z^2\right]\sigma_z, \\ \partial_t^2\sigma_z = \left[c_1(\partial_x^2 + \partial_y^2) + c_2\partial_z^2\right]\sigma_x + \left[d_1(\partial_x^2 + \partial_y^2) + d_2\partial_z^2\right]\sigma_z, \end{cases} \tag{49}$$

or

$$\partial_t^2\sigma = \begin{bmatrix} A & B \\ C & D \end{bmatrix}\sigma, \tag{50}$$

where $A = a_1(\partial_x^2 + \partial_y^2) + a_2\partial_z^2$ and so on. The adjoint equations are

$$\partial_t^2 \lambda = \begin{bmatrix} A^T & C^T \\ B^T & D^T \end{bmatrix} \lambda, \tag{51}$$

or

$$\begin{cases} \partial_t^2 \lambda_x = (\partial_x^2 + \partial_y^2)(a_1\lambda_x + c_1\lambda_z) + \partial_z^2(a_2\lambda_x + c_2\lambda_z), \\ \partial_t^2 \lambda_z = (\partial_x^2 + \partial_y^2)(b_1\lambda_x + d_1\lambda_z) + \partial_z^2(b_2\lambda_x + d_2\lambda_z). \end{cases} \tag{52}$$

Derivatives with respect to these coefficients (a_i, b_i, c_i, d_i) are

$$\frac{\partial \chi}{\partial a_1} = \int_0^T \lambda_x(\partial_x^2 + \partial_y^2)\sigma_x dt, \tag{53}$$

$$\frac{\partial \chi}{\partial a_2} = \int_0^T \lambda_x \partial_z^2 \sigma_x dt, \tag{54}$$

$$\frac{\partial \chi}{\partial b_1} = \int_0^T \lambda_x(\partial_x^2 + \partial_y^2)\sigma_z dt, \tag{55}$$

$$\frac{\partial \chi}{\partial b_2} = \int_0^T \lambda_x \partial_z^2 \sigma_z dt, \tag{56}$$

$$\frac{\partial \chi}{\partial c_1} = \int_0^T \lambda_z(\partial_x^2 + \partial_y^2)\sigma_x dt, \tag{57}$$

$$\frac{\partial \chi}{\partial c_2} = \int_0^T \lambda_z \partial_z^2 \sigma_x dt, \tag{58}$$

$$\frac{\partial \chi}{\partial d_1} = \int_0^T \lambda_z(\partial_x^2 + \partial_y^2)\sigma_z dt, \tag{59}$$

$$\frac{\partial \chi}{\partial d_2} = \int_0^T \lambda_z \partial_z^2 \sigma_z dt. \tag{60}$$

For the this system to correctly describe the kinematics, its dispersion relation has to be equal to that of system 2, which results in five constraints on eight medium parameters (a_i, b_i, c_i, d_i)

$$a_1 + d_1 = c_{11}, \tag{61}$$

$$a_2 + d_2 = c_{33}, \tag{62}$$

$$a_1 d_1 - b_1 c_1 = 0, \tag{63}$$

$$a_2 d_2 - b_2 c_2 = 0, \tag{64}$$

$$a_1 d_2 + a_2 d_1 - b_1 c_2 - b_2 c_1 = c_{11}c_{33} - c_{13}^2. \tag{65}$$

These constraints form an under-determined system. There are infinitely many equivalent solutions. Appendix B systematically shows that members of this family of equations with have zero parameter coeffficients are all unstable. There are still multiple solutions with all non-zero coefficients to consider. To simplify algebra and latter computation, one choice is to have symmetric operators

$$b_1 = c_1 = \frac{1}{2} r v_x^2, \tag{66}$$

$$b_2 = c_2 = \frac{1}{2} r v^2. \tag{67}$$

Other coefficients can be expressed in terms of (r, v_x, v) as

$$(a_1, d_1) = \frac{1}{2} v_x^2 \left(1 \pm \sqrt{1 - r^2}\right), \tag{68}$$

$$(a_2, d_2) = \frac{1}{2} v^2 \left(1 \mp \sqrt{1 - r^2}\right). \tag{69}$$

Substitute the above expressions into the last constraint 65 and solve for

$$r = \frac{v_n}{v_x}. \tag{70}$$

To compute the gradients, define new variables

$$\alpha = \sqrt{2(\epsilon - \delta)}, \tag{71}$$

$$\beta = \sqrt{1 + 2\epsilon}, \tag{72}$$

$$\gamma = \sqrt{\beta^2 - \alpha^2}, \tag{73}$$

so that

$$1 + 2\delta = \gamma^2, v_x = v\beta, v_n = v\gamma, r = \frac{\gamma}{\beta}, \sqrt{1 - r^2} = \frac{\alpha}{\beta}, \tag{74}$$

and

$$(a_1, d_1) = \frac{1}{2} v^2 \beta \left(\beta \pm \alpha\right), \tag{75}$$

$$(a_2, d_2) = \frac{1}{2} v^2 \left(1 \mp \frac{\alpha}{\beta}\right), \tag{76}$$

$$b_1 = c_1 = \frac{1}{2} v^2 \beta\gamma, \tag{77}$$

$$b_2 = c_2 = \frac{1}{2} v^2 \frac{\gamma}{\beta}. \tag{78}$$

Now derivatives with respect to (v, α, β) can be computed from chain rule

$$\begin{aligned} \frac{\partial \chi}{\partial v} = v \Bigg\{ & \frac{\partial \chi}{\partial a_1} \beta(\beta + \alpha) + \frac{\partial \chi}{\partial d_1} \beta(\beta - \alpha) + \frac{\partial \chi}{\partial a_2} \left(1 - \frac{\alpha}{\beta}\right) + \frac{\partial \chi}{\partial d_2} \left(1 + \frac{\alpha}{\beta}\right) \\ & + \gamma \left[\left(\frac{\partial \chi}{\partial b_1} + \frac{\partial \chi}{\partial c_1} \right) \beta + \left(\frac{\partial \chi}{\partial b_2} + \frac{\partial \chi}{\partial c_2} \right) \frac{1}{\beta} \right] \Bigg\}, \end{aligned} \tag{79}$$

$$\begin{aligned} \frac{\partial \chi}{\partial \alpha} = \frac{1}{2} v^2 \Bigg\{ & \left(\frac{\partial \chi}{\partial a_1} - \frac{\partial \chi}{\partial d_1} \right) \beta - \left(\frac{\partial \chi}{\partial a_2} - \frac{\partial \chi}{\partial d_2} \right) \frac{1}{\beta} \\ & - \frac{\alpha}{\gamma} \left[\left(\frac{\partial \chi}{\partial b_1} + \frac{\partial \chi}{\partial c_1} \right) \beta + \left(\frac{\partial \chi}{\partial b_2} + \frac{\partial \chi}{\partial c_2} \right) \frac{1}{\beta} \right] \Bigg\}, \end{aligned} \tag{80}$$

$$\begin{aligned} \frac{\partial \chi}{\partial \beta} = \frac{1}{2} v^2 \Bigg\{ & \left(\frac{\partial \chi}{\partial a_1} - \frac{\partial \chi}{\partial d_1} \right) (2\beta + \alpha) + \left(\frac{\partial \chi}{\partial a_2} - \frac{\partial \chi}{\partial d_2} \right) \frac{\alpha}{\beta^2} \\ & - \frac{1}{\gamma} \left[\left(\frac{\partial \chi}{\partial b_1} + \frac{\partial \chi}{\partial c_1} \right) (2\beta^2 - \alpha^2) + \left(\frac{\partial \chi}{\partial b_2} + \frac{\partial \chi}{\partial c_2} \right) \frac{\alpha^2}{\beta^2} \right] \Bigg\}, \end{aligned} \tag{81}$$

After the inversion, (ϵ, δ) can be retrieved from (α, β) by

$$\epsilon = \frac{\beta^2 - 1}{2}, \tag{82}$$

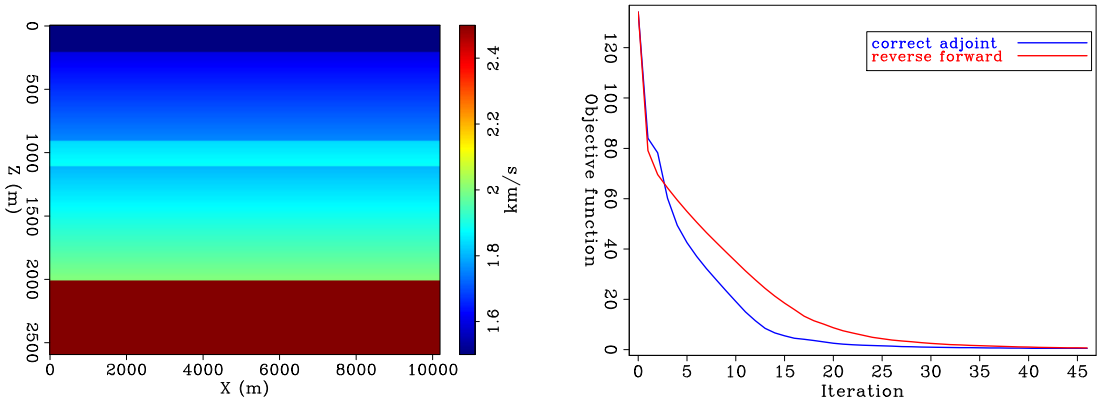
$$\delta = \epsilon - \frac{\alpha^2}{2}. \tag{83}$$

As with our stable self-adjoint system 17, using system 49 involves a change of variables and cumbersome expressions for the gradients, but the forward equations are computationally more efficient.

CORRECT ADJOINT AND TIME-REVERSED FORWARD

Both of the above proposed solutions (equations 17 and 49) are significantly more expensive than the original adjoint equations 8. Instead of using the correct adjoint equations, one might time-reverse the forward equations to compute the adjoint wavefields. In this section, we study how incorrect adjoints affect the convergence of FWI. In the two following 2D synthetic examples, we have uniformly distributed sources and receivers on the water surface and use a Ricker wavelet of 5 Hz central frequency to model our systhetic data. We use a LBFGS solver and invert simutaneously for velocity, v, and Thomsen parameters, ϵ and δ. We do not concern with parameterization and trade-off in these experiments.

Figure 4(a) shows a simple velocity model (ϵ and δ models are similar). For this model, we try to recover the high velocity layer at 1000 meter depth. Figure 4(b) plots the objective functions of two inversions with correct adjoint and time-reversed forward. Figures 5(a) and 5(b) respectively show the inverted velocity models after 50 iterations using the correct adjoint equations and time-reversed forward equations. The instability of the correct adjoint equations does not affect our inversion because it is stationary at injection locations (receivers, for adjoint equations). These figures show that upon convergence, the incorrect adjoint performs just as well as the correct one. Figures 6(a) and 6(b) show the inverted models after only 12 iterations, at which the two objective functions are most different. As one expects, the correct adjoint gives a slightly better result. Although, the two inversions carry out the same number of objective function and gradient evaluations per iteration, a more truly cost comparison is between correct adjoint iteration 12^{th} (Figure 6(a)) and time-reversed forward iteration 18^{th} (Figure 6(c)) because solving the correct adjoint equations is about 1.5 times more expensive than reversing the forward equations. In this metric, using the time-reversed forward for adjoint wavefields actually leads to a better result.



(a) (b)

Figure 4: Simple velocity model (a) and objective functions from two inversions (b) using the correct adjoint and time-reversed forward. The initial model does not include the high velocity layer at 1000 m depth. [**CR**] huyle/. v0,objfunc0

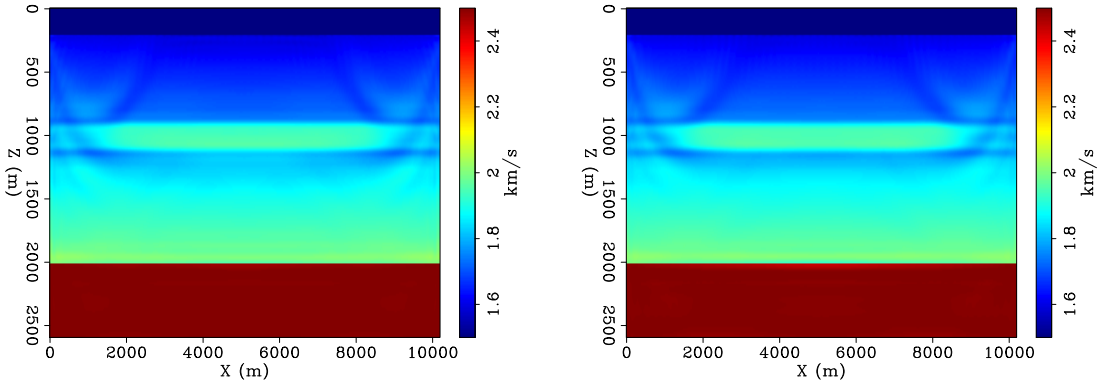


Figure 5: Inverted velocity models after 50 iterations using the correct adjoint (a) and time-reversed forward (b). [**CR**] huyle/. v0iter50,v0aiter50

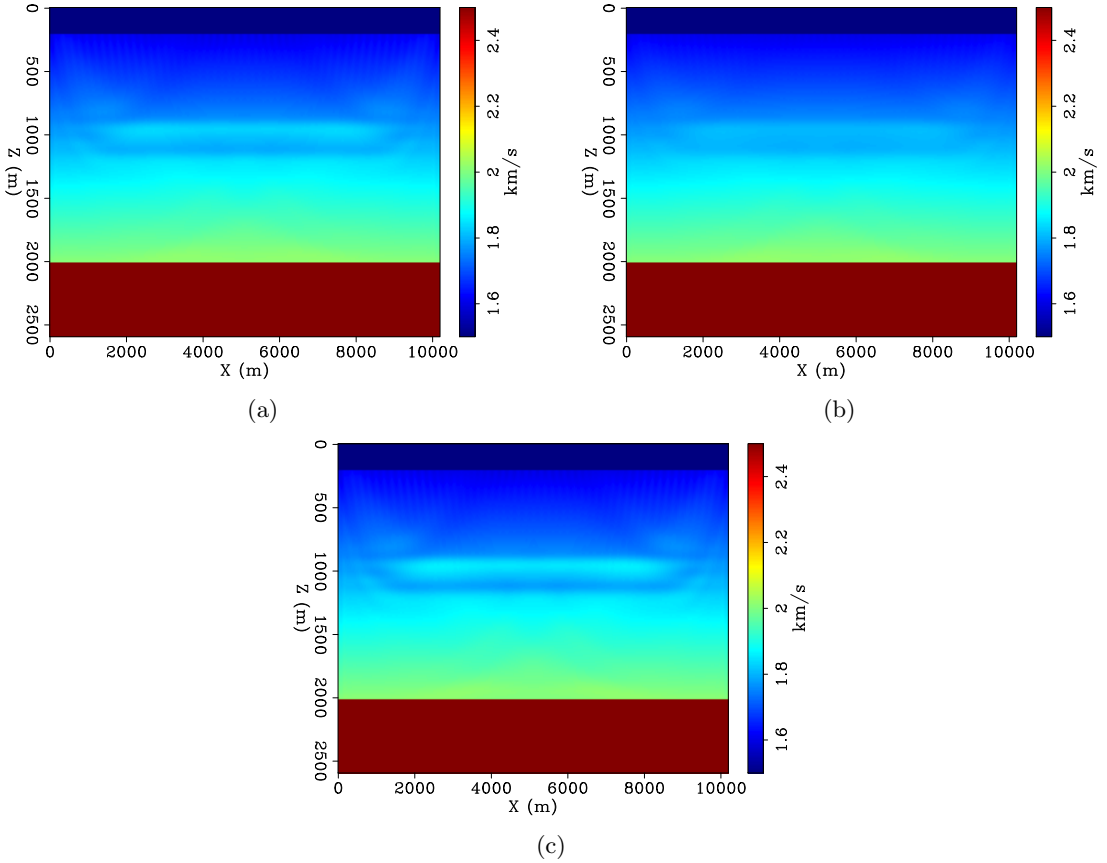


Figure 6: Inverted velocity models after 12 iterations using the correct adjoint (a) and time-reversed forward (b). (c) shows the inverted velocity model after 18 iterations using the time-reversed forward. True cost comparison is between (a) and (c). [**CR**] huyle/. v0iter12,v0aiter12,v0aiter18

In another experiment to understand the effect of incorrect adjoint equations, we use BP 2007 synthetic anisotropic models. Figures 7 show the true and initial velocity models. Figures 8(a) and 8(b) show the inverted models ater 100 iterations with the correct adjoint and time-reversed forward. We again observe that the two methods perform equally well if converged. This can also be seen in Figure 8(c) plotting the objective functions. For such complicated model as this one, the difference between correct and incorrect adjoints at early iterations are less significant. Figures 9(a) and 9(b) show the inverted models after 10 iterations and Figure 9(c) shows the inverted model with incorrect adjoint after 15 iterations. These three results are virtually indistinguishable.

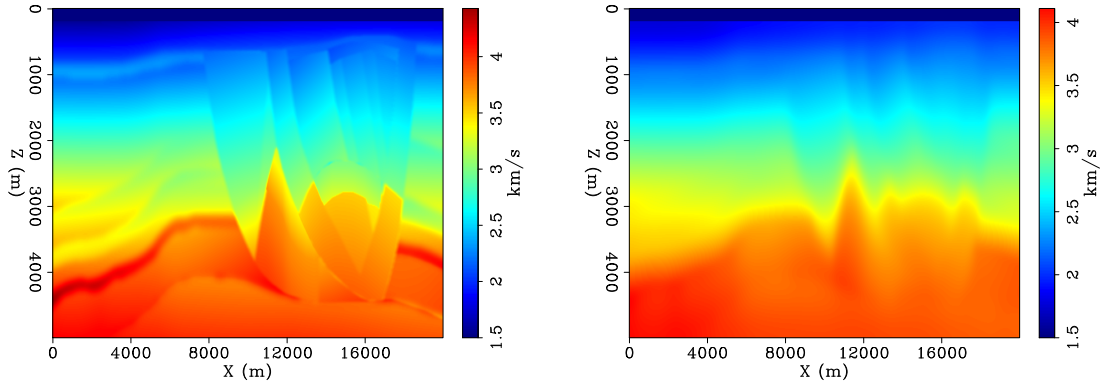


Figure 7: True (a) and initial (b) velocity models from BP2007 synthetic. [**CR**] huyle/. vBP,bgvBP

CONCLUSIONS

The second-order pseudo-acoustic anisotropic wave equations can have unstable adjoint solutions with linearly growing magnitude and zero wave speed. This kind of weak instability is an inherent property of this particular type of wave equations. We propose two alternative systems of equations that are kinematically equivalent and free of this instability. The first system is self-adjoint and obtained by splitting the medium parameter matrix while the second system is a generalization of the original system with all non-zero coefficients. To use in waveform inversion, both of these systems involve a change of variables and are computationally more expensive than the original system. Instead, one can use the time-reversed forward wave equations to compute the gradients. Our numerical experiments indicate that in terms of convergence, the difference between the correct and incorrect adjoints is minimal.

APPENDIX A: ADJOINTS OF COMMON DERIVATIVES

Define functional inner product as

$$\langle u, v \rangle = \int_0^T \int_\Omega uv d\mathbf{x} dt. \tag{A-1}$$

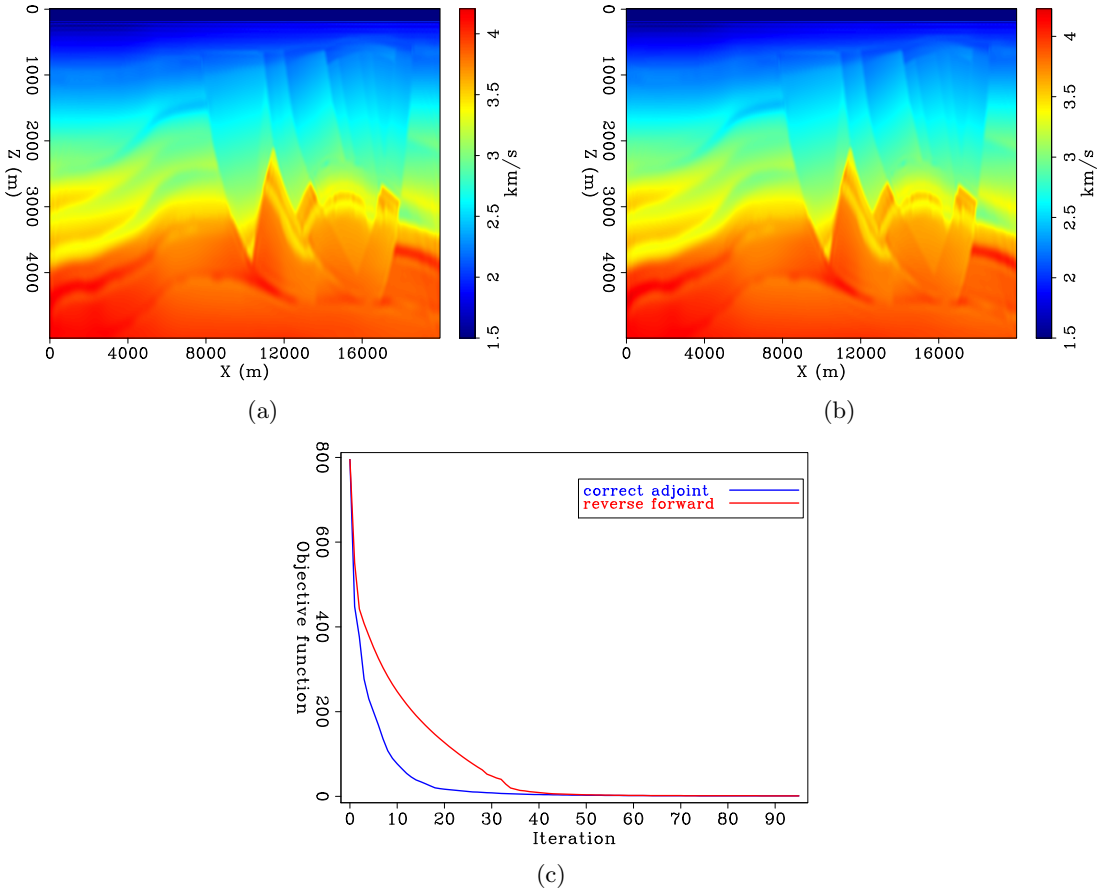


Figure 8: Inverted velocity models after 100 iterations using the correct adjoint (a) and time-reversed forward (b). Objective functions of the two inversions (c). [**CR**] huyle/. vBPiter100,vBPaiter100,objfuncBP

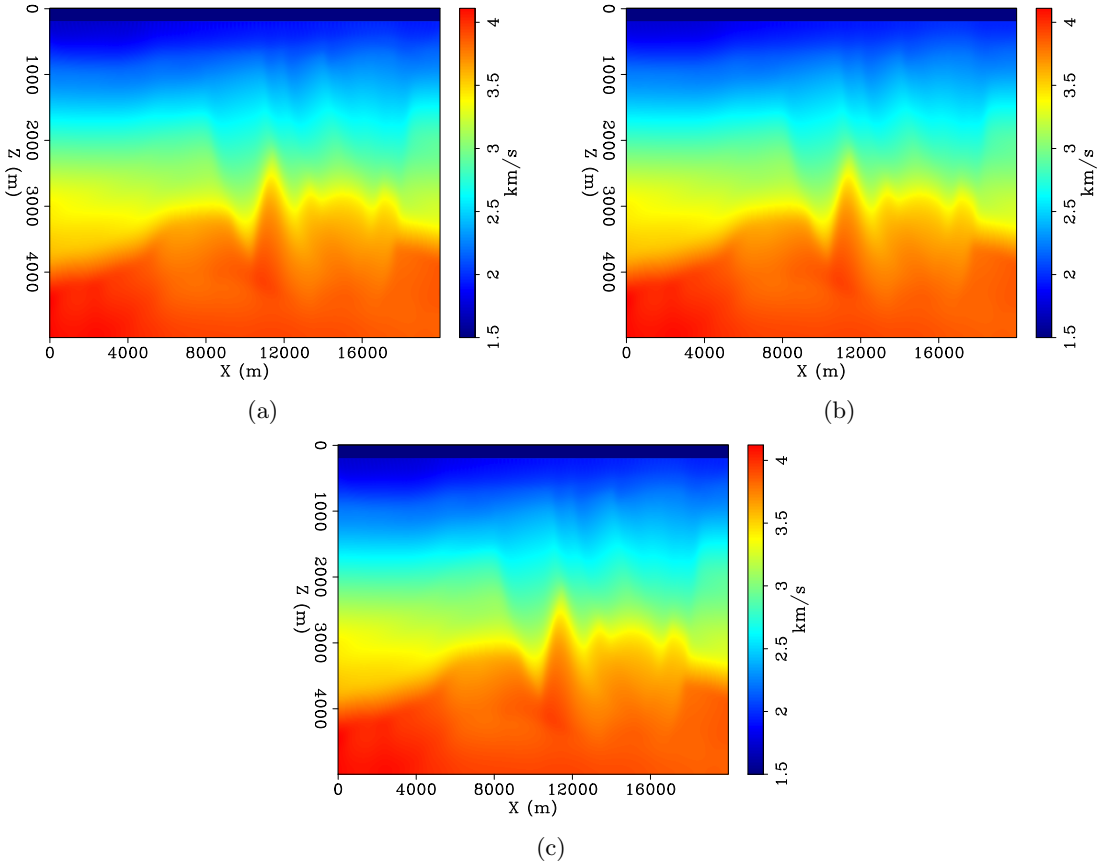


Figure 9: Inverted velocity models after 10 iterations using the correct adjoint (a) and time-reversed forward (b). (c) shows the inverted velocity model after 15 iterations using the time-reversed forward. True cost comparison is between (a) and (c). [**CR**] huyle/. vBPiter10,vBPaiter10,vBPaiter15

Adjoint operator is defined by

$$\langle u, Lv\rangle = \langle L^* u, v\rangle. \tag{A-2}$$

For a particular form of $L = \partial_x^2$

$$\langle u, Lv\rangle = \langle u, \partial_x^2 v\rangle = \int_0^T \int_\Omega u\partial_x^2 v d\mathbf{x}dt \tag{A-3}$$

$$= \int_0^T \int_{\Omega_{yz}} u\partial_x v dydzdt|_{\Omega_x} - \int_0^T \int_{\Omega_{yz}} v\partial_x u dydzdt|_{\Omega_x} + \int_0^T \int_\Omega v\partial_x^2 u d\mathbf{x}dt \tag{A-4}$$

$$= \int_0^T \int_\Omega v\partial_x^2 u d\mathbf{x}dt = \langle \partial_x^2 u, v\rangle. \tag{A-5}$$

where integration by parts has been carried out twice in the x-direction and boundary conditions $u|_{\Omega_x} = v|_{\Omega_x} = 0$. So by definition, $L = \partial_x^2$ is self-adjoint $L^* = \partial_x^2$.

Similarly, for $L = c\partial_x^2$, where c can be velocity or any medium parameter,

$$\langle u, Lv\rangle = \langle u, c\partial_x^2 v\rangle = \langle cu, \partial_x^2 v\rangle = \langle \partial_x^2 cu, v\rangle. \tag{A-6}$$

As a result, this operator is not self-adjoint $L^* = \partial_x^2 c$.

One can easily find the adjoints of the following common differential operators

L	L^*	Self-adjoint
∂_i^2	∂_i^2	yes
$c\partial_i^2$	$\partial_i^2 c$	no
$a\partial_i^2 b$	$b\partial_i^2 a$	no
∂_i	$-\partial_i$	no
$a\partial_i b$	$-b\partial_i a$	no
$\partial_i c\partial_j$	$\partial_j c\partial_i$	no

APPENDIX B: MEMBER SYSTEMS WITH ZERO COEFFICIENTS

Because of constraint equations 63, $a_1 d_1 = b_1 c_1$, and 64, $a_2 d_2 = b_2 c_2$, if one of the coefficients, a_i, b_i, c_i, d_i, is zero, at least one other must also vanish.

Case 1: $a_1 = b_1 = 0$

System 49 becomes

$$\begin{cases} \partial_t^2 \sigma_x = a_2 \partial_z^2 \sigma_x + b_2 \partial_z^2 \sigma_z, \\ \partial_t^2 \sigma_z = \left[c_1(\partial_x^2 + \partial_y^2) + c_2\partial_z^2\right]\sigma_x + \left[d_1(\partial_x^2 + \partial_y^2) + d_2\partial_z^2\right]\sigma_z. \end{cases} \tag{B-1}$$

Case 1.1: $a_2 b_2 \neq 0$ and $c_2 d_2 \neq 0$

Because of equation 64, define $r = \frac{a_2}{c_2} = \frac{b_2}{d_2}$. Because $r \neq 0$, multiply the second equation in B-1 by r and subtract the first equation, resulting in an equivalent system

$$\begin{cases} \partial_t^2 \sigma_x = a_2 \partial_z^2 \sigma_x + b_2 \partial_z^2 \sigma_z, \\ \partial_t^2 (r\sigma_z - \sigma_x) = rc_1(\partial_x^2 + \partial_y^2)\sigma_x + rd_1(\partial_x^2 + \partial_y^2)\sigma_z, \end{cases} \tag{B-2}$$

which, after a change of variable $\sigma'_z = r\sigma_z - \sigma_x$ or $\sigma_z = \frac{\sigma'_z + \sigma_x}{r}$, becomes

$$\begin{cases} \partial_t^2 \sigma_x = (a_2 + \frac{b_2}{r})\partial_z^2 \sigma_x + \frac{b_2}{r}\partial_z^2 \sigma'_z, \\ \partial_t^2 \sigma'_z = (rc_1 + d_1)(\partial_x^2 + \partial_y^2)\sigma_x + d_1(\partial_x^2 + \partial_y^2)\sigma'_z. \end{cases} \tag{B-3}$$

This system is a special case of the general system 49 with $a_1 = b_1 = c_2 = d_2 = 0$.

Case 1.2: $a_2 b_2 \neq 0$ **and** $c_2 d_2 = 0$

Constraint 64 dictates that $c_2 = d_2 = 0$, which leads us back to the above case of $a_1 = b_1 = c_2 = d_2 = 0$.

Case 1.3: $a_2 b_2 = 0$ **and** $c_2 d_2 \neq 0$

In this case, $a_2 = b_2 = 0$, resulting in no solution because constraint 65 is violated.

Case 1.4: $a_2 b_2 = c_2 d_2 = 0$

This results in four cases

- $a_2 = c_2 = 0$: this case has a solution.
- $a_2 = d_2 = 0$: violation of constraint 62.
- $b_2 = c_2 = 0$: the five constraints reduce to

$$d_1 = c_{11}, \tag{B-4}$$
$$a_2 + d_2 = c_{33}, \tag{B-5}$$
$$a_2 d_2 = 0, \tag{B-6}$$
$$a_2 d_1 = c_{11} c_{33} - c_{13}^2. \tag{B-7}$$

  The last constraint requires that $a_2 \neq 0$, so $d_2 = 0$, which, means $a_2 = c_{33}$. This does not satisfy the last constraints.

- $b_2 = d_2 = 0$: the five constraints reduce to

$$d_1 = c_{11}, \tag{B-8}$$
$$a_2 = c_{33}, \tag{B-9}$$
$$a_2 d_1 = c_{11} c_{33} - c_{13}^2, \tag{B-10}$$

  in which the last constraint is not satisfied.

In conclusion, for the case $a_1 = b_1 = 0$, there are two solutions $a_1 = b_1 = c_2 = d_2 = 0$ and $a_1 = b_1 = a_2 = c_2 = 0$. The first solution results in

$$a_2 = c_{33}, \tag{B-11}$$
$$d_1 = c_{11}, \tag{B-12}$$
$$b_2 c_1 = c_{13}^2. \tag{B-13}$$

If one chooses $b_2 = c_1 = c_{13}$ for example, one gets the system

$$\begin{cases} \partial_t^2 \sigma_x = c_{33}\partial_z^2 \sigma_x + c_{13}\partial_z^2 \sigma_z, \\ \partial_t^2 \sigma_z = c_{13}(\partial_x^2 + \partial_y^2)\sigma_x + c_{11}(\partial_x^2 + \partial_y^2)\sigma_z. \end{cases} \tag{B-14}$$

The second solution results in

$$d_1 = c_{11}, \tag{B-15}$$

$$d_2 = c_{33}, \tag{B-16}$$

$$b_2 c_1 = c_{13}^2 - c_{11}c_{33} = v^4(\delta - \epsilon). \tag{B-17}$$

If one chooses $b_2 = v^2(\delta - \epsilon)$ and $c_1 = v^2$ for example, one gets the system

$$\begin{cases} \partial_t^2 \sigma_x = v^2(\delta - \epsilon)\partial_z^2 \sigma_z, \\ \partial_t^2 \sigma_z = v^2(\partial_x^2 + \partial_y^2)\sigma_x + \left[v^2(1+2\epsilon)(\partial_x^2 + \partial_y^2) + v^2\partial_z^2\right]\sigma_z. \end{cases} \tag{B-18}$$

Exchanging b_2 and c_1 results in

$$\begin{cases} \partial_t^2 \sigma_x = v^2\partial_z^2 \sigma_z, \\ \partial_t^2 \sigma_z = v^2(\delta - \epsilon)(\partial_x^2 + \partial_y^2)\sigma_x + \left[v^2(1+2\epsilon)(\partial_x^2 + \partial_y^2) + v^2\partial_z^2\right]\sigma_z. \end{cases} \tag{B-19}$$

Case 2: $a_1 = c_1 = 0$

This case is similar to the case $b_1 = d_1 = 0$, considered below, if one interchanges $\sigma_x \leftrightarrow \sigma_z$.

Case 3: $b_1 = d_1 = 0$

System 49 becomes

$$\begin{cases} \partial_t^2 \sigma_x = \left[a_1(\partial_x^2 + \partial_y^2) + a_2\partial_z^2\right]\sigma_x + b_2\partial_z^2 \sigma_z, \\ \partial_t^2 \sigma_z = \left[c_1(\partial_x^2 + \partial_y^2) + c_2\partial_z^2\right]\sigma_x + d_2\partial_z^2 \sigma_z, \end{cases} \tag{B-20}$$

Similar analysis as in the first case applies.

Case 3.1: $a_2 b_2 \neq 0$ **and** $c_2 d_2 \neq 0$

Define $r = \frac{a_2}{c_2} = \frac{b_2}{d_2}$, multiply the second equation in B-20 by r, and subtract from the first equation

$$\begin{cases} \partial_t^2 \sigma_x = \left[a_1(\partial_x^2 + \partial_y^2) + a_2\partial_z^2\right]\sigma_x + b_2\partial_z^2 \sigma_z, \\ \partial_t^2(\sigma_x - r\sigma_z) = (a_1 - rc_1)(\partial_x^2 + \partial_y^2)\sigma_x, \end{cases} \tag{B-21}$$

which, after a change of variable $\sigma_z' = \sigma_x - r\sigma_z$ or $\sigma_z = \frac{\sigma_x - \sigma_z'}{r}$, becomes

$$\begin{cases} \partial_t^2 \sigma_x = \left[a_1(\partial_x^2 + \partial_y^2) + (a_2 + \frac{b_2}{r})\partial_z^2\right]\sigma_x - \frac{b_2}{r}\partial_z^2 \sigma_z', \\ \partial_t^2 \sigma_z' = (a_1 - rc_1)(\partial_x^2 + \partial_y^2)\sigma_x. \end{cases} \tag{B-22}$$

This system is a special case of the general system 49 with $b_1 = d_1 = c_2 = d_2 = 0$.

Case 3.2: $a_2b_2 \neq 0$ **and** $c_2d_2 = 0$

In this case, $c_2 = d_2 = 0$, which leads us back to the above case of $b_1 = d_1 = c_2 = d_2 = 0$.

Case 3.3: $a_2b_2 = 0$ **and** $c_2d_2 \neq 0$

In this case, $a_2 = b_2 = 0$, the five constraints reduce to

$$a_1 = c_{11}, \tag{B-23}$$
$$d_2 = c_{33}, \tag{B-24}$$
$$a_1d_2 = c_{11}c_{33} - c_{13}^2. \tag{B-25}$$

There is no solution because the last constraint is not satisfied.

Case 3.4: $a_2b_2 = c_2d_2 = 0$

Consider four cases

- $a_2 = c_2 = 0$: this case has a solution.
- $a_2 = d_2 = 0$: violation of constraint 62.
- $b_2 = c_2 = 0$: the five constraints reduce to

$$a_1 = c_{11}, \tag{B-26}$$
$$a_2 + d_2 = c_{33}, \tag{B-27}$$
$$a_2d_2 = 0, \tag{B-28}$$
$$a_1d_2 = c_{11}c_{33} - c_{13}^2. \tag{B-29}$$

  The last constraint requires that $d_2 \neq 0$, so $a_2 = 0$, which, means $d_2 = c_{33}$. This does not satisfy the last constraints.

- $b_2 = d_2 = 0$: violation of constraint 65.

In conclusion, for the case $b_1 = d_1 = 0$, there are two solutions $b_1 = d_1 = c_2 = d_2 = 0$ and $b_1 = d_1 = a_2 = c_2 = 0$. The first solution results in

$$a_1 = c_{11}, \tag{B-30}$$
$$a_2 = c_{33}, \tag{B-31}$$
$$b_2c_1 = c_{13}^2 - c_{11}c_{33} = v^4(\delta - \epsilon). \tag{B-32}$$

If one chooses $b_2 = v^2$ and $c_1 = v^2(\delta - \epsilon)$ for example, one gets the system

$$\begin{cases} \partial_t^2\sigma_x = \left[v^2(1+2\epsilon)(\partial_x^2 + \partial_y^2) + v^2\partial_z^2\right]\sigma_x + v^2\partial_z^2\sigma_z, \\ \partial_t^2\sigma_z = v^2(\delta - \epsilon)(\partial_x^2 + \partial_y^2)\sigma_x, \end{cases} \tag{B-33}$$

Exchanging b_2 and c_1 gives

$$\begin{cases} \partial_t^2 \sigma_x = \left[v^2(1+2\epsilon)(\partial_x^2+\partial_y^2) + v^2\partial_z^2\right]\sigma_x + v^2(\delta-\epsilon)\partial_z^2\sigma_z, \\ \partial_t^2 \sigma_z = v^2(\partial_x^2+\partial_y^2)\sigma_x, \end{cases} \tag{B-34}$$

The second solution results in

$$a_1 = c_{11}, \tag{B-35}$$

$$d_2 = c_{33}, \tag{B-36}$$

$$b_2 c_1 = c_{13}^2. \tag{B-37}$$

If one chooses $b_2 = c_1 = c_{13}$ for example, one gets back the system 1

$$\begin{cases} \partial_t^2 \sigma_x = c_{11}(\partial_x^2+\partial_y^2)\sigma_x + c_{13}\partial_z^2\sigma_z, \\ \partial_t^2 \sigma_z = c_{13}(\partial_x^2+\partial_y^2)\sigma_x + c_{33}\partial_z^2\sigma_z. \end{cases} \tag{B-38}$$

Case 4: $c_1 = d_1 = 0$

This case is similar to the case $a_1 = b_1 = 0$, already considered above, if one interchanges $\sigma_x \leftrightarrow \sigma_z$.

The above four cases consider what happens when two of four coefficients (a_1, b_1, c_1, d_1) is zero. Analysis of four other cases when two of (a_2, b_2, c_2, d_2) becomes zero is similar by just interchanging $(\partial_x^2 + \partial_y^2) \leftrightarrow \partial_z^2$. Unfortunately, none of the above analyzed solutions produces stable forward and adjoint systems. The only stable solution is one with all non-zero coefficients.

REFERENCES

Alkhalifah, T., 2000, An acoustic wave equation for anisotropic media: Geophysics, **65**, 1239–1250.

Bube, K. P., T. Nemeth, J. P. Stefani, R. Ergas, W. Liu, K. T. Nihei, and L. Zhang, 2012, On the instability in second-order systems for acoustic VTI and TTI media: Geophysics, **77**, No. 5, T171–T186.

Bube, K. P., J. Washbourne, R. Ergas, and T. Nemeth, 2016, Self-adjoint, energy-conserving second-order pseudo-acousti system for VTI and TTI media for reverse-time mingration and full-waveform inversion: SEG Annual International Meeting, Expanded Abstracts, 1110–1115, Society of Exploration Geophysicists.

Claerbout, J. and S. Fomel, 2014, Geophysical Image Estimation by Example: Stanford Exploration Project.

Duveneck, E. and P. M. Bakker, 2011, Stable P-wave modeling for reverse-time migration in tilted TI media: Geophysics, **76**, No. 2, WA3–WA11.

Fletcher, R. P., X. Du, and P. J. Fowler, 2009, Reverse time migration in tilted transversely isotropic (TTI) media: Geophysics, **74**, No. 6, WCA179–WCA187.

Fowler, P. J., X. Du, and R. P. Fletcher, 2010, Coupled equations for reverse time migration in transversely isotropic media: Geophysics, **75**, No. 1, S11–S22.

Le, H. and S. A. Levin, 2014, Removing shear artifacts in acoustic wave propagation in orthorhombic media: SEG Annual International Meeting, Expanded Abstracts, 486–490, Society of Exploration Geophysicists.

Maharramov, M., B. Biondi, and M. Meadows, 2015, Resolving the effects of production-induced overburden dilation using simultaneous tv-regularized time-lapse FWI: SEP-Report, **158**, 1–10.

Xu, S. and H. Zhou, 2014, Accurate simulations of pure quasi-P-waves in complex anisotropic media: Geophysics, **79**, no. 6, T341–T348.

Zhang, Y., H. Zhang, and G. Zhang, 2011, A stable TTI reverse time migration and its implementation: Geophysics, **76**, No. 3, WA3–WA11.

Extended linearized waveform inversion with velocity updating

Alejandro Cabrales-Vargas

ABSTRACT

In this report I recast the objective function of linearized waveform inversion with velocity updating into a subsurface offset extended version. This approach prevents the corresponding Gauss-Newton Hessian from becoming non-positive definite. Both theory and numerical examples appear to support this asseveration, although some work is still required for proper precondition of the extended wave-equation migration velocity analysis operator. Additionally, the new implementation demands more computational resources as a consequence of the extension.

INTRODUCTION

Linearized waveform inversion with velocity updated (LWIVU) consists of coupling the full waveform inversion (FWI) Gauss-Newton Hessian with the wave-equation migration velocity analysis (WEMVA) operator, in an effort to obtain a better estimation of the subsurface reflectivity. The original method employs maximization of the WEMVA-updated migration image power as a constraint:

$$\Phi(\mathbf{r}, \mathbf{\Delta b}) = \frac{1}{2}\|\mathbf{Hr} - \mathbf{W}\mathbf{\Delta b} - \mathbf{r}_m\|_2^2 - \frac{\lambda^2}{2}\|\mathbf{W}\mathbf{\Delta b} + \mathbf{r}_m^0\|_2^2. \tag{1}$$

Here $\mathbf{H}$ is the FWI Gauss-Newton Hessian, $\mathbf{W}$ is the WEMVA operator, $\mathbf{r}$ is the reflectivity, $\mathbf{\Delta b}$ is the perturbation in the background slowness squared, $\mathbf{r}_m^0 = \mathbf{r}_m(\mathbf{b}_0)$ is the migration image using the most background model, $\mathbf{b}_0$, and λ is a trade-off factor that controls the influence of the second functional in Equation 1. The minus sign before $\frac{\lambda^2}{2}$ allows maximization of the image power by minimizing its negative value.

In the preparation of the numerical examples of the previous report (Cabrales-Vargas, 2018) I found that the solution of Equation 1 was not unique. In fact, the result depended on the solving method (*e.g.*, conjugate directions, steepest descent, or a combination of both). Therefore, in this report I start with the analysis of the current objective function (Equation 1). Next, I recast LWIVU to an extended version, using the differential semblance optimization (DSO) operator as part of the constraint function, discussing the new challenges posed and the proposed solutions. Finally, I comment on the future steps to be followed.

THEORY

Original approach: Regularize maximizing the power of the migration image

The gradient of the objective function in Equation 1 is obtained by deriving it with respect to the parameters $\mathbf{r}$ and $\mathbf{\Delta b}$:

$$\nabla\Phi = \begin{bmatrix} \nabla_{\mathbf{r}}\Phi \\ \nabla_{\mathbf{\Delta b}}\Phi \end{bmatrix} = \begin{bmatrix} \mathbf{H}'\big(\mathbf{Hr} - \mathbf{W\Delta b} - \mathbf{r}_m^0\big) \\ -\mathbf{W}'\big(\mathbf{Hr} - \mathbf{W\Delta b} - \mathbf{r}_m^0\big) - \lambda^2\mathbf{W}'\big(\mathbf{W\Delta b} + \mathbf{r}_m^0\big) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{H}' & \mathbf{0} \\ -\mathbf{W}' & -\lambda\mathbf{W}' \end{bmatrix} \begin{bmatrix} \mathbf{Hr} - \mathbf{W\Delta b} - \mathbf{r}_m^0 \\ \lambda\big(\mathbf{W\Delta b} + \mathbf{r}_m^0\big) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{H}' & \mathbf{0} \\ -\mathbf{W}' & -\lambda\mathbf{W}' \end{bmatrix} \left\{ \begin{bmatrix} \mathbf{H} & -\mathbf{W} \\ \mathbf{0} & \lambda\mathbf{W} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{\Delta b} \end{bmatrix} - \begin{bmatrix} \mathbf{r}_m^0 \\ -\lambda\mathbf{r}_m^0 \end{bmatrix} \right\} \tag{2}$$

The last row in Equation 2 can be compared with the gradient of a generic misfit function $\Phi(\mathbf{m}) = \frac{1}{2}\|\mathbf{Lm} - \mathbf{d}\|_2^2$, given as

$$\nabla\Phi = \mathbf{L}'\big(\mathbf{Lm} - \mathbf{d}\big). \tag{3}$$

From Equation 2 we identify

$$\mathbf{L} = \begin{bmatrix} \mathbf{H} & -\mathbf{W} \\ \mathbf{0} & \lambda\mathbf{W} \end{bmatrix}, \tag{4}$$

however, we also notice that

$$\mathbf{L}' \neq \begin{bmatrix} \mathbf{H}' & \mathbf{0} \\ -\mathbf{W}' & -\lambda\mathbf{W}' \end{bmatrix}, \tag{5}$$

The discrepancy between the signs of the terms $\lambda\mathbf{W}$ and $\lambda\mathbf{W}'$ in Equations 4 and 5 prevents the LWIVU operators from becoming adjoint of each other, hence Equation 2 cannot be expressed as the generic form (Equation 3). In addition, if we cascade such operators to build the LWIVU Gauss-Newton Hessian, $\mathcal{H}$, we obtain

$$\mathcal{H} = \begin{bmatrix} \mathbf{H}' & \mathbf{0} \\ -\mathbf{W}' & -\lambda\mathbf{W}' \end{bmatrix} \begin{bmatrix} \mathbf{H} & -\mathbf{W} \\ \mathbf{0} & \lambda\mathbf{W} \end{bmatrix} = \begin{bmatrix} \mathbf{H}'\mathbf{H} & -\mathbf{H}'\mathbf{W} \\ -\mathbf{W}'\mathbf{H} & (1-\lambda^2)\mathbf{W}'\mathbf{W} \end{bmatrix}. \tag{6}$$

The term $1-\lambda^2$ plays a crucial role in the potential ill-posedness of the method. If $\lambda > 1$ the problem is no longer positive definite. As a consequence, gradient-based methods will not converge to a minimum. In particular, if $\lambda \approx 1$, the model null space becomes prominent, thus jeopardizing the reliability of the solution. Therefore, albeit not necessarily invalidating the method, this limitation seriously restricts our freedom of choice for the values of λ. This fact explains why in early examples the method appeared to be promising, but failed as bigger values of λ were required.

Alternative approach: Regularizing with differential semblance optimization

As aforementioned, the main problem of non-extended LWIVU is the risk of ill-posedness due to coupling two functionals where one is minimized whilst the other maximized. The

obvious solution is replacing the latter with a functional that requires minimization. One way of accomplishing this task is employing the DSO operator (Symes and Carazzone, 1991) in the subsurface offset domain. We recast LWIVU as

$$\Phi(\hat{\mathbf{r}}, \boldsymbol{\Delta}\mathbf{b}) = \frac{1}{2}\|\hat{\mathbf{H}}\hat{\mathbf{r}} - \hat{\mathbf{W}}\boldsymbol{\Delta}\mathbf{b} - \hat{\mathbf{r}}_m^0\|_2^2 + \frac{\lambda^2}{2}\|\mathbf{D}\big(\hat{\mathbf{W}}\boldsymbol{\Delta}\mathbf{b} + \hat{\mathbf{r}}_m^0\big)\|_2^2, \tag{7}$$

where $\mathbf{D}$ is the DSO operator. The hat "$\wedge$" is used to represent both operators and parameters that are extended in subsurface offset. Note that the perturbation in the background, $\boldsymbol{\Delta}\mathbf{b}$, remains non-extended. On the other hand, the DSO operator is extended by definition.

One important change introduced with Equation 7 is the plus sign in the second functional. It translates into the minimization of both members. As a consequence, when we obtain the gradient,

$$\begin{aligned}
\nabla\Phi = \begin{bmatrix} \nabla_{\hat{\mathbf{r}}}\Phi \\ \nabla_{\boldsymbol{\Delta}\mathbf{b}}\Phi \end{bmatrix} &= \begin{bmatrix} \hat{\mathbf{H}}'\big(\hat{\mathbf{H}}\hat{\mathbf{r}} - \hat{\mathbf{W}}\boldsymbol{\Delta}\mathbf{b} - \hat{\mathbf{r}}_m^0\big) \\ -\hat{\mathbf{W}}'\big(\hat{\mathbf{H}}\hat{\mathbf{r}} - \hat{\mathbf{W}}\boldsymbol{\Delta}\mathbf{b} - \hat{\mathbf{r}}_m^0\big) + \lambda^2\hat{\mathbf{W}}'\mathbf{D}'\mathbf{D}\big(\hat{\mathbf{W}}\boldsymbol{\Delta}\mathbf{b} + \hat{\mathbf{r}}_m^0\big) \end{bmatrix} \\
&= \begin{bmatrix} \hat{\mathbf{H}}' & \mathbf{0} \\ -\hat{\mathbf{W}}' & \lambda\hat{\mathbf{W}}'\mathbf{D}' \end{bmatrix} \begin{bmatrix} \hat{\mathbf{H}}\hat{\mathbf{r}} - \hat{\mathbf{W}}\boldsymbol{\Delta}\mathbf{b} - \hat{\mathbf{r}}_m^0 \\ \lambda\mathbf{D}\big(\hat{\mathbf{W}}\boldsymbol{\Delta}\mathbf{b} + \hat{\mathbf{r}}_m^0\big) \end{bmatrix} \\
&= \begin{bmatrix} \hat{\mathbf{H}}' & \mathbf{0} \\ -\hat{\mathbf{W}}' & \lambda\hat{\mathbf{W}}'\mathbf{D}' \end{bmatrix} \left\{ \begin{bmatrix} \hat{\mathbf{H}} & -\hat{\mathbf{W}} \\ \mathbf{0} & \lambda\mathbf{D}\hat{\mathbf{W}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{r}} \\ \boldsymbol{\Delta}\mathbf{b} \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{r}}_m^0 \\ -\lambda\mathbf{D}\hat{\mathbf{r}}_m^0 \end{bmatrix} \right\},
\end{aligned} \tag{8}$$

we can identify the forward and adjoint operators strightforwardly as

$$\mathbf{L} = \begin{bmatrix} \hat{\mathbf{H}} & -\hat{\mathbf{W}} \\ \mathbf{0} & \lambda\mathbf{D}\hat{\mathbf{W}} \end{bmatrix}, \quad \mathbf{L}' = \begin{bmatrix} \hat{\mathbf{H}}' & \mathbf{0} \\ -\hat{\mathbf{W}}' & \lambda\hat{\mathbf{W}}'\mathbf{D}' \end{bmatrix}, \tag{9}$$

and therefore, the extended LWIVU Gauss-Newton Hessian becomes

$$\hat{\mathcal{H}} = \begin{bmatrix} \hat{\mathbf{H}}' & \mathbf{0} \\ -\hat{\mathbf{W}}' & \lambda\hat{\mathbf{W}}'\mathbf{D}' \end{bmatrix} \begin{bmatrix} \hat{\mathbf{H}} & -\hat{\mathbf{W}} \\ \mathbf{0} & \lambda\mathbf{D}\hat{\mathbf{W}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{H}}'\hat{\mathbf{H}} & -\hat{\mathbf{H}}'\hat{\mathbf{W}} \\ -\hat{\mathbf{W}}'\hat{\mathbf{H}} & (\mathbf{I} + \lambda^2\mathbf{D}'\mathbf{D})\hat{\mathbf{W}}'\hat{\mathbf{W}} \end{bmatrix}. \tag{10}$$

The model and the data vectors are $\mathbf{m} = \begin{bmatrix} \hat{\mathbf{r}} & \boldsymbol{\Delta}\mathbf{b} \end{bmatrix}^T$ and $\mathbf{d} = \begin{bmatrix} \hat{\mathbf{r}}_m^0 & -\lambda\mathbf{D}\hat{\mathbf{r}}_m^0 \end{bmatrix}^T$ respectively. Note that we no longer have restrictions about the λ value that would induce ill-posedness (other than make it excessively large). This scheme sucessfully passes the dot product test, which did not occur with non-extended LWIVU. However, now there arise three challenges:

1. Higher computational cost and storage because of extending: This problem becomes exacerbated in the 3-D case.
2. Unstable solution: DSO needs careful preconditioning to avoid artifacts, particularly in a linear problem.
3. Stacking extended volumes: Conventional stacking software dramatically slows down when extending.

In the next section I elaborate about these challenges.

EXTENDED LWIVU

Computational cost of subsurface offset extension

Extending the migration image domain in subsurface offset (hence producing offset-domain common image gathers, ODCIG) offers attractive features over the non-extended approach, such as going further and computing angle-domain common image gathers (ADCIG) for amplitude versus offset/azimuth analyses, and enabling migration velocity analysis. Meanwhile, time-lag extension has been proposed as one solution for the slow convergence of least-squares migration in the presence of velocity inaccuracies (**?**). In this report we have seen that it becomes possible to recast LWIVU in such a way that the occurence of ill-posedness as the result of minimizing a functional whilst maximazing the other is avoided altogether.

However, expanding in subsurface offset adds computational cost that is proportional to the total number of subsurface offset samples. Expansion operates exclusively during the scattering and imaging condition stages. Yet, it can easily overcome the computation time that it takes for wavefields propagations. For such a reason, I implement the so-called Nyquist approach (Sun and Fu, 2013). It consists of applying scattering or imaging only at time intervals fine enough to honor the maximum frequency in the data. For example, if the Courant-Friedrichs-Lewy (CFL) condition dictates that the wave propagation time step must be kept no greater than 1 ms, but the maximum frequency in the data is 40 Hz, then scattering/imaging can be applied every 12.5 ms, i.e., about once every twelve CFL time steps. Therefore, the impact of these costly tasks shrinks to one twelveth of the conventional approach. While this scheme works well for conventional RTM imaging, numerical experiments suggest that for seismic inversion it is safer to remain somewhat above the higher frequency. In the previous example this means that one may want to scatter/image every 7-9 CFL time steps. The reason for this, more conservative, approach is that decimating the scatter/imaging time step can produce numerical errors that affect the accuracy of the dot-product test and, as a consequence, the inversion result.

Differential semblance optimization in linear velocity inversion

Since its conception (Symes and Carazzone, 1991) DSO has been, to my knowledge, almost exculsively applied to non-linear velocity inversion (e.g., Vyas and Tang, 2010; Tang, 2011; Lameloise et al., 2015; Yang and Sava, 2015). DSO-based WEMVA typically consists of minimizing the following objective function:

$$\Phi(\mathbf{b}) = \frac{1}{2}\|\,|\mathbf{h}|\mathbf{r}_m(\mathbf{x},\mathbf{h};\mathbf{b})\|_{\mathbf{2}}^{\mathbf{2}}. \tag{11}$$

Here $\mathbf{b}$ represents the background model, containing both low-wavenumber and high-wavenumber components ($\mathbf{b}_0$ and $\mathbf{\Delta b}$, respectively), and $|\mathbf{h}|$ represents the size of the subsurface offset vector that constitutes here the DSO operator, $\mathbf{D}$. The corresponding gradient is

$$\nabla\Phi(\mathbf{b}) = \left[\frac{\mathrm{d}\mathbf{r}_m(\mathbf{b})}{\mathrm{d}\mathbf{b}}\right]' |\mathbf{h}|^2\mathbf{r}_m(\mathbf{b}). \tag{12}$$

I dropped the $\mathbf{x}$ and $\mathbf{h}$ dependencies to simplify the notation. It is important to notice that solving Equation 12 means that we invert for the *whole* background model $\mathbf{b}$. In contrast,

for a linear problem we approximate the migrated image as

$$\mathbf{r}_m(\mathbf{b}) \approx \mathbf{r}_m(\mathbf{b}_0) + \left[\frac{\mathrm{d}\mathbf{r}_m(\mathbf{b})}{\mathrm{d}\mathbf{b}}\bigg|_{\mathbf{b}=\mathbf{b}_0}\right]\boldsymbol{\Delta}\mathbf{b} = \mathbf{r}_m(\mathbf{b}_0) + \mathbf{W}\boldsymbol{\Delta}\mathbf{b} \tag{13}$$

and now we invert for *perturbations in the background* assuming that we know $\mathbf{b}_0$. Substituting Equation 13 in Equation 11 we obtain the linear WEMVA misfit function,

$$\Phi(\boldsymbol{\Delta}\mathbf{b}) = \frac{1}{2}\|\mathbf{D}\big(\mathbf{W}\boldsymbol{\Delta}\mathbf{b} + \mathbf{r}_m^0\big)\|_2^2, \tag{14}$$

and its gradient,

$$\nabla\Phi(\boldsymbol{\Delta}\mathbf{b}) = \mathbf{W}'\mathbf{D}'\mathbf{D}\big(\mathbf{W}\boldsymbol{\Delta}\mathbf{b} + \mathbf{r}_m^0\big). \tag{15}$$

I go back to using $\mathbf{D}$ to represent DSO because I shall experiment with functions other than $\mathbf{D} = |\mathbf{h}|$ in the future. For instance, I am testing tapers for high values of $|\mathbf{h}|$, in an effort reduce the influence of events not related to velocity inaccuracies. The latter are assume to be small (which keeps the method within the linear regime), therefore I do not expect to have significant contributions from large offset values.

It is a common practice to precondition DSO-based WEMVA with low-pass filters. I follow Tang (2011) and employ interpolation using B-splines to penalize high-wavenumber components when inverting for $\boldsymbol{\Delta}\mathbf{b}$ (see Appendix). B-splines also can be used for non-extended WEMVA based on migration image power maximization. I also experiment this strategy for the WEMVA-DSO stage of extended LWIVU.

However, my implementation of linear DSO-based WEMVA still suffers from convergence issues. To illustrate the problem I chose a small section of the Sigsbee A model shown in Figure 1(a). It includes a negative Gaussian velocity perturbation with minimum value of -400 ft/s as seen in Figure 1(b), corresponding to the positive slowness squared perturbation shown in Figure 1(c). I synthesized data using such "true" velocity. For imaging purposes I employed a smoothed version shown in Figure 1(d), which ignores the anomaly (it is assumed to be unknown). The result from conventional extended reverse-time migration (RTM) of the synthetic data is shown in zero-offset section in Figure 2(a), and in an ODCIG at $x = 36000$ ft in Figure 2(b). The latter is extracted at the center of the anomaly, and we observe unfocused energy around that depth. However, there is also low-wavenumber energy at the upper portion of the ODCIG. These artifacts are rejected by masks incorporated into the forward and adjoint operators.

With the data and the migration image we can run the WEMVA inversion. I first performed non-extended WEMVA by maximizing the image power, using a non-extended RTM image (not shown here; it is virtually identical to Figure 2(a)). After 15 iterations we recover the perturbation in the background shown in Figure 3(a), plotted at the same scale as the true anomaly in slowness squared. We observe that the anomaly was satisfactorily recovered in amplitude and shape.

I also performed extended, DSO-based WEMVA using the extended RTM image (Figure 2(a)). After 15 iterations we recover the perturbation shown in Figure 3(a), again plotted at the same scale as the true anomaly. Notice that this time I have failed in matching the amplitude of the anomaly, althought I indeed recovered its shape. Running the process for more iterations proved to be useless, as artifacts are introduced in the image. I am currently working on finding a solution to this problem.

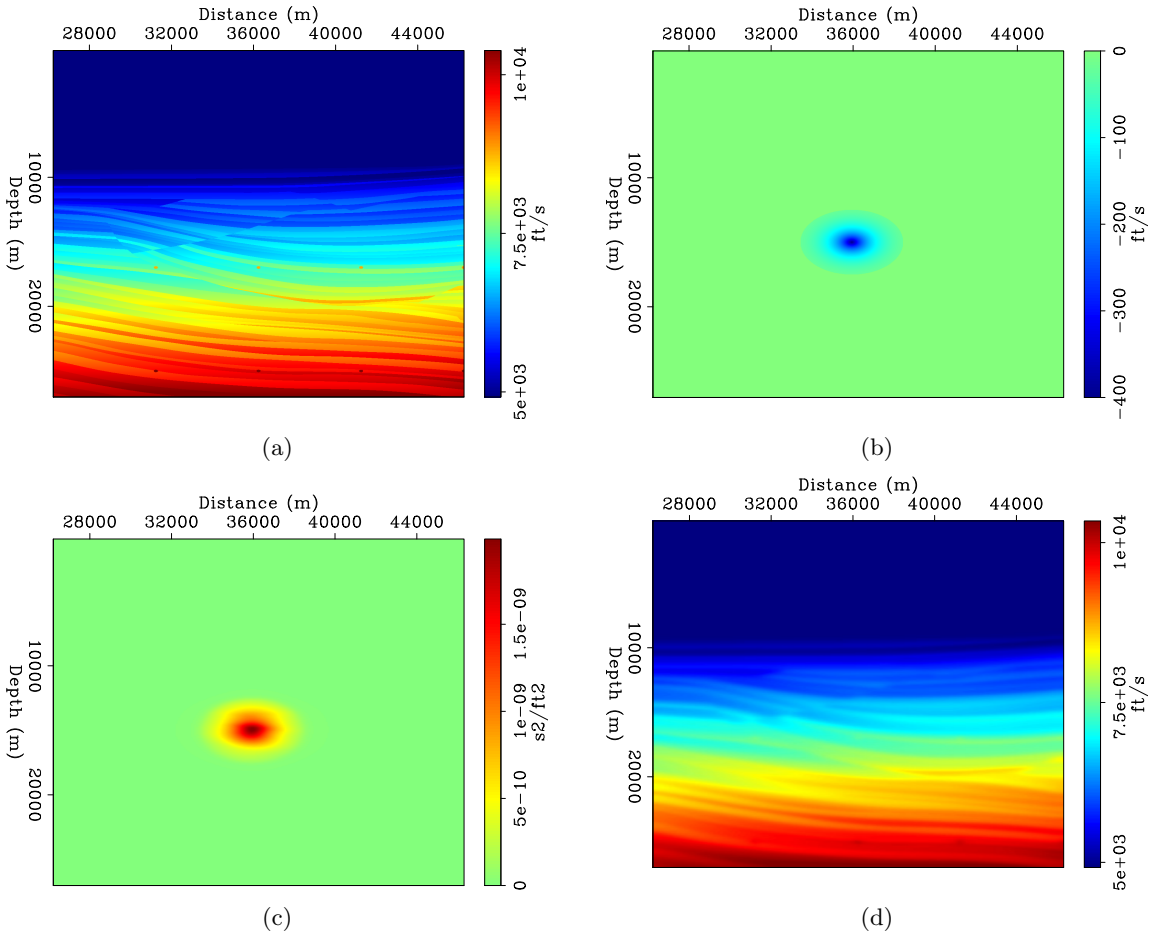
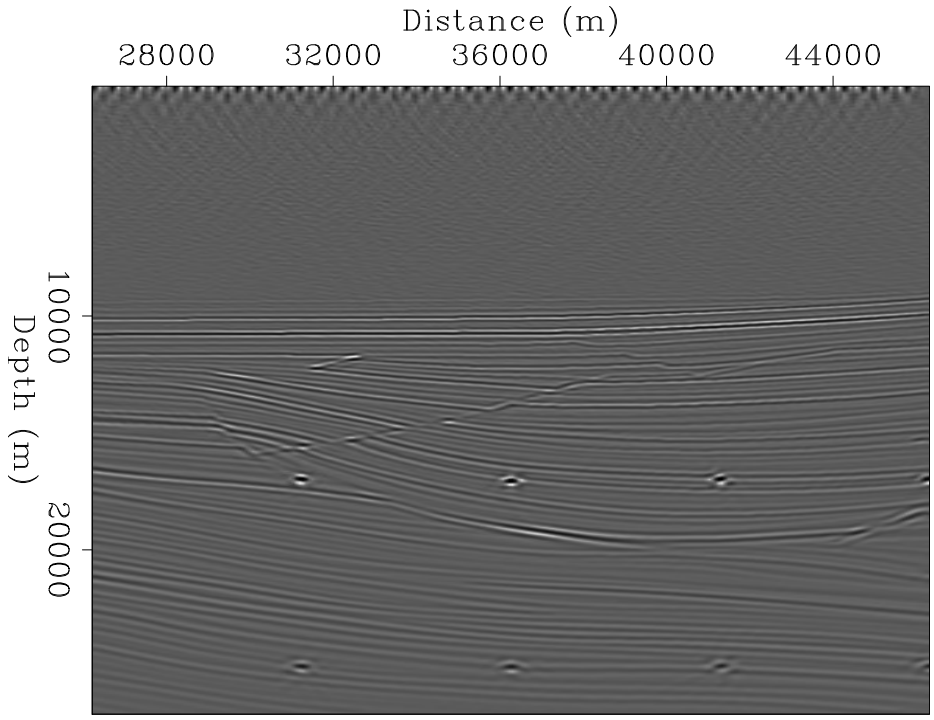
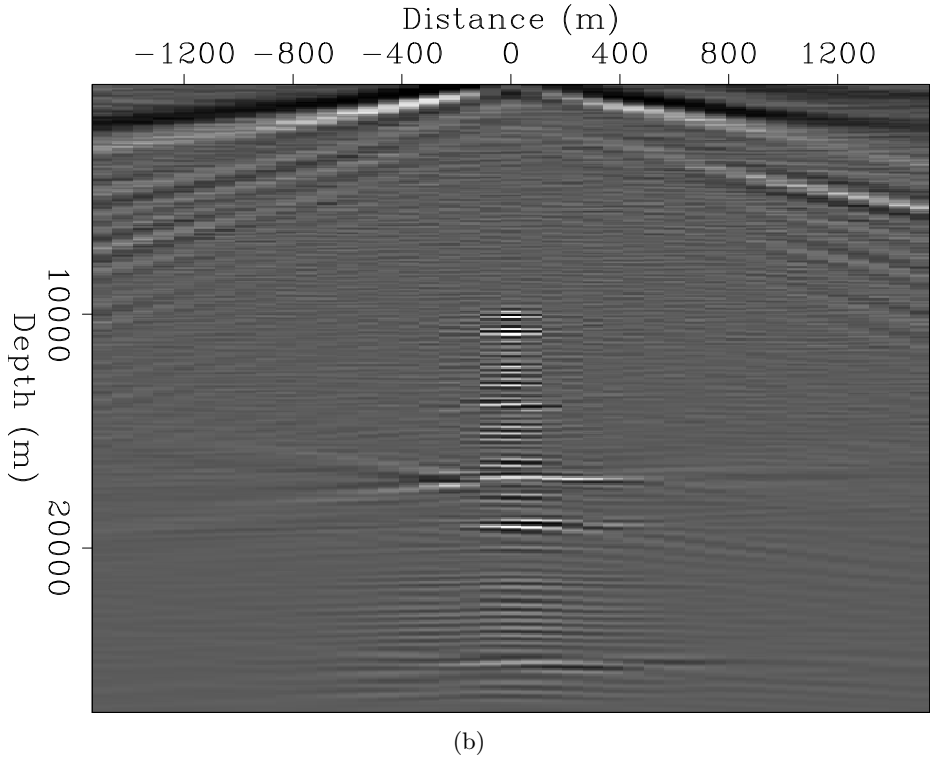


Figure 1: (a) Original velocity model extracted from the Sigsbee A model, including (b) a Gaussian negative anomaly in velocity. I synthesized data using this model. Hence, such data carry the information about the anomaly. (c) The same anomaly expressed in slowness squared becomes positive. (d) Smooth velocity employed for imaging, which does not include the velocity anomaly. [**ER**] alejandro1/. sigs-vel,sigs-vel-pert,sigs-slow2-pert,sigs-svel

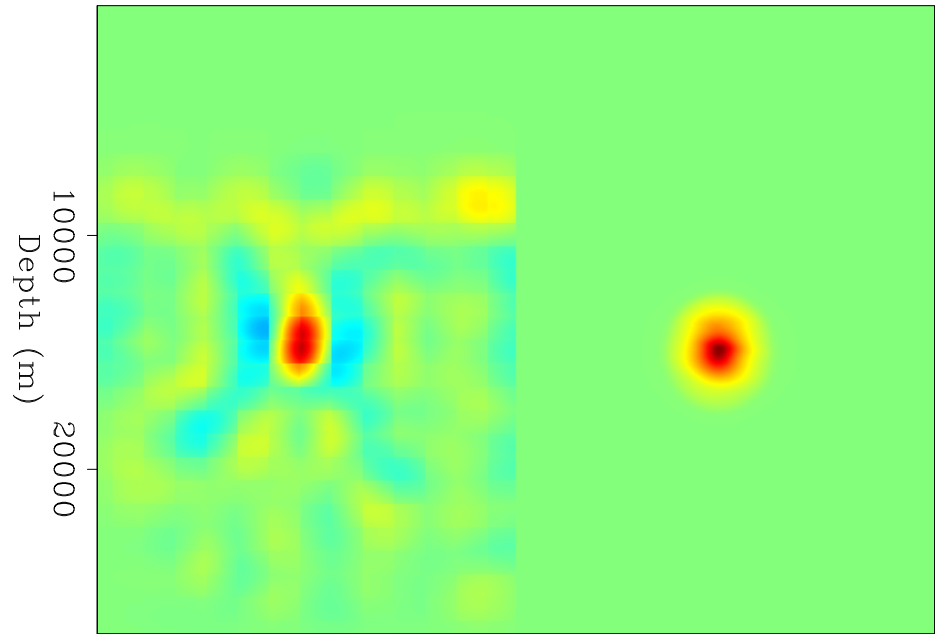


(a)

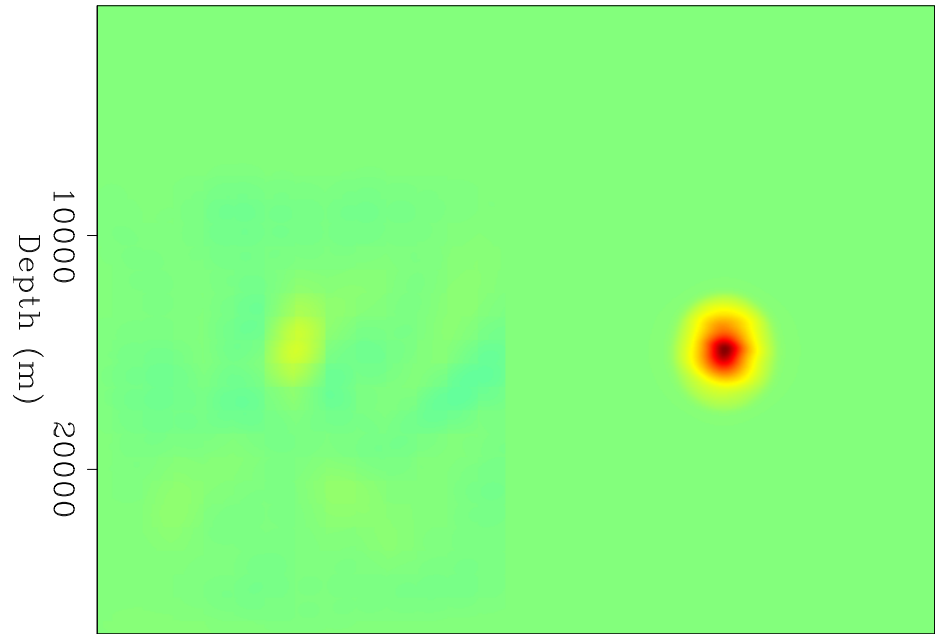


(b)

Figure 2: (a) Extended migration image of the Sigsbee A subset synthetic data, showing the zero subsurface offset section. (b) Same extended migration image, showing an ODCIG extracted at $x = 36000$ ft. The low-wavenumber noise at the upper part of the section is suppressed during the inversions by means of masks. [**CR**] alejandro1/. sigs-rtm1,sigs-rtm2



(a)



(b)

Figure 3: (a) Perturbation in the background recovered with non-extended WEMVA (left), in comparison with the true anomaly (right). (b) Perturbation in the background recovered with extended DSO-based WEMVA (left), in comparison with the true anomaly (right). In both figures, left and right panels were plotted at the same scale. [**CR**] alejandro1/. sigs-wemva-vs-anomaly,sigs-wemva-dso-vs-anomaly

Stacking extended volumes

The multiplicity of dimensions introduced by extending in subsurface offset slows down conventional stacking to the point that it takes longer than imaging. This issue is particularly notorious in 3-D, where two offset dimensions are employed. The inversion processes demand at least one stacking operation by iteration, therefore it represents an important bottleneck.

Bob Clapp (personal communication) suggested some solutions to this problem. I chose the use of MPI parallelization to distribute stacking among the processors. In a 3-D test that I performed, conventional staking took around 50 minutes to completion. For comparison, by using MPI stacking I was able to reduce the computational time to 8 minutes. Further time reduction is still possible.

FUTURE WORK

The current stumbling block is the inadequate perturbation in slowness squared that DSO-based WEMVA recovers. Tests that I will perform include the use of different DSO functions and possibly new masks to confine the updating areas of the gradient and the residual.

Another important issue is the estimation of the parameter λ. Claerbout (2014) and Biondi and Barnier (2017) have proposed different methods to estimate this parameter for balancing the fitting goals in a regularized problem of the type $\Phi(\mathbf{m}) = \frac{1}{2}\|\mathbf{Lm} - \mathbf{d}\|_2^2 + \frac{\lambda^2}{2}\|\mathbf{Rm}\|_2^2$. I will test both approaches for the LWIVU algorithm.

ACKNOWLEDGEMENT

Thanks to Petróleos Mexicanos for the financial support and to the SEP crew for fruitful comments and corrections.

APPENDIX

Preconditioning LWIVU using B-splines

If we make the variable change

$$\mathbf{\Delta b} = \mathbf{B}\mathbf{\Delta p}, \tag{C-1}$$

where $\mathbf{\Delta p}$ is a coarse-grid representation of $\mathbf{\Delta b}$, and $\mathbf{B}$ represents interpolation using B-splines basis functions, then the DSO-based WEMVA misfit function becomes

$$\Phi(\mathbf{\Delta p}) = \frac{1}{2}\|\mathbf{D}\big(\mathbf{WB\Delta p} + \mathbf{r}_m^0\big)\|_2^2 \tag{C-2}$$

and the gradient becomes

$$\begin{aligned}\nabla\Phi(\mathbf{\Delta p}) &= \mathbf{B}'\hat{\mathbf{W}}'\mathbf{D}'\mathbf{D}\big(\hat{\mathbf{W}}\mathbf{B\Delta p} + \hat{\mathbf{r}}_m^0\big)\\ &= \mathbf{B}'\hat{\mathbf{W}}'\mathbf{D}'\mathbf{D}\hat{\mathbf{W}}\mathbf{B\Delta p} + \mathbf{B}'\hat{\mathbf{W}}'\mathbf{D}'\mathbf{D}\hat{\mathbf{r}}_m^0\\ &= \mathbf{B}'\underset{\sim}{\hat{\mathbf{W}}}'\underset{\sim}{\hat{\mathbf{W}}}\mathbf{B\Delta p} - \mathbf{B}'\underset{\sim}{\hat{\mathbf{W}}}'\underset{\sim}{\hat{\mathbf{r}}}_m^0,\end{aligned} \tag{C-3}$$

where $\underset{\sim}{\hat{\mathbf{W}}} = \mathbf{D}\hat{\mathbf{W}}$, $\underset{\sim}{\hat{\mathbf{W}}}' = \hat{\mathbf{W}}'\mathbf{D}'$, and $\underset{\sim}{\hat{\mathbf{r}}}^0_m = -\mathbf{D}\hat{\mathbf{r}}^0_m$. Notice that Equation C-3 has the normal equations structure, including the minus sign. The data are the negative of the extended migration image after DSO.

We can proceed in a similar fashion in LWIVU. However, this time the DSO operator is not applied everytime we compute WEMVA. We start with the preconditioned version of Equation 7

$$\Phi(\hat{\mathbf{r}}, \mathbf{\Delta p}) = \frac{1}{2}\|\hat{\mathbf{H}}\hat{\mathbf{r}} - \hat{\mathbf{W}}\mathbf{B}\mathbf{\Delta p} - \hat{\mathbf{r}}^0_m\|_2^2 + \frac{\lambda^2}{2}\|\mathbf{D}\big(\hat{\mathbf{W}}\mathbf{B}\mathbf{\Delta p} + \hat{\mathbf{r}}^0_m\big)\|_2^2. \tag{C-4}$$

The corresponding gradient is given by

$$\begin{aligned}
\nabla\Phi = \begin{bmatrix} \nabla_{\hat{\mathbf{r}}}\Phi \\ \nabla_{\mathbf{\Delta p}}\Phi \end{bmatrix} &= \begin{bmatrix} \hat{\mathbf{H}}'\big(\hat{\mathbf{H}}\hat{\mathbf{r}} - \hat{\mathbf{W}}\mathbf{B}\mathbf{\Delta p} - \hat{\mathbf{r}}^0_m\big) \\ -\hat{\mathbf{B}}'\hat{\mathbf{W}}'\big(\hat{\mathbf{H}}\hat{\mathbf{r}} - \hat{\mathbf{W}}\mathbf{B}\mathbf{\Delta p} - \hat{\mathbf{r}}^0_m\big) + \lambda^2\hat{\mathbf{B}}'\hat{\mathbf{W}}'\mathbf{D}'\mathbf{D}\big(\hat{\mathbf{W}}\mathbf{B}\mathbf{\Delta p} + \hat{\mathbf{r}}^0_m\big) \end{bmatrix} \\
&= \begin{bmatrix} \hat{\mathbf{H}}' & \mathbf{0} \\ -\hat{\mathbf{B}}'\hat{\mathbf{W}}' & \lambda\hat{\mathbf{B}}'\hat{\mathbf{W}}'\mathbf{D}' \end{bmatrix} \begin{bmatrix} \hat{\mathbf{H}}\hat{\mathbf{r}} - \hat{\mathbf{W}}\mathbf{B}\mathbf{\Delta p} - \hat{\mathbf{r}}^0_m \\ \lambda\mathbf{D}\big(\hat{\mathbf{W}}\mathbf{B}\mathbf{\Delta p} + \hat{\mathbf{r}}^0_m\big) \end{bmatrix} \\
&= \begin{bmatrix} \hat{\mathbf{H}}' & \mathbf{0} \\ -\hat{\mathbf{B}}'\hat{\mathbf{W}}' & \lambda\hat{\mathbf{B}}'\hat{\mathbf{W}}'\mathbf{D}' \end{bmatrix} \left\{ \begin{bmatrix} \hat{\mathbf{H}} & -\hat{\mathbf{W}}\mathbf{B} \\ \mathbf{0} & \lambda\mathbf{D}\hat{\mathbf{W}}\mathbf{B} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{r}} \\ \mathbf{\Delta p} \end{bmatrix} - \begin{bmatrix} \hat{\mathbf{r}}^0_m \\ -\lambda\mathbf{D}\hat{\mathbf{r}}^0_m \end{bmatrix} \right\}.
\end{aligned} \tag{C-5}$$

The forward and adjoint operators and the model and the data vectors are defined as the non-preconditioned case.

REFERENCES

Biondi, E. and G. Barnier, 2017, A flexible out-of-core solver for linear/non-linear problems: SEP-Report, **168**, 295–308.

Cabrales-Vargas, A., 2018, Improving reflectivity using linearized waveform inversion with velocity updating: SEP-Report, **172**, 193–208.

Claerbout, J. F., 2014, Geophysical image estimation by example: Jon Claerbout.

Lameloise, C., H. Chauris, and M. Noble, 2015, Improving the gradient of the image-domain objective function using quantitative migration for a more robust migration velocity analysis: Geophysical Prospecting, **63**, 391–404.

Sun, W. and L. Y. Fu, 2013, Two effective approaches to reduce data storage in reverse time migration: Computers & Geosciences, **56**, 69–75.

Symes, W. and J. Carazzone, 1991, Velocity inversion by differential semblance optimization: Geophysics, **56**, 654–663.

Tang, Y., 2011, Imaging and velocity analysis by target-oriented wavefield inversion: PhD thesis, Stanford University.

Vyas, M. and Y. Tang, 2010, Gradients for wave-equation migration velocity analysis: SEG Technical Program Expanded Abstracts, 4077–4081.

Yang, T. and P. Sava, 2015, Image-domain wavefield tomography with extended common-image-point gathers: Geophysical Prospecting, **63**, 1086–1096.

Full-waveform inversion problem using one-way wave extrapolation operators

Rustam Akhmadiev, Biondo Biondi, and Robert G. Clapp

ABSTRACT

We pose the problem of waveform inversion using one-way wave extrapolation and derive the modeling operator nonlinear with respect to slowness model and its linearized operator. We show that linearized operator is composed of three parts corresponding to upward and downward scattering (low-wavenumber components of slowness perturbation) and perturbation in reflectivity (high-wavenumber components). Using the phase-shift method we simulate wave propagation using full nonlinear and linearized operators in simple 2D acoustic slowness models and discuss future work.

INTRODUCTION

The one-way wave equation has been widely and successfully used for decades in the seismological community. The methods used for solving it have proven to be very efficient and accurate for seismic modeling and imaging. However, with the development of fast computers and rising demand in accuracy of wave simulation in complex geological areas, the methods based on one-way wave equation gave way to more accurate methods operating in time domain and solving full wave equation such as finite-difference modeling and reverse-time migration. Currently, most of the methods for solving full-waveform inversion are based on these time-domain techniques.

The presence of low frequencies and transmitted waves in the data (long offsets in the acquisition) is crucial for recovering all the range of wavenumbers of the velocity model (Mora, 1989). However, historically most of the seismic observations were focused on reflections and even up to this date they prevail in the majority of the seismic data. Using the observations dominated by reflected energy still presents challenges for successful application of FWI (Gauthier et al., 1986).

Modeling and migration methods based on the one-way wave equation are not capable of handling overturned events and have limited angle range compared to full wave equation solutions (Stolt and Weglein, 2012). However, within the range of angles typically observed in the land and streamer seismic data, they are comparable with time-domain methods for accurately modeling the reflected waves. At the same time frequency-domain methods are computationally less expensive (Biondi, 2018), which oftentimes may be a bottleneck for iterative solutions. Moreover, the linearized one-way wave extrapolation operators provide natural scale separation of low- and high-wavenumber components of the slowness model that may also be used when solving waveform inversion problem.

Following the recipe of posing the nonlinear waveform inversion problem, here we show all the required ingredients necessary for finding its solution. First, we present the nonlinear

modeling operator in the matrix form. Then we find its linearization with respect to slowness and derive the operators necessary for solving FWI problem. Finally, we show examples of data modeling in simple slowness models using full nonlinear and linearized operators based on phase-shift extrapolation.

THEORY

The full-waveform inversion problem is generally posed as minimization of an objective function:

$$\mathbf{J_{FWI}} = \frac{1}{2}||\mathbf{f(m)} - \mathbf{d_{obs}}||^{\mathbf{2}}.$$

There are several constituent parts needed for solving this nonlinear inverse problem. One of the most important is knowledge of forward modeling operator $\mathbf{f(m)}$ that describes the process (the wave propagation in case of FWI). Another crucial part for solving optimization problem is finding the linearization of the modeling operator with respect to the model $\mathbf{m}$ (e.g., velocity or slowness in case of FWI). The first one allows us to reproduce the process, while the second one (namely its adjoint) is used for calculating the gradient of the objective function, which is needed for updating the model using gradient-based methods.

Nonlinear modeling operator

When using the full wave equation, the typical way of solving it is in the time domain using finite-difference scheme and updating the wavefields as they progress in time. Therefore, all the interactions of the wavefields with the media are evolving also with time.

If, however, we are to use the one-way wave equation to model the wave propagation, we will observe waves advancing sequentially with depth. Henceforth, in this case the interaction of the wavefields with underlying media evolve with depth rather than time. Consequently, the wavefields $\mathbf{P}_i$ are computed recursively at every i-th depth level and can be described in the matrix form (Biondi, 2006):

$$\begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_2} \\ \vdots \\ \mathbf{P_{z_{max}}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ \mathbf{E_0(s)} & 0 & 0 & \dots & 0 \\ 0 & \mathbf{E_1(s)} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P_0} \\ \mathbf{P_1} \\ \mathbf{P_2} \\ \vdots \\ \mathbf{P_{z_{max}}} \end{bmatrix} + \begin{bmatrix} \mathbf{W}_\omega \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $\mathbf{E}_i$ is the propagation operator at the $i-$th depth level nonlinearly depending on the slowness $\mathbf{s}$ and $\mathbf{W}_\omega$ stands for the source wavelet injected in the propagation domain at the angular frequency ω.

The forward modeling of the wavefields observed at the surface using one-way wave extrapolation can be represented by the sequence of three linear operators (Berkhout, 1982): propagator "surface-reflector", the reflection operator and propagator "reflector-surface". Using this logic and representing all the operators $\mathbf{E}_i$ and wavefields $\mathbf{P}_i$ as one operator $\mathbf{E(s)}$ and wavefield vector $\mathbf{P}$ respectively, we can write the forward modeling process in the following form:

$$\begin{cases} \mathbf{P_{down}} &= \mathbf{E}_+(\mathbf{s})\mathbf{P_{down}} + \mathbf{I}\mathbf{W}_\omega \\ \mathbf{P_{up}} &= \mathbf{E}_-(\mathbf{s})\mathbf{P_{up}} + \mathbf{C}\mathbf{R}(\mathbf{s})\mathbf{P_{down}}. \end{cases} \tag{1}$$

Propagation using system of equations 1 happens in two steps. First, we inject the source $\mathbf{W}_\omega$ (namely its Fourier spectrum) into the media using operator $\mathbf{I}$ and propagate the waves downwards using propagator $\mathbf{E}_+(\mathbf{s})$ that nonlinearly depends on the slowness model $\mathbf{s}$. Then we use reflection operator $\mathbf{R}(\mathbf{s})$, that is approximated by normal-incidence reflectivity, to weight the downgoing wavefield according to the reflectivity in the model. After that we inject this modified wavefield as the source for upgoing waves $\mathbf{P_{up}}$ and propagate them upwards using propagator $\mathbf{E}_-$. The operator $\mathbf{C}$ is a spreading operator that is needed to match the dimensions of the operators and is extending the reflectivity over all the frequencies.

However, for the purpose of full-waveform inversion we need a single expression that models the data in the form of $\mathbf{f}(\mathbf{s}) = \mathbf{d}$. Reorganizing the equation 1 it is straightforward to get the modeling operator $\mathbf{f}(\mathbf{s})$ in the form:

$$\begin{aligned} \mathbf{d} = \mathbf{K}\mathbf{P_{up}} &= \mathbf{K}[\mathbf{1} - \mathbf{E}_-(\mathbf{s})]^{-\mathbf{1}}\mathbf{C}\mathbf{R}(\mathbf{s})[\mathbf{1} - \mathbf{E}_+(\mathbf{s})]^{-\mathbf{1}}\mathbf{I}\mathbf{W}_\omega \\ &= \mathbf{K}\mathbf{U}\mathbf{p}(\mathbf{s})\mathbf{C}\mathbf{R}(\mathbf{s})\mathbf{D}\mathbf{own}(\mathbf{s})\mathbf{I}\mathbf{W}_\omega = \mathbf{f}(\mathbf{s}), \end{aligned} \tag{2}$$

where $\mathbf{K}$ is an operator sampling wavefield at the receiver locations and operators $\mathbf{Down}(\mathbf{s})$ and $\mathbf{Up}(\mathbf{s})$ are propagating a given source downward and upward respectively.

This expression splits the modeling operator into its constituent parts that is convenient for the modular implementation. Moreover, now it is somewhat straightforward to find its linearization with respect to a slowness perturbation.

Linearized forward operator

An important ingredient of any optimization problem using gradient-based methods relies on the accurate estimation of the gradient of the objective function

$$\nabla \mathbf{J_{FWI}} = \left(\frac{\mathbf{df}}{\mathbf{dm}}\right)^* [\mathbf{f}(\mathbf{m}) - \mathbf{d}],$$

$\frac{\mathbf{df}}{\mathbf{dm}}$ is a Frechet derivative of a nonlinear operator $\mathbf{f}(\mathbf{m})$ and can be found using the perturbation analysis.

In the case of operator $\mathbf{f}(\mathbf{s})$ represented by equation 2 we have:

$$\begin{aligned} \frac{\mathbf{df}}{\mathbf{ds}}(\mathbf{s_0}) = \mathbf{K}\frac{\mathbf{dUp}}{\mathbf{ds}}(\mathbf{s_0})\mathbf{R}(\mathbf{s_0})\mathbf{D}\mathbf{own}(\mathbf{s_0})\mathbf{I}\mathbf{W}_\omega + \\ + \mathbf{K}\mathbf{U}\mathbf{p}(\mathbf{s_0})\mathbf{R}(\mathbf{s_0})\frac{\mathbf{dDown}}{\mathbf{ds}}(\mathbf{s_0})\mathbf{I}\mathbf{W}_\omega + \\ + \mathbf{K}\mathbf{U}\mathbf{p}(\mathbf{s_0})\frac{\mathbf{dR}}{\mathbf{ds}}(\mathbf{s_0})\mathbf{D}\mathbf{own}(\mathbf{s_0})\mathbf{I}\mathbf{W}_\omega \end{aligned} \tag{3}$$

Since there are three nonlinear operators with respect to slowness, we are going to have three linearized operators, which constitute the full expression for $\frac{\mathbf{df}}{\mathbf{ds}}$.

Downward extrapolation operator

The downward $\mathbf{E}_+$ and upward $\mathbf{E}_-$ extrapolation operators are exactly the same except that in the first we compute the wavefields starting from top down to the bottom of the model and in the latter the other way around – from bottom to the top. The nonlinearity of these operators is hidden in the complex exponentials that constitute the core of the extrapolation (Claerbout, 1985). At every j-th depth level:

$$\mathbf{E}_{\mp\mathbf{j}} = \mathbf{diag}\left[\exp\left(\pm i\Delta z\sqrt{\omega^2 s_j^2 - |k|^2}\right)\right],$$

where i is the imaginary unit, Δz is the depth step used in extrapolation, ω is the angular frequency, s_j is the reference slowness at the j-th depth level and k is the horizontal wavenumber.

Hence, first of all we need to find the linear approximation of the complex exponential with respect to the slowness perturbation $\mathbf{ds} = [ds_0, \ldots, ds_{nz}]^T$. Using Taylor expansion around background slowness $\mathbf{s_0}$ and ignoring higher order terms we get

$$\begin{aligned}\mathbf{E}_{\mp\mathbf{j}}(s_{0j} + ds_j) &= \mathbf{diag}\left[\exp\left(\pm i\Delta z\sqrt{\omega^2(s_{0j} + ds_j)^2 - |k|^2}\right)\right] \\ &\approx \mathbf{diag}\left[\exp\left(\pm i\Delta z\sqrt{\omega^2 s_{0j}^2 - |k|^2}\right)\right] \times \\ &\times \left(\mathbf{1} + \mathbf{diag}\left[\frac{\pm i\omega\Delta z}{\sqrt{1 - |k|^2/\omega^2 s_{0j}^2}} ds_j\right]\right).\end{aligned} \tag{4}$$

This additional term is easily recognized to be the correction used in split-step and Fourier finite-difference migration methods.

Consequently, the linearization of downward extrapolation operator $\mathbf{E}_+(\mathbf{s})$ is represented in the form

$$\mathbf{E}_+(\mathbf{s_0} + \mathbf{ds}) = \mathbf{E}_+(\mathbf{s_0}) + \mathbf{E}_+(\mathbf{s_0})\mathbf{G}_+(\mathbf{s_0})\mathbf{ds}, \tag{5}$$

where now operator $\mathbf{G}_+(\mathbf{s_0})$ is a forward scattering operator consisting of the aforementioned correction term and is nonlinear with respect to background slowness $\mathbf{s_0}$. Because the expression under the square root is in the mixed space-wavenumber domain, its numerical implementation in the laterally heterogeneous medium is challenging but can be done in the similar fashion with split-step and Fourier finite-difference migration algorithms.

Now using the perturbation analysis:

$$\mathbf{P^0_{down}} + \mathbf{dP_{down}} = [\mathbf{E}_+(\mathbf{s_0}) + \mathbf{E}_+(\mathbf{s_0})\mathbf{G}_+(\mathbf{s_0})\mathbf{ds}][\mathbf{P^0_{down}} + \mathbf{dP_{down}}] + \mathbf{IW}_\omega.$$

Therefore, the background $\mathbf{P^0_{down}}$ and perturbed wavefield $\mathbf{dP_{down}}$ can be computed using following system of equations (based on equation 3) :

$$\begin{cases}\mathbf{P^0_{down}} &= \mathbf{E}_+(\mathbf{s_0})\mathbf{P^0_{down}} + \mathbf{IS}_\omega \\ \mathbf{dP_{down}} &= \mathbf{E}_+(\mathbf{s_0})\mathbf{dP_{down}} + \mathbf{E}_+(\mathbf{s_0})\mathbf{G}_+(\mathbf{s_0})\mathbf{P^0_{down}}\mathbf{ds} \\ \mathbf{dP^{(1)}_{up}} &= \mathbf{E}_-(\mathbf{s_0})\mathbf{dP_{down}} + \mathbf{CR}(\mathbf{s_0})\mathbf{dP_{down}}.\end{cases} \tag{6}$$

First, we propagate the downgoing wavefield $\mathbf{P^0_{down}}$ in the background slowness model $\mathbf{s_0}$. Then, we scatter it off of slowness perturbation $\mathbf{ds}$ using operator $\mathbf{G_+}$ and propagate scattered wavefield downward. Finally, we reflect the scattered wavefield from the reflectors existing in the background model using operator $\mathbf{R(s_0)}$ and propagate the resulting wavefield up to the surface with $\mathbf{E_-(s_0)}$. Hence, this part of the linearized operator represents the downward scattering (Figure 1).

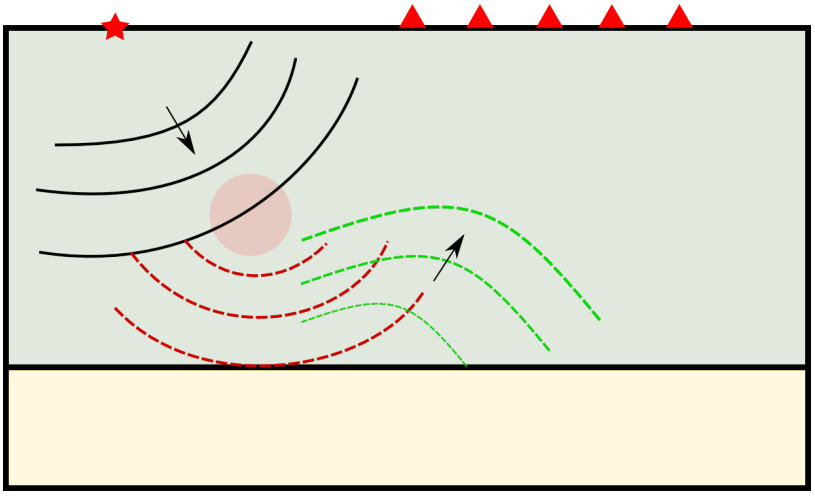


Figure 1: Downward scattering off the slowness perturbation shown in red. The wavefront shown in black corresponds to the first line, red – to the second line and green – to the third line of equation 6. [**NR**] arustam1/. down

Upward extrapolation operator

In similar manner we can analyze the second part of the full linearized operator that represents upward scattering. The linearized expression comes from Taylor expansion of complex exponential, which is exactly the same as for the downward extrapolation operator (equation 4) except that it propagates the energy from bottom to the top of the model.

$$\mathbf{E_-(s_0 + ds)} = \mathbf{E_-(s_0)} + \mathbf{E_-(s_0)G_-(s_0)ds} \tag{7}$$

Again using the perturbation analysis and equation 3 we write

$$\begin{cases} \mathbf{P^0_{up}} & = \mathbf{E_-(s_0)P^0_{up}} + \mathbf{CR(s_0)P^0_{down}} \\ \mathbf{dP^{(2)}_{up}} & = \mathbf{E_-(s_0)dP_{up}} + \mathbf{E_-(s_0)G_-(s_0)P^0_{up}ds}. \end{cases} \tag{8}$$

Here we see that the upgoing background wavefield $\mathbf{P^0_{up}}$ is formed by reflecting of the downgoing background wavefield $\mathbf{P^0_{down}}$ obtained at the previous step and propagating the result upwards. This upgoing background wavefield is now scattered on its way up and contributes to the final scattered wavefield.

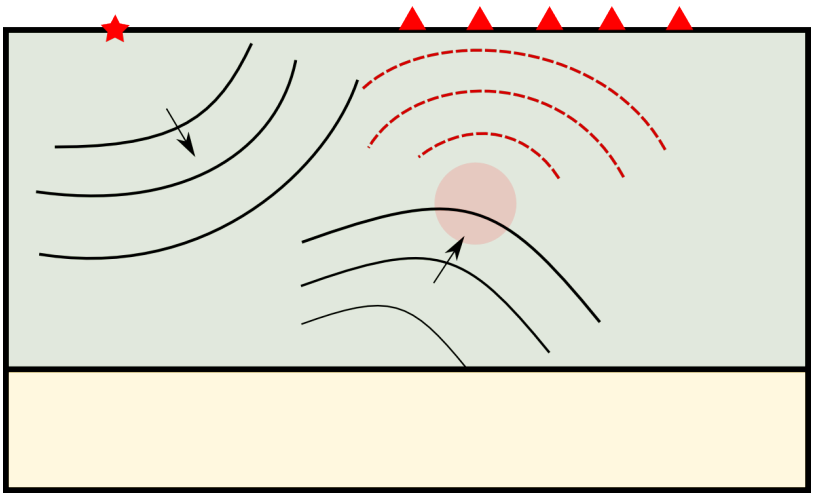


Figure 2: Upward scattering off the slowness perturbation shown in red. The upgoing wavefront shown in black corresponds to the first line, red – to the second line of equation 8. [**NR**] arustam1/. up

Reflection operator

Correct representation of the reflection operator is important because it affects the amplitudes of the waves. The reflection coefficient varies depending on the P- and S-velocities, densities and angles of propagation. For this reason, theoretically, in order to model the amplitudes correctly, the reflection operator should be dependent on frequency and wavenumber (analogous to angle dependency). Moreover, it is not necessarily diagonal, because realistic wave interactions with the reflector are not local (Aki and Richards, 2002). As a result, it may potentially be applied as the weighting operator in the wavenumber domain, or equivalently, convolution in space.

Nevertheless, at this stage of our project as a first approximation we can construct reflection operator as a diagonal weighting operator with diagonal entries equal to normal incidence reflectivity. This means essentially that in forward modeling we are primarily aiming at reproducing the kinematics of wave propagation that have adequate but not necessarily exact amplitudes.

In this way, the reflection operator is equal to

$$\mathbf{R}(\mathbf{s}) = \mathbf{diag}\left[\frac{s_i - s_{i+1}}{s_i + s_{i+1}}\right] \tag{9}$$

To find its linearized version we again use perturbation theory. After several simplifications in the fraction and neglecting higher-order terms:

$$\begin{aligned}\mathbf{R}(\mathbf{s_0}+\mathbf{ds}) &= \mathbf{diag}\left[\frac{s_i + ds_i - s_{i+1} - ds_{i+1}}{s_i + ds_i + s_{i+1} + ds_{i+1}}\right] \approx \\ &\approx \mathbf{R}(\mathbf{s_0}) + \mathbf{diag}\left[\frac{2ds_i}{(s_i+s_{i+1})^2} + \frac{-2ds_{i+1}}{(s_i+s_{i+1})^2}\right] = \\ &= \mathbf{R}(\mathbf{s_0}) + \mathbf{dR}(\mathbf{s_0})\end{aligned} \tag{10}$$

It is easy to see that linearized operator $\mathbf{dR}(\mathbf{s_0})$ corresponds to weighted backward difference along the depth axis with weights depending on local slowness values and consequently, its adjoint is a weighted forward difference.

Using this expression and equation 3 it is easy to see that slowness perturbation in the reflection operator gives rise to the scattered wavefield $\mathbf{dP^{(3)}_{up}}$ that can be computed as:

$$\mathbf{dP^{(3)}_{up}} = \mathbf{E_{-}(s_0)dP_R} + \mathbf{CdR(s_0)P^0_{down}} \tag{11}$$

This part of the linearized operator accounts for reflections of the background wavefield off the scatterers in contrast with scattering in the previous equations 6 and 8.

It is easy to show that its adjoint corresponds to the one-way wave equation migration that restores high-wavenumber component of the slowness model (reflectors). This fact is in concordance with the previously mentioned natural scale separation inherent to the waveform inversion using one-way wave extrapolation operators.

RESULTS

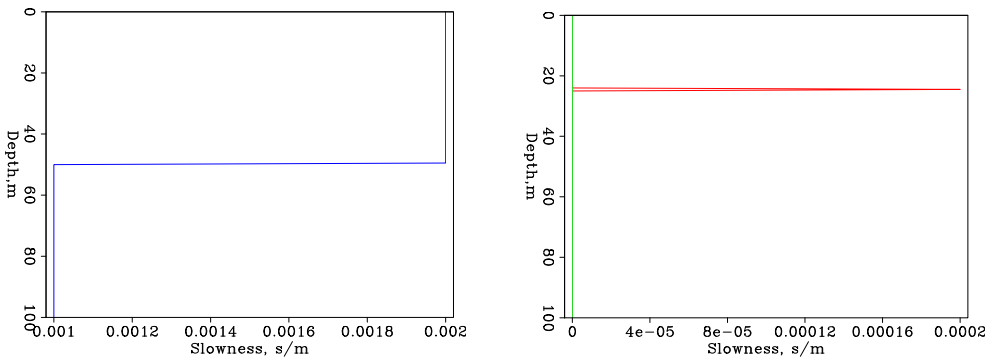
The full nonlinear modeling and linearized operators were implemented using phase-shift extrapolation (Gazdag, 1978) with one reference slowness equal to the average slowness at the current depth level. Due to the modular approach (following from equation 2), extending the propagator to more complicated wavefield extrapolation methods such as split-step or Fourier finite-difference and including multiple reference slownesses can be done in the same framework.

The wavefields are simulated in a simple model consisting of two layers (Figure 3(a)) of 500 and 1000 m/s with the reflector located at 50 m. The source located at the surface in the center and the receivers are uniformly distributed at the top of the model. The source wavelet is the minimum-phase analogue of the Ricker wavelet with the central frequency of 15 Hz. To test the linearized operator (equation 3), one scattering point with slowness perturbation equal to 10% of the background value was added right under the source location (Figure 3(b)) at the depth of 25 m.

Reflected event has the expected hyperbolic moveout with correct zero-offset time of 0.2 s (Figure 4(a)). The artifacts are resulting from the wraparound effect inherent to the frequency-domain computations. They can be suppressed by adding the lateral tapering of the propagated wavefields at the model boundaries. The data simulated using linearized operator (equation 3) has three distinct events (Figure 4(b)). The first hyperbola corresponds to the reflection of the background wavefield off the scattering point (equation 11). The second event corresponds to the downward scattering of the background wavefield at the slowness perturbation (the hyperbola with faster aparent velocity) and results from equation 6. The third event (with slower aparent velocity) corresponds to the upward scattering and equation 8.

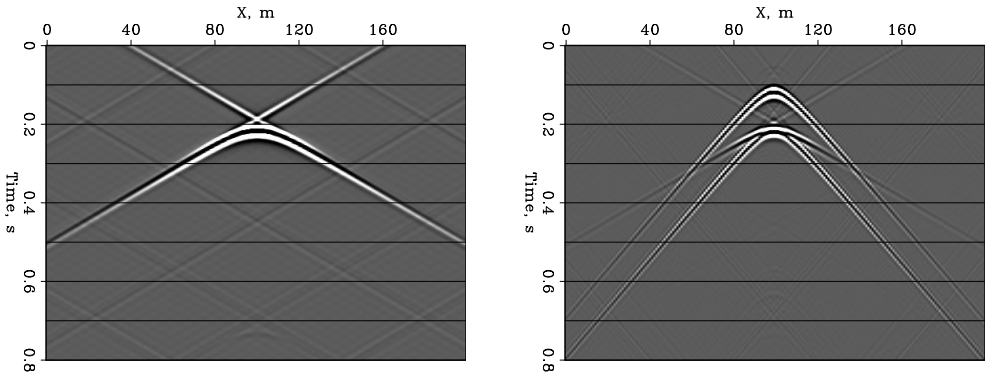
DISCUSSION AND FUTURE WORK

As we have seen the proposed method is able to model the wave propagation using the full nonlinear operator and its linearized version. The observed data behaves as expected and



(a) (b)

Figure 3: Models used for wave propagation: (a) – background slowness model, (b) – scattering point used as an input for linearized operator. [**ER**] arustam1/. slow,dslow



(a) (b)

Figure 4: Wavefield modeling using phase-shift extrapolation: (a) – full nonlinear operator, (b) – linearized forward operator. [**ER**] arustam1/. ph1,born

is in agreement with theoretical predictions. Easy parallelization of the algorithm over the frequency will allow faster computations than time-domain methods. Moreover, absence of limitations on the stability of the method will potentially permit high-resolution waveform inversion. The future work will include extending the propagator to more complex extrapolation operators (e.g., split-step) and adding more reference slownesses. The following steps will involve implementing the adjoint operators and finally, running waveform inversion.

ACKNOWLEDGEMENT

The authors would like to thank SEP sponsors for the support.

REFERENCES

Aki, K. and P. Richards, 2002, Quantitative seismology, 2nd ed.: University Science Books.

Berkhout, A. J., 1982, Seismic migration. imaging of acoustic energy by wave field extrapolation: Elsevier Scientific Publishing Company.

Biondi, B., 2006, 3d seismic imaging: Society of Exploration Geophysicists.

———, 2018, Can we beat ffts computational speed for downward-continuation?: SEP-Report, **174**, 81–90.

Claerbout, J., 1985, Imaging the earth's interior.

Gauthier, O., J. Virieux, and A. Tarantola, 1986, Two-dimensional nonlinear inversion of seismic waveforms: Numerical results: Geophysics, **51**, 1387–1403.

Gazdag, J., 1978, Wave equation migration with the phase-shift method: Geophysics, **43**, 1342–1351.

Mora, P., 1989, Inversion = migration + tomography: Geophysics, **54**, 1575–1586.

Stolt, R. H. and A. B. Weglein, 2012, Seismic imaging and inversion: Cambridge University Press.

Wavefield component separation and debubble on a 3D OBN dataset

Taylor Dahlke, Alejandro Cabrales-Vargas, Rustam Akhmadiev

ABSTRACT

In this report we demonstrate the pre-processing steps performed on the OBN Gulf of Mexico data set provided to us by Shell in preparation for an FWI type workflow. We show the results of performing PZ-summation on the hydrophone data for up and downgoing wavefield separation, and follow with the extraction of the estimated source wavelet and creation of a shaping filter to remove the bubble and facilitate better phase matching with our synthetic data. We apply this filter to the remaining data and find we can effectively remove the bubble.

INTRODUCTION

In order to successfully perform an FWI style of inversion with our dataset, we first need to apply a standard processing flow. An important goal to keep in mind is that we ultimately want to calculate a 'good' residual for use in our inversion. To do this, we need to have some similarity between our synthetically modeled data and the observed data. One feature in our observed data that can be difficult to accurately recreate in our synthetic data is the bubble that follows the initial source injection. For this reason, we choose to remove it from the observed data.

This paper intends to show that a straight-forward way to simultaneously remove the bubble and increase similarity between our observed and synthetic data is to shape the observed data to the source wavelet we use in our synthetic modeling. To do this, we first perform separation of the up and downgoing components from the hydrophone data using PZ-summation. Next, we extract a representative wavelet from the observed data first-arrival, and then estimate a filter that shapes it to the synthetic data source wavelet. We then apply this filter to the entire dataset, resulting in a debubbled dataset that has a high degree of phase similarity with our synthetic data.

PZ-summation

Since the dataset was recorded on ocean bottom nodes (OBN), the first arrival in the hydrophone component contains an up-going ocean bottom reflection that nearly coincides with the downgoing direct arrival event. This means the wavelet we extract from the first arrival event will have an ocean bottom event mixed into it. To get an accurate estimated source wavelet, we need to isolate the downgoing direct arrival. For this reason, we first need to perform PZ-summation before we do any source wavelet estimation.

PZ-summation is a technique used to separate the up and down going components of hydrophone data (figure 1) using the complimentary information found in the vertical

component data (figure 2). We base our application of this processing step on the approach and assumptions used by Biondi and Levin (2014), which represents the up and downgoing data with the following equations:

$$P_{up}(f,k) = \frac{1}{2}P(f,k) + a(f)\frac{\rho}{2q(f,k)}Z(f,k), \tag{1}$$

$$P_{down}(f,k) = \frac{1}{2}P(f,k) - a(f)\frac{\rho}{2q(f,k)}Z(f,k), \tag{2}$$

where P is the designatured pressure data, Z is the designatured vertical data, $a(f)$ is the calibration filter, ρ is the water density, and q is the vertical slowness of the water layer defined as:

$$q(f,k) = \sqrt{c^{-2} - p^2(f,k)}. \tag{3}$$

In this case, c is the water velocity at the receiver position and p is the ray parameter.

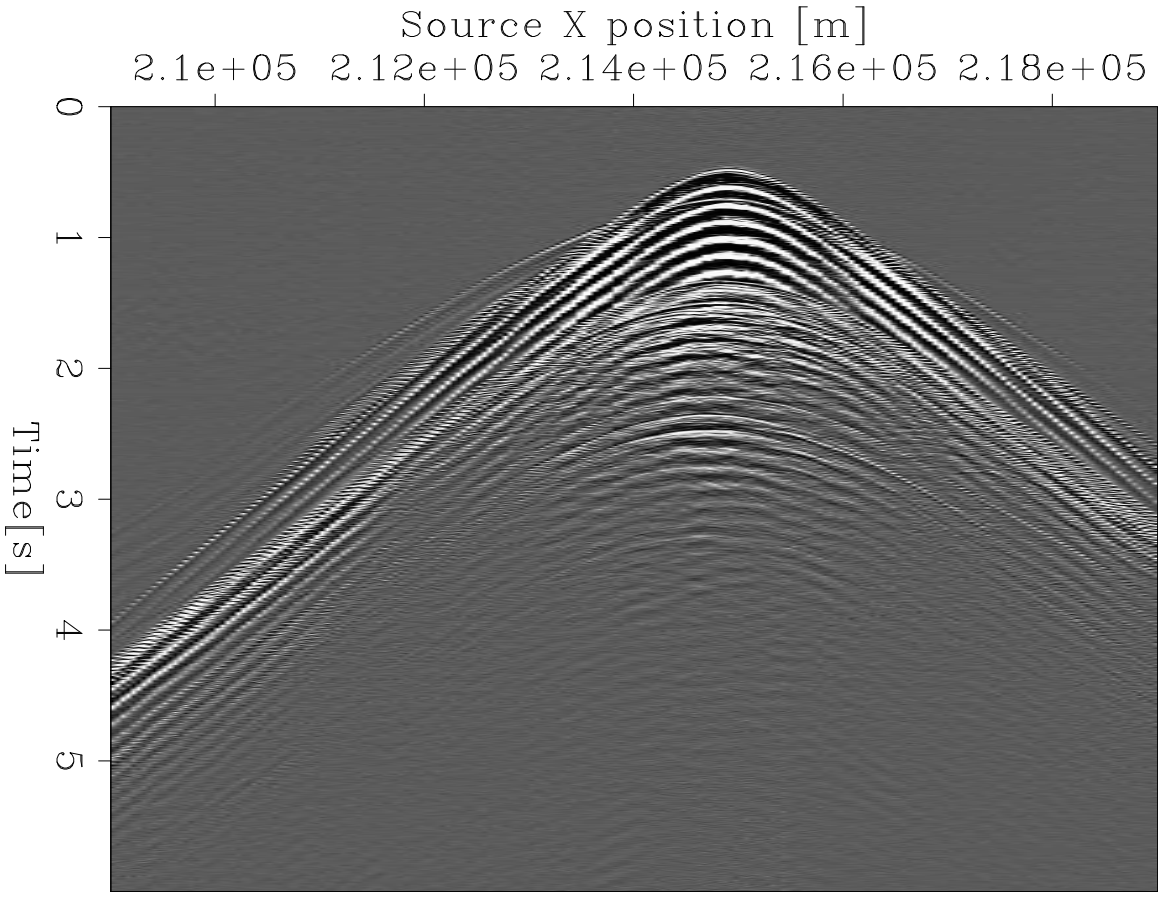


Figure 1: 2D line receiver gather example from designatured and bandpassed hydrophone data. [**CR**] taylor1/. inputHydro

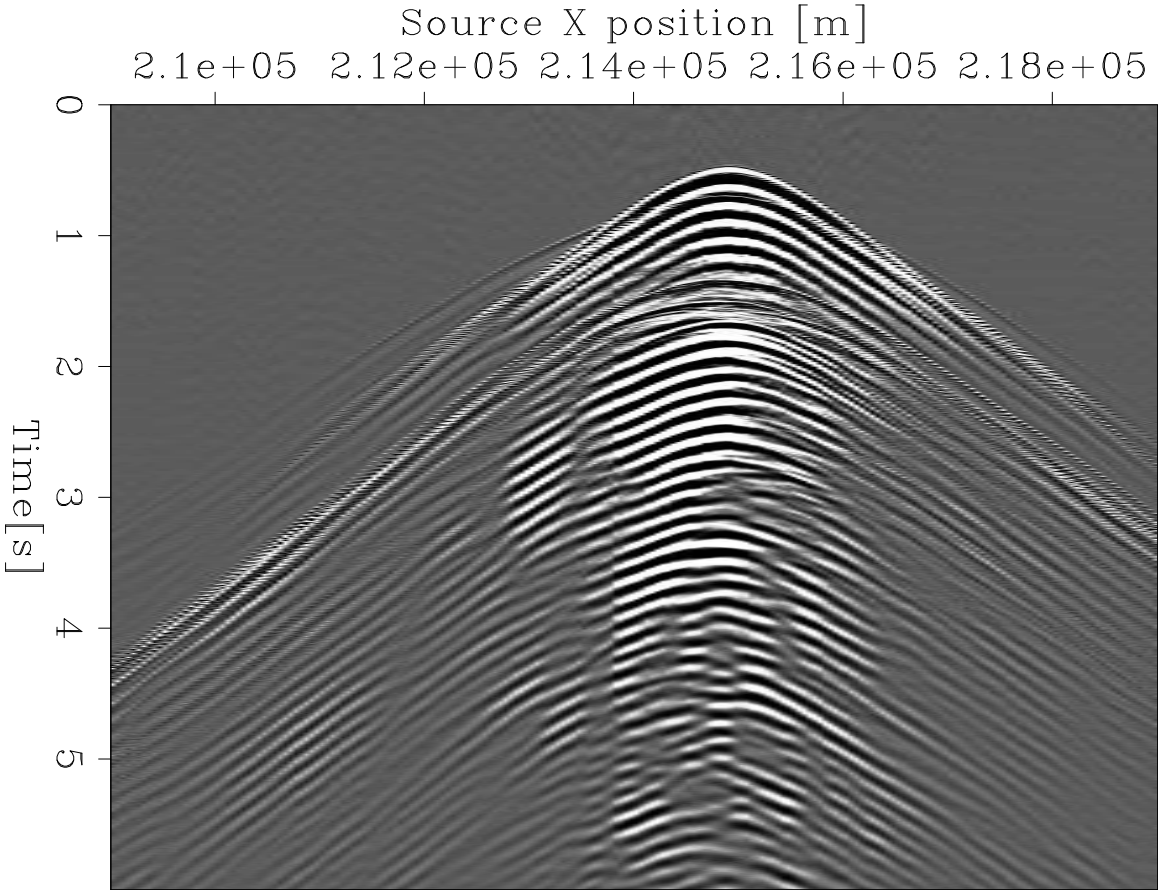


Figure 2: 2D line receiver gather example from designatured and bandpassed geophone data. [**CR**] taylor1/. inputGeo

One event in the data we can leverage is the refraction event, which by definition is an upgoing event. Furthermore, the refraction event is naturally separated in the time domain at far offsets, making it easy to isolate (see figures 3 and 4).

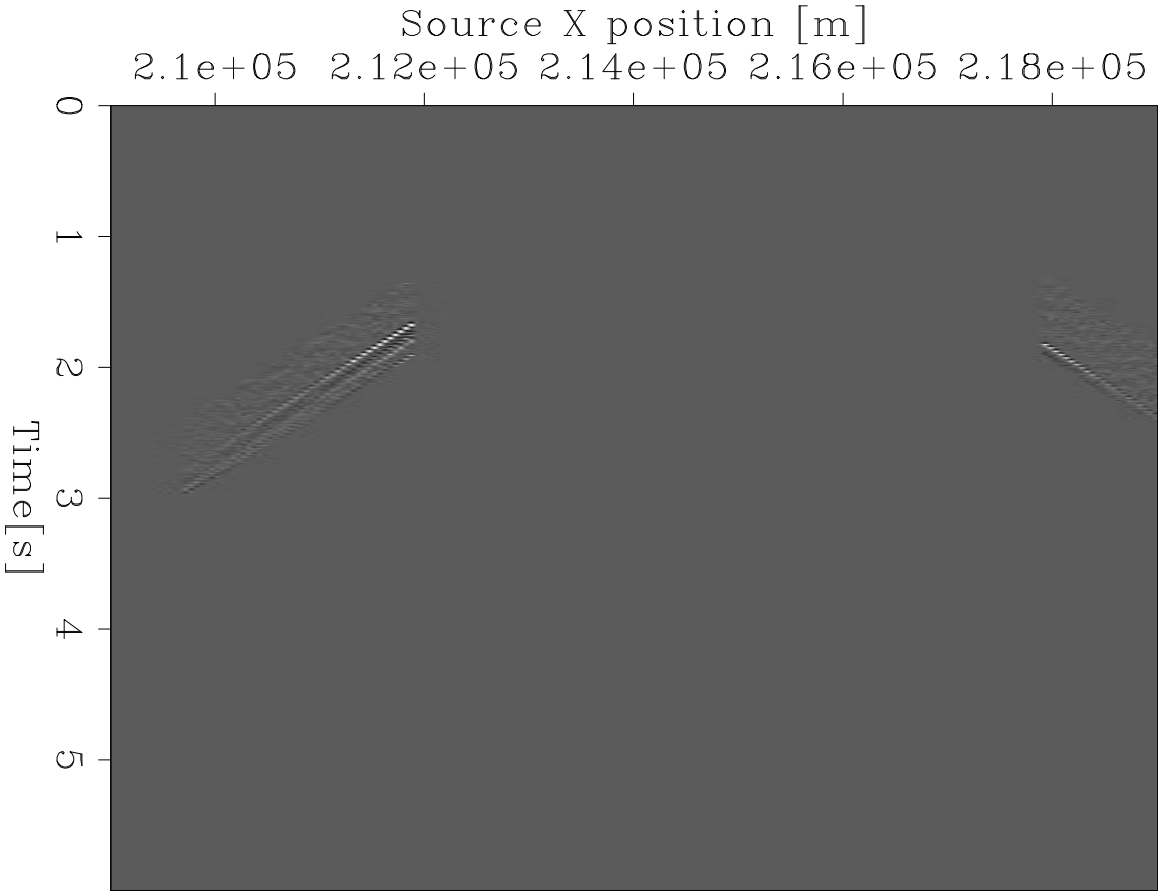


Figure 3: Hydrophone data window (isolating the refraction events) used for estimating PZ-summation filter. [**CR**] taylor1/. Pwindow

We can estimate a filter $\hat{a}(f)$ to apply to the windowed refraction event such that it minimizes its energy. However, the inverted $\hat{a}(f)$ contains the effect of the vertical slowness and water density. We can represent this with equation 4:

$$\hat{a}(f) = a(f)\frac{\rho}{2q(f,k)}. \tag{4}$$

However, we assume the water velocity and density are constant throughout, and that the ray parameter is constant in the window we minimize. We can then remove this effect from $\hat{a}(f)$ to get $a(f)$. We can estimate the ray parameter p in the rest of the data which allows us to reuse $a(f)$ to separate the upgoing (figure 5) and downgoing (figure 6) components using equations 1 and 2.

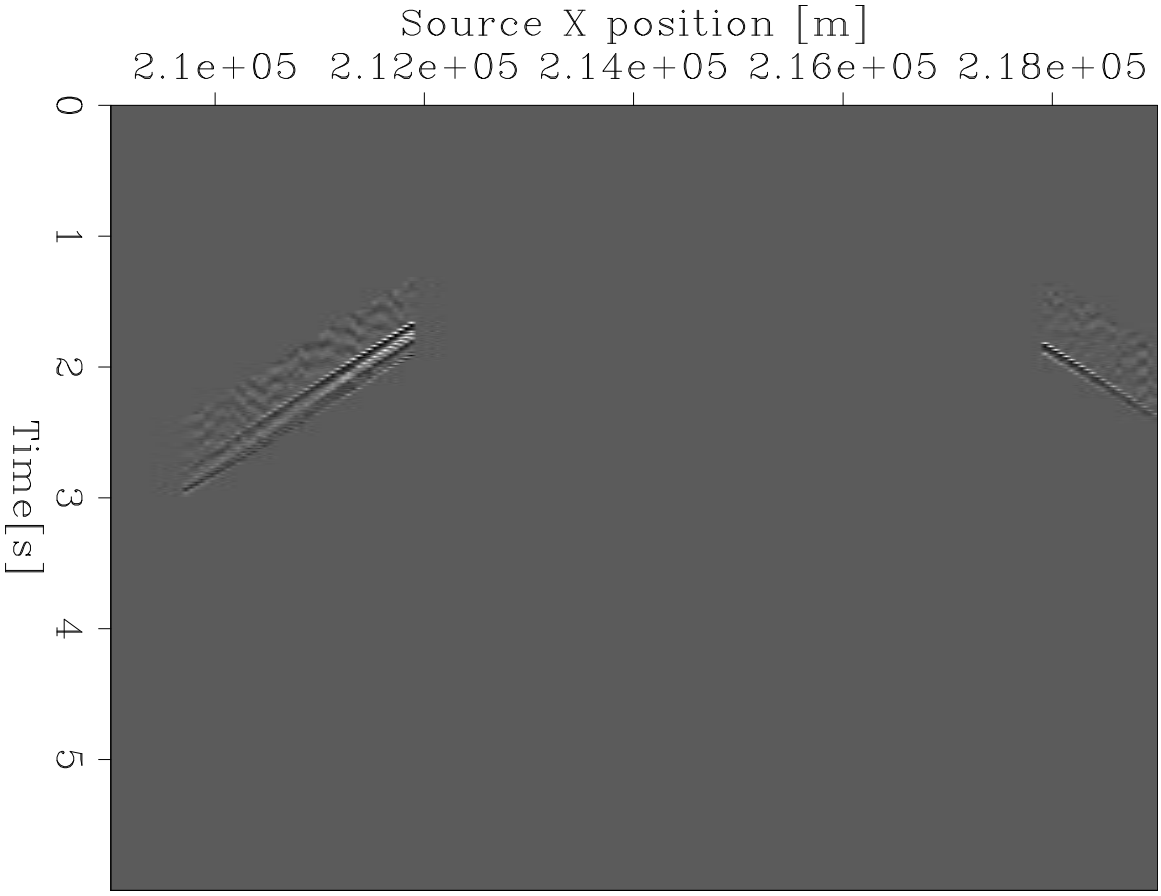


Figure 4: Geophone data window (isolating the refraction events) used for estimating PZ-summation filter. [**CR**] taylor1/. Zwindow

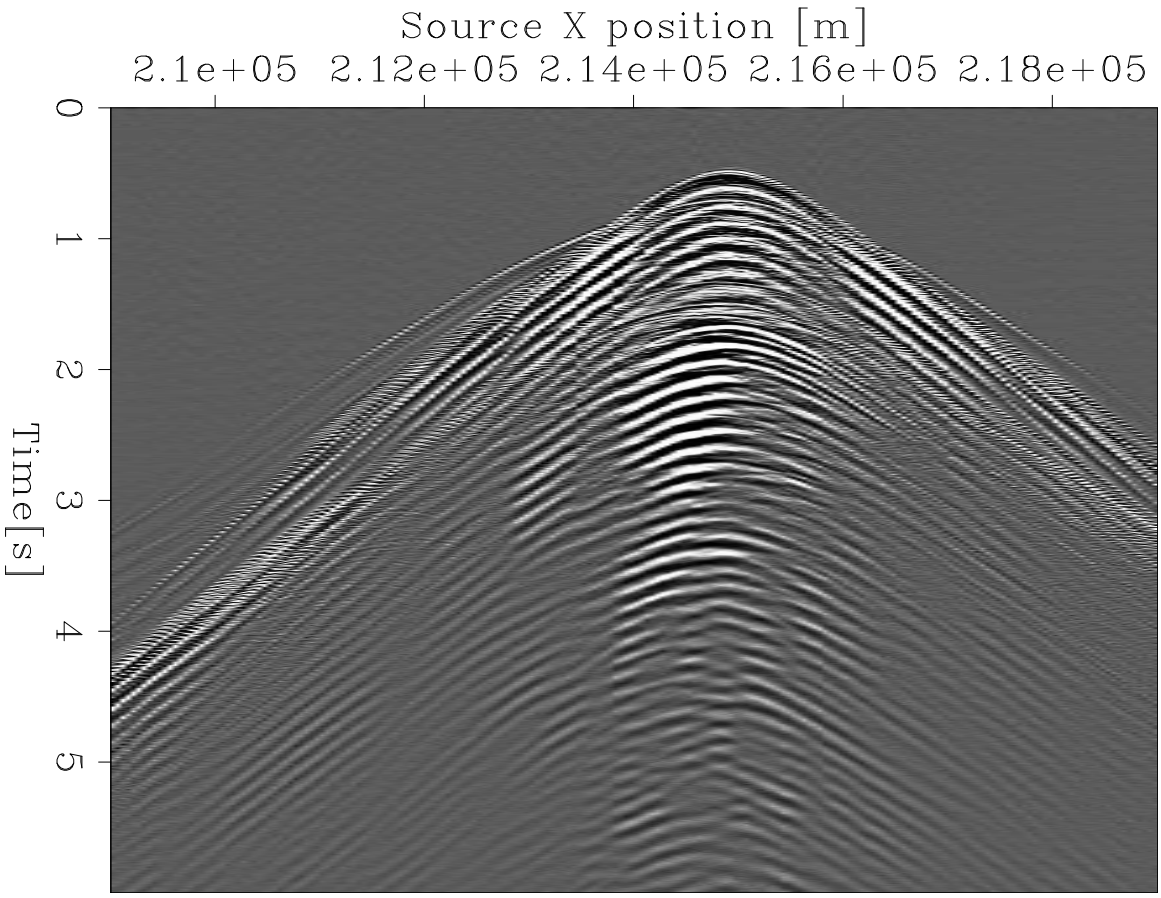


Figure 5: Upgoing component output from PZ-summation. [**CR**] taylor1/. upgoing

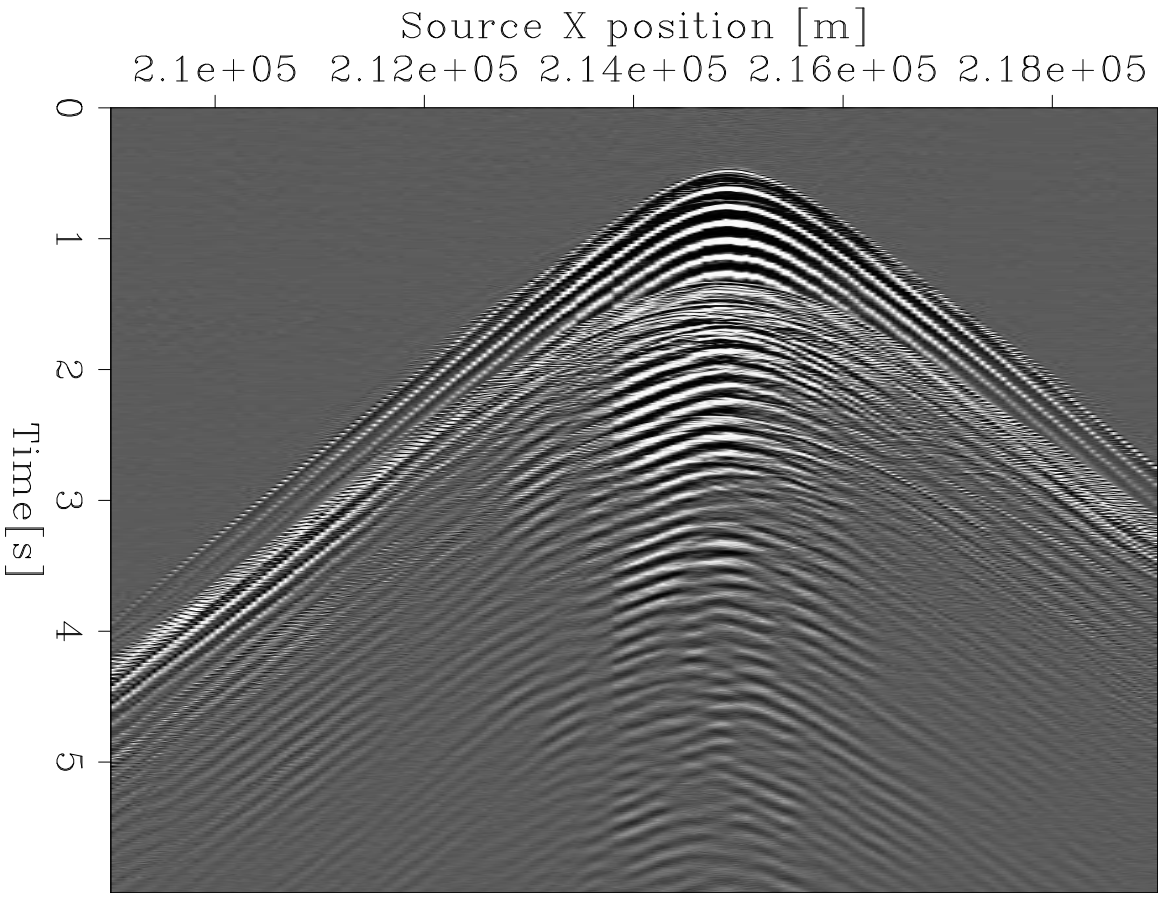


Figure 6: Downgoing component output from PZ-summation. [**CR**] taylor1/. downgoing

Debubble and wavelet shaping

Picking the data wavelet

Once we have the downgoing component of the hydrophone data separated, we can estimate a source wavelet from a near-offset subset of it (figure 7). First, we perform hyperbolic moveout (HMO) for each node gather to align the first arrival event across all offsets (figure 8). We can then stack across all offsets to find a single wavelet representative of that node gather (figure 10). To increase our averaging further, we can stack across all node gathers (figure 11). To do this however, we need to align the first arrival event in each node gather to a common time position ($t_0 = 0$ for example). This means shifting based on the relative depth of each of the nodes respectively. We perform this shifting and stacking across nodes to find a final wavelet representative of the average of the sources actually used. In this wavelet we notice the bubble signature as periodic, fading pulses.

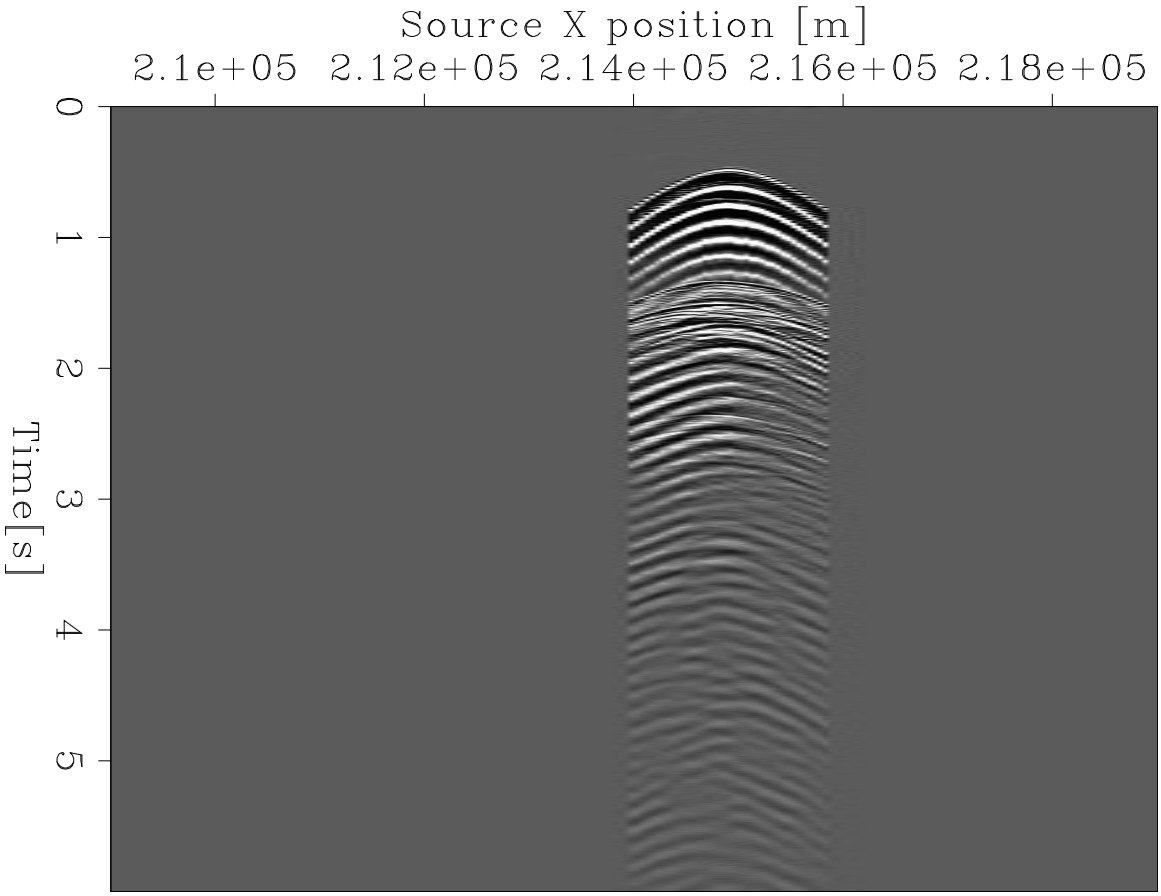


Figure 7: Near offset ($<$ 1000m) subset of figure 6 chosen for estimating observed source wavelet. [**CR**] taylor1/. UncorrectedNear

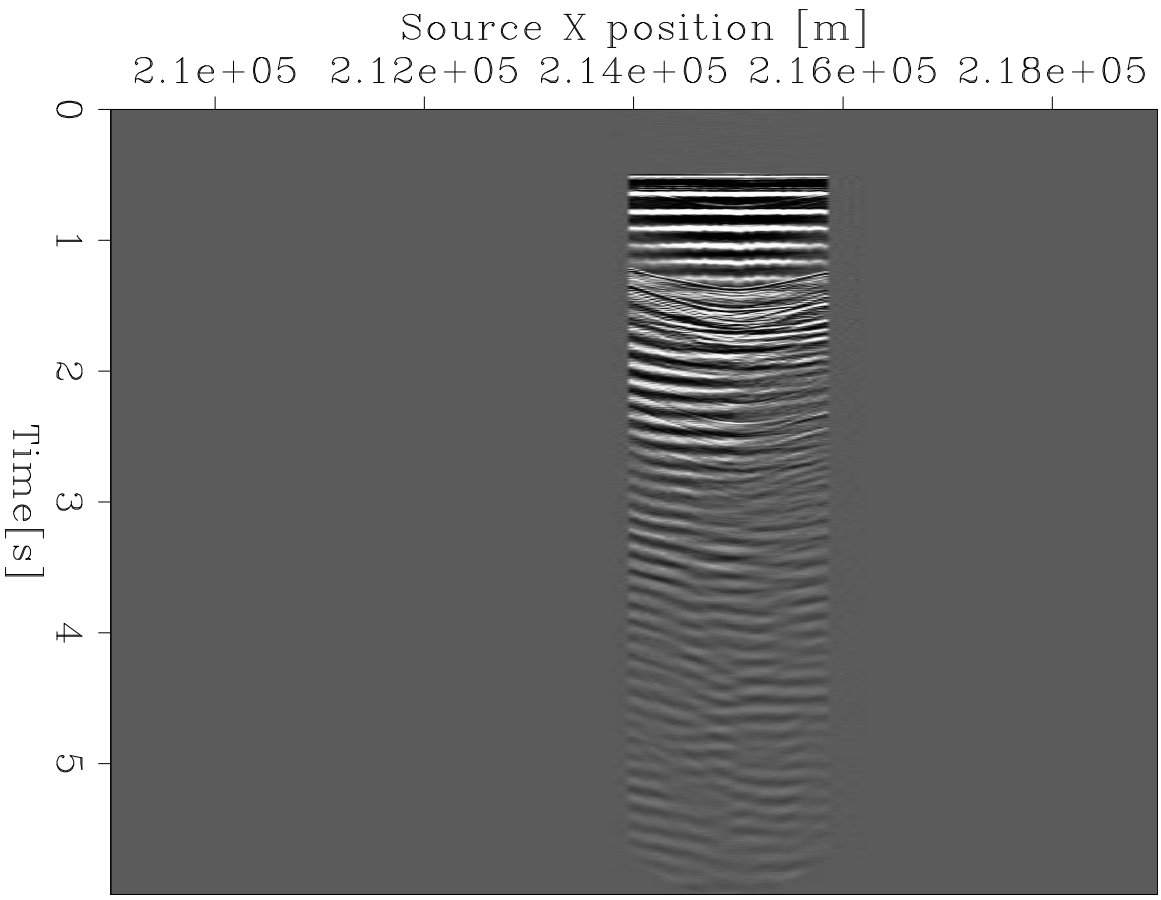


Figure 8: 1500 m/s HMO correction applied to data in figure 7. [**CR**] taylor1/. HMOcorrected

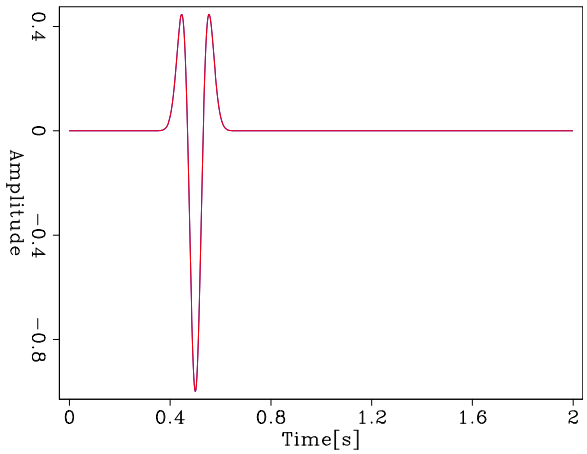


Figure 9: Wavelet used in synthetic data modeling. [**CR**] taylor1/. synWavelet

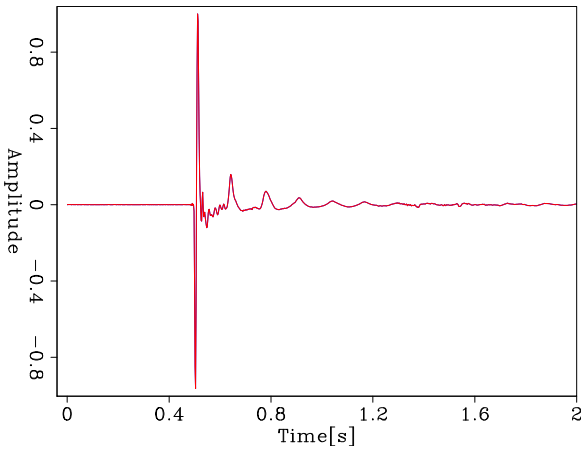


Figure 10: Wavelet produced from stacking across traces in figure 8. [**CR**] taylor1/. Node3StackedWavelet

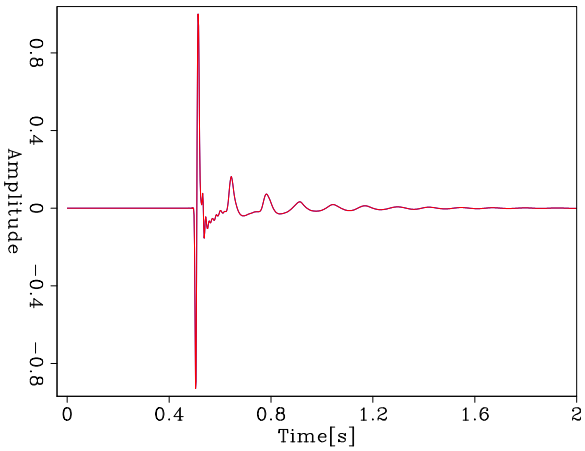


Figure 11: Observed source wavelet built from average of wavelets extracted from 381 node gathers (just as in figure 10). [**CR**] taylor1/. AverageStackedWavelet

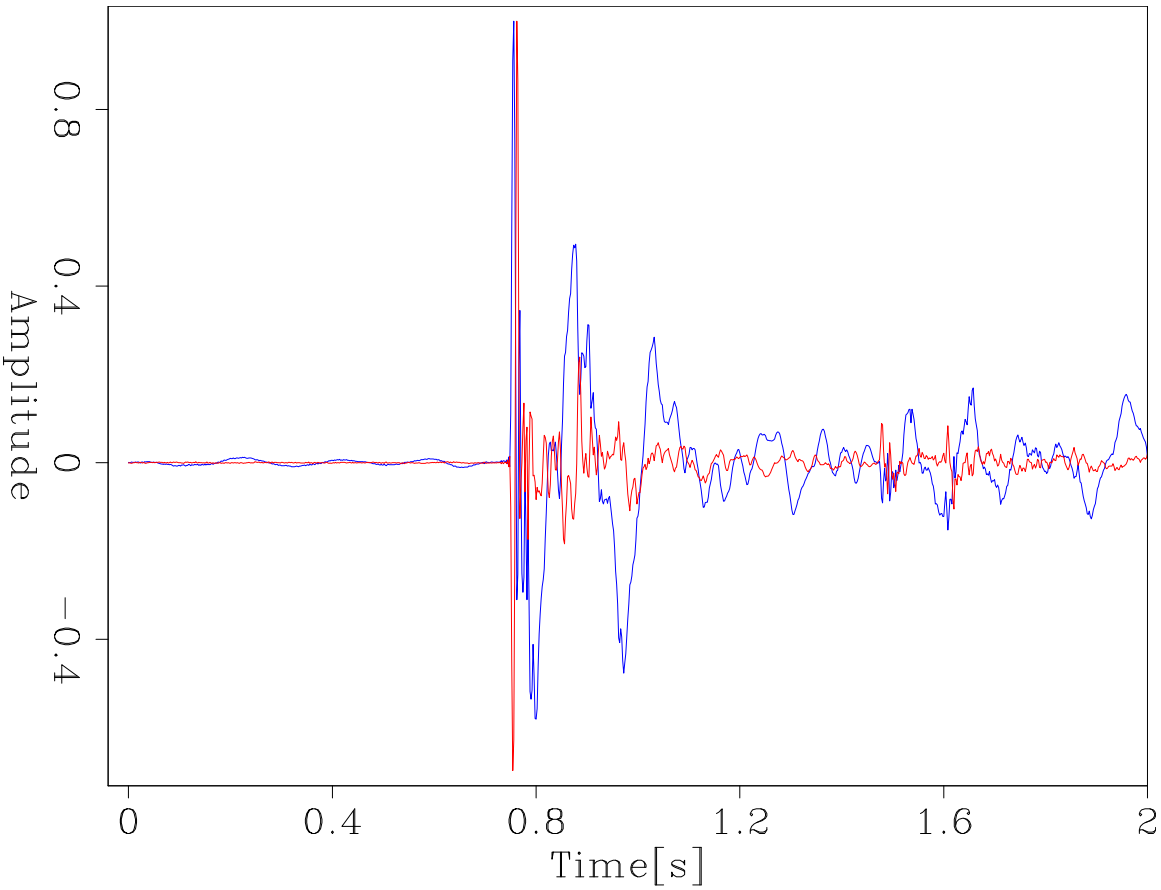


Figure 12: Example trace before (red) and after (blue) shaping filter applied. [**CR**] taylor1/. ShapedTraceCompare

Finding the shaping filter

In our synthetic data, we model using a simple 8[Hz] central frequency Ricker wavelet (see figure 9). When we compare against the averaged observed data wavelet (figure 11), we can see that there are some significant differences that warrant the use of a shaping filter. We estimate the filter that shapes the averaged observed data wavelet to the synthetic wavelet (see Yilmaz (1987)), and then apply to the full dataset (figures 13 and 14). When we compare figure 6 with figure 13, we can see that the bubble removal is effective, and that the frequency and phase content has become more similar to what we would expect given the lower frequency Ricker wavelet to which we match.

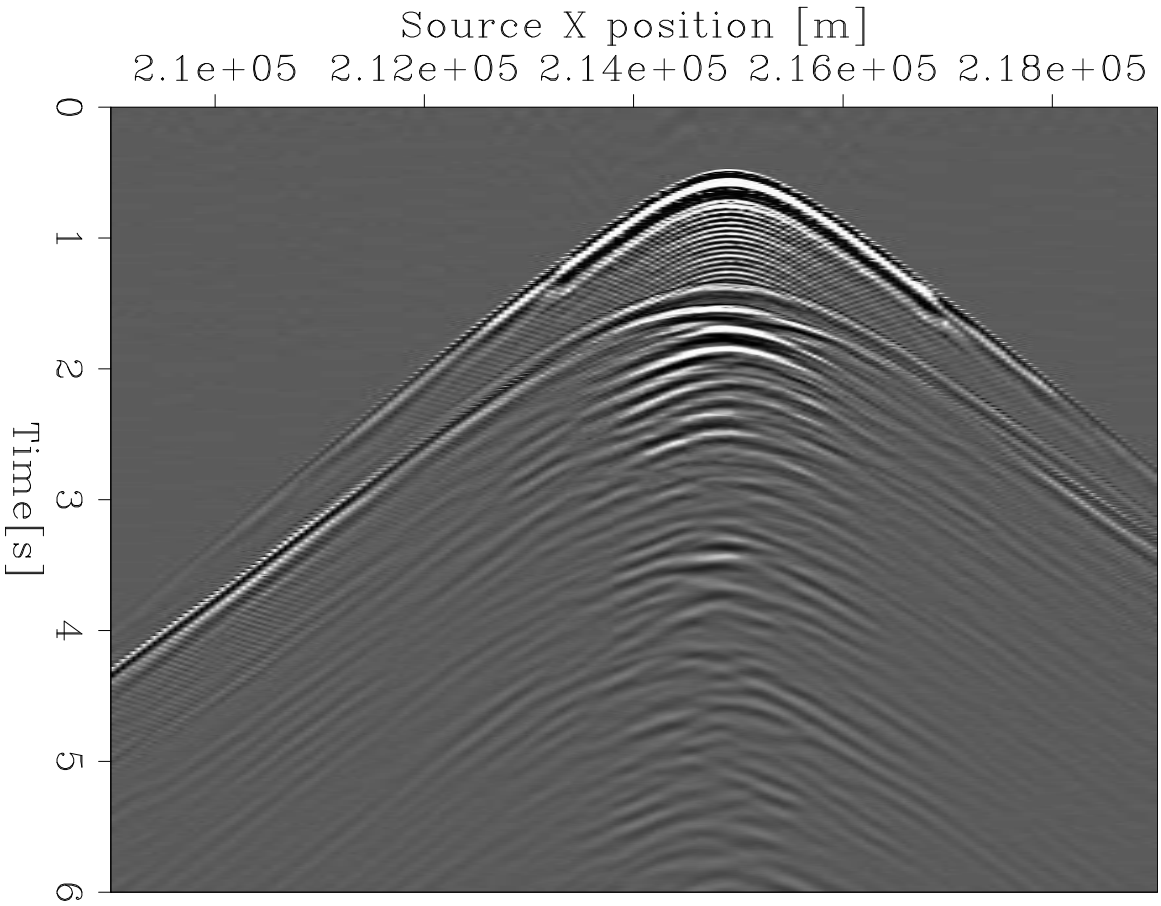


Figure 13: Downgoing hydrophone data after shaping filter applied. [**CR**] taylor1/. ShapedDOWNData

CONCLUSIONS

We applied a standard processing flow to the Shell OBN dataset that began with PZ-summation to separate the upgoing and downgoing components, which are more suitable for imaging purposes than the original pressure and vertical component data. A designature process was performed by reshaping the upgoing and downgoing components to a synthetic wavelet, which will be used for synthesizing data during FWI. We estimated the original

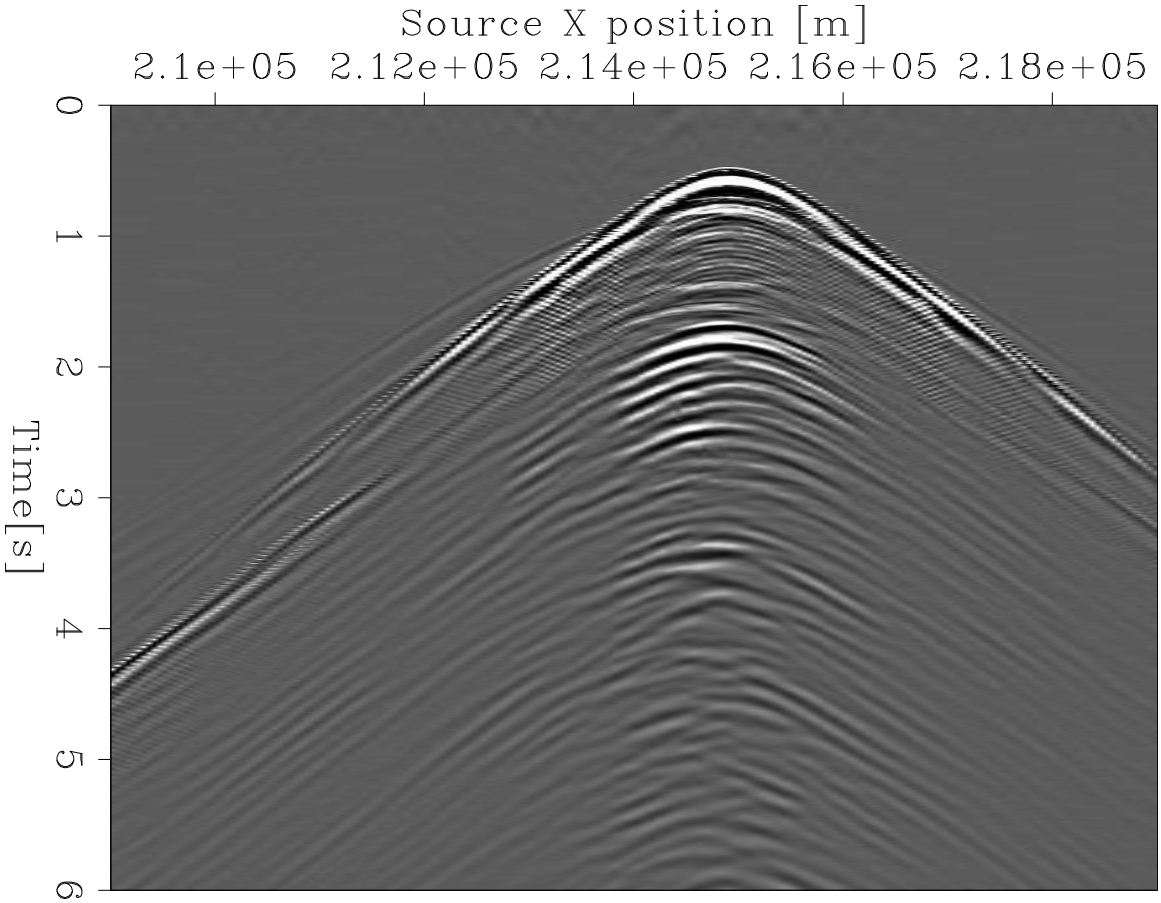


Figure 14: Upgoing hydrophone data after shaping filter applied. [**CR**] taylor1/. ShapedUPData

wavelet by applying HMO correction, and stacking near-offset subsets of the node gathers. The reshaped data are to ultimately be tested in FWI.

ACKNOWLEDGEMENT

Thanks to Shell Exploration, Inc. for the use of the OBN dataset used in this work.

REFERENCES

Biondi, E. and S. A. Levin, 2014, Application of the up-down separation using PZ calibration filter based on critically refracted waves: SEP-Report, **155**, 119–132.

Yilmaz, O., 1987, Seismic data processing: Society of Exploration Geophysicists.

Earthquake detection through cross-correlation of the Stanford Fiber Seismic Observatory (SFSO) data

Siyuan Yuan, Biondo Biondi and Robert G. Clapp

ABSTRACT

Earthquakes occurring in similar locations can have repeatable waveforms. We show that with the Stanford Fiber Seismic Observatory (SFSO) earthquake recordings of Ladera and Felt Lake earthquakes as templates, we can detect other earthquakes occurred in similar locations through cross-correlation. As a benchmark, we applied cross-correlation to the recordings of the nearby seismometer stations. We found that the cross-correlation results of the SFSO recordings tend to have higher signal-to-noise ratios (SNR) and fewer false positives compared with the seismometer results. We also show that the SFSO is more sensitive to differences in earthquake mechanisms occurring in Ladera area compared to the two broadband stations. Through cross-correlation of the SFSO data, we identified two Felt Lake earthquakes that were not recorded in the United States Geological Survey (USGS) online catalog. With Felt Lake event recordings, we computed the amplitude ratios of the template to the four detected events. The ratios are similar to those computed with broadband stations for the two relatively large events, which shows the potential of magnitude estimation with SFSO data. For other two events, that are too weak to be included in the USGS catalog, SFSO tends to underestimate the ratios.

INTRODUCTION

Distributed Acoustic Sensing (DAS) holds great promise for application in cost-effective monitoring of microseismic signals and detecting earthquakes. Unlike the mainstream work previously done using DAS (e.g., using fibers in wells or burying the array in trenches), the Stanford Fiber Seismic Observatory (SFSO) uses a fiber-optic cable laying in an already existing polyvinyl chloride (PVC) conduit buried in the ground that makes the installation more convenient and economic (Martin et al., 2017a; Biondi et al., 2017). Understanding how well this type of cost-effective array records seismic events is one of our first-order objectives. With a repeatability analysis, we showed that our SFSO DAS array records coherent signals from earthquakes and quarry blasts occurred in the similar locations and with presumably similar mechanisms (Biondi et al., 2017; Yuan et al., 2017; Martin et al., 2017b). Yuan et al. (2017) showed that with cross-correlation we can find a time shift to cancel out the waveforms of two Ladera earthquakes with coherent waveforms, which validates that the SFSO records coherent signals and demonstrates that cross-correlation can be potentially used as a template-matching approach to identify new earthquakes.

One major advantage of our DAS array is the cheap cost per sensor enabling a dense channel distribution. With hundreds of traces to average out the noise and enhance the signal, it has the potential, to some extent, to outperform 3C traditional seismometers with sparse channel distributions in detecting small and nearby earthquakes.

The cross-correlation amplitude between the events with similar epicenters and mechanisms can reflect the SNR level and sensitivity of our array to weak and nearby earthquakes. Li and Zhan (2018) demonstrate that with cross-correlation on SFSO data recorded in the Brady Hot Springs geothermal field, microseismicity below the noise level can be detected. Herein, we investigate if through cross-correlation of the SFSO recordings, we can detect events with similar epicenters and mechanisms. Then, to understand how well the detected events are correlated with the templates, we compare the cross-correlation results with those obtained from two nearby broadband stations - Jasper Ridge Seismic Station (JRSC) and Stanford Telescope Station (JSFB). The comparison of the correlated value can show the SNR level and the sensitivity of our array relative to the broadband stations. In the end, we compute the amplitude ratios of the Felt Lake template to each of the four detected Felt Lake events. We compare the ratios with the results obtained from the data of JRSC and JSFB to illustrate whether SFSO recordings can be effectively used to estimate the magnitude of the nearby weak events.

RECORDINGS OF THE LADERA AND FELT LAKE EARTHQUAKES

Martin et al. (2017a) provide most of the relevant information on the SFSO experiment. Figure 1 shows a simplified geometry of the SFSO array, namely six segments spreading out into two directions, colored in red and blue, respectively. The data displayed in the following sections have channels in same segments grouped together to see the move-out in each segment more conveniently. The channel spacing is 4 meters and the sampling rate is 50 samples per second. According to the USGS online catalog, two places close to the Stanford campus - Felt Lake and Ladera - have three and five small earthquakes, respectively since our SFSO array started recording. The following list provides the main information for the eight events - *Lad#1* through *Fl#3*. The event time (UTC), magnitude (M) and depth (z) of each event are based on the online USGS database. Distance (Δ) to our SFSO array of each event is calculated based on the 3D distance between the SFSO and the epicenter. *Fl#4* and *Fl#5* are two events detected by cross-correlation as described in the following sections. Both of them occurred in Felt Lake area. The event times for these two events are from the cross-correlation results. Since they are not included by the USGS online catalog, no metadata of the two events are available to us.

- *Lad#1*: 2017-08-10 03:19:47 – M1.79 – z=+3.25 km – Δ=4.62 km
- *Lad#2*: 2017-08-10 02:53:50 – M1.63 – z=+3.63 km – Δ=4.62 km
- *Lad#3*: 2017-11-27 15:30:45 – M1.94 – z=+3.64 km – Δ=5.37 km
- *Lad#4*: 2018-01-23 17:30:54 – M1.66 – z=+4.01 km – Δ=5.38 km
- *Lad#5*: 2017-01-11 23:04:40 – M2.03 – z=+4.15 km – Δ=5.75 km
- *Fl#1*: 2017-07-12 18:46:41 – M1.34 – z=+3.24 km – Δ=5.45 km
- *Fl#2*: 2017-07-12 18:47:50 – M0.95 – z=+3.05 km – Δ=5.34 km
- *Fl#3*: 2017-07-13 04:02:49 – M0.81 – z=+3.64 km – Δ=5.72 km

- *Fl#4*: 2017-05-10 06:35:38 – M???? – z=????? km – Δ=???? km
- *Fl#5*: 2017-07-13 05:56:06 – M???? – z=????? km – Δ=???? km

Figure 2 shows the locations of the eight epicenters and rough distances between the SFSO and Felt Lake and Ladera.

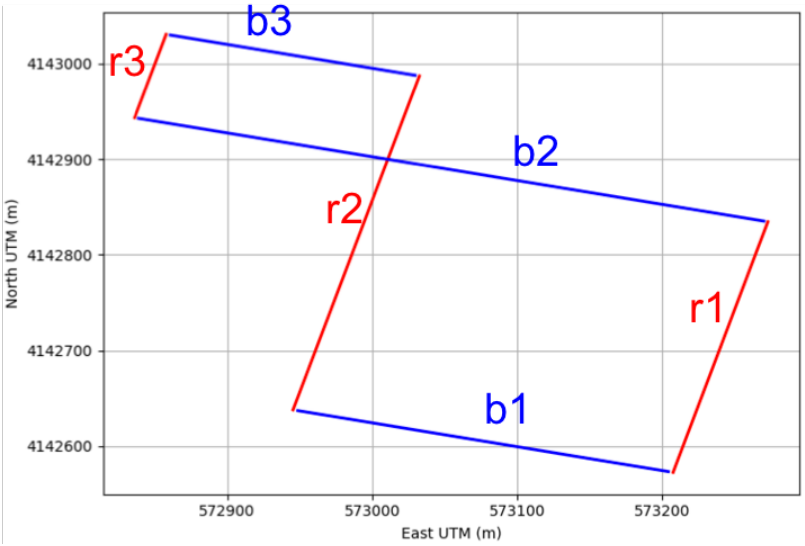


Figure 1: The layout of the simplified SFSO channel map. The fibers are extended into two directions colored in blue and red, respectively. 534 channels are distributed with a spacing of 4 meters along each line in the plot. The three vertical segments in the map are denoted by letters from r1 to r3. The three horizontal segments are denoted by letters from b1 to b3. [**ER**] siyuan1/. sim-map

Figures 3 and 4 show the recordings of our SFSO array along with the vertical components of the two broadband stations, JRSC and JSFB, respectively. The SFSO recordings are shown in six segments separated by red and blue dash lines and a black line separating the two directions of fibers. The labels of the segments correspond to the labels shown in Figure 1. Zero time of the time axis corresponds to the event occurring time according to the USGS. Because the locations of the stations are different and the near-surface conditions and ray paths are different, the waveforms are not directly comparable. However, JRSC and JSFB data provide a rough indication of the arrival time and relative strength of the signal corresponding to different arrivals. Herein, we used *Lad#1* and *Fl#1* as templates to perform cross-correlation since they have the clearest waveforms among the other recordings cataloged by USGS. We chose 10-second time windows to include the waveforms and noise before and after the wave arrival to capture the characteristics of the waveforms and energy changes before and after the events arrive.

CROSS-CORRELATION

The first step is to develop an appropriate way to perform cross-correlation. We tried the events in the USGS catalog, which are listed in the previous section, to see if cross-correlation can detect them with enough confidence. SFSO data shown in Figure 3 and

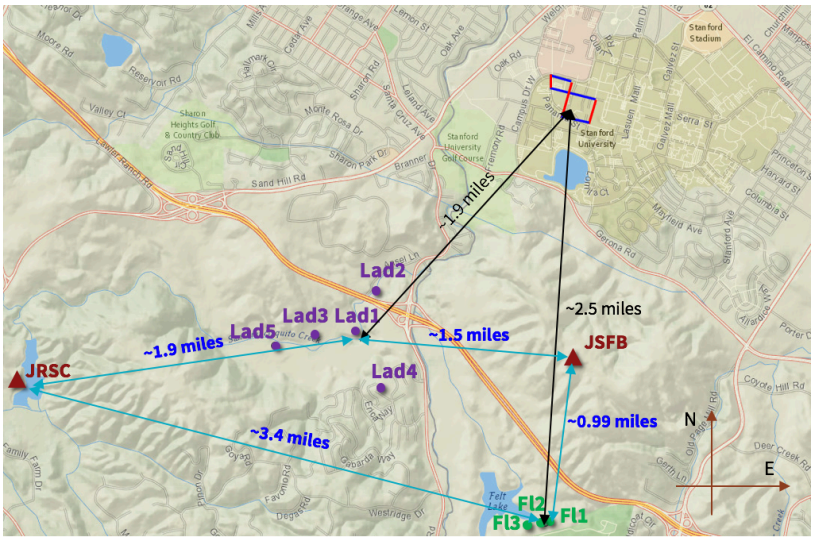


Figure 2: Epicenters of five Ladera earthquakes and three Felt Lake earthquakes are labeled on the map in purple and green, respectively. The distance from the SFSO array to the Ladera area is 1.9 miles, while the distance to the Felt Lake is 2.5 miles. The brown triangles show the locations of the two broadband stations, JRSC and JSFB. The distances from the broadband stations to the epicenters are indicated in the plot. [**NR**] siyuan1/. epicenters-loc

Figure 4 can be thought of as two-dimensional (2D) matrices, with a time axis and a channel number axis. We first performed 2D cross-correlation: looping a 10-second time window through the continuous data and computing the summation of the element-wise product of the 2D matrices of the time window and the template. The operation is known as matrix Frobenius inner product. Then the product is normalized with respect to the Frobenius norm of the two matrices to only consider the patterns or the shape of the data instead of the absolute value. With the normalization, the cross-correlation value ends up being -1 to 1. For comparison, we performed one-dimensional (1D) cross-correlation on the vertical component of the broadband data because the vertical component has the highest SNR among all the other components. However, the 2D cross-correlation of the SFSO data produces great false positives. Figure 5(a) shows the cross-correlation of the Ladera event template on the continuous data recorded on August 10, 2017. From the previous section, we see that there is a weaker Ladera event with a magnitude of 1.6 (*Lad#2*), 26 minutes before the template event. The peak with an amplitude of 1 corresponds to the template. The other peak on the left with amplitude approximately 0.7 corresponds to the *Lad#2*. *Lad#2* stands out from the noise. But we also notice there are a few small spikes with amplitude approximately 0.1. It turns out that those spikes correspond to noise data instead of events. From Figure 6(b) and Figure 6(c), which show the cross-correlation of broadband stations on the same time range, we do not see those small spikes at the corresponding time lags as we see in the SFSO results. The reason why these small spikes show up is because 2D correlation can be considered as the average of the trace-wise correlation which can be affected by the outliers of the trace-wise correlations. Those small spikes may correspond to the time window with high coherent noise in some traces, acting

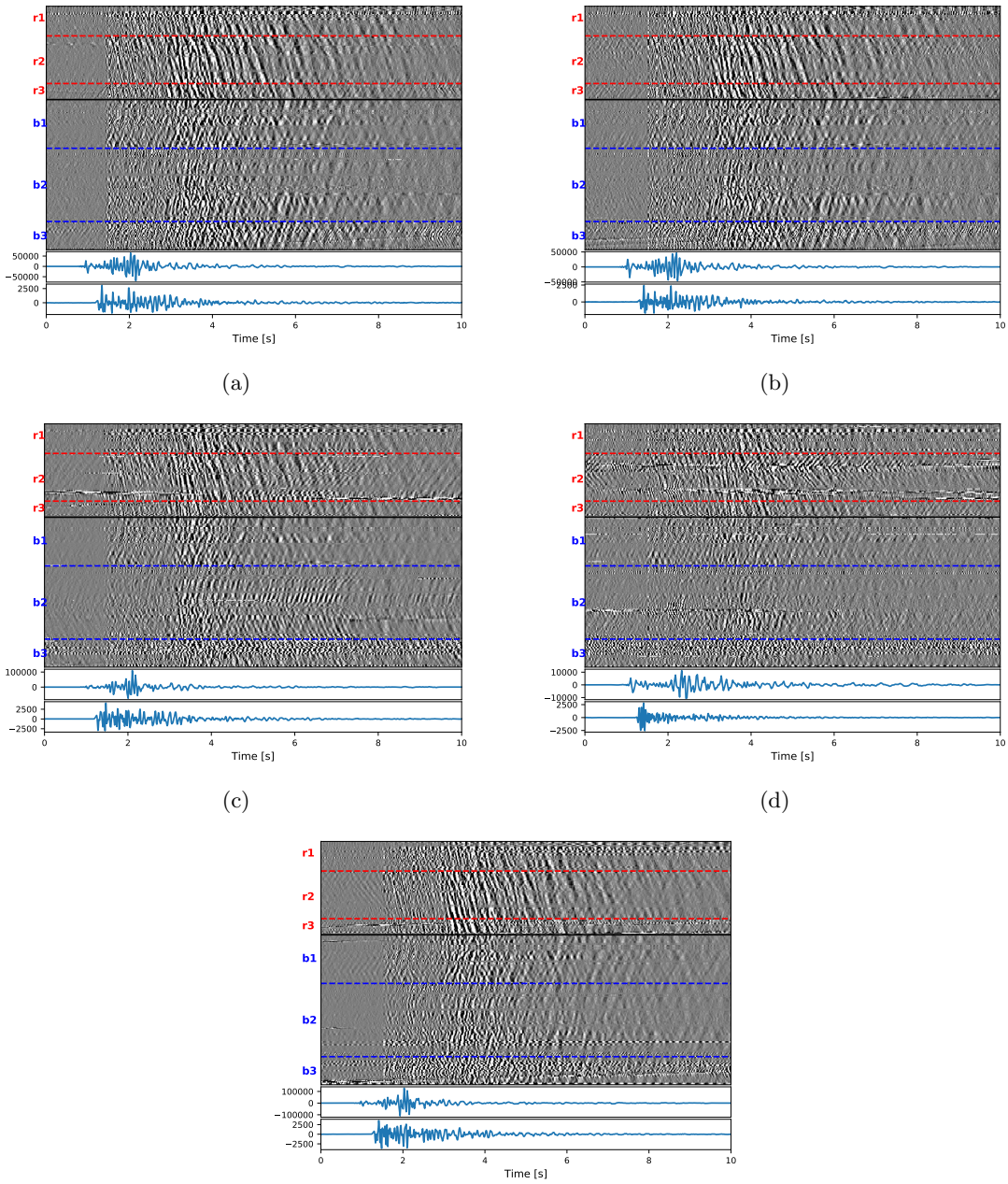


Figure 3: (a) to (e) show the data from *Lad#1* to *Lad#5* , respectively. The data shown in (a) is the template we use to perform cross-correlation. In each panel, a 2D plot shows our SFSO recording which is shown in six segments separated by red and blue dash lines and a black line separating the two directions of fibers. Each of the SFSO recordings has been clipped independently at 98 percentile. The vertical component of JRSC highpassed by 1.5 Hz is shown immediately following the SFSO data. The vertical component of the JSFB is shown in the bottom. Zero time on the time axis corresponds to the event occurring time according to the USGS. [**ER**] siyuan1/. lad1-data,lad2-data,lad3-data,lad4-data,lad5-data

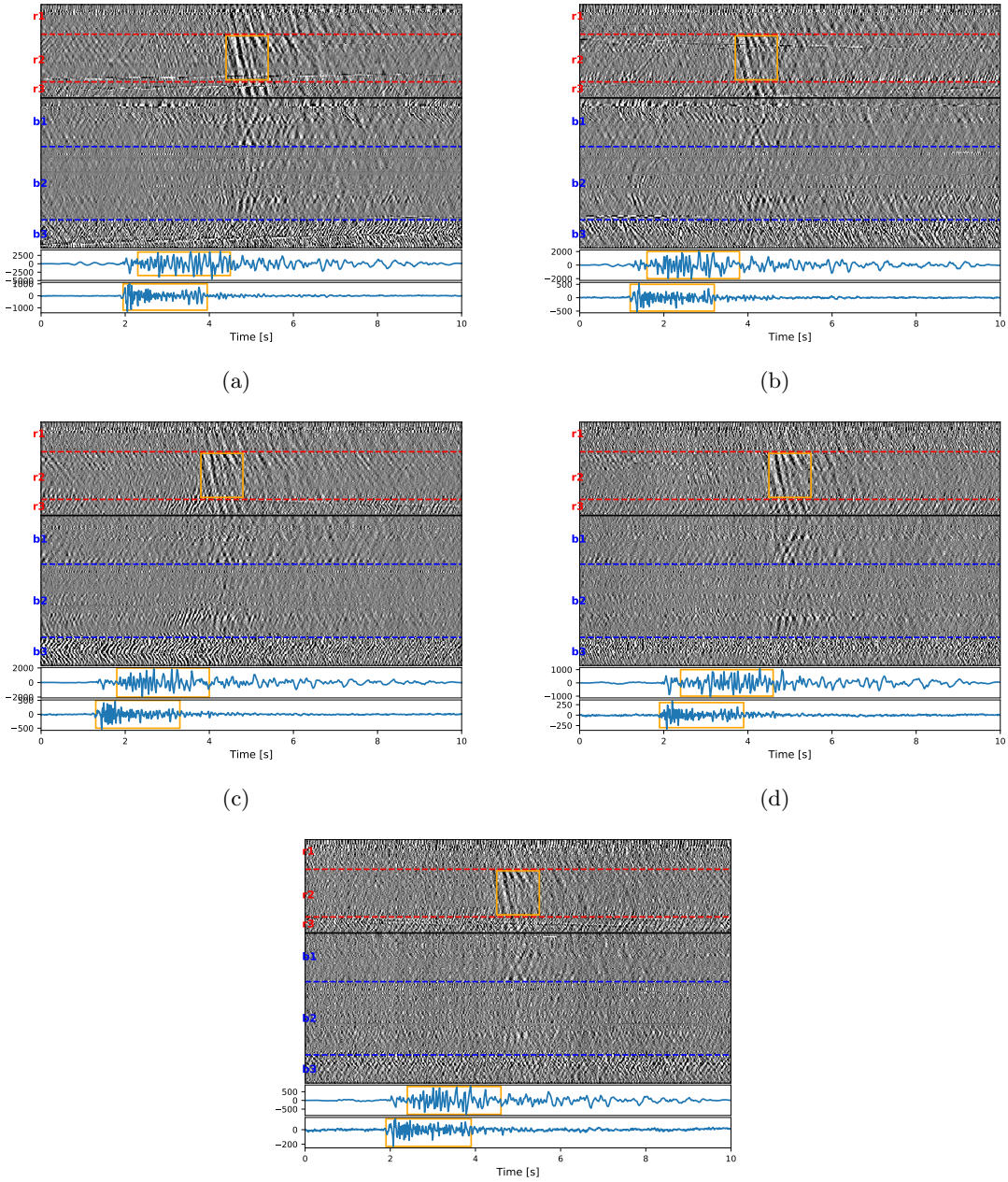
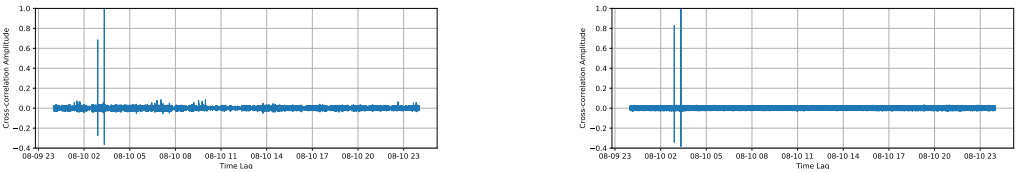


Figure 4: (a) to (e) show the data for *Fl#1*, *Fl#2*, *Fl#3*, *Fl#4* and *Fl#5*, respectively. The data shown in (a) is the template we use to perform cross-correlation. In each panel, a 2D plot shows our SFSO recording which is shown in six segments separated by red and dash lines and a black line separating the two directions of fibers. Each of the SFSO recordings has been clipped independently at 98 percentile. The vertical component of JRSC highpassed by 1.5 Hz is shown immediately following the SFSO data. The vertical component of the JSFB is shown in the bottom. Zero time on the time axis corresponds to the event occurring time according to USGS. The data surrounded by orange rectangles are used to compute the amplitude ratios as described in following sections. [**ER**] siyuan1/. fl1-data-rec,fl2-data-rec,fl3-data-rec,fl4-data-rec,fl5-data-rec

as outliers to pull up the 2D correlation results. Thus, instead of performing a 2D cross-correlation, we performed a trace-wise 1D cross-correlation; and then, we picked the median of those values to be more robust to outliers while at the same time taking advantage of the dense channel distribution of our SFSO array. The results of picking the median of trace-wise cross-correlation at each time lag are shown in Figure 5(b). The evaluate the cross-correlation results here and in later discussions, we define the **average noise level** of cross-correlation results to be the 99.95 percentile of the absolute values of the cross-correlation of the template event with the whole 24-hour continuous recordings. And we define the **signal level** of cross-correlation to be the peak value of cross-correlation between template and another event occurring in similar location and with similar mechanisms (e.g. peak value of 0.8 of the spike corresponding to *Lad#2* in Figure 5(b)). The **SNR** is defined as the ratio between the signal level to the average noise level. As can be seen, the SNR is evidently higher compared with Figure 5(a). Thus, in the later analysis, we adopted the median cross-correlation method.

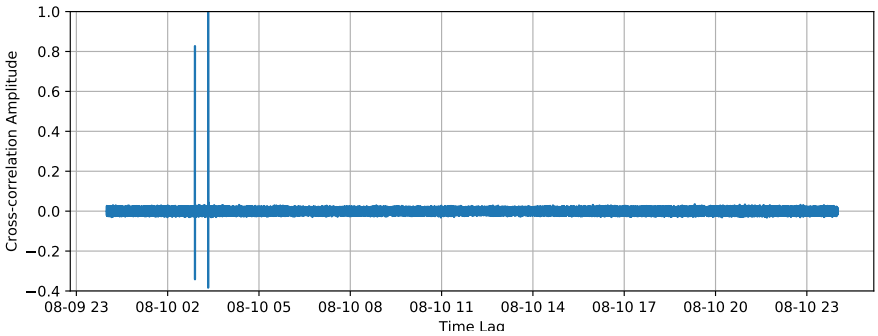


(a) (b)

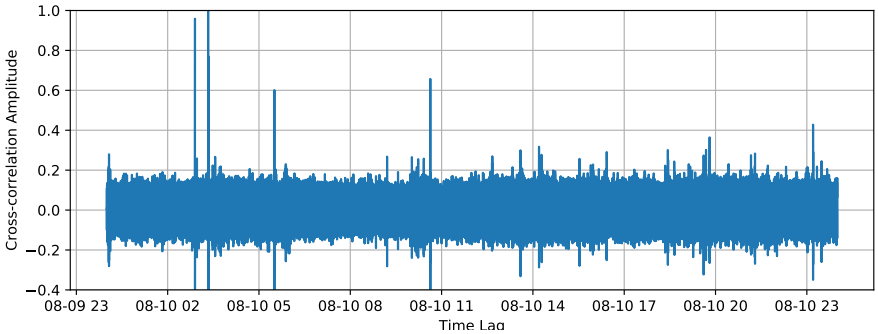
Figure 5: (a) 2D cross-correlation results between the *Lad#1* data and the continuous recordings on August 10, 2017. The spike with amplitude of 1 corresponds to the *Lad#1*. The other spike with amplitude of 0.7 corresponds to *Lad#2*. *Lad#2* does stand out from the noise. However, there are a few weak spikes that are false positives. (b) results by picking the median of trace-wise cross-correlations at each time lag. The SNR increases evidently. The false positive spikes shown in the 2D cross-correlation results are not apparent. [**ER**] siyuan1/. lad-xcorr-170810-2DCorr,lad-xcorr-170810

CROSS-CORRELATION WITH THE LADERA TEMPLATE

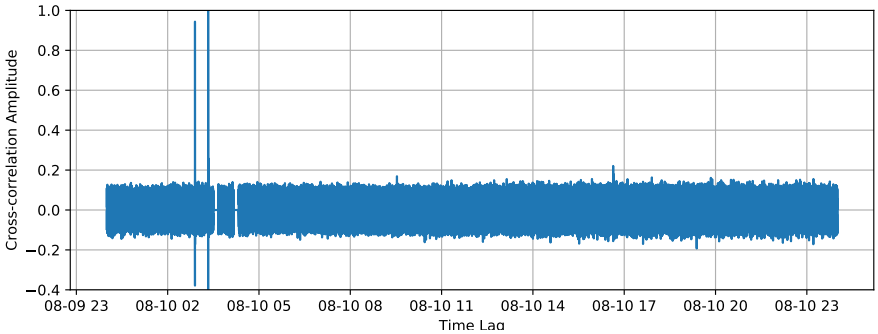
Figure 6 shows the comparison between the cross-correlation of SFSO and the two broadband stations for the data recorded on August 10, 2017, when *Lad#1* and *Lad#2* occurred. All three instruments have *Lad#2* detected at the right timing. *Lad#2* corresponds to the second strongest spike in each panel. As can be seen, JRSC has a few spikes that are false positives and are not detectable in the other two panels. Both broadband stations have a higher amplitude of *Lad#2* than SFSO does. However, their noise levels are also higher compared with the SFSO results. For JRSC, the average noise amplitude is approximately 0.121, while the amplitude corresponding to *Lad#2* is approximately 0.9588. SNR for JRSC is approximately 7.90. For JSFB, the average noise amplitude is approximately 0.090, while the amplitude corresponding to *Lad#2* is approximately 0.944. SNR for JSFB is approximately 10.53. For our SFSO array, the average noise amplitude is approximately 0.018, while the signal amplitude is approximately 0.827. SNR for SFSO is approximately 45.97. We can see that with the hundreds of channels to take the median of the results, SFSO array produces higher SNR while at the same time with a less false positive rate compared with the broadband stations.



(a)



(b)



(c)

Figure 6: Cross-correlation results of the recordings on August 10, 2017, the day both *Lad#1* (Figure 3(a)) and *Lad#2* (Figure 3(b)) occurred. (a) shows the cross-correlation results of SFSO data; (b) shows the results of the JRSC data. We can see a few false positives; (c) shows the results of JSFB data. In each panel, the spike with amplitude of one corresponds to *Lad#1*, which is the template. The other major spike corresponds to *Lad#2*. The SNRs for SFSO, JRSC and JSFB are 45.97, 7.90 and 10.53 respectively. [**ER**] siyuan1/. lad-xcorr-170810-2,JRSC-lad-xcorr-170810,JSFB-lad-xcorr-170810

Figure 7 compares the cross-correlation results on November 27, 2017, the day when *Lad#3* occurred. JRSC is quieter this time and has SNR approximately 16.96. SNRs for SFSO and JSFB are 16.17 and 5.17, respectively. In this case, JRSC performs the best and our SFSO array also performs well. From Figure 3(c), we see that the SFSO recording overall has good signal detected. However, the noise level is stronger in a few channels compared to the SFSO recording of *Fl#1* shown in Figure 3(a). The higher ambient noise level could explain why the SNR of SFSO results is lower than that obtained from JRSC.

Figure 8 compares the cross-correlation results on January 23, 2018, the day when *Lad#4* (Figure 3(d)) occurred. Surprisingly, as Figure 8(a) shows, no signal is detected by the SFSO array. SNRs for JRSC and JSFB are 5.49 and 4.22, respectively. From visually inspecting Figure 3, we can see that the SFSO and JRSF recordings of *Lad#4* shown in Figure 3(d) have evidently different waveforms compared to the other four events. Thus, it would be reasonable to think of *Lad#4* as a event that originated in the Ladera area but with different mechanisms from the other four events. And from Figure 2, we can see the epicenter of *Lad#4* is an outlier of those Ladera earthquake epicenters, which could also imply that the *Lad#4* has different mechanisms. If this is the case, we can say that our SFSO array is most sensitive to the mechanisms of earthquakes, while JRSC is the least sensitive one.

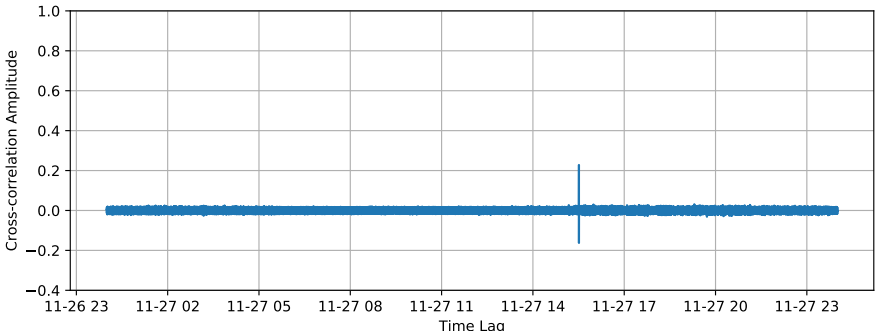
Figure 9 compares the cross-correlation results on January 11, 2017, the day when *Lad#5* occurred. SNRs for the SFSO array, JRSC and JSFB are 14.37, 15.40 and 4.66, respectively. JSFB has several false positive spikes showing up.

CROSS-CORRELATION WITH THE FELT LAKE TEMPLATE

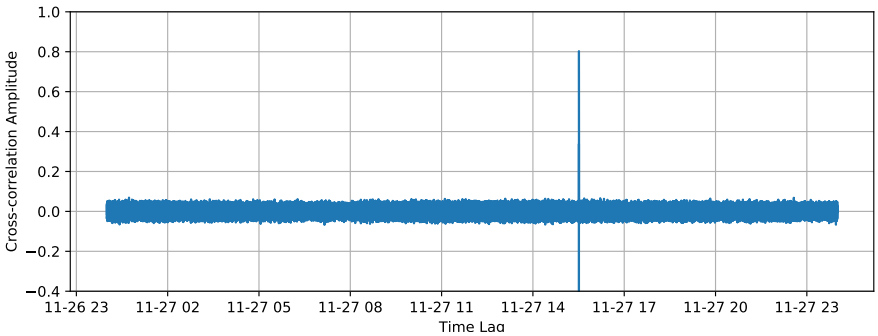
Figure 10 and Figure 11 show the cross-correlation results of the continuous recordings on July 12, 2017, and July 13, 2017. The two strongest spikes in each panel of Figure 10 correspond to the *Fl#1* and *Fl#2*, respectively. For *Fl#2*, the SNRs of the SFSO array, JRSC and JSFB are approximately 5.39, 3.64, and 6.30 respectively. Our SFSO array has the highest SNR among the three. JSFB has a clear false positive spike approximately at 18:42:00 UTC Time.

In Figure 11, there are two evident spikes in each panel. The two spikes appear in coherent timings across the three instruments. They are all true positives. The spike on the left corresponds to the *Fl#3*. The SNRs for *Fl#3* of SFSO, JRSC and JSFB are 5.06, 3.64 and 3.68 respectively. SFSO has the highest SNR value. The right spike is an event that is not recorded in the USGS catalog. We call it an aftershock (*Fl#5*) of the Felt Lake earthquakes. The SNRs of SFSO, JRSC and JSFB are 3.53, 3.82 and 4.86 respectively. Figure 4(e) shows the SFSO recording and the vertical components of JRSC and JSFB of *Fl#5*.

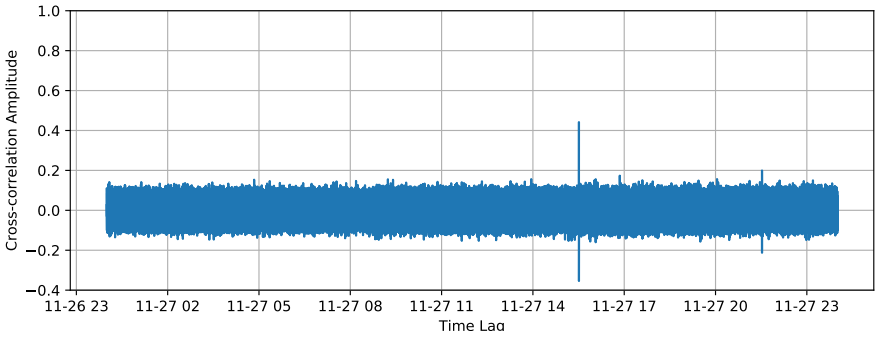
In addition to the aftershock event, we found another weak Felt Lake event on May 10, 2017, which is not recorded by the USGS online catalog. Since it occurs before the major Felt Lake earthquake, *Fl#1*, we call it a precursor (*Fl#4*) of the Felt Lake earthquake. Figure 12 shows the cross-correlation results of the SFSO data and the vertical components of the two broadband stations. The spike shown in the SFSO results corresponds to the precursor. In this case, SNRs for SFSO, JRSC and JSFB are 5.56, 4.14 and 4.00 respectively. As can be seen, the SFSO results have the best SNR and no clear false positives.



(a)

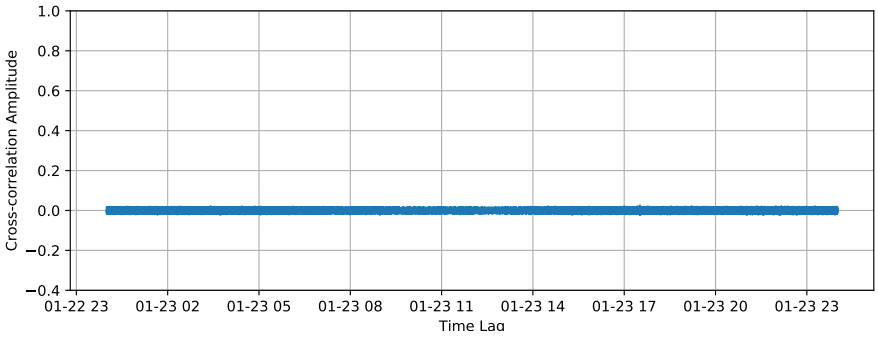


(b)

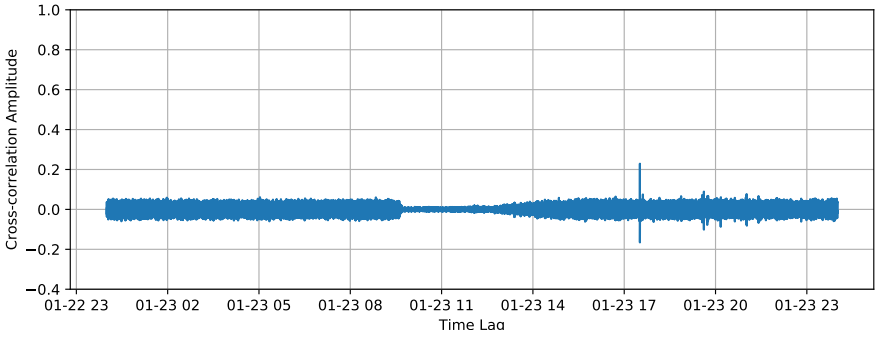


(c)

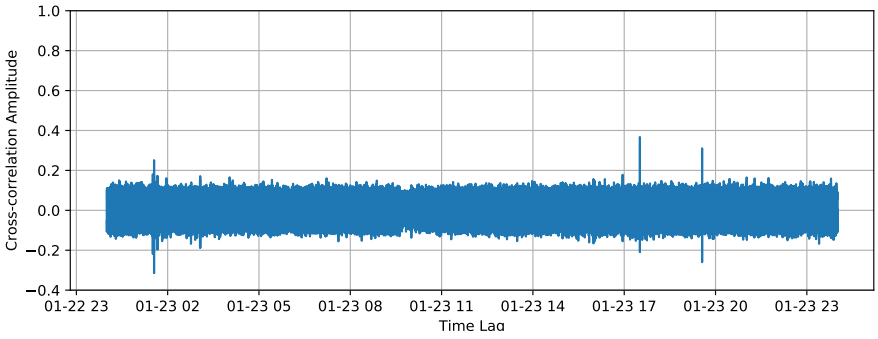
Figure 7: Cross-correlation results on November 27, 2017, the day when *Lad#3* (Figure 3(c)) occurred. (a) shows the cross-correlation results of SFSO data; (b) shows the results of the JRSC data; (c) shows the results of JSFB data. The SNRs for SFSO, JRSC and JSFB are 16.17, 16.96 and 5.17 respectively. [**ER**] siyuan1/. lad-xcorr-171127,JRSC-lad-xcorr-171127,JSFB-lad-xcorr-171127



(a)

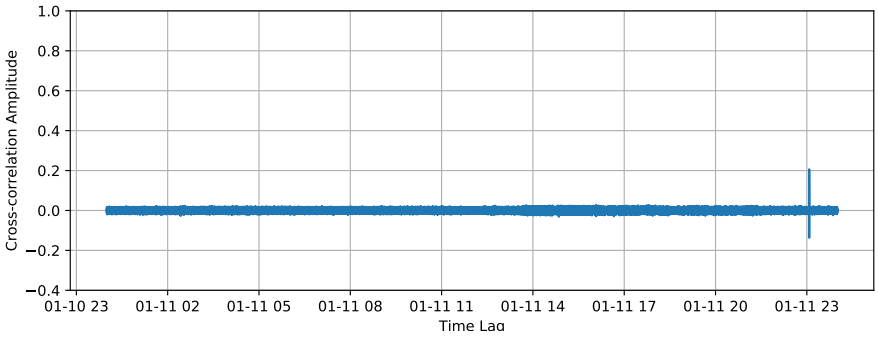


(b)

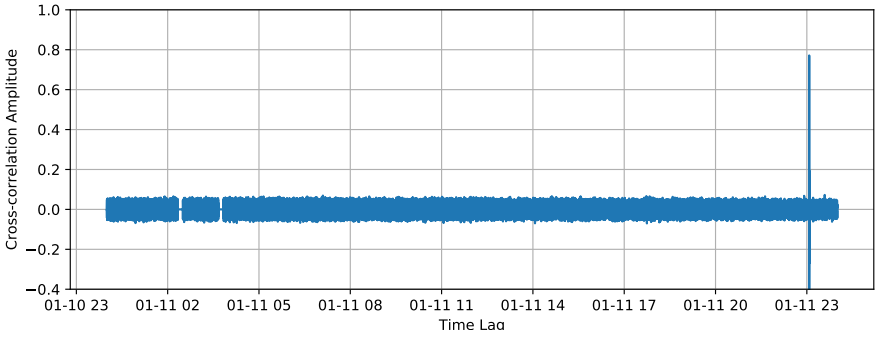


(c)

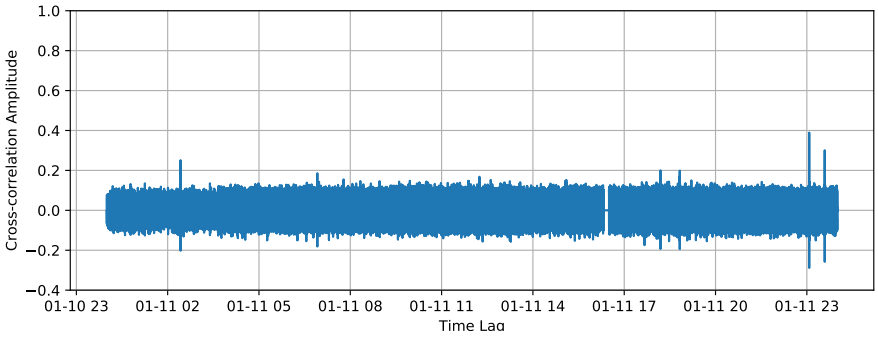
Figure 8: Cross-correlation results on January 23, 2018, the day when *Lad#4* (Figure 3(d)) occurred. (a) shows the cross-correlation results of SFSO data; (b) shows the results of the JRSC data; (c) shows the results of JSFB data. SFSO results have no clear signal detected. The SNRs for JRSC and JSFB are 5.49 and 4.22 respectively. [**ER**] siyuan1/. lad-xcorr-180123,JRSC-lad-xcorr-180123,JSFB-lad-xcorr-180123



(a)

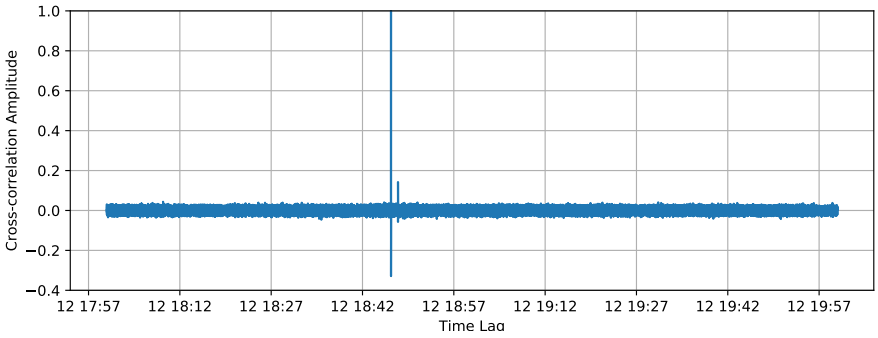


(b)

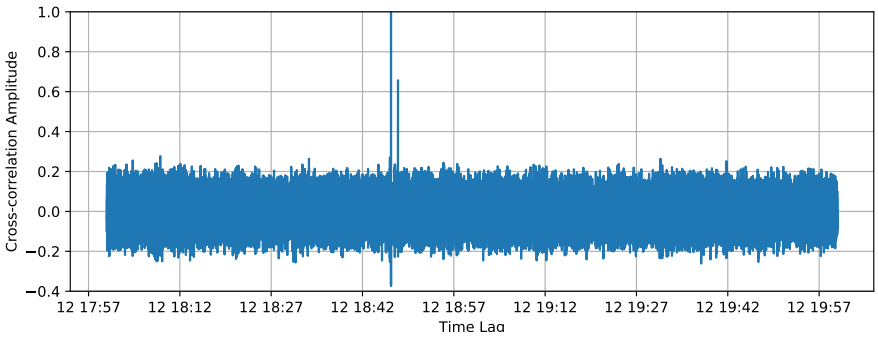


(c)

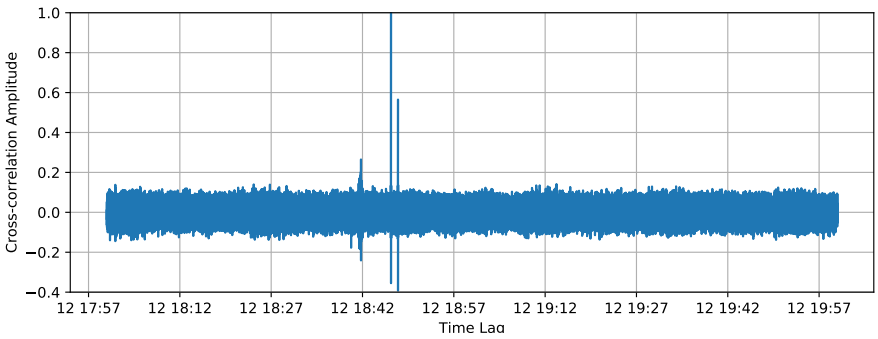
Figure 9: Cross-correlation results on January 11, 2017, the day when *Lad#5* (Figure 3(e)) occurred. (a) shows the cross-correlation results of SFSO data; (b) shows the results of the JRSC data; (c) shows the results of JSFB data. The SNRs for SFSO, JRSC and JSFB are 14.37, 15.40 and 4.66 respectively. [**ER**] siyuan1/. lad-xcorr-170111,JRSC-lad-xcorr-170111,JSFB-lad-xcorr-170111



(a)

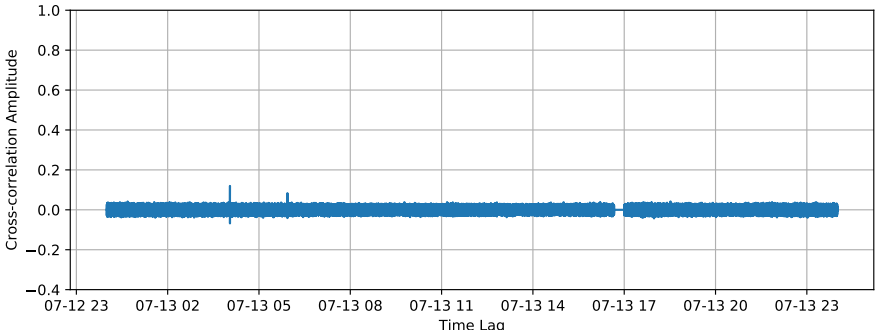


(b)

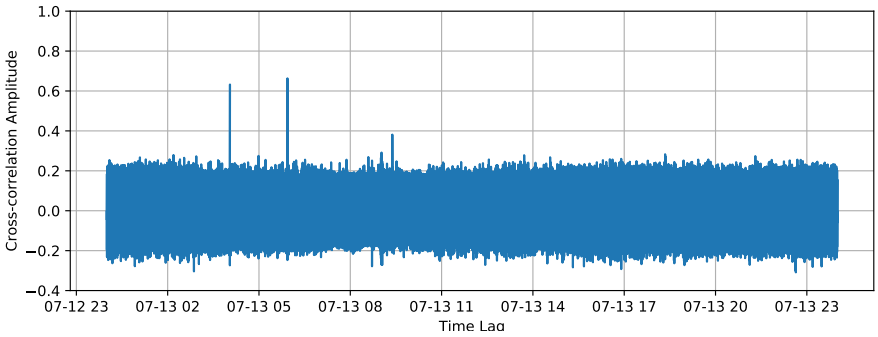


(c)

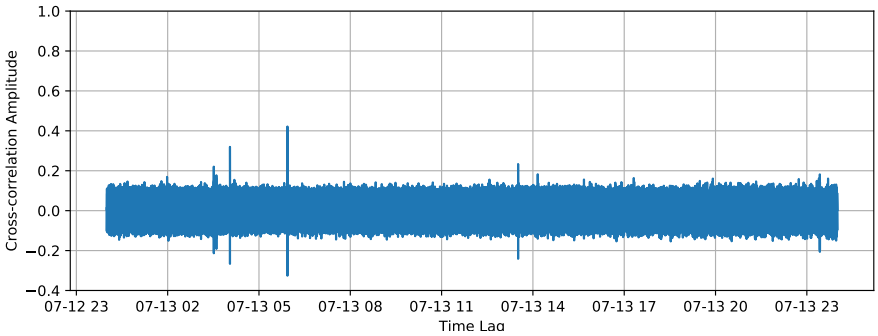
Figure 10: Cross-correlation results on July 12, 2017, the day when *Fl#1* (Figure 4(a)) and *Fl#2* (Figure 4(b)) occurred. (a) shows the cross-correlation results of SFSO data; (b) shows the results of the JRSC data; (c) shows the results of JSFB data. For *Fl#2*, SNRs for SFSO, JRSC and JSFB are 5.39, 3.64 and 6.30 respectively. [**ER**] siyuan1/. fl-xcorr-170712,JRSC-fl-xcorr-170712,JSFB-fl-xcorr-170712



(a)

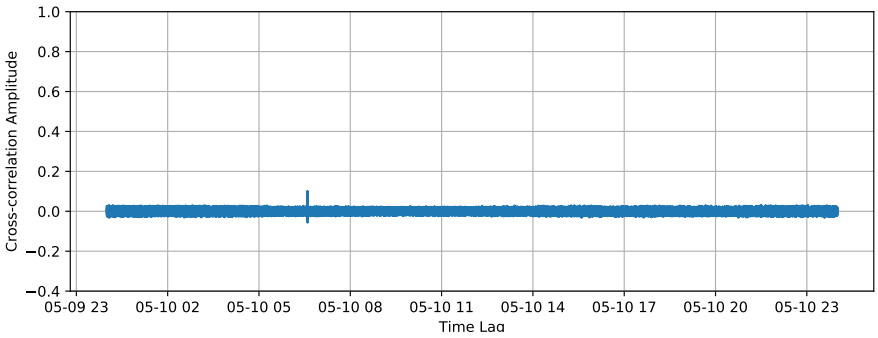


(b)

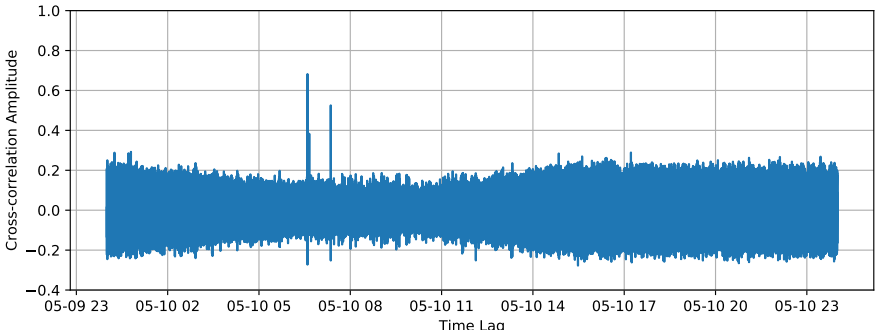


(c)

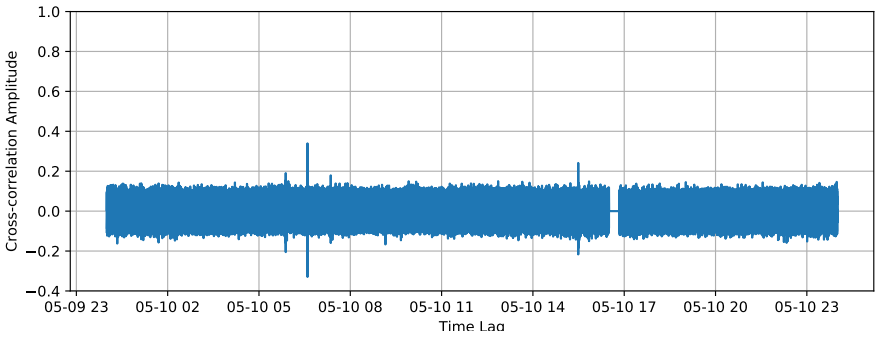
Figure 11: Cross-correlation results on July 13, 2017, the day when *Fl#3* (Figure 4(c)) occurred. (a) shows the cross-correlation results of SFSO data; (b) shows the results of the JRSC data; (c) shows the results of JSFB data. There are two spikes shown in (a), the left one corresponds to *Fl#3*. The other one corresponds to the detected aftershock, *Fl#5*(Figure 4(e)). For *Fl#3*, SNRs for SFSO, JRSC and JSFB are 5.06, 3.64 and 3.68 respectively. For *Fl#5*, SNRs for SFSO, JRSC and JSFB are 3.53, 3.82 and 4.86 respectively. [**ER**] siyuan1/. fl-xcorr-170713,JRSC-fl-xcorr-170713,JSFB-fl-xcorr-170713



(a)



(b)



(c)

Figure 12: Cross-correlation results on May 10, 2017, the day when the detected precursor, *Fl#4* (Figure 4(d)), occurred. (a) shows the cross-correlation results of SFSO data; (b) shows the results of the JRSC data; (c) shows the results of JSFB data. SNRs for SFSO, JRSC and JSFB are 5.56, 4.14, 4.00 respectively. [**ER**] siyuan1/. fl-xcorr-170510,JRSC-fl-xcorr-170510,JSFB-fl-xcorr-170510

AMPLITUDE RATIOS OF THE FELT LAKE EARTHQUAKES

Whether or not and how well our SFSO array can estimate magnitudes of the nearby weak events are also our first-order concerns. Herein, we compute the amplitude ratios of the Felt Lake template to the other four Felt Lake events, *Fl#2* through *Fl#5*, for SFSO recordings and the two broadband stations. If the ratios between the different instruments agree with each other, it means that our SFSO array can provide us with a reasonable magnitude estimation. For SFSO, we chose the part with the orange rectangles (shown in Figure 4) since it contains clear waveforms. For data within the orange rectangles, we set a 80% threshold. We set an absolute value threshold on the weaker events because we do not want to divide the template data by a value close to zero. We set the threshold on the template events because division between relatively great values tends to give us a more reliable ratio estimation. Thus only data points with absolute value greater than the threshold are used to compute the point-wise amplitude ratios. The median of the point-wise ratios are used as the amplitude ratio computed by SFSO data. For broadband stations, we first subtract the median value from the 1D time series; then following similar process as SFSO, for data within orange rectangles, we set a 80% threshold for each of the five events, only data points with absolute value greater than the threshold are used to compute the point-wise amplitude ratios. The median of the point-wise ratios are used as the amplitude ratio computed by broadband data. Because the SFSO recordings used to calculate amplitude ratios correspond to mostly S-wave and surface wave, for a fair comparison, we do not include the P-wave arrivals of the JRSC recordings in the rectangles. However, for JSFB, the P-wave and S-wave arrivals are hard to be differentiated because the epicenters of the Felt Lake events are fairly close to JSFB. Thus, we include the first arrivals of the JSFB recordings in the rectangles. For JRSC data, we did highpass preprocessing by 1.5 Hz to detrend the data.

The following table summarizes the results we obtained. Each column of the table refers to the amplitude ratios of template event (*Fl#1*) to the other four events respectively.

	Fl#2	*Fl#3*	*Fl#4*	*Fl#5*
SFSO	1.66	2.02	2.87	3.98
JRSC	1.92	2.03	3.97	4.72
JSFB	1.67	2.14	3.85	6.09

As the table above shows, all three instruments lead to a same conclusion that, in terms of amplitude, the order of the five events are: *Fl#1* > *Fl#2* > *Fl#3* > *Fl#4* > *Fl#5*. Note that the magnitude of *Fl#1* is estimated in the USGS database to be 1.3. This implies that our SFSO array could contain the information of relative magnitudes between small events. Additionally, we see that our array has the lowest ratios for all of the four events. For *Fl#2* and *Fl#3*, the ratios among the three stations are fairly close. For the two weaker events, *Fl#3* and *Fl#4*, SFSO has more underestimated ratios. This could be explained that our SFSO array has non-linear responses of events with different magnitudes. For events causing weak ground motion, our SFSO array may have a good coupling with the ground. For events with large magnitudes, our SFSO array may have loose coupling with the ground. Therefore, from *Fl#2* to *Fl#5*, the ratios obtained from SFSO recordings decreases. Further investigation is needed to be confirmed.

CONCLUSION AND FUTURE WORK

In conclusion, we show that cross-correlation can be applied on SFSO data as a template matching method to detect earthquakes. Comparing to the cross-correlation results obtained from the nearby broadband stations, we show that SFSO results have higher or similar SNR level with no evident false positives due to the advantage of the dense channel distribution, which means our SFSO array has a good sensitivity of the nearby weak events. Additionally, with the cross-correlation results for *Lad#4*, we show that the SFSO array is more sensitive to events with different mechanisms. From analyzing the amplitude ratios between the *Fl#1* and the other four detected Felt Lake earthquakes, we show that all three instruments get the same amplitude order. The amplitude ratios agree with each other well for the two relative large events, *Fl#2* and *Fl#3*. For *Fl#4* and *Fl#5*, our SFSO array tends to underestimate the ratios, which could be explained that our SFSO array has non-linear responses of events with different magnitudes.

The future work would be to keep exploring a way of increasing the SNR of the SFSO array taking advantage of the dense channel distribution of the SFSO array. One way could be performing cross-correlation or other template matching algorithms on the six segments independently, because each segment could have different data quality due to their different locations. Performing analyses independently could prevent us from being biased by some segments with particularly poor SNR. Then, we would perform cross-correlation with various template recordings (with various magnitudes) to understand the detection capacity of our SFSO array and how well we can estimate event magnitudes from computing amplitude ratios of SFSO data.

ACKNOWLEDGEMENT

We would like to thank OptaSense Ltd for making the SFSO recording equipment available. We thank Martin Karrenbach, Steve Cole and our colleagues at Stanford for helpful discussions and insights.

REFERENCES

Biondi, B., E. Martin, S. Cole, and M. Karrenbach, 2017, Earthquakes analysis using data recorded by the Stanford DAS Array: SEP-Report, **168**, 11–26.

Li, Z. and Z. Zhan, 2018, Pushing the limit of earthquake detection with distributed acoustic sensing and template matching: a case study at the brady geothermal field: Geophysical Journal International, **215**, 1583–1593.

Martin, E., B. Biondi, S. Cole, and M. Karrenbach, 2017a, Overview of the Stanford DAS Array-1 (SDASA-1): SEP-Report, **168**, 1–10.

Martin, E. R., C. M. Castillo, S. Cole, P. Sawasdee, S. Yuan, R. Clapp, M. Karrenbach, and B. L. Biondi, 2017b, Seismic monitoring leveraging existing telecom infrastructure at the sdasa: Active, passive, and ambient-noise analysis: The Leading Edge, **36**, 1025–1031.

Yuan, S., E. R. Martin, J. P. Chang, S. Cole, and B. Biondi, 2017, Catalog of Northern California earthquakes recorded by DAS : SEP-Report, **170**, 81–96.

Prediction error interpolation in bathymetry

Stewart A. Levin & Christopher M. Castillo

ABSTRACT

We have implemented missing data interpolation for bathymetry under an ArcGIS™ Python framework. We report on both stationary and nonstationary prediction error filter applications to synthetic and field data and the method compares favorably to several widely available interpolation methods.

INTRODUCTION

Claerbout and Fomel (2014, chap. 7.6–9) show how to apply prediction error filter (PEF) theory to interpolation of missing data in seismic and topographic datasets. After optional preprocessing and regridding, a PEF is designed on the existing data and the missing data are determined by calculating what values produce the least prediction error when the PEF is applied to the whole grid.

To assist in coauthor Castillo's thesis work, we reimplemented the PEF interpolation under the ArcGIS framework using the ArcPy tool API. Subsequently, Claerbout and Wang (2018) introduced a spatially-variable version of PEF interpolation which we also implemented in ArcGIS.

TOOL DESIGN

ArcGIS Python provides has a well-defined template for building a Python toolbox that leverages an extensive library of GIS functionality (Zandbergen, 2013) interfaced to the NumPy package (SciPy.org, 2018). A toolbox contains one or more individual tools which retrieve and validate parameters, execute code, and inform the user of progress status. In addition, tools may be combined into scripts and macros.

Our PEF-Tools Arc Toolbox provides both a missing data interpolation "PEF-S" using a stationary PEF as described in Claerbout and Fomel (2014) and one, "PEF-N", using the nonstationary PEF from Claerbout and Wang (2018). These are supplemented and/or wrapped with:

Auto-PEF Automatically identifies holes in the input data, creates areas of interest (AOI) around them based on a user-specified edge width, and runs PEF-S separately for each. Useful for large datasets with many small holes.

Define AOI Creates polygons around data gaps in the input data to be used as the area of interest for PEF-S infill.

Detrend Subtracts box-car filtered data from the input to emphasize the local texture around data gaps.

Benchmark Compares available interpolation methods in ArcGIS Spatial Analyst for in input tiff image.

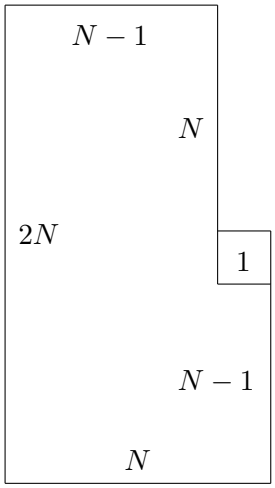
No-Edge Slope Outputs a slope map and digital elevation map (DEM) with no visible edges.

No-Edge Shade Outputs a hillshade and DEM with no visible edges.

SESRM Outputs a slope-enhanced shaded hillside and DEM with no visible edges.

IMPLEMENTATION CHOICES AND ISSUES

For coding and computational efficiency, we limited our PEFs to have the shape



where the sample count N is limited to be at most 5 (45 coefficients) and the coefficient at the prediction location is unity. N is determined by counting the number of places it can overlay known data in the current AOI and ensuring that number is at least 9 times as large as the number of coeffients. We note that the orientation of the PEF, e.g. 90° rotation or 180° reflection, factors into this calculation.[1]

Having chosen the PEF shape, the next step for both the stationary and nonstationary cases is to compute the PEF that minimized the L_2 prediction error for the known data. This serves as the known PEF for the stationary case and as the initial PEF estimate for the nonstationary case.

PEF-S

PEF-S designs its single PEF on the live data in the AOI using the LSMR iterative solver (Fong and Saunders, 2011) in SciPy. The missing data are obtained using the SciPy *spsolve*

[1]Claerbout and Fomel (2014,chap. 7.10.3) suggests that orientation of the PEF should not make a difference in the interpolation output. We must note that, unlike them, we are not solving for the filter simultaneously with the missing data.

direct sparse solver. We have found that the filter orientation can affect the result when the input is nonstationary or otherwise too complex to be fully represented with our largest choice of filter. Detrending the surrounding data has helped to minimize this effect when data gaps were relatively small.

PEF-N

For this algorithm, there are two adaptation parameters as well as 8 choices of filter orientation and the ability to make multiple passes over the grid to improve the missing data estimates. We chose to use all 8 orientations sequentially in each outer iteration. One adaptation parameter, ϵ_A, controls how quickly the PEF, A, is allowed to change from sample to sample as it shifts along the grid; the other, ϵ_Y, controls how much to update the missing data, Y, estimate with the changing PEF. Of the two, ϵ_Y proved to be the most critical. Indeed, as Fig. 1 illustrates, the interpolated values can diverge for what seems a reasonable choice of unitless ϵ_Y.

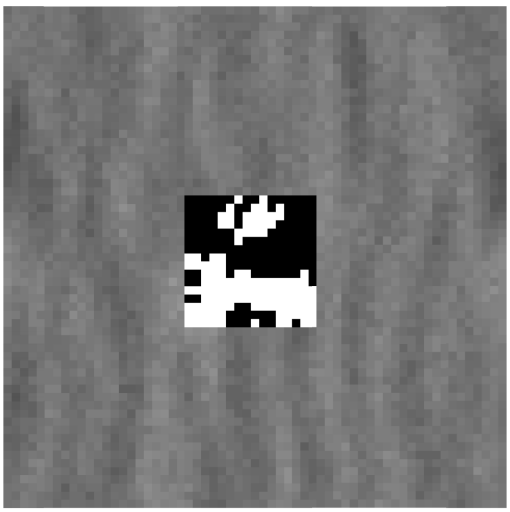


Figure 1: Infill of hole surrounded by random Gaussian numbers with $\epsilon_Y = \frac{1}{2}$ [**CR**] stew1/. PEFinterpBigepsd

After experimenting with a range of smaller values for ϵ_Y, we realized that the issue was tied to the PEF and not ϵ_Y. While a PEF reduces the norm of its output with respect to the input on which it was designed, the fact that the PEF begins with a leading coefficient of 1 says that, as an operator, it always has a norm greater than 1 unless it is the trivial identity filter. There is no a priori reason that its action on trial values for missing data should not reflect that potential magnification. So, to account for this, we rescaled our ϵ_Y by dividing it by the norm of the initial PEF filter we designed on the live data in the AOI. This, too, managed to blow up! We finally realized that because the filter, A, was adapting as it entered the gaps, and adapting differently from iteration to iteration, we had to recalculate the filter norm at every step in order to ensure that our missing data updates were under control and stayed usefully incremental.

The situation for ϵ_A was less challenging because the guideline in Claerbout and Wang (2018,chap. 1.3) to use the reciprocal of the variance of the data, Y, to rescale it to a unitless value worked reasonably well. The value we settled on, $\epsilon_A = 0.05/\sum d_i^2$, where d_i are the samples under the current filter, was a compromise between adapting painfully slowly and allowing the last orientation in an iteration to introduce a visible directional bias in the infill.

In addition to setting ϵ_Y and ϵ_A, we also could vary the number of interpolation passes over the grid. We settled on 64 eight-way passes, admittedly overkill, but sufficient to converge the infill of all but the largest gaps.

EXAMPLES

Prediction error filters are designed to track individual sinusoids and combinations of them, limited by just the number of PEF coefficients. Figures 2 and 3 are evidence that our code is working. In both of these examples we specifically avoided cases where either the hole or the sinusoids serendipitously aligned with coordinate axes. In addition, we made sure that the criss-crossing sinusoids had distinctly different wavenumbers and slopes.

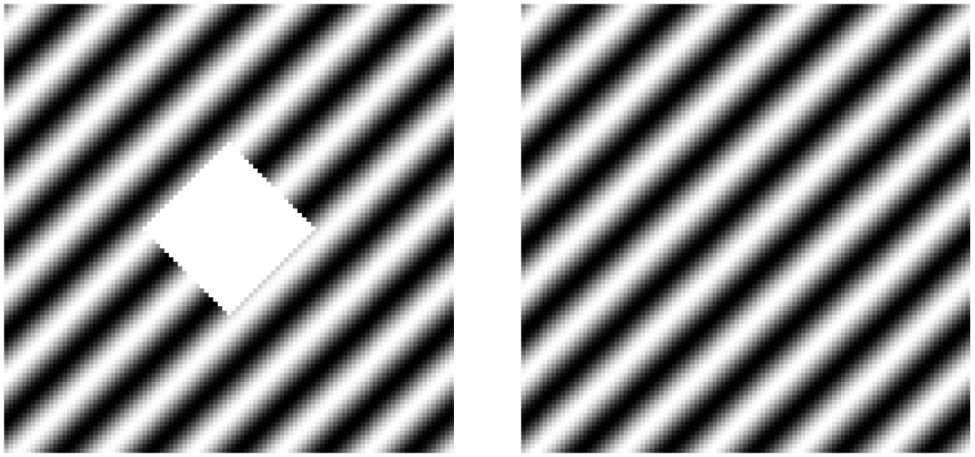


Figure 2: Test of PEF-S to ensure it reproduces input spatial sinusoid in a hole that is not aligned with x and y axes. [**CR**] stew1/. DiamondPair

While Fourier theory says that even the most complex seafloor grid of bathymetry can be perfectly reconstructed with a finite number of sinusoids, there is no reason to expect that a global Fourier reconstruction will infill data gaps in a useful way when, as is almost always the case, the statistics and pattern of the bathymetry change locally. This is illustrated in the challenging dataset of Figure 4. Our interpolation is highly realistic, exhibiting the characteristics of the data that our eye expects to see in the hole.

Finally, Figure 5 shows a large scale application of PEF interpolation to infill typical gaps in seafloor coverage that arise as the echo sounding vessel shifts and meanders as it moves along its nominal path.

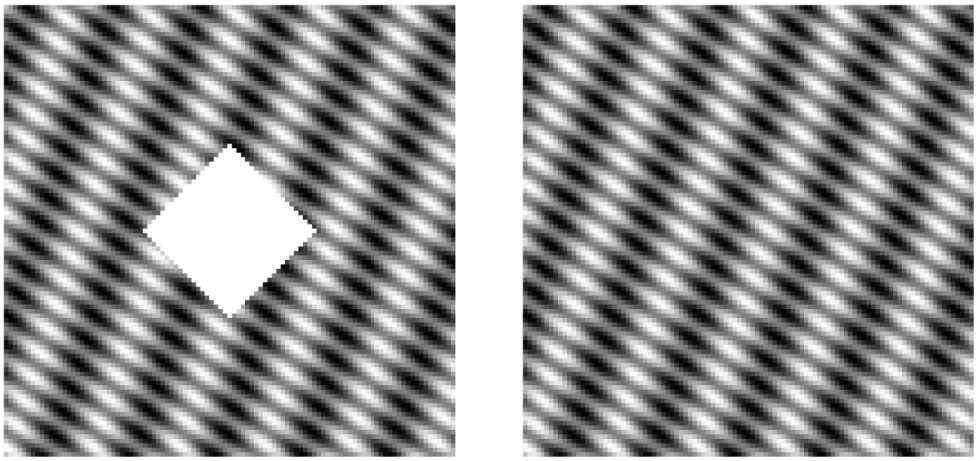


Figure 3: Test of PEF-S to ensure it reproduces two input spatial sinusoids with incommensurate orientations and wavenumbers. [**CR**] stew1/. CrissCross

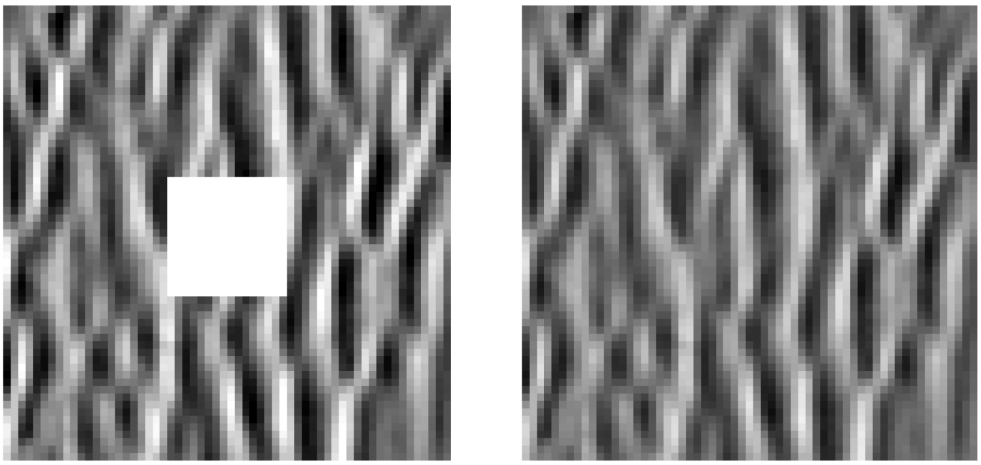


Figure 4: Portion of seafloor texture after detrending and removing a 16x16 square in the center alongside a PEF-N interpolation. [**CR**] stew1/. SeafloorTexture

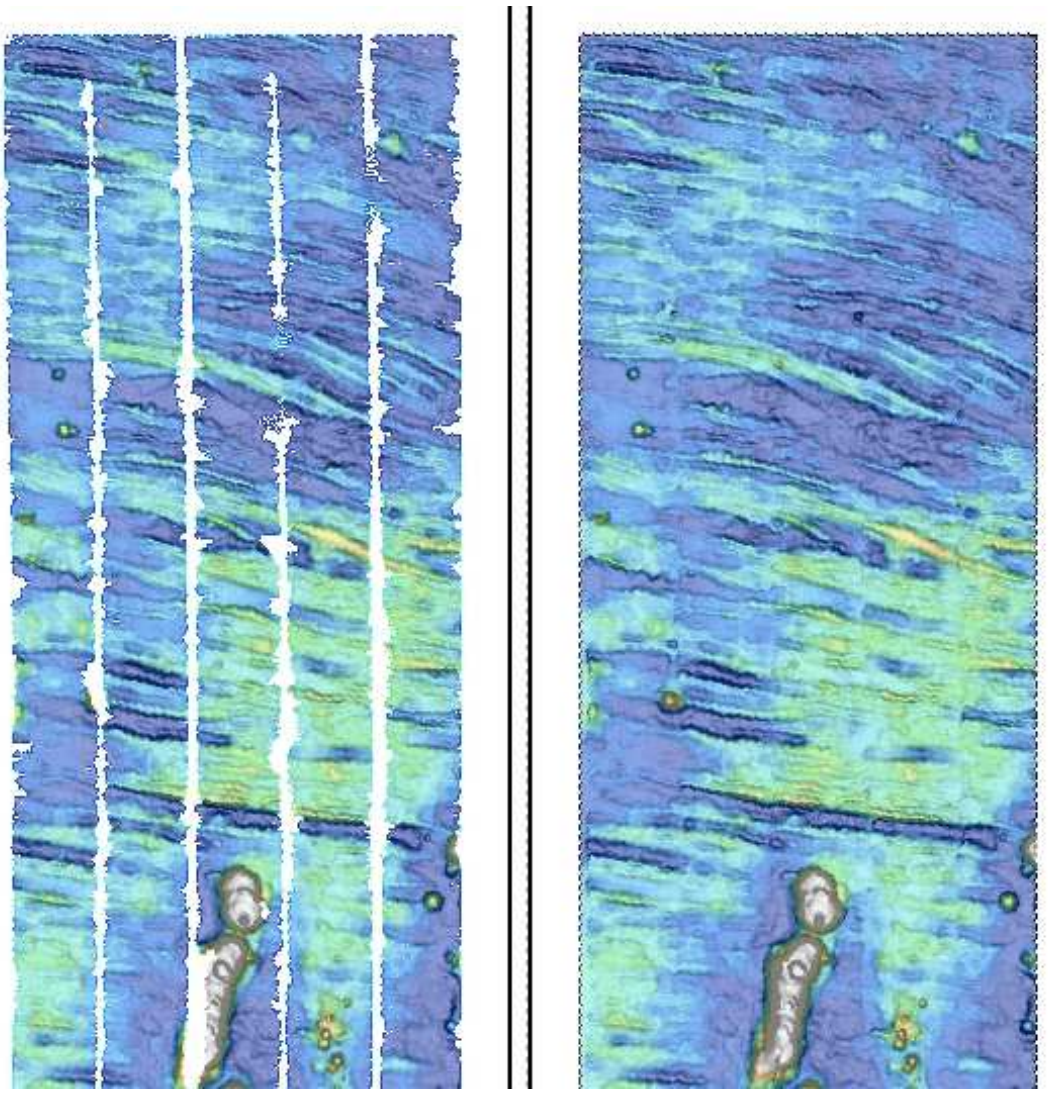


Figure 5: PEF-S bathymetry interpolation. [**CR**] stew1/. TrackInterp

SUMMARY

We have implemented both stationary and nonstationary PEF missing data interpolation within the ArcGIS system. We developed reasonable choices for the adjustable parameters in the nonstationary setting using a combination of theoretical reasoning and empirical trials. While further optimization, including GPU coding, can help reduce the time it takes to iterate to an answer, we would challenge anyone to find the location of the originally missing data in our results.

REFERENCES

Claerbout, J. and K. Wang, 2018, Data Fitting with Nonstationary Statistics: Lulu Press, www.lulu.com.

Claerbout, J. F. and S. Fomel, 2014, Geophysical Image Estimation by Example: Stanford Exploration Project.

Fong, D. C.-L. and M. A. Saunders, 2011, LSMR: An iterative algorithm for sparse least-squares problems: SIAM J. Sci. Comput., **33**, 2950–2971.

SciPy.org, 2018, NumPy v1.14 Manual.

Zandbergen, P. A., 2013, Python scripting for ArcGIS: Esri Press, Redlands, California.

Can we beat FFTs computational speed for one-way wavefield propagation?

Biondo Biondi

ABSTRACT

I compare the computational efficiency of one-way wavefield propagation by FFT-based methods and by implicit finite-differences methods. On modern computer architectures, the theoretical computational advantages of implicit finite-differences methods are not realized in practice. The speed tests that I present indicate that FFTs run much more efficiently than sparse linear solvers and thus they should be the foundation for efficient waveform inverse solutions based on one-way propagation.

INTRODUCTION

Imaging the full bandwidth of seismic data by using full waveform inversion (FWI) and linearized waveform inversion (LWI) is hampered by the computational cost of two-way propagation that scales with the fourth power of the frequency. The computational advantages of one-way propagation compared to two-way propagation suggest its use for imaging subsurface structures that are not sufficiently complex to require full two-way propagation. Furthermore, Shan and Biondi (2008) showed that by combining plane-wave decomposition and one-wave propagation we can image steeply dipping reflectors and overturned events.

For high-resolution imaging, the main advantage of one-way propagation is that its computational cost grows slower with frequencies than the cost of two-way propagation. This is particularly true when we are interested in imaging with anisotropic models because one-way operators can be generalized to approximate the kinematics of wave propagation in anisotropic media more cost effectively than two-way propagators. The computational-cost advantages becomes even more attractive when considering waveform inversion with extended model such as tomographic FWI (Biondi and Almomin, 2014; Barnier et al., 2018). Sarkar and Biondi (2018) showed the time-domain velocity model extension using correlation-time is equivalent to a simple extension in frequencies when the wave-equation is solved in the frequency domain. Therefore, no additional correlations are needed for extended-model frequency-domain methods, although frequency-domain extended models are still substantially larger than not extended model and their handling may still lead to additional I/O costs.

Because accurate wide-angle imaging requires the propagation of wavefield by an operator with large spatial width, the computational kernels of the most commonly used one-way propagation methods require the application of FFTs or the solution of large linear system of equations (implicit methods). Mixed-domain methods that require both the application of FFTs and solution of linear systems are also often implemented (Gazdag and Sguazzero, 1984; Stoffa et al., 1990; Ristow and Ruhl, 1994; Biondi, 2006). A theoretical analysis of the computational complexity suggests that implicit methods might be particularly efficient

because the linear system is solved by inverting a matrix that depends only on the velocity model. Consequently the cost of factorizing this matrix can be amortized an all the shots that share the same velocity model. Furthermore, for methods such as LWI and TFWI (Barnier et al., 2018) that require several linearized iterations with the same background model, the factorization does not change with iterations and thus can be reused across iterations, further decreasing the relative computational burden of matrix factorization.

However, comparative theoretical efficiency advantages may not translate into practical ones on modern computer architecture. In this report Akhmadiev et al. (2018) present the beginning of a project on waveform inversion by using one-way propagation. Therefore, I investigated the efficiency of implicit one-way propagation compared to FFT-based one-way propagation to determine which type of one-way extrapolator we may want to base that project. In my comparison I used the FFTW3 library to perform FFTs and the Pardiso solver part of the Intel Math Kernel Library (MKL) to factorize the matrix and recursively solve the system (Schenk and Gartner, 2004).

NUMERICAL TESTS

I run my speed tests on a 2.3 GHz dual-CPU Intel Xeon "Haswell" processor (E5-2670-v3) with 12 cores per CPU for a total of 24 cores and a maximum of 48 threads. I run my downward continuation code by parallelizing the computations in two ways: 1) across temporal frequencies by using OpenMP threading and 2) within a single solver call using the intrinsic parallelism controlled by the MKL library. The coarse frequency parallelism that is exploited by OpenMP threading is the most natural and efficient way of parallelizing both the FFTs and the linear solver solution. I report the FFT speed only using this parallelization mode, whereas I measured and report the linear solver speed using both parallelization modes.

Vectorization is also essential to achieve efficiency with modern CPUs. When implicit one-way propagation is used to solve a waveform inversion problem, both parallelization and vectorization are possible when solving the same linear system with many right hand sides (RHS). The multiple RHS correspond to multiple shot gathers that share the same computational domain that may encompass the spatial extent of the whole survey, or a shared subset of it. Therefore, I run tests using different number of right hand sides ($\mathrm{N_{RHS}}$) as well as three different lengths of the horizontal axis ($\mathrm{N_x}$): two short ones (Nx=512 for OpenMP parallelization and Nx=544 for MKL internal parallelization) and a long one ($\mathrm{N_x}$=65,536 only for OpenMP parallelization). I report only the solver speeds and not of the matrix factorization because the same factorization of the matrix is used for all RHS. The cost of the matrix factorization is amortized on all RHS, and thus the controlling speed is the one of the solver, not the factorization.

The computational throughputs shown in the figures below were measured using the system clock. They show how many millions of horizontal grid points can be propagated by one depth level per second. The measured throughputs were averaged over 220 depth steps and 240 temporal frequencies.

Parallelism across temporal frequencies by OpenMP threads

Figures 1–3 show the throughputs of the FFTs parallelized over frequencies as a function of the number of OpenMP threads (N_{OMP}) used. The asterisks correspond to actual measurements, whereas the solid lines interpolate between measurements. The dashed lines show the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. The results shown in these figures correspond to different values of N_x and N_{RHS} to test the scalability in different situations.

Figure 1 shows the results for N_x=512 and N_{RHS}=48. Computational throughput scales almost linearly up to 12 OpenMP threads, and then starts flattening out most likely because memory bandwidth is increasingly saturated. The maximum throughput is achieved for 48 threads. When this measurement is converted to flop/s using the "classical" formula for estimating the number of flops needed to compute FFTs ($N_{flop} = 5N_x \log N_x$), it results in an impressive 478 Gflop/s. This speed is about 48% of the maximum theoretical speed of the processor when the clock frequency is "turbo" boosted to 2.6 GHz and 16 flop are performed for each cycle of all the 24 cores available. When the number of right hand-sides is scaled by a factor 20 to N_{RHS}=960, the behavior is similar but the maximum speed of 416 Gflop/s is slightly lower because of memory-bandwidths limitations, as shown in Figure 2.

Figure 3 shows that the throughput decreases when N_x is scaled by a factor of 128; that is $N_x = 65,536$ and N_{RHS}=2. This decrease is caused by the more-than-linear increase in computational complexity of FFTs. The actual computational speed with N_{OMP}=48 is 425 Gflop/s; that is still a high percentage of the maximum theoretical speed of the hardware. It is unlikely that in a real 2D problem the horizontal axis is as long as 65,536 samples. However, my code is currently limited to 2D wave propagation, and this choice of parameters was motivated to simulate the computational complexity of 2D FFTs when performing 3-D wave propagation on a fairly small problem (65,536=256*256).

Figures 4–6 show the throughputs of the implicit propagator parallelized over frequencies as a function of the number of OpenMP threads (N_{OMP}) used. As for the previous figures, the asterisks correspond to actual measurements, whereas the solid lines interpolate between measurements. The dashed lines show the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads.

The general scaling trend in Figure 4 is similar to the trend in Figure 1, however the throughput axis is now scaled down by a factor of 100. The maximum throughput achieved by FFTs with 48 OpenMP threads is higher by a factor 75 than the one achieved by the linear solver using the same number of threads. When the number of RHS is scaled by 20 (Figure 5) the comparison with FFTs throughput is slightly less unfavorable because the computations are better vectorized over the RHS axis. The linear solver throughput scales better than before up to a 24 OpenMP threads and the ratio between the maximum throughputs achieved with 48 OpenMp threads is "only" 65.

Figure 6 shows that the throughput substantially decreases when N_{RHS} decreases to 2 and N_x is scaled up by a factor of 128. The ratio between the maximum throughputs achieved by FFTs and linear solver with 24 OpenMp threads is 78. This decrease in relative efficiency of the linear solver is caused by the lack of vectorization over the RHS axis; it more than compensates for the relative advantage that the solution phase of the linear scales linearly with N_x, in contrast with the superlinear increase of computational complexity of

FFTs.

Parallelism within solver call by MKL threads

The sparse-matrix solver in the Pardiso library has been developed to take advantage of multi-threaded computers for parallelizing computations within each library call. This capability is accessed by setting the number of MKL threads (N_{MKL}) using an environment variable. Therefore, I tested this alternative way of parallelizing the solution of the linear system to verify whether it leads to higher performances than the coarse parallelization over temporal frequencies achieved by calling the library solver from different OpenMP threads that I discussed in previous subsection. Figures 7 and 8 show the throughputs measured with different MKL threads and with different number of RHS. Notice that I show the results with N_x=544 because performances were slightly higher than with my first choice (N_x=512). Performances improve when the dimension of the system is not a power-of-two because the reuse of in-cache data increases (Stew Levin, personal communication). The maximum number of threads that the Pardiso library can utilize is equal to the number of cores available in hardware, therefore I measured performances using up to 24 threads.

Figure 7 shows that show the throughputs of the implicit propagator as a function of N_{MKL} used, with N_{RHS}=1920; that is the maximum I could fit into memory. As for the previous figures, the asterisks correspond to actual measurements, whereas the solid line interpolates between measurements. The dashed line shows the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. The performance scales almost linearly with the number of threads up to $N_{MKL} = 12$ but it decreases with $N_{MKL} = 24$. This is probably caused by saturation of the memory bandwidth.

Figure 8 demonstrates that the solver efficiency strongly depends on vectorization as well as parallelization by showing that throughput substantially improves as N_{RHS} increases up to 960, although it starts to flatten out after 960. It shows these gains for both $N_{MKL} = 12$ and $N_{MKL} = 24$. However, by comparing Figure 8 with Figure 5 we can observe that the maximum throughput achieved by the intrinsic parallelism within each solver call is lower by a factor of two with respect to the maximum throughput achieved by parallelizing over temporal frequencies. Consequently we can conclude that parallelizing over frequencies is a better choice than using the intrinsic parallelization in the Pardiso library, even when the number of RHS is sufficiently large to enable full vectorization. Some mixed-domain methods require both the performance of FFTs and the solutions of linear systems of equations (Ristow and Ruhl, 1994; Biondi, 2006); therefore, parallelization over frequencies seems to be the optimal choice when these methods are implemented.

DISCUSSION AND CONCLUSIONS

The throughput results presented in the previous section clearly show that the computational efficiency of FFTs on modern architectures tilts the plane in favor of one-way propagation algorithms that use FFTs to perform the computational heavy lifting. Even when solving relatively large problems for which FFTs computational cost scales faster than linearly, the measured throughput is still faster by a factor between 50 and 100. In practice,

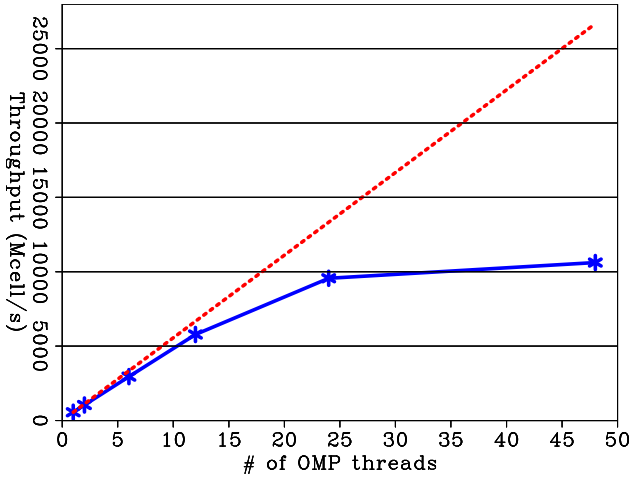


Figure 1: Throughputs of FFTs parallelized over frequencies as a function of the number of OpenMP threads (N_{OMP}) with N_x=512 and N_{RHS}=48. The dashed line shows the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. [**NR**] biondo1/. FFT-nx512-nh48-omp

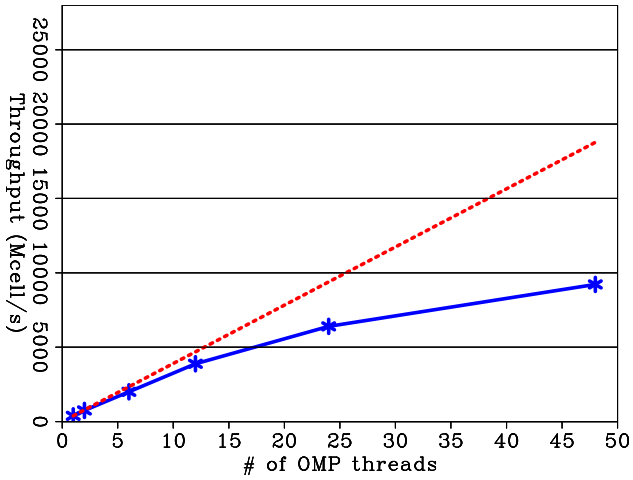


Figure 2: Throughputs of FFTs parallelized over frequencies as a function of the number of OpenMP threads (N_{OMP}) with N_x=512 and N_{RHS}=960. The dashed line shows the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. [**NR**] biondo1/. FFT-nx512-nh960-omp

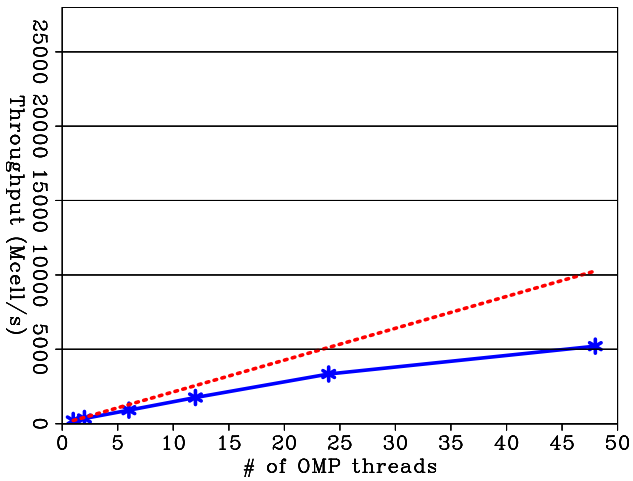


Figure 3: Throughputs of FFTs parallelized over frequencies as a function of the number of OpenMP threads (N_{OMP}) with N_x=65,536 and N_{RHS}=2. The dashed line shows the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. [**NR**] biondo1/. FFT-nx65536-nh2-omp

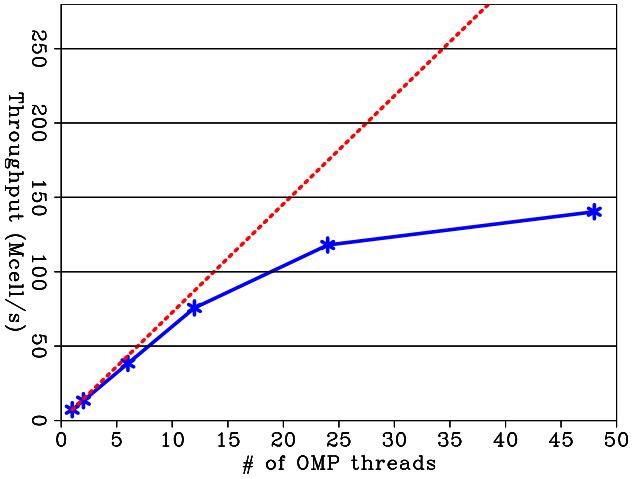


Figure 4: Throughputs of Pardiso solver parallelized over frequencies as a function of the number of OpenMP threads (N_{OMP}) with N_x=512 and N_{RHS}=48. The dashed line shows the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. [**NR**] biondo1/. Pard-nx512-nh48-omp

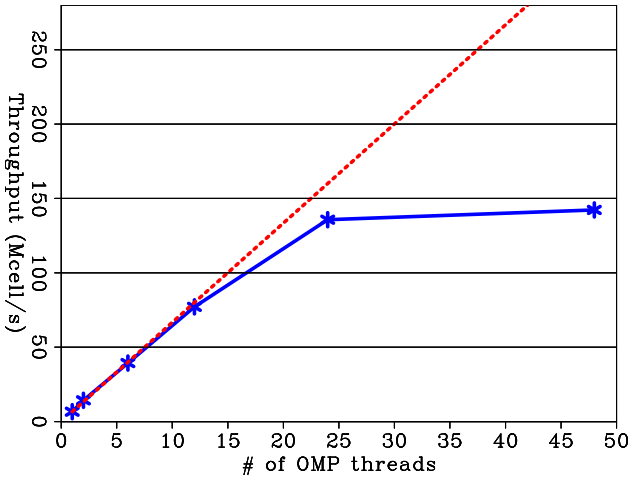


Figure 5: Throughputs of Pardiso solver parallelized over frequencies as a function of the number of OpenMP threads (N_{OMP}) with N_x=512 and N_{RHS}=960. The dashed line shows the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. [**NR**] biondo1/. Pard-nx512-nh960-omp

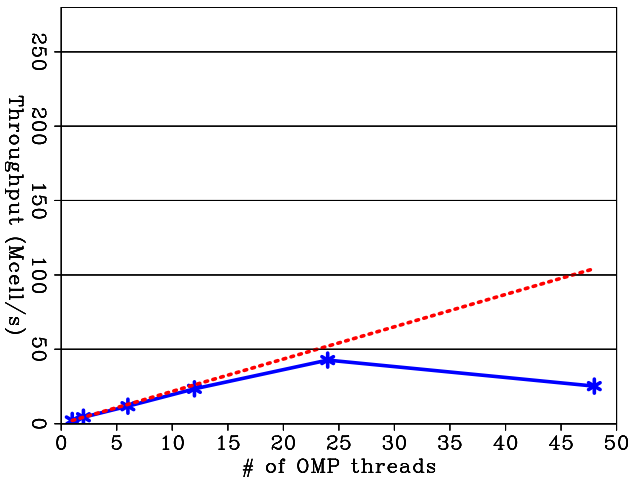


Figure 6: Throughputs of Pardiso solver parallelized over frequencies as a function of the number of OpenMP threads (N_{OMP}) with N_x=65,536 and N_{RHS}=2. The dashed line shows the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. [**NR**] biondo1/. Pard-nx65536-nh2-omp

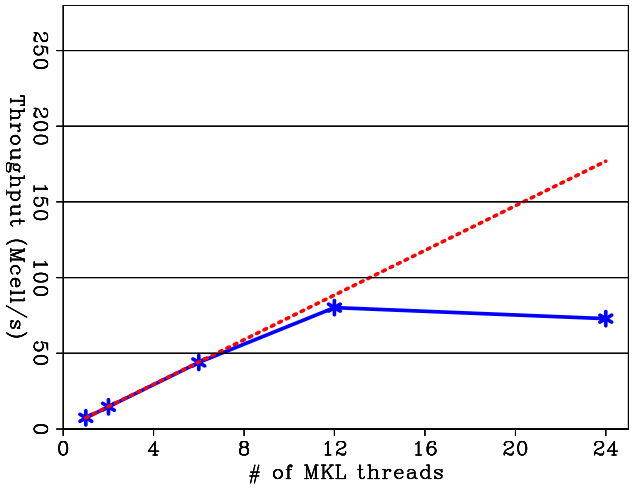


Figure 7: Throughputs of Pardiso solver parallelized within the library call using the intrinsic parallelization as a function of the number of MKL threads (N_{MKL}) with N_x=544 and N_{RHS}=920. The dashed line shows the theoretical throughput that would be achieved if the throughput measured for one thread scaled-up linearly with the number of threads. [**NR**] biondo1/. Pard-nx544-nh1920-mkl

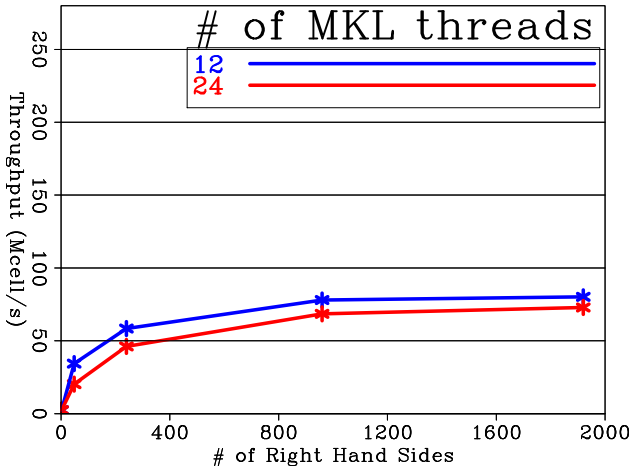


Figure 8: Throughputs of Pardiso solver parallelized within the library call using the intrinsic parallelization as a function of the number RHS with N_x=544 and for N_{MKL}=12 (blue line) and N_{MKL}=24 (red line). [**NR**] biondo1/. Pard-nx544-mkl12-24-rhs

we can use many reference velocities in FFT-based methods to approximate propagation with strong lateral variations and still achieve higher throughput than by using implicit methods.

The computational efficiency of the Pardiso solvers that I describe in this paper might be improved. First I may not have correctly used all the parallelizing options provided by the MKL library. However, an analysis of the graphs in Figure 7 and Figure 8 clearly shows that I succeeded to achieve a high degree of parallelization inside each solver call. Second, the MKL library provides the option of supplying user-written customized solver based on the matrix factorization performed by the library. Probably some efficiency gain could be achieved by using a customized solver that takes advantage of problem-specific knowledge of the structure of the matrix factors to improve the use of available memory bandwidth. However, it is doubtful that those gains would be large enough to change the balance in favor of implicit methods, and justify the additional code development efforts required for developing customized solvers.

When anisotropic propagation is implemented with FFTs, it is important to use smart reference velocity-parameters selection methods, such as Lloyd algorithm (Clapp, 2004; Tang and Clapp, 2006). Efficient algorithms may also include correction terms analogous to the ones presented by Ristow and Ruhl (1994), but applied as convolutional filters instead of as recursive linear-systems solutions. These convolutional correction terms would perform more efficiently than implicit methods on modern architectures; the challenge would be to assure unconditional stability of wavefield propagation using convolutional correction terms.

ACKNOWLEDGMENTS

I would like to thank Stew Levin and Bob Clapp for sharing their in-depth knowledge of modern architectures to improve the significance of the tests I present in this paper.

REFERENCES

Akhmadiev, R., B. Biondi, and R. G. Clapp, 2018, Full-waveform inversion problem using one-way wave extrapolation operators: SEP-Report, **174**, 31–40.

Barnier, G., E. Biondi, and B. Biondi, 2018, Full waveform inversion by model extension: SEP-Report, **172**, 153–171.

Biondi, B. and A. Almomin, 2014, Simultaneous inversion of full data bandwidth by tomographic full waveform inversion: Geophysics, **79**, WA129–WA140.

Biondi, B. L., 2006, 3D Seismic Imaging: Society of Exploration Geophysicists.

Clapp, R. G., 2004, Reference velocity selection by a generalized Lloyd method: 74th Ann. Internat. Mtg., Expanded Abstracts, 981–984, Soc. of Expl. Geophys.

Gazdag, J. and P. Sguazzero, 1984, Migration of seismic data by phase-shift plus interpolation: Geophysics, **49**, 124–131.

Ristow, D. and T. Ruhl, 1994, Fourier finite-difference migration: Geophysics, **59**, 1882–1893.

Sarkar, R. and B. Biondi, 2018, Frequency domain tomographic full waveform inversion: SEP-Report, **172**, 173–191.

Schenk, O. and K. Gartner, 2004, Solving unsymmetric sparse systems of linear equations with pardiso: Future Generation Computer Systems, **20**, 475–487.

Shan, G. and B. Biondi, 2008, Plane-wave migration in tilted coordinates: Geophysics, **73**, S185–S194.

Stoffa, P. L., J. Fokkema, R. M. de Luna Freire, and W. P. Kessinger, 1990, Split-step Fourier migration: Geophysics, **55**, 410–421.

Tang, Y. and R. G. Clapp, 2006, Selection of reference-anisotropy parameters for wavefield extrapolation by Lloyd's algorithm: 76th Ann. Internat. Mtg., Expanded Abstracts, ANI 2.7, Soc. of Expl. Geophys.

Buffers: A library for fast parallel IO and compression

Robert G. Clapp

ABSTRACT

Data transfer speed improvements from different memory levels has continually lagged behind floating point operations per second (FLOPS) improvements. For seismic applications these differing growth curves have made more and more applications IO bound. We created a library that breaks a dataset into blocks. These blocks can be read/written to disk in parallel, significantly reducing IO time when using parallel file system or object store. Each block can further be compressed, using multi-dimension compression schemes, reducing the amount of data that needs to be transferred between memory levels.

INTRODUCTION

Until the early 2000s algorithms were judged primarily by the operation count. Today, due to the drastically different growth rates of Central Processing Unit (CPU) speed and main memory (McCalpin, 1995, 2007) almost all applications are judged based on either their memory bandwidth or memory latency behavior. These bottlenecks can come from many different memory levels: from disk, from main memory on the CPU, or even global memory on a General Purpose Graphical Processing Unit (GPGPU). To address the latency issue, it is important to read in large chunks. To address the bandwidth issue, doing parallel reads can often prove performance particularly on parallel file and object store (Factor et al., 2005) system. Further performance improvements can be achieved through compression. Seismic data have been shown to be compressible when using multi-dimensional schemes(Villasenor et al., 1996).

In this paper we describe a library, called `buffers`, that breaks a multi-dimensional hypercube into blocks. These blocks can be compressed using public domain compression packages designed for scientific computing such as ZFP(Lindstrom and Isenburg, 2006). The library implements its own crude caching system that can be used to minimize memory usage. The library `genericIO`(Clapp, 2017) has been extended to use the `buffers` library. As a result, any program using this library for IO can automatically taking advantage of the speedup it offers.

We will begin by describing the interfaces to the library. We will then present some preliminary results showing both the speedup in IO and the error using various compression schemes.

LIBRARY

The `buffers` library functions are controlled by three objects (classes): how/whether to compress the data, the scheme used to block the hypercube, and how to handle memory

usage. By default it does not compress the data, it holds everything in memory, and it is a relatively smart blocking scheme for conventional CPU architectures.

Compression

Currently two different compression schemes are offered. The first is `noCompression`. The second option uses the ZFP library to compress the data. ZFP is basically a very fast 1, 2, or 3-D JPEG compresser. For seismic data it can achieve compression rates of 6-10x with acceptable loss for most applications. Its big advantage is its speed. It can compress and decompress at nearly memory bandwidth. It is ideally suited for applications where data access speed is essential, such as visualization or compressing wavefields. Currently the `buffers` library supports the tolerance, precision, and rate methods ZFP uses.

Blocking

Blocking describes how to break up an N-D cube. By default it tries to makes blocks in the low Megabytes in size, maxing out in 3-D blocks (4 and 5-D blocks are better for the SZ library). The user can create his own block object specifying a target blocksize along each axis and desired chunk, the minimum unit to use along each axis (think cache line).

Memory usage

As mentioned above the library defaults to keeping every block in memory(`memoryAll`), often in an uncompressed state. For many applications this is an undesired behavior. As a result a second memory option, `simpleMemoryLimit`, exists. This class allows the user to set a target memory for buffers library. Every time a `buffers` function is called the `simpleMemoryLimit` module will attempt to first compress then store to disk blocks that have been unused for the longest period. Further, more sophisticated caching schemes might be added later.

Buffer functions

A `buffers` object can be created by either initializing with a hypercube (description of the N-D cube) and potentially the `compression`, `blocking`, and `memory` objects or by a JSON file that describes these choices. Most of the access to the actual data is through the `getWindow` and `putWindow` functions. As their names imply, they are used to get and put data to/from the `buffers` class. The user also has the ability to `changeState` which will change between the three possible states of a buffer: on disk, compressed, and decompressed. The final function `setDirectory` can be used to specify where to store the dataset.

GenericIO

A new IO object type, `BUFFERS` has been created within the `GenericIO` library. The `BUFFERS` IO class uses JSON for parameter handling and file descriptions. Rather than specifying a file, the user specifies the directory to read/write a `buffers` object.

RESULTS

To test the speed of library, I compared using standard SEPlib IO routine calls such as `sreed`, `sreed_window` with all code compiled with the same optimization level (3). The test used a 2.5 GB file of migrated seismic data from the Gulf of Mexico. The file compressed to 670 MB using the ZFP technique using constant rate compression method. I tested the read speed on a Mac, a local disk on a Linux server, and off of a parallel file system. On all systems the blocked ZFP dataset proved to be faster with speedups varying from 3.2 (Mac) to 12 (on an active parallel file system). Speedup on the cloud should be even more significant.

As a second test, I used a smaller dataset with lower wave number features. When compressed the dataset shrunk in size from 216 MB to 14 MB. Figure 1 shows an example of the errors introduced by using the ZFP compression method. The top left shows a slice from a migrated cube from the North Sea, the top right shows the result of reconstructing the cube. The bottom left shows the difference. The bottom right show the difference scaled by 100. Note how the error is small and generally random.

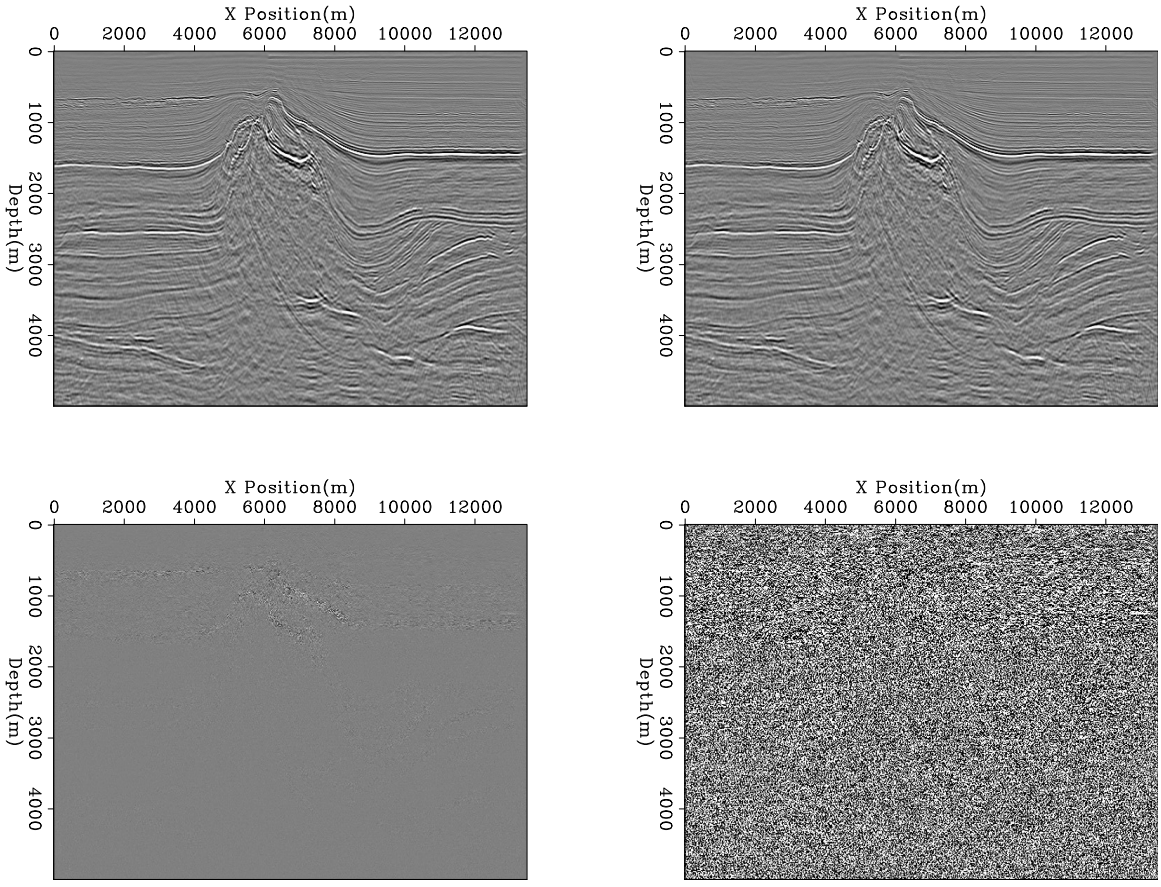


Figure 1: The top left shows a slice from a migrated cube from the North Sea, the top right shows the result of reconstructing the cube. The bottom left shows the difference. The bottom right show the difference scaled by 100. [**ER**] bob1/. tot

DISCUSSION AND CONCLUSIONS

In this paper, we described a new library `buffers`. The library is designed to help people reduce memory bandwidth bottlenecks. It allows the user to seamlessly read and write in parallel to improve disk IO. It also can be used in applications such as visualization and storing/compressing wavefields to greatly reduce the memory and disk bandwidth requirements.

REFERENCES

Clapp, R. G., 2017, Facilitating code distribution: Docker and Generic IO.

Factor, M., K. Meth, D. Naor, O. Rodeh, and J. Satran, 2005, Object storage: The future building block for storage systems: Local to Global Data Interoperability-Challenges and Technologies, 2005, IEEE, 119–123.

Lindstrom, P., and M. Isenburg, 2006, Fast and efficient compression of floating-point data: **12**, 1245–50.

McCalpin, J. D., 1991-2007, Stream: Sustainable memory bandwidth in high performance computers: Technical report, University of Virginia, Charlottesville, Virginia. (A continually updated technical report. http://www.cs.virginia.edu/stream/).

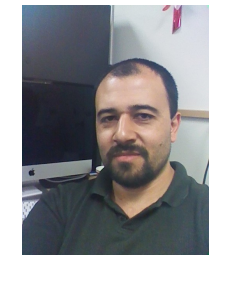
———, 1995, Memory bandwidth and machine balance in current high performance computers: IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter, 19–25.

Villasenor, J. P., R. A. Ergas, and P. L. Donoho, 1996, Seismic data compression using high-dimensional wavelet transforms: Snowbird, UT, USA, Proceedings of the Data Compression Conference, IEEE Computer Society Press, 396–405.

Research Personnel

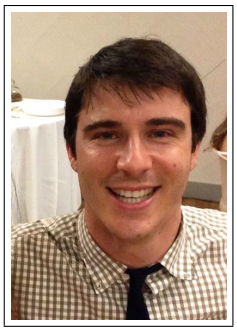
Rustam Akhmadiev received his Bachelor's and Master's degree in Geophysics from Lomonosov Moscow State University, Russia in 2015. After that Rustam has got another Master's degree in Petroleum geophysics from IFP, France. Upon finishing his study in IFP in 2016, he was interning in Total's scientific research center in Pau, France working on higher-order finite-difference schemes used in seismic inversion. In July 2017 he started his Ph.D. program at Stanford Exploration Project. He is interested in elastic and anisotropic wave phenomena with applications in velocity models reconstruction and inversion.

Biondo L. Biondi is professor of Geophysics at Stanford University. Biondo graduated from Politecnico di Milano in 1984 and received an M.S. (1988) and a Ph.D. (1990) in geophysics from Stanford. He is co-director of the Stanford Exploration Project and of the Stanford Center for Computational Earth and Environmental Science. In 2004 the Society of Exploration Geophysicists (SEG) has honored Biondo with the Reginald Fessenden Award. Biondo published a book, 3-D Seismic Imaging, that is the first text book to introduce the theory of seismic imaging from the 3-D perspective. The book is published by SEG in the Investigations in Geophysics series. During 2007 gave a one-day short course in 28 cities around the world as the SEG/EAGE Distinguished Short Course Instructor (DISC) . He is a member of AGU, EAGE, SEG and SIAM.



Alejandro Cabrales-Vargas obtained his Bachelors degree in Geophysics in the University of Mexico in 2002. He has been working for Petroleos Mexicanos since 2002, initially in seismic interpretation for oil and gas exploration, and more recently in the supervision of depth imaging processes. He obtained his Masters Degree in Geophysics in the University of Oklahoma in 2011. He joined SEP in the fall of 2014, and is currently working towards his PhD.

Robert Clapp received his B.Sc. (Hons.) in Geophysical Engineering from Colorado School of Mines in May 1993. He joined SEP in September 1993, received his Masters in June 1995, and his Ph.D. in December 2000. He is a member of the SEG and AGU.



Taylor Dahlke is a sixth year student with SEP. He received his B.S. in civil engineering from the University of California, Berkeley in 2012, and joined SEP in July 2012. Currently, he is working towards a Ph.D. in geophysics with his research focused on applying level set methodologies to perfom salt body segmentation. Taylor is a student member of SPE and SEG.

Huy Le earned his B.S. degree in Geophysics from the University of Oklahoma in 2012 and his M.S. degree in Computational Geoscience from Stanford University in 2014. He is now a PhD candidate in Geophysics. His research interests include anisotropy, full waveform inversion, finite difference method, and high-performance computing.

Stewart A. Levin was acting director of the Stanford Exploration Project during Jon Claerbout's 1993-4 sabbatical year. After a distinguished career in industry at Mobil and Halliburton, he has returned to Stanford as a senior research scientist in the Department of Geophysics.

Siyuan Yuan got his Bachelors degree in Civil Engineering, Tongji University, China. After graduation in 2016, He was admitted to the master program in Civil and Environmental Engineering at Stanford. He joined SEP in the summer of 2016, and is currently working on SDASA-1 related research.