Toward Realistic Modelling, Imaging and Inversion Testing

Daniel Trad and Ivan Sanchez

ABSTRACT

Research in academia often suffers from a limitation in the number of data sets employed for testing, resulting in a lack of feedback diversity that is crucial for comprehensive analysis. Applied research necessitates engagement with a broad spectrum of datasets, which significantly enriches research and development projects. Obtaining authentic datasets for publication in academia is not only challenging but also involves time-consuming preprocessing, making the pursuit of testing diversity a formidable task. Consequently, numerous tests are conducted on modelled data, often generated using similar algorithms employed in inversion processes, thereby giving rise to the "inverse crime scenario". Predominance of synthetic data testing in academia also comes as a consequence of the substantial difference in computational resources with industrial environments. Software developed in academia often lacks the capability in handling intensive computations with large seismic files with irregular acquisitions, capabilities that are required to work with real data sets used in industry. The consequence is a large gap between toy examples used in academia and realistic examples required for industrial use.

This report details the implementation advancements made in our seismic libraries, showcasing tests aimed at enhancing the reliability of results in diverse environments, including large models, salt environments, topography settings, and physical models. Furthermore, we elucidate the disparities between inverse crime scenarios and realistic situations. The immediate ramifications of these advancements include the ability to circumvent the inverse crime problem and conduct tests in a variety of environments. Moreover, we anticipate that this research will foster increased collaboration with industry and deepen our understanding of the practical capabilities of novel techniques developed at CREWES.

INTRODUCTION

Modelling seismic data is a key part of research for acquisition design, imaging and full waveform inversion (FWI). From the convolutional model to more sophisticated wave propagation with anisotropic visco-elastic 3D wave equations, there is a wide variety of approaches to simulate seismic data in different velocity models. The more sophisticated the approximation, the more varied the types of events we see on the data, but also the higher the computational cost.

The Finite Difference (FD) method is one of the most commonly used for modeling, especially structural modeling and even stratigraphic modeling, because it provides a wide range of options and its accuracy is sufficient for almost all applications. We use it for simulating surveys and forward and reverse modeling for reverse time migration (RTM) and FWI. The applicability of these techniques depends strongly on the computational cost of FD because usually a large number of modeling steps are required for iterative inversion. We could say that the computational cost of FD controls what we can do in research.

Traditionally, large elastic 3D FWI has been prohibitive slow except when done on very large computer clusters. However, thanks to the advances in computer science and high-performance computing, some resources permit us to go a long way in computing capability. For example, using Graphic Processing Units (GPUs) we have seen speeds up to 10-100X in finite difference algorithms. In addition, cloud computing has been made available to university systems with state-of-the art hardware. For example, most universities can access GPUs like the A100 (Dally et al., 2021) that are several times faster (4X in our tests) and one order of magnitude more memory (around 80Gb) than we have usually available by using regular desktops with standard general-purpose GPUs.

Although computational expense is a large part of the challenges we face doing research in academia, there are other limitations that can be solved with relative success by additional implementation complexity. For example, if we are going to test FWI on a real data set, our codes have to be able to read arbitrary geometries and deal with large data sets. In this report, we will discuss some work done towards improving our testing capabilities and also we will discuss different directions we are moving in CREWES in this regard.

This report is a continuation of a previous report (Trad, 2022), where the emphasis was mostly on creating an environment where students could model, migrate and invert seismic data with limited computational resources by using Graphics Processing Units (GPUs) with a 10-100X increased efficiency. In this new report, we discuss a further evolution of this work environment where students can test real data with irregular geometries in either 2D or, in progress, 3D. The immediate consequences of these changes are the possibility of working without the inverse crime problem and in a variety of environments. In addition, we expect this research can lead to an increased collaboration with industry and further understanding of the real capabilities of new techniques developed at CREWES.

DIFFERENCES BETWEEN SYNTHETIC AND REAL DATA TESTING

The development of migration and inversion algorithms starts invariably by creating synthetic data in a simple small model where we know the exact answer. This is essential to create new code that works correctly and to detect subtle implementation errors. Only by knowing the exact answers we can check that every step is correct. We are obligated to proceed with modelled data for most of the development process. The problem, however, is that in complex algorithms development never ends, in particular in academia. There is a tendency to continue testing with synthetic datasets and we have become so used to it that we become oblivious to the difference with real data.

The most important problem we fall into when testing with synthetic data is the inverse crime scenario. This refers to the case when the data are modelled with the same algorithm that we used for migration and inversion. Since all RTMs and FWI typically have a forward modeling phase with FD to create the source wavefield, it is common to use the same code to create a modeling program. In other methods that do not have an FD engine, like Kirchhoff migration for example, we don't fall into this because Kirchhoff uses ray tracing for modeling instead of FD. It is not impossible although not common, however, to see testing Kirchhoff migration by Kirchhoff modeling which is also an inverse crime scenario.

The inverse crime problem is well known but in general not that well understood. Sometimes people add random noise to address it but that random noise is not an issue. The following are important issues.

1. Artifacts and noise generated during the forward modelling become data. For example, artifacts reflecting from the border of the model are equivalent to a geometry that surrounds the model, converting a reflection into a transmission problem, which is significantly easier to solve. There are a few ways to improve on this, for example changing the cell size from the modeling to the inversion steps and therefore making the artifacts irreversible (Trad, 2021).
2. When the algorithm involves an iterative data fitting, like in the least squares migration (LSMIG) or FWI, the residuals in the inverse crime scenario will contain information about the model errors, producing a perfect gradient estimation and therefore facilitating convergence during the inversion. In real data, the residuals contain, in addition to model parameter errors, also deficiencies in the modeling algorithm and acquisition noise that lead to gradient errors too. We can improve on this by using a more complete physics for forward modeling program than for inversion.
3. Most synthetic tests typically create data in a regular layout with shots and receivers on a surface at constant intervals. In these geometries, each shot illuminates every receiver, leading to very large offsets (as long as the model). Real data have geometries with a shot patch that changes along the survey, with variable shot and receivers intervals and elevations. To work with real data is necessary to make our geometries have an independent shot/receiver position for each seismic trace.
4. Noise in real data can be very difficult to simulate properly with synthetics. Random noise does not represent the complexities of coherent noise generated on the near-surface. One way to address this is by algorithms to simulate Rayleigh waves and scattering noise on the surface as in Sanchez-Galvis (2023).
5. Data contains internal and surface multiples. These can be included in modeling but if we also generate them during the inversion then they become additional data. Although, much work has been done to use multiples as data, using them in the inverse crime scenario is just equivalent to prove that our algorithm are reversible, not that we can actually use them in real cases.

A MADAGASCAR DATAFLOW FOR ARBITRARY GEOMETRIES

The first issue that we will address in this report is how to work with arbitrary geometries with open-source processing systems. Outside the open-source scope, all commercial software is designed to efficiently handle real data but they are not freely available to distribute Academic software. Among the many open-source systems, in CREWES we often work with Crewes Toolbox (Margrave, 2000), Seismic Unix (Stockwell Jr, 1999), Madagascar (Fomel et al., 2012) and more recently experimenting with Devito (Witte et al., 2019). Many other libraries in Python and Julia have become available in the last few years. All these methods have advantages and disadvantages and a choice among them depends on the type of intended research. Seismic Unix for example, has a design that is very flexible for general processing since it uses a datatype similar to SEGY files, the standard for exchange of seismic data. This is an "array of structures" design, which typically has to be converted to a "structure of arrays" design when working with computationally demanding modules. Therefore, it is a great choice for general processing but it does require reformatting the data for efficiency. The CREWES toolbox is very useful for prototyping but for our FD problems seems too slow and memory-demanding. Devito has a great performance but works at a very high level and, at least in our tests, seems more demanding for memory consumption than compiled code.

In this work, we have chosen Madagascar (Fomel et al., 2012) which has a very flexible design that permits to combine different tools in one data flow using "Scons" (Knight, 2005; Fomel and Hennenfent, 2007) for the dataflow and building. "Scons" has a similar type of functionality as Makefiles (only modify what has changed), allowing a target-oriented processing. This permits to create a very complex dataflow, using all the flexibility of modern Python, and at the same time allowing the user to focus on specific parts of the dataflow by using targets. In addition, it keeps track of dependencies, deciding on demand which parts of the dataflow are up-to-date and which others have to re-run for a given target after some parameters have changed. Madagascar serves mostly as a glue for the different processing steps, allowing to run many or all of the tools mentioned above. For all the examples presented in this report, the modules are actually written in standard C++, using CUDA libraries in most cases, and they can be taken outside of Madagascar with minimum effort since the Madagascar libraries are only used for I/O and display. The other choices mentioned above have advantages in terms of simplicity of writing software but they represent higher-level choices that we feel require more specialization to a particular platform and we want to keep our tools low level enough to implement our algorithms using a traditional software style.

One difficulty we found with Madagascar is that it frames all seismic data as a hypercube. For example, a typical 2D dataset would be a cube with dimensions given by the number of shots, receivers and time samples as 3rd, 2nd and 1st dimensions. This cube is saved as a binary file which can be accessed with any system. The header, however, is reduced to the axis along each of these dimensions (first sample, number of samples and intervals). Although it is possible to save arbitrary information in an ASCII header attached to each file (for example the data "history"), handling a "trace by trace" detail information of sources and receivers is not directly implemented. To be able to access real data in a generic manner, we need to carry the information in separate files, which act as a database for the geometry.

In Figure 1 we explain the general dataflow. SEGY data, which contains an array of structures (seismic traces), are read into three files:

1. A variable-fold shot file with a sequential arrangement of seismic traces (size equal to the number of traces times number of samples).
2. A source file with 3 columns, x, y and z, for the location of each trace's shot (size equal to the number of traces times 3).
3. A receiver file with 3 columns, x, y and z, for the location of each trace's receiver

(size equal to the number of traces times 3).

These 3 files constitute the full data set, including geometry, that have to be read by seismic modules. The variable-fold shot file is typically converted to a hypercube by using zero padding which can be efficiently accessed by seismic modules. Notice that introducing an arbitrary geometry capability requires to develop several tools in Madagascar, as follows:

1. An extension for segy reading module to output the additional files.
2. A program to convert the sequential variable fold data to hypercubes.
3. A program to manipulate data selection with irregular geometries.
4. Changes on existing modules to read additional geometry files and use irregular design internally.

To carry the geometry information across modules, an object oriented design is crucial. In our implementation, there is a class that manipulates geometries and is shared across all modules. This class is in charge of reading the geometry files and making the information available to modules that access the data through 2 (4) indexes in 2D (3D) data for shots and receivers. Notice that this "geometry" class acts as a database handler, where the database is in the two files containing the trace by trace information. Although significantly more complex than using the standard Madagascar files with 3-axis ascii information, this complexity is the price to pay to use real data. This design has some resemblance to databases used in commercial processing systems.

Finally, in addition to supporting this dataflow, we also need a way to generate synthetic geometries with arbitrary x, y, and z coordinates for shots and receivers. This is implemented in 2D only at the moment in a module that creates the two geometry files with the flexibility to locate shots and receivers on a given horizon. Also, given a minimum-maximum patch size, it can generate the fixed shot-patch array type of geometry we normally find in real data. A 3D version of this program will be implemented in the future.

TOWARD REALISTIC RTM TESTING

Inverse crime as high-end baseline

An interesting sub-product of the tools explained above is the capability to create synthetic data using a geometry read from a real data set. This allows one to test algorithms using synthetic data by matching a real dataset before applying to the real case and understand the problems we will face. As an application, we show in this section a series of comparisons between real and synthetic data with the same geometry, which also illustrates the inverse crime problem. As we will see, the inverse crime result can be taken as an upper quality limit in the best possible scenario.

In Figure 2, we see the Pluto velocity model that we will use in several tests. Notice the 6 small reservoir areas in between and under the salt. We will use synthetic data from a third

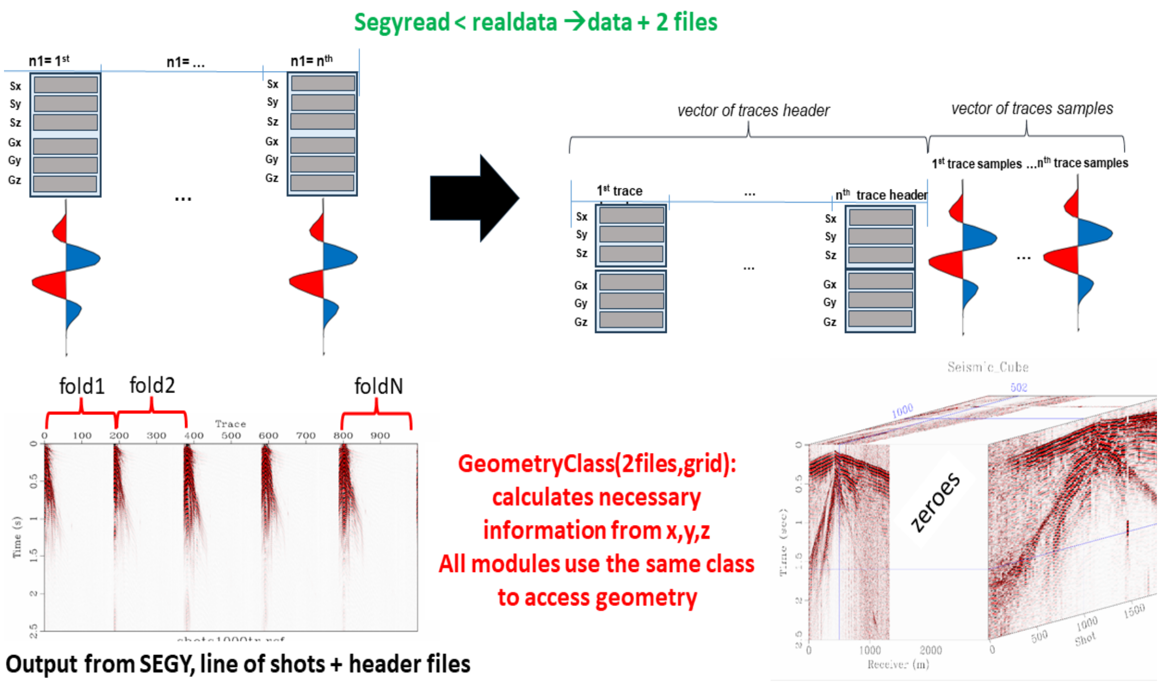

Segyread < realdata →data + 2 files
n1= 1st
n1= ...
n1= nth
Sx
Sy
Sz
Gx
Gy
Gz
vector of traces header
vector of traces samples
1st trace samples ...nth trace samples
1st trace
nth trace header
fold1
fold2
foldN
Trace
Time (s)
GeometryClass(2files,grid): calculates necessary information from x,y,z
All modules use the same class to access geometry
Seismic Cube
zeroes
Time (sec)
Receiver (m)
Shot
Output from SEGY, line of shots + header files

FIG. 1. Information from SEG files (array of structures) is read into two geometry files that contain trace by trace information and a binary file that contains a sequential arrangement of traces without any regularity assumption. The output shots have a variable fold, and the geometry files contain x, y z for the shots and receivers of each trace. The variable fold shot file can be converted to a hypercube by zero padding.

party, stored in the form of SEGY files, and then compare with data modelled by ourselves. Although the dataset from the third party is also synthetic, it has been created by using different algorithms, wavelet and parameters, which we don't have access to. Therefore for our purposes, it will serve as real data. Certainly, real data could have other issues like marine statics, adjustments by the movement of the streamers and additional noise, but these effects would have normally have been taken care of by a careful processing. What has not been taken care of is multiple elimination, therefore we can use the multiples to illustrate the inverse crime issue. However, we will have a great advantage over a real dataset case on that we will be using a (too) good velocity model, shown in Figure 3, but will relax that advantage in the next section when we discuss FWI.

To illustrate why we claim that there is no inverse crime in this example, we show in Figure 4 the difference between the original dataset and the data generated by ourselves. The differences that we see, bandwidth, multiples, wavelet, scaling and attenuation, take us away from the inverse crime scenario because our RTM code will predict wavefields matching our forward modeling on the right, not the input data on the left.

Figure 5 shows the RTM from the original data set. This migration was done from 671 shots, 18000 samples along time and a model with 1201×5561 cells, each with interval dz=7.6m and dx=7.6m. The velocity model was smoothed with a 3-pass rectangular window of 150m in x and z (Figure 3). The migration took around 4 hours in a Nvidia GPU A100, around 4X faster than in a general purpose GPU (GTX3060).

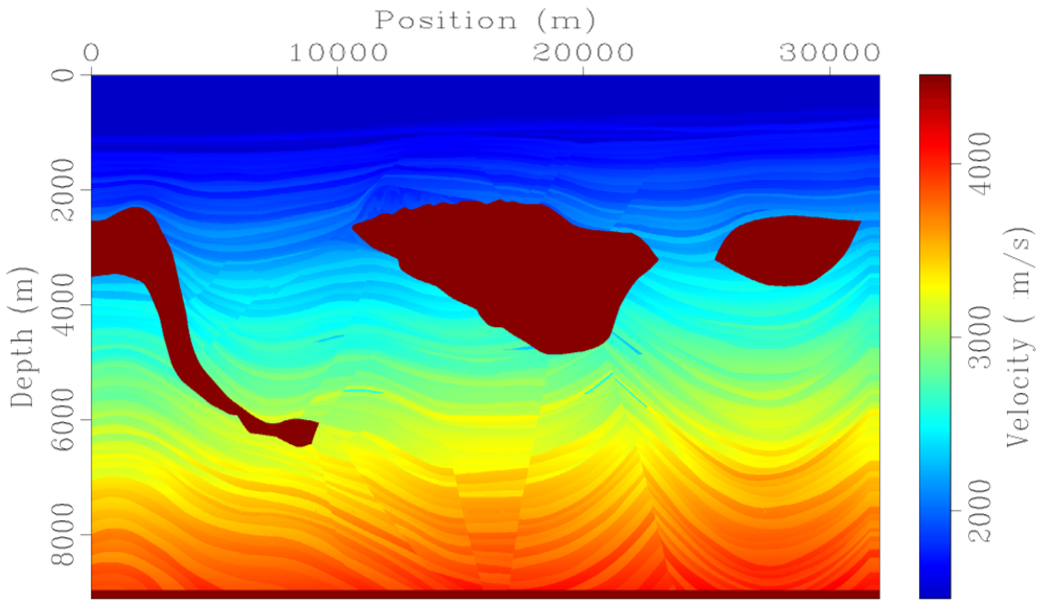


FIG. 2. Pluto velocity model.

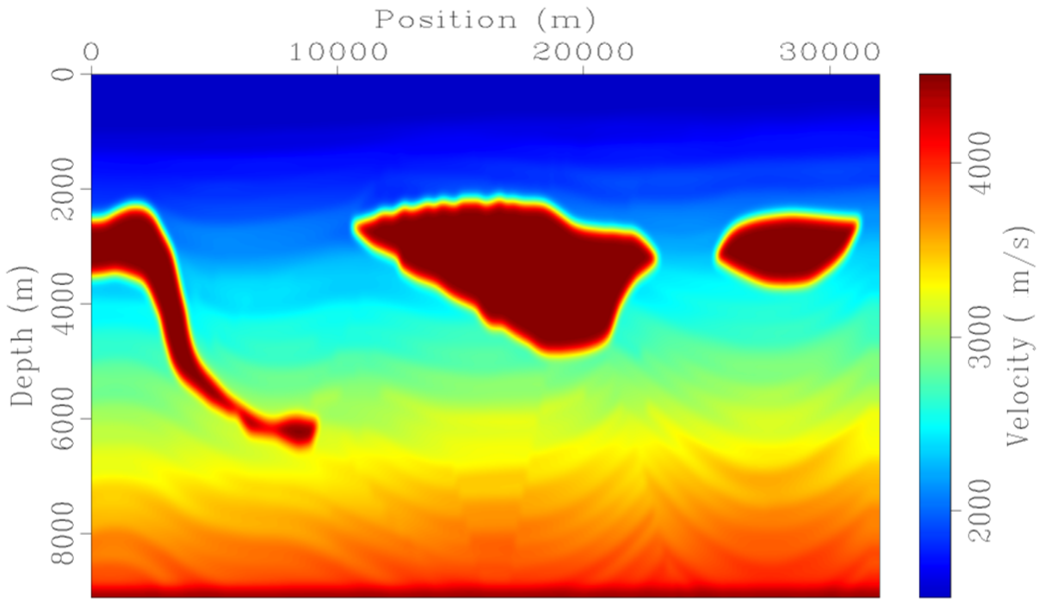


FIG. 3. Pluto migration velocity model for RTM.

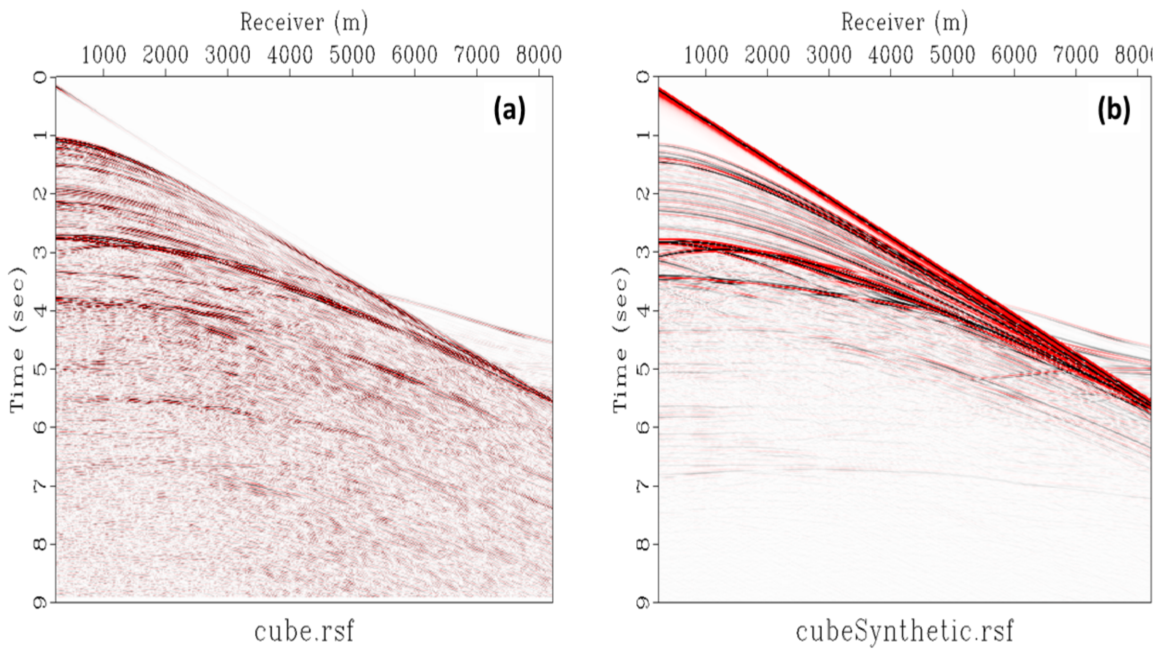


FIG. 4. One shot from SEGY and the same shot from our synthetic.

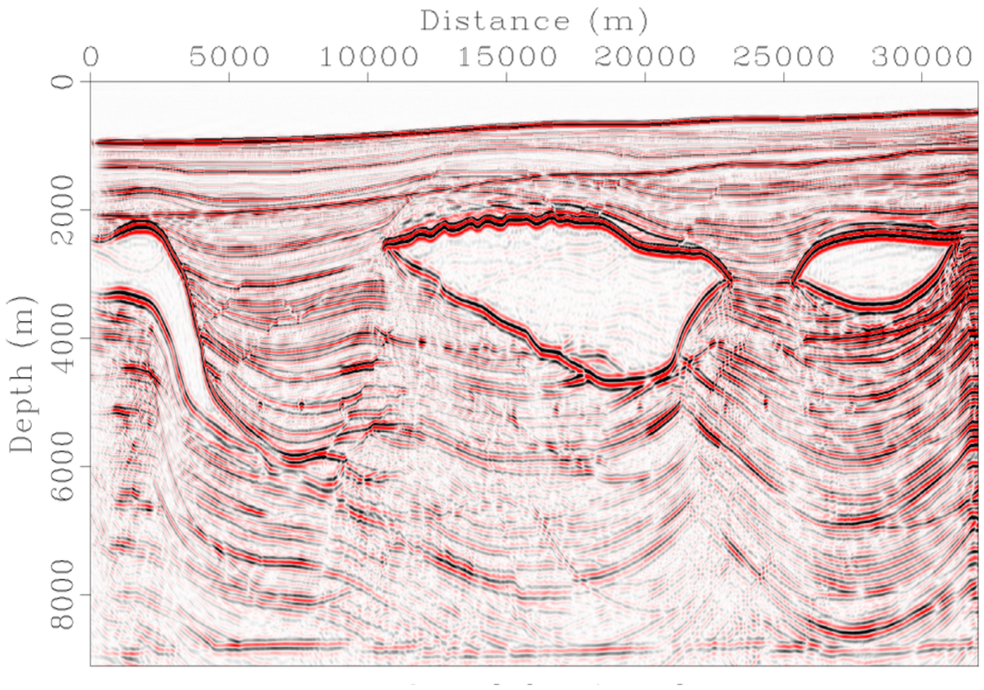


FIG. 5. RTM for Pluto using the original data (no inverse crime).

At this point, to emphasize the goal of this paper, it is worth to compare this test with a typical run with the Marmousi model using 96 shots. In this case, the model is 15 times bigger, the data set is 35 times bigger, so we are doing a problem that is approximately 540 times larger than the Marmousi test (2.5 orders of magnitude). If we were doing this in 3D, assuming a model with another 200 cells perpendicular to the main line, we would need to crunch 100000 times more numbers than when running tests in the Marmousi model. These numbers illustrate the importance of scaling up the size in our tests to take into account computational limitations. Even our test here becomes really modest with the computational cost of 3D imaging and inversion. However, we believe that by using a 540X larger problem than for the Marmousi model, we are moving towards the right direction.

The image in Figure 5 seems fine, but could we get a better result? We can use an inverse crime scenario to answer this question. Figure 6 shows a RTM image obtained with the same algorithm as before. The image has improved because we modelled the data ourselves with the original geometry (inverse crime) and used Perfectly Matched Layers (PML) on the top to avoid free-surface multiples. We can see all the reservoir areas with perfect clarity, as well as faults, and layers above and under the salt. Clearly, since the migration algorithm, the velocity model and the geometry are all identical, the large improvement in the image comes from the inverse crime. This may come in two flavours. For one side, artifacts in the boundaries act as data. For another, the synthetic data did not have multiples. Because the PML boundaries are usually quite efficient in suppressing artifacts the second factor seems to be the most influential in this case.

To further test this point, we run another test in Figure 7, where we generated data with multiples. In this case, we break the inverse crime by using a different FD algorithm. We see that now there is significant amount of noise under the salt. In this case, we are half the way between the best possible model and the one we obtain with external data. Notice that, the inverse crime image would also provide an upper quality limit to benchmark the efficiency of surface multiple attenuation algorithms.

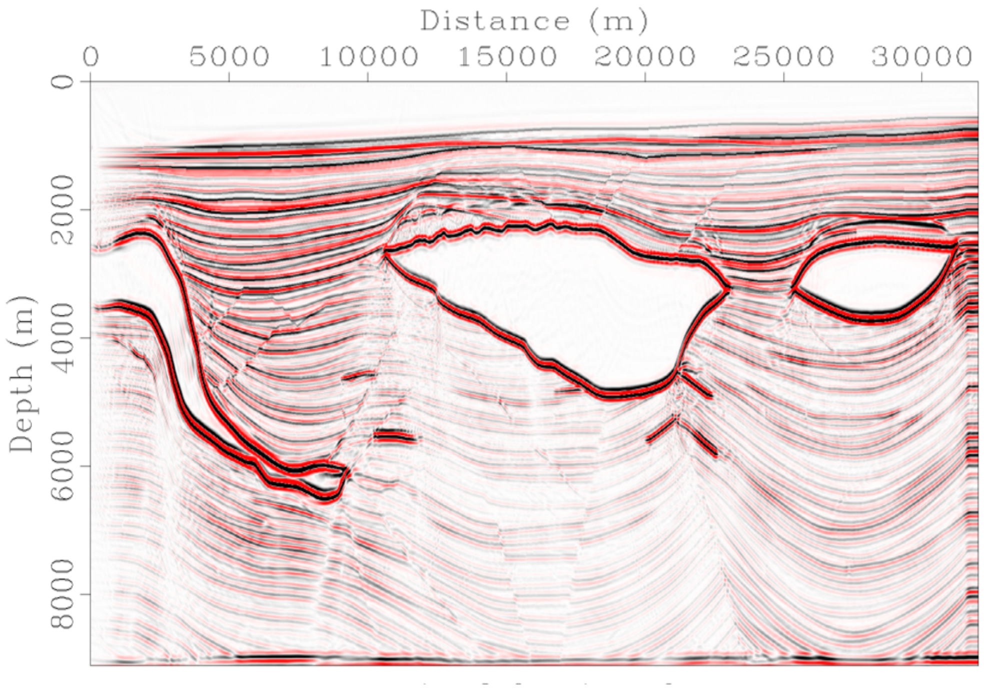


FIG. 6. RTM for the Pluto model using our own modelled data, therefore running into an inverse crime scenario. This would be the best possible result.

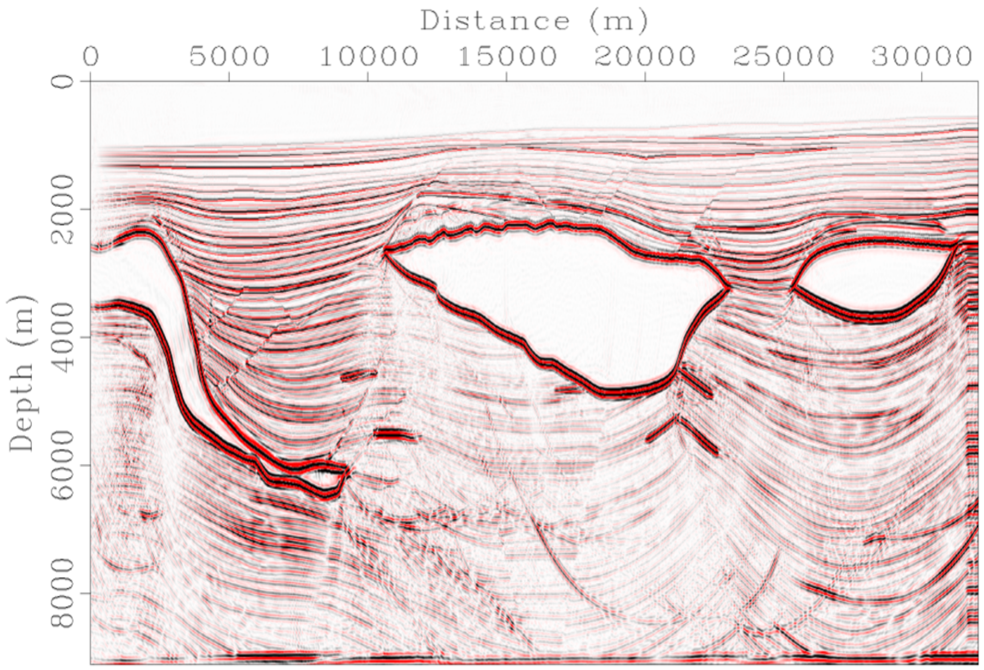


FIG. 7. RTM for the Pluto model using our own modelled data but this time breaking the inverse crime by including multiples during modelling but not during migration. Also, here the modelling and migration algorithms are different.

RTM from topography

The next scenario we want to discuss is realistic testing for surface seismic on top of complex topography. Because of the complexities of modeling, this is more challenging with FD than with ray tracing methods. We will discuss this challenge in different stages, increasing the realism as we go through the examples.

In Figure 8 we present RTM examples from a well known "Foothills" model. These examples all have different degrees of realism, and therefore they don't have the same value. First, we make a flooded model, similar to using a flat datum, and put the geometry at the top (Figure 8a). This could be considered normal practice except that we make the inverse crime of generating the data from the datum rather than moving the data with near surface statics. Near-surface statics in this type of topography would introduce significant data distortion unless it were done with a wave equation algorithm and good knowledge of the upper velocities. Therefore, the result of the flooded model that we show in Figure 8d should be considered very unrealistic. In Figure 8c we see a Kirchhoff Depth migration from topography using the original data (whose details are unknown to us). Such an image is an excellent test of the migration algorithm since there is no inverse crime, except perhaps that the migration velocity model is a smoothed version of the original. If the model were wrong the image would deteriorate. Also, we could create data with near-surface noise as we will see in the next test. Therefore, although the result in Figure 8d is better than in c), the last one is a more accurate test. In Figure 8e, we see the RTM from synthetic data created by us from true topography (from the velocity model in Figure 8b). This is a better test than in Figure 8d but it is still an inverse crime. Finally in Figure 8f, we see the RTM without inverse crime, where the input data was generated from topography, the same as the migration, and the data were created by a finite difference algorithm from a third party, whose parameters or details are unknown to us. Clearly, the image degrades but the test can give some faithful picture of how the algorithm would behave in reality and how it would compare to the Kirchhoff migration in Figure 8c. Again, if the model were wrong, all results would be much worse. For imaging tests, however, a correct velocity model is a common assumption. We should point out that adding velocity model errors would probably affect the Kirchhoff migration less than it would affect the RTM image.

However, when considering land data in complex topography, this is just a first step. The next step is to consider the complexities of the near-surface in terms of data and migration noise, which we will see in the following tests. To properly generate surface noise, we need to switch algorithms and use an elastic migration with the proper boundary conditions to honour the topography. For the following example, we add to our RTM an FD algorithm developed in Sanchez-Galvis (2023) and shown in Figure 9.

This FD method is designed for elastic wave propagation in environments with irregular topography, and, like all implementations in this report, is optimized for GPU using CUDA. It employs the 2D or 3D elastic wave equation (as shown in Figure 9a) and introduces an innovative Unstructured Index Array Representation (UIAR) for implementing the Parameter Modified (PM) formulation. This formulation adheres to the free-surface condition necessary for topographic variations (Figure 9b). The solver discretizes velocity and stress fields in the numerical medium using a staggered grid scheme (Figure 9c) and incorporates

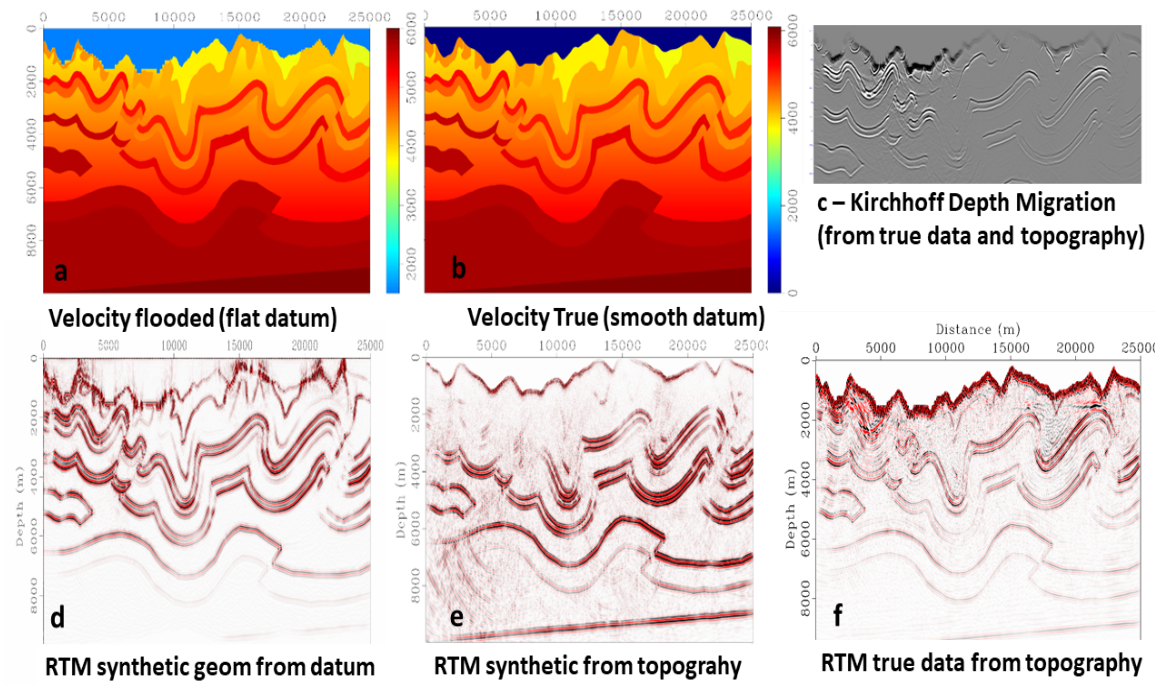
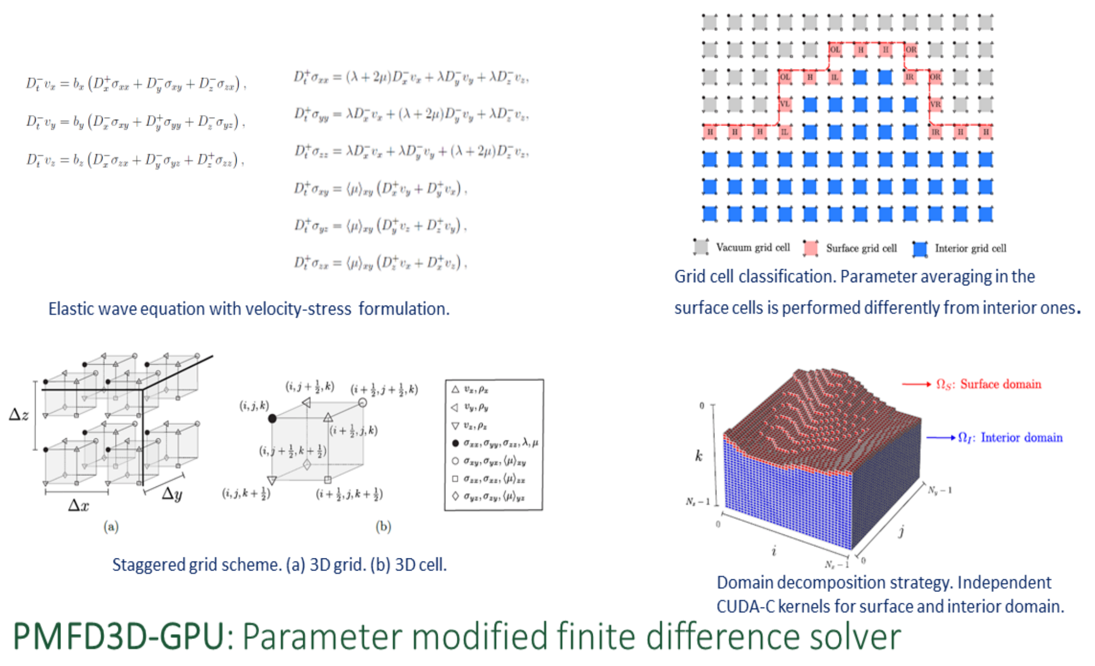


FIG. 8. A Foothills model with a) flooded topography (flat datum), b) real topography. c) Kirchhof depth migration from real topography and original data. d) RTM from synthetic data on top of the flooded velocity, e) RTM from synthetic data modelled from the true topographic terrain, and f) RTM from true topography using original data. d) and e) are inverse crime scenarios, c and f) are not.

a subdomain decomposition strategy. This approach, illustrated in Figure 9d, is efficient in minimizing memory usage while maintaining high computational performance. It involves dividing the effective domain into surface and interior subdomains, denoted as Ω_S and Ω_I, respectively. The surface subdomain covers grid cells at the surface, storing fifteen field parameters (in 3D), while the interior subdomain includes internal grid cells where original FDM equations are applied, with parameter averaging at each timestep.

This algorithm can produce the Rayleigh and scattered waves (ground roll) that we see on the near-surface. By incorporating these waves, we are one step closer to reality when evaluating RTM in complex topography. We show in Figure 10 the SEAM II V_p, V_s and density ρ models and the P, V_x and V_z wavefields from synthetic data from topography. The near-surface generates significant noise in the data which closely resembles the type of noise we see in real data. In Figure 11 we see the corresponding migration velocity models (150m smoothing) and the PP and SS migrations from the synthetic data. This test has inverse crime since we generated the data using the same elastic algorithm as the migration, but it does illustrate the capability of the algorithm to generate the near-surface noise and the need to include this noise in our testing. In this case, we see the migration produced a good image, however, the surface is not nearly as complex as the Foothills model we saw before and revisit next. Since we don't have an elastic version of the Foothills model, we create one by scaling the P velocities to get the S velocity and using constant density below the surface. Here we will have to rely in our data since the original dataset is acoustic, but will honour the acquisition geometry from topography.

In rough topography like the Foothills velocity model, the discretization jumps due



$$D_t^- v_x = b_x \left(D_x^+ \sigma_{xx} + D_y^- \sigma_{xy} + D_z^- \sigma_{zx}\right),$$
$$D_t^- v_y = b_y \left(D_x^- \sigma_{xy} + D_y^+ \sigma_{yy} + D_z^- \sigma_{yz}\right),$$
$$D_t^- v_z = b_z \left(D_x^- \sigma_{zx} + D_y^- \sigma_{yz} + D_z^+ \sigma_{zz}\right),$$

$$D_t^+ \sigma_{xx} = (\lambda + 2\mu) D_x^- v_x + \lambda D_y^- v_y + \lambda D_z^- v_z,$$
$$D_t^+ \sigma_{yy} = \lambda D_x^- v_x + (\lambda + 2\mu) D_y^- v_y + \lambda D_z^- v_z,$$
$$D_t^+ \sigma_{zz} = \lambda D_x^- v_x + \lambda D_y^- v_y + (\lambda + 2\mu) D_z^- v_z,$$
$$D_t^+ \sigma_{xy} = \langle \mu \rangle_{xy} \left(D_x^+ v_y + D_y^+ v_x\right),$$
$$D_t^+ \sigma_{yz} = \langle \mu \rangle_{xy} \left(D_y^+ v_z + D_z^+ v_y\right),$$
$$D_t^+ \sigma_{zx} = \langle \mu \rangle_{xy} \left(D_z^+ v_x + D_x^+ v_z\right),$$

Elastic wave equation with velocity-stress formulation.

Grid cell classification. Parameter averaging in the surface cells is performed differently from interior ones.

Staggered grid scheme. (a) 3D grid. (b) 3D cell.

Domain decomposition strategy. Independent CUDA-C kernels for surface and interior domain.

PMFD3D-GPU: Parameter modified finite difference solver

Vacuum grid cell
Surface grid cell
Interior grid cell
Δz
Δx
Δy
(a)
(b)
(i,j,k)
$(i,j+\frac{1}{2},k)$
$(i+\frac{1}{2},j+\frac{1}{2},k)$
$(i+\frac{1}{2},j,k)$
$(i,j+\frac{1}{2},k+\frac{1}{2})$
$(i,j,k+\frac{1}{2})$
$(i+\frac{1}{2},j,k+\frac{1}{2})$
$\triangle$ v_x, ρ_x
$\triangleleft$ v_y, ρ_y
$\triangledown$ v_z, ρ_z
● $\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \lambda, \mu$
○ $\sigma_{xy}, \sigma_{yx}, \langle\mu\rangle_{xy}$
□ $\sigma_{xz}, \sigma_{zx}, \langle\mu\rangle_{xz}$
◇ $\sigma_{yz}, \sigma_{zy}, \langle\mu\rangle_{yz}$
Ω_S: Surface domain
Ω_I: Interior domain
k
i
j

FIG. 9. The method of modeling from topography (Sanchez-Galvis, 2023)

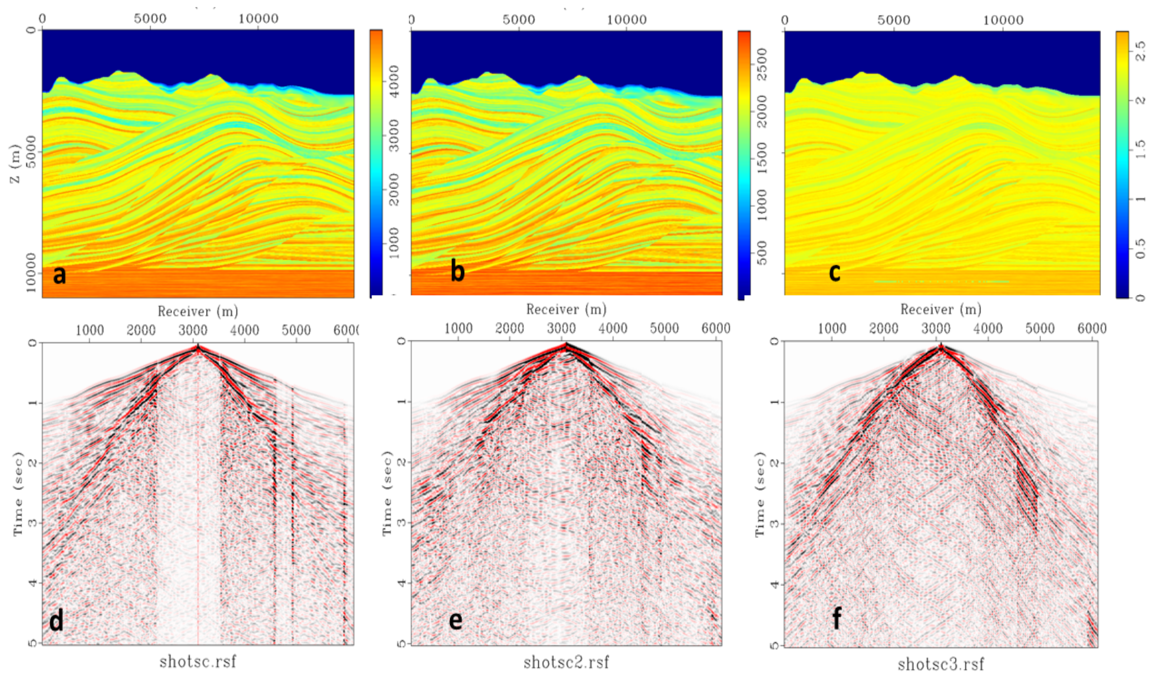


FIG. 10. The elastic SEAM 2 model and the results of elastic modeling from Topography. a) P velocity, b) S velocity, c) Density, d) PP data, e) Vx data, f) Vz data.

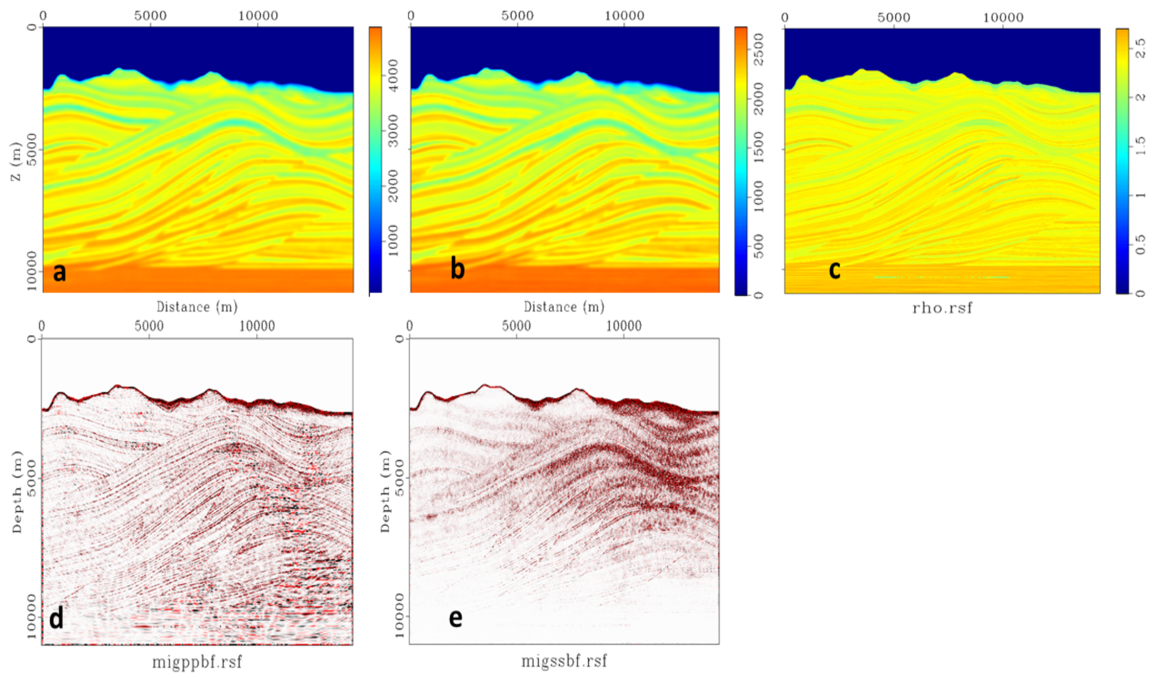


FIG. 11. The elastic SEAM 2 migration models and the elastic migration from Topography: a) P velocity, b) S velocity, c) Density, d) PP RTM, e) SS RTM,

to coarse velocity cells produce unrealistic scattered noise. To address this problem, we interpolated the velocity models to a fine grid, $5m \times 5m$, and then smoothed the surface just enough to remove the cell breaks. Also, we raise the topography enough to allow the Finite Difference acquisition to honour the original depth (we need sources and receivers to be at least two cells below the contour). In Figures 12a, b, c we show the V_p, V_s and ρ models after some light cell smoothing on the surface to remove discrete jumps in the cells.

Figure 12d shows one shot from the original dataset. In Figure 12e we see the same shot from our acoustic modelled data from topography. Neither dataset obey the boundary conditions of the rough topography, therefore they are just approximations of reality. The geometry is properly set on the surface, but amplitudes would be slightly different from reality. Figure 12f shows one shot from our elastic-modelled data using proper near-surface boundary conditions. We don't know how the external data in Figure 12d was modelled but shows the best quality in terms of direct wave continuity. However, this dataset does not have any near-surface noise as in Figure 12f. Reality and wavefield continuity are not always together.

The migration results, shown in Figure 12 g, h and i, correspond to the original data, our acoustic modelled data, and our elastic modelled data, respectively. The PP image from elastic modeling is comparable to the one obtained from the original dataset. In one sense, the test is less realistic since we are modelling the data ourselves, but on the other, we are now taking into account the elastic wave equation and considering the noise generated on the near surface. Since we are using the elastic wave equation we can obtain the reflectivity for different components. In Figure 13 we see on the left, the RTM result from the SEGY acoustic data (for comparison), at the center the PP RTM obtained from our own elastic

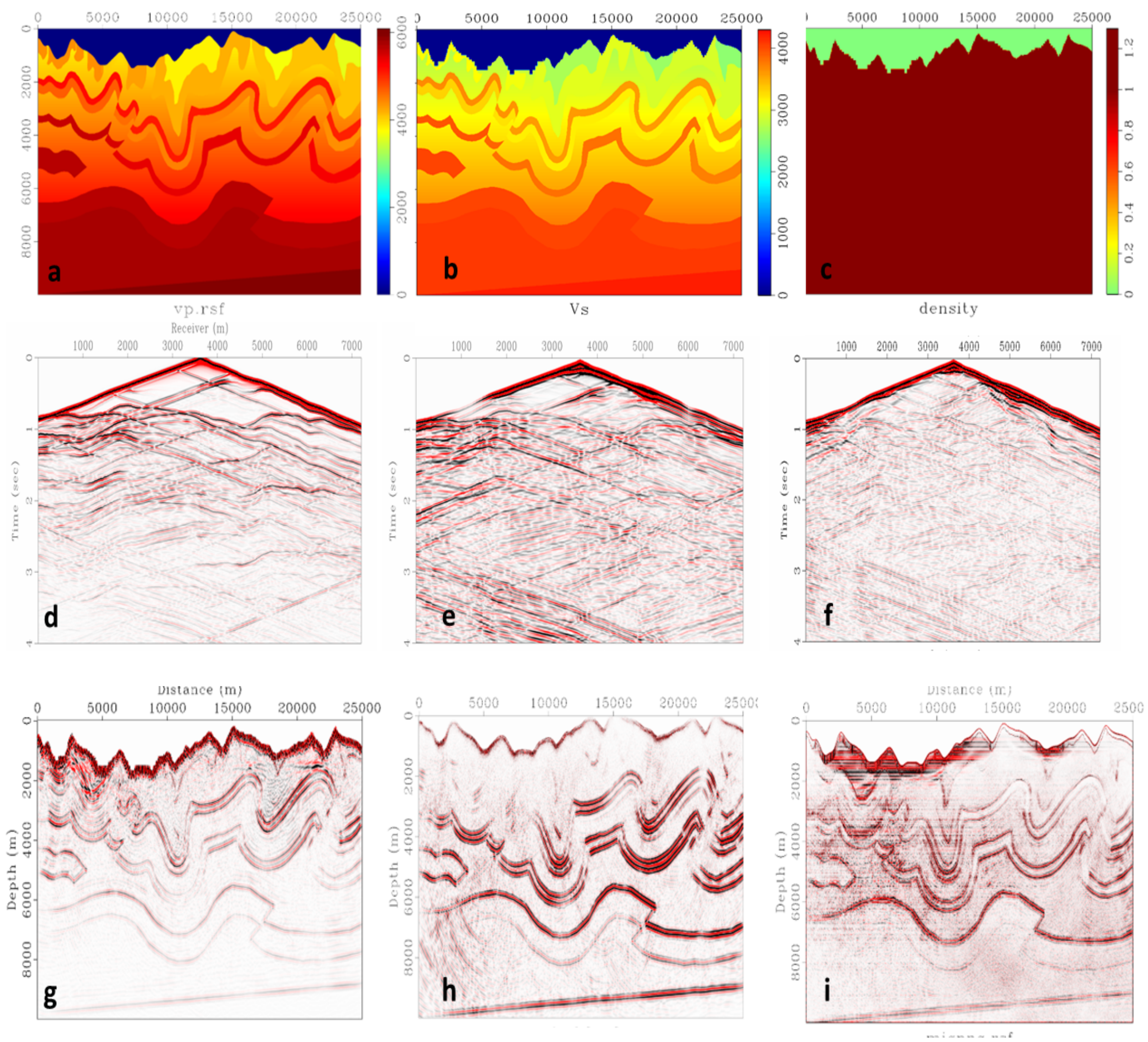


FIG. 12. A Foothills model with true topographic effects with cell smoothing on the surface. a) P velocity, b) S velocity, c) Density. d) Acoustic data modelled from a third party (from SEGYs in Madagascar). e) Our own acoustic data modelled from topography. f) Our own PP data modelled from topography following the elastic modeling with proper boundary conditions. g) migration from original SEGY data. h) RTM from synthetic from topography. i) PP RTM from elastic modelling from topography.

modeling and elastic RTM, and at the right the same for SS. Although some noise on the surface is still affecting the overall quality, we can judge the effect of the near-surface on the imaging.

REALISTIC FULL WAVEFORM INVERSION

Breaking the inverse crime by changing grids

Full Waveform Inversion (FWI) (Tarantola, 1984) is one of the most powerful but also difficult inversion algorithms to apply in real data processing. FWI can produce detailed velocity information about the subsurface by fitting modeled data to observations (Schuster, 2017). However, its high computational cost has been an obstacle to its application until GPU implementations (now the norm) appeared. For example, Yang et al. (2015), describes

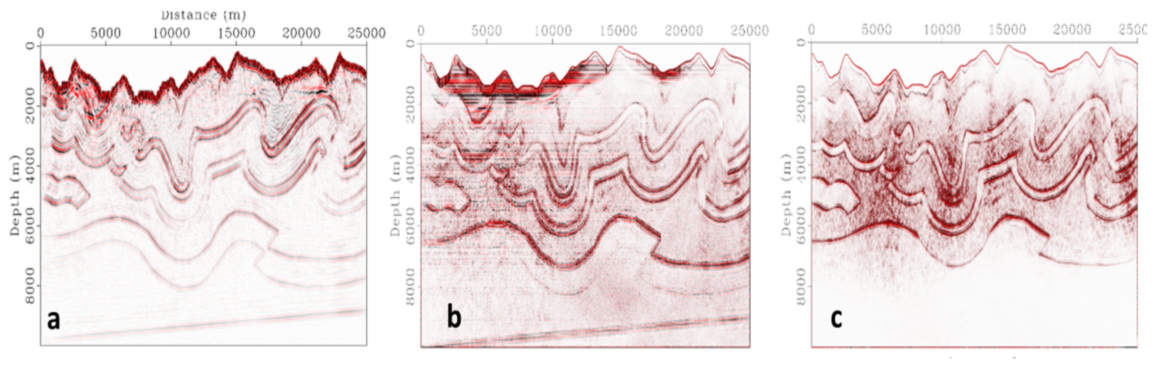


FIG. 13. RTM results from the Foothills model with true topographic effects and near surface noise. a) Acoustic migration from original SEGY data. b) PP-RTM from our elastic synthetic. i) SS-RTM from our elastic synthetic. b) and c) show the effect of near-surface noise.

early work on a GPU-FWI implementation that we have used as a starting point for the FWI used in this report.

Successful FWI is the result of success in several different areas, which up to some degree, can be considered independent: preprocessing, initial velocity model from tomography, a good and efficient migration algorithm to calculate the gradient, and an optimization algorithm that takes care of the non-linear nature of FWI that leads to cycle-skipping. That is why we see a large number of tests with inverse crime, since this allows one to focus on one aspect of the problem. However, it is important to realistically test FWI because the size of the gap between reality and inverse crime can be much larger than expected at first.

As said before, we need a good and efficient RTM to calculate the gradient and a good optimization method. In this report, we will use a Non-Linear Conjugate Gradient (NLCG) algorithm presented in (Yang et al., 2015). Our changes are mostly on the RTM algorithm and on the dataflow. Figure 14 shows an FWI result from the "Overthrust" model. This is a 3D model commonly used for testing FWI but here we only use a 2D slide of the 3D model. For this result, the FWI was run in one stage. Since the data were generated by ourselves, this is a typical inverse crime scenario. In Figures 15 and 16 we show also the popular Marmousi and the Foothills model we used in the previous section (Figures from Trad (2020)). These two examples show a slightly more realistic test than the Overthrust example. Here the original data were generated in a fine grid with high frequencies, and the inversions were done in stages starting from coarse models and refining them at each stage. The reason why these tests have less inverse crime than in Figure 14 is because we started from data generated in a different grid. Therefore, all artifacts are properly treated as noise not as data, since a reverse run of the algorithm would not reproduce the input artifacts.

However, in the Foothills example, the model has been flooded with a high-velocity layer and the data generated from the flat datum. As a consequence, we obtain a resolution of the upper section that can not be obtained in real data.

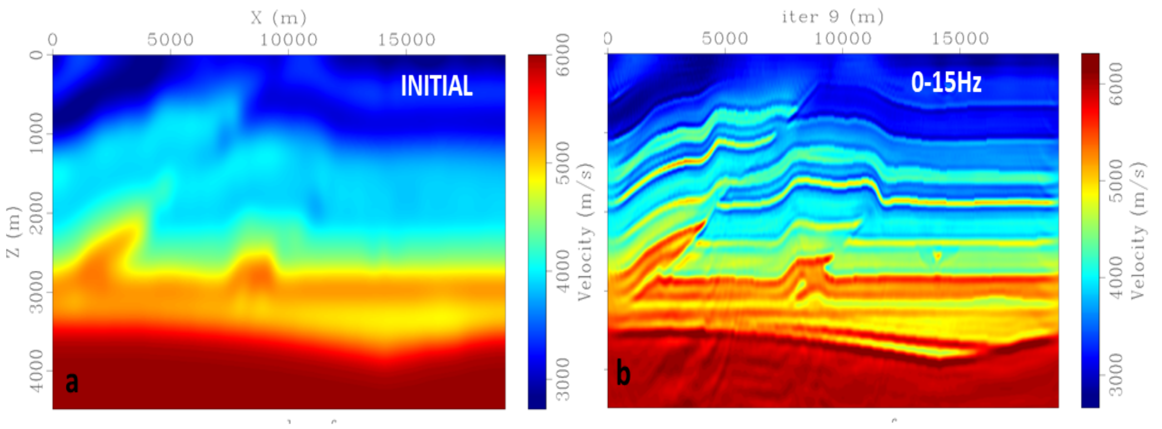


FIG. 14. Overthrust model. Left: initial velocity. Right: FWI result after 10 iterations. There is no attempt to eliminate the inverse crime problem.

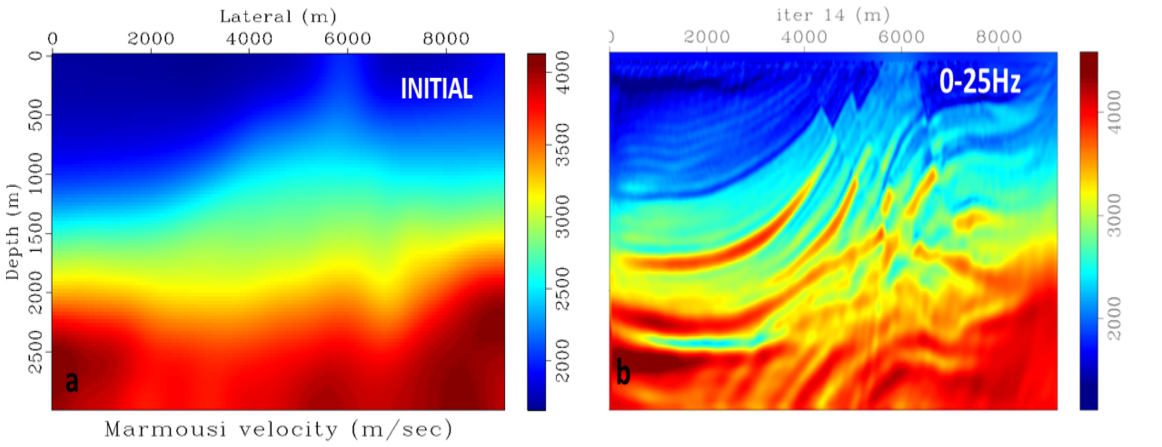


FIG. 15. Marmousi model. Left: initial velocity. Right: FWI result after 25 iterations in three stages. Inverse crime is mitigated by creating the data at a different resolution than the inversion.

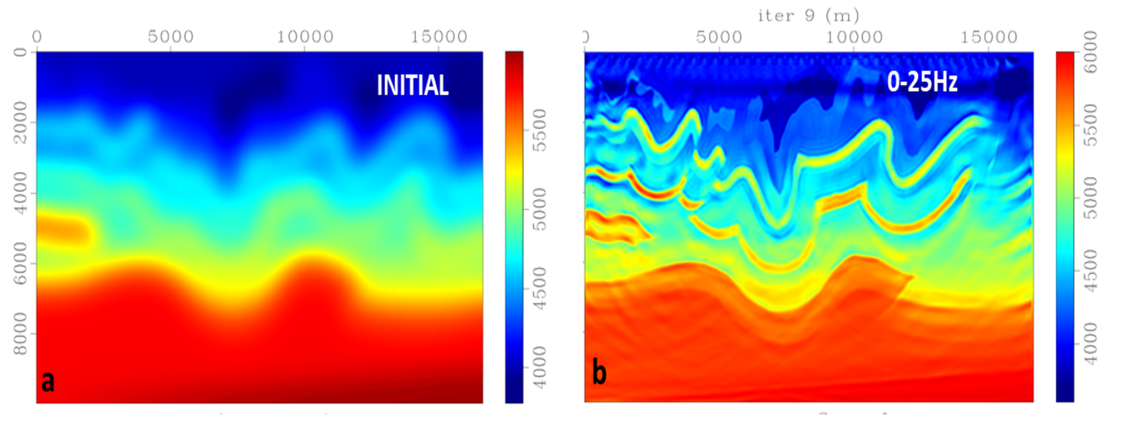


FIG. 16. Foothills model. Left: initial velocity. Right: FWI result after 20 iterations in three stages. As before, the grid size change mitigates the inverse crime. However, data was generated from a flat datum, producing a different version of inverse crime.

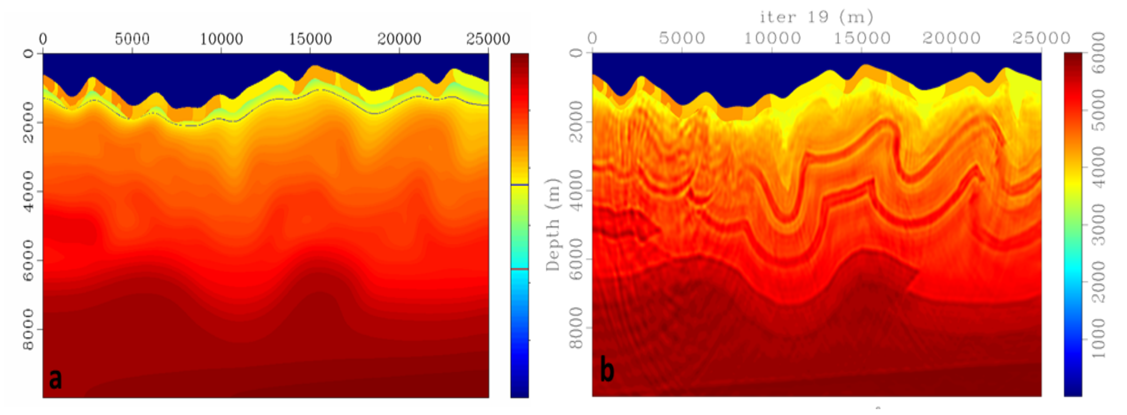


FIG. 17. Foothills model from topography. Left: initial velocity. Right: FWI result after 20 iterations. Comparing with the previous result, the image has less definition but the result is more realistic.

FWI from topography

We can now move one step further in realism by applying FWI directly from topography. Typically, processing on rough terrain, including imaging and FWI, results in better quality when working from a smooth datum that closely matches the topography. The reason is that any datuming approximation in rough terrain would distort the amplitudes. When the surface is very rough, like in the Foothills' model, realistic results from topography will suffer degradation in quality. In Figure 17 we see the Foothills initial model on the left with very strong smoothing except on the first few meters of the surface. Normally we know exactly the topography although it may be too optimistic to assume we know the velocity in the first few meters of depth. Let us assume a very good near-surface refraction survey has been done or a near-surface tomography has been applied. On the right, Figure 17b, we see the FWI result after 20 iterations. Most structures have been well-defined in depth but not as well as in Figure 16b. The main decrease in the quality of the model compared to the (unrealistic) flat datum test is on the left of the model. However, this test is much closer to reality. Some further steps to eliminate the inverse crime, other than using real data, would be to create an elastic model with the near-surface noise as in Figure 12f and apply acoustic FWI on the PP component. That FWI test including realistic noise has not been done for this report yet, but it is one of the goals of this research.

FWI for Salt

Another challenging test for FWI is for salt environments. Typically the top of the salt produces a significant reflection of energy that leads to very poor illumination deeper in the structure. In addition, the salt boundary would create many artifacts because of its rapid lateral variations, not too different from the Foothills model we saw before. Because of the strong reflectivity on top of the salt, it can be difficult to properly invert velocities above or below them. Figure 18a, shows an initial velocity model where the salt bodies are present but not properly delineated, and on its right, we see the FWI result after 20 iterations. To help the inversion, we hard coded the water bottom and a few meters below in the initial model. That is not too unrealistic, since the water bottom would be well-known after a marine survey has been performed. The salt bodies are present in the initial model but with vague boundaries because of a very strong smoothing, so the challenge for FWI is to find these boundaries. Looking at Figure 18b, it seems the salt bodies were well defined at the bottom but there are significant errors at the top.

In between the upper part with water velocity and the deepest part, we see an area with velocities at around 3000m/s that seems poorly resolved, and that section is important because that is where the reservoirs are in the original model. In reality, this poor definition is exaggerated by the color-map since when we look at the RTM obtained with this velocity model in (Figure 19b), this section seems well defined. In Figure 19a we compare the RTM obtained from a good velocity model (using the inverse crime to define an upper-quality result) with Figure 19b. The inverse crime result is obtained by using a smooth velocity model but with a very good definition of the salt boundaries (Figure 19a). We see that the resolution in the middle part is not too different. However, this test shows the main problem is the poor resolution on the top of the salt. A possible solution would be a second FWI run using the output velocity from the first run but editing the salt boundaries and fixing their values. This kind of flooding approach is common in tomography. Migration can be used to estimate the boundaries easier than FWI, but a good velocity estimation of the model above the salt is required.

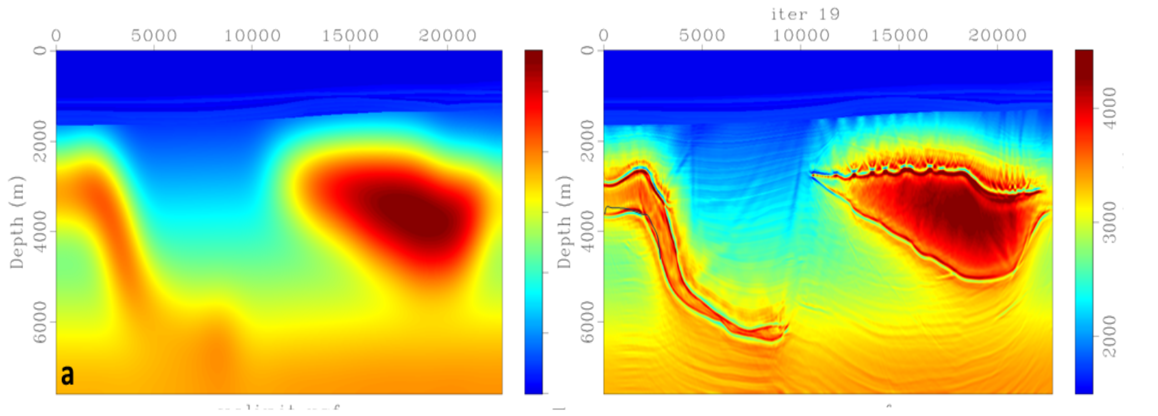


FIG. 18. Pluto salt model. Left: initial velocity. Right: FWI result after 20 iterations. The colormap emphasizes changes at the top and bottom, but the main difficulty is the top of the salt. A new iteration with an initial model with fixed salt boundaries could help to alleviate this.

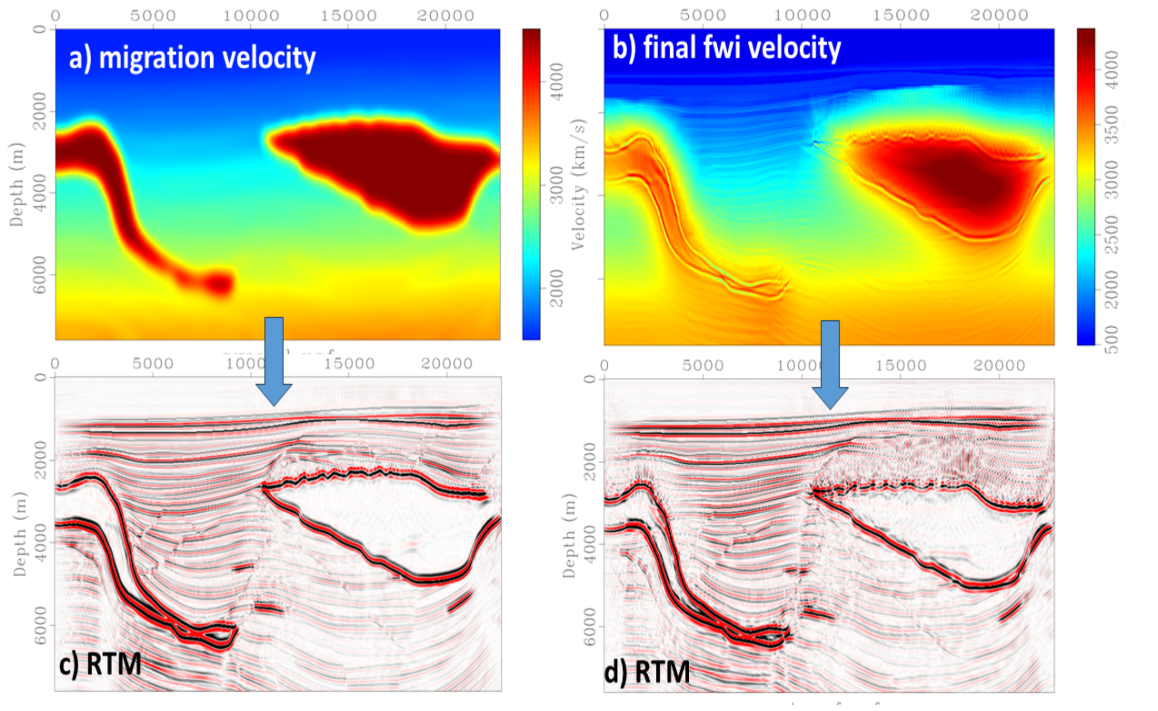


FIG. 19. Pluto salt model. Left: RTM from very good migration velocity. Right: RTM result from the FWI velocity obtained from the very poor initial velocity.

3D TESTING AND ANISOTROPY

A discussion on realistic testing is not complete without considering anisotropic three-dimensional (3D) models. We have started the effort to convert all our programs to 3D (isotropic for now). At the moment of this writing, we have achieved 3D modeling but not RTM or FWI yet. The computational cost and memory demands grow very abruptly in 3D FD algorithms. In the 2D case, we have been able to overcome the computational costs by using GPUs and low-level coding in C++ CUDA. For 3D, this approach has to be combined with Message Passage Interface (MPI) (Gabriel et al., 2004), which combined with the GPU coding increases complexity for coding, testing and, in particular, debugging effort.

Nonetheless, it is clear this has to be done and we have walked the first steps with the module PMFD3D-GPU (Sanchez-Galvis, 2023). An application of PMFD3D-GPU is demonstrated with the generation of realistic seismic synthetic data using a segment of the SEAM Foothills Phase II model (Figure 20a). This segment measures 6000 m x 2000 m x 4400 m, uniformly discretized with $\Delta h = 10$ m. The topographic elevation map of this segment is shown in (Figure 20b). Receivers were placed on the surface from 500 to 5500 m at Y=1000 m, spaced 10 m apart (Figure 20b), and an explosive source was positioned at (200,1000) m with a depth of 12 m. Applying PMFD3D-GPU for a 4-second simulation produced a multicomponent shot gather (Figure 20c), showcasing realistic seismic events like Rayleigh waves and near-surface scattering, further evident in (Figure 20d).

This example illustrates PMFD3D-GPU's realism in conducting elastic wave simulations in heterogeneous media. The module is also computationally efficient because of its C++ GPU implementation. In comparison, (Cao and Chen, 2018) who developed the original method and tested in CPU-based Matlab, reported over two days for its 3D simulations. This would impractical for multiple runs in waveform inversion problems. In contrast, the GPU-accelerated PMFD3D-GPU significantly reduces this time, completing 3D simulations in minutes. It's important to note, however, that Matlab implementations are generally not the most efficient. To provide a fair comparison, a C language version utilizing OpenMP was also developed and tested on a 12-core processor. Remarkably, the GPU-accelerated PMFD3D-GPU was found to be about 20 times faster than even this optimized CPU implementation, further highlighting its significant speed advantage.

PHYSICAL MODEL TESTING

Although realistic seismic testing would only be possible by using real seismic data, we will finish this report with a non-seismic real data test from a physical model. The Physical Model Facility in CREWES (Wong et al., 2009) can produce different data sets that can be inverted by using the same FWI codes we used in the seismic examples. However, some extra difficulties appear and depending on the layout of the experiment the data may look unfamiliar from the surface seismic point of view.

The physical data in this example was designed to simulate an MRI medical application and presented in last year's CREWES report by Keating et al. (2022). In this experiment, a plate was set horizontally with two high-density bodies representing the targets, placed on

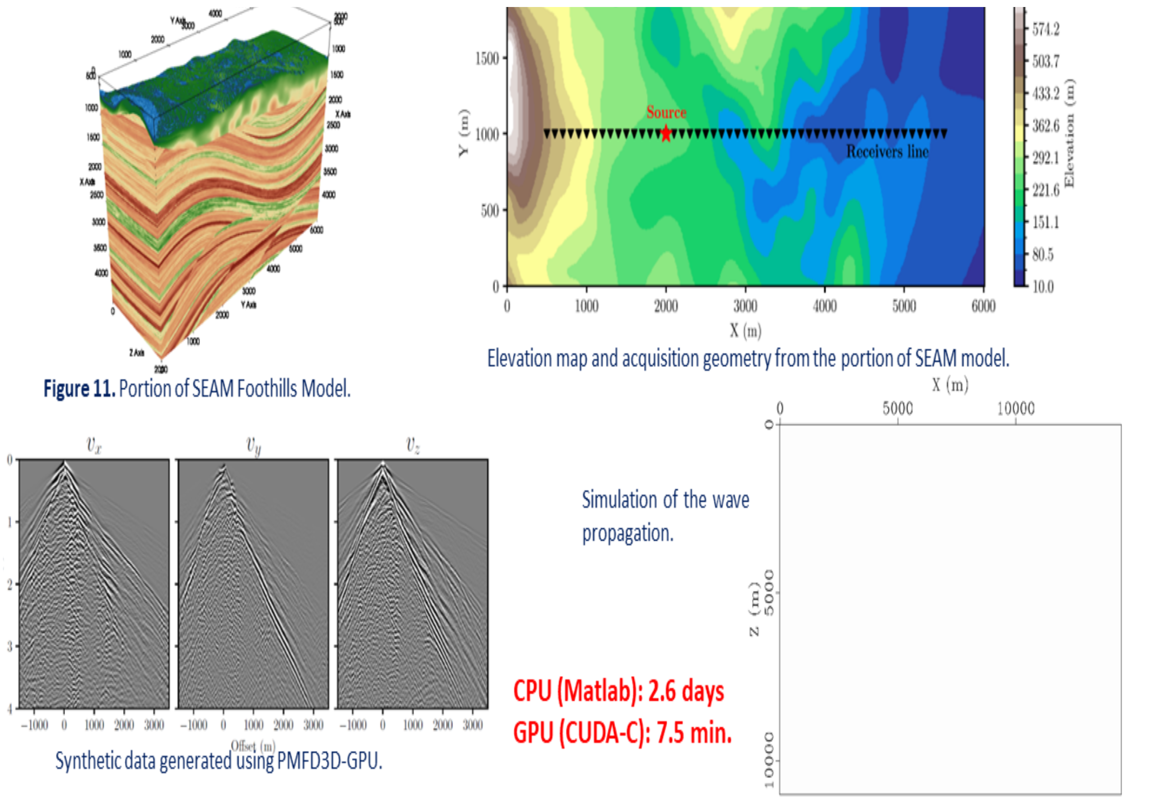


FIG. 20. Three dimensional modeling from topography

top of the plate. Two arms with shot and receiver devices (transducers) rotate on top of the plate to create seismic traces. This produces a 2D acquisition where the shots and receivers are in a circular design. In Figure 21 we see the trajectory of the shot and receiver arms in a non-concentric circular pattern. This produces shot and receiver coordinates with some periodic pattern. In Figure 21b and c we see the pattern for the shot coordinates for all the traces. Such a geometry produces shot gathers that look like shown in Figure 22a. This shot is one of 72 shots in the experiment and has been muted after the first arrival. The corresponding wavelet is shown in Figure 22b. Working with this data set directly can be confusing because our intuitions obtained from seismic data do not translate well to this setting of a horizontal plate.

To understand the problem, we first create a numerical model shown in Figure 22c and generate synthetic data honouring the real geometry. This allows us to visualize how the information is mapped into the seismic traces (Figure 22d). Although we did modeling with the real wavelet, we found the results easier to understand by using a Ricker minimum phase wavelet (Figure 22e). Running FWI in this synthetic model produces good results in just 1 or 2 iterations since we are in the inverse crime scenario (Figure 22f). This practice of running a synthetic experiment with the same geometry provides a good baseline of what to expect in the best-case scenario and also gives us an understanding of the data. For example, comparing Figures 22a and d shows that much of the information from the targets has been muted. This suggests going back to preprocessing and using the original data without mute.

However, we found it difficult to extract information from the original gathers without the mute. One reason is the large reverberation of the true wavelet with a short time range. Instead, we proceed with a different technique suggested in (Keating et al., 2022) which mimics inversion for time-lapse data. Thanks to the physical model laboratory (Wong et al., 2009), we have available a version of the dataset acquired on the plate without the targets (Figure 23b), which we can subtract from the original data in Figure 23a to produce a data set (Figure 23c) that emphasizes the contribution of the targets to the data. We can see in the RTM result in Figure 23e, and in the FWI result in Figure 23f, that the FWI algorithm can now detect the targets. Up to some degree, we can also see the hollow nature of the targets on the FWI result.

CONCLUSIONS

Testing seismic algorithms is often affected by simplifications and assumptions that eliminate key complications of real applications. In this report, we have presented new developments that can help to reduce these simplifications. We have also illustrated how to use the inverse crime scenario as the best possible result (as opposite to a baseline), to evaluate the quality of our results in several contexts: modeling, migration and FWI of data in salt environments, rough terrain (Foothills), and real data obtained from physical experiments.

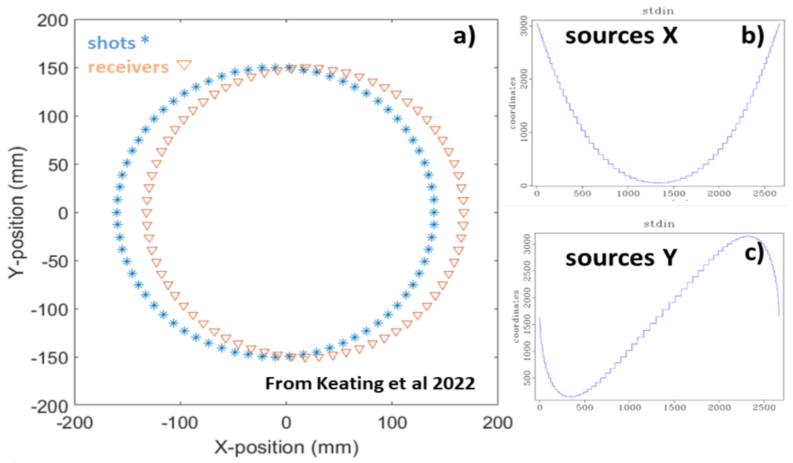


FIG. 21. Acquisition geometry for the physical model test. a) Top view for shot and receiver locations on a plate. b) Shot X coordinates for the whole data set, c) Shot Y coordinates.

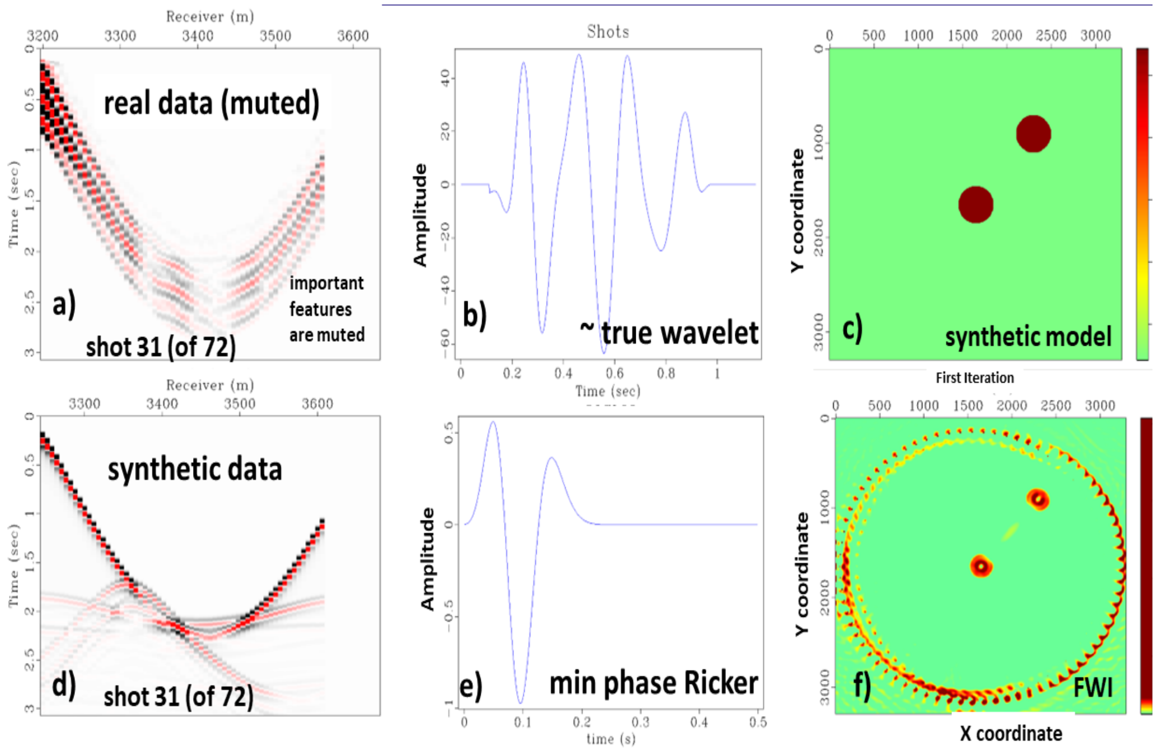


FIG. 22. Physical vs synthetic model: a) one of the acquired shot gathers after muting. b) The approximated input wavelet extracted from the data. c) A numerical model created to imitate the physical problem. d) The synthetic shot gather equivalent to "a" but modelled from the model in "c" (same geometry but no muting). e) The minimum phase wavelet used for modeling. f) The FWI result from the synthetic data set after 2 iterations (from constant initial velocity)

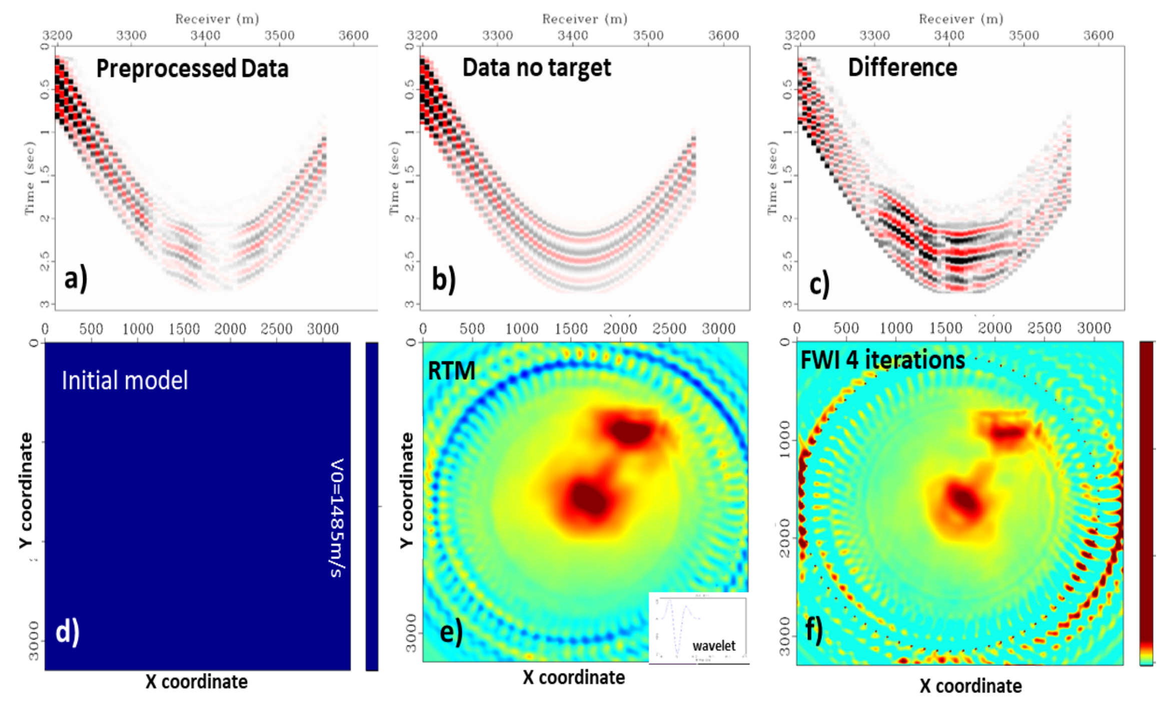


FIG. 23. Physical model test. a) one shot gather after muting. b) one shot gather acquired without targets (baseline). c) Difference between "a" and "b" to emphasize the information coming from the targets themselves (similar to a time-lapse survey). d) Initial velocity model for FWI. e) RTM result from the difference data in "c" and velocity model in "d". f) FWI result after 4 iterations again from "c" and "d".

ACKNOWLEDGMENTS

We thank Joe Wong by providing the physical model data and his help with preprocessing, Peliang Yang for RTM and FWI examples in Madagascar, Sam Gray for weekly fruitful discussions, and CREWES sponsors for their continued support over the years. This work was funded by CREWES industrial sponsors, NSERC (Natural Science and Engineering Research Council of Canada) through the grant CRDPJ 543578-19. We also gratefully acknowledge support from NSERC Discovery Grant and the Emerging Leaders in the Americas Program (ELAP) from Global Affairs Canada for funding the research internship.

REFERENCES

Cao, J., and Chen, J.-B., 2018, A parameter-modified method for implementing surface topography in elastic-wave finite-difference modeling: Geophysics, **83**, No. 6, T313–T332.

Dally, W. J., Keckler, S. W., and Kirk, D. B., 2021, Evolution of the graphics processing unit (gpu): IEEE Micro, **41**, No. 6, 42–51.

Fomel, S., and Hennenfent, G., 2007, Reproducible computational experiments using scons, *in* 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07, vol. 4, IEEE, IV–1257.

Fomel, S., Sava, P., Vlad, I., Liu, Y., Jennings, J., Browaeys, J., Bashk ardin, V., Godwin, J., Song, X., and Hennenfent, G., 2012, Madagascar software package and reproducible research.

Gabriel, E., Fagg, G. E., Bosilca, G., Angskun, T., Dongarra, J. J., Squyres, J. M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A. et al., 2004, Open mpi: Goals, concept, and design of a next generation mpi implementation, *in* European Parallel Virtual Machine/Message Passing Interface UsersâĂŹ Group Meeting, Springer, 97–104.

Keating, S., Innanen, K., and Wong, J., 2022, Cross-gradient regularization for multiparameter FWI of physically modeled data: CREWES Research Report, **34**.

Knight, S., 2005, Building software with scons: Computing in Science & Engineering, **7**, No. 1, 79–88.

Margrave, G. F., 2000, New seismic modelling facilities in matlab: CREWES Res. Rep, **12**, 1–45.

Sanchez-Galvis, I. J., 2023, Modeling and processing near-surface seismic scattering in multicomponent data: An approach based on elastic wave modeling and polarization filtering: Ph.D. thesis, School of Electrical, Electronics and Telecommunications Engineering - Universidad Industrial de Santander.

Schuster, G. T., 2017, Seismic inversion: Society of Exploration Geophysicists.

Stockwell Jr, J. W., 1999, The cwp/su: seismic unâĹŮ x package: Computers & Geosciences, **25**, No. 4, 415–419.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, No. 8, 1259–1266.

Trad, D., 2021, Computational aspects of fwi, *in* Geoconvention 2021, GeoConvention, Conference Abstracts.

Trad, D., 2022, Gpu applications for modelling, imaging, inversion and machine learning., *in* Geoconvention 2022.

Trad, D. O., 2020, A multigrid approach for time domain FWI: CREWES Research Report, **32**, 54.1–54.20.

Witte, P. A., Louboutin, M., Kukreja, N., Luporini, F., Lange, M., Gorman, G. J., and Herrmann, F. J., 2019, A large-scale framework for symbolic implementations of seismic inversion algorithms in julia: Geophysics, **84**, No. 3, F57–F71.

Wong, J., Hall, K. W., Gallant, E. V., Bertram, M. B., and Lawton, D. C., 2009, Seismic physical modeling at the university of calgary, *in* SEG Technical Program Expanded Abstracts 2009, Society of Exploration Geophysicists, 2642–2646.

Yang, P., Gao, J., and Wang, B., 2015, A graphics processing unit implementation of time-domain full-waveform inversion: Geophysics, **80**, No. 3, F31–F39.