geoconvention
Virtual Event
September 13 - 15
2021

Computational aspects of FWI

Daniel Trad –
CREWES, University of Calgary

Summary

Full Waveform inversion (FWI) is implemented in either time or frequency domains. The frequency-domain version (FD) permits the use of a multigrid approach consisting of bootstrapping from low to high frequencies where data bandwidth is gradually increased. In 3D it becomes very expensive because a large system of equations must be solved at each frequency. The time-domain (TD) approach, on the other hand, scales better for 3D and large surveys. By using parallelization in clusters with Graphics Processing Units (GPUs), the TD approach becomes very efficient. However, the TD has additional complications when using a multigrid approach. In this abstract, we will discuss a multigrid TD implementation of FWI which also helps to attenuate inverse crime effects when working with synthetic data. The dataflow proceeds by stages with different grid sizes and increasing bandwidth. In each stage, the high-frequency data are converted to the modeling bandwidth by using a matching filter and a cross-correlation shift. The filter is obtained from the estimated input wavelet and the modeling wavelet used in the finite-difference simulation. The abstract will also discuss the computational cost increase for high-resolution FWI and how it changes when GPUs are used.

Introduction

Full Waveform Inversion (FWI) (Lailly, 1993, Tarantola 1984) can provide detailed velocity information about the subsurface by fitting modeled data to observations. In practice, many difficulties need to be overcome to reach its potential. Some of these issues are related to the non-linearity of the inversion, usually manifested in the form of cycle-skipping (Bunks et al., 1995, Virieux and Operto, 2009, Warmer and Guash, 2016). Other issues are related to the difficulties of replicating the complexities of wave propagation to predict realistic data and computational cost. Frequency domain (FD) implementations (Pratt, 1999) need to solve large systems of equations with as many equations as cells in the velocity model. Therefore, for large surveys, it requires significant computing power which can be achieved by decomposing a large matrix into several computing nodes. Clusters with fast communication across nodes, low latency and fine grain parallelization are required. The FD has some advantages: 1) the non-linearity (cycle-skipping) has a clean solution by bootstrapping from low to high frequencies (Pratt et al. 1999, Sirgue and Pratt, 2004); 2) only some selected frequencies need to be inverted; 3) the Hessian can be decomposed to solve for many shots simultaneously; 4) independent frequency modeling permits to replicate dispersive energy.

From the computational point of view, a time-domain (TD) version (Tarantola, 1984) has nice scalability properties and inexpensive clusters (slow inter-node communication) are sufficient for large surveys. If the cluster as many nodes as shots, each iteration requires about the time it takes to perform 4 forward models, which is very fast with GPUs (Yang et al., 2015) or multicore systems. Therefore, TD implementations have an easier time scaling to large surveys. Parallel implementations are simpler in the TD than in the FD using Memory Passing Interface (MPI). On the other hand, the non-linearity issue of cycle skipping brings some additional complications. When working in the FD the data separation across frequency bands follows naturally by selecting

frequency slices on the frequency-space plane domain (Sirgue et al. 2004). In the TD, the separation between bands requires careful tapering and adaptation for different bandwidths, like changing the source wavelet and the cell size. These factors introduce some discrepancies between predictions and data that make convergence harder.

In the multigrid implementation used here, synthetic data were generated on a fine grid and shaping filters were applied in the different stages for amplitude matching. The inversion starts from the result from a previous stage after interpolation to a finer grid. Time-shift corrections calculated by cross-correlation improves the matching even more. The grid changes prevented the inverse crime issue of artifacts becoming data. For computational comparison, the FWI was implemented in two ways, 1) an MPI coarse parallelization with very low communication overhead and a single GPU implementation.

Theory

A high-resolution FWI leads to a high-frequency velocity model that can provide the reflectivity directly by differentiation (Kalinicheva et al, 2020). Its advantage is bypassing processing steps, like multiple attenuation and migration, but its drawback is the computational cost of inverting high frequencies that can be much higher than FWI followed by migration. Increasing frequency without a multigrid approach leads to cycle skipping (Bunks et al., 1995).

FWI implementations address this issue by solving in bandwidths with increasing frequencies which is straightforward in the FD (Pratt, 1999). In the TD, multigrid approaches have been proposed by Bunks et al. (1995) and many authors after. To avoid dispersion and instability, cell sizes and time steps must decrease with increasing frequency. However, coarse cells produce better convergence (fewer unknowns) and faster computation (fewer operations). Thus, an optimal FWI requires changing the grid for different stages. Figure 1 shows a multigrid solution starting with the initial model of Figure 1a in a coarse grid of 32x32m. Using a source wavelet with a maximum frequency of 8Hz we see the result in Figure 1b after 20 iterations using 40 shots. To move to higher frequencies, I refine the grid by interpolating this result to a 16x16m grid, injecting as an initial velocity, and inverting up to around 16Hz (Figure 1c). Finally, this model is interpolated to 8x8m cell size and used as initial velocity to go up to a maximum frequency of 32Hz (Figure 1d). The increase in resolution is noticeable from stage to stage and could continue to higher frequencies with further grid refining, although changes are less noticeable at high frequencies.

The multigrid method

The workflow presented in the previous section requires being careful on how to separate the data into bands to preserve waveform matching. The input data have frequencies too high for the coarse grids of the initial stages. A simple band-pass filter does not produce successful inversions because mismatches between the input wavelets and source wavefield.

Among several options, a relatively straightforward method is a shaping filter, which can be estimated by least-squares fitting between the filtered data wavelet and the source wavelet injected for predictions. This is the least-squares minimization of the cost function:

$$J = \left\| wavelet_{FD} - wavelet_{data} * filter \right\| \qquad (1)$$

Here we are seeking to minimize the energy of the differences between the prediction and filtered data wavelets. Notice that if we do this matching with the data instead of the wavelet, the filter

would take information away from the inversion. Instead, by estimating the filter from the source wavelets (one estimated and one chosen) the inversion sensitivity is preserved. This method seems closely related to the work of Warmer and Guash (2016). The filter that solves equation 1 is a shaping filter:

$$filter = \frac{wavelet_{input} * wavelet_{modelling}}{wavelet_{input} * wavelet_{input}}$$

In Figure 2a we see a seismic trace chosen randomly from the input dataset. This trace was obtained by finite differences with a Ricker wavelet with a dominant frequency of 16Hz. To the right, in Figure 2b we see the corresponding seismic trace predicted by the finite-difference algorithm at a dominant frequency of 4Hz and 32m cell size. A simple Butterworth bandpass filter applied to the input produces sufficient distortions (Figure 2c) to hinder convergence. Instead, the shaping filter result in Figure 2d is a better choice. Some prediction differences can still be observed, and other corrections can be used to address minor overall phase distortions. An additional phase correction can be applied by calculating the cross-correlation between the predictions and the filtered data and using the time of the cross-correlation peak to shift the traces vertically. This matching can be integrated into the dataflow.

High-resolution FWI and computation times

The computational cost for high-resolution FWI is a major obstacle. The numerical cost of finite differences and other methods depends linearly on the number of cells and time steps. For example, the cost will increase by 8 each time we double the maximum frequency in 2D or by 16 in 3D. If a result with 8Hz maximum takes one hour, it will take 64 hours by the time we reach 32 Hz. For the work presented here with the Marmousi model, the computation times follow this pattern (Table1). This was achieved by using a hybrid MPI-OPENMP parallelization (Figure 3) that gives almost perfect scalability by dividing the data across nodes. This design is optimal for this problem and has almost no overhead. The times shown in Table 1 would increase dramatically in a 3D survey and much more if using elastic modeling and generating angle gathers.

A possible solution to this issue is to break the linear dependence between the number of cells and cost by using GPUs (Wang et al., 2011, Yang et al., 2015). The cost of convolution-type techniques like finite difference decreases two orders of magnitude when using GPU's shared memory. For example, for the modeling of one shot in the Marmousi model, 4th order in space and 8x8m cell size, the times go from ~ 30 seconds (12 threads) to 0.3 seconds (in a GPU RTX2070). The GPU time is the same for 2nd, 4th or 8th order FD. Running these FWI tests with a CUDA implementation took around 20 minutes in one GPU instead of the 2 hours that took in a 10 nodes cluster. GPUs can do better than a linear dependency because their resources are under-used. Modeling in 2nd or 8th order takes the same time because more calculations just make better use of resources.

Eliminating the inverse crime problem

A common issue when working with synthetic data is to achieve over-optimistic results, called inverse crime, because of using the same engine for both data synthetics and data predictions (Schuster, 2017). This pitfall eliminates modeling errors from the residuals and transforms modeling artifacts into a source of information for inversion since their residuals are minimized with the true velocity. One way to reduce the inverse crime is to change the modeling engine. Another approach, that automatically occurs in the multigrid method, is to change the modeling

grid. This makes artifacts to be different, preventing them from becoming inversion information, turning them into modeling errors.

Conclusions

This abstract discussed an FWI implementation and dataflow capable of achieving High-Resolution FWI by a multigrid approach. The method requires shaping filters to accommodate the waveform of the data to the waveforms predicted on each band. Tests using cluster parallelization show how the computational cost increases when moving to higher frequencies but also the advantage of using GPUs for the forward modeling. As a side effect of the multigrid design, inverse crime problems related to synthetic data are largely removed.

Acknowledgements

My sincere thanks to Penliang Yang for his Madagascar examples and Sam Gray for insightful discussions. This work was funded by the industrial sponsors of the Consortium for Research in Elastic Wave Exploration Seismology (CREWES) and by the NSERC grants CRDPJ 461179-13 and CRDPJ 543578-19. We gratefully acknowledge continued support.

References

Bunks C. Saleck F. M. Zaleski S. Chavent G., 1995, Multiscale seismic waveform inversion: Geophysics, 60, 1457–1473, doi: http://dx.doi.org/10.1190/1.1443880. [PubMed] 0016-8033

Kalinicheva T,, M. Warner, and F. Mancini, (2020), "Full-bandwidth FWI," SEG Technical Program Expanded Abstracts: 651-655, ttps://doi.org/10.1190/segam2020-3425522.1

Lailly, P., 1983,The seismic inverse problem as a sequence of before stack migrations: Conference on Inverse Scattering, Theory and Application, Society for Industrial and Applied Mathematics, Expanded Abstracts, 206–220.

Yang P., Gao J. and Wang B.; A graphics processing unit implementation of time-domain full-waveform inversion. Geophysics 2015;; 80 (3): F31–F39. doi: https://doi.org/10.1190/geo2014-0283.1

Pratt, R. G., 1999, Seismic waveform inversion in the frequency domain, Part 1: Theory and verification in a physical scale model," GEOPHYSICS 64: 888-901. https://doi.org/10.1190/1.1444597

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

Sirgue, L. and Pratt R. G., 2004, Efficient waveform inversion and imaging: A strategy for selecting temporal frequencies: Geophysics, 69, 231–248, doi: https://doi.org/10.1190/1.1649391. [PubMed] 0016-8033

Tarantola A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, 49, 1259–1266, doi: https://doi.org/10.1190/1.1441754. [PubMed] 0016-8033

Virieux J. and Operto S., 2009, An overview of full-waveform inversion in exploration geophysics: Geophysics, 74, no. 6, WCC1–WCC26, doi: http://dx.doi.org/10.1190/1.3238367. [PubMed] 0016-8033

Wang B. Gao J. Zhang H. Zhao W., 2011, CUDA-based acceleration of full waveform inversion on GPU: 81st Annual International Meeting, SEG, Expanded Abstracts, 2528–2533.

Warner M. and Guasch L., 2016, Adaptive waveform inversion: Theory: Geophysics , 81, no. 6, R429–R445, doi: http://dx.doi.org/10.1190/geo2015-0387.1. [PubMed] 0016-8033

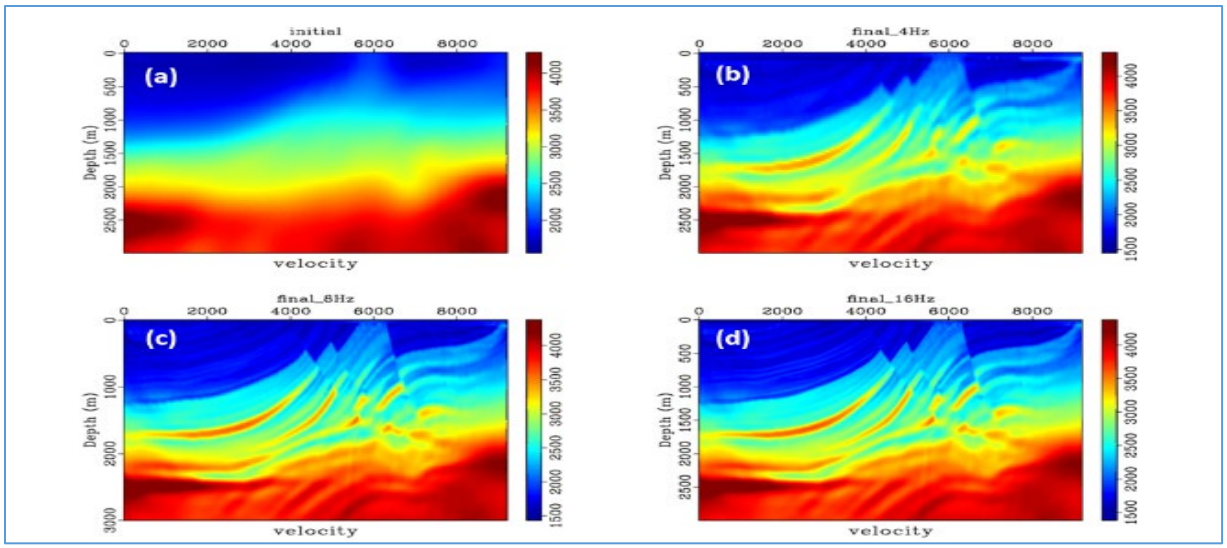

*Figure1: Multigrid FWI result by going through 3 stages. a) Initial model, b) 4Hz dominant frequency in a 32x32m grid, c) 8Hz in 16x16m, d) 16Hz in 8x8m. Maximum frequencies are around double than the dominant frequencies.*Type equation here.

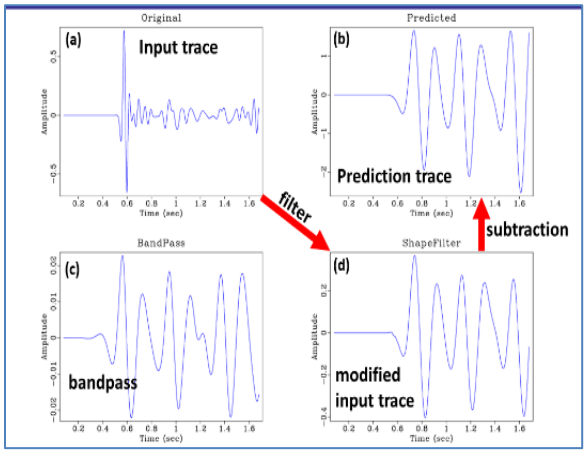

Figure 2: a) Original trace, b) modeled FD trace, c) band-pass filtered input trace and d) input trace after matching filter calculated from original to modeling wavelet.

Table 1. Computation times

model size	cell size	time steps	nshots	iterations	time
96 x 288	32, 32	2800	40	20	196 secs
188 x 576	16, 16	2800	40	20	576 secs
376 x 1151	8, 8	4600	40	20	4481 secs

Table 1: – Computing time for examples shown in a 10-node cluster. As the frequency increases, cell number and time steps do as well by an 8 factor for 2D and 16 for 3D.

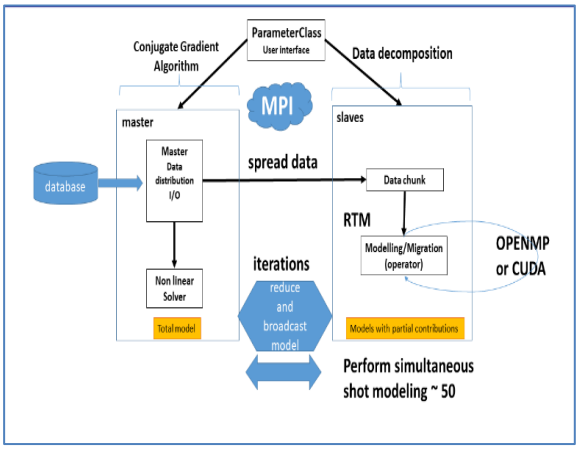

Figure 3: Hybrid MPI-OPENMP-CUDA data flow. By splitting the data space across surveys instead of the model, the number of required communications across nodes decreases to 3 per iteration instead of the number of time steps as in model decomposition.