Compression for effective memory bandwidth use in forward modeling

Eileen Martin

ABSTRACT

A common bottleneck in seismic imaging is moving data. Lossy compression during wave propagation simulations may be a useful tool for decreasing the amount of data that must be moved. In the future, hardware compression may be added between main memory and cache memory, so I explore the application of the fpzip algorithm to compress small sets of data at each time step of acoustic wave propagation. To make the most of modern architectures, I investigate the use of wavelet compression of entire wave fields before writing to disk. The experiments presented show limited promise for using fpzip in seismic imaging and potential for using wavelet and curvelet compression when writing to disk.

INTRODUCTION

Modern computing is limited by the speed of moving data in computers. Modeling wave propagation is a fundamental data processing component. It suffers from this limitation despite being relatively computationally intensive, both between different levels of cache and main memory, and from main memory to disk. One strategy for dealing with the data movement during forward modeling is to reduce the amount of data that is moved. Compression during forward modeling could be a viable option for seismic imaging if the following conditions are met: 1) compression and decompression can be done quickly, 2) the amount of data being moved is significantly reduced, and 3) the resulting image is not too corrupted.

Why would forward modeling in seismic imaging be a particularly good candidate for lossy compression? Much of the early work in data compression was motivated by a desire to compress seismic data. Scientists developed methods to compress raw seismic data so that it is very sparse in some wavelet and wavelet-like bases (Mallat, 2008); (Bosman and Reiter, 1993); (J.D. Villasenor and Donoho, 1996). Additionally, we may be able to tolerate more errors in forward modeling because we care little about that wave field. Rather, we care about the end result of a seismic imaging problem which is fundamentally ill-posed. If we use full waveform inversion (FWI) we are not guaranteed to calculate a solution close to the true solution even with a perfect forward modeling operator, so we might be able to allow some information about an intermediate step in the process to be lost to compression. In the context of reverse time migration (RTM), the image is the result of many sums of correlations,

so some errors in the source and receiver wave fields can be tolerated as long as they are incoherent. If lossy compression allows us to speed up our calculations enough, we could use the same computer resources to process larger data sets or further explore the probability distribution of images given our current data set.

In this report, I present two ways of using lossy compression to help alleviate the computational bottleneck often seen when simulating wave propagation due to our memory bandwidth limitations of our machines. First, I investigate the possibility of hardware compression between DRAM and cache using the fpzip algorithm on small chunks of data (Lindstrom and Isenberg, 2010). Then, I investigate the use of compressing a full wave field with a wavelet decomposition (R. Wu and Geng, 2008) prior to writing to disk with the aim of performing these operations on graphics processing units (GPUs). I look at this strategy for an individual compressed image that would be used as the input to a reverse time migration, and as a series of compressions that are used for restarting the forward modeling. For both strategies, I present reproducible computer experiments to test the robustness of finite-difference time domain (FDTD) two-way two-dimensional acoustic isotropic wave propagation to the proposed compression strategies.

COMPRESSION BETWEEN DRAM AND CACHE

Several partial differential equations (PDE) simulations have been shown to withstand up to a factor of 3-5x lossy compression as chunks of data the size of a cache line are moved between main memory and cache in D. Laney and Wegener (2013). Two of these codes are fluid dynamics, and one is a multiphysics plasma-laser interaction model. The multiphysics code includes an electromagnetic wave propagation package, but it does not appear that the effect of compression on the wave propagation was studied in isolation.

Because it is user-friendly and easy to obtain, I chose to use the fpzip algorithm and code for compression and decompression (Lindstrom and Isenberg, 2010). The results from D. Laney and Wegener (2013) suggest that fpzip might be expected to achieve a factor of two compression (the 3-5x ratios were using APAX, which is produced by a now out-of-business company). The assumption behind fpzip is that the repetitive data are near each other, and it uses the Lorenz predictor to compress data based on this assumption (Lindstrom and Isenberg, 2006). There does not seem to be any prior published work on the use of lossy compression at the scale of cache lines in acoustic wave propagation. I used Algorithm 1 to compress and decompress the wave field at each time step.

In this algorithm I assume that the finite difference scheme is second order in time, but it could be higher order. For the numerical experiments presented later, I

Algorithm 1 Software compression of wave field between main memory and cache

Initial conditions: $u(0,\mathbf{x}) = u_0(\mathbf{x})$, $\partial_t u(0,\mathbf{x}) = 0$
Compress initial wave field
for $i = 1, \ldots, nt$ **do**
  Uncompress wave fields $u_c(t_{i-1},\mathbf{x})$ and $u_c(t_{i-2},\mathbf{x})$ with `fpzip_file_read`
  Update $u_u(t_i,\mathbf{x}) = L\left(u_c(t_{i-1},\mathbf{x}), u_c(t_{i-2},\mathbf{x})\right)$
  Compress current wave field $u_u(t_i,\mathbf{x})$ with `fpzip_file_write`
end for

use the isotropic acoustic second order in time and space wave operator:

$$\begin{aligned} u_u(t_i,\mathbf{x}_{j,k}) &= \left(\frac{v(\mathbf{x}_{j,k})dt}{dx}\right)^2 (u_c(t_{i-1},\mathbf{x}_{j-1,k}) + u_c(t_{i-1},\mathbf{x}_{j+1,k}) - 4u_c(t_{i-1},\mathbf{x}_{j,k}) \\ &+ u_c(t_{i-1},\mathbf{x}_{j,k-1}) + u_c(t_{i-1},\mathbf{x}_{j,k+1})) + 2u_c(t_{i-1},\mathbf{x}_{j,k}) - u_c(t_{i-2},\mathbf{x}_{j,k}) \quad (1) \end{aligned}$$

where dt is the time step, dx is the spatial grid size, $v(\mathbf{x})$ is the velocity model, u_c is the wave field after compression, and u_u is the wave field before compression. This compression scheme could be extended to other wave operators that are more physically realistic.

WAVELET COMPRESSION OF FULL WAVE FIELDS

Data movement between cache and DRAM takes as long as a couple hundred cycles, but it takes even longer to write to disk. Some algorithms for reverse time migration (RTM) require writing to disk once every four time steps in a forward model. Other algorithms may require writing to disk less frequently for checkpoints that could potentially be used for restarting the computation if it is interrupted.

Wavelet compression was a strategy used in the nineties for storing and transmitting large volumes of seismic data from the field to computers at offices. For two-dimensional wavelets, it was possible to get acceptable compression by up to a factor of approximately 20. However, the real savings were found for three dimensional wavelets, which could accept compression by a factor of 100 while maintaining a close approximation to the original data (J.D. Villasenor and Donoho, 1996). It is important that the reduction in memory bandwidth is more significant than the increase in computation, so this strategy may be most effective on architectures which can perform fast compressions. GPUs have been shown to perform significantly faster than traditional processors for computing wavelet compression and decompression(T. Wong and Wang, 2007). This suggests that multidimensional wavelet compression of a wave field (perhaps on the CPU while the GPU continues propagation calculations) before each disk write may be useful for reducing the amount of data actually written to disk.

The data compressed in previous studies (J.D. Villasenor and Donoho, 1996) was only the recorded wave field over many samples at the surface sensor locations. We

need to compress an entire wave field from just one or two time steps for each disk write, which may require a different wavelet basis or produce different acceptable compression ratios. Here I present results for two-dimensional acoustic modeling as a first step towards compression for three-dimensional modeling. Wavelet compression of individual photographs has been explored on GPUs (T. Wong and Wang, 2007), but our report investigates compression both in the context of individually compressed images that are used as inputs to RTM, and in the context of restarting a wave propagation simulation.

One way to deal with computational failures is restarting a simulation from checkpoints. Hopefully errors would not happen very often, and we would rely on these checkpoints much less frequently than we actually are writing to disk. Is it possible to rely on only compressed wave fields for checkpointing, or do the errors grow too much? Algorithm 2 describes one possible compression scheme in this context where the user would set some threshold of error or compression ratios.

Algorithm 2 Software compression of wave field for checkpoint/restart from disk

Initial conditions: $u(0, \mathbf{x}) = u_0(\mathbf{x})$, $\partial_t u(0, \mathbf{x}) = 0$
Compress initial wave field into wavelet basis with threshold
Write to disk
f is number of time steps between writing to disk
for $i = 1, \ldots, nt$ **do**
  if restart is needed **then**
    Go back to time step $i - (i \mod f) + 1$
    Uncompress wave fields $u_c(t_{i-1}, \mathbf{x})$ and $u_c(t_{i-2}, \mathbf{x})$ from wavelet basis
  end if
  Update $u_u(t_i, \mathbf{x}) = L\left(u_c(t_{i-1}, \mathbf{x}), u_c(t_{i-2}, \mathbf{x})\right)$
  if $i \mod f == 0$ **then**
    Compress $u_u(t_i, \mathbf{x}), u_u(t_{i-1}, \mathbf{x})$ in wavelet basis
    Write to disk
  end if
end for

As before, L in Algorithm 2 could be any wave operator, and the dependence on two time steps is just for the assumption that it is a second order in time scheme. This algorithm could be adapted to leave out the 'if restart is needed' step, and the wave fields written to disk could be used for RTM or another imaging procedure. In the following experiments we use a second order in space, second order in time FDTD isotropic acoustic two-dimensional forward model. In this study, we only investigate how robust this forward modeling operator is to compression.

HOW TO MEASURE ERROR

For these toy experiments, I measured errors by also calculating the wave field with the same wave operator but without any compression. This is only done for small

problems to build an idea of good compression ratios and types of compression; it would never be done in practice. I chose to measure the average relative error per time step as measured by the Frobenius norm of pointwise differences

$$\frac{1}{nt}\sum_{i=1}^{nt}\frac{\|u(t_i,\cdot)-u_u(t_i,\cdot)\|_2^2}{\|u(t_i,\cdot)\|_2^2} \tag{2}$$

where u is the comparison wave field that is never compressed, and u_u is the wave field that has undergone compression. A phase shift might appear as a large error when measured this way, but it may not be much of an issue if it does not result in different structures in the final image. A better way to measure errors would show if new shapes are introduced, as can be done with prediction error filters (Clapp, 2008). In fact, an ideal measure of error for test problems would look at the resulting image to detect any changes rather than the wave field.

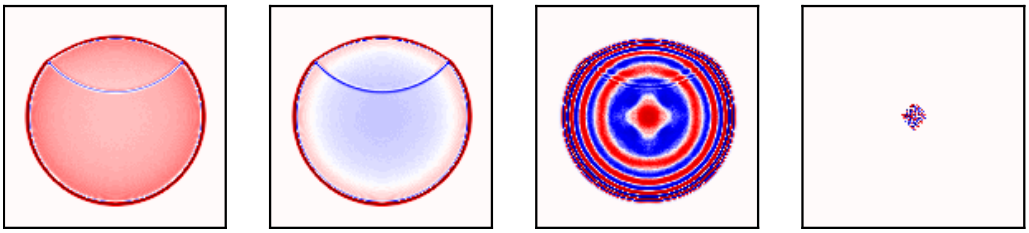
EXPERIMENTS WITH FPZIP

To test what compression ratios could be achieved with fpzip, I ran an example for which I already had a good idea what the wave field should look like: a two layer velocity model. I studied a 16 km by 16 km region discretized by 16 m by 16 m grid blocks with a top layer that was 4 km deep and has velocity 4 km/s, and a 12 km thick bottom layer with wave speed of 3 km/s. To satisfy the CFL condition, I used a time step of 0.002 seconds, and observed the domain for 1.6 seconds (800 time steps). To keep the code simple, the region has reflecting boundary conditions (wave field set to zero), and a single source in the middle (8 km, 8 km) which was a Ricker wavelet with peak frequency of 10 Hz:

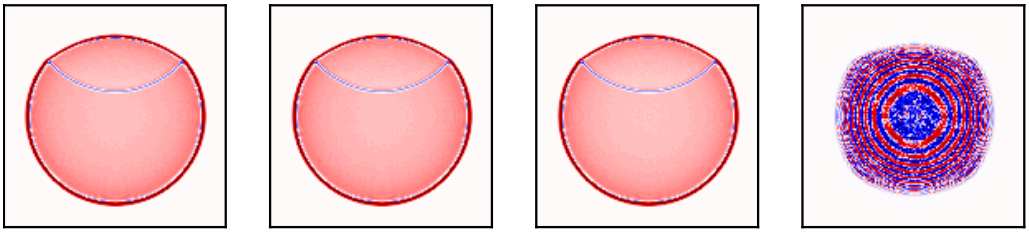
$$s(t_i) = (1-2(10\pi(10-i)dt)^2)e^{-(10\pi(10-i)dt)^2} \tag{3}$$

I did these calculations in both single and double precision, with four levels of compression (including lossless), as seen in Figures 1(a) and 1(b).

Although it is possible to visually inspect these images and see that some are nearly identical, it is preferable to quantify these differences. I calculated the average relative Frobenius norm error (Figure 2). As seen in both the error plot and the wave fields, a factor of 4/3 compression is acceptable, as is a factor of two for double precision. A factor of two compression for the single precision calculation seems to maintain the information about the discontinuity in the velocity model, but it gives the false appearance of dispersion. This false dispersion results in a relatively large pointwise error that is even worse than the factor of four compression. The factor of four compression retains a little information about the double precision, but much less than the factor of two compression of single precision. Why? The difference occurs because the single precision allows more points to be included in the Lorenz predictor. A factor of four compression of single precision loses all useful information.



(a)



(b)

Figure 1: Wave field at 1.6 seconds produced by a centrally located source in the two layer velocity model for both single (top) and double (bottom) precision computations with compression ratios (left to right): 1x (lossless), 4/3x (24 bit single, 48 bit double), 2x (16 bit single, 32 bit double), 4 (8 bit single, 16 bit double). [**ER**]

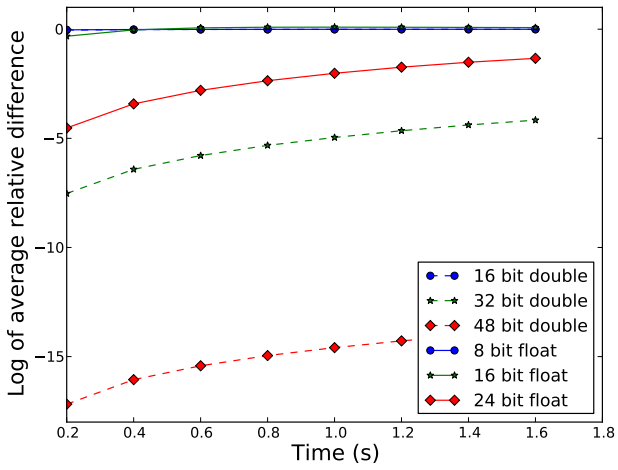
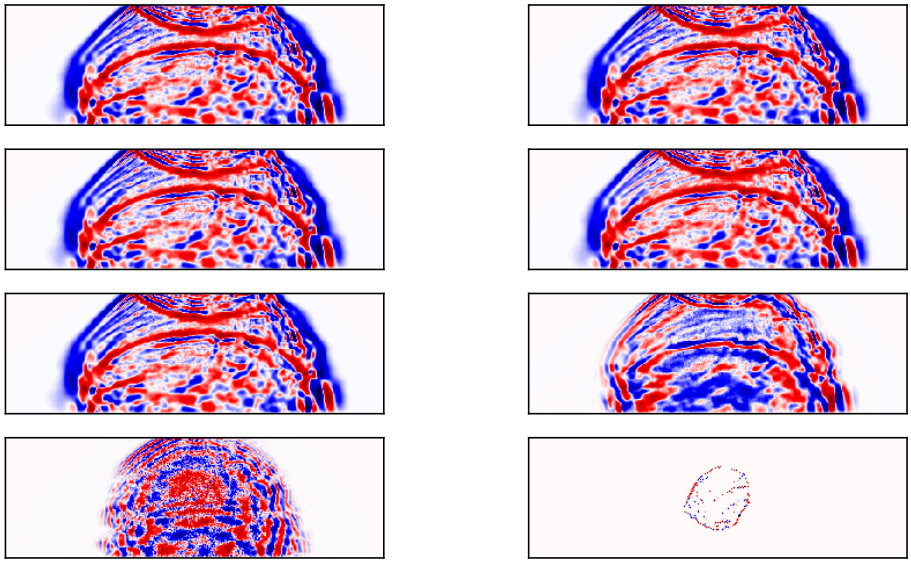


Figure 2: Lossy compression with fpzip on double precision and single precision was tested. Lossless compression (0 error) was 32 bits for single precision and 64 bits for double precision. The average relative error per time step as sampled every 100 time steps in the Frobenius norm is plotted. [**ER**]

Although it is not realistic, a two layer velocity model is nice because it is easy to predict what the wave field should look like. To look at the result of compression on a more realistic synthetic example, I turned to the Marmousi model(Benamou, 1996). The model I used was 9192 m by 2904 m divided into 24 m by 24 m blocks. I used the same source as for the two layer model, and I again placed it in the center of the model with reflective boundaries on all sides. I recorded this wave field everywhere for 500 time steps, which were 0.00218 s long to respect the CFL condition. The resulting wave fields at just over a second after the start can be seen in Figures 3(a) and 3(b). The error results were nearly identical to those of Figure 2.



(a) (b)

Figure 3: (Left) Double precision and (right) single precision at 500 time steps from source in center of Marmousi model for (top to bottom) compression by factor of 1, 4/3, 2, and 4. [**ER**]

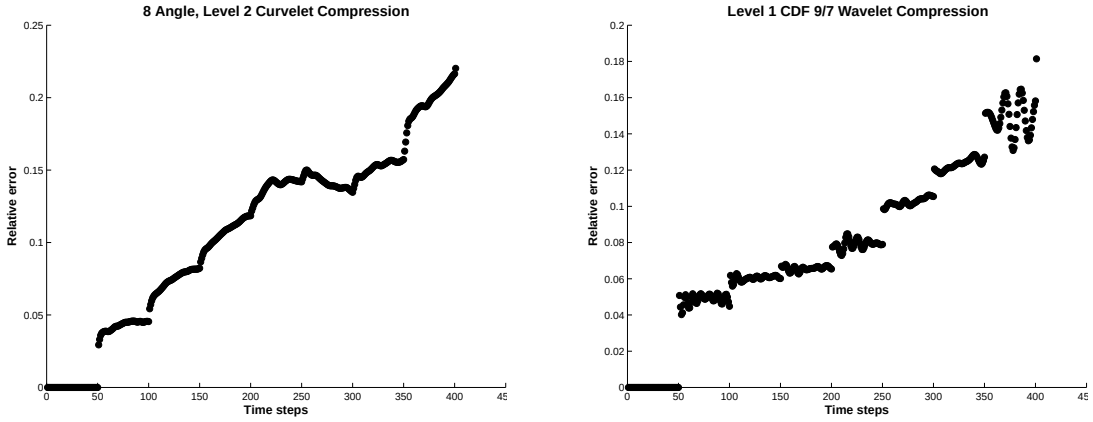
A factor of two compression for double precision is nearly identical to lossless compression. For single precision, the major features of the wave field look similar, but it is unclear what the result of this will be on the final image. A factor of four compression for double precision does not contain enough information for a good image. A factor of four compression contains no useful information for single precision.

EXPERIMENTS WITH WAVELET AND CURVELET COMPRESSION

The high compression ratios (around 100) achieved in J.D. Villasenor and Donoho (1996) required using the 9/7 biorthogonal CDF wavelet filter, which is available in the toolbox, but called the biorthogonal 4.4 wavelet. For two dimensions, a ratio of at least 25 could be achieved, although it is unclear what wavelet set was used to

achieve these ratios (Bosman and Reiter, 1993). To quickly get a wavelet compression for the purpose of evaluating robustness of forward modeling to compression, I used the Matlab wavelet toolbox. The other approach I used was compression with the CurveLab Matlab library (E. Candès, 2007). For these experiments, I again used the Marmousi model from Benamou (1996) with the same source, reflective boundaries on all sides, and 400 times steps that are each 0.00218 s long.

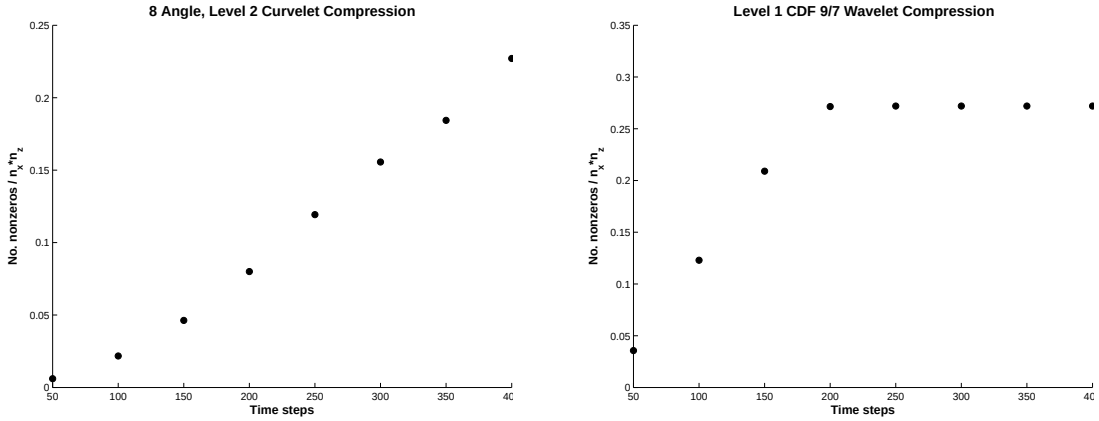
In the first experiment, I simulated compression when writing to disk for the purposes of checkpointing and restarting to see whether the CDF wavelet or curvelets were better suited to the purpose of restarting computation. For both wavelets and curvelets, errors introduced by compression will build on each other and grow, so the question is whether this growth is more controlled by one type of compression. I used a level two decomposition with the CDF 9/7 wavelet, with a hard global threshold set at ten. For the curvelet compression I used eight angles and for each angle, kept any nonzero coefficients at least one fiftieth the size of the maximum nonzero coefficient. To quickly see the effects of restarting from a compressed wave field, I compressed the wave field every fifty time steps, then used the compressed field to continue the forward modeling.The errors and compression ratios can be seen in figures 4(a), 4(b), and 5(a), 5(b).



(a) (b)

Figure 4: Differences between the wave field with and without compression grow as time goes on when compressed wave fields are used for restarting wave propagations every fifty time steps. A simple measure of this is relative l_2 error for (left) a two level, eight angle curvelet compression keeping coefficients at least one fifteenth the maximum coefficient, and (right) a one level CDF 9/7 compression with a hard threshold at ten. [**CR**], [**ER**]

The results from Figures 4(a), 4(b), 5(a) and 5(b) show that errors due to compression in both the curvelet and wavelet domains grow as compression is applied to previously compressed wavefields. The level of compression begins at a similar level but as the computation progresses, the sparsity level in the wavelet domain appears to be more controlled than in the curvelet domain. The growth of errors in



(a) (b)

Figure 5: When using compressed wave fields for restarting wave propagation calculations every fifty time steps, the ratios of nonzeros kept in compressed version to size of the spatial domain increase. This happens both for (left) level two eight angle curvelet that retains coefficients at least one fifteenth the maximum coefficient, and (right) a one level CDF 9/7 compression with a hard threshold at ten. [**CR**], [**ER**]

both domains is overall about the same size, but the growth is more continuous in the curvelet domain, while in the wavelet domain most of the error appears to occur right at the compression step. However, it is difficult to draw conclusions on this measure of error, because visual inspection shows that smaller amplitude events are picked up much better by the curvelet domain than the wavelet domain.

The final experiment I conducted was simply compressing an individual wave field to get an idea of how much compression might be acceptable in both the wavelet and curvelet domains when those compressed wave fields are only going to be used as an input to RTM. Figure 6 shows the 400th time step of a second order FDTD propagation of a 10 Hz peak Ricker wavelet at the center of the the Marmousi domain with reflective boundaries and two compressed versions.

The results that led to Figure 6 also show that in the curvelet domain with eight angles, a compression that keeps only coefficients at least one fifteenth the size of the maximum coefficient (i.e. fourteen percent as many nonzeros as $n_x n_z$) gives acceptable results with $\|u_c(400 * 0.00218, x) - u(400 * 0.00218, x)\|_2 / \|u(0.00218, x)\|_2 = 13\%$. A two level CDF 9/7 wavelet decomposition with hard global threshold ten can achieve a reasonable result with $\|u_c(400 * 0.00218, x) - u(400 * 0.00218, x)\|_2 / \|u(0.00218, x)\|_2 = 35\%$ by using just 8% as many nonzero coefficients as $n_x n_z$.

This is not quite as good a compression ratio as was achieved for two-dimensional wavelet compression in Bosman and Reiter (1993), but the spatial relation of the data being compressed was different. Also note that the choice of wavelet filter was motivated by three dimensional data compression results rather than the two-

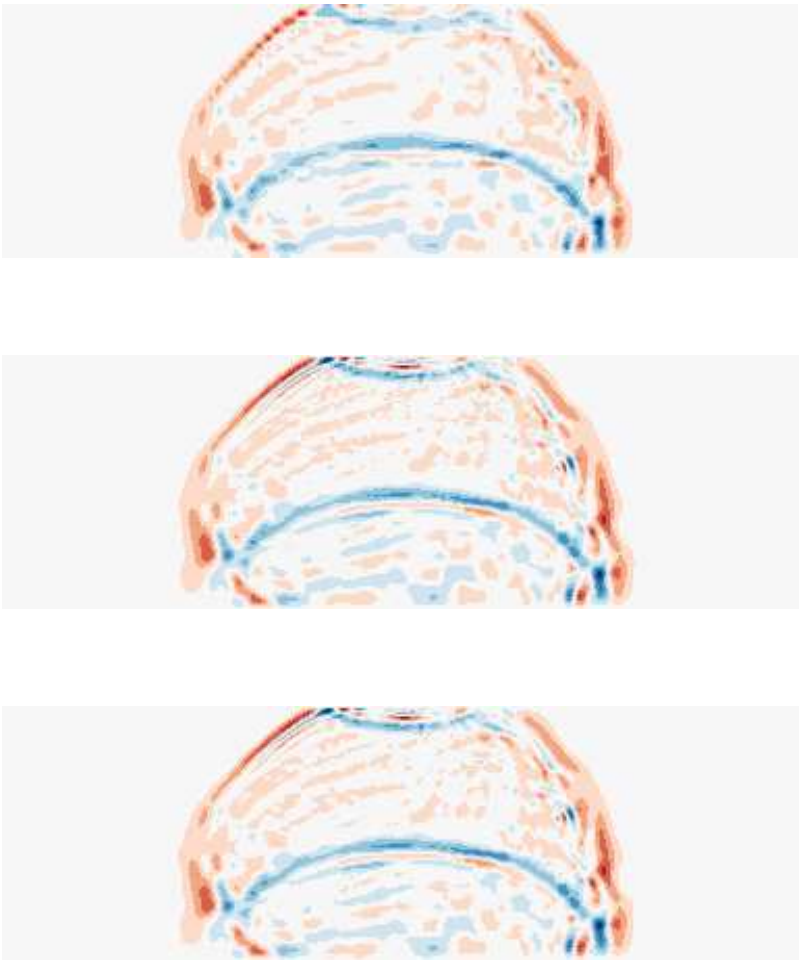


Figure 6: (Bottom) compressed with two level eight angle curvelet and only retaining coefficients at least one fifteenth the max coefficient size, (middle) uncompressed, (top) compressed with CDF 9/7 two level wavelets with hard threshold set to ten. [**CR**]

dimensional results since the aim is to have a method that will eventually make large three dimensional problems more manageable. However, these errors propagate in a nonlinear way to the final image, so this forward modeling procedure will need to be put into a more complete imaging workflow to understand the errors introduced in the image due to lossy compression in the middle of wave (forward and adjoint) propagation.

DISCUSSION

These experiments show that compression with fpzip has limited success, yielding reasonable results while requiring as little as half the bandwidth between DRAM and cache memory. Experiments for wavelet compression of full wave fields when writing to disk show that the data could be reduced by a factor of around ten without significant impact on the wave field. For both strategies, more informative measures of error should be used. The impact on the imaging process as a whole also needs to be evaluated by substituting either Algorithm 1 or Algorithm 2 for the forward and adjoint modeling, and by using the output of isolated compressions as an input to RTM. Because of the limitations of small scale compression and the fact that hardware compression between DRAM and cache has yet to be implemented, future work will likely focus on full wave field compression on GPUs when writing to disk. This includes a study of compression of three-dimensional wave fields, which we expect to accomodate higher compression ratios, with more realistic finite difference operators and moving the wavelet compression and decompression to GPUs to get timing results.

ACKNOWLEDGEMENTS

I would like to thank Bob Clapp for helpful conversations about compression, and in particular for suggesting that I look at wavelet compression on GPUs when writing to disk. I would also like to thank Steve Langer and Peter Lindstrom for helpful conversations about fpzip and lossy compression in PDE solvers. This work was supported by the Department of Energy Computational Science Graduate Fellowship, provided under grant number DE-FG02-97ER25308.

REFERENCES

Benamou, J., 1996, Computation of multi-valued traveltimes in the marmousi model: Technical report.

Bosman, C. and E. Reiter, 1993, Seismic data compression using wavelet transforms: Expanded Abstracts of the 63rd Ann. Internat. Mtg., 1261–1264, Soc. Expl. Geophys.

Clapp, R., 2008, Prediction error filters to enhance differences: SEP Report, **131**.

D. Laney, S. Langer, C. W. P. L. and A. Wegener, 2013, Assessing the effects of data compression in simulations using physically motivated metrics: Presented at the SC'13, Supercomputing.

E. Candès, L. Demanet, D. D. L. Y., 2007, Curvelab software.

J.D. Villasenor, R. E. and P. Donoho, 1996, Seismic data compression using high-dimensional wavelet transforms: , 396–405, IEEE.

Lindstrom, P. and M. Isenberg, 2006, Fast and efficient compression of floating-point data: IEEE Trans Vis Comput Graph, **12**, 1245–1250.

———, 2010, fpzip software.

Mallat, S., 2008, A wavelet tour of signal processing, 3 ed.: Academic Press.

R. Wu, B. W. and Y. Geng, 2008, Seismic wave propagation and imaging using time-space wavelets: Expanded Abstracts of the 78th Ann. Internat. Mtg., 2983–2987, Soc. Expl. Geophys.

T. Wong, C. Leung, P. H. and J. Wang, 2007, Discrete wavelet transform on consumer-level graphics hardware: IEEE Transactions on Multimedia, **9**, 668–673.