



Effect of lossy ZFP compression on Least-squares migration convergence

Átila Saraiva Quintela and Mauricio D. Sacchi

Signal Analysis and Imaging Group (SAIG), Department of Physics, University of Alberta

Summary

The least-squares reverse time migration (LSRTM) is a powerful tool for imaging complex geologic structures while compensating for the lack of extensive offset data. However, the gradient calculation done at each of its iterations is extremely costly because it requires the storage of the entire background wavefield, which for large 3D models can be in the order of terabytes for each shot. This puts in perspective that even sophisticated techniques such as optimal checkpointing, effective boundary conditions, and decimated wavefield reconstruction can be prohibited, given the sheer size of large 3D models. Hence, the more straightforward solution of saving the whole background wavefield to the disk gains renewed significance. However, this raises many issues, such as the lifecycle of storage devices, performance, and available space. A solution to trade a computation to increase the storage footprint, both in space usage and lifetime, is to use compression. However, lossless compression on floating-point data fails to achieve higher space savings on scientific simulation data, highlighting the necessity of lossy compression. This work aims to better understand, through experimentation, the effect on LSM convergence, performance and image quality for different levels of lossy compression, more explicitly using the ZFP compression algorithm.

Introduction

The least-squares reverse time migration (LSRTM) efficiency is challenged by the necessity to store the forward background wavefield for the adjoint state method's gradient computation during reverse time propagation, a problem common to inversion/optimization using adjoint methods and time-dependent PDE solvers (Cardesa et al., 2020; Hascoet and Pascual, 2013; Dussaud et al., 2008). Solutions like checkpointing (Griewank, 1992; Symes, 2007), decimated wavefield reconstruction (Yang et al., 2016), and boundary techniques (Dussaud et al., 2008; Yang et al., 2014) partially address the data storage issue but still require significant storage and computational resources, particularly for complex 3D models.

A practical yet storage-intensive solution involves saving the entire background wavefield to disk, leveraging modern NVMe drives' high throughput. Nevertheless, this approach risks reducing storage device lifespans due to intensive read-write cycles during LSRTM iterations. Compression, especially lossy algorithms like SZ (Di and Cappello, 2016) and ZFP (Lindstrom, 2014), emerges as a viable alternative to balance computation and storage efficiency despite the potential impact on simulation fidelity.

This study explores the impact of lossy compression, specifically using ZFP to store the background wavefield in the LSRTM iterations, addressing a gap in the literature on compression's effects on convergence and image quality for least-square migration. It builds on existing research on compression in forward wave inversion (FWI) and reverse time migration (RTM) (Kukreja et al., 2022; Huang et al., 2023), aiming to contribute insights into the trade-offs involved in employing lossy data compression in high-performance computing applications for LSRTM.

For clarity, vectors are denoted in bold lowercase, linear operators in bold uppercase, mappings and

other operators in uppercase, and constants in Greek letters.

Least-squares reverse time migration

The problem of least-squares migration is quite old, dating back to the 80s, with the basic mathematical framework being developed by Tarantola (1984). It is essentially equivalent to linearized seismic inversion. For the acoustic case, instead of using the second-order acoustic wave equation:

$$\frac{1}{c(\mathbf{x})^2} \frac{\partial^2 p(\mathbf{x}, t)}{\partial t^2} - \nabla^2 p(\mathbf{x}, t) = f_s(\mathbf{x}, t), \quad (1)$$

where c is the velocity model, p is the pressure wavefield, and f_s is the source wavelet, one uses its linearized version. However, a more rigorous treatment of the problem requires the usage of the adjoint state method applied to the augmented lagrangian of the cost function. We will omit the calculation details, but they used the method described in Plessix (2006).

In the code implemented in this work, we calculated the forward and adjoint states, along with the gradient based on the L2 norm cost function:

$$J(\mathbf{m}) = \frac{1}{2} \sum_{s,r} \int_0^T (S_{s,r} p_s(\mathbf{x}, t) - d_{s,r}(\mathbf{x}, t))^2 dt, \quad (2)$$

where $S_{s,r}$ is the source-receiver sampling operator, p_s is the forward state variable, m is the reflectivity model, and $d_{s,r}$ is the data. We used the following parametrization for the forward-state equations:

$$F(m) = \begin{cases} p_o(\mathbf{x}, 0) = 0, \\ \frac{\partial p_o(\mathbf{x}, 0)}{\partial t} = 0, \\ \frac{1}{c_o(\mathbf{x})^2} \frac{\partial^2 p_o(\mathbf{x}, t)}{\partial t^2} - \nabla^2 p_o(\mathbf{x}, t) = f_s(\mathbf{x}, t), \\ \frac{1}{c_o(\mathbf{x})^2} \frac{\partial^2 \delta p(\mathbf{x}, t)}{\partial t^2} - \nabla^2 \delta p(\mathbf{x}, t) = \frac{\partial^2 p_o(\mathbf{x}, t)}{\partial t^2} m, \end{cases} \quad (3)$$

where p_o is the background wavefield, c_o is the background velocity model, δp is the perturbation wavefield. The adjoint state equations are:

$$F^*(m) = \begin{cases} \lambda(\mathbf{x}, T) = 0, \\ \frac{\partial \lambda(\mathbf{x}, T)}{\partial t} = 0, \\ \frac{1}{c_o(\mathbf{x})^2} \frac{\partial^2 \lambda(\mathbf{x}, t)}{\partial t^2} - \nabla^2 \lambda(\mathbf{x}, t) = \sum_r (S_{s,r}^T (S_{s,r} \delta p(\mathbf{x}, t) - d_{s,r})), \end{cases} \quad (4)$$

where λ is the adjoint state variable, and S^T is the adjoint of the sampling operator. Finally, the gradient is given by:

$$\frac{\partial}{\partial m} J(m) = \sum_s \int_0^T \lambda(\mathbf{x}, t) \frac{\partial^2 p_o(\mathbf{x}, t)}{\partial t^2} dt. \quad (5)$$

Please observe that the gradient calculation requires the background wavefield p_o to be saved at all time steps for it to be correlated with λ . Given λ has final boundary conditions (i.e. $\lambda(\mathbf{x}, T) = 0$), the efficient way to compute it is to propagate the adjoint wavefield associated with it backwards together with the calculation of the gradient in Equation 5, which constitutes the main problem of the LSRTM and RTM in general. In this work, the ZFP compression algorithm is used to compress p_o to decrease the storage size it occupies.

Dot product test

The forward state and the result of the gradient calculation should be orthogonal, which makes it a linear operator. A good test to check if the implementation is correct is to test for orthogonality using

the dot product test. From now on, the operator \mathbf{L} will be used to represent the application of the forward state on a discretized variable \mathbf{m} , which gives the following relation:

$$\mathbf{L}(\mathbf{m}) = \mathbf{d}_{calc}. \quad (6)$$

Given the above linear operator, the dot product test is given by:

$$\langle \mathbf{L}\hat{\mathbf{m}}, \hat{\mathbf{d}} \rangle = \langle \hat{\mathbf{m}}, \mathbf{L}^*\hat{\mathbf{d}} \rangle, \quad (7)$$

where $\hat{\mathbf{m}}$ is a random model, and $\hat{\mathbf{d}}$ is a random data. The left-hand side of the equation is the dot product between the forward state applied to the random model and the random data, and the right-hand side is the dot product between the adjoint state applied to the random data and the random model. The dot product test is a good test to check if the implementation is correct because if the result is not close to zero, there is something wrong with the implementation. For double precision data, the result is normally around 1×10^{-15} , and for single precision data, it is 1×10^{-6} . If the test does not pass, linear iterative solutions like CGLS (conjugate gradient least squares, Paige and Saunders (1982)), which are normally used for least-squares migration, might diverge after a few iterations. In this work, I actually use a relative dot product test result defined by:

$$\alpha = \left| \frac{\langle \mathbf{L}\hat{\mathbf{m}}, \hat{\mathbf{d}} \rangle - \langle \hat{\mathbf{m}}, \mathbf{L}^*\hat{\mathbf{d}} \rangle}{\langle \mathbf{L}\hat{\mathbf{m}}, \hat{\mathbf{d}} \rangle} \right|. \quad (8)$$

Overview of ZFP Compression Technique

ZFP is an innovative compression algorithm tailored for multidimensional arrays, employing a unique approach by segmenting the data into blocks of 4^d elements, where d is the number of dimensions of the input array. This strategy, reminiscent of techniques used in graphical texture compression, allows for independent processing of each block, enhancing the efficiency and flexibility of the compression process. The original work by Lindstrom (2014) described the first iteration of the ZFP algorithm. However, it has changed over the years, and the algorithm employed on version 1.0.0 is described in greater detail on Diffenderfer et al. (2019). Since the version employed in this work is the latter, the following explanation will focus on explaining the algorithm associated with it. A significant advantage of the ZFP compression algorithm is its three compression modes, fixed rate, fixed precision, and fixed accuracy, with different levels of control over the lossyness.

The compression workflow has several steps, the first of which is block segmentation. Afterwards, the values in the block are converted into a block-floating-point representation by doing a bit shift to the right and truncating (if using lossy compression modes), saving the highest base two exponent. Then, the resulting values are transformed using a near orthogonal decorrelating transform similar to the discrete cosine transform, sharing most properties with it. The values are then ordered with decreasing amplitude. Given the sign of an integer number, normally stored in the first bit, carries very little information, the next step is to convert the numbers in the block to a -2 binary base, called negabinary. The values in the block are then transposed, making the columns of this resulting binary matrix represent the bit planes, in which the conversion from binary to integer values would recover the original values with more accuracy the more columns are used, with the highest value bit planes being the lowest index columns. An embedded coding program then processes the bit planes to deduplicate zeros in the binary numbers. The different compression modes are realized at the end of the compression workflow, which consists of truncating the embedded coding program at a certain number of bit planes in a manner that aligns with specific compression objectives, whether they be related to size, precision, or accuracy, thereby offering a customizable compression output. If the compression is fixed-rate, the goal is to truncate the bitstream when the number of bits per value normalized for the block ($\frac{\text{bits}}{4^d}$) is equal to the specified rate. The number of bitplanes is pre-determined

for the fixed precision, so the truncation happens when the number of encoded bitplanes reaches that number. For fixed accuracy, the condition to stop is when a minimum number of bit planes are encoded, a number that is calculated based on the predefined tolerance and the maximum exponent of the values in the block. This tolerance is the maximum absolute value in a block. These steps are done independently per block, and the decompression consists of doing the steps in reverse.

In the experiments performed in this work, the multiframe compressor of the `SequentialCompression.jl` package¹ was used. It consists of splitting the input array in its last dimension, compressing each part in parallel using the ZFP algorithm, and saving the result in different files, one per thread. Additionally, the experiments in this work base themselves on analyzing rate and tolerances, and their effect on least-squares migration, given precision has very similar behaviour to rate.

Numerical experiment

The experiment constitutes running 10 CGLS iterations, along with a simple RTM (single iteration of LSRTM), with the linear operator described at Section , while saving the background wavefield using the multiframe compressor from the `SequentialCompression.jl`. The model is the a smoothed Marmousi synthetic model(Brougois et al., 1990), downsampled to the dimensions of $\{nz, nx\} = \{201, 512\}$, with $\{dz, dx\} = \{10, 10\}$ m. The number of shots used for the modelling and inversion was 100, equally distributed between the $x = 100$ m and $x = L_x$ m, where L_x is the domain size at that direction. The absorbing boundary condition border has $nb = 50$ in every direction. The number of timesteps is $nt = 2048$. The maximum frequency was 15 Hz. The observed data used in the inversion was calculated using the full second-order acoustic wave equation shown in Equation 1. Figure 1a shows the convergence curve for different rates and tolerances, compared against the lossless case. The curves only start to diverge at the 8th iteration for smaller rates and tolerances closer to one, as seen in the inset. Figure 1b shows the L2 norm of the lossless least-squares model and the lossy one for different rates and tolerances. A linear relationship exists between the absolute error measures in the L2 norm and the error tolerance, which is expected from its very definition. However, this is still important because the tolerance parameter only controls the error per block, not the absolute error of the whole array. For the rate case, the error sharply decreases as the rate increases linearly. Figure 2, and 3 show the resulting least squares migrated images for the different rates and tolerances, all in the same colour scale for comparison. There is essentially no difference to the naked eye at this scale. These figures also show a very important parameter called space-saving, which is defined as:

$$\text{space saving} = 1 - \frac{\text{compressed size}}{\text{uncompressed size}}, \quad (9)$$

where the uncompressed size is the size of the background wavefield in single-precision floating-point, and the compressed size is the size of the compressed background wavefield, summed up for all shots in both cases. For reference, the original size of the background wavefield summed over all shots without compression in this experiment is 168.61 Gb. These parameters show that even with the lowest rate and tolerance closest to one, you can get a good quality image with a space-saving upwards of 90%.

Conclusions

The results underscore that the LSRTM (Least Squares Reverse Time Migration) workflow can be effectively compressed using the ZFP compression algorithm, incurring only a marginal loss in convergence and image quality. The fidelity of the implementation is affirmed by the dot product test results, which confirm that the compression algorithm introduces no significant errors, maintaining integrity within the bounds of controlled lossiness. Notably, the space savings achieved are substan-

¹The package can be found at the GitHub repository: <https://github.com/AtilaSaraiva/SequentialCompression.jl>.

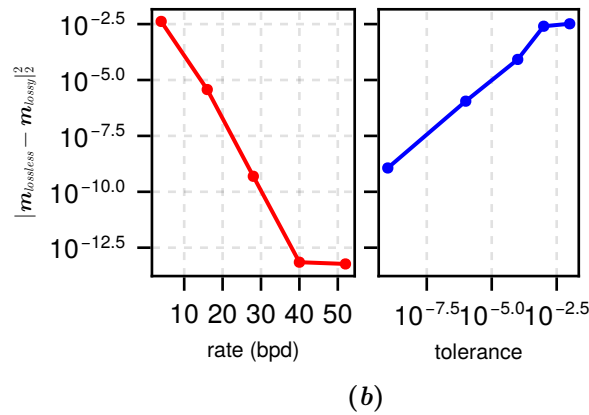
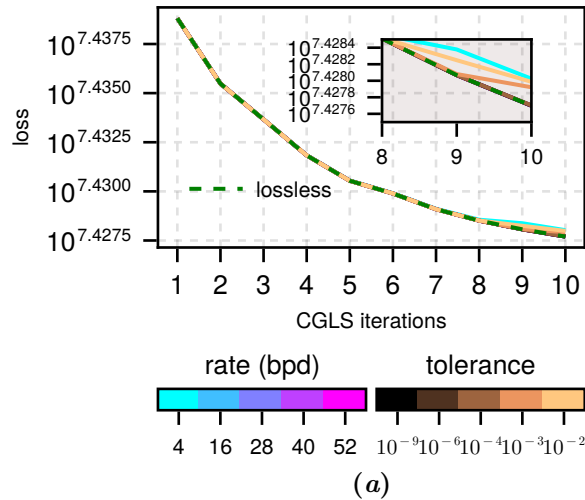


Figure 1: Convergence curves for different rates and tolerances (a). L2 norm of the lossless least-squares model and the lossy one for different rates and tolerances (b).

tial, even at higher rates and lower tolerances, underscoring the efficiency of the ZFP algorithm in data compression.

In conclusion, this study validates the practicality of employing the ZFP compression algorithm within the LSRTM workflow. It opens pathways for enhancing data management and computational efficiency in seismic inversion tasks. The broader implication of this research suggests a promising direction for optimizing data-intensive workflows in geophysical research, with potential applications extending beyond seismic inversion to other areas where large datasets are a significant challenge.

Acknowledgements

We want to thank the Signal Analysis and Imaging Group sponsors at the University of Alberta for supporting the stimulating research environment that allowed the preparation of this work.

References

- Brougois, A., M. Bourget, P. Lailly, M. Poulet, P. Ricarte, and R. Versteeg, 1990, Marmousi, model and data: Presented at the EAEG Workshop - Practical Aspects of Seismic Data Inversion, European Association of Geoscientists & Engineers.
- Cardesa, J., L. Hascoët, and C. Airliau, 2020, Adjoint computations by algorithmic differentiation of a parallel solver for time-dependent PDEs: *Journal of Computational Science*, **45**, 101155; doi: 10.1016/j.jocs.2020.101155.
- Di, S., and F. Cappello, 2016, Fast Error-Bounded Lossy HPC Data Compression with SZ: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 730–739.
- Diffenderfer, J., A. L. Fox, J. A. Hittinger, G. Sanders, and P. G. Lindstrom, 2019, Error Analysis of ZFP Compression for Floating-Point Data: *SIAM Journal on Scientific Computing*, **41**, A1867–A1898; doi: 10.1137/18M1168832.

- Dussaud, E., W. W. Symes, P. Williamson, L. Lemaistre, P. Singer, B. Denel, and A. Cherrett, 2008, Computational strategies for reverse-time migration: SEG Technical Program Expanded Abstracts 2008, Society of Exploration Geophysicists, 2267–2271.
- Griewank, A., 1992, Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation: *Optimization Methods and Software*, **1**, 35–54; doi: 10.1080/10556789208805505.
- Hascoet, L., and V. Pascual, 2013, The Tapenade automatic differentiation tool: Principles, model, and specification: *ACM Transactions on Mathematical Software*, **39**, 1–43; doi: 10.1145/2450153.2450158.
- Huang, Y., K. Zhao, S. Di, G. Li, M. Dmitriev, T.-L. D. Tonellot, and F. Cappello, 2023, Towards Improving Reverse Time Migration Performance by High-speed Lossy Compression: 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), IEEE, 651–661.
- Kukreja, N., J. Hükelheim, M. Louboutin, J. Washbourne, P. H. J. Kelly, and G. J. Gorman, 2022, Lossy checkpoint compression in full waveform inversion: a case study with ZFPv0.5.5 and the overthrust model: *Geoscientific Model Development*, **15**, 3815–3829; doi: 10.5194/gmd-15-3815-2022.
- Lindstrom, P., 2014, Fixed-Rate Compressed Floating-Point Arrays: *IEEE Transactions on Visualization and Computer Graphics*, **20**, 2674–2683; doi: 10.1109/TVCG.2014.2346458.
- Paige, C. C., and M. A. Saunders, 1982, LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares: *ACM Transactions on Mathematical Software*, **8**, 43–71; doi: 10.1145/355984.355989.
- Plessix, R.-E., 2006, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications: *Geophysical Journal International*, **167**, 495–503; doi: 10.1111/j.1365-246X.2006.02978.x.
- Symes, W. W., 2007, Reverse time migration with optimal checkpointing: *GEOPHYSICS*, **72**, SM213–SM221; doi: 10.1190/1.2742686.
- Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: *GEOPHYSICS*, **49**, 1259–1266; doi: 10.1190/1.1441754.
- Yang, P., R. Brossier, and J. Virieux, 2016, Wavefield reconstruction by interpolating significantly decimated boundaries: *GEOPHYSICS*, **81**, T197–T209; doi: 10.1190/geo2015-0711.1.
- Yang, P., J. Gao, and B. Wang, 2014, RTM using effective boundary saving: A staggered grid GPU implementation: *Computers & Geosciences*, **68**, 64–72; doi: 10.1016/j.cageo.2014.04.004.

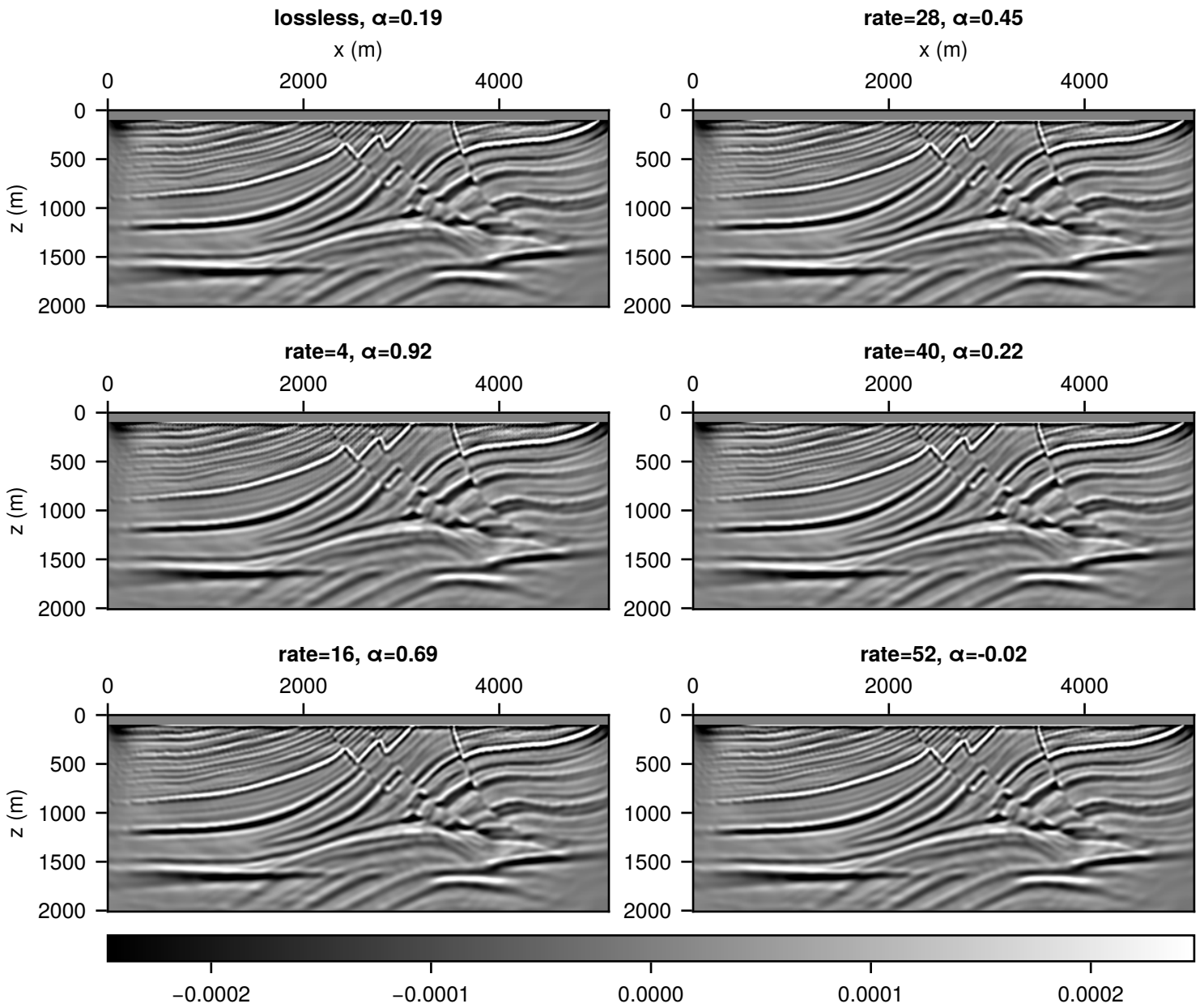


Figure 2: CGLS migrated image after ten iterations for different compression rates. Original summed over shots background wavefield size=168.61 GB

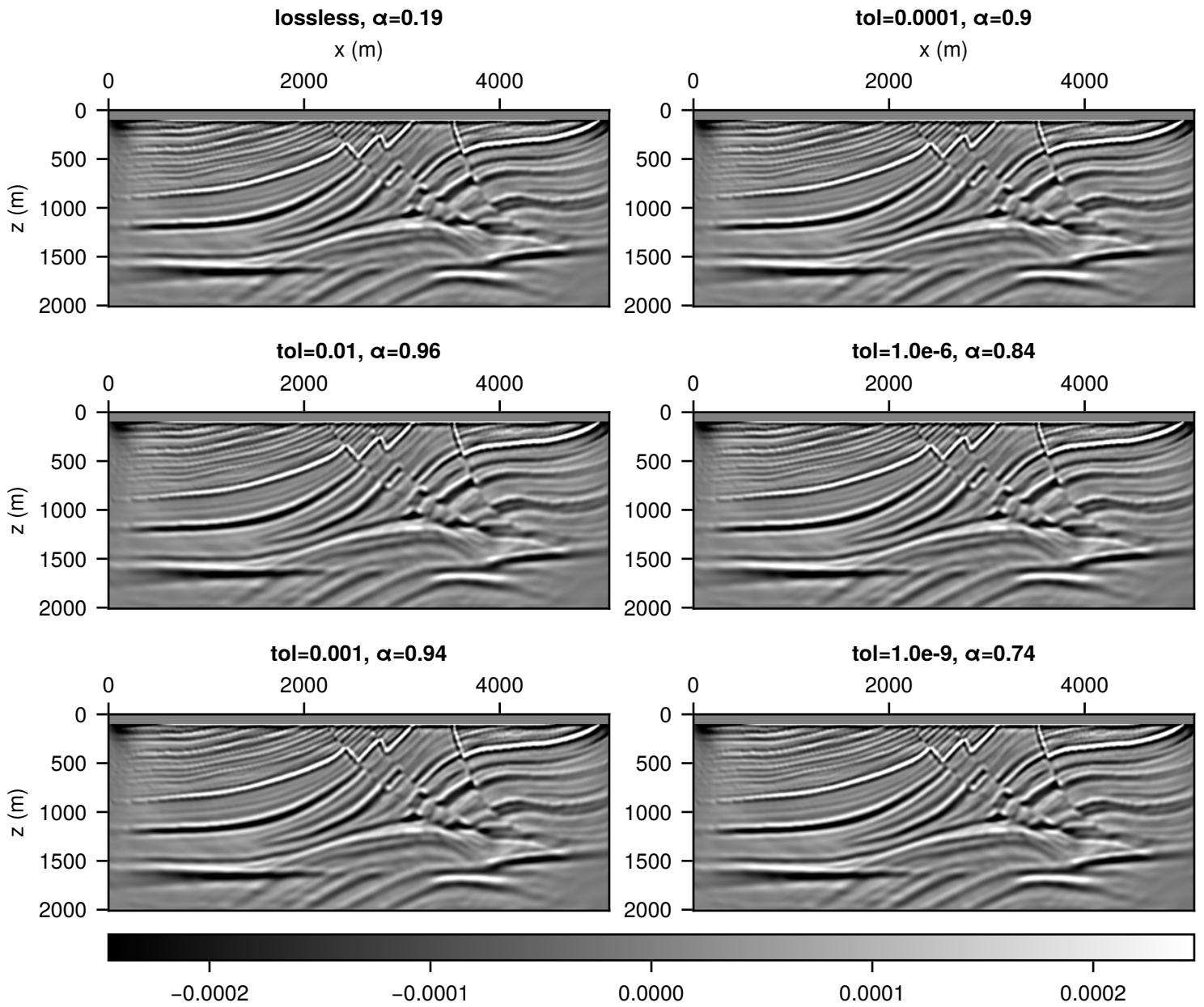


Figure 3: CGLS migrated image after ten iterations for different compression tolerances. Original summed over shots background wavefield size=168.61 GB