International Journal of Parallel Programming (2026) 54:18
https://doi.org/10.1007/s10766-026-00822-w

Parallelization as a Source of Randomness in Iterative Tomography: A Case Study

Zachary H. Levine[1] · **Collin Guan**[2]

Received: 14 August 2025 / Accepted: 4 April 2026

Abstract
We ported a tomographic reconstruction code from MPI to OpenMP (formally, the software systems Message Passing Interface and Open Multi-Processing). Due to randomness associated with the order of floating point operations, we found that verifying the correctness of the port required a statistical comparison of the answers. These agree within acceptable bounds. As a function of iteration number, the solutions generated with different numbers of cores at first diverge, then converge. We also studied the effect of having different random number starts. In this case, the answers always converge toward each other, although their variation is greater than the variation associated with changing the number of cores with a fixed random number seed.

Keywords Parallelization · Floating-point randomness · Iterative methods · Tomography

Mathematics Subject Classification 65N21–Numerical Methods for Inverse Problems · 65Y05–Parallel Computation

✉ Zachary H. Levine
zlevine@nist.gov

Collin Guan
collin.guan@nist.gov

[1] Quantum Measurement Division, National Institute of Standards and Technology, Gaithersburg, MD 20899-8410, USA

[2] Research Data and Computing Office, National Institute of Standards and Technology, Gaithersburg, MD 20899-6410, USA

Published online: 29 June 2026

1 Introduction

Transform-based reconstruction techniques such as filtered backprojection [1–3] dominated the thinking and practice in the early days of tomography. Early iterative techniques such as the Algebraic Reconstruction Technique (ART) [4, 5] and the Simultaneous Iterative Reconstruction Technique (SIRT) [6, 7] have also been widely used. These techniques require stopping when the image appears to be reconstructed to avoid an unreasonable solution.

Gradually, model-based reconstruction based on known physical interactions and a maximum likelihood or Bayesian principle were introduced. Having a well-defined objective function allows the user to iterate to some chosen stopping criterion without fear of divergence. These methods are more computationally intensive than transform-based techniques, but they can operate with less dose to the sample and with unstructured data. The Bouman-Sauer prior [8] is an important precursor to the more widely adopted Minimization of Total Variation [9] associated with compressed sensing. Although the mathematical arguments behind compressed sensing are more developed [10] than those of Bouman-Sauer, if the weights of the Bouman-Sauer prior are chosen to be a finite difference approximation to a derivative, the methods are similar in practice. This work is based on the Bouman-Sauer prior, in part because it has an analytic derivative which is needed by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) iterative method. [11, 12]

In the present work, we describe a port of the code TOMOSCATT based on the Bouman-Sauer prior from MPI [13] to OpenMP [14] because the latter can be more memory efficient, particularly if a large number of cores share a common memory. [15] Increasingly, this is the case in contemporary hardware. We needed to verify that the answer was independent of the number of cores and the type of parallelism. In doing so, we found that changing order of floating point operations due to changing from MPI to OpenMP or by changing the number of cores was a source of noise. The verification of the correctness of the port in the face of this noise source was sufficiently subtle that we choose to describe it here.

Earlier work includes a study of the convergence of parallel tomography algorithms as a function of the number of cores [16] and an attempt to minimize the number of iterations required when changing the number of cores. [17] The effect of inexact arithmetic on convergence in iterative methods generally has also been explored. [18]

2 Data and Algorithm

Before discussing the port of the TOMOSCATT code, we discuss the research context. The code was used to reconstruct data acquired by a hybrid electron-x-ray microscope which was built at the National Institute of Standards and Technology in Boulder. [19] Results from the reconstruction of an integrated circuit interconnect were presented earlier along with the reconstruction algorithm. [20] To avoid duplication, we summarize the main points here. Briefly, the code maximizes an *a posteriori* log likelihood (MAP) presented originally by Bouman and Sauer. [8] The Bayesian

weighting is chosen to closely resemble the minimization of total variation. [21] The algorithm is iterative with two principal phases. In the first phase, an estimate for the reconstruction (which can be random numbers to start) is frozen and projections are taken to predict various observations. These are compared to experimental results. A likelihood and its gradient are computed from these projections. In the second phase, the likelihood and its gradient are used with the bounded variant of the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS-B). [11, 12] We use their publicly available subroutines [12] in TOMOSCATT. Although the code is capable of handling scattering, [22] the results presented here are based on projections.

Our port was tested on a lower resolution reconstruction of data which we had published recently [20] from a hybrid electron-x-ray microscope. The principal difference between the figure given here and the one published is that we used a 7 point source here instead of an extended 200 point source. The results in Fig. 1 show wiring and chemical–mechanical-polishing (CMP) fill. The wiring is approximately 160 nm across. We accept lower spatial resolution than in the previous study because we have 66 reconstructions for this study vs. 1 in the previous study. By using a seven-point star, the computer time requirement was about a factor of 16 less than for the earlier published results which used 200 points to represent the source. The parameters which were used in the reconstruction are given in Table 1. The parameter "factor" is just below what is described as "medium accuracy" by Nocedal's group. [11, 12] All of the reconstructions required at least 675 iterations and none required as many as 725 iterations to reach convergence. We used up to iteration 675 for comparison regardless of whether the convergence criterion was strictly met. The solution is restricted to [0, 2] at each voxel. This is done on physical grounds. Because TOMOSCATT uses physical absorption cross sections and the reconstructed density is scaled to the density of copper, we expect the solution to lie between [0, 1] at each voxel. This is largely true, although the reconstruction does have a few values above 1.

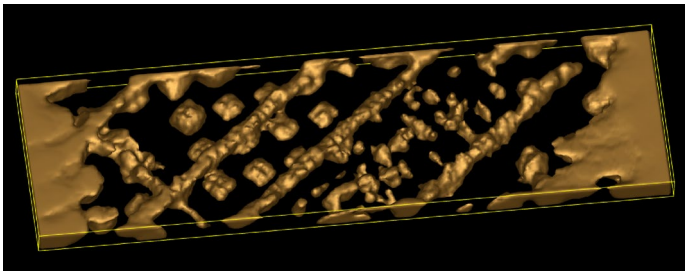

Fig. 1 An isosurface from the reconstruction used throughout this paper created using IMOD. [23] The features are approximately 500 nm across. The reconstructions are based on the same data as used in Ref. [20]. Two related movies are given in the supplementary material

Table 1 Parameters used in the reconstruction, similar to Ref. [20]. The variable names are for the number of observations, the number of voxels, and the reconstruction volume respectively. The variable p is a parameter to the Bayesian prior distribution recommended by Bouman and Sauer for the material inspection problem

Variable	Value	Comment
N_{obs}	2 631 025	Unstructured
N_{voxel}	7 500 000	$500 \times 150 \times 100$ voxels
V_{recon}	960 μm^3	20 μm $\times$ 6 μm $\times$ 8 μm
Voxel size	40 nm $\times$ 40 nm $\times$ 80 nm	x, y, and z
p	1.1	Recommended for material inspection [8]
BFGS iterations	16	Retained before restart [11, 12]
Convergence parameter	10^8	L-BFGS-B variable "factor" [11, 12]
Source points	7	Each is a 7 point star
Source point spacing	60 nm 50 nm 50 nm	x, y, and z
Bounds	[0,2]	Random numbers started in [0, 1)

3 Results from the Port of the Code

The reason for porting TOMOSCATT from MPI to OpenMP was to save memory. MPI assumes that each core has its own memory, whereas OpenMP was designed with shared cores in mind. Such computers were not available in the early 1990's when MPI was developed. OpenMP was developed starting in the late 1990s to take advantage of emerging shared memory computers. In Fig. 2, we see that the speed of our MPI implementation is comparable to our OpenMP implementation. The MPI version is a little faster with modest numbers of cores, but its speed plateaus in a way which the OpenMP version does not. The difference of memory usage is dramatic. Whereas the MPI version requires about 2.7 GB per core with linear scaling of memory (2.5 GB to 88.5 GB for 1 and 32 cores, respectively), the OpenMP version only increases its memory requirement from 2.7 GB to 4.4 GB as the number of threads is increased from 1 to 32.

The wall clock times are 65105 s to 7857 s and 64024 s to 6740 s for MPI and OpenMP, respectively. The scaled reciprocals of these times are shown in Fig. 2a. The efficiency of the additional cores is about 33% to 50% over most of the range. One limit is that the L-BFGS-B subroutines are serial. We are not aware of a public, parallel version available in Fortran, although it is available in other languages. [24]

Proving the correctness of the port was more involved than we first imagined and motivated us to write this short report. We found that the results were diverging from each other in the early iterations. Eventually, we realized that floating point arithmetic is a source of randomness, [18] and the L-BFGS-B iterations were causing the solutions to grow apart from each other as the number of cores was varied. We knew that the algorithm would converge given a random number start, so we conjectured that the reconstructed values would grow apart until they were as diverse as those starting from random numbers, and then converge along the same curve.

We formalized the design of our study as follows. For one part, we picked 17 seeds for the random number generator and solved the tomography problem shown in Fig. 1 independently, 2×17 times where the factor of 2 is for the independent

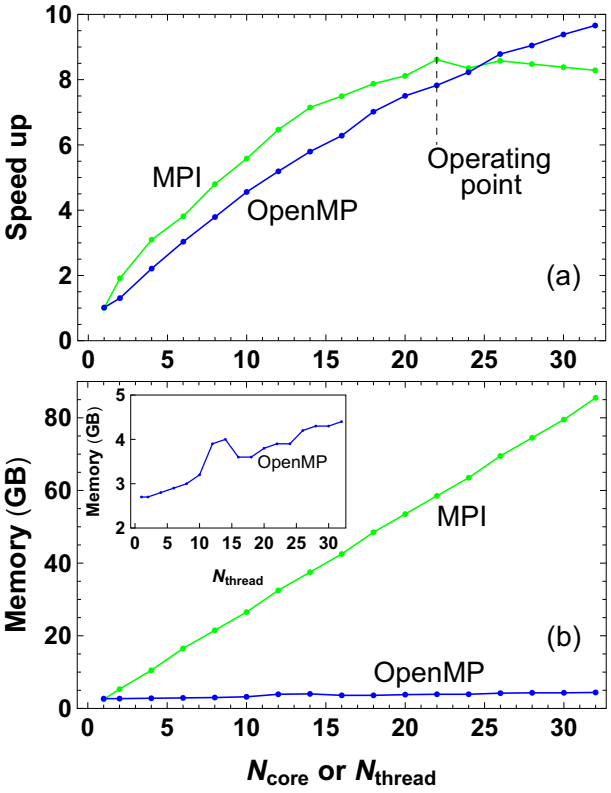

Fig. 2 Resource requirements as a function of the number of cores: **a** the speed up and **b** memory use for (green) MPI and (blue) OpenMP versions of TOMOSCATT. The speed up is the factor of time reduction compared to the single-core MPI case. The dashed line marked "Operating point" marks the 22 cores or threads which were used with varying random number starts. The inset in part (b) repeats the OpenMP data on an expanded scale. Numerical values of the data are given in the supplementary material

runs with MPI or OMP. The same seeds were used for each, implying they each got the same 17 randomly selected initial guesses. In the second part, we froze the seed and instead varied the number of cores (for MPI) or the number of threads (for OMP) from 1, 2, 4, ..., 32. Again, there are $2 \times 17 = 34$ reconstructions. However, we reused two of the reconstructions, so the total study has 66 reconstructions.

The difference between the solutions was quantified as follows. First, we found the mean solution at each iteration

$$\bar{f}_r^{(n)} = \frac{1}{N_i} \sum_i f_r^{(n,i)} \tag{1}$$

where r indexes the N_r voxels in 3D, i denotes one of the N_i samples, $\bar{f}_r^{(n)}$ is the mean reconstruction at iteration number n. We kept only the central $\frac{1}{8}$ of the voxels in order to reduce disk storage given that we saved the many reconstructions and

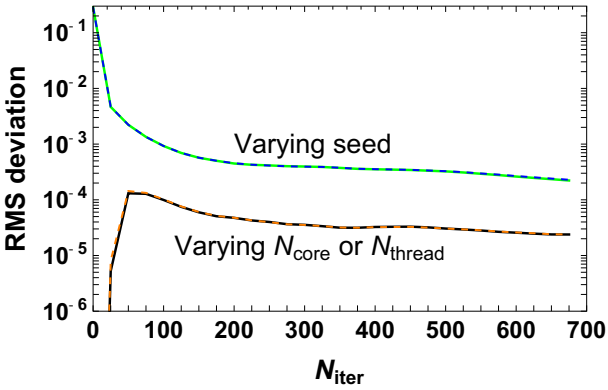

Fig. 3 The estimated standard deviation of 17 solutions generated with each of four conditions as a function of iteration number (0 to 675). The four conditions are (solid green) MPI, varying the random seed, (dashed blue) OpenMP, varying the random seed, (solid orange) MPI, varying the number of cores, (dashed black) MPI, varying the number of threads. Numerical values of the data are given in the supplementary material

many iterations. Because the central region of a reconstruction is the most important in practice, this was reasonable choice. Then, we found the population estimate of the standard deviation s_n averaged over the voxels using

$$s_n^2 = \frac{1}{N_i - 1} \frac{1}{N_r} \sum_{ri} \left(f_r^{(n,i)} - \bar{f}_r^{(n)} \right)^2 . \tag{2}$$

These values were found separately for each of four conditions as detailed in the caption to Fig. 2.

When we started from random seeds distributed uniformly over [0, 1) with a fixed number of cores or threads (specifically 22), the standard deviation was reduced systematically by four orders of magnitude from its analytically-known initial value (plotted as iteration number 0) very near $1/\sqrt{12} \approx 0.288$. The observed standard deviations arising from varying the number of cores followed a similar pattern to our conjecture, rising from a machine zero ($< 10^{-14}$) to a peak near 10^{-4}, before following the same convergence curve as before, albeit displaced by a factor of 10 as shown in Fig. 3. We did not conjecture the factor of 10 difference. Somehow, the influence of the randomness injected by floating point arithmetic is less than the randomness associated with starting with different pseudo-random numbers.

Returning to our starting point, we wanted to validate the port of our code from MPI to OpenMP in the face of randomness associated with changing the order of floating point operations. To this end, we form a *Z*-score-like quantity

$$Z = \frac{\left[N_r^{-1} \sum_r \left(\bar{f}_r^{(N,OMP)} - \bar{f}_r^{(N,MPI)} \right)^2 \right]^{1/2}}{N_i^{-1/2} \left[\left(s_N^{(OMP)} \right)^2 + \left(s_N^{(MPI)} \right)^2 \right]^{1/2}} \tag{3}$$

where $n = N$ is the final iteration (i.e., 675). The factor of $N_i^{-1/2}$ is included to turn the estimated standard deviation into an estimated standard deviation of the mean. From Fig. 3, $s_N^{(OMP)} \approx s_N^{(MPI)}$. The values are $Z = 0.987$ and $Z = 0.258$ for the random number start and varying the number of cores or threads, respectively. In other words, the RMS variations in the average of the solutions with each parallelization method is no more than we would expect based on the standard deviation of the mean estimated for each method. The meaning of the standard deviation of the mean is as follows: if we had simply run either the MPI code or the OMP code with two sets of 17 random numbers, we would have comparable variation to what we found switching from MPI to OMP. Our port does generate the same answer as the original code up to the noise which is inherent in the algorithm when implemented with double precision floating point numbers.

4 Conclusions

Computer architectures have evolved over the past three decades to increase the amount of cores sharing memory. As such, OpenMP may be a better choice for parallelization than MPI for smaller projects because it makes better use of the available memory. For larger projects which require node-to-node communication, a hybrid OpenMP-MPI code may be a good choice.

Verifying the correctness of the the iterative tomography code TOMOSCATT when ported from one system to another required the use of statistical techniques to ensure that the solution obtained was in agreement. The divergence associated with early L-BFGS-B iterations does not lead to a difference in the converged solution. This is not too surprising since the method will tend to the same solution even with a random number start. Characterizing such behavior might be an interesting problem for our mathematical colleagues. The larger number of practitioners may need to consider noise arising from the order of floating point operations when verifying parallel codes.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10766-026-00822-w.

Acknowledgements The project used resources in NIST's Research Computing / High-Performance Computing (RCHPC) facility. ZHL was supported in part by the NIST CHIPS Metrology Program.

Author Contributions Zachary H Levine wrote the legacy code which was ported, made the argument to justify doing the port, and made the suggestion that parallelization was a source of randomness which needed to be accounted for statistically. He wrote the original draft. Collin Guan did the port from MPI to OpenMP and the computer runs. He also made suggestions to revise the manuscript.

Data availability The data plotted in Figs. 2 and 3 are available in the file LevineGuanFigureInfo.pdf in the Supplementary Material. This file also contains a description of two short videos visualizing the reconstruction shown in Fig. 1 which are also available in the Supplementary Material.

Declarations

Conflicts of interest Zachary H. Levine declares potential competing financial interest in "Efficient Method for Tomographic Reconstruction in the Presence of Fresnel Diffraction," US Patent 12,367,563 (2025) and "A Dimensional Reference for Tomography," US Patent 7,967,507 B2 (2011). Collin Guan declares no competing financial interests.

References

1. Shepp, L.A., Logan, B.F.: The Fourier reconstruction of a head section. IEEE Trans. Nucl. Sci. **21**, 21–43 (1974)
2. Hanson, K.M.: On the optimality of the filtered backprojection algorithm. J. Comput. Assist. Tomogr. **4**, 361–363 (1980)
3. Katsevich, A.: An improved exact filtered backprojection algorithm for spiral computed tomography. Adv. Appl. Math. **32**, 681–697 (2004)
4. Gordon, R., Bender, R., Herman, G.T.: Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and x-ray photography. J. Theor. Biol. **29**, 471–481 (1970)
5. Andersen, A.H.: Algebraic reconstruction in CT from limited views. IEEE Trans. Med. Imaging **8**, 50–55 (1989)
6. Trampert, J., Leveque, J.-J.: Simultaneous iterative reconstruction technique: Physical interpretation based on the generalized least squares solution. J. Geophys. Res.: Solid Earth **95**, 12553–12559 (1990)
7. Gregor, J., Benson, T.: Computational analysis and improvement of SIRT. IEEE Trans. Med. Imaging **27**, 918–924 (2008)
8. Bouman, C., Sauer, K.: A generalized Gaussian image model for edge-preserving MAP estimation. IEEE Trans. Image Process. **2**, 296–310 (1993)
9. Candes, E.J., Guo, F.: New multiscale transforms, minimum total variation synthesis: Applications to edge-preserving image reconstruction. Signal Process. **82**, 1519–1543 (2002)
10. Candès, E.J., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Trans. Inf. Theory **52**, 489–509 (2006)
11. Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math. Softw.(TOMS) **23**, 550–560 (1997)
12. Morales, J.L., Nocedal, J.: Remark on Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. ACM Trans. Math. Softw.(TOMS) **38**, 1–4 (2011)
13. https://www.mpi-forum.org/docs, accessed 26 June 2025
14. https://www.openmp.org/specifications/, accessed 26 June 2025
15. Kang, S.J., Lee, S.Y., Lee, K.M.: Performance comparison of OpenMP, MPI, and MapReduce in practical problems. Adv. Multim. **2015**, 575687 (2015)
16. Zheng, J., Saquib, S.S., Sauer, K., Bouman, C.A.: Parallelizable Bayesian tomography algorithms with rapid, guaranteed convergence. IEEE Trans. Image Process. **9**, 1745–1759 (2000)
17. Smith, I.M., Margetts, L.: The convergence variability of parallel iterative solvers. Eng. Comput. **23**, 154–165 (2006)
18. Loizou, N., Richtárik, P.: Convergence analysis of inexact randomized iterative methods. SIAM J. Sci. Comput. **42**, A3979–A4016 (2020)

19. Nakamura, N., Szypryt, P., Dagel, A.L., Alpert, B.K., Bennett, D.A., Doriese, W.B., Durkin, M., Fowler, J.W., Fox, D.T., Gard, J.D., et al.: Nanoscale three-dimensional imaging of integrated circuits using a scanning electron microscope and transition-edge sensor spectrometer. Sensors **24**, 2890 (2024)
20. Levine, Z.H., Alpert, B.K., Dagel, A.L., Fowler, J.W., Jimenez, E.S., Nakamura, N., Swetz, D.S., Szypryt, P., Thompson, K.R., Ullom, J.N.: A tabletop X-ray tomography instrument for nanometer-scale imaging: Reconstructions. Microsyst. Nanoeng. **9**, 47 (2023)
21. Rodríguez, P., Wohlberg, B.: Efficient minimization method for a generalized total variation functional. IEEE Trans. Image Process. **18**, 322–332 (2008)
22. Levine, Z.H., Blattner, T.J., Peskin, A.P., Pintar, A.L.: Scatter corrections in x-ray computed tomography: a physics-based analysis. J. Res. Nat. Inst. Stand. Technol. **124**, 1 (2019)
23. Kremer, J.R., Mastronarde, D.N., McIntosh, J.R.: Computer visualization of three-dimensional image data using IMOD. J. Struct. Biol. **116**, 71–76 (1996)
24. Fei, Y., Rong, G., Wang, B., Wang, W.: Parallel L-BFGS-B algorithm on GPU. Computers Graph. **40**, 1–9 (2014)