SINTECHCOM: Science, Technology and Communication Journal
Vol. 6, No. 3, June 2026, pp. 241-254, DOI: 10.59190/stc.v6i3.374

Workflow reproducibility and observability layer for Tomofast-x gravity inversion experiments

I Wayan Pio Pratama
Department of Information Technology, Politeknik eLBajo Commodus, Labuan Bajo 86763, Indonesia

ABSTRACT

Tomofast-x is an open-source parallel platform for gravity and magnetic inversion; however, reproducible execution, runtime observability, and workflow traceability are commonly managed outside the inversion software itself. This study presented a provenance-aware and telemetry-aware experimentation environment for reproducible Tomofast-x workflows and evaluated it using four publicly available reference scenarios archived on Zenodo. Each scenario was reproduced three times, resulting in 12 platform-managed runs executed through isolated workspaces with structured provenance and telemetry collection. The reproduced solutions showed close agreement with the reference outputs, with mean absolute root mean square error differences ranging from approximately 2.47×10^{-11} to 4.72×10^{-9}. Runtime telemetry revealed substantial operational differences between scenarios. The uncompressed baseline required approximately 10.9 GB peak memory, whereas compressed scenarios required approximately 202 – 247 MB. Runtime decreased from approximately 436 s in the baseline case to approximately 300 – 340 s in compressed executions. Telemetry was successfully collected for all runs, including processor utilization, observed process-level RAM, runtime progression, and message passing interface (MPI) worker activity. Workflow robustness was further evaluated using injected failure cases involving corrupted parameter files, missing data grids, invalid execution paths, and simulated MPI failures. The results demonstrated that the proposed platform provided reproducible, provenance-aware, and telemetry-aware experimentation support for workstation-scale Tomofast-x workflows.

ARTICLE INFO

Article history:

Received May 27, 2026
Revised Jun 10, 2026
Accepted Jun 11, 2026

Keywords:

Gravity Inversion
Provenance Tracking
Reproducible Workflows
Runtime Observability
Tomofast-x

Corresponding Author
E-mail address: piopratama2@gmail.com

1. INTRODUCTION

Gravity inversion is widely used in computational geophysics to estimate subsurface density distributions from observed gravity data. Modern potential-field inversion increasingly depends on numerical modelling, large model grids, repeated parameter testing, and computationally intensive execution workflows. Tomofast-x has provided an open-source parallel platform for gravity and magnetic inversion, including support for constrained inversion, topography, and wavelet-based matrix compression [1, 2]. These capabilities make Tomofast-x suitable for large geophysical inversion problems, but the execution of repeated inversion experiments still often depends on manually prepared parameter files, command-line runs, separate output folders, and post-processing outside the inversion software itself.

Reproducibility has become a central requirement in computational science because scientific results should be traceable, repeatable, and verifiable by other researchers [3, 4]. In computational workflows, reproducibility does not only depend on source code availability. It also depends on the preservation of input data, parameter settings, runtime environment, execution logs, software versions, and output artifacts [5]. The FAIR principles emphasize that digital research objects should be findable, accessible, interoperable, and reusable. Recent work on FAIR computational workflows has

extended this concern from static datasets to executable computational processes, where workflow descriptions, dependencies, execution traces, and provenance records become part of the research object itself [6].

Provenance is especially important for scientific workflows because it records the relationship between inputs, processes, agents, and outputs. The W3C PROV model formalizes provenance as information about entities, activities, and agents involved in producing data or other research objects [7]. Workflow-oriented provenance systems such as AiiDA and Workflow Run RO-Crate show that structured provenance can improve automation, traceability, sharing, and reproducibility across computational experiments [8, 9]. These developments suggest that scientific software should not only produce numerical results but also preserve the context in which they were generated [10].

Runtime observability is another important requirement for computational workflows. Scientific inversion runs can be computationally expensive, long-running, and difficult to diagnose when they fail or stall. Recent studies on monitoring and observability for scientific workflows argue that runtime metrics, traces, and execution behavior are necessary to understand performance, detect failures, and support workflow-level diagnosis [11, 12]. In inversion workflows, useful runtime signals include processor utilization, memory usage, MPI worker activity, output growth, log activity, and execution state. Without these signals, a user may know only whether a run ended, but not how it behaved during execution.

Open-source geophysical inversion frameworks have improved transparency and extensibility in computational geophysics. For example, Harmonica provides an open-source framework for forward modeling, inversion, and processing of gravity and magnetic data [13]. Tomofast-x similarly contributes an open-source inversion engine for potential-field data, with emphasis on parallel execution and matrix compression. However, these inversion engines do not by themselves fully solve the workflow-level problems of experiment cloning, provenance preservation, runtime telemetry, failure traceability, and structured comparison between reproduced runs and reference results. These concerns are increasingly important when inversion experiments are repeated on workstation-scale systems, where memory limits, runtime variability, and execution failures can directly affect research productivity.

This study addresses that gap by presenting a provenance-aware and telemetry-aware experimentation environment for reproducible Tomofast-x gravity inversion workflows. The proposed platform was designed to preserve immutable reference scenarios, clone execution workspaces, rewrite parameter paths safely, collect runtime telemetry, store provenance metadata, classify workflow failures, and support structured comparison between reproduced executions and available reference outputs when such references are available. The platform was evaluated using publicly available Tomofast-x reference scenarios archived on Zenodo. Four scenarios were reproduced three times each, resulting in 12 platform-managed runs. The evaluation focused on workflow reproducibility, runtime observability, provenance traceability, and failure robustness rather than proposing a new inversion algorithm or a new compression method.

The main contribution of this work is not a new mathematical formulation for gravity inversion, but a structured experimentation framework for reproducible Tomofast-x workflow execution and runtime observability. The framework integrates provenance tracking, telemetry collection, repeated reference-scenario reproduction, failure traceability, and visual output inspection within a workstation-scale experimentation environment. The study specifically focuses on validating reproducibility and workflow observability using archived reference scenarios executed on a Windows workstation environment, rather than proposing a general-purpose HPC workflow orchestration system. This contribution is complementary to previous Tomofast-x research, which focused on inversion algorithms and computational performance, whereas this work focuses on execution traceability, runtime monitoring, and reproducible workflow management.

Unlike general workflow orchestration frameworks such as AiiDA or provenance packaging approaches such as RO-Crate, the proposed framework focuses specifically on reproducible Tomofast-x execution monitoring at the workflow level. The platform emphasizes runtime telemetry collection, repeated scenario reproduction, execution traceability, failure-state preservation, and experiment observability during workstation-scale gravity inversion runs. Rather than replacing existing workflow-management ecosystems, the framework was designed as a lightweight experimentation and reproducibility layer tailored to Tomofast-x workflow execution and validation.

2. RESEARCH METHODS

This research used an experimental software-validation approach to evaluate whether a platform-managed workflow could reproduce, monitor, and trace Tomofast-x gravity inversion experiments. The method consisted of five main stages: data acquisition, reference scenario preparation, platform-based reproduction, telemetry and provenance collection, and validation through repeated execution and controlled failure testing. The study did not modify the Tomofast-x inversion algorithm. Instead, it evaluated the reproducibility and observability of Tomofast-x workflows when executed through a structured experimentation environment.

2.1. Data Acquisition and Reference Scenarios

The experimental data were obtained from the publicly available Gravity Inversions Wavelet compression dataset archived on Zenodo. The dataset contains gravity inversions run with Tomofast-x, model sizes ranging from approximately 0.4 million to 12 million cells, and wavelet compression applied to the sensitivity matrix [14]. The present study used the smallest available model group, 417720, to keep the experiments feasible on a workstation-scale system. The Zenodo record includes the H_417720.tar.gz archive and the associated data_grids.zip file, which provide the reference inversion scenarios and grid data used in this work [14]. The dataset-generation methodology, compression experiments, and theoretical memory-behavior analysis associated with the archived scenarios are further described in the accompanying dataset paper by Bruce et al. [15].

Four reference scenarios were selected: 0, 2.4382, 7.6448, and 42.166. Each scenario contained a Tomofast-x parameter file, model grid, data grid, reference output model, convergence cost file, final gravity misfit file, and execution log. The selected scenarios were treated as immutable reference cases. They were not modified directly during the experiments. This design followed reproducible workflow principles, where original inputs and reference outputs should remain preserved while derived executions are recorded separately [3, 4].

Table 1. Reference scenarios used in the experiment scenario label.

Scenario label	Model cells	Main files used
0	417720	parfile.txt, 417720_mgrid.txt, 417720_dgrid.txt, model.vtu, costs.txt
2.4382	417720	parfile.txt, 417720_mgrid.txt, 417720_dgrid.txt, model.vtu, costs.txt
7.6448	417720	parfile.txt, 417720_mgrid.txt, 417720_dgrid.txt, model.vtu, costs.txt
42.166	417720	parfile.txt, 417720_mgrid.txt, 417720_dgrid.txt, model.vtu, costs.txt

The compression rate shown in Table 1 was reconstructed from the original Tomofast-x parameter file of each scenario. This prevented the experimental analysis from relying only on folder labels and ensured that the actual inversion configuration was recorded as part of the provenance metadata.

2.2. Experimental Environment

All experiments were conducted on a workstation-scale system running Windows 11 Home Single Language 64-bit (Build 26200). The hardware platform used an AMD Ryzen 9 8940HX processor with 32 logical CPUs and 32 GB RAM. Runtime telemetry, processor utilization, observed process-level RAM, and workflow observability measurements were collected during execution through the proposed platform environment. The experiments used DirectX 12-enabled system drivers and workstation-local storage throughout all reproduced runs.

The workstation-scale configuration was intentionally selected to evaluate whether the proposed platform could support reproducible Tomofast-x workflows under practical hardware constraints rather than requiring dedicated HPC infrastructure.

2.3. Platform Workflow Design

The proposed platform was designed to manage Tomofast-x experiments without changing the numerical inversion engine. The platform workflow consisted of four core components: a reference scenario loader, an isolated execution workspace generator, a telemetry monitor, and a comparison module. This structure was consistent with scientific workflow and provenance concepts, where

computational experiments are represented as a relationship between input entities, execution activities, software agents, and output artifacts [16, 17].

The reference scenario loader read the original author scenario and extracted the required files. The execution workspace generator copied the parameter file, model grid, and data grid into a new isolated folder for each run. The parameter file was then rewritten so that Tomofast-x used the copied grid files and wrote outputs to a run-specific output folder. This prevented accidental overwriting of reference data and preserved a clear lineage between the original scenario and the reproduced run.

The telemetry monitor recorded runtime information during execution. The monitored signals included processor utilization, observed process-level RAM, runtime progression, MPI worker count, log activity, and output-folder growth. Monitoring runtime behavior is important for long-running scientific workflows because failures and stalled executions cannot always be understood from final outputs alone [11, 12]. When reference outputs were available, the comparison module evaluated reproduced outputs using root mean square error, data cost, output metadata, and visual model inspection.

Figure 1 shows the main interface of the proposed platform, including scenario management, execution monitoring, runtime telemetry collection, and reproduced-run traceability during Tomofast-x workflow execution.

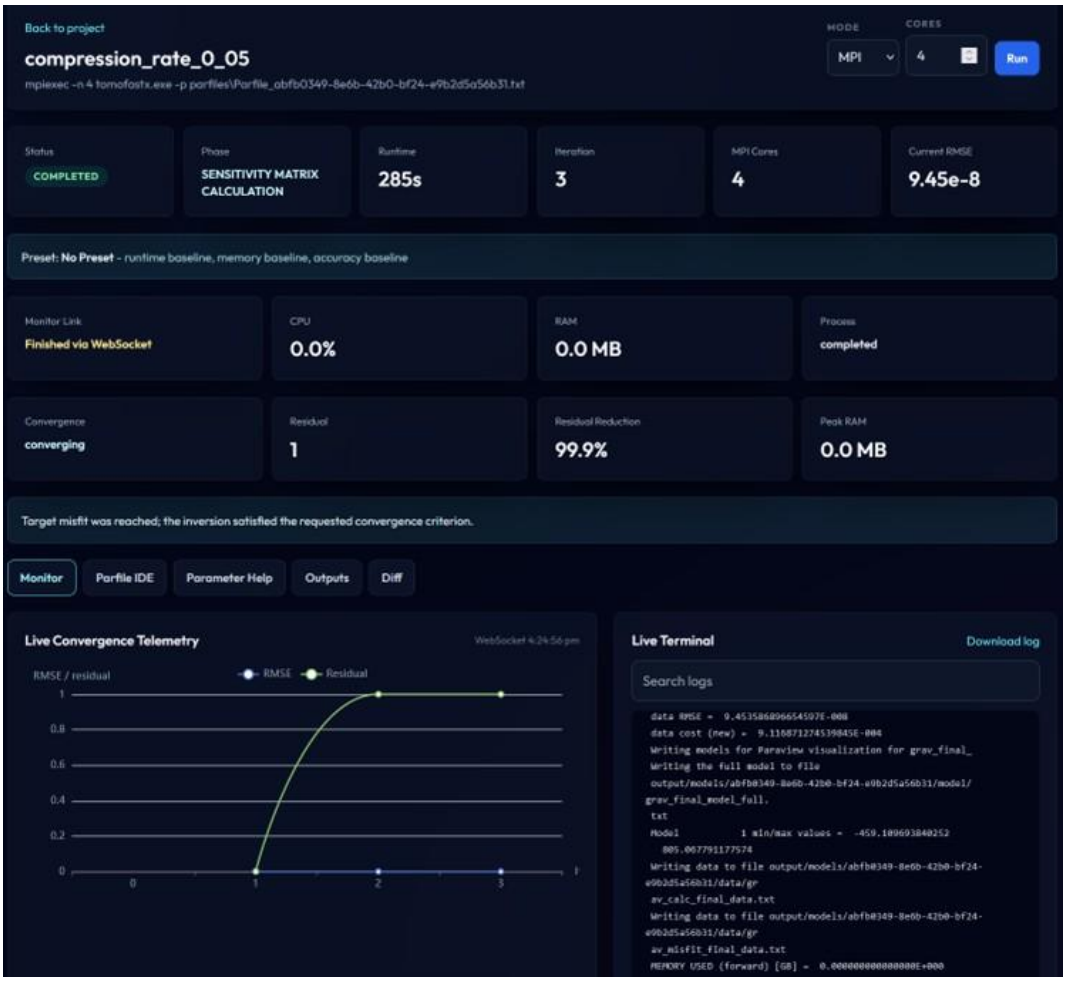

Figure 1. Runtime telemetry and execution-monitoring interface of the proposed workflow platform during a Tomofast-x inversion experiment.

2.4. Experimental Procedure

The chronological procedure of the experiment is shown in Figure 2. The same workflow was applied to all selected scenarios.

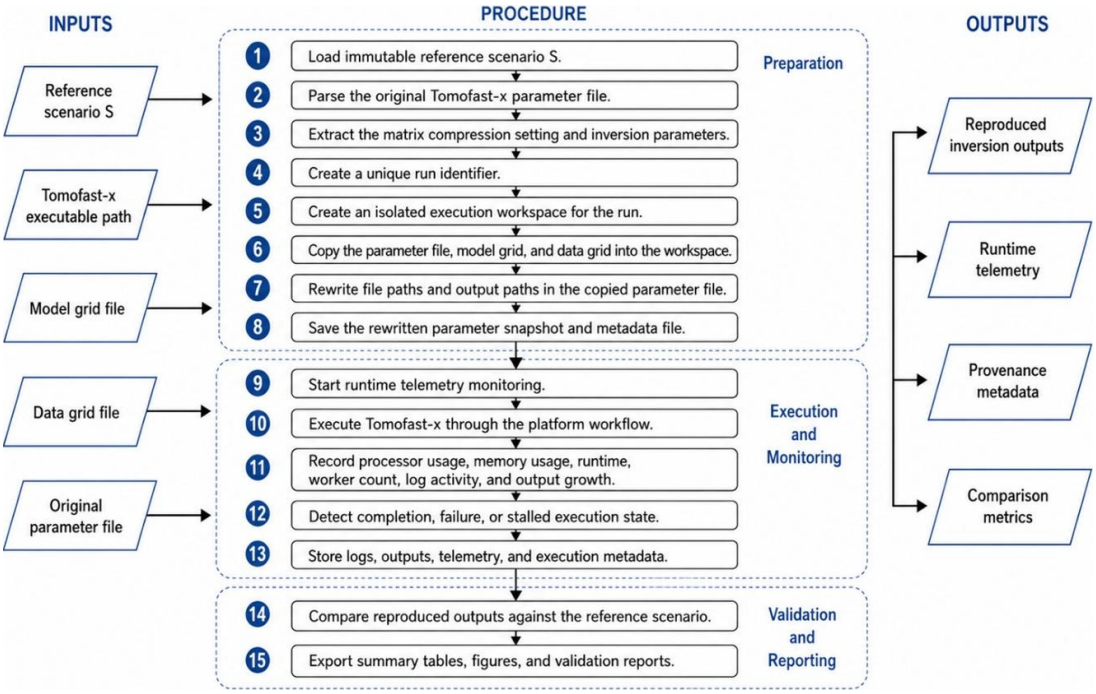

Figure 2. Platform-managed reproduction workflow showing the transformation of reference inputs into reproduced outputs, telemetry records, provenance metadata, and comparison metrics.

This procedure ensured that each reproduced inversion was executed from a clean workspace while preserving the relationship between the original scenario, copied inputs, runtime process, and generated outputs. Similar provenance-aware approaches have been used in computational workflow systems such as AiiDA and Workflow Run RO-Crate, where execution metadata and workflow artifacts are stored to improve traceability and reproducibility [8, 9].

2.5. Repeated Reproduction Experiment

Each of the four selected scenarios was reproduced three times through the platform, resulting in 12 platform-managed Tomofast-x runs. All runs used cloned workspaces, rewritten parameter files, isolated outputs, and telemetry logging. The repeated-run design was used to evaluate workflow reproducibility behavior and runtime variability across repeated platform-managed executions. For each run, the following metrics were collected:

1. Execution status
2. Runtime in seconds
3. Platform-observed peak RAM usage
4. Average processor utilization
5. MPI worker count
6. Number of telemetry samples
7. Output folder size
8. Reproduced root mean square error
9. Absolute root mean square error difference relative to the reference
10. Completion and target-misfit status

The repeated experiment results were summarized using the mean and standard deviation of runtime, platform-observed peak RAM usage, processor utilization, and telemetry sample count. The comparison module then evaluated reproduced outputs against the corresponding reference scenario using root mean square error, data cost, output metadata, and visual model inspection, consistent with provenance-oriented reproducibility analysis approaches [18]. The reproducibility error for each scenario was calculated as:

$$E_{RMSE} = \left| RMSE_{reproduced} - RMSE_{reference} \right| \quad (1)$$

2.6. Telemetry and Observability Metrics

Runtime observability was evaluated by recording telemetry signals throughout each Tomofast-x execution. The platform collected processor utilization, memory usage, runtime, MPI worker count, output-folder growth, and log activity. These measurements were used to determine whether the inversion process was active, stalled, completed, or failed.

The telemetry sample count was used as an operational indicator of monitoring coverage. For each run, the telemetry record was stored together with the run metadata and output path. Observability was treated as a workflow-level property rather than as a geophysical result. This distinction is important because telemetry does not change the inversion result, but it improves the ability to diagnose runtime behavior, memory demand, and execution failures. The memory reduction ratio between the uncompressed baseline and compressed scenarios was calculated as:

$$R_{RAM} = \frac{M_{baseline}}{M_{scenario}} \tag{2}$$

where $M_{baseline}$ is the mean platform-observed peak RAM usage of the uncompressed baseline scenario and $M_{scenario}$ is the mean observed peak RAM usage of the selected compressed scenario.

2.7. Failure Validation Experiment

In addition to successful reproduction, controlled failure cases were created to evaluate whether the platform preserved diagnostic information when a workflow did not complete normally. Four failure types were tested: corrupted parameter file, missing data grid, invalid execution path, and simulated MPI failure. For each case, the platform recorded execution status, phase, exit code, error category, log path, and active workspace.

Table 2. Failure validation cases.

Failure type	Expected behavior	Recorded evidence
Corrupted parameter file	Runtime or parser should reject malformed configuration	status, phase, log path, active workspace
Missing data grid	Execution should fail during data allocation	status, exit code, phase, log path
Invalid paths	Execution should be stopped or classified as stalled	status, phase, runtime, log path
MPI failure	Launcher failure should be logged and preserved	status, exit code, error category, log path

Failure traceability was calculated as:

$$T_f = \frac{N_{traceable}}{N_{failure}} \tag{3}$$

where $N_{traceable}$ is the number of injected failure cases that preserved diagnostic metadata and $N_{failure}$ is the total number of injected failure cases. This metric was used to evaluate whether the platform supported post-mortem workflow analysis.

2.8. Structural Output Inspection

The platform also produced minimal visual inspection outputs from volumetric model files. Each selected scenario contained a model.vtu file with 417720 cells and 438991 points. The platform extracted scalar model statistics, including minimum, maximum, mean, and standard deviation. It also generated two-dimensional slice previews and a model-difference preview between the aggressive compression scenario and the uncompressed baseline.

The structural model difference was computed as:

$$D_i = m_i^{scenario} - m_i^{baseline} \tag{4}$$

where $m_i^{scenario}$ is the scalar value of cell i in a selected scenario and $m_i^{baseline}$ is the corresponding scalar value in the baseline model. Summary statistics of the difference field were then used to describe structural variation between scenarios. This step was intended as a basic output-inspection function rather than a full three-dimensional visualization system.

2.9. Research Validation Summary

The research design evaluated the platform from four perspectives: numerical reproduction, runtime observability, provenance preservation, and workflow failure traceability. Numerical reproduction was evaluated by comparing reproduced root mean square error values with reference outputs. Runtime observability was evaluated from telemetry signals. Provenance preservation was evaluated through isolated workspaces, parameter snapshots, log paths, and metadata. Failure traceability was evaluated through controlled failure injection. Together, these tests were used to determine whether the platform could support reproducible and observable workstation-scale Tomofast-x gravity inversion workflows.

3. RESULTS AND DISCUSSIONS

3.1. Reproduction Fidelity of Reference Scenarios

The repeated reproduction results are summarized in Table 3. The mean absolute root mean square error difference between the platform-reproduced output and the corresponding reference output ranged from 2.47×10^{-11} to 4.72×10^{-9}. These values indicate that the platform could reproduce the selected reference workflows with very small numerical differences.

Table 3. Runtime and memory observability across reference scenarios.

Scenario	Compression rate	n	Runtime mean (s)	Runtime SD (s)	Peak RAM mean (MB)	R_{RAM}	Peak RAM SD (MB)	Mean $\lvert\Delta RMSE\rvert$
0	1.0	3	435.7	116.7	10945.3	1.0	18.36	3.99×10^{-11}
2.4382	0.01	3	311.3	24.01	246.6	44.4	68.03	2.47×10^{-11}
7.6448	0.00316...	3	340.0	65.00	214.4	51.1	20.89	4.72×10^{-9}
42.166	0.000316...	3	300.3	13.61	202.1	54.2	0.23	2.06×10^{-10}

Runtime telemetry additionally revealed substantial operational differences between compression settings. The uncompressed scenario required substantially larger peak memory allocation compared with compressed scenarios. Mean peak RAM usage decreased from approximately 10945.3 MB in the baseline case to between 202.1 MB and 246.6 MB in the compressed scenarios, corresponding to an approximate memory reduction ratio of 44× – 54× depending on the selected compression rate. Telemetry monitoring continuously recorded processor utilization, runtime progression, and output growth throughout execution. Figure 3 summarizes the reproducibility and runtime observability behavior observed during the repeated reproductions.

The results show that successful output generation alone is insufficient to verify computational reproducibility. Quantitative comparison against the original reference outputs is necessary to confirm that reproduced runs preserve the numerical state of the inversion workflow. Across all reproduced scenarios, the platform produced very small RMSE differences while telemetry measurements showed substantial reductions in platform-observed memory usage for compressed scenarios relative to the uncompressed baseline.

3.2. Runtime and Memory Observability

The telemetry results showed a clear operational difference between the uncompressed baseline scenario and the compressed scenarios. The uncompressed baseline scenario required approximately 10945 MB of peak memory, while the compressed scenarios required approximately 202 – 247 MB. Therefore, the compressed scenarios required approximately 44× – 54× lower peak RAM usage relative to the baseline scenario.

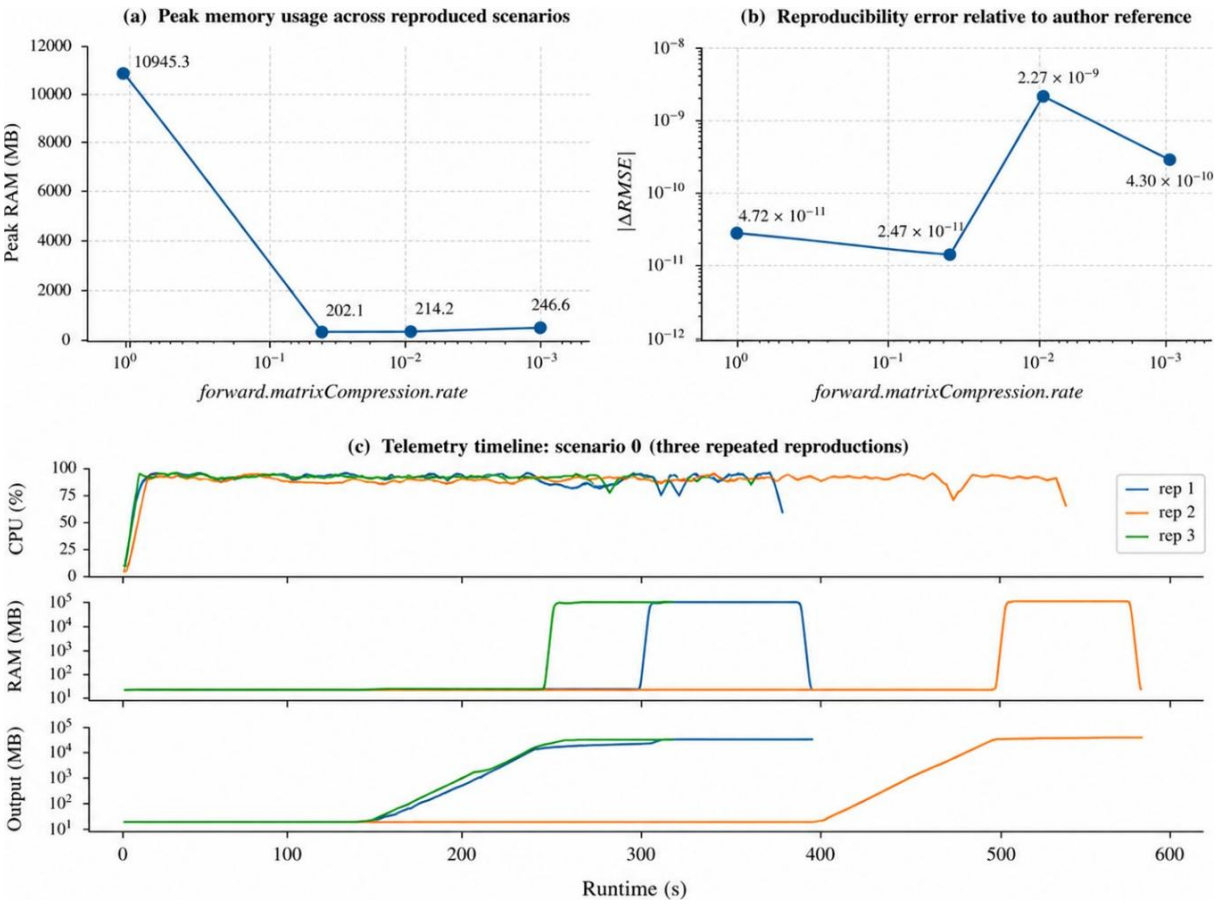

Figure 3. Representative telemetry and observability behavior for the baseline reproduction scenario together with aggregate reproducibility metrics across all scenarios.

The reported memory measurements should be interpreted as platform-observed runtime telemetry collected from the execution environment rather than as exact internal Tomofast-x distributed-memory accounting. The monitoring system aggregates operating-system-level resident set size (RSS) measurements from Tomofast-x and MPI-related processes detected during execution. Although the framework is MPI-aware at the process-discovery level, shared memory pages may still be double-counted depending on operating-system reporting behavior. The present study did not integrate the internal Tomofast-x distributed-memory measurement utility into the telemetry summaries or publication metrics. Therefore, the reported measurements primarily describe workflow-level operational memory behavior observed during reproduced executions.

The memory reduction was larger than the runtime reduction. The baseline scenario required an average runtime of 435.7 s, while the compressed scenarios required 300.3 – 340.0 s. This corresponds to a runtime improvement of approximately 1.28 to 1.45 times. This result indicates that, on the workstation used in this experiment, compression mainly improved memory feasibility rather than producing a proportional runtime reduction.

The runtime standard deviation also showed that workstation-scale execution should be reported statistically. The baseline scenario had a runtime standard deviation of 116.7 s, while scenario 42.166 had a runtime standard deviation of only 13.61 s. This difference indicates that repeated executions can reveal operational variability that would be hidden in a single-run report. Therefore, reporting only one successful execution would be insufficient to describe the behavior of the workflow.

The platform also recorded processor utilization and telemetry sample counts throughout execution. Average processor utilization remained high across all scenarios, ranging from approximately 383.6% to 393.2%. Because utilization was measured across multiple logical CPU cores, values above 100% indicate sustained multi-core CPU utilization during Tomofast-x execution.

The mean telemetry sample count ranged from 44.0 to 63.0 samples per scenario, and telemetry data were successfully collected for every reproduced run. These results support the use of runtime telemetry as an observability layer for monitoring computational inversion workflows.

The telemetry analysis should not be interpreted as a modification of the Tomofast-x inversion algorithm itself. The proposed framework operates at the workflow and execution-monitoring level, where runtime behavior, resource usage, and execution states are observed externally during reproduced inversion runs. The objective of the telemetry layer is therefore to improve workflow observability and execution traceability rather than to modify the numerical inversion methodology implemented within Tomofast-x.

3.3. Provenance Preservation and Output Traceability

The platform preserved provenance information for each reproduced run. Each execution was assigned a unique run identifier and was executed in an isolated active workspace. The copied parameter file, rewritten file paths, output folder, run log, telemetry records, and comparison metrics were stored as part of the run metadata. This design separated immutable reference scenarios from reproduced experiment outputs [7].

The run-level metadata confirmed that every reproduced execution stored the active workspace, output path, log path, compression rate, runtime, platform-observed memory usage, processor usage, output size, root mean square error values, and data cost values. This traceability is important because it allows each result in Table 3 to be linked back to the exact execution context that produced it.

Output size also reflected the effect of compression on generated artifacts. The uncompressed baseline produced an output size of approximately 10774.8 MB, while the compressed scenarios produced approximately 161.1 MB, 87.8 MB, and 57.3 MB. This result shows that workflow telemetry can reveal operational behavior that is not observable from the final inversion error alone.

3.4. Failure Traceability

In addition to successful reproduction, the platform was evaluated using controlled failure cases. Four failure types were tested: corrupted parameter file, missing data grid, invalid path or stalled execution, and simulated MPI failure. The goal was not to make these cases succeed, but to determine whether the platform preserved enough information for post-mortem diagnosis [19].

Table 4. Failure validation results.

Failure type	Status	Execution phase	Exit code	Runtime (s)	Traceability result
Corrupted parameter file	failed	Process terminated	-1	0	Trace preserved
Missing data grid	failed	Data allocation	29	0	Trace preserved
Invalid path or stalled case	stopped	Stalled detection / manual stop	-	243	Trace preserved
MPI failure	failed	Process terminated	-1	3	Trace preserved

The failure validation results showed that all injected failures preserved diagnostic information, including status, phase, log path, and active workspace. The missing data grid case failed during data allocation, which was consistent with the expected behavior. The invalid path case was classified as a stalled or manually stopped execution after 243 s. The MPI failure simulation terminated rapidly and preserved the execution log and provenance context.

All four injected failure cases preserved diagnostic metadata, resulting in a failure traceability ratio $T_f = 1.0$. This indicates that every controlled failure maintained sufficient execution evidence for post-mortem workflow analysis, including execution status, phase, runtime information, log paths, and active workspace references.

A practical robustness issue was also found during validation. On Windows, a MPI failure can return a large unsigned exit code, such as 4294967295. This value can exceed the range of a standard PostgreSQL integer column. The platform was therefore hardened by normalizing large Windows return codes into signed exit-code values before storing them in the database. This fix prevented a secondary database failure from hiding the original execution failure.

3.5. Structural Output Inspection

The platform also generated basic visual inspection outputs from volumetric model files. Each selected scenario contained a model.vtu file with 417720 cells and 438991 points. The platform extracted scalar metadata from each model and generated two-dimensional slice previews. These outputs provided a practical way to inspect the model structure without requiring a full external visualization workflow.

The baseline, 2.4382, and 7.6448 scenarios showed similar large-scale spatial patterns in the slice previews. The 42.166 scenario showed stronger local variations and a wider model-value range. The extracted metadata confirmed this observation. The standard deviation of the scalar model increased from approximately 45.08 in the baseline scenario to approximately 84.32 in the 42.166 scenario. This indicates that the aggressive compression scenario produced a more variable model field.

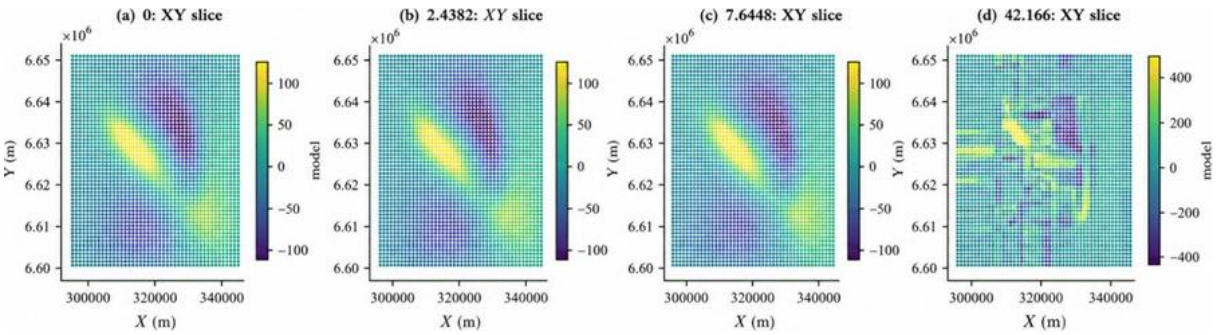

Figure 4. Volumetric model slice previews for the selected reference scenarios.

A model-difference preview was also generated by comparing scenario 42.166 against the baseline scenario. The difference field had a minimum value of approximately -641.0, a maximum value of approximately 814.4, a mean value of approximately 2.39, and a standard deviation of approximately 65.81.

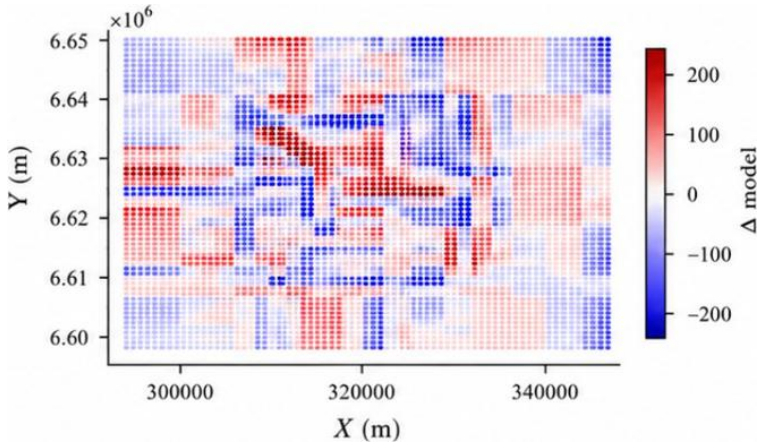

Figure 5. Model-difference preview between scenario 42.166 and the baseline scenario.

The structural-difference statistics derived from $D_i = m_i^{scenario} - m_i^{baseline}$ showed that the mean difference remained relatively small compared with the overall scalar range of the volumetric models, although localized variations became more visible under aggressive compression. The larger standard deviation observed in scenario 42.166 indicates that compression can introduce stronger local structural variability even when the reproduced inversion workflow remains numerically stable in terms of root mean square error. This result shows that the platform can support structural comparison between scenarios, not only numerical comparison of final root mean square error values [20].

The structural visualization results should be interpreted carefully. They do not represent a new inversion method or a new compression theory. Instead, they demonstrate that the platform can produce model-inspection artifacts that help users understand how outputs differ between reference scenarios when comparing compressed inversion outputs across different workflow scenarios.

3.6. Overall Discussion

The results demonstrate that the platform provided value beyond simply launching Tomofast-x. First, the platform reproduced four reference scenarios with small root mean square error differences relative to the reference outputs. Second, telemetry revealed substantial differences in memory use and runtime behavior between scenarios. Third, provenance metadata preserved the relationship between the reference scenario, copied inputs, rewritten parameter file, execution workspace, log file, and output folder [21]. Fourth, controlled failure tests showed that failed workflows could still produce useful diagnostic evidence.

The clearest operational difference was the change in memory behavior between the baseline and compressed scenarios. The uncompressed baseline required approximately 10.9 GB of peak memory, while the compressed scenarios required approximately 202–247 MB. This result supports the importance of runtime observability during practical inversion experiments where memory availability can determine whether a workflow is computationally feasible. However, the runtime improvement was smaller than the memory improvement, which shows that memory reduction and runtime reduction should be reported separately.

The repeated-run results also showed that runtime should not be treated as a fixed property of a scenario. Even when the same workflow was reproduced, runtime varied across repeated executions [22]. Therefore, repeated platform-managed runs provide a more informative operational picture than a single successful run. This supports the broader goal of the platform: not only to reproduce inversion outputs, but also to make the execution behavior observable and traceable through the implementation of reproducible computational steps in geo-information sciences [23].

The failure-validation results further show that workflow robustness is not only measured by successful execution. In real computational research, corrupted inputs, missing files, path errors, and launcher failures can occur [24]. The platform achieved this by storing failure status, execution phase, log path, workspace path, and error categories as part of the workflow provenance metadata.

The current validation scope should also be interpreted carefully. The experiments were performed only on a Windows workstation environment using the smallest publicly available reference model group from the Zenodo dataset. Linux deployment, HPC-scale execution, and larger inversion models were not evaluated in the present study, and cross-platform compatibility was therefore not validated. In addition, the current workflow evaluation relied on archived reference scenarios and reference outputs for reproducibility comparison. Therefore, the reported results should be interpreted as workstation-scale reproducibility and observability validation rather than as a general benchmark of Tomofast-x performance across heterogeneous computing infrastructures.

Overall, the results support the proposed framework as an initial reproducibility- and observability-oriented experimentation layer for workstation-scale Tomofast-x workflows. The framework was designed to support repeated execution, telemetry monitoring, provenance preservation, failure traceability, and structured comparison against archived reference scenarios rather than to replace general-purpose scientific workflow orchestration systems. Tomofast-x provides the inversion engine itself, while the proposed framework focuses on workflow-level execution monitoring and reproducibility validation inspired by reproducible computational research and workflow-traceability practices [3, 4]. The current study focused on workflow reproducibility and observability evaluation rather than on public release of a generalized workflow software package.

4. CONCLUSION

This study presented a reproducibility- and observability-oriented experimentation framework for Tomofast-x gravity inversion workflows. The platform was evaluated using four publicly available Tomofast-x reference scenarios. Repeated platform-managed reproductions demonstrated that the proposed workflow could reproduce the reference inversion outputs with small numerical differences while simultaneously preserving execution provenance, telemetry records, and workflow metadata.

The results showed that the platform successfully supported reproducible workstation-scale inversion experiments through isolated execution workspaces, automated parameter management, runtime monitoring, and structured result collection. Runtime telemetry revealed substantial operational differences between compression scenarios, particularly in platform-observed memory usage, where the uncompressed baseline required approximately 10.9 GB RAM while compressed scenarios required approximately 202–247 MB. In addition, the repeated executions demonstrated that runtime variability should be evaluated statistically rather than from a single execution result.

Controlled failure experiments further demonstrated that the platform preserved execution traces, logs, provenance metadata, and failure diagnostics under corrupted parameter files, missing data grids, invalid execution paths, and simulated MPI failures. The validation campaign also resulted in a practical robustness improvement through normalization of Windows MPI exit codes for reliable database persistence.

The generated VTU slice previews and model-difference visualizations demonstrated that the platform can produce structured visualization outputs suitable for comparative inspection and reporting in addition to numerical validation results. Overall, the study showed that the proposed environment can support reproducible, traceable, and observable Tomofast-x experimentation workflows suitable for research validation and computational geophysics studies.

Future work may include integration with distributed HPC infrastructures, interactive three-dimensional visualization, automated inversion benchmarking, uncertainty quantification, and multi-user collaborative provenance systems for large-scale geophysical inversion campaigns.

ACKNOWLEDGMENTS

The author would like to express sincere gratitude to Dr. Vitaliy Ogarko for his valuable comments, insightful discussions, and technical suggestions during the revision of this work, particularly regarding workflow scope, memory observability, and reproducibility considerations in Tomofast-x gravity inversion experiments.

REFERENCES

[1] Giraud, J., Ogarko, V., Martin, R., Jessell, M., & Lindsay, M. (2021). Structural, petrophysical, and geological constraints in potential field inversion using the Tomofast-x v1. 0 open-source code. *Geoscientific Model Development*, **14**(11), 6681–6709.

[2] Ogarko, V., Frankcombe, K., Liu, T., Giraud, J., Martin, R., & Jessell, M. (2024). Tomofast-x 2.0: an open-source parallel code for inversion of potential field data with topography using wavelet compression. *Geoscientific Model Development*, **17**(6), 2325–2345.

[3] Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLoS Computational Biology*, **9**(10), e1003285.

[4] National Academies of Sciences, Medicine, Policy, Global Affairs, Board on Research Data, Information, Division on Engineering, Physical Sciences, Committee on Applied, Theoretical Statistics, Board on Mathematical Sciences & Replicability in Science. (2019). *Reproducibility and replicability in science*. National Academies Press.

[5] Boettiger, C. (2015). An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, **49**(1), 71–79.

[6] Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., Da Silva, R. F., Livny, M., & Wenger, K. (2015). Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, **46**, 17–35.

[7] Groth, P. & Moreau, L. (2013). PROV-overview: An overview of the PROV family of documents. *World Wide Web Consortium (W3C)*.

[8] Huber, S. P., Zoupanos, S., Uhrin, M., Talirz, L., Kahle, L., Häuselmann, R., Gresch, D., Müller, T., Yakutovich, A. V., Andersen, C. W., Ramirez, F. F., Adorf, C. S., Gargiulo, F., Kumbhar, S., Passaro, E., Johnston, C., Merkys, A., Cepellotti, A., Mounet, N., Marzari, N., Kozinsky, B., & Pizzi, G. (2020). AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance. *Scientific Data*, **7**(1), 300.

[9] Leo, S., Crusoe, M. R., Rodríguez-Navas, L., Sirvent, R., Kanitz, A., De Geest, P., Wittner, R., Pireddu, L., Garijo, D., Fernández, J. M., Colonnelli, I., Gallo, M., Ohta, T., Suetake, H., Capella-Gutierrez, S., de Wit, R., Kinoshita, B. P., & Soiland-Reyes, S. (2024). Recording provenance of workflow runs with RO-Crate. *PLoS One*, **19**(9), e0309210.

[10] Katz, D. S., Hong, N. P. C., Clark, T., Muench, A., Stall, S., Bouquin, D., Cannon, M., Edmunds, S., Faez, T., Feeney, P., Fenner, M., Friedman, M., Grenier, G., Harrison, M., Heber, J., Leary, A., MacCallum, C., Murray, H., Pastrana, E., Perry, K., Schuster, D., Stockhause, M., & Yeston, J. (2021). Recognizing the value of software: a software citation guide. *F1000Research*, **9**, 1257.

[11] Nica, R., Götz, S., & Moltó, G. (2024). CMK: Enhancing Resource Usage Monitoring across Diverse Bioinformatics Workflow Management Systems: R. Nica et al. *Journal of Grid Computing*, **22**(3), 62.

[12] Balis, B., Czerepak, K., Kuzma, A., Meizner, J., & Wronski, L. (2024). Towards observability of scientific applications. *arXiv preprint arXiv:2408.15439*.

[13] Uieda, L., Soler, S. R., Pesce, A., Oliveira Jr, V. C., & Shea, N. (2020). Harmonica: forward modeling, inversion, and processing gravity and magnetic data. *J. Open Source Softw.*, **5**(51), 2574.

[14] Bruce, E., Ogarko, V., Giraud, J., & Jessell, M. (2025). Gravity Inversions Wavelet compression dataset. *Zenodo*, 17786745.

[15] Bruce, E. M., Ogarko, V., Giraud, J., & Jessell, M. W. (2026). Wavelet Compression for Gravity Inversion: Optimising and Predicting Memory Efficiency in 3D Geophysical Modelling. *SSRN*, 6262158.

[16] Goble, C., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M. R., Peters, K., & Schober, D. (2020). FAIR computational workflows. *Data Intelligence*, **2**(1-2), 108–121.

[17] Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J. W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J.,'t Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., & Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, **3**(1), 1–9.

[18] Missier, P., Woodman, S., Hiden, H., & Watson, P. (2016). Provenance and data differencing for workflow reproducibility analysis. *Concurrency and Computation: Practice and Experience*, **28**(4), 995–1015.

[19] Zhang, R., Zhou, J., Hai, T., Zhang, S., Iwendi, M., Biamba, C., & Anumbe, N. (2022). Quality assurance awareness in higher education in China: big data challenges. *Journal of Cloud Computing*, **11**(1), 56.

[20] Garia, S., Pal, A. K., Katre, S., Nayak, S., Ravi, K., & Nair, A. M. (2023). Mapping petrophysical properties with seismic inversion constrained by laboratory based rock physics model. *Earth Science Informatics*, **16**(4), 3191–3207.

[21] Suetake, H., Fukusato, T., Igarashi, T., & Ohta, T. (2023). A workflow reproducibility scale for automatic validation of biological interpretation results. *Gigascience*, **12**, giad031.

[22] Ma, C., Morrison, S. M., Muscente, A. D., Wang, C., & Ma, X. (2023). Incorporate temporal topology in a deep-time knowledge base to facilitate data-driven discovery in geoscience. *Geoscience Data Journal*, **10**(4), 489-499.

[23] Roldán-Gómez, J., Boubeta-Puig, J., Pachacama-Castillo, G., Ortiz, G., & Martínez, J. L. (2021). Detecting security attacks in cyber-physical systems: a comparison of Mule and WSO2 intelligent IoT architectures. *PeerJ Computer Science*, **7**, e787.

[24] Ince, D. C., Hatton, L., & Graham-Cumming, J. (2012). The case for open computer programs. *Nature*, **482**(7386), 485–488.