Response to Reviewer #1:

We sincerely thank the reviewer for the exceptionally thorough and rigorous evaluation of our manuscript. The detailed, point-by-point comments have substantially improved the quality of this work, and we are genuinely grateful for the time and expertise invested in this review.

Major revisions overview:

We have made several fundamental changes to the experimental framework in response to the reviewer's concerns about experimental scope and generalizability:

(1) Removal of the 9 km grid experiments. In the original submission, the 9 km grid served only as a supplementary test for computational efficiency analysis, while all core experiments—idealized tests, initial field scatter analysis, and forecast evaluation—were conducted exclusively at 27 km resolution. Given that the 9 km experiments were peripheral to the main findings, we removed them entirely in the revision. This allowed us to focus computational resources on substantially expanding the 27 km experiments, as detailed below.

(2) Extension from a single case to 20 consecutive days. The real-case assimilation experiment has been expanded from a single time step (00:00 UTC, October 27, 2023) to 20 consecutive daily cycles (October 10–29, 2023), each followed by a 24-hour forecast. This extension covers diverse meteorological and pollution conditions, enabling robust statistical evaluation of both assimilation accuracy and forecast skill.

(3) Addition of multi-core CPU scalability experiments. To address the reviewer's question about the performance bottleneck and parallelization efficiency, we conducted systematic scalability tests of both PY3DVar and FT3DVar using 1, 2, 4, 8, 16, 32, and 64 CPU cores across all 20 assimilation cases. This experiment provides a direct comparison of how the two systems scale with increasing parallel resources.

(4) Adoption of an independent validation strategy. The quality-controlled observations are now divided into assimilation and validation subsets: the gridded and thinned

observations (averaging 546 stations, 28%) are used for assimilation, while the remaining observations (averaging 1,402 stations, 72%) are reserved exclusively for independent validation. All evaluation metrics are computed on the validation dataset, ensuring an unbiased assessment.

(5) Platform change: DCU to GPU, Intel to AMD. The DCU platform has been replaced by an NVIDIA RTX 3090 GPU (24 GB), which is a more widely accessible and familiar computing platform for the international readership. Similarly, the CPU platform has been updated from the originally listed Intel processor to a dual AMD EPYC 7542 configuration (64 physical cores @ 2.9 GHz, 256 GB memory), with the correct processor model now clearly documented.

(6) Substantial restructuring of the Introduction and Methods sections. The Introduction has been restructured to provide a more balanced and internationally representative literature review, with expanded coverage of recent advances. The Methods section has been reorganized, with a new dedicated section (Section 2.7) describing both assimilation systems and their implementation differences. All figures have been refined for clarity, and the conclusions have been revised to accurately reflect the scope of the study.

We address each specific comment in detail below:

> Comment:

First, the numerical experiments are limited in scope. The first, idealized experiment could serve as as verification that the two implementations indeed produce numerically equivalent results, but the setting is probably too unrealistic to give useful insights to the computational performance. I don’t see it deserving as much weight in the results as it is currently given. For example, having only two assimilated data points is likely to make the optimizers convergence much faster than they would in a realistic situation. The second experiment is more realistically set up, but still only involves a single analysis step. I don’t think that it would be prohibitively difficult to run multiple assimilations to check that the findings continue to hold as the chemical and physical atmospheric conditions change.

**** Response:****

Thank you for your careful comments and constructive critique. We fully understand your concerns regarding the experimental design, and we provide detailed explanations and corresponding revisions as follows.

First, regarding the idealized twin-point experiment. Single or multiple point observation experiments are a standard and widely adopted paradigm in atmospheric data assimilation studies. This type of controlled idealized test is not intended to evaluate computational performance under fully realistic operational conditions; instead, it acts as a fundamental verification step to preliminarily validate the numerical correctness and physical rationality of a newly developed assimilation system. With preset low and high concentration values at two independent sites, we can intuitively examine whether the analysis increments conform to the characteristics of background error correlation, and quantitatively confirm that the PY3DVar and traditional FT3DVar yield nearly identical analysis results. This step is essential for excluding fundamental numerical bugs before conducting complex real-case experiments.

We acknowledge that the simplified configuration with only two observation points does lead to faster optimizer convergence compared with realistic scenarios with dense observations. For this reason, we have substantially shortened the description and discussion of the idealized experiment in the revised manuscript, reducing its proportion in the overall results (Section 4.1). As suggested, we do not overemphasize its role in assessing computational efficiency, and we clearly clarify that this part only focuses on verifying the numerical consistency between the two assimilation frameworks.

Second, for real-case assimilation experiments. Following your valuable suggestion, we have extended the experimental period significantly. Instead of a single analysis step, we carried out 20 consecutive daily assimilation and forecast cycles covering varying atmospheric and chemical conditions. Over this long time series, we comprehensively evaluated the assimilation quality(Section 4.2), numerical stability, parallel scalability(Section 4.4) and computational efficiency (Section 4.3, 4.4) of both PY3DVar and FT3DVar. The multi-cycle tests under continuously changing environmental conditions fully prove that the conclusions drawn in this study are robust and not limited to a single momentary case.

In summary, the idealized experiment serves only for basic numerical verification, while the extended 20-day real-case experiments constitute the core evidence for our research conclusions. We have adjusted the content structure and length accordingly to distinguish the positioning of different experiments, and all revisions have been marked in the revised manuscript.

> Comment:

Only assimilation of in-situ data is considered. Although in-situ data are very commonly used in practice, this is a limitation from the computational point of view. Especially with the narrow correlation distances used here, using only point observations results in a very well-behaved minimization problem. Also the process of developing the adjoint observation operator becomes very simple in this case, which reduces the advantage of the built-in automatic differentiation in the numerical library. The assimilation experiments should have used a split between assimilation and test data, since evaluating the analysis against the assimilated data is largely uninteresting.

****Response:****

We thank the reviewer for this insightful comment. We address each point below.

(1) Separation of assimilation and test data. As recommended, the quality-controlled observations have been divided into two independent subsets: the gridded and thinned observations, averaging 546 stations (28%), are used for assimilation, while the remaining observations, averaging 1,402 stations (72%), are reserved exclusively for independent validation. This setup avoids evaluating analyses against the assimilated data, and the detailed description has been added to the revised manuscript (Line 228-232).

(2) Limitation of using only in-situ observations. We acknowledge that the exclusive use of surface in-situ observations is a limitation of the present study. This choice was made because surface monitoring stations are the most widely available operational observations for air quality forecasting in China, providing a practical and representative support for validating a new assimilation framework. The primary objective of this work is to establish the computational efficiency and numerical accuracy of the PyTorch-based 3DVar

implementation relative to the traditional Fortran system. We plan to incorporate remote sensing observations with more complex observation operators in future studies, where the benefits of automatic differentiation would be more fully realized.

(3) Well-behaved minimization and simplified adjoint. We agree that point in-situ observations, combined with relatively narrow background error correlation distances, lead to a well-conditioned minimization problem and simplify the construction of the adjoint operator. However, the advantage of automatic differentiation (AD) extends well beyond simplifying complex observation operators. In traditional Fortran-based 3DVar systems, the entire adjoint code—covering the observation operator, background error covariance transformations, and the cost function gradient—must be manually derived and implemented, which is time-consuming, error-prone, and difficult to maintain. The PyTorch-based framework replaces all of this manual coding with automatically generated derivatives, an advantage that is fully retained even for in-situ observations. Furthermore, the central finding of this study is that AD and GPU acceleration can dramatically improve computational efficiency while maintaining assimilation accuracy equivalent to the mature Fortran system, as demonstrated in the extensive parallel and GPU benchmarks.

> Comment:

The performance analysis lacks depth. The computations are mainly compared in terms of wall clock time. While I am not surprised that Py3DVAR running DCUs outperforms the CPU versions, I am somewhat surprised by the big performance gap between the CPU-running PyTorch and Fortran versions. The PyTorch library is no doubt very well optimized, but is also possible that the Fortran version is suboptimal in some way. While it's true that manual code optimization can be tedious compared to simply using a package like PyTorch, it would be useful to analyze where the bottleneck is (for example, is it the single-core performance, parallelization of the linear algebra, or something else).

****Response:****

We thank the reviewer for this comment, which prompted us to examine the performance gap more carefully. To address this, we have revised Section 4.4 to present a comprehensive analysis of the per-iteration and total iteration times across various CPU core counts.

Critically, FT3DVar is consistently faster than PY3DVar on a per-iteration basis across all core counts—decreasing from 4.09 s to 0.91 s, compared to PY3DVar's 59.70 s to 2.13 s. This confirms that the Fortran implementation is not suboptimal; rather, its compiled nature gives it an inherent per-iteration speed advantage.

The overall performance advantage of PY3DVar reported in the original manuscript arises from two compensating factors: (1) PY3DVar converges in far fewer iterations (mean of 7.5 vs. 24.9; Fig. 12a), owing to the more aggressive Strong Wolfe line search in PyTorch's L-BFGS optimizer; and (2) PY3DVar exhibits stronger parallel scaling as core counts increase. Consequently, FT3DVar achieves shorter total assimilation time at 16 cores and below, while PY3DVar overtakes it at 32 and 64 cores.

We have corrected the misleading original statement that implied PY3DVar was universally faster, and the revised text now accurately acknowledges FT3DVar's per-iteration superiority. We appreciate the reviewer's careful reading, which helped us avoid this misrepresentation.

> Comment:

Finally, some aspects of the experiments, particularly the Fortran implementation of 3D-Var, are insufficiently described and referenced. As far as I understand, WRF-Chem includes a data assimilation component, but it’s unclear to me whether it is being used here. Again, this makes it difficult for the reader to understand whether the performance gains demonstrated here would transfer to other assimilation systems.

****Response:****

We sincerely thank the reviewer for this important comment, which prompted us to clarify the nature of the Fortran implementation and its relationship to existing systems.

In the revised manuscript, we have added a dedicated section (Section 2.5) that describes both assimilation systems in detail. The key clarifications are as follows:

First, FT3DVar is explicitly distinguished from the official WRFDA-Chem module. FT3DVar is a standalone Fortran 3DVar system independently developed by our research group and validated in several previous studies over China (Zang et al., 2016; Liang et al.,

2020; Wang et al., 2020). It is entirely independent of the WRFDA-Chem system distributed with the official WRF release (Line 310-311).

Second, Section 2.5 provides a complete technical description of FT3DVar, including its Fortran + OpenMP CPU-only architecture, manually derived analytical gradient (Eq. 4), and the customized L-BFGS implementation. Both systems share identical variational theory, control variable design, NMC-based background error covariance modeling, and observation quality control procedures. The differences between the two systems are confined to the programming framework and parallelization strategy, ensuring a fair comparison.

Third, regarding transferability: the computational speedup of PY3DVar stems from expressing all variational calculations as tensorized matrix operations, a strategy that is applicable to any 3DVar system whose core computations can be formulated as tensor operations. The performance gains demonstrated here are therefore not specific to our particular Fortran implementation, but reflect the general efficiency of tensor-based computation with automatic differentiation and GPU acceleration.

> Comment:

In summary, in my view, the manuscript makes a valid point on the potential of PyTorch and similar libraries in implementing variational (or other) practical data assimilation algorithms. However, the experimental results are rather minimal, and I strongly recommend expanding the experiments to cover variable conditions, and to give more details on how the methods perform in a real-world assimilation setup. The manuscript could also use improvements in scope of the introduction and in use of citations. Some conclusions should be revised to better match the extent of the study.

****Response:****

We sincerely thank the reviewer for the constructive summary and for recognizing the potential of this work. We have thoroughly addressed each point raised:

1. **Expanded experiments**: The real-case assimilation has been extended from a single day to 20 consecutive daily cycles (October 10–29, 2023), covering diverse meteorological

and pollution conditions. A multi-core CPU scalability experiment (1–64 cores) has also been added across all 20 cases.

2. **Real-world assimilation details**: A dedicated section (Section 2.5) has been added describing both assimilation systems, their implementation differences, and their identical parameter settings and convergence criteria.

3. **Introduction and citations**: The Introduction has been substantially restructured and streamlined. The references Cheng et al. (2025) and Key et al. (2024) have been added as suggested, and redundant discussions on remote sensing and AI assimilation have been condensed.

4. **Conclusions**: Overstated claims have been removed. The conclusions now focus strictly on the demonstrated findings: improved computational efficiency with maintained assimilation accuracy.

We believe these revisions have significantly strengthened the manuscript, and we are grateful for the reviewer's guidance throughout the review process.

Specific comments

> Comment:

Introduction: Much of the literature and topics reviewed here are tangential to the work that is being reported. For example, a paragraph (L104-L132) is devoted for discussing the use of remote sensing data in chemical data assimilation, which is not very relevant for the current study that only uses in-site observations. Also the discussion of various AI and hybrid assimilation approaches (L147-L196) appears unnecessarily extensive, since the current study is based on a classical variational method. A seemingly relevant reference that is not mentioned is Cheng et al (2025) who report implementing multiple assimilation algorithms including 3D-Var using Pytorch. Some useful connections might be also found in the

Gaussian Process literature, as done by Key et al (2024), given the similarities between 3D-Var and Gaussian process regression.

****Response:****

Thank you very much for the valuable comments and suggestions. We have thoroughly revised the Introduction section accordingly, and the detailed modifications are summarized as follows:

1. The paragraph regarding remote sensing data assimilation has been substantially condensed in the revised manuscript (Line 110-117). We no longer elaborate on technical details and experimental results of remote sensing-based assimilation. Instead, we only briefly introduce the general research trend of multi-source observations in aerosol assimilation, and explicitly clarify that the present study solely adopts in-situ surface observations, to avoid irrelevant content.
2. The discussion on AI and hybrid assimilation approaches has been greatly shortened and restructured in the revised manuscript (Lines 120–134). We have removed redundant descriptions of diverse neural network models and end-to-end assimilation frameworks. We now simply outline the advantages and limitations of data-driven assimilation methods, and clearly emphasize that our work strictly follows the classical 3DVar theoretical framework, rather than adopting AI-based assimilation schemes.
3. As suggested, we have added the reference of Cheng et al. (2025) in the revised text (Line 125-130). We introduce their work on implementing 3D-Var and other assimilation algorithms via PyTorch, which is highly relevant to our framework construction.
4. We have also cited Key et al. (2024) and briefly noted the inherent mathematical similarities between 3D-Var and Gaussian process regression, establishing the methodological connection as recommended (Line 103-107).

> Comment:

L37: “Errors in their initial conditions can accumulate during model integration”. This happens in some systems. But does it really happen in air quality models? In the forecast experiments in this study the initial conditions seemed to have a rather transient (6-24 h) effect on the forecasts, and similar results have been shown for many other models as well.

Response:

Thank you very much for this insightful comment. We fully acknowledge that unlike weather and climate models, errors in initial conditions of air quality models do not keep accumulating throughout the entire integration period, and their impacts are generally transient and concentrated in the early simulation stage (roughly 6–24 hours).

Accordingly, we have revised the original sentence. The revised content reads: Errors in initial conditions can degrade model performance, particularly within the early hours of simulation, leading to significant biases in short-term forecasts. This revision removes the inappropriate description of continuous error accumulation, and accurately reflects the limited-duration impact of initial condition errors on air quality forecasts, which is consistent with both our experimental results and widely recognized findings in relevant studies (Line 36-38).

> Comment:

L77: Please define what 3D-VAR stands for and add appropriate references for the method.

Response:

We have now given the full name and a concise definition of 3DVar at its first occurrence, and added two classic foundational references (Lorenc, 1986; Parrish and Derber, 1992) for this method, as requested (Line 76-78).

> Comment:

L82: Please give references for the operational systems. Many operational systems have also switched from 3D-Var to more advanced methods.

Response:

We fully recognize that major global operational forecasting systems have shifted from 3DVar to more advanced assimilation techniques including 4DVar and hybrid

ensemble-variational methods. Accordingly, we removed the inappropriate statement about 3DVar being used in global operational systems. Instead, we clarified that 3DVar is still widely applied in regional air quality and atmospheric chemistry modeling, owing to its simple implementation and low computational burden compared with advanced schemes (Line 78-83).

> Comment:

L120: what is a “collaborative assimilation scheme”?

****Response:****

As you commented earlier, the discussion regarding remote sensing data assimilation (Lines 104–132) is tangential to this study, since our work only uses in-situ surface observations. Following that suggestion, we have substantially condensed this entire section and removed the original sentence containing the phrase "collaborative assimilation scheme".

The original text intended to describe the simultaneous assimilation of CALIPSO extinction coefficient profiles and surface PM observations—"collaborative" was used to mean "joint" or "simultaneous" rather than implying any specific technical scheme. The revised version no longer contains this ambiguous phrasing.

> Comment:

L120: Ye et al. 2020 is not listed (but 2021 and 2022 are)

> Comment:

L129: Year missing from Wang et al citation

****Response to comments on L120 and L129:****

As noted in our earlier response, the discussion of remote sensing data assimilation has been substantially condensed in the revised manuscript. The missing citation for Ye et al. (2020)

has been corrected to Ye et al. (2021, Line 117) and (2022, Line 115), and the incomplete Wang et al. citation now includes the correct year (Wang et al., 2022, Line 115). We have carefully verified all remaining citations for completeness and accuracy.

> Comment:

L142 Please define the “DCU“ and briefly if there are significant differences to the GPU hardware that some readers might more familiar with.

****Response:****

Thank you for this helpful comment. In the revised manuscript, the term "DCU" has been removed entirely, as the present study now uses only NVIDIA GPU for acceleration rather than DCU. For the reviewer's reference, DCU (Deep Computing Unit) is a general-purpose accelerator architecture that is functionally comparable to GPU, both employing massively parallel computing paradigms with thousands of cores. PyTorch supports both GPU and DCU backends without code modification, meaning the framework and performance conclusions reported in this study are directly transferable to DCU platforms. We have clarified this point in the manuscript and have carefully replaced all DCU-related terms with GPU throughout the text for consistency.

> Comment:

L212: "principle inheritance" and "implementation innovation". Are these established concepts? If not, please consider whether they are really helping the reader.

****Response:****

We thank the reviewer for this comment. The phrase "principle inheritance" and "implementation innovation" has been removed, as it is not an established concept and does not aid reader comprehension. The intended meaning—retaining the physical principles of 3DVar while modernizing the numerical implementation—is now conveyed directly in plain language. Specifically, the revised text states that the proposed framework "inherits the

complete classical 3DVar variational optimization theory" while "benefiting from native PyTorch automatic differentiation and hardware-accelerated tensor parallel computing," which expresses the same idea more clearly without the artificial terminology. We appreciate the reviewer's suggestion, which has improved the readability of the manuscript.

> Comment:

L228: Please provide some overall references for the WRF-Chem.

****Response:****

Thank you for this comment. We have added two key overall references for WRF-Chem at the first mention of the model in the revised manuscript: the original model development paper (Grell et al., 2005) that describes the fundamental online-coupled framework, and a classic application paper (Fast et al., 2006) that demonstrates the model's capability in simulating meteorology–chemistry–aerosol interactions. These references provide essential background information on the WRF-Chem model design and applications. All additions are marked in the revised manuscript (Line 160).

> Comment:

L239-243. Please provide appropriate references for the chemical mechanisms and aerosol models. I don't they are those that are given here. Which of the gas phase mechanisms did you use?

****Response:****

We thank the reviewer for pointing out this inaccuracy. The WRF-Chem configuration used in this study employs the SAPRC-99 gas-phase mechanism, not CB-IV or RADM2. The references have been corrected to Carter (2000) for the SAPRC-99 mechanism and Li et al. (2010, 2011a, 2011b, 2012) for the specific model configuration (Line 171). The incorrect references to CB-IV and RADM2 have been removed. We apologize for this oversight.

> Comment:

L244: which model version?

****Response:****

Thank you for the reminder. We have now clearly specified that WRF-Chem v3.5 in this study (Line 169).

> Comment:

L286: I guess there's a paper that could be referred about MEIC?

****Response:****

Thank you for the comment. We have added the core references for the MEIC inventory at L200-201: Tong et al. (2020) describes the MEIC methodology and dynamic emission projections, and Zheng et al. (2021), both of which are key papers for the MEIC model.

> Comment:

L362: Does the observation operator include horizontal and/or vertical interpolation?

****Response:****

We thank the reviewer for this suggestion. The core references for the MEIC emission inventory have been added in Section 2.2: Li et al. (2017) and Zheng et al. (2018), which describe the MEIC methodology and its application in emission characterization.

> Comment:

L372: Two issues: first, the idea (“method of Li et al.”) of preconditioning the iteration with the square root of B is quite fundamental and has been described already in early variational DA literature (see e.g. the discussion in El Akkraoui et al., 2012). Second, the transformation alone would probably not solve the memory issue, since factorization of B would in general case be very large. It is arguably the separation of dimensions explained in section 2.6 that makes the problem more easily solvable.

****Response:****

We appreciate this insightful comment. Following the reviewer's suggestion, we have reorganized the manuscript to clarify the logical flow. Specifically:

(1) The square-root preconditioning is now properly attributed to the classical variational data assimilation literature (El Akkraoui et al., 2013, Line 281-283).

(2) The tensor product decomposition $C = Cx \otimes Cy \otimes Cz$ has been moved from Section 2.6 to the cost function optimization part (Section 2.4, Line 285-288), immediately following the decomposition $B = DCD^T$. This arrangement clearly illustrates that dimensional separation, rather than the preconditioning transformation alone, substantially cuts memory usage and improves computational efficiency.

(3) Section 2.5 now focuses exclusively on the statistical estimation of D, Cz, and the horizontal correlation scale L via the NMC method, avoiding redundancy with the earlier methodological description.

> Comment:

L387: Please give a reference for the L-BFGS method.

****Response:****

Thank you for the reminder. We have added the original reference for the L-BFGS method (Liu et al., 1989, Line 302) in the revised manuscript, and the full citation is included in the reference list.

> Comment:

L390: It's surprising that you manually would implement the L-BFGS algorithm, since there are many well-established implementations available for Fortran.

****Response:****

Thank you for the comment. We did not implement the L-BFGS algorithm from scratch. This Fortran solver is ported from classic open-source L-BFGS and MINPACK line search routines (Liu & Nocedal, 1989, Line 302; Moré et al., 1994, Line 306-307).

Standard precompiled libraries cannot be used directly due to incompatible calling interfaces. We only adjusted the interfaces to fit our 3DVar system, while keeping the original algorithm logic unchanged.

Unlike the highly encapsulated torch.LBFGS in Python, the complete source code of the Fortran solver is exposed, which gives an impression of manual implementation. We have revised the corresponding description in the manuscript (Line 312-314).

> Comment:

L390: Is the Fortran version an existing framework (reference?) or did you write it from scratch? How is it parallelized? What numerical linear algebra library (if any) does it link with?

****Response:****

The FT3DVar system is a self-developed variational assimilation framework. Its L-BFGS solver is not implemented from scratch; rather, it is adapted from the MINPACK library (Moré and Thuente, 1994), with tailored data interface modifications to ensure compatibility with our 3DVar system. Parallelization is achieved at the outer framework level using OpenMP multi-threading, while the core optimization subroutines themselves remain thread-local and are not explicitly parallelized. Furthermore, the code does not rely on any

external numerical linear algebra libraries; all basic matrix and vector operations are implemented natively within the framework.

> Comment:

L402: Automatic differentiation tools have for long time been available for Fortran and other languages, even if they are not always easy to use.

****Response:****

We fully agree with the reviewer that automatic differentiation (AD) tools are available for Fortran and, as noted, are not always easy to use in practice. For the FT3DVar system, the manual derivation of the adjoint code remains a reasonable and practical choice for two reasons. First, the cost function is composed of a small set of modular operations—primarily tensor-product decomposed matrix-vector products and observation operators—whose adjoints are analytically straightforward to derive. In this specific context, the manual effort is comparable to that required for configuring an AD tool. Second, manual derivation offers full control over memory allocation and execution flow, which is crucial for achieving stable and efficient multi-threaded OpenMP parallelization. By contrast, AD is adopted in the PY3DVar system because it is natively and readily accessible within the PyTorch ecosystem. The manuscript has been revised accordingly to clarify this distinction (Line 317-320).

> Comment:

L408: I actually don't find Eq. 4 especially complex as far as adjoint operators go. Please elaborate why it struggles to exploit parallel computing.

****Response:****

We fully agree with the reviewer that Eq. (4) is not particularly complex in terms of adjoint formulation. We therefore acknowledge that our original phrasing was overstated, and we have removed this statement from the revised manuscript.

To address the reviewer's underlying question regarding parallelization: the challenge does not arise from the complexity of the analytical expression itself, but rather from the practical difficulty of implementing loop-level parallelization (e.g., OpenMP) in manually derived adjoint codes while preserving data dependency and numerical reproducibility. In hand-coded adjoints, the forward and backward passes often involve intricate nested loops with irregular memory access patterns, making it challenging to safely parallelize without introducing race conditions or synchronization overhead. This is precisely the kind of engineering overhead that automatic differentiation (AD) can circumvent by automatically generating structured, parallelizable computational graphs. In PY3DVar, the use of automatic differentiation is primarily adopted for its convenience and development efficiency, as it eliminates the need for manual coding of the adjoint and significantly simplifies the integration of complex operators, especially when deploying the system across different hardware platforms (CPU and GPU) without code modification (Line 324-329). The manuscript has been revised to reflect this clarification.

> Comment:

L412: This should probably be section 2.6.

****Response:****

Thank you for catching this error. We have corrected the section number from 2.5 to 2.6 in the revised manuscript.

> Comment:

L414: You might consider also referring to earlier work where the tensor product formulation has been introduced.

****Response:****

We thank the reviewer for this valuable suggestion. In the revised text, we have accordingly cited two representative early studies—Derber and Rosati (1989) and Li et al. (2008)—that

introduced the tensor product decomposition for background error correlation matrices (Line 287-288).

> Comment:

Figure 2: It is very difficult to see the differences between the different lines here.

****Response:****

We thank the reviewer for this helpful suggestion. In the revised manuscript, Figure 2 has been split into six sub-panels (Fig. 2a–f) grouped by variable type, as follows:

• Fig. 2a: EC, NH_4, SO_4, NO_3

• Fig. 2b: ORG, CPO, CL, P25

• Fig. 2c: DUST1, DUST2

• Fig. 2d: ANTHA, SOILA

• Fig. 2e: SO_2, NO_2

• Fig. 2f: CO, O_3

This regrouping makes the differences between individual lines clearly distinguishable, and the corresponding figure caption and text have been updated accordingly.

> Comment:

L462: Were the forecasts for estimating the background error standard deviations run using the same setup as the assimilation experiments? Which resolution was used?

****Response:****

We thank the reviewer for this question. Yes, the 24 h and 48 h forecasts used for estimating the background error statistics via the NMC method were run using the same model configuration as the assimilation experiments, including identical physical parameterizations and emission inventories. Both the NMC forecast runs and the assimilation experiments were performed on the same 27 km grid. This consistency ensures that the background error covariances derived from the forecast differences are fully representative of the forecast errors encountered in the actual assimilation–forecast cycle. A clarifying statement has been added to the revised manuscript to reflect this (Line 337-339).

> Comment:

Figure 4: I'm quite puzzled with what is shown here. Is this a fit, or the original, empirical correlation function? How many pairs data points are there for each distance to evaluate the correlation coefficient? It seems that the SO2 fields must have very different spatial features from the others to explain this level of difference, or, put differently, the correlation functions for all other pollutants appear to be exceptionally regular. This is surprising, since also pollutants like NO2 or PM10 are emitted by point sources and they do typically show strong spatial heterogeneity. What is the x axis unit?

****Response:****

We sincerely thank the reviewer for the careful examination of Fig. 4. We agree that the original description was ambiguous regarding the calculation method, the X-axis unit, and the specific features of the SO_2 curve. We have thoroughly revised the manuscript and the figure to address your concerns. Below are our point-by-point responses:

(1) Is this a fit, or the original empirical correlation function?

The curves represent the empirical correlation coefficients computed directly from the 30-day NMC forecast differences, without any functional fitting (Line 405-406). For each distance d (in grid units), the correlation coefficient is computed as the average over all valid grid point pairs separated by that distance, following the standard NMC procedure (Parrish and Derber, 1992).

(2) How many data point pairs are there for each distance?

Since only surface in-situ observations are assimilated, the horizontal correlations are computed using the model surface layer (k=1) only. Under the assumption of horizontal isotropy, the x-direction and y-direction correlations are computed separately and then averaged. For a given distance d:

In the x-direction: for each of the 168 rows (south-north), there are (210 − d) point pairs spanning distance d, yielding up to 168 × (210 − d) pairs.

In the y-direction: for each of the 210 columns (west-east), there are (168 − d) point pairs spanning distance d, yielding up to 210 × (168 − d) pairs.

These two sets of correlations are then averaged to produce the final correlation value at distance d. The number of available pairs decreases linearly with increasing distance. At d=1 (27 km), there are approximately 168 × 209 + 210 × 167 ≈ 70,000 pairs; at d=100 (2700 km), there are approximately 168 × 110 + 210 × 68 ≈ 32,000 pairs. The correlation values at all distances are thus computed from a statistically robust number of samples.

(3) What is the x-axis unit?

The x-axis represents horizontal distance in grid units (1 grid unit = 27 km). In the revised Fig. 4, the x-axis is shown in physical distance (km) up to 500 km, beyond which the correlations have largely decayed and are no longer of practical interest for assimilation. This has been explicitly stated in the revised figure caption (Line 408).

(4) Why do the correlation functions for most pollutants appear exceptionally regular, while SO_2 previously showed fluctuations?

The fluctuations previously observed in the SO_2 curve beyond 20 grid units were not of physical origin; rather, they were a numerical artifact introduced by the vectorized computation approach used in the earlier version of our code. After a thorough re-examination of the numerical code, the revised statistics have been recomputed using an explicit grid-point-wise loop implementation, which is numerically consistent with the standard Fortran NMC procedure. As a result, all 16 control variables now exhibit smoothly decaying empirical correlation functions, consistent with the Gaussian-type correlation model. The previous discussion attributing the SO_2 fluctuations to point-source heterogeneity

has been removed, as it is no longer applicable. The revised Fig. 4 has been updated accordingly. We believe this correction strengthens the manuscript (Line 407-414).

> Comment:

L494: That's what the kernel prescribes, but is it a physical mechanism?

****Response:****

We fully agree with the reviewer that the term "physical mechanism" is not appropriate in this context. We have accordingly revised the phrase to "Gaussian-type correlation model" in the updated manuscript. This revised terminology more accurately reflects the mathematical nature of the prescribed correlation decay, rather than implying a physical mechanism (Line 410-412).

> Comment:

L537: Can you verify that the use of single precision floating point numbers is not in itself introducing numerical errors to the analysis?

****Response:****

We thank the reviewer for this important question. We have verified that the use of single precision does not introduce significant numerical errors to the analysis. First, it is worth noting that both FT3DVar and PY3DVar are implemented using 32-bit floating-point precision, ensuring a consistent numerical basis for a fair comparison. The following evidence further supports that this precision is sufficient:

(1) In the idealized experiment, the differences between PY3DVar and FT3DVar in the generated increment fields and analysis fields are negligible (Section 4.1), despite the two systems being implemented in entirely different frameworks.

(2) In the 20-day real-case assimilation experiment, the two systems achieved nearly identical analysis accuracy when evaluated against independent validation stations, and their

cost function convergence curves are indistinguishable. Such close agreement across two completely independent implementations would not be possible if single precision were introducing significant numerical errors.

(3) FT3DVar was originally developed using single precision and has been validated in several previous studies (Zang et al., 2016; Liang et al., 2020; Wang et al., 2022). Upgrading it to double precision would substantially increase computational cost without demonstrated benefit for this application.

(4) To directly test the sensitivity, we conducted additional PY3DVar runs using double precision (float64) for the same 20-day assimilation experiment and found the results to be identical to the single-precision results to within machine epsilon. This confirms that 32-bit precision is sufficient for the current assimilation configuration.

> Comment:

L542 I cannot find any information about Intel Xeon 7495 processors. Please check.

****Response:****

We sincerely appreciate the reviewer for catching this error. The processor model originally listed (Intel Xeon 7495) was based on preliminary tests conducted on a different machine, which was an oversight in our submission.

In the revised manuscript, all experiments have been re-conducted on a single unified platform equipped with dual AMD EPYC 7542 processors (each with 32 physical cores, totaling 64 cores @ 2.9 GHz, https://www.amd.com/en/support/downloads/drivers.html/processors/epyc/epyc-7002-series/amd-epyc-7542.html) and 256 GB of system memory. We have updated the hardware description in Section 4.4 accordingly. We apologize for any confusion caused by this inconsistency and thank the reviewer again for pointing it out (Line 421-423).

> Comment:

Section 4.1: This section should be shortened. Most of the findings here are very unsurprising and basically confirm that the code is working. I don't think that the reader needs to be explained that the increments "exhibit an approximately circular distribution", and so on.

****Response:****

We fully agree with the reviewer's comment and have substantially condensed Section 4.1. The idealized experiment is intended primarily for fundamental code validation and initial numerical consistency checks between PY3DVar and FT3DVar, rather than to present novel scientific findings. In the revised version, we have removed the detailed descriptions of spatial increment patterns and reduced the overall discussion to only the most essential results and quantitative comparisons. The shortened text now focuses strictly on the core numerical evidence that confirms the reliability of the new system, without elaborating on intuitively expected spatial characteristics. The corresponding revisions have been made in the manuscript (Section 4.1, Line 464-473).

> Comment:

L737: Is it possible to examine how sensitive the results are to the choice of parameters of the optimization algorithm (which might be more or less arbitrary)? Here it would be useful to perform a longer experiment to find out if the differences could be caused by essentially random differences in the iterations.

****Response:****

We appreciate the reviewer's suggestion regarding parameter sensitivity. We respectfully offer the following considerations:

First, the two systems already share identical values for the influential L-BFGS parameters: the memory size M = 4 and the maximum line search evaluations per iteration MAXFEV = 6. The convergence criterion is also identical, which is based on the scaled gradient norm:

$\|\nabla J\| / \max(1, \|x\|) \leq 1.0 \times 10^{-2}$. The remaining differences reside in the internal line search algorithms (More-Thuente in MINPACK vs. Strong Wolfe in PyTorch), which are not tunable parameters but fundamental design choices of the respective software libraries.

Second, we have tested stricter convergence thresholds ($\|\nabla J\| / \max(1, \|x\|) \leq 10^{-3}$ and 10^{-4}) for both systems. The resulting analysis fields and final cost values remained almost unchanged. This indicates that the observed convergence behavior is a stable characteristic of the different line search strategies, rather than an artifact of a specific threshold setting.

Given that the two algorithms are inherently different, a full systematic sensitivity study aimed at isolating the effect of individual parameters would require a large number of additional experiments and is methodologically nontrivial. Such an investigation would be valuable for optimization method development, but is beyond the primary scope of this work, which focuses on the overall efficiency and accuracy of the assimilation system. We believe the supplementary tests described above, combined with the close agreement in final cost values and analysis fields, are sufficient to confirm the robustness of our comparison.

We have added a brief acknowledgment of the algorithmic differences (Section 2.5) and their effect on convergence behavior in Section 4.3, and we believe the existing evidence—including the close agreement in final cost values, analysis fields, and forecast performance—is sufficient to support the conclusions of this work.

> Comment:

L751: Why not show the single iteration time for the Fortran version?

****Response:****

We thank the reviewer for the suggestion. The run time of a single iteration for the Fortran-based FT3DVar is included and illustrated in Fig. 12c and 13a.

> Comment:

L793: Can you elaborate on how the automatic parallel optimization works on the current problem? It would be very useful to see the speedup of the both CPU codes as a function of cores.

****Response:****

We thank the reviewer for this insightful comment, which prompted us to re-examine our original statement. We agree that the previous wording—"leverages vectorized operations and automatic parallel optimization... substantially enhancing computational efficiency"—could be misleading, as it might imply PY3DVar achieves better per-iteration parallel performance than FT3DVar. We have therefore corrected this phrasing in the revised manuscript to accurately reflect the data.

As shown in Fig. 13a, FT3DVar consistently outperforms PY3DVar in per-iteration time across all tested CPU core counts (1 to 64). The compiled Fortran implementation with explicit OpenMP loop-level parallelism is inherently faster on a per-iteration basis. The overall computational advantage of PY3DVar at higher core counts arises not from superior per-iteration efficiency, but from its substantially fewer iterations to convergence (mean of 7.5 vs. 24.9; Fig. 12a). This allows PY3DVar to overcome its per-iteration disadvantage and achieve a shorter total assimilation time when 32 or more cores are utilized (Fig. 13b).

Regarding the parallelization mechanism: PyTorch automatically parallelizes tensor operations across multiple CPU cores through its multi-threaded backend, without requiring manual parallel directives. This is why PY3DVar exhibits a pronounced reduction in both per-iteration and total time as core counts increase. FT3DVar also scales with core count, but its per-iteration time is already very fast at low core counts, leaving limited room for further acceleration. The speedup curve for PY3DVar relative to FT3DVar in terms of total iteration time is shown in Fig. 13c. The relevant discussions in Section 4.4 have been updated accordingly. We appreciate the reviewer's careful reading, which helped us avoid an inaccurate characterization.

> Comment:

L827: This is a very common finding in chemical data assimilation and it might be worth citing some earlier studies with similar results.

Response:

We thank the reviewer for this suggestion. We have added citations to earlier chemical data assimilation studies that report similar forecast improvement decay, including Pagowski et al. (2010), Jiang et al. (2013), and Li et al. (2013) in Line 581.

> Comment:

L867: “significantly optimizes” sounds ambiguous in a statistical context.

Response:

We agree that "significantly optimizes" is ambiguous in a statistical context. The phrase has been revised to "substantially improved" (Line 621), which more accurately describes the assimilation performance.

> Comment:

L870-881: The listing of statistical parameters is very tedious and should be summarized.

Response:

We sincerely thank the reviewer for this constructive comment. We completely agree that the original extensive enumeration of statistical parameters made the section tedious and distracted from the core conclusions. In the revised manuscript, we have accordingly condensed this part into a concise narrative summary. Specifically, we removed the lengthy data lists and replaced them with a single sentence that describes the overall performance improvements in CORR, RMSE, and MAE (Line 622-624). The specific numerical values for each species are now left to the scatter plots and forecast evolution diagrams (Figs. 14 and 15), which present the results more intuitively than repetitive text.

> Comment:

L884: “due to its use of vectorized […] computations”. With the material that has been presented (and no source code for the Fortran implementation), it’s actually not possible to identify the causes for the performance difference. For all we know, it could be be due to a sub-optimally ordered for loop, inefficient use of parallelization directives, or a poorly performing linear algebra library.

****Response:****

We sincerely thank the reviewer for this rigorous comment. We completely agree that attributing the CPU performance difference to "vectorized computations" without providing the Fortran source code was an overstatement and not strictly verifiable.

Following the reviewer's suggestion, we re-examined the computational performance from a more fundamental perspective and conducted parallel scalability tests across 1, 2, 4, 8, 16, 32, and 64 CPU cores for both FT3DVar and PY3DVar. These tests led us to refine our conclusions in the revised manuscript as follows:

1. FT3DVar is inherently faster in per-iteration computational speed. Across all tested core counts, the mean per-iteration time of FT3DVar consistently outperforms that of PY3DVar, confirming that the Fortran implementation remains superior in single-step calculation speed. The earlier draft has been corrected accordingly in the new Section 4.4.
2. PY3DVar achieves shorter total time through algorithmic advantages. The faster total assimilation time of PY3DVar at high core counts (≥ 32 cores) is not due to faster per-iteration speed, but rather to its substantially fewer iterations required for convergence (mean of 7.5 vs. 24.9; Fig. 12a). Combined with its superior parallel scaling, PY3DVar overtakes FT3DVar in total iteration time at high core counts, despite its higher per-iteration cost.

We hope this more rigorous and objective analysis addresses the reviewer's concern. The corresponding text in Section 4.4 (Line 538-557) and Section 5 (Line 627-638) has been revised accordingly.

> Comment:

L900: Again, there is no evidence of the role of automatic differentiation in the speedup. I suspect it to be rather small, since the gradient (Eq. 4) is easy to implement in any computational framework.

****Response:****

We sincerely thank the reviewer for pointing out this crucial distinction. We fully agree that automatic differentiation (AD) plays a negligible role in accelerating the computation itself, especially since the gradient formulation for the cost function (Eq. 4) is straightforward to implement in any framework.

In the revised manuscript, we have carefully corrected this misinterpretation and clarified the respective roles of the different technologies:

Role of AD: As the reviewer correctly noted, AD does not contribute to the computational speedup. Its primary value lies in simplifying the software engineering process by eliminating the need for tedious and error-prone manual adjoint coding. We have explicitly stated this in the revised summary(Lines 609–610 and 647–648).

Source of the computational speedup: The observed dramatic speedup is now attributed exclusively to GPU-accelerated tensor computations (yielding a 22.2-fold reduction in total time) and the faster convergence of the PyTorch L-BFGS optimizer (which reduces the required iterations from 24.9 to 7.5 due to the Strong Wolfe line search) (Line 626-638).

> Comment:

L904: The closing statement is an unnecessary hyperbole. Besides using an ML framework for doing data assimilation, you have not shown any results on integration of deep learning models or “intelligent data assimilation”, whatever it might mean.

****Response:****

We thank the reviewer for this comment. The original closing statement has been removed, as it contained unsupported claims regarding "intelligent data assimilation" and the "integration of deep learning models," which are beyond the scope of the present study. The revised summary now focuses strictly on the demonstrated findings: PY3DVar achieves a substantial computational speedup over FT3DVar through GPU-accelerated tensor computation and faster convergence of the L-BFGS optimizer, while maintaining equivalent assimilation accuracy. Automatic differentiation is acknowledged for its role in simplifying code development. We appreciate the reviewer's guidance in helping us avoid overstatement.

References

Carlton, A. G., Bhave, P. V., Napelenok, S. L., Edney, E. O., Sarwar, G., Pinder, R. W., Pouliot, G. A., and Houyoux, M.: Model Representation of Secondary Organic Aerosol in CMAQv4.7, Environ. Sci. Technol., 44, 8553-8560, https://doi.org/10.1021/es100636q, 2010.

Cheng, S., Min, J., Liu, C., and Arcucci, R.: TorchDA: A Python package for performing data assimilation with deep learning forward and transformation functions, Comput. Phys. Commun., 306, 109359, https://doi.org/10.1016/j.cpc.2024.109359, 2025.

Derber, J. and Rosati, A.: A Global Oceanic Data Assimilation System, J. Phys. Oceanogr., 19, 1333-1347, https://doi.org/10.1175/1520-0485(1989)019%3C1333:AGODAS%3E2.0.CO;2, 1989.

El Akkraoui, A., Trémolet, Y., and Todling, R.: Preconditioning of variational data assimilation and the use of a bi-conjugate gradient method, Q. J. R. Meteorol. Soc., 139, 731-741, https://doi.org/10.1002/qj.1997, 2013.

Fast J D , Gustafson W I , Easter R C ,et al.Evolution of Ozone, particulates, and aerosol direct radiative forcing in the vicinity of Houston using a fully coupled meteorology-chemistry-aerosol model. J. Geophys. Res.-Atmos., 2006, 111(D21).DOI:10.1029/2005jd006721.

Grell, G. A., Peckham, S. E., Schmitz, R., McKeen, S. A., Frost, G., Skamarock, W. C., and Eder, B.: Fully coupled “online” chemistry within the WRF model, Atmos. Environ., 39, 6957-6975, https://doi.org/10.1016/j.atmosenv.2005.04.027, 2005.

Key, O., Takao, S., Giles, D., and Deisenroth, M. P.: Scalable data assimilation with message passing, Environ. Data Sci., 4, e1, https://doi.org/10.1017/eds.2024.47, 2025.

Li, Z., Chao, Y., McWilliams, J. C., and Ide, K.: A three‐dimensional variational data assimilation scheme for the Regional Ocean Modeling System: Implementation and basic experiments, J. Geophys. Res., 113, 2006JC004042, https://doi.org/10.1029/2006JC004042, 2008.

Liu, D. C. and Nocedal, J.: On the limited memory BFGS method for large scale optimization, Math. Program., 45, 503-528, https://doi.org/10.1007/BF01589116, 1989.

Lorenc, A. C.: Analysis methods for numerical weather prediction, Q. J. R. Meteoro. Soc, 112, 1177-1194, https://doi.org/10.1002/qj.49711247414, 1986.

Moré, J. J., and Thuente, D. J.: Line search algorithms with guaranteed sufficient decrease, ACM Trans. Math. Softw., 20(3), 286–307, https://doi.org/10.1145/192115.192132, 1994.

Tong, D., Cheng, J., Liu, Y., Yu, S., Yan, L., Hong, C., Qin, Y., Zhao, H., Zheng, Y., Geng, G., Li, M., Liu, F., Zhang, Y., Zheng, B., Clarke, L., and Zhang, Q.: Dynamic projection of anthropogenic emissions in China: methodology and 2015‐2050 emission pathways under a range of socio-economic, climate policy, and pollution control scenarios, Atmos. Chem. Phys., 20, 5729-5757, https://doi.org/10.5194/acp-20-5729-2020, 2020.

Zheng, B., Cheng, J., Geng, G., Wang, X., Li, M., Shi, Q., Qi, J., Lei, Y., Zhang, Q., and He, K.: Mapping anthropogenic emissions in China at 1 km spatial resolution and its application in air quality modeling, Sci. Bull., 66, 612-620, https://doi.org/10.1016/j.scib.2020.12.008, 2021.