*atmosphere*

MDPI

*Article*

# Optimizer-Dependent Trade-Offs in Variational Multiple-Doppler Wind Retrieval: A Comparison of Adam, L-BFGS, and Newton-CG

**Seong-Jun Hwang [1], Jisun Lee [2,*], Dong-In Lee [2] and Yu-Chieng Liou [3]**

[1] Department of Environmental Atmospheric Sciences, Pukyong National University, Busan 48513, Republic of Korea; sungjune12@gmail.com
[2] Atmospheric Environmental Research Institute, Pukyong National University, Busan 48513, Republic of Korea; leedi@pknu.ac.kr
[3] Department of Atmospheric Sciences, National Central University, Taoyuan 320317, Taiwan; tyliou@atm.ncu.edu.tw
* Correspondence: jisun_lee@pukyong.ac.kr

**Abstract**

The Wind Synthesis System using Doppler Measurements (WISSDOM) retrieves three-dimensional winds by minimizing a variational objective that combines Doppler-radar observations and physical constraints. Using a common Python implementation with PyTorch (2.9.1+cu130) automatic differentiation on an NVIDIA GPU, this study compared Adam, limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), and trust-region Newton conjugate-gradient (Newton-CG) for the same Typhoon Bavi case, initial wind field, and objective. At solver-specific stopping points, all three methods substantially reduced the objective and produced broadly similar principal horizontal- and vertical-wind structures. L-BFGS and Newton-CG produced the closest final wind fields, whereas Adam showed somewhat larger local differences. Their final objective values and gradient norms nevertheless differed: Newton-CG reached the smallest objective and relative full-gradient norm, while L-BFGS reached similar final values at much lower computational cost. Separate timing runs compared execution with compilation disabled for all solvers and with compilation enabled for Adam and L-BFGS only. Comparisons at common objective targets and common elapsed-time budgets provided complementary measures of computational cost and objective reduction. Within the tested solver-specific configurations, these results identify different balances between computational cost and convergence: Adam provided the lowest-cost route to broadly similar principal wind structures, L-BFGS balanced computational cost and numerical convergence, and Newton-CG reached the smallest monitored gradient norm at the highest computational cost.

**Keywords:** variational multiple-Doppler wind retrieval; WISSDOM; Doppler radar; numerical optimization; automatic differentiation; Adam; L-BFGS; Newton-CG

Check for updates

Academic Editor: Jimy Dudhia

Received: 1 September 2026
Revised: 1 October 2026
Accepted: 15 October 2026
Published: 16 October 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

## 1. Introduction

Three-dimensional wind analyses derived from scanning Doppler radars provide an important basis for investigating the kinematics of convective storms, tropical cyclones, and precipitation systems. A Doppler radar, however, measures only the component of hydrometeor motion along its beam. Recovering the full wind vector therefore requires observations from multiple viewing directions together with assumptions or constraints

for poorly observed components, particularly vertical velocity. Variational multi-Doppler techniques address this inverse problem by minimizing a cost function that combines radial-velocity observations with kinematic, dynamical, background, and regularization constraints [1,2]. Recent studies have extended variational multi-Doppler retrievals to complex terrain and shown that both terrain-boundary treatment and spatial interpolation can materially affect the retrieved three-dimensional wind fields [3,4]. This formulation avoids explicit vertical integration of the continuity equation and permits observations and physical constraints to be imposed simultaneously in a common analysis, although the retrieved fields remain sensitive to data coverage, boundary assumptions, and the specification of individual constraints [5].

The Wind Synthesis System using Doppler Measurements (WISSDOM) is a variational multiple-Doppler framework designed to retrieve three-dimensional winds at two analysis times [2]. Its objective function combines the fit to observed radial velocities with constraints associated with mass continuity, vertical vorticity, background winds, spatial regularity, and domain boundaries. An immersed-boundary formulation further enables the analysis to incorporate terrain forcing on a Cartesian grid over complex topography [6]. More recently, WISSDOM-retrieved meteorological state variables were assimilated into a numerical model to improve short-term precipitation forecasts over mountainous terrain [7]. These capabilities make WISSDOM useful for retrieving physically constrained flow structures in environments where radar sampling is incomplete and terrain effects are important. At the same time, the discretized wind components form a large control vector, and the relative objective coefficients and nonlinear interactions among the objective terms can produce an optimization problem with strongly differing sensitivities across variables and physical constraints. The wind analysis is therefore determined not only by the objective function and observations, but also by how that objective is minimized in practice.

Large-scale meteorological variational systems have commonly relied on gradient-based limited-memory methods because explicitly forming and storing a Hessian is infeasible at realistic problem sizes [8]. L-BFGS constructs an inverse-Hessian approximation from a limited history of parameter and gradient differences and has become a standard option for large-scale unconstrained minimization [9,10]. It is also used by the original WISSDOM implementation and therefore provides the algorithmic baseline for the present study. However, an optimizer does more than determine the computational cost of minimization. Its search directions, step-acceptance mechanism, use of curvature information, and stopping behavior determine the path by which competing objective terms are reduced and the degree to which the final state satisfies first-order stationarity. Consequently, similar final objective values or visually similar wind fields do not by themselves establish that different optimizers have reached numerically equivalent solutions.

The role of curvature information in meteorological optimization has been examined previously. Limited-memory quasi-Newton and truncated-Newton methods were compared in large-scale meteorological and oceanographic applications, with truncated-Newton methods showing performance competitive with L-BFGS but not uniformly superior across the problems examined [11]. An adjoint truncated-Newton method using exact Hessian-vector products was subsequently applied to a limited-area shallow-water variational data-assimilation problem and compared with L-BFGS [12]. The role of the Hessian in diagnosing local convexity, solution uniqueness, and optimization convergence was also examined, together with an automatic-differentiation and second-order-adjoint implementation of exact Hessian-vector products for truncated-Newton minimization [13]. Collectively, these studies established the importance of curvature information and the distinct convergence properties of quasi-Newton and truncated-Newton methods in meteorological optimization. Adaptive first-order methods such as Adam emerged more than

a decade later, creating a new opportunity to examine how optimizers that use different levels of curvature information behave when applied to the same physically formulated multi-Doppler wind-retrieval objective.

Adam was introduced for stochastic gradient-based optimization and subsequently became widely used to minimize training losses with respect to neural-network parameters [14]. This conventional role in deep learning differs from using Adam as a numerical solver that directly updates physical state variables in a deterministic variational inverse problem. A more closely related, but still distinct, application used Adam to minimize an auto-differentiable 3D-Var objective with respect to latent atmospheric control variables while keeping a pretrained variational autoencoder fixed [15]. Although that application involved minimization of a data-assimilation objective rather than neural-network training, the optimization was performed in a learned, reduced-dimensional representation. It therefore did not directly optimize grid point wind components subject to radar-observation and physical-constraint terms. To the best of our knowledge, Adam has not been evaluated against quasi-Newton and truncated second-order methods for the same physically formulated multi-Doppler wind-retrieval objective.

Modern automatic differentiation frameworks make such a controlled comparison practical. Reverse-mode automatic differentiation evaluates gradients of a programmed objective without requiring a separately derived adjoint for each physical term [16], and the same computational graph can provide Hessian-vector products without explicitly assembling the Hessian [17]. PyTorch combines these differentiation capabilities with hardware-accelerated tensor operations [18], allowing several optimization classes to operate on the same implementation of the WISSDOM objective. In this study, Adam represents adaptive first-order optimization based on exponential moments of the gradient [14]; L-BFGS represents limited-memory quasi-Newton optimization based on curvature inferred from iteration history [9]; and a CG-Steihaug trust-region Newton-CG method represents truncated second-order optimization based on current Hessian-vector products and an approximate trust-region subproblem solution [19]. This ordering spans progressively more explicit use of curvature information without requiring storage of the full Hessian.

Using this common computational framework, this study provides a controlled comparison of three optimization methods—adaptive first-order Adam, limited-memory quasi-Newton L-BFGS, and matrix-free trust-region Newton-CG—for the same physically formulated variational multiple-Doppler wind-retrieval problem. The three methods were evaluated while holding the observations, initial wind state, physical constraints, objective coefficients, numerical precision, and evaluation procedure fixed. The comparison addresses three questions: (1) how consistently do the algorithms produce the principal retrieved wind structures; (2) how do they differ in objective reduction, first-order stationarity, and the balance among individual physical cost components; and (3) what trade-offs arise between computational cost and the numerical and physical properties of the retrieved solution? The comparison evaluates the chosen solver settings and update limits for a fixed WISSDOM objective; it is not intended as a benchmark of theoretical optimizer efficiency or as a search for a universally superior optimizer. Instead, the study aims to identify solver choices for different retrieval priorities: rapidly obtaining the principal wind structures, maintaining continuity with the established L-BFGS solution, or pursuing stronger numerical stationarity with Newton-CG.

## 2. Materials and Methods

### *2.1. WISSDOM Variational Retrieval Framework*

The Wind Synthesis System using Doppler Measurements (WISSDOM) retrieves three-dimensional wind fields by variationally adjusting the wind components so that radar

observations and a set of kinematic constraints are approximately satisfied in a weak-constraint sense [2]. For radar observations collected at two analysis times, the control vector is defined as

$$x = (u_1, v_1, w_1, u_2, v_2, w_2). \tag{1}$$

Here, $u_t$, $v_t$, and $w_t$ are the three Cartesian wind components at time level $t$. Let $J_{\text{sum}}$ denote the unnormalized WISSDOM penalty sum used in the Fortran implementation. It contains seven terms [2]:

$$J_{\text{sum}}(x) = \sum_{m=1}^{7} J_m = J_1 + J_2 + J_3 + J_4 + J_5 + J_6 + J_7. \tag{2}$$

The objective minimized and reported in the optimizer comparison was $J = J_{\text{sum}}/(2N_{\text{grid}})$, where $N_{\text{grid}} = N_x N_y N_z$. All seven terms remained in the formulation. Their coefficients set the size of each contribution, and a zero coefficient gave a zero contribution.

The first term measures the discrepancy between the radial velocities observed by $N$ Doppler radars and the radial velocities projected from the retrieved Cartesian winds:

$$J_1 = \sum_{t=1}^{2} \sum_{i=1}^{N} \sum_{\Omega} \alpha_{1,i} T_{1,i,t}^2, \tag{3}$$

where

$$T_{1,i,t} = (V_r)_{i,t} - \frac{x - P_x^i}{r_i} u_t - \frac{y - P_y^i}{r_i} v_t - \frac{z - P_z^i}{r_i} (w_t - W_{T,t}). \tag{4}$$

Here, $(V_r)_{i,t}$ is the radial velocity observed by radar $i$, $\left(P_x^i, P_y^i, P_z^i\right)$ denotes the radar location, $r_i$ is the distance between the radar and an analysis grid point, and $W_{T,t}$ is the hydrometeor terminal fall velocity estimated from radar reflectivity. The spatially varying coefficient $\alpha_{1,i}$ determines whether a grid point is covered by observations from radar $i$. Thus, $J_1$ penalizes disagreement between the retrieved wind field and the assimilated radial-velocity observations.

The second term constrains the retrieved winds using a background wind field:

$$J_2 = \alpha_2 \sum_{t=1}^{2} \sum_{r \in \Omega_{\text{bg},t}} \| V_t(r) - V_{B,t}(r) \|_2^2. \tag{5}$$

where

$$V_t = u_t i + v_t j + w_t k. \tag{6}$$

The background wind $V_{B,t}$ may come from vertical profile data, an analysis, or a model forecast. It provides wind information where radar data are not available. Here, $\Omega_{\text{bg},t}$ denotes the atmospheric-flow grid cells where no radar observation was used at analysis time $t$.

The third term penalizes residuals of the anelastic mass-continuity equation:

$$J_3 = \sum_{t=1}^{2} \sum_{\Omega} \alpha_{3,t} \left[ \rho_0 \left( \frac{\partial u_t}{\partial x} + \frac{\partial v_t}{\partial y} + \frac{\partial w_t}{\partial z} \right) + w_t \frac{\partial \rho_0}{\partial z} \right]^2, \tag{7}$$

where $\rho_0$ is the reference atmospheric density, assumed to vary only with height. Unlike conventional multiple-Doppler retrieval methods that explicitly integrate the continuity equation vertically, WISSDOM incorporates its residual directly into the objective function.

The fourth term is constructed from the residual of the vertical-vorticity equation:

$$J_4 = \alpha_4 \sum_{\Omega} R_\zeta^2, \tag{8}$$

where $R_\zeta$ represents the imbalance in the vertical-vorticity equation evaluated from the retrieved winds at the two analysis times. This constraint introduces temporal and dynamical information and helps constrain the wind field where radial-velocity geometry alone provides insufficient information.

The fifth term combines Laplacian smoothing and height-dependent damping:

$$J_5 = J_{\text{Lap}} + J_{\text{damp}}. \tag{9}$$

The Laplacian part follows the original WISSDOM formulation of Liou and Chang (2009) and is also used in the complex-terrain version of Liou et al. (2012) [2,6]. In the present configuration, this term combined Laplacian smoothing with height-dependent damping toward the background wind, and only the vertical-wind damping coefficient was nonzero. The same formulation and coefficients were used for all three solvers.

The final two terms are optional constraints on vertical velocity at the upper and lower boundaries. The upper-boundary term penalizes a nonzero horizontally averaged vertical velocity:

$$J_6 = \alpha_6 \sum_{t=1}^{2} \left[ \frac{1}{N_x N_y} \sum_{x,y} (w_t)_{\text{top}} \right]^2, \tag{10}$$

whereas the lower-boundary term constrains vertical velocity over atmospheric-flow cells at the lowest atmospheric level:

$$J_7 = \alpha_7 \sum_{t=1}^{2} \sum_{(x,y) \in \Gamma_{\text{bot}}^f} w_t \Big(x, y)^2. \tag{11}$$

Here, $\Gamma_{\text{bot}}^f$ denotes the atmospheric-flow cells at the lowest atmospheric level. Among the two boundary penalties, $J_7$ was the only boundary term with a nonzero coefficient. The radar, background, continuity, vorticity, and smoothing terms $J_1$–$J_5$ were also active.

The objective coefficients control the relative contributions of the observational, dynamical, regularization, and boundary constraints. Consequently, minimizing J does not enforce each equation exactly; instead, it produces a wind field representing a weighted compromise among radar-observation fit, background information, mass continuity, vertical-vorticity consistency, spatial smoothness, and boundary behavior. The coefficient values in Table 1 were taken from the original Fortran configuration and kept identical for all three solvers. This comparison therefore evaluates optimizer behavior for that fixed physical configuration.

The complex-terrain setup follows Liou et al. (2012) [6], who added a ghost-cell immersed-boundary method to WISSDOM and changed how wind boundary conditions were applied over terrain. Wind values at ghost cells were set using nearby image points, so terrain boundary conditions could be applied on the Cartesian grid without a terrain-following grid. The same objective coefficients, terrain treatment, and boundary conditions were used for all three solvers. Thus, only the optimizer update and step-acceptance rules differed.

**Table 1.** Common physical coefficients in the WISSDOM configuration.

| Term or Component | Coefficient |
|---|---|
| $J_1$: radial velocity | 1000 |
| $J_2$: background wind | 0.1 |
| $J_3$: density-weighted continuity | $4 \times 10^8$ |
| $J_4$: vertical vorticity | $4 \times 10^{13}$ |
| $J_5$: horizontal-wind smoothing (horizontal/vertical derivatives) | $10^{15}/10^{11}$ |
| $J_5$: vertical-wind smoothing (horizontal/vertical derivatives) | $10^{14}/10^{12}$ |
| $J_5$: height-dependent vertical-wind damping | 0–1000 (see note) |
| $J_6$: upper boundary/$J_7$: lower boundary | 0/100 |
| Additional reflectivity-conservation and domain-mean damping categories | 0 |

Note: These coefficients follow the Fortran configuration and are identical for all three solvers. The vertical-wind damping coefficient was zero up to 9.25 km and increased linearly to 1000 at 10 km. Horizontal-wind height-dependent damping is zero. The total penalty and its reporting components are divided by $2N_{grid}$, as defined in Section 2.1.

### *2.2. Common Computational Framework and Execution Environment*

The WISSDOM workflow from the Fortran implementation was reimplemented in Python. NumPy and Numba were used mainly for data input, preprocessing, and postprocessing, with Numba accelerating many CPU calculations [20,21]. PyTorch [18] was used for the variational optimization. The processed radar observations, radar geometry, background fields, terrain data, interpolation data, and objective coefficients were converted to PyTorch tensors before optimization. The objective terms were combined in one common cost evaluator used by all three solvers. It returned the normalized total objective and the unnormalized value of each reporting component, and the same normalization was then applied to each component.

The six gridded wind fields at the two analysis times ($u_1$, $v_1$, $w_1$, $u_2$, $v_2$, $w_2$) were represented directly as optimization variables. Reverse-mode automatic differentiation was used to calculate the gradient of the discrete objective with respect to these wind components [16]. The gradients were therefore obtained by differentiating the numerical operations executed by the common cost evaluator, without requiring separately maintained analytical gradient routines for the individual objective components. Adam used these gradients in adaptive first-order updates. L-BFGS used them with a limited history of previous steps and gradient changes to form a quasi-Newton inverse-Hessian approximation. Newton-CG additionally obtained Hessian-vector products by differentiating the gradient without explicitly constructing or storing the full Hessian matrix [17].

The optimizers directly updated six wind tensors before the boundary conditions were applied; these are referred to here as the optimizer-state fields. Before each objective evaluation, the common boundary treatment described in Section 2.1 was applied to produce the projected fields: the Neumann boundary treatment was applied to the horizontal-wind components, and the terrain Dirichlet treatment was applied to the vertical-wind components. The projected fields, rather than the optimizer-state fields, were used to calculate the objective. After the final optimizer update, the same boundary treatment was applied once more, and the resulting projected fields were passed to the standard WISSDOM postprocessing and output procedures.

The optimizer-state fields and the array-valued observational, geometric, physical, and objective-function inputs were represented as tensors on an NVIDIA GPU through CUDA during optimization. Scalar grid and physical parameters were supplied to the same tensor-based objective evaluator. The objective calculation, automatic differentiation, and

optimizer updates were performed with PyTorch on this GPU. The same processed inputs, initial wind fields, objective coefficients, boundary treatment, normalization, numerical precision, and computing hardware were used by all solvers within each case. The comparison without compilation used the same objective evaluator for all solvers. Separate timing runs enabled compilation for Adam and L-BFGS while keeping it disabled for Newton-CG, as described in Section 2.4.3. The algorithm settings are presented in Section 2.4, and the evaluation and monitoring procedures are described in Section 2.5.

### *2.3. Optimization Methods*

In the controlled comparison, all three optimization methods minimized the same differentiable WISSDOM objective from the same initial wind state. They differed in how the update direction and step were determined. Adam used adaptive rescaling based on the first and second raw moments of the gradient. L-BFGS constructed an inverse-Hessian approximation from a limited history of variable and gradient changes and selected the step through a line search. Trust-region Newton-CG used Hessian-vector products to incorporate the current local curvature while solving a constrained quadratic subproblem. Holding the physical formulation and common evaluation procedure fixed allowed convergence, numerical stationarity, and retrieved wind fields to be compared across the three update and step-acceptance procedures.

#### 2.3.1. Adam

Adam is an adaptive first-order method that maintains exponentially weighted estimates of the first and second raw moments of the gradient [14]. To avoid confusion between the second-moment variable and the meridional wind component $v$, let $q_k$ denote the second raw-moment estimate. The optimization variables were divided into horizontal- and vertical-wind blocks,

$$x^{(uv)} = (u_1, v_1, u_2, v_2), x^{(w)} = (w_1, w_2).$$

For parameter block $b \in \{uv, w\}$, the block gradient is $g_k^{(b)} = \nabla_{x^{(b)}} J(x_k)$. Starting from $m_0^{(b)} = q_0^{(b)} = 0$, the moment estimates are updated as

$$\begin{aligned} m_k^{(b)} &= \beta_1 m_{k-1}^{(b)} + (1-\beta_1) g_k^{(b)}, \\ q_k^{(b)} &= \beta_2 q_{k-1}^{(b)} + (1-\beta_2)\left(g_k^{(b)} \odot g_k^{(b)}\right), \end{aligned} \tag{12}$$

where $\odot$ denotes elementwise multiplication. The initialization bias is corrected according to $\hat{m}_k^{(b)} = m_k^{(b)} / \left(1-\beta_1^k\right)$ and $\hat{q}_k^{(b)} = q_k^{(b)} / \left(1-\beta_2^k\right)$. Each parameter block is then updated as

$$\begin{aligned} x_{k+1}^{(b)} &= x_k^{(b)} - \eta_b \hat{m}_k^{(b)} \left[ \left( \hat{q}_k^{(b)} \right)^{1/2} + \epsilon \right]^{-1}, b \in \{uv, w\}, \\ \eta_{uv} &= s_{uv}\eta, \ \eta_w = s_w \eta. \end{aligned} \tag{13}$$

Here, $\eta$ is the base learning rate, while $s_{uv}$ and $s_w$ are learning-rate multipliers for the horizontal- and vertical-wind blocks, respectively. The optimization variables are the grid point u, v, and w values at the two analysis times. Separate multipliers allow the horizontal- and vertical-wind update scales to be adjusted independently. Their values were selected through preliminary Bavi trials, as described in Section 2.4.2.

The official PyTorch Adam implementation was used with its default moment-decay parameters, numerical-stability constant, and no weight decay. Although Adam was originally developed for stochastic optimization, no mini-batches or stochastic gradient

estimates were used here. Each update was based on the gradient of the complete deterministic WISSDOM objective. Adam was included to evaluate whether an adaptive first-order method requiring one full-objective gradient per update, but no line search, quasi-Newton correction history, or Hessian-vector products, could recover the principal wind structures while exhibiting a different convergence profile from the other methods.

#### 2.3.2. L-BFGS

L-BFGS is a limited-memory quasi-Newton method that represents inverse-Hessian information using a finite history of changes in the optimization variables and their gradients [9]. For two successive iterates, the correction pair is defined as

$$s_k = x_{k+1} - x_k, y_k = g_{k+1} - g_k. \tag{14}$$

Using a limited number of recent correction pairs, the two-loop recursion applies an inverse-Hessian approximation without forming or storing a dense matrix. The resulting search direction and update can be written as

$$p_k = -H_k^{\mathrm{LBFGS}} g_k, x_{k+1} = x_k + \alpha_k p_k. \tag{15}$$

Here, $H_k^{\mathrm{LBFGS}}$ is the limited-memory inverse-Hessian approximation and $\alpha_k$ is determined by a line search. The L-BFGS experiments used the optimizer implementation provided by PyTorch, with a strong-Wolfe line search. The line search seeks a step length satisfying the sufficient-decrease and strong-curvature conditions

$$\begin{gathered} J(x_k + \alpha_k p_k) \le J(x_k) + c_1 \alpha_k g_k^{\mathrm{T}} p_k, \\ \left| \nabla J(x_k + \alpha_k p_k)^{\mathrm{T}} p_k \right| \le c_2 \left| g_k^{\mathrm{T}} p_k \right|, 0 < c_1 < c_2 < 1, \end{gathered} \tag{16}$$

as described by Moré and Thuente (1994) [22].

L-BFGS was selected as the baseline because the original Fortran WISSDOM implementation also employed L-BFGS [2]. Retaining the same optimization method provides continuity with the established WISSDOM retrieval procedure and a reference against which Adam and Newton-CG can be evaluated within the common PyTorch framework. The correction-history limit and line-search settings used in the experiments are reported in Section 2.4.

#### 2.3.3. Trust-Region Newton-CG with Steihaug Truncation

The third solver was a matrix-free trust-region Newton conjugate-gradient (Newton-CG) method. At outer iteration $k$, the step was obtained by approximately solving the local quadratic subproblem

$$\underset{\|p\|_2 \le \Delta_k}{\text{minimize}} m_k(p) = J(x_k) + g_k^{\mathrm{T}} p + \frac{1}{2} p^{\mathrm{T}} H_k p, \tag{17}$$

where $g_k$=$\nabla J(x_k)$, $H_k = \nabla^2 J(x_k)$, and $\Delta_k$ is the trust-region radius [10,19]. The trust region limits the step to a neighborhood in which the local quadratic model is considered sufficiently reliable.

The full Hessian matrix was neither constructed nor stored. For each conjugate-gradient direction $d$, the required Hessian-vector product was computed by differentiating the gradient,

$$\begin{gathered} H_k d = \left. \frac{d}{d\varepsilon} \nabla J(x_k + \epsilon d) \right|_{\varepsilon = 0}, \\ H_k d = \left. \nabla_x \left[ \nabla J(x)^{\mathrm{T}} d \right] \right|_{x = x_k}, \end{gathered} \tag{18}$$

where $d$ is held fixed during differentiation [17]. This matrix-free calculation provides local curvature information without requiring storage of a dense Hessian.

The trust-region subproblem was solved approximately using Steihaug truncated conjugate gradients [19]. The inner iteration was terminated when the residual tolerance was satisfied, the next CG iterate reached the trust-region boundary, nonpositive or numerically insufficient curvature was detected, or the prescribed inner-iteration limit was reached. At a curvature-truncation event, the boundary intersection giving the lower quadratic-model value was retained. If the CG calculation became numerically invalid or returned a non-descent step, a radius-limited Cauchy step was used as a safeguard [10].

The trial step was evaluated using the ratio of actual to predicted reduction,

$$\rho_k = \frac{J(x_k) - J(x_k + p_k)}{-g_k^{\mathrm{T}} p_k - \frac{1}{2} p_k^{\mathrm{T}} H_k p_k}. \tag{19}$$

The trial state was accepted only when $\rho_k$ exceeded the acceptance threshold; otherwise, the original wind fields were restored. In the implemented radius-update rule, the trust-region radius was reduced by a factor of 0.25 when $\rho_k$ was unavailable or below 0.25. It was doubled, subject to the prescribed maximum radius, when $\rho_k$ exceeded 0.75 and the trial step reached the trust-region boundary. The numerical thresholds and inner-iteration limits are reported in Section 2.4.

Newton-CG was selected because Hessian-vector products incorporate current local curvature without explicitly storing the Hessian, while the Steihaug trust-region procedure provides a defined response to weak or indefinite curvature. The comparison evaluates whether this additional curvature information changes numerical stationarity, convergence behavior, and the balance among the WISSDOM objective components.

### *2.4. Experimental Configuration*

#### 2.4.1. Analysis Case and Common Configuration

The optimizer comparison was conducted using a dual-Doppler wind-retrieval case for Typhoon Bavi on 26 August 2020. Pregridded constant-altitude plan-position indicator (CAPPI) reflectivity and radial-velocity data were obtained from the Korea Meteorological Administration Gosan radar (GSN; 126.1630° E, 33.2943° N; antenna altitude 101 m) and Seongsan radar (SSP; 126.8800° E, 33.3871° N; antenna altitude 68 m). Radar volumes were available at 10 min intervals and were interpolated to the analysis grid using a Cressman weighting function [23]. Observations at 1200 and 1210 Korea Standard Time (KST) were used to define the two analysis times, corresponding to a temporal interval of 600 s.

The analysis domain consisted of 361 × 361 × 41 Cartesian grid points. The horizontal grid spacing was $\Delta x = \Delta y = 1.0$ km, and the vertical grid spacing was $\Delta z = 0.25$ km. The horizontal grid-point span was therefore 360 km in each direction, and the vertical domain extended from 0 to 10 km. The six wind fields at the two analysis times contained a total of 32,058,966 optimization variables.

The background horizontal winds were supplied by the Cloud Resolving Storm Simulator (CReSS) version 3.4 (Tsuboki and Sakakibara), a three-dimensional, non-hydrostatic atmospheric model developed at Nagoya University to simulate clouds and storms [24]. The CReSS fields were preprocessed to match the radar analysis grid and times. They initialized u and v and provided the background constraint where radar observations were unavailable; the initial vertical wind was zero. Vertical profile data extracted from CReSS supplied the reference density and temperature. The CAPPI reflectivity and radial velocities were read from the supplied data files. Terrain and interpolation inputs supported the common boundary treatment and spatial interpolation. Within each typhoon case, the same observations, background fields, and initial wind fields were used by all three solvers.

Within the Bavi comparison, all runs used the same radar data, background and atmospheric fields, interpolation weights, terrain and boundary inputs, objective coefficients, and initial wind fields. Before the first optimizer update, the six optimizer-state fields were identical across the solvers. After the same boundary rules were applied, the projected fields passed to the objective evaluator were also identical, giving the same initial normalized objective $J_0$ = 32,109.2520. After the first update, each solver produced its own sequence of optimizer-state and projected fields.

#### 2.4.2. Optimizer Configurations

The solver-specific configurations are summarized in Table 2. During implementation verification, before the comparative diagnostics were analyzed, preliminary outer-iteration sweeps were used to define fixed solver-specific stopping points based on agreement of the saved projected u, v, and w fields with the established Fortran WISSDOM analysis: 900 outer updates for Adam, 625 outer iterations for L-BFGS, and 350 outer iterations for Newton-CG. These limits were retained for the Typhoon Bavi comparison and transferred unchanged to the Typhoon Lingling case; they do not represent equal computational work or a common mathematical stopping rule. The established Fortran WISSDOM analysis served as the reference solution for implementation verification and for selecting the solver-specific stopping points based on wind-field agreement. It was not treated as independently observed atmospheric truth. Accordingly, comparisons at these stopping points describe the tested solver settings and update limits and should not be interpreted as equal-budget or equal-tolerance optimizer benchmarks.

**Table 2.** Solver-specific configurations used in the WISSDOM experiments.

| Principal Configuration | Outer Iterations | Optimizer |
|---|---|---|
| Base learning rate 0.17; UV and W multipliers 1.80 and 0.22; $(\beta_1, \beta_2) = (0.9, 0.999)$; $\varepsilon = 10^{-8}$; no weight decay | 900 | Adam |
| Five correction pairs; strong-Wolfe line search; nominal trial-step scale 1.0; one internal L-BFGS iteration per outer iteration | 625 | L-BFGS |
| Initial and maximum trust-region radii 1.0 and 1000; acceptance threshold 0.15; maximum 20 inner CG iterations | 350 | Newton-CG |

Note: The L-BFGS value of 1.0 is the nominal trial-step scale supplied to the strong-Wolfe line search, not a fixed accepted step. The Newton-CG initial trust-region radius of 1.0 is a separate quantity. The outer-iteration limits are solver-specific and do not represent equal computational-work budgets.

The selection metric was the root mean square of six component-wise RMSEs for u, v, and w at the two analysis times over the common flow volume, relative to the Fortran analysis. The tested outer-update limits were 400, 500, 600, 700, 800, 850, 875, 900, 925, 950, and 1000 for Adam; 425–650 in steps of 25 for L-BFGS; and 225–350 in steps of 25 for Newton-CG. The selected limits of 900, 625, and 350 gave the smallest metric among the tested candidates: 0.340009, 1.074451, and 1.053518 m $s^{-1}$, respectively. The Newton-CG selection was the largest tested limit; no optimum beyond the tested range is implied.

Preliminary Adam experiments began with a base learning rate of $10^{-3}$ and no block-specific scaling. The final settings were selected from the internal Bavi trials to give the closest agreement with the original Fortran u, v, and w fields among the settings tested. The Fortran fields were an implementation reference, not ground truth. The selected base rate was 0.17, with horizontal- and vertical-wind multipliers of 1.80 and 0.22, giving the effective learning rates

$$\eta_{uv} = 0.17 \times 1.80 = 0.306, \eta_w = 0.17 \times 0.22 = 0.0374. \quad (20)$$

At a base rate of 0.17, the recorded trials used horizontal/vertical multiplier pairs of (1.75, 0.20), (1.75, 0.35), and (1.80, 0.22). Another recorded trial used effective rates of about (0.298, 0.068). These were preliminary trials, not a full search of all combinations.

The standard PyTorch Adam moment parameters, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and the numerical-stability constant $\varepsilon = 10^{-8}$ were used without weight decay or learning-rate scheduling. The learning rates were selected in the preliminary Bavi trials and are not presented as universally optimal. The final optimizer settings were also used unchanged for the Lingling case.

L-BFGS retained five correction pairs, consistent with the memory depth used in the established WISSDOM configuration, and used the PyTorch strong-Wolfe line search. One internal L-BFGS iteration was performed during each reported outer iteration. The value 1.0 supplied as the line-search step parameter was a nominal trial-step scale rather than a fixed accepted step length. The initial trial step could be rescaled internally, and the final accepted step was determined by the strong-Wolfe conditions. The gradient and parameter-change tolerances were $10^{-7}$ and $10^{-9}$, respectively. Because the line search could evaluate the objective and gradient multiple times, the outer-iteration count was not treated as the number of objective evaluations.

Newton-CG used an initial trust-region radius of 1.0, a maximum radius of 1000, and a trial-step acceptance threshold of 0.15. The Steihaug CG calculation was limited to 20 inner iterations. At outer iteration k, its residual stopping target was

$$\tau_{CG,k} = \max\left[10^{-10}, \min\left(0.5, \|g_k\|_2{}^{1/2}\right)\|g_k\|_2\right]. \tag{21}$$

Here, $g_k$ is the current full-objective gradient. Curvature truncation was applied when

$$d^T H_k \mathrm{d} \leq 10^{-12}\|d\|_2{}^2. \tag{22}$$

Thus, this condition included both nonpositive curvature and positive curvature that was numerically too small to support a reliable CG step. The Adam learning rate, its horizontal-wind and vertical-wind multipliers, and the L-BFGS line-search step parameter were not used by Newton-CG.

#### 2.4.3. Execution Configurations

The variational optimization was performed on a single NVIDIA GB10 GPU using PyTorch 2.9.1+cu130, CUDA 13.0, and NVIDIA driver 580.173.02, with Python 3.12.3 on Linux 6.17.0-1029-nvidia. The system was an MSI EdgeXpert MS-C931 with 128 GB of LPDDR5x unified memory shared by the CPU and GPU. The optimization variables and floating-point tensor calculations used 32-bit floating-point (FP32) precision. Integer arrays used for indexing and interpolation were retained in their appropriate integer formats. The objective calculation, automatic differentiation, and optimizer updates ran on this GPU. The separate memory measurements used NVIDIA driver 580.178.04 on the same GPU, with the same PyTorch and CUDA versions.

Runs used either detailed or reduced diagnostic output. The runs with detailed diagnostic output retained the detailed per-iteration scientific diagnostics required to examine convergence, numerical stationarity, objective-component evolution, and optimizer-specific behavior. The runs with reduced diagnostic output used the same numerical inputs, physical objective, optimizer parameters, floating-point precision, and solver-update rules but omitted per-iteration gradient, component, and update-norm diagnostics, the per-run optimizer-convergence figures, and the additional final physical-diagnostics report. Standard WISSDOM outputs and radial-velocity verification remained enabled. The runs with reduced diagnostic output retained the post-step objective, computational-work

counters, and vertical-wind range checks and were used for low-overhead computational-performance experiments. The evaluation and timing procedures for these runs are described in Section 2.5.

The comparison used the same PyTorch objective evaluator with compilation disabled for all three solvers. Adam used reverse-mode gradients in its first-order update. L-BFGS used the same gradients to form its limited-memory quasi-Newton inverse-Hessian approximation. Newton-CG additionally used higher-order automatic differentiation to compute Hessian-vector products. This configuration was used to examine algorithm-dependent convergence and computational work without introducing solver-specific compilation effects.

A separate set of timing runs enabled cost-function compilation for Adam and L-BFGS [25]. Adam and L-BFGS used compiled cost evaluators, whereas Newton-CG used uncompiled PyTorch execution for its higher-order automatic-differentiation path. All three solvers ran on the same NVIDIA GPU through CUDA. These results were interpreted as solver-implementation performance rather than backend-independent algorithmic speed. The timing procedures for both compilation settings are described in Section 2.5.

### *2.5. Evaluation and Monitoring Procedures*

The evaluation considered three aspects separately: numerical convergence, agreement and physical consistency of the retrieved wind fields, and computational cost. These measures were not combined into a single score because they describe different properties of the solution.

#### 2.5.1. Numerical Convergence and Objective Components

The objective history was normalized by the common initial value as

$$R_{J,k} = \frac{J_k}{J_0}, \tag{23}$$

where $J_0$ is the normalized objective at the common initial state and $J_k$ is the normalized objective after outer iteration $k$. The absolute normalized objective values were also retained.

Because a low objective does not necessarily indicate that the solution is close to a stationary point, the gradient norm was evaluated separately. A stationary point is a state at which the local gradient approaches zero, meaning that sufficiently small changes to the wind variables are not predicted to produce a first-order reduction in the objective. A small gradient is a necessary condition for a local minimum of a differentiable objective, but it does not establish a global minimum or a physically true wind field [10].

The gradient was divided into horizontal- and vertical-wind blocks,

$$g_{uv,k} = \left(g_{u_1,k}, g_{v_1,k}, g_{u_2,k}, g_{v_2,k}\right), g_{w,k} = \left(g_{w_1,k}, g_{w_2,k}\right). \tag{24}$$

The relative full, horizontal-wind, and vertical-wind gradient norms were

$$R_{g,k} = \frac{\|g_k\|_2}{\|g_0\|_2}, R_{uv,k} = \frac{\|g_{uv,k}\|_2}{\|g_{uv,0}\|_2}, R_{w,k} = \frac{\|g_{w,k}\|_2}{\|g_{w,0}\|_2}. \tag{25}$$

In the runs with detailed diagnostic output, the objective, its individual components, and these gradient norms were recalculated after every completed optimizer step using the same uncompiled PyTorch evaluation procedure for all three solvers. The objective and its components were evaluated from the projected fields, whereas the gradient norms were calculated with respect to the six optimizer-state fields defined in Section 2.2. Because the boundary treatment remained within the automatic-differentiation graph, its effects on

the projected fields were included in these gradients. The value returned directly by an optimizer was not used because optimizer implementations can return objectives evaluated at different points within an update. The common monitoring calculations were excluded from optimizer time.

In the runs with reduced diagnostic output, the projected post-step objective was retained after every update, whereas the complete gradient and component evaluation was performed only at the initial and final states. This reduced monitoring overhead while preserving a common definition of the recorded objective.

Each raw reporting component $J_{c,k}^{\mathrm{raw}}$ was converted to a normalized component using

$$J_{c,k} = \frac{J_{c,k}^{\mathrm{raw}}}{2N_{\mathrm{grid}}}. \tag{26}$$

For reporting, the implementation output was grouped into eight categories: radial velocity, mass continuity, vertical vorticity, background wind, Laplacian smoothing and height-dependent damping, reflectivity conservation, boundary, and domain-mean velocity damping. In the present configuration, the separately recorded reflectivity-conservation and domain-mean damping categories were zero and did not change the seven-term objective in Equation (2). Height-dependent pointwise damping was included in the Laplacian category. At each monitored state, the normalized category values were summed and checked against the total objective using a tolerance for FP32 calculations. Runs that failed this check were not used in the scientific comparison.

#### 2.5.2. Wind-Field Agreement and Physical Evaluation

The wind-field and physical evaluations used the projected final iterate stored by the standard WISSDOM output procedure, rather than the optimizer-state fields. The projected fields were compared at the same analysis times, heights, and grid points using common atmospheric-flow and observation masks. L-BFGS was used as the algorithmic baseline because it corresponds to the established WISSDOM optimization procedure, but it was not treated as observational truth. Differences between optimizer solutions were therefore described as pairwise RMS differences or algorithm-induced differences rather than retrieval errors.

For each pair of solvers, component RMS differences were calculated as the square root of the spatial mean of $(q^A - q^B)^2$ for $q \in \{u,v,w\}$ over the full three-dimensional common flow volume and separately at each vertical level. The three-dimensional wind-vector RMS difference over the full common volume was the square root of the spatial mean of $(u^A - u^B)^2 + (v^A - v^B)^2 + (w^A - w^B)^2$. Spatial correlations were evaluated over the same grid cells. Regional comparisons used a common precipitation mask constructed from the shared observations with reflectivity of at least 20 dBZ. Vertical-velocity distributions were compared using the 1st, 5th, 50th, 95th, and 99th percentiles, RMS magnitude, and absolute maximum.

Following the measures used in established WISSDOM studies [2,6], the fit to radar observations was evaluated using root-mean-square error (RMSE), relative RMSE (RRMSE), and spatial correlation coefficient (SCC). For radar $r$ and analysis time $t$, let $V_{r,t,i}^{\mathrm{obs}}$ denote observed radial velocity and $V_{r,t,i}^{\mathrm{ana}}$ denote radial velocity calculated by projecting the retrieved wind onto the corresponding radar beam. For the set $\mathcal{V}_{r,t}$ of atmospheric grid cells where both values were available,

$$\mathrm{RMSE}_{r,t} = \left[ \frac{1}{N_{r,t}} \sum_{i \in \mathcal{V}_{r,t}} \left( V_{r,t,i}^{\mathrm{ana}} - V_{r,t,i}^{\mathrm{obs}} \right)^2 \right]^{1/2}, \tag{27}$$

$$\mathrm{RRMSE}_{r,t} = \left[ \frac{\sum_{i \in \mathcal{V}_{r,t}} \left( V^{\mathrm{ana}}_{r,t,i} - V^{\mathrm{obs}}_{r,t,i} \right)^2}{\sum_{i \in \mathcal{V}_{r,t}} \left( V^{\mathrm{obs}}_{r,t,i} \right)^2} \right]^{1/2}, \tag{28}$$

and

$$\mathrm{SCC}_{r,t} = \frac{\sum_{i \in \mathcal{V}_{r,t}} \left( V^{\mathrm{ana}}_{r,t,i} - \overline{V}^{\mathrm{ana}}_{r,t} \right) \left( V^{\mathrm{obs}}_{r,t,i} - \overline{V}^{\mathrm{obs}}_{r,t} \right)}{\left[ \sum_{i \in \mathcal{V}_{r,t}} \left( V^{\mathrm{ana}}_{r,t,i} - \overline{V}^{\mathrm{ana}}_{r,t} \right)^2 \sum_{i \in \mathcal{V}_{r,t}} \left( V^{\mathrm{obs}}_{r,t,i} - \overline{V}^{\mathrm{obs}}_{r,t} \right)^2 \right]^{1/2}}. \tag{29}$$

Here, $N_{r,t}$ is the number of valid grid cells and the overbars denote spatial means over those cells. SCC denotes the mean-centered Pearson correlation coefficient evaluated across the valid spatial samples. RMSE was reported in $\mathrm{ms}^{-1}$, and RRMSE was retained as a dimensionless ratio; values presented as percentages were reported as 100 × RRMSE. The three statistics were calculated separately for each radar and analysis time over the three-dimensional volume and at each vertical level.

The radial velocities from both radars were included in the objective function. These statistics therefore describe the fit of the final analysis to assimilated observations and do not represent independent validation.

For the summary comparison, the four three-dimensional RMSE values from the two radars and two analysis times were combined using their root mean square. SCC was summarized by the arithmetic mean of the four values, and RRMSE was reported as their minimum-to-maximum range.

Physical consistency was assessed from the mass-continuity, vertical-vorticity-equation, and reflectivity-conservation residuals, together with departure from the background wind. Continuity, vorticity, and reflectivity residual RMS values were evaluated over common valid flow cells. Background-departure RMS was evaluated over common flow cells where radar data were not available. Among the two boundary constraints, only the lower-boundary vertical-velocity constraint had a nonzero coefficient in this experiment. Its weighted and normalized objective contribution was reported; a separate boundary-residual RMS was not calculated. The density from the vertical profile data was used directly, without normalization. Taking its unit as kg $\mathrm{m}^{-3}$, the continuity-residual RMS is in kg $\mathrm{m}^{-3}$ $\mathrm{s}^{-1}$. The vorticity and reflectivity residual RMS values are in $\mathrm{s}^{-2}$ and dBZ $\mathrm{s}^{-1}$, respectively; the reflectivity residual was calculated from the dBZ field.

#### 2.5.3. Computational Work, Timing and Memory

Computational work was described using separate counts of objective evaluations, full-gradient evaluations, and Hessian-vector products. These counts were not combined into a single weighted score because the three operations have different computational costs. Evaluations used only for common monitoring were recorded separately from work requested by the optimizers.

Optimizer time was measured around each complete optimizer update after GPU synchronization. It included objective and derivative calculations requested by the optimizer, the L-BFGS line search, Newton-CG Hessian-vector products and trust-region calculations, and the parameter update. Common monitoring, reporting, preprocessing, postprocessing, and figure generation were excluded. The reported value therefore represents synchronized optimizer-update time rather than pure GPU-kernel time.

Timing was measured with compilation disabled for all solvers and, separately, with compilation enabled for Adam and L-BFGS only (Section 2.4.3). Compilation initiated during the first optimizer update was included in the measured time.

Each solver was executed five times under each compilation setting with reduced diagnostic output, giving 30 timing runs. Every run started as a separate Python process on the same GPU, and previously generated compilation files were not reused in the compiled experiments. The solver order was rotated among repetitions. Before each timing run, nvidia-smi reported no other GPU compute processes. Optimizer time was summarized using the median and interquartile range.

The repeated timing experiments report the total optimizer time to each solver-specific stopping point. These times therefore compare the configured WISSDOM analyses and do not represent equal computational-work budgets or one common mathematical stopping rule.

As a complementary post hoc comparison, the remaining fraction of the initial-to-reference objective gap was defined as $r = (J_{target} - J_{ref})/(J_0 - J_{ref})$. The common targets used $r = 0.01$ and 0.001, with $J_0 = 32{,}109.251953125$, and $J_{ref} = 1824.1944580078125$. $J_{ref}$ was the lowest objective recorded across six runs with compilation disabled: one run per solver with detailed diagnostic output and one with reduced diagnostic output, using the same stopping limits. The repeated timing runs were excluded; this reference is not a known minimum. For each timing run, the first recorded objective at or below the target was paired with the cumulative synchronized optimizer time, without interpolation. A target not reached within the configured update limit was recorded as not reached. This analysis uses existing histories and does not represent runs stopped by a common tolerance.

We also compared the objective at common elapsed-time budgets of 300 and 600 s using the existing five timing histories per solver with compilation disabled for all solvers. This clock covers the optimization loop, including initial diagnostics and monitoring, but excludes input preparation; it differs from the cumulative optimizer-update time used for the common-target comparison. For each budget, we selected the last completed record at or before that time, without interpolation. All runs continued beyond 600 s, so neither comparison used a result extended past the end of a run. The objective was summarized by the median and full range across the five runs.

In a separate memory test, each solver was run once for Typhoon Bavi with compilation disabled, using the same settings and update limits as the timing runs. PyTorch peak counters were reset after CUDA initialization and covered preprocessing, optimization, and postprocessing. Peak allocated memory measures memory occupied by tensors; peak reserved memory includes the PyTorch memory cache. These values exclude other processes and memory outside the PyTorch CUDA allocator and do not represent total GPU memory use. Other GPU tasks were present, so these runs were not used for timing comparisons.

## 3. Results

All three runs with detailed diagnostic output used the same initial wind field after the standard boundary conditions were applied. The normalized objective was $J_0 = 32{,}109.252$ in every run. The initial objective terms and gradient norms were also identical, and the terms summed to the total objective within numerical tolerance. Unless noted otherwise, the wind comparisons use the final fields saved by the standard WISSDOM output step, not the internal fields held by each optimizer.

### *3.1. Objective Reduction and Gradient Behavior*

All three solvers greatly reduced the objective, but their gradient norms changed differently (Figure 1 and Table 3). At their planned stopping points, the final objectives were 1858.007 for Adam, 1826.949 for L-BFGS, and 1824.194 for Newton-CG. These values correspond to reductions of 94.21%, 94.31%, and 94.32% from the common initial objective,

respectively. The L-BFGS objective was only 0.151% higher than the Newton-CG value, whereas the Adam objective was 1.70% higher than the L-BFGS value and 1.85% higher than the Newton-CG value.

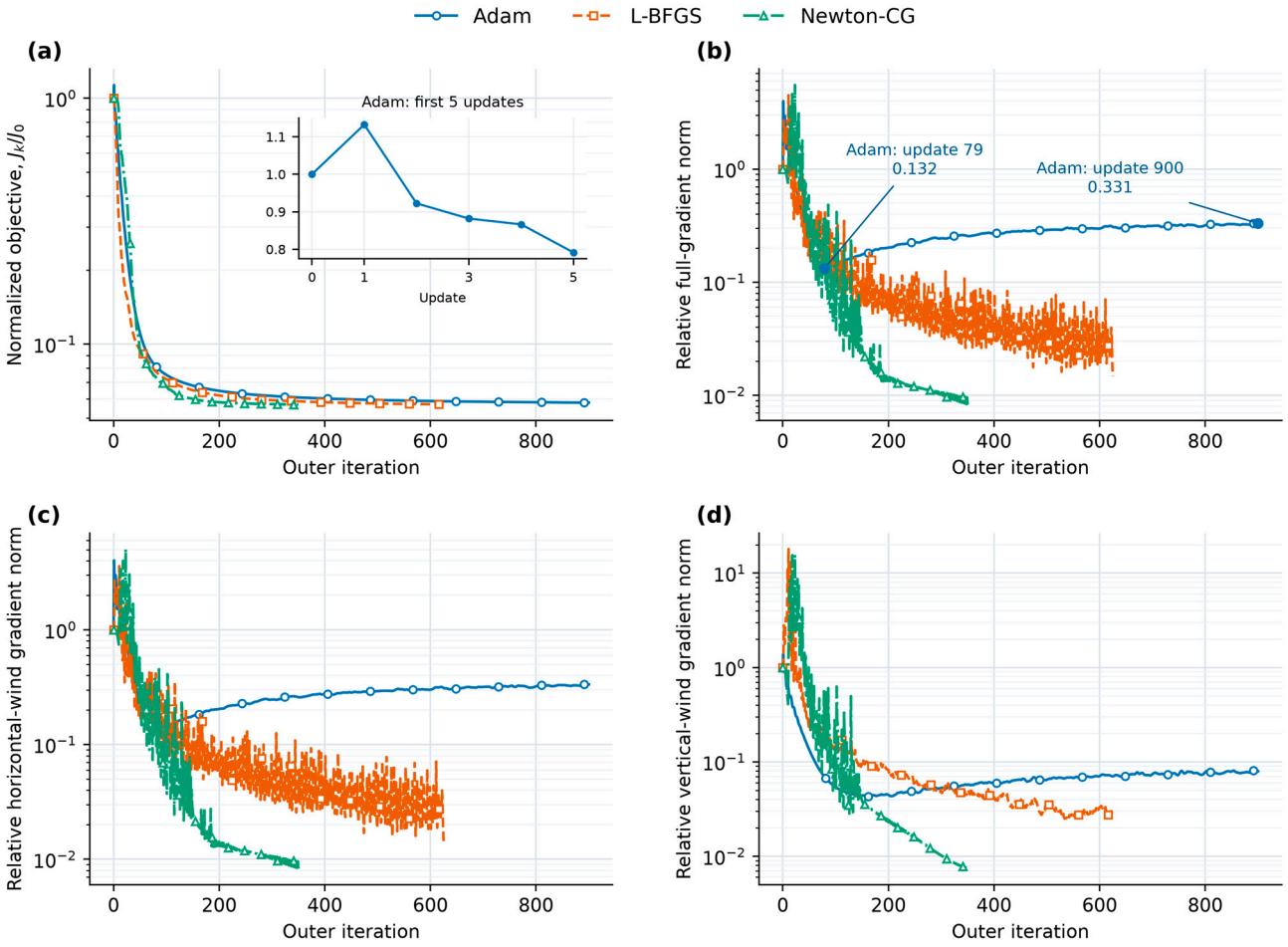



**Figure 1.** Convergence histories from the runs with detailed diagnostic output. (**a**) Normalized objective; (**b**) relative full-gradient norm; (**c**) relative horizontal-wind gradient norm; and (**d**) relative vertical-wind gradient norm. The solver-specific iteration limits were selected as described in Section 2.4 and do not represent equal computational-work budgets. The inset in panel (**a**) shows the first five updates. The markers in panel (**b**) identify Adam at updates 79 and 900; the minimum gradient norm is not the minimum objective.

**Table 3.** Final numerical measures and optimizer work in the runs with detailed diagnostic output.

| HVP | g Eval. | f Eval. | Rel. W Grad. Norm | Rel. UV Grad. Norm | Rel. Full Grad. Norm | Final J | Outer Limit | Solver |
|---|---|---|---|---|---|---|---|---|
| 0 | 900 | 900 | 0.07946 | 0.3348 | 0.3310 | 1858.007 | 900 | Adam |
| 0 | 1276 | 1276 | 0.02738 | 0.01429 | 0.01474 | 1826.949 | 625 | L-BFGS |
| 5162 | 350 | 700 | 0.007390 | 0.008281 | 0.008261 | 1824.194 | 350 | Newton-CG |

Note: f, g, and HVP denote optimizer-requested objective, full-gradient, and Hessian-vector-product evaluations, respectively. These counts belong to the runs with detailed diagnostic output and are not combined with the repeated timing medians reported later.

For Adam, after a brief rise at the first update, the relative full-gradient norm fell to 0.132 at update 79 and rose to 0.331 at update 900, while the objective decreased overall after its initial rise. This rebound was mainly associated with the horizontal-wind gradient

block. L-BFGS and Newton-CG also showed large early changes in the gradient norm, but both subsequently approached substantially smaller gradients. Their final relative full-gradient norms were 0.0147 and 0.00826, respectively.

The final relative full-gradient norm for Adam was 22.5 times that for L-BFGS and 40.1 times that for Newton-CG. Thus, similar final objective values did not mean that the final gradient norms were equally small. Adam ended with an objective close to those of L-BFGS and Newton-CG, but its final gradient norm remained larger. A smaller gradient norm means that the local slope of the objective is closer to zero. This result alone does not show that additional fixed-learning-rate Adam updates would produce a better wind field.

Table 3 also lists Hessian-vector-product (HVP) evaluations. Only Newton-CG required them.

### *3.2. Comparison of the Final WISSDOM Wind Fields*

Although their final gradient norms differed, the three solvers produced similar large-scale wind patterns in the final WISSDOM output. Figure 2 compares the horizontal wind speed and vectors from the three solvers at 5 km using one common wind-speed range and one arrow scale. The common flow domain includes only grid cells with valid wind values from all three solvers. The L-BFGS and Newton-CG fields were the most similar: their three-dimensional wind-vector RMS differences over this domain were 0.425 and 0.444 m $s^{-1}$ at the two analysis times. Their spatial correlations for the individual u, v, and w components ranged from 0.9959 to 0.9997.

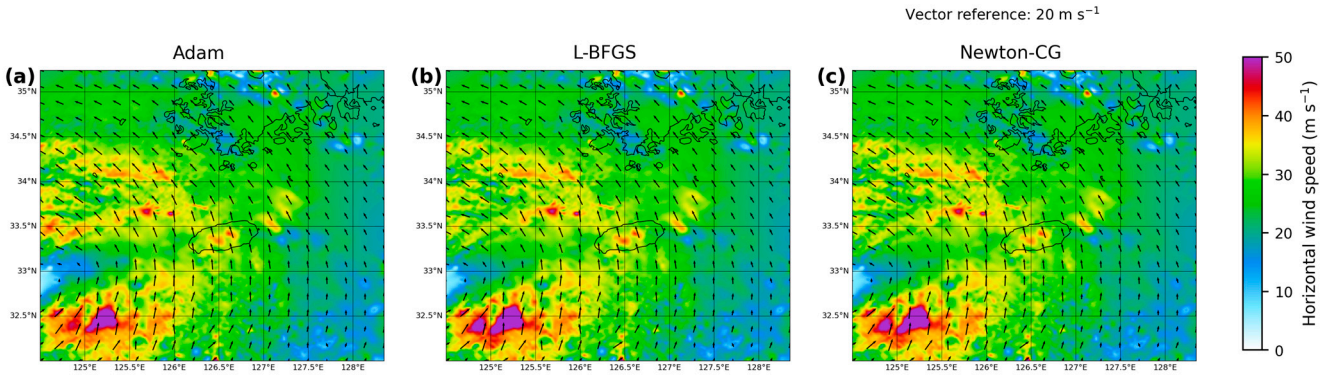



**Figure 2.** Final exported horizontal wind fields at the first analysis time and 5 km height: (**a**) Adam, (**b**) L-BFGS, and (**c**) Newton-CG. Colors show horizontal wind speed using the same 0–50 m $s^{-1}$ range, and arrows show horizontal wind direction and magnitude using a common 20 m $s^{-1}$ reference. L-BFGS is used only as a comparison baseline, not as truth. The L-BFGS baseline is from the common Python implementation.

Adam retained the same main flow structures but showed larger differences from the L-BFGS baseline. The Adam–L-BFGS three-dimensional vector RMS differences were 1.600 and 1.643 m $s^{-1}$ over the common flow domain. Within the common region having reflectivity of at least 20 dBZ, these differences increased to 1.849 and 1.941 m $s^{-1}$. The differences between L-BFGS and Newton-CG remained below 0.5 m $s^{-1}$ in both masks.

Table 4 separates the differences in u, v, and w. For comparisons involving Adam, the largest RMS difference was in v, followed by u and then w. Over the two analysis times, the Adam–L-BFGS RMS differences were 0.829–0.872 m $s^{-1}$ for u, 1.324–1.351 m $s^{-1}$ for v, and 0.339–0.345 m $s^{-1}$ for w. Thus, the three-dimensional difference was not dominated by vertical velocity. Figure 3 examines w separately because vertical velocity is more weakly constrained by Doppler observations and is an important retrieved component [5].

**Table 4.** Pairwise RMS differences in the final WISSDOM wind fields.

| u/v/w Correlation Range | 3D RMS, dBZ $\geq$ 20 | 3D RMS | w RMS | v RMS | u RMS | Time | Solver Pair |
|---|---|---|---|---|---|---|---|
| 0.9722–0.9975 | 1.849 | 1.600 | 0.345 | 1.324 | 0.829 | 1 | Adam–L-BFGS |
| 0.9732–0.9972 | 1.941 | 1.643 | 0.339 | 1.351 | 0.872 | 2 | Adam–L-BFGS |
| 0.9764–0.9980 | 1.743 | 1.518 | 0.319 | 1.286 | 0.742 | 1 | Adam–Newton-CG |
| 0.9762–0.9978 | 1.825 | 1.557 | 0.317 | 1.309 | 0.780 | 2 | Adam–Newton-CG |
| 0.9959–0.9997 | 0.444 | 0.425 | 0.149 | 0.266 | 0.297 | 1 | L-BFGS–Newton-CG |
| 0.9962–0.9996 | 0.481 | 0.444 | 0.146 | 0.282 | 0.311 | 2 | L-BFGS–Newton-CG |

Note: All RMS values are in m $s^{-1}$. The u, v, w, and 3D RMS values are calculated over the full three-dimensional common flow volume; the dBZ column uses the common region with reflectivity of at least 20 dBZ. The final column gives the range of the separate u, v, and w correlations over the common flow domain. These are differences between solver results, not errors relative to truth.

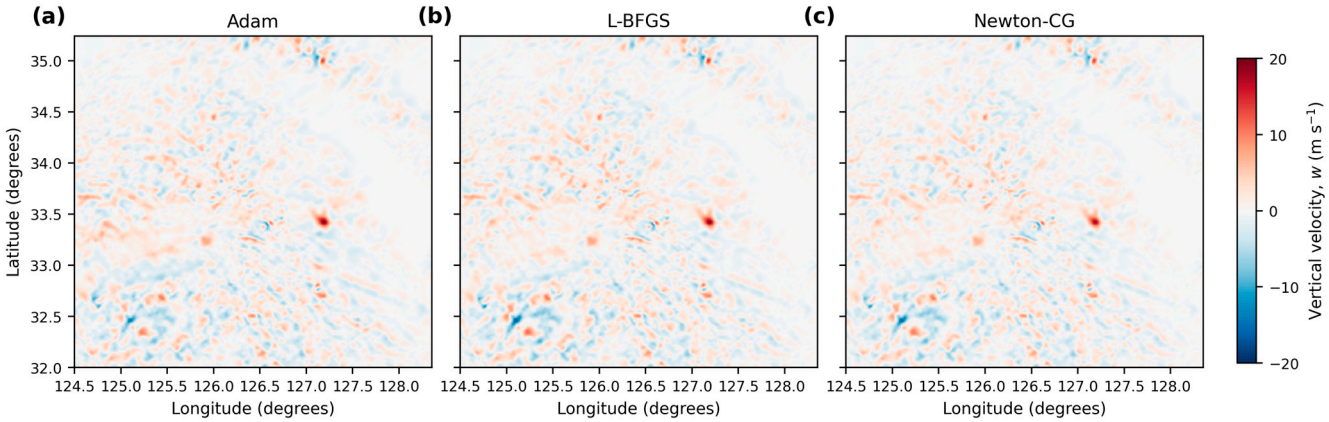



**Figure 3.** Final exported vertical velocity at the first analysis time and 5 km height from (**a**) Adam, (**b**) L-BFGS, and (**c**) Newton-CG. All panels use the same fixed range of −20 to 20 m $s^{-1}$. Red shows rising air and blue shows sinking air.

The three vertical-velocity fields in Figure 3 have nearly the same spatial pattern on this common scale. At 5 km, the RMS difference from L-BFGS was 0.424 m $s^{-1}$ for Adam and 0.174 m $s^{-1}$ for Newton-CG over the common flow domain. These values compare solver results; they are not errors relative to truth. Maps that enlarge the small solver differences are provided in Figure A2.

The final vertical-velocity distributions were centered close to zero for all three solvers (Figure A1). Over all vertical levels in the common flow domain, the vertical-velocity RMS values at the first analysis time were 1.465, 1.467, and 1.391 m $s^{-1}$ for Adam, L-BFGS, and Newton-CG, respectively. Newton-CG produced slightly fewer grid cells with large upward or downward velocities. This shows that solver choice affected some of the strongest vertical motions, but it does not show which result is closer to the real atmosphere.

### *3.3. Objective-Component Balance and Physical Diagnostics*

Although the total objectives were similar, the weighted objective terms differed among the solvers (Figure 4). Equation (2) contains seven terms. The upper-boundary term had zero coefficient in this experiment, whereas the active lower-boundary term is included in the boundary category. Figure 4 therefore shows the six nonzero contributions. The separately recorded reflectivity-conservation and domain-mean damping categories were also zero and are omitted. L-BFGS produced the smallest radar-observation contribution. Adam produced the smallest continuity contribution, whereas Newton-CG produced the smallest vertical-vorticity, background, combined Laplacian and height-dependent

damping, and boundary contributions. Newton-CG therefore obtained the lowest total objective even though its radar contribution was 0.75% higher than that of L-BFGS.

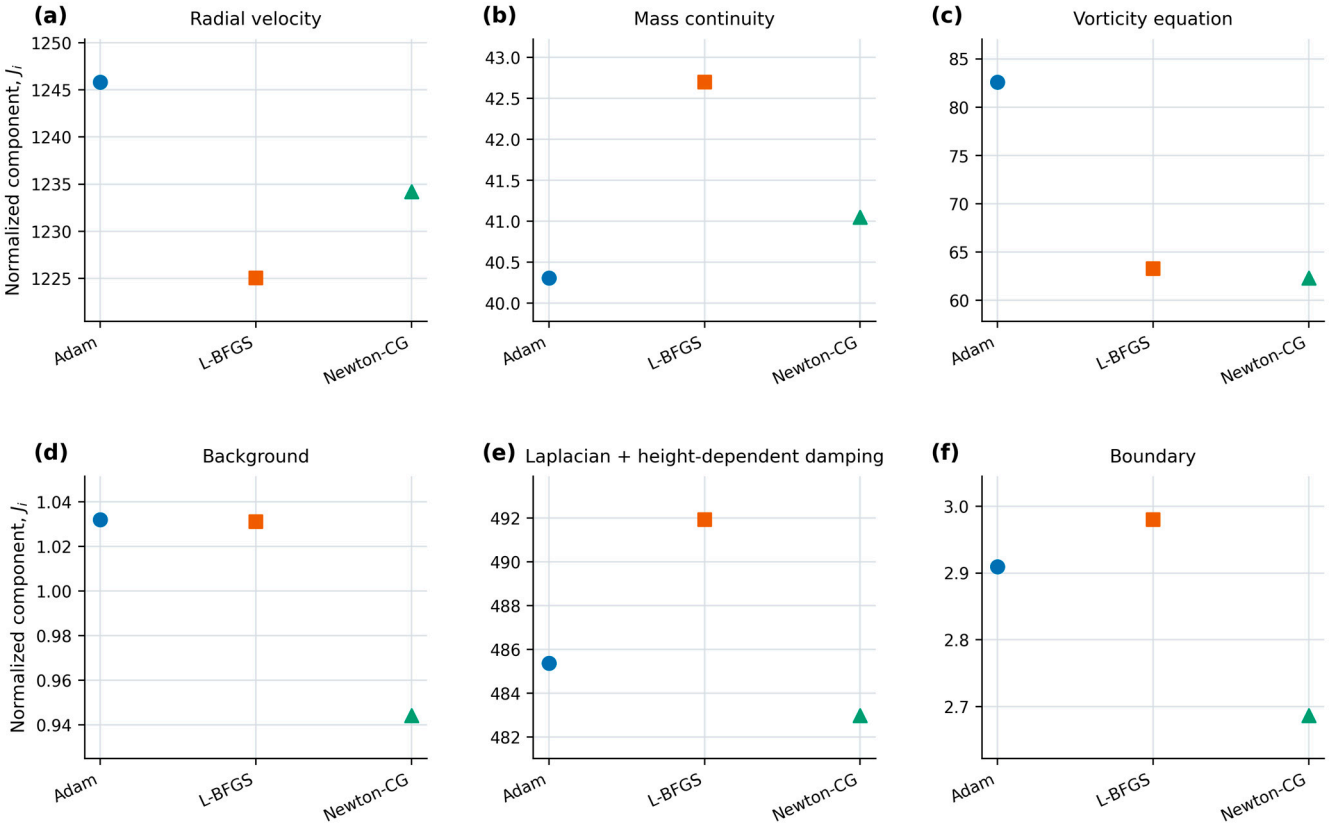



**Figure 4.** Final normalized contributions of the six nonzero objective terms: (**a**) radial velocity; (**b**) mass continuity; (**c**) vertical vorticity; (**d**) background wind; (**e**) Laplacian smoothing and height-dependent damping; and (**f**) lower-boundary penalty. Equation (2) contains seven terms. The upper-boundary term had zero coefficient, whereas the active lower-boundary term is included in the boundary category. The Laplacian category includes the active height-dependent damping. The separately recorded reflectivity-conservation and domain-mean damping categories were also zero and are omitted.

All three solvers fit the radial-velocity observations similarly (Table 5). The summary three-dimensional Vr RMSE values were 1.313 m $s^{-1}$ for Adam, 1.302 m $s^{-1}$ for L-BFGS, and 1.307 m $s^{-1}$ for Newton-CG. Mean spatial correlation coefficient (SCC) values ranged only from 0.998108 to 0.998140, while radar- and time-specific relative RMSE (RRMSE) values ranged from 5.33% to 6.85%. The same observations were used during optimization, so these statistics are not an independent validation of the wind field.

No solver was best for every physical diagnostic. Adam had the smallest continuity-residual RMS at its final update 900, whereas Newton-CG had the smallest vertical-vorticity-equation residual, background departure, reflectivity-conservation residual, and normalized boundary penalty. The reflectivity-conservation term had zero coefficient in the objective used here, but its unweighted residual is reported as an additional consistency measure. Reflectivity was also used to estimate hydrometeor fall velocity, so this diagnostic is not independent validation.

**Table 5.** Final observation-fit and physical-consistency measures. The final updates were 900 for Adam, 625 for L-BFGS, and 350 for Newton-CG.

| Boundary J | Reflectivity RMS | Background RMS (m $s^{-1}$) | Vorticity RMS ($s^{-2}$) | Continuity RMS | Mean SCC | RRMSE Range (%) | Vr RMSE (m $s^{-1}$) | Solver |
|---|---|---|---|---|---|---|---|---|
| 2.910 | 0.025923 | 2.647 | $2.037 \times 10^{-6}$ | $3.181 \times 10^{-4}$ | 0.998108 | 5.365–6.852 | 1.3132 | Adam |
| 2.980 | 0.025954 | 2.646 | $1.783 \times 10^{-6}$ | $3.274 \times 10^{-4}$ | 0.998140 | 5.327–6.783 | 1.3020 | L-BFGS |
| 2.687 | 0.025898 | 2.532 | $1.769 \times 10^{-6}$ | $3.211 \times 10^{-4}$ | 0.998127 | 5.345–6.811 | 1.3069 | Newton-CG |

Note: Vr statistics show agreement with observations used during optimization. Boundary J is the normalized weighted boundary contribution, not a lower-boundary vertical-velocity RMS. Continuity RMS is in kg $m^{-3}$ $s^{-1}$ when the density from the vertical profile data is interpreted in kg $m^{-3}$. Reflectivity RMS is in dBZ $s^{-1}$.

### *3.4. Computational Cost and Solver Trade-Offs*

With compilation disabled for all solvers and, separately, enabled for Adam and L-BFGS only (Section 2.5.3), the timing experiments showed a clear trade-off between the time required to complete the configured retrieval and the numerical convergence reported in Section 3.1 (Figure 5 and Table 6). With compilation disabled for all solvers, the median total optimizer times were 13.42 min for Adam, 19.21 min for L-BFGS, and 72.54 min for Newton-CG. With compilation enabled for Adam and L-BFGS only, the corresponding times were 2.60, 3.67, and 72.57 min. Each value is the median of five separate runs, and the interquartile ranges were small relative to the differences among the solvers.

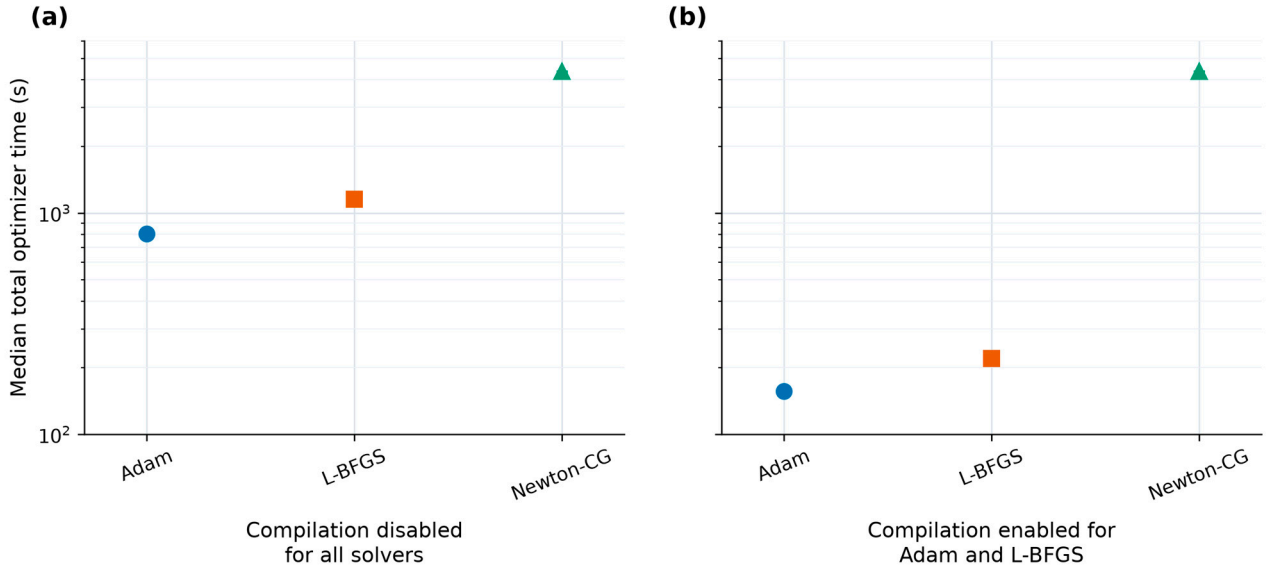



**Figure 5.** Total optimizer time across five repetitions with (**a**) compilation disabled for all solvers and (**b**) compilation enabled for Adam and L-BFGS only. Markers show the median, and error bars show the interquartile range. The vertical axis uses a logarithmic scale. Compilation startup time is included where compilation is enabled. Newton-CG runs without compilation in both sets of runs.

**Table 6.** Median total optimizer time across five repetitions.

| Median (Min) | Q1–Q3 (s) | Median (s) | Solver | Compilation Enabled for |
|---|---|---|---|---|
| 13.42 | 790.29–805.15 | 805.12 | Adam | None |
| 19.21 | 1134.15–1156.98 | 1152.44 | L-BFGS | None |
| 72.54 | 4291.75–4363.09 | 4352.57 | Newton-CG | None |
| 2.60 | 156.26–156.42 | 156.30 | Adam | Adam and L-BFGS |
| 3.67 | 219.37–220.57 | 220.25 | L-BFGS | Adam and L-BFGS |
| 72.57 | 4346.31–4360.64 | 4353.92 | Newton-CG | Adam and L-BFGS |

Note: These times cover the complete solver-specific stopping limits and do not represent equal computational-work budgets or a common mathematical stopping rule. Final objectives, gradient norms, and work counts from the runs with detailed diagnostic output are reported separately in Table 3. Newton-CG runs without compilation in both settings.

The first set of timing runs disabled compilation for all three solvers. The second enabled compilation for Adam and L-BFGS only and included compilation startup time. Compilation reduced the total optimizer times of Adam and L-BFGS by factors of 5.15 and 5.23, respectively. Newton-CG time changed little because its HVP calculations used uncompiled PyTorch execution. The results with compilation enabled for Adam and L-BFGS therefore describe the performance of the implemented solver configurations rather than a comparison of theoretical algorithmic complexity.

The total times cover the solver-specific stopping limits selected in Section 2.4. They therefore represent the time required by each solver to produce its configured WISSDOM analysis, not the time required to satisfy one common mathematical stopping rule. The timing runs with compilation enabled for Adam and L-BFGS showed that Adam required the least time. In the separate run with detailed diagnostic output, Adam had the largest final relative full-gradient norm. L-BFGS required 1.41 times the Adam time in the timing runs with compilation enabled for Adam and L-BFGS and obtained a much smaller final relative full-gradient norm and an objective close to that of Newton-CG in the run with detailed diagnostic output. Newton-CG achieved the smallest final objective and relative full-gradient norm in the run with detailed diagnostic output, but its median optimizer time in the timing runs with compilation enabled for Adam and L-BFGS was 19.8 times that of L-BFGS. The Newton-CG/L-BFGS median time ratio was 3.78 with compilation disabled for all solvers and 19.77 with compilation enabled for Adam and L-BFGS.

The timing medians and the detailed convergence measures came from separate sets of runs using the same solver settings. Table 6 reports only the repeated timing experiments, whereas Table 3 reports the final objective, gradient norms, and computational-work counts from the runs with detailed diagnostic output. The results should therefore be interpreted together as results for the tested solver settings rather than as runtime and solution measures obtained from the same runs.

At the less stringent target, Adam was fastest under both compilation settings (Table 7). At the tighter target, only L-BFGS and Newton-CG reached the target within their configured limits, with L-BFGS requiring less time. The cost ranking therefore depends on the requested objective level; non-attainment here is limited to the tested update budget.

**Table 7.** First recorded attainment of common objective targets in the repeated timing runs.

| Compilation Enabled for | Solver | r = 0.01: Time (s), n/5 | r = 0.001: Time (s), n/5 |
|---|---|---|---|
| None | Adam | 151.81 (5/5) | NR (0/5) |
| None | L-BFGS | 255.19 (5/5) | 797.24 (5/5) |
| None | Newton-CG | 709.30 (5/5) | 2353.69 (5/5) |
| Adam and L-BFGS | Adam | 63.41 (5/5) | NR (0/5) |
| Adam and L-BFGS | L-BFGS | 80.80 (5/5) | 166.07 (5/5) |
| Adam and L-BFGS | Newton-CG | 711.35 (5/5) | 2357.20 (5/5) |

Note: $r = (J_{target} - J_{ref})/(J_0 - J_{ref})$, where $J_0$ = 32,109.251953125 and $J_{ref}$ = 1824.1944580078125. The targets are $J_{target}$ = 2127.045033 for r = 0.01 and 1854.479516 for r = 0.001. Times are medians of cumulative synchronized optimizer-update time among runs reaching the target; parentheses give the number of runs reaching the target out of five. NR means the target was not reached within the configured update limit. Monitoring and input preparation are excluded. Compilation startup is included when enabled; Newton-CG runs without compilation in both settings.

At common elapsed-time budgets of 300 and 600 s, Adam had the lowest median objective, followed by L-BFGS and Newton-CG (Table 8). The ranges for Adam and L-BFGS overlapped at 600 s; these results do not establish a statistically significant difference or a difference in wind-field accuracy. Table 7 measures the time needed to reach a common objective level, whereas Table 8 compares the objective reached within a common elapsed-time budget.

**Table 8.** Objective values at common elapsed-time budgets with compilation disabled.

| Solver | Time Budget (s) | Median J | Minimum J | Maximum J |
|---|---|---|---|---|
| Adam | 300 | 1987.81 | 1981.23 | 1996.88 |
| L-BFGS | 300 | 2092.74 | 2081.80 | 2095.63 |
| Newton-CG | 300 | 2888.54 | 2888.54 | 2888.54 |
| Adam | 600 | 1889.78 | 1887.08 | 1898.76 |
| L-BFGS | 600 | 1900.33 | 1897.67 | 1902.17 |
| Newton-CG | 600 | 2222.40 | 2222.40 | 2222.40 |

Note: Median, minimum, and maximum J are calculated across five runs per solver. Compilation was disabled for all solvers. For each run, we used the last completed record at or before each time budget. The optimization-loop clock includes diagnostics and monitoring but excludes input preparation. No interpolation was used; the largest gap before the 600 s limit was 16.45 s. Intermediate wind-field accuracy was not evaluated.

In the separate memory measurements, peak allocated memory was 18.02 GiB for Adam, 19.94 GiB for L-BFGS, and 20.67 GiB for Newton-CG. Peak reserved memory was 18.54, 20.67, and 21.26 GiB, respectively. These single runs do not establish memory use with compilation or its variation across repeated runs.

## 4. Discussion and Conclusions

Adam, L-BFGS, and trust-region Newton-CG minimized the same WISSDOM objective from the same initial wind field, but they followed different convergence paths and reached different final objective values and gradient norms. In the main Typhoon Bavi case, all three solvers reproduced the main horizontal- and vertical-wind structures in the final WISSDOM output. The choice of solver therefore affected the numerical behavior and details of the retrieved field more strongly than the main flow pattern.

L-BFGS and Newton-CG produced the closest final wind fields, with component correlations above 0.9959 and three-dimensional wind-vector RMS differences below 0.5 m $s^{-1}$. Adam showed larger local differences from L-BFGS, especially in the horizontal-wind

components and in the region with reflectivity of at least 20 dBZ, but retained the same main flow structures.

Adam ended with a final objective that was 1.85% higher than the Newton-CG value, but its relative full-gradient norm was much larger. Adam's larger monitored relative full-gradient norm indicates weaker first-order stationarity at its configured stopping point. This is a numerical convergence measure, not a measure of wind-field error or meteorological quality. In the Adam run, the objective decreased overall even as the relative full-gradient norm rose after its minimum.

L-BFGS had the smallest radar-observation contribution and the lowest radial-velocity RMSE. Adam had the smallest continuity residual, whereas Newton-CG had the smallest values for several other weighted terms and physical measures. The overall agreement with the assimilated radial-velocity observations was nevertheless very similar among the three solvers. These results show that a lower total objective or a smaller gradient norm does not guarantee the best value for every individual physical measure. The problem-dependent behavior of the curvature-based methods is consistent with previous meteorological optimization studies [11–13].

The WISSDOM formulation was previously evaluated against model winds and independent observations [2,6], including measurements from a radar excluded from the retrieval [2] and withheld radiosonde profiles [6]. The present study compares solvers in an equivalent Python/GPU implementation of the Fortran formulation. It evaluates convergence, computational cost, physical residuals, and differences among retrieved wind fields; independent atmospheric-accuracy validation is outside its scope. The radial-velocity statistics measure fit to observations used during optimization. Neither the Fortran reference nor the L-BFGS solution is treated as ground truth.

The work counts in Table 3 help explain the computational trade-offs. Adam required one objective and gradient evaluation per update, whereas L-BFGS requested additional evaluations during its line search. Newton-CG also required 5162 Hessian-vector products to construct its trust-region steps. These extra derivative calculations distinguish its workload, although their separate share of runtime was not measured.

With compilation enabled for Adam and L-BFGS only, Adam completed its configured optimization in a median of 2.60 min, compared with 3.67 min for L-BFGS and 72.57 min for Newton-CG. The timing medians and convergence measures came from separate runs using the same solver settings.

The timing comparison represents the performance of the solver implementations available in the present PyTorch framework rather than the theoretical complexity of the algorithms alone. Compilation increased the time advantage of Adam and L-BFGS, but the timing order was unchanged when compilation was disabled for all solvers. The total optimizer times in Table 6 represent the cost of producing each configured WISSDOM analysis, not a comparison at a common gradient threshold or an equal amount of computational work. The common-target results in Table 7 and the common elapsed-time results in Table 8 address cost at the same objective level and progress within the same time budget, respectively. Both analyses use existing histories without changing the original stopping rules.

The main quantitative and timing comparisons focused on the Typhoon Bavi case. Appendix B presents an additional Typhoon Lingling case (Figure A3) to examine whether the numerical-convergence ordering and wind-field similarities persist under a different meteorological condition; repeated timing experiments were not performed for that case. Further comparisons across weather cases, radar configurations, initial wind fields, and optimizer settings are needed before treating the observed solver ranking as universal.

Within the tested configurations, the three solvers provided different choices. Adam may be useful when rapidly obtaining the principal wind structures is more important than reaching the smallest final relative full-gradient norm. L-BFGS provided continuity with the established WISSDOM solution and a balance among computational cost, objective reduction, and final relative full-gradient norm. Newton-CG achieved the smallest objective and monitored relative full-gradient norm at its configured stopping point, but at greater computational cost. No single solver was best for every purpose.

**Author Contributions:** S.-J.H.: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Visualization, and Writing—Original Draft Preparation. J.L.: Conceptualization, Methodology, Validation, Formal Analysis, Data Curation, Supervision, Project Administration, Funding Acquisition, and Writing—Review and Editing. D.-I.L.: Conceptualization, Methodology, Validation, Supervision, Project Administration, Funding Acquisition, and Writing—Review and Editing. Y.-C.L.: Resources, Software, and Writing—Review and Editing. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00460019).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The datasets used in this study are subject to institutional restrictions and are therefore not publicly available. The data can be made available to the editor and anonymous reviewers during peer review upon request via the journal's submission system. The full source code cannot be made publicly available because it contains third-party copyrighted and licensed material. Editors and reviewers may request specific code sections, settings, or analysis scripts through the journal submission system. We can provide the requested portions separately where sharing is permitted.

**Acknowledgments:** The authors acknowledge the use of ChatGPT (5.6) solely for English-language polishing and grammar checking during manuscript preparation.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| WISSDOM | Wind Synthesis System using Doppler Measurements |
| 3D-Var | three-dimensional variational data assimilation |
| CAPPI | constant-altitude plan-position indicator |
| KMA | Korea Meteorological Administration |
| KST | Korea Standard Time |
| GPU | graphics processing unit |
| CUDA | Compute Unified Device Architecture |
| FP32 | 32-bit floating-point precision |
| L-BFGS | limited-memory Broyden–Fletcher–Goldfarb–Shanno |
| Newton-CG | Newton conjugate-gradient |
| CG | conjugate gradient |
| HVP | Hessian-vector product |
| RMSE | root-mean-square error |
| RRMSE | relative root-mean-square error |
| SCC | spatial correlation coefficient |
| ECDF | empirical cumulative distribution function |

## Appendix A. Additional Vertical-Velocity Comparisons

Figure A1 compares how often the three solvers produced weak and strong vertical velocities. It uses the final exported vertical velocity at the first analysis time over all vertical levels. Panels (a) and (c) retain the sign of w. For example, a value of 0.90 on the vertical axis means that 90% of the grid cells have vertical velocity at or below the value on the horizontal axis. Panels (b) and (d) ignore the sign and show how often the magnitude of w reaches or exceeds each value. Their vertical axes use a logarithmic scale, so a lower curve means that large vertical velocities occur less often. The curves are similar for most values, but Newton-CG produced slightly fewer grid cells with large upward or downward velocities. This comparison does not show which solver is closer to the true atmosphere.

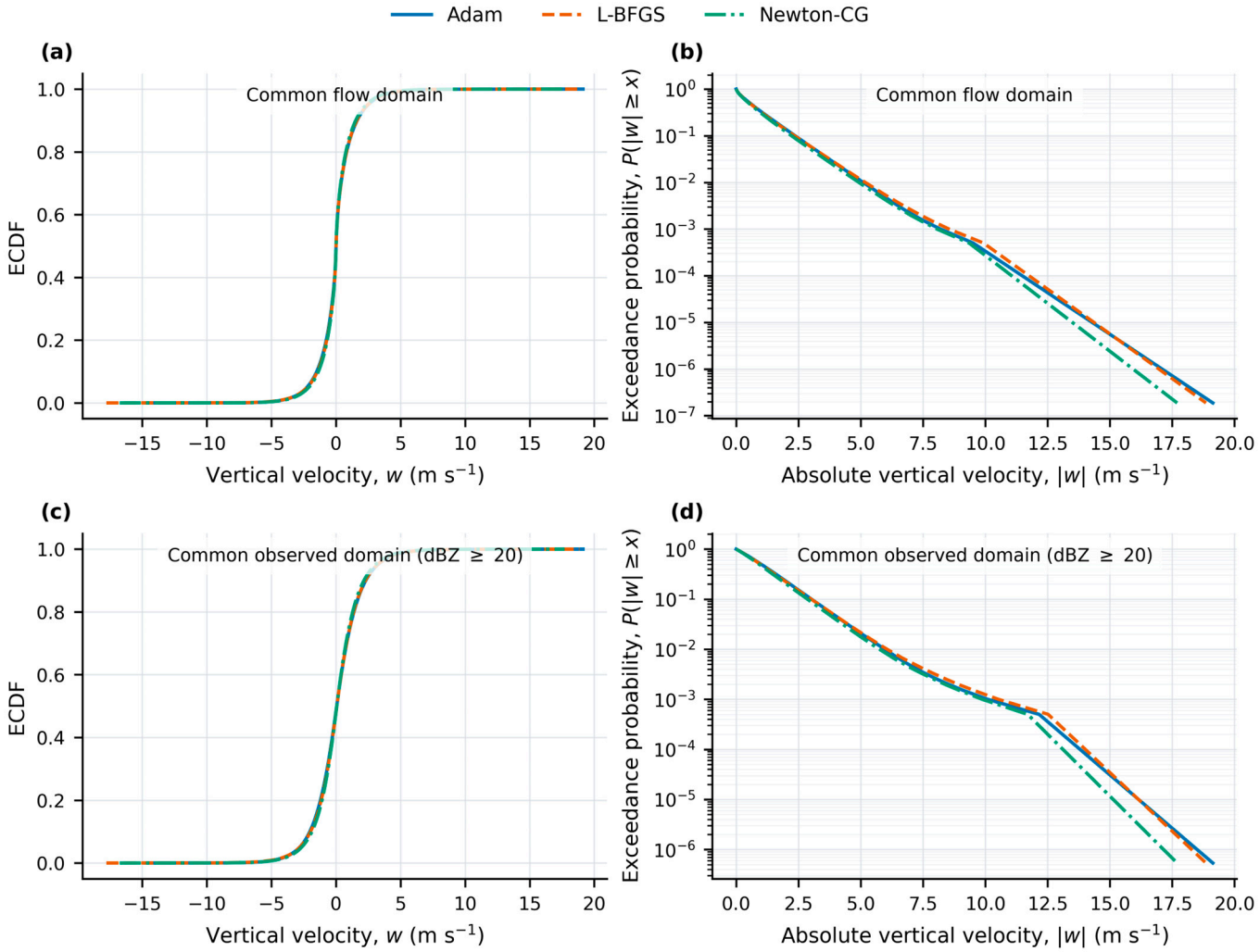



**Figure A1.** Distributions of the final exported vertical velocity at the first analysis time over all vertical levels. Panels (**a**,**b**) use grid cells with valid wind values from all three solvers. Panels (**c**,**d**) use the common region with reflectivity of at least 20 dBZ. Panels (**a**,**c**) show the fraction of grid cells with vertical velocity less than or equal to each value. Panels (**b**,**d**) show the fraction with an absolute vertical velocity greater than or equal to each value; their vertical axes use a logarithmic scale.

Figure A2 shows where the small vertical-velocity differences occur. Both panels use one fixed difference range. The limit was set to the larger of the 99th percentiles of the absolute differences in the two displayed maps, 1.68 m s$^{-1}$. This allows at least 99% of the values in each map to be displayed without saturation while preventing a few extreme grid cells from setting the entire color range. Values beyond the limit use the end colors. The

maps are diagnostic enlargements and should be interpreted together with the RMS values shown in the panels and Table 4.

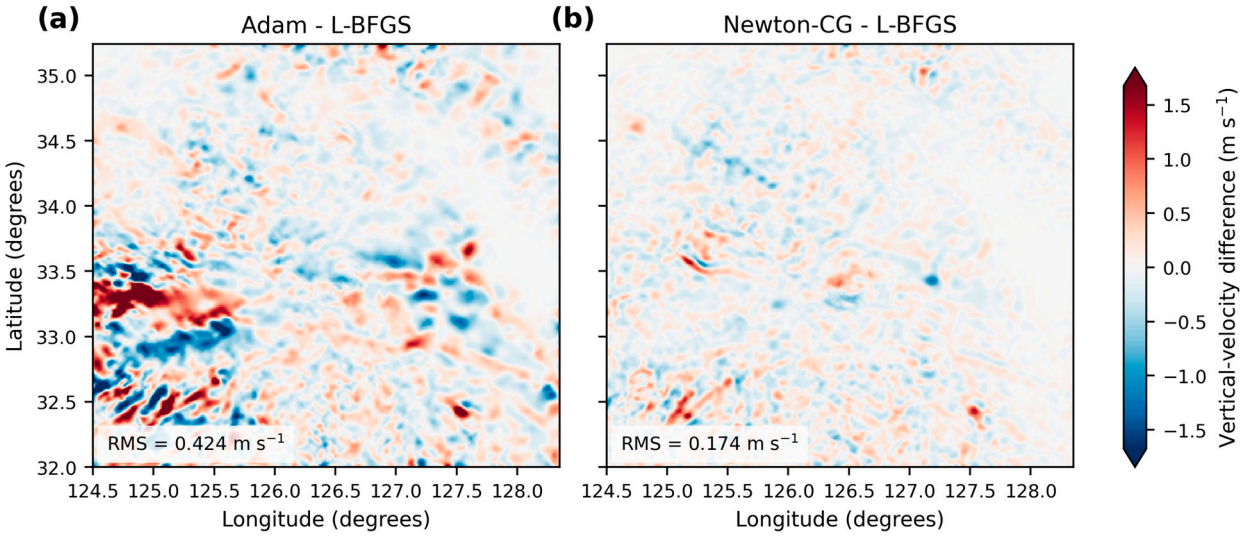



**Figure A2.** Differences in the final exported vertical velocity at the first analysis time and 5 km height: (**a**) Adam minus L-BFGS and (**b**) Newton-CG minus L-BFGS. Both panels use the same ±1.68 m s$^{-1}$ range, set to the larger of the 99th percentiles of the absolute differences in the two maps. Values outside this range are shown using the end colors. L-BFGS is a comparison baseline, not truth.

## Appendix B. Additional Case: Typhoon Lingling

To examine whether the solver-dependent patterns found for Typhoon Bavi also appeared in a different typhoon retrieval, the three solvers were applied to a Typhoon Lingling case at 0000 and 0010 UTC 7 September 2019 using observations from the GSN and SSP radars. Within the Lingling comparison, all three solvers used the same processed observations, background fields, objective coefficients, and initial wind fields. They therefore shared an initial objective of $J_0$ = 33,922.961. Adam, L-BFGS, and Newton-CG were run for 900, 625, and 350 updates, respectively, using the same optimizer settings and solver-specific iteration limits as in the Typhoon Bavi experiment, without case-specific retuning for any of the three solvers. This additional case was used to examine numerical and wind-field consistency; repeated timing experiments were not performed.

All three solvers reduced the objective by more than 96.3% (Table A1). Newton-CG produced the smallest final objective, 1187.740, and L-BFGS finished only 0.144% above this value. Adam's final objective was 4.36% higher than Newton-CG's. The final relative full-gradient norms were 0.34310 for Adam, 0.01647 for L-BFGS, and 0.00472 for Newton-CG. This ordering was consistent with the Typhoon Bavi case. The larger Adam gradient norm is a numerical convergence measure and should not be interpreted directly as wind-field error.

The three solvers reproduced the same broad horizontal-wind structure (Figure A3). L-BFGS and Newton-CG again produced the closest final wind fields. Averaged over the two analysis times, their common-domain RMS differences were 0.361, 0.370, and 0.166 m s$^{-1}$ for u, v, and w, respectively. Their corresponding spatial correlations were 0.99943, 0.99932, and 0.99396. Adam retained the same broad flow pattern but showed larger local differences. The average Adam–L-BFGS RMS differences were 1.199, 1.933, and 0.401 m s$^{-1}$ for u, v, and w, respectively, with the largest differences occurring locally within the observed precipitation region.

**Table A1.** Final numerical measures and radial-velocity fit for the Typhoon Lingling case.

| Mean SCC | Vr RMSE (m s$^{-1}$) | Relative Full-Gradient Norm | Reduction (%) | Final J | Updates | Solver |
|---|---|---|---|---|---|---|
| 0.998016 | 1.4787 | 0.34310 | 96.346 | 1239.501 | 900 | Adam |
| 0.998085 | 1.4510 | 0.01647 | 96.494 | 1189.448 | 625 | L-BFGS |
| 0.998066 | 1.4586 | 0.00472 | 96.499 | 1187.740 | 350 | Newton-CG |

Note: Objective reduction was calculated from the common initial objective, $J_0$ = 33,922.961. The relative full-gradient norm is defined in Section 2.5. Vr RMSE and SCC describe the fit to the radial-velocity observations used during optimization.

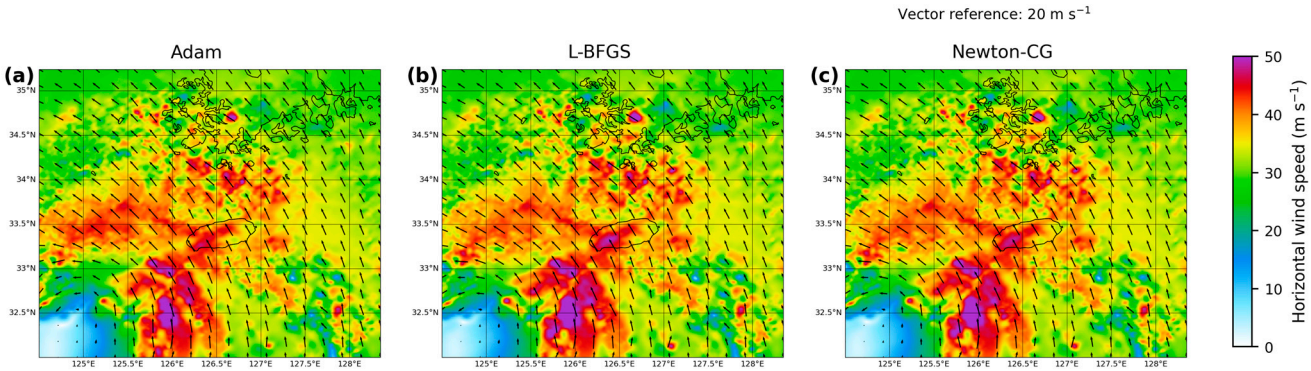



**Figure A3.** Final horizontal wind fields at 3 km height for the Typhoon Lingling case at 0010 UTC 7 September 2019 obtained with (**a**) Adam, (**b**) L-BFGS, and (**c**) Newton-CG. Colors show horizontal wind speed, and arrows show horizontal wind direction and magnitude. The same color range and vector reference are used in all panels.

The fits to the assimilated radial velocities were similar among the solvers. Their mean SCC values were approximately 0.998, and their Vr RMSE values ranged from 1.451 to 1.479 m s$^{-1}$. L-BFGS produced the smallest Vr RMSE, whereas Newton-CG produced the smallest final objective and relative full-gradient norm. The Lingling case therefore supports the main Typhoon Bavi result: the principal retrieved wind structure remained broadly consistent across the three solvers, while numerical convergence and local wind details remained solver-dependent. The exact differences, however, varied between the two cases.

## References

1. Gao, J.; Xue, M.; Shapiro, A.; Droegemeier, K.K. A Variational Method for the Analysis of Three-Dimensional Wind Fields from Two Doppler Radars. *Mon. Weather Rev.* **1999**, *127*, 2128–2142. [CrossRef]
2. Liou, Y.-C.; Chang, Y.-J. A Variational Multiple–Doppler Radar Three-Dimensional Wind Synthesis Method and Its Impacts on Thermodynamic Retrieval. *Mon. Weather Rev.* **2009**, *137*, 3992–4010. [CrossRef]
3. Brook, J.P.; Protat, A.; Potvin, C.K.; Soderholm, J.S.; McGowan, H. The Effects of Spatial Interpolation on a Novel, Dual-Doppler 3D Wind Retrieval Technique. *J. Atmos. Ocean. Technol.* **2023**, *40*, 1325–1347. [CrossRef]
4. Cha, T.-Y.; Bell, M.M. Three-Dimensional Variational Multi-Doppler Wind Retrieval over Complex Terrain. *J. Atmos. Ocean. Technol.* **2023**, *40*, 1381–1405. [CrossRef]
5. Potvin, C.K.; Wicker, L.J.; Shapiro, A. Assessing Errors in Variational Dual-Doppler Wind Syntheses of Supercell Thunderstorms Observed by Storm-Scale Mobile Radars. *J. Atmos. Ocean. Technol.* **2012**, *29*, 1009–1025. [CrossRef]
6. Liou, Y.-C.; Chang, S.-F.; Sun, J. An Application of the Immersed Boundary Method for Recovering the Three-Dimensional Wind Fields over Complex Terrain Using Multiple-Doppler Radar Data. *Mon. Weather Rev.* **2012**, *140*, 1603–1619. [CrossRef]
7. Liou, Y.-C.; Chou, T.-J.; Cheng, Y.-T.; Teng, Y.-L. The Improvement of Short-Term Quantitative Precipitation Forecast in Mountainous Areas by the Assimilation of Meteorological State Variables Retrieved by Multiple Doppler Radar Data. *Mon. Weather Rev.* **2024**, *152*, 1339–1355. [CrossRef]

8. Fisher, M. Minimization Algorithms for Variational Data Assimilation. In *Proceedings of the ECMWF Seminar on Recent Development in Numerical Methods for Atmospheric Modelling, Reading, UK, 7–11 September 1998*; ECMWF: Reading, UK, 1998.
9. Liu, D.C.; Nocedal, J. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Program.* **1989**, *45*, 503–528. [CrossRef]
10. Nocedal, J.; Wright, S.J. *Numerical Optimization*, 2nd ed.; Springer Series in Operations Research and Financial Engineering; Springer: New York, NY, USA, 2006.
11. Zou, X.; Navon, I.M.; Berger, M.; Phua, K.H.; Schlick, T.; Le Dimet, F.X. Numerical Experience with Limited-Memory Quasi-Newton and Truncated Newton Methods. *SIAM J. Optim.* **1993**, *3*, 582–608. [CrossRef]
12. Wang, Z.; Navon, I.M.; Zou, X.; Le Dimet, F.X. A Truncated Newton Optimization Algorithm in Meteorology Applications with Analytic Hessian/Vector Products. *Comput. Optim. Appl.* **1995**, *4*, 241–262. [CrossRef]
13. Le Dimet, F.-X.; Navon, I.M.; Daescu, D.N. Second-Order Information in Data Assimilation. *Mon. Wea. Rev.* **2002**, *130*, 629–648. [CrossRef]
14. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.
15. Melinc, B.; Zaplotnik, Ž. 3D-Var Data Assimilation Using a Variational Autoencoder. *Quart. J. R. Meteoro Soc.* **2024**, *150*, 2273–2295. [CrossRef]
16. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic Differentiation in Machine Learning: A Survey. *J. Mach. Learn. Res.* **2018**, *18*, 1–43.
17. Pearlmutter, B.A. Fast Exact Multiplication by the Hessian. *Neural Comput.* **1994**, *6*, 147–160. [CrossRef]
18. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8024–8035.
19. Steihaug, T. The Conjugate Gradient Method and Trust Regions in Large Scale Optimization. *SIAM J. Numer. Anal.* **1983**, *20*, 626–637. [CrossRef]
20. Harris, C.R.; Millman, K.J.; Van Der Walt, S.J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N.J.; et al. Array Programming with NumPy. *Nature* **2020**, *585*, 357–362. [CrossRef]
21. Lam, S.K.; Pitrou, A.; Seibert, S. Numba: A LLVM-Based Python JIT Compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, Austin, TX, USA, 15 November 2015*; ACM: New York, NY, USA, 2015; pp. 1–6.
22. Moré, J.J.; Thuente, D.J. Line Search Algorithms with Guaranteed Sufficient Decrease. *ACM Trans. Math. Softw.* **1994**, *20*, 286–307. [CrossRef]
23. Cressman, G.P. An Operational Objective Analysis System. *Mon. Weather Rev.* **1959**, *87*, 367–374. [CrossRef]
24. Tsuboki, K.; Sakakibara, A. Large-Scale Parallel Computing of Cloud Resolving Storm Simulator. In *High Performance Computing*; Zima, H.P., Joe, K., Sato, M., Seo, Y., Shimasaki, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 243–259. [CrossRef]
25. Ansel, J.; Yang, E.; He, H.; Gimelshein, N.; Jain, A.; Voznesensky, M.; Bao, B.; Bell, P.; Berard, D.; Burovski, E.; et al. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, La Jolla, CA, USA, 27 April–1 May 2024*; ACM: New York, NY, USA, 2024; Volume 2, pp. 929–947.


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.