https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

ADDA: a Modular Framework for Representing, Simulating and Assimilating Dynamics with End-to-end Differentiability

Anthony Frion, Vien Minh Nguyen-Thanh, Ali Can Bekar, Pauleo R. Nimtz, Vadim Zinchenko, and David S. Greenberg

Helmholtz-Zentrum Hereon, Geesthacht, Germany

Correspondence: Anthony Frion (anthony.frion@hereon.de)

Abstract. Data assimilation (DA) is an essential tool for prediction and understanding in the geosciences. DA combines simulation programs representing scientific knowledge with observations that constrain system dynamics, resulting in analyses and forecasts that incorporate both knowledge and data. DA tasks can be addressed with a diverse toolset, including variational, ensemble and learning-based methods. In particular, many recent works have proposed using automatic differentiation tools for variational, learning-based or hybrid methods. However, comprehensive comparisons across algorithms and dynamical systems remain challenging, due to the incompatibility of simulation and assimilation codes, inflexible handling of spatial and temporal discretizations, specialization of DA methods to specific simulations, and limited support for automatic differentiation and parallel computation in simulations. To address this challenge, we introduce Automatic Differentiation for Data Assimilation (ADDA), a software framework for defining and working with system states, simulations, observation schemes and DA methods. ADDA provides a powerful and flexible set of base classes for representing dynamical systems and observation operators, with support for collocated and staggered grids, unstructured meshes, Lagrangian state variables and irregular or continuous-time observations. ADDA implements the Kalman filter and smoother, ensemble Kalman filter and smoother, strong-constraint and weak-constraint 4D-Var with a single or a sliding window. It can be easily extended or modified to support new methods. Parallel processing and differentiability are first-class features, with support for batch axes and automatic differentiation throughout. ADDA is implemented in PyTorch library, but supports DA for JAX-based computation of dynamics and their gradients. To demonstrate its features and facilitate development and evaluation of DA methods, we further provide differentiable, ADDA-compatible implementations of 10 dynamical systems of various dimensionalities and scales, from which we design multiple illustrative DA examples. All of our code is publicly available at https://github.com/m-dml/ADDA.

1 Introduction

Data assimilation (DA) (Carrassi et al., 2018), a critically important endeavor in the geosciences, combines simulation and observation to obtain system state sequences consistent with both. It can be used to estimate system states more accurately than simulation or observation alone (Carrassi et al., 2018; Hersbach et al., 2020), to tune simulation parameters (Aksoy et al., 2006; Bocquet and Sakov, 2013) and to identify key situations or phenomena for which simulation and observations conflict (Carrió

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

et al., 2025). DA is the focus of much theoretical and empirical research, and its operational practice demands significant financial, computational and energy resources (Balsamo et al., 2023; Drillet et al., 2024).

DA unites a diverse suite of methods with an equally varied array of applications. Major classes of well-established "classical" methods include variational methods such as strong- (Rabier et al., 2000) and weak-constraint 4D-Var (Zupanski, 1997), gradient-free ensemble methods (Evensen, 1994; Evensen and Leeuwen, 2000) and hybrid techniques combining gradients with ensembles (Liu et al., 2008; Sakov et al., 2012). Kalman filtering provides precise Bayesian inference for linear dynamics with Gaussian uncertainty (Kalman, 1960), while particle filters can assimilate nonlinear dynamics or observations but suffer from a curse of dimensionality that limits their application in the geosciences (Wills and Schön, 2023). Within these classes of DA methods, individual methods exhibit subtle but consequential differences, and implementations of a single method can vary widely in usability, flexibility, numerical stability and computational efficiency.

Advances in data-driven DA present an important, new and highly diverse class of methods (Cheng et al., 2023). Fully supervised approaches avoid simulation during training, but generally require simulation or another DA method to produce training data (Hersbach et al., 2020), in the form of full system states. Directly training on sparse and/or noisy observations usually assumes access to a differentiable simulation (Zinchenko and Greenberg, 2024; Wang et al., 2024) or emulator (Xiao et al., 2024a; Li et al., 2024). 4DVarNets (Fablet et al., 2021) learn a 4D-Var-like loss and pass its gradients to a recurrent DA network, supporting both supervised (Beauchamp et al., 2023; Febvre et al., 2024; Fablet et al., 2024) and unsupervised training (Dorffer et al., 2025). Other approaches train a fast neural emulator on simulations that are too slow or lack gradient routines, enabling classical variational DA (Nonnenmacher and Greenberg, 2021; Hatfield et al., 2021; Maulik et al., 2022; Seabra et al., 2024; Xiao et al., 2024a; Li et al., 2024; Durand et al., 2025). In some cases, emulators trained on coarse-resolution simulations exhibit better consistency with high-resolution simulations than the coarse simulations they were trained on (Koehler and Thuerey, 2026). However, emulation is unnecessary when simulations are implemented in deep learning frameworks with automatic differentiation (autodiff), since gradient backpropagation is closely related to variational DA (Cheng et al., 2023). Latent data assimilation (LDA) reduces computation and memory costs by operating in latent spaces to reduce the dimension (Amendola et al., 2020; Peyron et al., 2021; Cheng et al., 2024; Melinc and Zaplotnik, 2024; Cheng et al., 2025), to leverage simplified (e.g. linear) dynamics (Frion et al., 2024; Singh et al., 2024; Frion et al., 2025; Shoji et al., 2025; Tong et al., 2026) or to work with complex non-Gaussian prior distributions (Qu et al., 2024; Xiao et al., 2024b, 2025; Pasmans et al., 2026; Fan et al., 2026). Diffusion models have been used for probabilistic DA, both by directly mapping observations onto system state posteriors (Huang et al., 2024) and by training a prior on system state sequences (Rozet and Louppe, 2023) before conditioning with Tweedie's formula (Efron, 2011). Diffusion models and related approaches can incorporate latent representations (Qu et al., 2024), exploit novel training strategies (Shysheya et al., 2024; Chen et al., 2025; Sun et al., 2025), learn from observations (Rozet et al., 2024) and scale to global atmospheric states (Andry et al., 2025). Overall, data-driven DA methods are nearly always faster than classical DA, and can be used to initialize iterative classical methods (Frerix et al., 2021; Frion and Greenberg, 2026).

The complex landscape of DA applications is at least as varied as that of its methods. These include the familiar cases of atmospheric (Navon, 2009; Gustafsson et al., 2018; Hersbach et al., 2020; Xu et al., 2025) and ocean dynamics (Stammer et al.,

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

2016; Drillet et al., 2024; Grande et al., 2026), as well as hydrology (Sun et al., 2016), ecosystems (Niu et al., 2014), biogeochemistry (Dowd et al., 2014; Rayner et al., 2019), agriculture (Jin et al., 2018; Pandya et al., 2022), pose estimation (Aharon et al., 2025), economics (Nadler et al., 2019), water quality modeling (Cho et al., 2020), forest inventory (Xu et al., 2023), land surface modeling (Raoult et al., 2025) and agent-based modeling (Ghorbani et al., 2023). The simulation models underlying these diverse tasks rely on diverse programming languages, compilers, computing hardware, discretization schemes and data formats. Software packages for some simulations include their own assimilation routines (Brajard et al., 2021; Xiao et al., 2024b; Xu et al., 2025), but these are often highly specialized and cannot be applied to other cases.

This complexity and diversity limit compatibility. Researchers developing a new DA method must contend with the difficulty of integrating it with diverse simulation codebases, system representations and observation types. Likewise, comparing assimilation methods on a new (or newly modified) simulation requires adapting the simulation to provide inputs with the format and structure required by each algorithm, and translating observation operators to an accepted format. When simulations support parallel or gradient computation, extending these capabilities to DA methods is nontrivial. The laborious nature of matching DA methods to simulations in each case costs precious research time, limits the feasibility of careful evaluations and increases the risk of implementation errors. Thus, at present it is a significant challenge to evaluate a new DA method, compare DA methods across assimilation tasks, or select the optimal method for each task.

Here we introduce ADDA, a modular framework for representing and simulating dynamical systems, for implementing observation operators, and for building, applying and evaluating data assimilation methods. We built ADDA to satisfy the following set of core requirements:

- End-to-end autodiff support for variational DA methods.
- Support for diverse system state structures, including regular and unstructured grids and meshes, variable staggering, and mixed dimensionality (e.g. surface vs. atmospheric fields for Earth models).
- Observation operators describing the dependence of observations on system states, supporting spatiotemporally structured and unstructured observations, masking and noise.
- Support for flexible classes of dynamics, including time-varying boundary conditions and forcing data and numerical simulators as well as neural emulators.
- Modularity allowing simulation models, observation operators and DA methods to be individually defined, modified and extended before being combined to carry out DA tasks.
- Parallelized batched computation on CPUs and GPUs.
- Concrete, easily-modified examples demonstrating major functionalities.

We implemented ADDA as a set of algorithms, functions and classes in Python and PyTorch. ADDA satisfies our stated core requirements and provides a clearly defined interface through which simulations and DA algorithms can interoperate.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

ADDA is inspired by existing efforts to unify DA models, observation operators and simulations in a common framework. DART (Anderson et al., 2009) includes many types of dynamics and observation types, but focuses on ensemble-based DA and lacks variational methods. Its Fortran implementation aids efficiency but increases the complexity of its workflows and of interfacing with new simulations. PDAF (Nerger, 2023) applies ensemble Kalman methods to large scale dynamical systems with emphasis on parallelization, but does not support 4D-Var. NEDAS (Ying, 2025) is an ensemble DA system in Python with CPU parallelization, designed to efficiently scale to large geophysical applications. However, it lacks automatic differentiation, and its required workflows are complex and incompletely documented. PyDA (Ahmed et al., 2020) provides a simple data assimilation tutorial in Python, but also lacks support for automatic differentiation, so its variational DA requires explicit derivation and implementation of cost function gradients, which can become cumbersome for more complex systems. DAPPER (Raanes et al., 2024) is a Python package implementing a variety of variational, ensemble and other methods, with many examples reproducing published studies. It presents a major achievement in unifying DA methods and applications, but with significant limitations: it is not designed for large systems (>60000 DOF), uses inefficient user-specified forward adjoints instead of autodiff backpropagation, provides only limited CPU parallelization and imposes restrictions on the spatial discretization, time stepping and observation scheme. TorchDA (Cheng et al., 2025) supports autodiff in data assimilation, but only for a limited set of methods and examples focused on DA with neural emulators of dynamical systems. Each of these frameworks serves an important community need, but none fulfill the above aims. Generally speaking, one can note that variational methods have historically been considered to be complex to implement due to the necessity of building an adjoint model for the dynamical system of interest, leading to a preference for ensemble Kalman methods in general-purpose data assimilation software. The recent development of automatic differentiation tools has now considerably simplified the usage of 4D-Var, but this change is not yet reflected in major DA packages. Our work aims to fill this gap. As multiple modern frameworks exist for automatic differentiation, ADDA is implemented in PyTorch, the most popular for deep learning. To support a wide and flexible usage, we provide support for solving DA on systems implemented in JAX, which is another major automatic differentiation framework. An associated experiment is presented in Sect. 4.8.

We emphasize that ADDA is not a benchmark or benchmarking tool, and this manuscript does not describe a benchmarking study. Arriving at a fair and comprehensive benchmark will require a concerted community effort to ensure that all relevant DA methods are adequately implemented and properly tuned, that a fair and representative range of DA tasks has been selected, and that reasonable metrics have been used to measure accuracy and efficiency. Rather, we designed and developed ADDA to provide a necessary foundation for implementing DA methods and interfacing them with simulations, keeping in mind the goal of enabling future benchmarking efforts. Our set of examples in Sect. 4 has been designed to showcase the versatility of ADDA.

We describe and demonstrate ADDA as follows. We first frame the DA task and establish notation in Sect. 2. In Sect. 3, we describe essential DA concepts and components and their realizations in ADDA, and the implementation of DA algorithms. Section 4 then illustrates ADDA's capabilities on DA examples ranging across dynamical systems, observation operators and algorithms. Section 5 concludes with a summary of ADDA's capabilities and the outlook for future developments and applications.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

2 Data Assimilation: Methods and Formulations

In this section we formalize the DA task. We begin by framing the "standard" setup of a nonlinear dynamical system with per-time-point observations as well as Gaussian process and observation noise (Sect. 2.1). We then derive and describe the Bayesian posterior on states given observations (Sect. 2.2), and how various DA approaches relate to it (Sect. 2.3 and 2.4). Beyond the standard setup, we consider a more general class of scenarios supported by ADDA in Sect. 2.5.

2.1 DA task formulation: Gaussian process and observation noise

We consider a state-space model evolving over discrete time steps t with Gaussian process and observation noise terms. Each observation $\mathbf{y}_t$ depends on the simultaneous state $\mathbf{x}_t$, according to the following equations:

$$\mathbf{x}_{t+1} = \mathcal{M}(\mathbf{x}_t) + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}) \tag{1}$$

$$\mathbf{y}_t = \mathcal{H}_t(\mathbf{x}_t) + \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\eta}_t}) \tag{2}$$

$$\mathbf{x}_0 \sim \mathcal{N}(\mathbf{x}^B, \boldsymbol{\Sigma}_B) \tag{3}$$

Equations (1) and (2) are respectively referred to as the state and observation equations. The state $\mathbf{x}_t \in \mathbb{R}^s$ evolves through the discrete dynamical model $\mathcal{M} : \mathbb{R}^s \to \mathbb{R}^s$, with zero-mean Gaussian process noise $\boldsymbol{\epsilon}_t$. Observations $\mathbf{y}_t \in \mathbb{R}^{o_t}$ depend on $\mathbf{x}_t$ through the observation operator $\mathcal{H}_t : \mathbb{R}^s \to \mathbb{R}^{o_t}$. Observation noise $\boldsymbol{\eta}_t \in \mathbb{R}^{o_t}$ follows a Gaussian distribution with mean 0 and covariance $\boldsymbol{\Sigma}_{\boldsymbol{\eta}_t}$. A simple but important case is the linear observation $\mathcal{H}_t(\mathbf{x}_t) = \mathbf{H}_t\mathbf{x}_t$, where $\mathbf{H}_t$ is a $o_t \times s$ matrix. Clearly, $\mathcal{H}_t$ is not invertible when $o_t < s$.

In DA, our task is to estimate the system state trajectory $\mathbf{x}_{0:T}$ from an observation sequence $\mathbf{y}_{0:T}$. In addition to the observation sequence, we assume full knowledge of the dynamics $\mathcal{M}$, process noise $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$, dependence of expected observations on states $\mathcal{H}_t$, observation noise $\boldsymbol{\Sigma}_{\boldsymbol{\eta}_t}$ and initial state prior $p(\mathbf{x}_0)$.

2.2 Bayesian derivation of the posterior distribution of the state

Given a sequence of observations, the full probabilistic solution to the DA problem describes the full range of possible system state sequences, and how likely they are given the available data. In practice, many effective DA methods approximate the full posterior distribution with a point estimate, a marginal distribution for each time point or an ensemble of finite size, or by taking into account only a subset of observations. We describe these in Sect. 2.3 and 2.4. The posterior distribution over states given observations derives from the prior, state and observation equations according to Bayes' theorem:

$$p(\mathbf{x}_{0:T}|\mathbf{y}_{0:T}) = \frac{p(\mathbf{x}_{0:T})p(\mathbf{y}_{0:T}|\mathbf{x}_{0:T})}{p(\mathbf{y}_{0:T})}. \tag{4}$$

Equation (1) is an example of a state space model, which has the Markovian property that each state's distribution, when conditioned on all earlier states, depends only on the immediately preceding state, so that the prior on state sequences factorizes:

$$p(\mathbf{x}_{0:T}) = p(\mathbf{x}_0)\prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{5}$$

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Similarly, assuming that observations are independent in time and that each of them depends only on a single, simultaneously occurring state yields a factorized likelihood:

$$p(\mathbf{y}_{0:T}|\mathbf{x}_{0:T}) = \prod_{t=0}^{T} p(\mathbf{y}_t|\mathbf{x}_t). \tag{6}$$

Combining these assumptions, and dropping the marginal likelihood $p(\mathbf{y}_{0:T})$ which is constant with respect to the unknown states $\mathbf{x}_{0:T}$, we arrive at the factorized posterior:

$$p(\mathbf{x}_{0:T}|\mathbf{y}_{0:T}) \propto p(\mathbf{x}_0) \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}) \prod_{t=0}^{T} p(\mathbf{y}_t|\mathbf{x}_t). \tag{7}$$

Plugging in the Gaussian densities from Eq. (1)-(3), we arrive at:

$$\log p(\mathbf{x}_{0:T}|\mathbf{y}_{0:T}) = -\frac{1}{2}\left(\left\|\mathbf{x}_0 - \mathbf{x}^B\right\|^2_{\mathbf{\Sigma}_B^{-1}} + \sum_{t=1}^{T} \|\mathbf{x}_t - \mathcal{M}(\mathbf{x}_{t-1})\|^2_{\mathbf{\Sigma}_\epsilon^{-1}} + \sum_{t=0}^{T} \|\mathbf{y}_t - \mathcal{H}_t(\mathbf{x}_t)\|^2_{\mathbf{\Sigma}_{\eta_t}^{-1}} \right) + C \tag{8}$$

where $||\mathbf{x}||^2_{\mathbf{\Sigma}^{-1}} = ||\mathbf{x}^\intercal \mathbf{\Sigma}^{-1} \mathbf{x}||^2$ is a weighted inner product and C does not depend on $\mathbf{x}_{0:T}$. Note that if the prior on the initial state $\mathbf{x}_0$ is taken to be uniform instead of Gaussian, the first term of Eq. (8) vanishes.

For the case of nonlinear dynamics $\mathcal{M}$, the posterior distribution is intractable and non-Gaussian, and calculating or sampling from this density is challenging. This motivates specialized DA algorithms that employ various approximations while exploiting the temporal structure of the assimilation task.

2.3 The 4D-Var cost and its minimization

4D-Var, the most common variational data assimilation method, is a *maximum a posteriori* (MAP) technique that seeks the mode of the posterior distribution from Eq. (8), by minimizing the generic variational cost J:

$$\underset{\mathbf{x}_{0:T} \in \mathbb{R}^{s\times(T+1)}}{\arg\min}\ J(\mathbf{x}_{0:T}) = \underset{\mathbf{x}_{0:T}}{\arg\min}\ \frac{1}{2}\left(\left\|\mathbf{x}_0 - \mathbf{x}^B\right\|^2_{\mathbf{\Sigma}_B^{-1}} + \sum_{t=1}^{T} \|\mathbf{x}_t - \mathcal{M}(\mathbf{x}_{t-1})\|^2_{\mathbf{\Sigma}_\epsilon^{-1}} + \sum_{t=0}^{T} \|\mathbf{y}_t - \mathcal{H}_t(\mathbf{x}_t)\|^2_{\mathbf{\Sigma}_{\eta_t}^{-1}} \right). \tag{9}$$

The minimization problem of Eq. (9) is classically referred to as *weak-constraint* 4D-Var: see e.g. Trémolet (2007); Bannister (2017); Carrassi et al. (2018); Evensen et al. (2022) for similar expressions. It should however be noted that multiple alternative expressions exist: see e.g. Trémolet (2007); Laloyaux et al. (2020) for some examples. The first term in Eq. (9) measures deviation of the initial state from prior expectations, the second measures deviation of the state's evolution from the expected dynamics (hence the name "weak-constraint"), and the third measures deviation from observations. While computing the second term in Eq. (9) requires T time steps of simulated dynamics $\mathcal{M}$ to evaluate J, it does not require autoregressive iteration of $\mathcal{M}$, allowing straightforward parallel computation over the time axis.

The *strong-constraint* formulation of 4D-Var (sometimes referred to simply as 4D-Var) instead requires that $\mathbf{x}_t = \mathcal{M}(\mathbf{x}_{t-1})$ precisely (Carrassi et al., 2018). Thus, the model error $\boldsymbol{\epsilon}_t = 0$ in Eq. (1), and the state evolves deterministically from its initialization $\mathbf{x}_0$. In this case, the variational cost in Eq. (9) is redefined for optimization only over $\mathbf{x}_0$:

$$\underset{\mathbf{x}_0 \in \mathbb{R}^s}{\arg\min}\ J(\mathbf{x}_0) = \underset{\mathbf{x}_0 \in \mathbb{R}^s}{\arg\min}\ \frac{1}{2}\left(\left\|\mathbf{x}_0 - \mathbf{x}^B\right\|^2_{\mathbf{\Sigma}_B^{-1}} + \sum_{t=0}^{T} \left\|\mathbf{y}_t - \mathcal{H}_t(\mathcal{M}^{(t)}(\mathbf{x}_0))\right\|^2_{\mathbf{\Sigma}_{\eta_t}^{-1}} \right), \tag{10}$$

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

where $\mathcal{M}^{(t)}$ means t compositions of $\mathcal{M}$. The effectiveness of weak- vs. strong-constraint 4D-Var can depend on how accurately we know $\mathcal{M}$ and $\mathbf{\Sigma}_{\boldsymbol{\eta}_t}$, the sensitivity of the dynamics to initial perturbations and the nature of available observations.

To minimize J, weak- and strong-constraint 4D-Var methods rely on some form of iterative gradient-based optimization. This generally requires a so-called "adjoint" model of the dynamics, $\mathbf{M} = \frac{d\mathcal{M}}{d\mathbf{x}}$. As detailed in e.g. Bannister (2001), the gradient of Eq. (10) can then be written as

$$\nabla_{\mathbf{x}_0} J = \mathbf{\Sigma}_B^{-1}(\mathbf{x}_0 - \mathbf{x}^B) - \sum_{t=0}^{T} (\mathbf{M}^\intercal)^t \mathbf{H}_t^\intercal \mathbf{\Sigma}_{\boldsymbol{\eta}}{}^{-1}(\mathbf{y}_t - \mathcal{H}_t(\mathbf{x}_t)), \tag{11}$$

where $\mathbf{H}_t$ is a linearization of the observation operator $\mathcal{H}_t$ at time t. Deriving, implementing and maintaining the adjoint $\mathbf{M}$ of a dynamical system $\mathcal{M}$ using the chain rule is widely considered to be a cumbersome operation, making it the main disadvantage of 4D-Var with regards to ensemble Kalman methods. However, one major contribution of our proposed ADDA package is the substitution of the manual derivation of the adjoint $\mathbf{M}$ by an automatic computation of this adjoint using automatic differentiation. Thus, given that the system $\mathcal{M}$ of interest has a differentiable implementation in PyTorch or JAX, the usage of various 4D-Var variants becomes straightforward. Although re-implementing classical models to make them auto-differentiable can also be a cumbersome process, it has repeatedly proven to be a successful approach (Gelbrecht et al., 2023; Zhou et al., 2024; Solvik et al., 2025; Meunier et al., 2025). Alternatively, a non-differentiable implementation of the dynamics $\mathcal{M}$ can be replaced by a neural emulator (Nonnenmacher and Greenberg, 2021; Hatfield et al., 2021; Penny et al., 2022; Maulik et al., 2022; Durand et al., 2025), which is then differentiable by design.

2.4 Sequential methods

Besides variational methods, the other major class of data assimilation techniques is sequential methods, which assimilate each observation individually while moving forward along the time axis. The most prominent sequential methods include Kalman filters (Kalman, 1960) and smoothers (Rauch et al., 1965) as well as ensemble Kalman filters (Evensen, 1994) and smoothers (Evensen and Leeuwen, 2000). An important advantage of these methods compared to variational approaches is that they do not require an adjoint model of the state dynamics $\mathcal{M}$. Unlike variational methods that optimize to find the posterior mode, at each time step a sequential method directly computes or approximates the *filtering distribution* $p(\mathbf{x}_t|\mathbf{y}_{0:t})$, the posterior over states given past and present (but not future) observations. Filtering methods alternate between forecasting steps that simulate a single time step through $\mathcal{M}$ to advance $p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1}) \rightarrow p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$, and analysis steps that condition on a new observation $\mathbf{y}_t$ to obtain $p(\mathbf{x}_t|\mathbf{y}_{0:t})$. After filtering, a subsequent smoothing step can then incorporate future observations to compute the *smoothing distribution* $p(\mathbf{x}_t|\mathbf{y}_{0:T})$. A *fixed-lag smoother* incorporates observations from some fixed number d of future time steps to obtain $p(\mathbf{x}_t|\mathbf{y}_{0:t+d})$.

Filtering and smoothing distributions can be computed precisely only in a narrow range of circumstances. Kalman filters and smoothers do this for the linear-Gaussian case, where $\mathcal{M}$ in Eq. (1) and $\mathcal{H}_t$ in Eq. (2) are linear operators. In this case the filtering, smoothing and fixed-lag smoothing distributions are Gaussian as well, and their means and covariances can be computed in closed form through iterative updates along the time axis.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

In contrast, ensemble Kalman filters and smoothers represent probability distributions through discrete ensembles sampled from these distributions. This allows for accurate assimilation with nonlinear dynamics and observation operators in many practically important scenarios. In the forecast step, each ensemble member is advanced independently through the nonlinear dynamics $\mathcal{M}$. In the analysis step, the empirical covariance of state variables across ensemble members is used to condition on observations similarly to Kalman filtering of linear-Gaussian systems. The resulting posterior distribution is realized by nudging the individual ensemble members towards observations, according to a deterministic or probabilistic update rule. Critically, the full state covariance matrix need not be explicitly calculated, enabling a significant reduction in the computational and memory costs for ensemble size $< s$. However, approximating the state covariance from a small ensemble can produce spurious correlations, possibly causing the well-known problem of "filter divergence" (Houtekamer and Mitchell, 1998). This problem has been addressed through various techniques such as inflation (Anderson and Anderson, 1999) and localization (Greybush et al., 2011).

Many ensemble filter and smoother variants exist. The ensemble adjustment Kalman filter (Anderson, 2001) is a popular deterministic variant, belonging to the family of ensemble square root filters (Tippett et al., 2003). The iterative ensemble Kalman filter (IEnKF, Sakov et al. (2012)) and smoother (IEnKS, Bocquet and Sakov (2014)) have been shown to clearly outperform the standard EnKF and EnKS for certain dynamical systems. IEnKF has also been extended in order to take into account the model errors in Sakov et al. (2018). The recently proposed quantile-conserving ensemble filter framework (Anderson, 2022) supports arbitrary distributions for the prior and observation likelihood. Detailed descriptions of the ensemble Kalman smoother can be found in e.g. Cosme et al. (2012); Bannister (2017); Bergou et al. (2019).

2.5 Extensions beyond the standard data assimilation setup

The "standard" DA scenario in the context of Eq. (1)-(3) applies to many important problems and provides a useful framework for defining and comparing the major classes of DA methods. However, as ADDA supports assimilation tasks beyond this standard scenario, we next describe its capabilities in full generality.

ADDA supports a more general class of dynamical systems than the nonlinear dynamics with Gaussian noise of Eq. (1), including both deterministically and probabilistically evolving systems with arbitrary noise distributions as in Eq. (5). ADDA's strong-constraint 4D-Var and Ensemble Kalman methods require only a (possibly stochastic) time stepping function $\mathcal{M}$. The process noise distribution is restricted to be Gaussian for (ensemble) Kalman methods but can take any form for weak-constraint 4D-Var, as long as the associated log-probability can be computed analytically. In addition, while the time-discretized dynamics in Eq. (1) imply a fixed time step, ADDA also supports systems evolving continuously in time, for which a time step dt controls the extent of time integration on each call to $\mathcal{M}$. This provides support for irregularly spaced observations, as demonstrated in Sect. 4.2.3. Furthermore, while Eq. (1) assumes an autonomous system, in ADDA $\mathcal{M}$ can receive external inputs such as boundary conditions or forcings. An illustrative example of assimilating a system with a forcing input will be considered in Sect. 4.5.

ADDA also supports a more diverse set of observation types. Eq. (2) assumes Gaussian observation noise $\boldsymbol{\eta}_t$ around a deterministically computed conditional mean $\mathcal{H}_t(\mathbf{x}_t)$, but ADDA supports arbitrary conditional distributions of observations

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

given states $p(\mathbf{y}_t|\mathbf{x}_t)$. This allows for heteroscedastic and non-Gaussian observations, which can occur in atmospheric, ocean and ecosystem dynamics. It also supports observations that depend on the state at multiple time points, for example through temporal interpolation or direct observation of the slope for a long-term trend.

3 Implemented structures and methods

Here, we give an overview of DA methods implemented in ADDA: the Kalman filter and smoother, ensemble Kalman filter and smoother, and strong- and weak-constraint 4D-Var with a single or a sliding window. We also give some technical details on the expected structure of the state and observation variables, which are implemented in a general enough way to allow for easy extensions to numerous other assimilation methods. We do not specifically discuss the implementations of the Kalman filter and smoother, as those are simply consistent with standard practice.

3.1 The state variable

The state $\mathbf{x}_t$ of a dynamical system, evolved in time by Eq. (1), is implemented in ADDA's `State` class, designed for maximum flexibility and used by all DA methods. A `State` instance represents the system state at one or more time steps, i.e. it contains $\mathbf{x}_{0:T}$, along with time stamps $\tau_{0:T}$ for each step. This design choice allows for representation of state sequences with variable time stepping. ADDA's `State` class builds on the PyTorch `TensorDict` class (Bou et al., 2024), which encapsulates multiple `torch.Tensor` arrays with support for parallelization across shared dimensions. `State` objects incorporate two shared dimensions: a batch/ensemble dimension and the time axis. The resulting parallelization of time stepping across ensemble members and time steps is critical for efficient implementation of certain DA methods.

In addition to these shared dimensions, the system state can incorporate multiple named fields, each with its own dimensionality and spatial structure. Thus, system states can range from simple unstructured vectors to complex structures incorporating many variables on staggered or collocated grids, or unstructured meshes. This ability to incorporate variable fields at multiple resolutions is used for the two-level Lorenz system (Lorenz, 2005), with a practical example in Sect. 4.3.

3.2 Observation operators

A central element of data assimilation is the observation operator, which relates the state $\mathbf{x}$ to the observations $\mathbf{y}$ through Eq. (2) (for Gaussian observation noise), Eq. (6) (other distributions, one time point per observation) or Eq. (4) (full generality). In ADDA, the observation operator is a class implementing methods for sampling, evaluating and generally working with the density $p(\mathbf{y}_{0:T}|\mathbf{x}_{0:T})$.

The observation operator's density evaluation method must support automatic differentiation for use with variational methods, but this is not required for (ensemble) Kalman methods. Observation operators can also optionally implement the conditional mean $\mathcal{H}$ for use by Ensemble Kalman methods. For Kalman filtering and smoothing, the conditional mean must also be a linear function of the state.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

While ADDA users can easily define their own classes inheriting from a base `ObservationOperator`, implementations are also provided for several common observation scenarios. The simplest of these describes noisy observations of all state variables, with the option for noise levels that vary across time, location or field. Another pre-built observation operator supports masking, with observations for only some fields, locations or time steps. ADDA also provides operators describing arbitrary linear transformations from states to observations for use with (ensemble) Kalman methods, as well as temporal interpolation of state variables between time steps.

3.3 System dynamics

ADDA implements the system dynamics $\mathcal{M}$ as a function (python `Callable` type) with four inputs: $\mathbf{x}_t$, dt, F_t^d and F^s. The time step dt controls how far in time the system should be advanced (possibly using smaller or adaptive internal steps). Dynamic inputs F_t^d are provided at each time step to represent time-varying forcings or boundary conditions, while static inputs F^s represent configurable aspects of the system that do not change over time, such as scalar parameters (e.g. fluid viscosity) or spatially extended fields (e.g. ocean bathymetry). While $\mathcal{M}$ always takes 4 inputs to ensure compatibility across ADDA's interfaces, in many practical cases, not all 4 inputs will be used in a specific system implementation. We present in Sect. 4.2.3 an experiment where specifying the time increment dt is necessary, and in Sect. 4.5 an example where dynamic inputs are provided.

3.4 Heuristics for initializing the states

For both variational and ensemble methods, the initialization of system states can impact both the accuracy and speed of DA algorithms. For variational methods, the assimilation cost is usually not convex, so different initializations might lead to different local minima of the cost function. Ensemble methods require an initial state ensemble, and may never converge to sensible posterior estimates of the state distribution if the initial estimate is too far from the true initial state.

In some cases, system states can be initialized using historical averages such as Earth system climatologies. When these are unavailable or insufficiently precise, ADDA offers a flexible set of initialization heuristics that make use of available observations. Examples of these initializations include linear interpolation of observations over time, nearest neighbor interpolation, and user-provided default values for each field and/or location.

3.5 Variational methods

3.5.1 Strong-constraint 4D-Var

Strong-constraint 4D-Var, often simply called 4D-Var, is a widely used variational DA method (technical description in Sect. 2.3). While deriving an adjoint model for complex dynamical systems is notoriously difficult, computing the necessary adjoints is straightforward in an automatic differentiation framework. In the ADDA implementation, the user is asked to provide the dynamical model $\mathcal{M}$ following the conventions described in Sect. 3.3, the set of observations $\mathbf{y}_{0:T}$ and the corresponding observation operator as described in Sect. 3.2. The prior distribution on $p(\mathbf{x}_0)$ can be provided, but will otherwise be

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

assumed to be uniform so the first term of the right-hand side vanishes in Eq. (10). Forcing and/or boundary data can also be provided for each time step.

Strong-constraint 4D-Var usually assumes Gaussian distributions for the initial state prior and observation noise, but ADDA's implementation does not require these assumptions. While the examples in Sect. 4 use Gaussian distributions, ADDA users can define and employ other distribution classes implementing density evaluation and sampling methods.

3.5.2 Weak-constraint 4D-Var

ADDA computes the loss in Eq. (9) using parallelization over the time axis for $\mathcal{M}(\mathbf{x}_{0:T-1})$. This parallelization considerably reduces the overall computation time for each optimization step, especially on GPUs.

In addition to the inputs required for strong-constraint 4D-Var, the weak-constraint variant requires a model error distribution for all time steps on which the optimization is performed. ADDA does not require model error distributions to be Gaussian.

3.5.3 Sliding-window 4D-Var

ADDA's variational DA methods can be used with a sliding-window approach, in which observations are divided into assimilation sub-windows, and a separate 4D-Var problem is solved for each sub-window. The result of each sub-window's optimization is used to define the mean of a background prior for the subsequent sub-window, by directly selecting (weak-constraint) or simulating to (strong-constraint) the appropriate time step. ADDA's sliding-window 4D-Var methods further support a `covariance_factor` argument that controls the tightening of the background prior for all but the first assimilation sub-window, reflecting the fact that a previously assimilated sub-window decreased the uncertainty on the initial state for the next sub-window. As illustrated in Sect. 4.4, a covariance factor < 1 can be used to express increased confidence in the prior distribution when some observations have been assimilated in a previous sub-window. ADDA supports both overlapping and non-overlapping assimilation sub-windows.

This approach allows variational DA to be applied to long observation sequences, for which optimization over a single window would encounter memory constraints, or for which the growing input-output gradients of chaotic dynamical systems would destabilize the optimization. Our implementations of sliding-window strong- and weak-constraint 4D-Var use essentially the same arguments as their single-window versions, with additional arguments for the sizes of the sub-windows and of the shifts between them, as well as construction of background priors after the first sub-window.

3.5.4 Gradient descent for variational methods

ADDA's variational DA methods employ gradient-based optimization. While computing gradients through $\mathcal{M}$ by automatic differentiation, we leverage the extensive tools for differentiable optimization implemented in PyTorch. Variational methods support PyTorch built-in as well as user-defined `Optimizer` classes, with support for `LRScheduler` classes for finer control of the optimization process. Additional arguments controlling optimizer behavior (e.g. line search function or momentum

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

weight) can be passed to variational DA methods. The default optimizer is the limited-memory BFGS (Liu and Nocedal, 1989).

3.6 Ensemble methods

An advantage of ensemble-based DA methods is that they do not require gradients from an adjoint model of the dynamics $\mathcal{M}$. Thus, ADDA's ensemble methods do not employ automatic differentiation, but nonetheless benefit from efficient GPU-accelerated computation in PyTorch, with parallelization across the ensemble to accelerate DA for large ensembles or states. Our implementation follows the classical "perturbed observations" approach, introduced by Burgers et al. (1998).

ADDA implements EnKF as a special case of the EnKS with zero lag, resulting in a simple and compact codebase. When employing an EnKS method, key choices include:

- The distribution from which to sample the initial ensemble (usually a Gaussian)
- The size of the ensemble, i.e. its number of members
- The lag over which to perform the smoothing
- The choice of linearizing the observation operator or using a nonlinear expression

All of these choices can be easily made in ADDA by specifying arguments when calling the EnKF or EnKS functions. In our implementation, the distribution of the initial ensemble is a diagonal Gaussian. Importantly, while some variants of the EnKS algorithm model the full smoothing distribution $p(\mathbf{x}_t|\mathbf{y}_{0:T})$, ADDA's EnKS is a fixed-lag smoother which makes the common assumption that the posterior on states at a given time step t is only influenced by the observations that are located at most d time steps later than t, where d is a chosen delay parameter. However, one can simply define $d \geq T$ in order to obtain the full smoothing distribution.

ADDA's default behavior follows most EnKF/S methods in linearizing the conditional mean observation $\mathcal{H}$ as a function of the state, but ADDA also supports the method described by e.g. Bannister (2017) that instead applies a nonlinear $\mathcal{H}$ to each ensemble member. Linearization requires the observation operator's class implement a `linearize` method, and all included observation operators (Sect. 3.2) do so. Note that the resulting matrix $\mathbf{H}$ has a size of (s, o_t). When both of these numbers are high, this can represent a very large matrix. Yet, in some cases, such as the "binary masking" observation operator, the structure of this matrix is extremely sparse by design. Thus, in order to avoid unnecessary computations, our implementations use sparse data structures from PyTorch in such cases.

3.7 Specification of the state time axis

ADDA's structure is centered on the `State` object (Sect. 3.1), whose attributes include the `TensorDict` class `fields` and the `Tensor` class `time_axis`. One important consideration when solving a data assimilation problem is the definition of the state time axis for the inferred trajectory. In the simplest cases, the inferred state will be defined on the same time axis as the set of available observations, which is the default behavior of our package. However, in some cases, the state variable

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

might be defined with a much finer time resolution than the observations for numerical stability reasons. Conversely, the observations might be defined at a very fine temporal resolution, so that inferring a state trajectory with this resolution would be unnecessarily costly. Observations may also be defined with irregular time steps. For these reasons, ADDA enables defining a time axis for inferring a state trajectory in 4D-Var, independently of the observation time axis. In this case, the observation operators must define the mapping from state trajectories to distributions on observation sets, defined on different time axes. ADDA currently provides one such operator that involves linear interpolation of a state trajectory with a regular time axis on an arbitrary (possibly irregular) time axis. An example illustrating this feature is presented in Sect. 4.2.3. ADDA's ensemble-based methods require the states and observations to use the same time axis, but allow uneven irregular spacing along this axis as shown in Sect. 4.2.3. ADDA's weak-constraint 4D-Var supports irregularly spaced observations along the time axis but requires regularly spaced system states to avoid complicating parallel computations.

4 Illustrative experiments

We demonstrate ADDA's capabilities on multiple experiments, covering a wide variety of dynamical systems, observation operators and DA techniques. As previously stated, variational DA requires dynamical systems to support automatic differentiation, with an implementation in either PyTorch or JAX. We used our own PyTorch differentiable implementations of the dynamical systems for most of the examples associated with the package, including linear dynamics, Lorenz-63 (L63), Lorenz-96 with one (L96-1L) and two (L96-2L) timescales, Kuramoto-Sivashinsky (KS-1D), Korteweg–De Vries (KdV), Kolmogorov flow[1] (KF) and advection and diffusion operators with an inert tracer. For the three-layer quasi-geostrophic system (QG-3L), we use the PyTorch implementation from Thiry et al. (2023) in Sect. 4.4. For the two-dimensional Kuramoto-Sivashinsky system (KS-2D), we use the Exponax library (Koehler et al., 2024) built in JAX, in order to demonstrate the ability of ADDA to bridge PyTorch and JAX gradients, in Sect. 4.8. We also used a JAX-PyTorch bridge with the Exponax implementation of the KdV system, in addition to our own PyTorch implementation. Our codebase includes multiple notebooks to illustrate the ease with which ADDA can be applied to these systems with diverse observation operators, without requiring the user to implement extensive custom software routines for each case. In the following sections, we present a selected subset of these examples in greater detail.

4.1 An introduction to variational data assimilation on the Lorenz-63 system

Our first example illustrates strong- and weak-constraint 4D-Var on the Lorenz-63 dynamical system, described in Sparrow (1982). The Lorenz-63 system is a major testbed for data assimilation. It is a very simple and low-dimensional set of equations, yet its chaotic behavior makes long-term forecasting impossible. The system has 3 variables x, y, z, from which the respective time derivatives can be written as follows:

$$\dot{x} = \sigma(y - x); \quad \dot{y} = x(r - z) - y; \quad \dot{z} = xy - bz. \tag{12}$$

[1] For this particular system, we use a neural emulator rather than a differentiable simulation.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

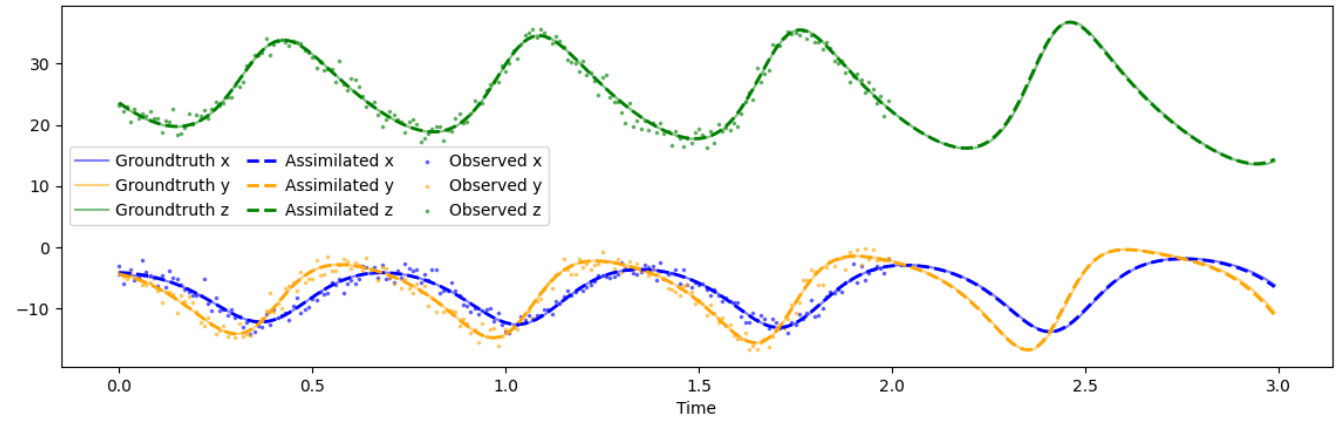

Figure 1. Visualization of the true values, noisy observations and assimilated trajectory for the Lorenz-63 system, with strong-constraint 4D-Var, in a case where all variables are observed with a Gaussian noise until time 2.0.

The parameters of these equations are set to their most common values $(\sigma, r, b) = (10, 28, 8/3)$. With these settings, the system's Lyapunov time (for which small perturbations grow by a factor of e) is roughly 1.10. As a reminder, we will use the notation $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ for the full state of the system, not to be mistaken with $x \in \mathbb{R}$, the first variable of the system. Conversely, $\mathbf{y}$ (not to be mistaken with y) denotes the set of available observations. The groundtruth time series for all experiments on this system are obtained by integrating the system (12) with a time step of 0.01, using the 4th order Runge-Kutta method.

4.1.1 Assimilating noisy but complete observations

We first consider an observation operator that adds Gaussian noise:

$$\mathbf{y}_t = \mathbf{x}_t + \boldsymbol{\eta}_t, \tag{13}$$

where $\boldsymbol{\eta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_3)$, and $\mathbf{I}_3$ is the 3×3 identity matrix, so that the observation noise is uncorrelated. In this case, the assimilation task reduces to a denoising task. We solve it using strong-constraint 4D-Var over a window of 2 time units, i.e., 200 time steps with no background prior term (implicitly corresponding to a uniform background prior on $\mathbb{R}^3$). We initialize the variables with the observed values at the initial time and perform 5 L-BFGS optimization steps with a learning rate of 0.1. The result is an assimilated initial state, which is expected to be significantly closer to the true initial state than the initial guess (i.e., the noisy observations at the first time step). This estimated initial state is then rolled out in time using the same numerical integration scheme as for generating the groundtruth time series. As can be seen from Fig. 1, we can convincingly denoise the observations and even accurately extend the predictions beyond the range of the observations.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

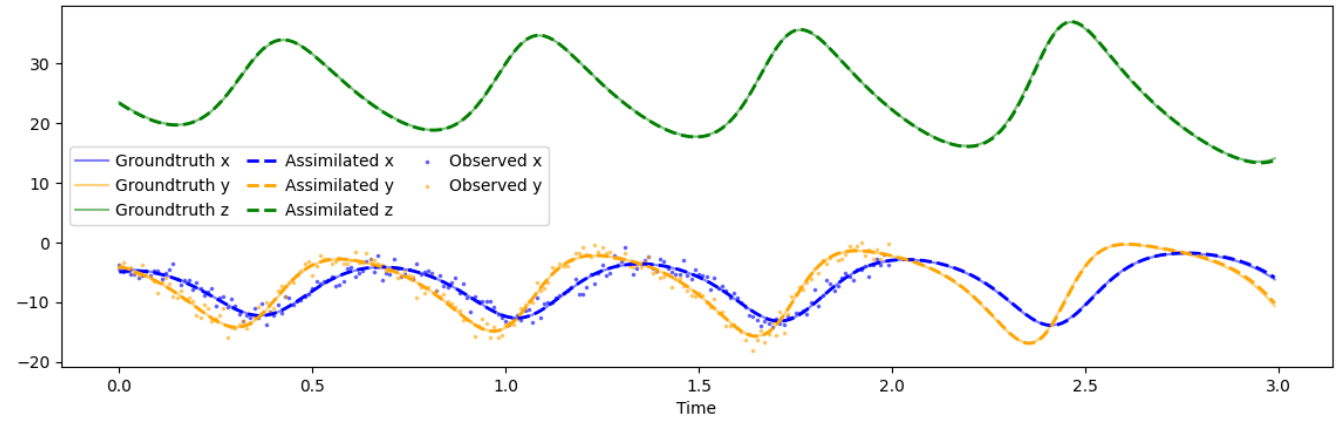

Figure 2. Visualization of the true values, noisy observations and assimilated trajectory for the Lorenz-63 system, with strong-constraint 4D-Var, in a case where observations are entirely missing for the z variable.

4.1.2 Imputing a missing system state variable

We next consider a case where the variables x and y are always observed with a Gaussian noise of standard deviation 1, while the variable z is never observed:

$$\mathbf{y}_t = \mathcal{H}(\mathbf{x}_t) + \boldsymbol{\eta}_t = (x_t, y_t) + \boldsymbol{\eta}_t, \quad \boldsymbol{\eta}_t \sim \mathcal{N}(0, \mathbf{I}_2) \tag{14}$$

where $\mathcal{H} : \mathbb{R}^3 \to \mathbb{R}^2$ truncates the third variable: $\mathcal{H}(x, y, z) = (x, y)$. A key goal of the assimilation is to infer the missing variable z over time by using only the available (noisy) observations on x and y. Since the observation operator $\mathcal{H}$ can be differentiated using straightforward automatic differentiation, strong-constraint 4D-Var can be applied as in the denoising example. The only difficulty is initializing the unobserved variable z before starting the gradient descent. We arbitrarily set the initial value to 25.0, which is relatively close to the stationary mean of this variable.

As can be seen in Fig. 2, despite the more difficult observation setting, we can accurately reconstruct the true state, including the unobserved third variable. Extending the assimilation beyond the window of observations again results in accurate forecasts.

4.1.3 Weak-constraint 4D-Var on a long window

Single-window strong-constraint 4D-Var limits the duration of observation sequences for chaotic systems such as Lorenz-63, as chaotic behavior makes it impossible to optimize initial conditions based on observations far into the future. In practice, this results in ill-behaved gradients when the assimilation window is long relative to the Lyapunov time of a system, preventing the method from converging to the correct solution, even with a favorable observation pattern. In contrast, weak-constraint 4D-Var does not require the optimized state trajectory to precisely follow the known system dynamics $\mathcal{M}$, and its long-term predictions need not arise deterministically from a single initial state. Instead, our implementation of weak-constraint 4D-Var relies only on one-time-step predictions, as detailed in Sect. 3.5.2. This enables both numerical stability and fast parallel computation. To

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

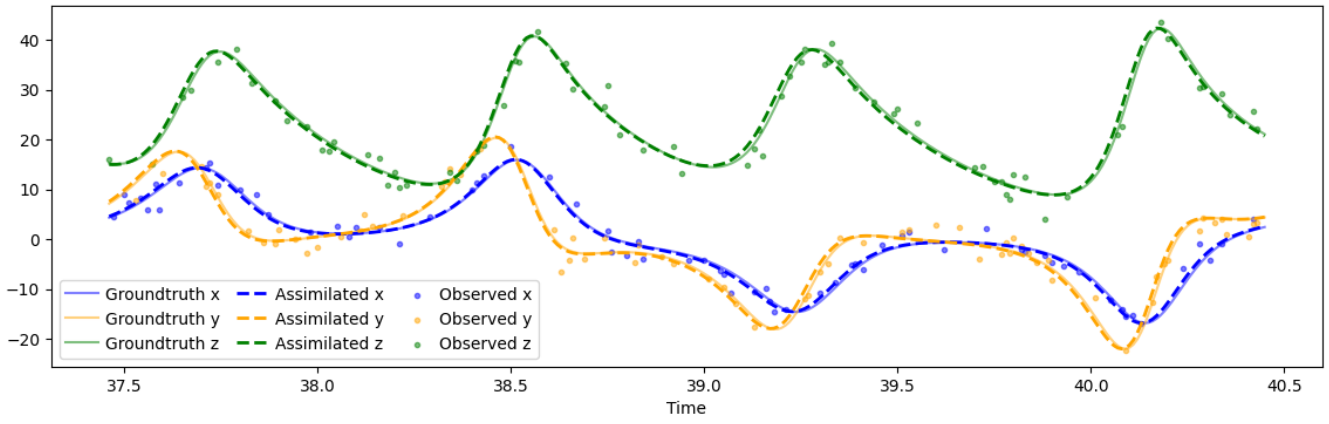

Figure 3. Visualization of the true values, noisy observations and assimilated trajectory for the Lorenz-63 system, with weak-constraint 4D-Var performed on a long time series with a sparse observation pattern.

demonstrate this, we now consider a much longer observation window with 100 time units, corresponding to 10^4 time steps. Observations are randomly masked, and each variable at each time step is observed with $25\,\%$ probability. Observations are corrupted by additive Gaussian white noise with standard deviation 2, higher than in the previous examples. Note that this strategy does not result in regularly spaced observations for any of the 3 variables.

We use weak-constraint 4D-Var with no background prior term and a diagonal Gaussian with mean zero and standard deviation $10^{-2}\mathbf{I}_2$ for the model error distribution. As weak-constraint 4D-Var optimizes over the full system state trajectory, we initialize this trajectory using nearest-neighbors interpolation over time for each variable. We optimize using 50 L-BFGS iterations, with a learning rate of 1. We display the assimilation result for a randomly selected crop of the observation window in Fig. 3. Despite the relatively low signal-to-noise ratio and the sparsity of the observations, the assimilated trajectory closely matches the true trajectory.

4.2 Data assimilation on the Lorenz-96 system

The Lorenz-96 system (Lorenz and Emanuel, 1998) is another relatively simple chaotic system, designed to represent the evolution of an atmospheric variable over a latitude band. It is more complex than Lorenz-63 due to its spatial structure: the system has n variables $x_1, ..., x_n$ on a periodic domain, with their time derivatives expressed as:

$$\dot{x}_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F. \tag{15}$$

Time is expressed in arbitrary units with one unit corresponding to 5 days of the evolution of the atmosphere, in terms of Lyapunov time and predictability. We use the most common setup, with $n = 40$ and $F = 8$, for which the Lyapunov time is approximately 0.60. In Sect. 4.2.1, and 4.2.2, the equations are integrated numerically using a 4th-order Runge-Kutta scheme

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

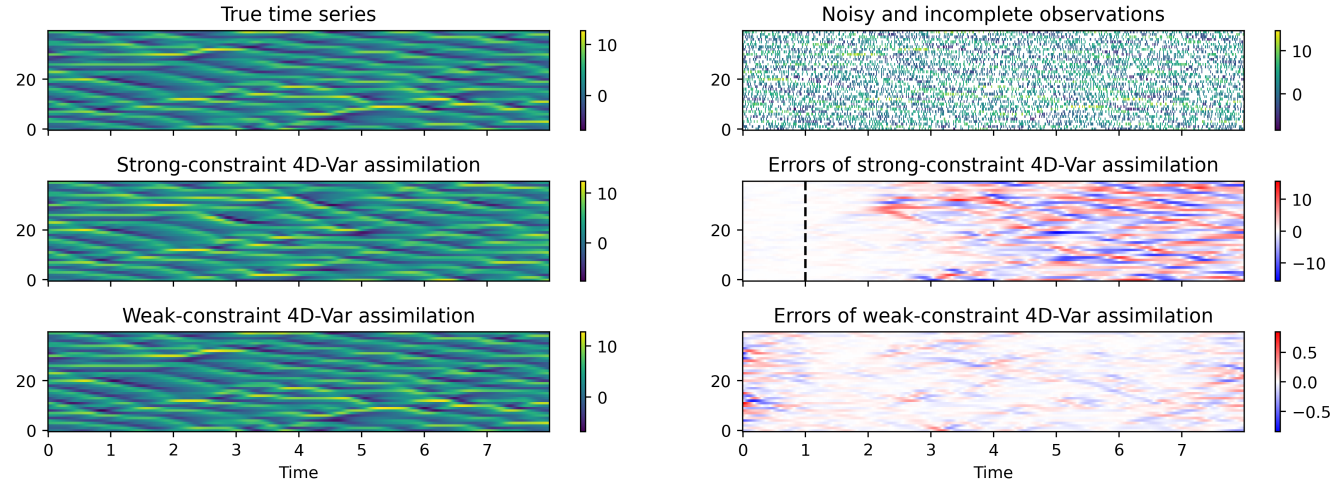

Figure 4. Visualization of the true state values, the corresponding sparse and noisy observations, and the state estimations and errors obtained by strong- and weak-constraint 4D-Var on the Lorenz-96 dynamical system. The dashed vertical line marks the limit of assimilated observations for strong-constraint 4D-Var.

with a time step of 0.01, yielding weakly nonlinear dynamics at each time step. In Sect. 4.2.3, they are integrated with an irregular time step, as specified later.

The experiments of Sect. 4.2.1, 4.2.2 and 4.2.3 all use the same observation setup, consisting of a binary mask and white Gaussian noise. At each time step, we randomly mask 30 of the 40 variables. For the remaining variables, we add a Gaussian noise of variance 1.

4.2.1 Strong- and weak-constraint 4D-Var

The 4D-Var algorithms are initialized with nearest-neighbor interpolation over time, as in Sect. 4.1.3. For the strong-constraint variant, only the estimate of the initial state is required. For the weak-constraint variant, we consider the model errors to follow a Gaussian distribution with zero mean and a diagonal covariance matrix with diagonal coefficients 10^{-4}, as in Sect. 4.1.3. Due to the chaotic nature of the Lorenz-96 system, we restrict the strong-constraint optimization to an observation window of 100 time steps to maintain stability, while the weak-constraint assimilation is performed on a window of 800 time steps. The assimilation costs of both 4D-Var variants are minimized using L-BFGS, with a learning rate of 0.1 for strong-constraint and 1 for weak-constraint.

Results, presented in Fig. 4, are consistent with the expected behavior of the methods; the absolute error at the beginning of the time series with respect to the true state is significantly smaller than the amplitude of the observation noise. Due to the chaotic nature of the Lorenz-96 system, the absolute error of strong-constraint 4D-Var grows exponentially once the integration extends beyond its observation window. Thus, after roughly 4 time units (i.e., 400 time steps), the forecast becomes entirely unskilled, in the sense that the estimated state is only as close to the true state of the system as the expected distance between two

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

independent states sampled from the stationary distribution of the Lorenz-96 system. This could be alleviated by periodically performing a new assimilation with new observations, corresponding to sliding-window 4D-Var.

In contrast to single-window strong-constraint 4D-Var, weak-constraint 4D-Var can successfully assimilate all observations in a single window spanning several Lyapunov times of the system, which is made possible by the introduction of model errors. As a result, assimilation errors are relatively low and stable in time. The largest errors are observed on the boundaries of the assimilation window, particularly on the left boundary, a common phenomenon explained by the relative scarcity of surrounding observations at those time steps.

4.2.2 Ensemble Kalman filter and smoother

We next apply the Ensemble Kalman Filter (EnKF) and Smoother (EnKS) to the Lorenz-96 system, with the same observation scenario as previously. Key factors influencing the success of ensemble-based methods include the ensemble size and initial state distribution. In many works involving variants of EnKF for the Lorenz-96 system (e.g. Sakov et al. (2012); Bocquet and Sakov (2013)), knowledge of the true initial state is used to define the ensemble mean. While this sort of experiment can reflect a realistic setup in which a previous observation window provides an informative prior, we found it essential for ADDA to incorporate and test initializations relying solely on available observations, as for variational methods. This choice is crucial since the quality of the initialization of EnKF and EnKS largely affects their capacity to converge to values close to the true state.

We therefore initialize EnKF/S using nearest-neighbor interpolation over time, as previously. With this heuristic, the error of the initial guess relative to the true initial state is expected to exceed the observation noise, since it is driven by two factors: the observation noise itself and the fact that the observed value comes from a subsequent time step. We neglect this second factor and set the standard deviation of the initial ensemble to 1.0, the same as the observation noise. With such an initialization, a relatively high ensemble size was required to prevent divergence as the ensemble methods proceeded over subsequent time steps. A more accurate initial ensemble mean would enable a significant reduction in the ensemble size, as thoroughly analyzed in e.g. Bocquet et al. (2021). We choose an ensemble size of 50 for our main results, which ensures fast convergence and an accurate estimate of the system state. We also use an inflation factor of 1.01. For the EnKS, we use a lag of $d = 50$ time steps.

The results of the EnKF and EnKS assimilations are shown in Fig. 5. The largest errors for both methods occur in the first few time steps of the assimilation window. These large errors are caused by the imperfect initialization described above, after which the ensemble mean progressively gets closer to the true state. The figure also clearly shows the benefit of the smoothing over filtering, as the root mean squared error from the former is significantly lower throughout the assimilation window.

4.2.3 Handling observations with an irregular time sampling pattern

In the previous examples, the observations are sampled from a regular temporal grid, with a fixed time step. Here, we show that our framework is substantially more flexible, as it imposes no constraint on the regularity of the time steps of the observations.

Regarding strong-constraint 4D-Var and the ensemble Kalman methods, the irregular temporal sampling does not induce additional difficulty. By default, these methods estimate the state on exactly the timestamps of the observations. For the ensem-

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

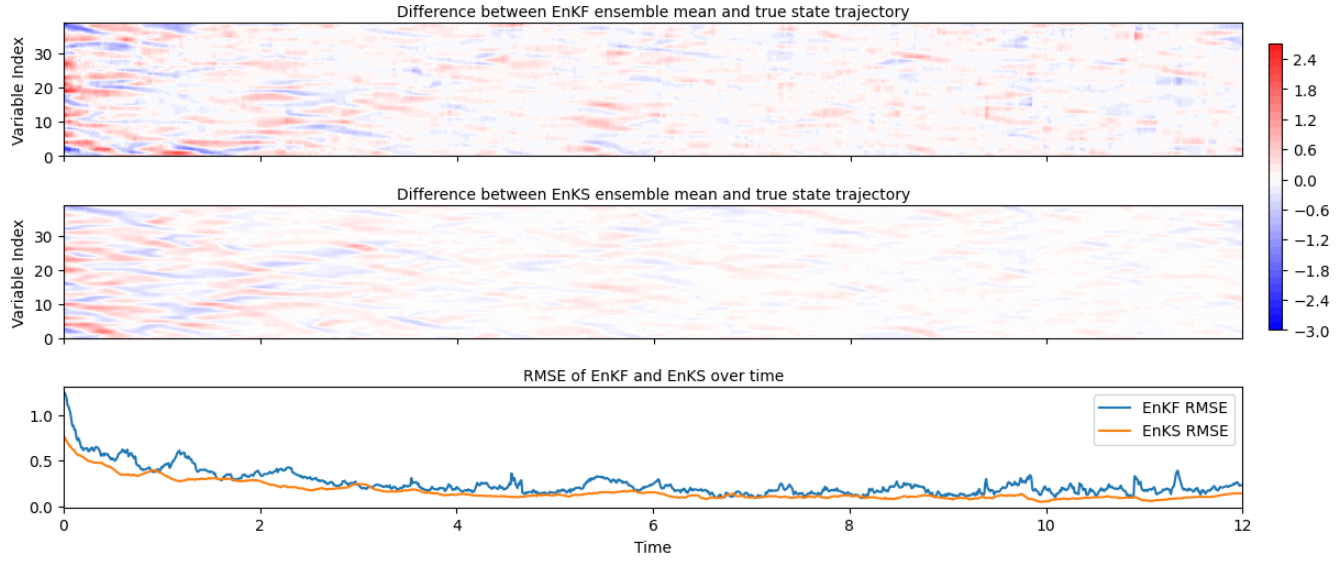

Figure 5. Comparison between the errors obtained by the assimilations of an Ensemble Kalman Filter (EnKF) and a similarly parameterized Ensemble Kalman Smoother (EnKS) on the same set of observations from the Lorenz-96 dynamical system.

ble Kalman methods, this is in fact the only available mode in ADDA, as these methods do not have a `state_time_axis` argument. In contrast, the 4D-Var methods allow the state time axis to be specified independently from the observation time axis, with the constraint that the state time axis must be regular for weak-constraint 4D-Var (to enable the parallel computations). When the state time axis differs from the observation time axis (which, in this section, applies only to weak-constraint 4D-Var), the observation operator should be able to relate them. As mentioned in Sect. 3.2, we currently provide an observation operator class that linearly interpolates a regularly sampled state time axis to the observation time axis.

We uniformly sample 800 observation times over a time domain of 8 time units and sort them, so that the mean difference between two consecutive observation times is 0.01, matching our other L96 experiments. The groundtruth time series is generated by integrating Eq. (15) on precisely this irregular time axis.

We show the groundtruth, observations and assimilation results of all methods in Fig. 6. These results are largely consistent with those from the regular sampling case in the previous sections. Indeed, the single-window strong-constraint 4D-Var method can accurately match the true state over the first few time units before it diverges due to the chaotic nature of the system, while the weak-constraint 4D-Var and EnKF methods accurately reconstruct the true time series, with the highest errors on the edges of the observation window for weak-constraint 4D-Var and at the beginning of the window for EnKF. None of these data assimilation methods appears to be significantly affected by the irregular time axis of the observations. Weak-constraint 4D-Var obtains, as in the previous experiments, the lowest assimilation errors.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

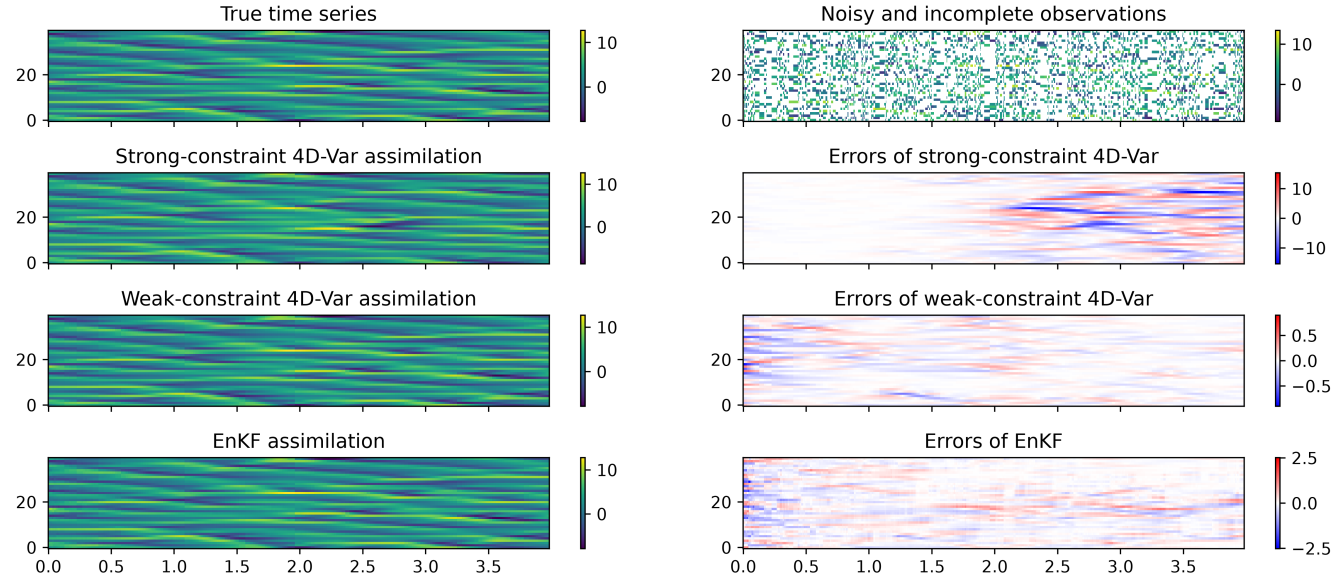

Figure 6. Visualization of the groundtruth time series for the Lorenz-96 dynamics with irregular time sampling, the corresponding sparse and noisy observations and the assimilated states and associated errors for the strong- and weak-constraint 4D-Var and EnKF. For ease of visualization, we only show the first half of the whole observation domain.

4.3 Weak-constraint 4D-Var on the two-timescale Lorenz-96 system

Following the examples on the basic Lorenz-96 system described in Sect. 4.2, we now consider the two-timescale extension of this system, discussed in e.g. Lorenz and Emanuel (1998) and Lorenz (2005). The system consists of two sets of variables: n slow-evolving variables $(X_k)_{k=1}^n$ and, for each of these, J fast-evolving variables $Y_{1,k},...,Y_{J,k}$. Thus, the system comprises n slow variables and nJ fast variables, for a total dimension of $n(J+1)$. These variables evolve in time according to:

$$\dot{X}_k = (X_{k+1} - X_{k-2})X_{k-1} - X_k + F - \frac{hc}{b}\sum_{j=0}^{J} Y_{j,k}, \tag{16}$$

$$\dot{Y}_{j,k} = -cbY_{j+1,k}\,(Y_{j+2,k} - Y_{j-1,k}) - cY_{j,k} + \frac{hc}{b}X_k. \tag{17}$$

As shown by Eq. (16), the slow variables $X_1,...,X_n$ evolve similarly to the state of the single-timescale Lorenz-96 system in Eq. (15), except that they are also coupled to the fast variables through the last term of the equation. The fast variables $Y_{j,k}$ are influenced by their neighbors and their associated slow variables, and typically evolve more rapidly. F, b, c and h respectively characterize the forcing on the slow variables, the mutual influence between the fast variables, the relative rate of evolution of the fast variables compared to the slow ones and the strength of the coupling between slow and fast variables. We numerically integrate Eq. (16) and (17) using the most common parameter values: $h = 1, b = c = F = 10$. We set the size of the system to

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

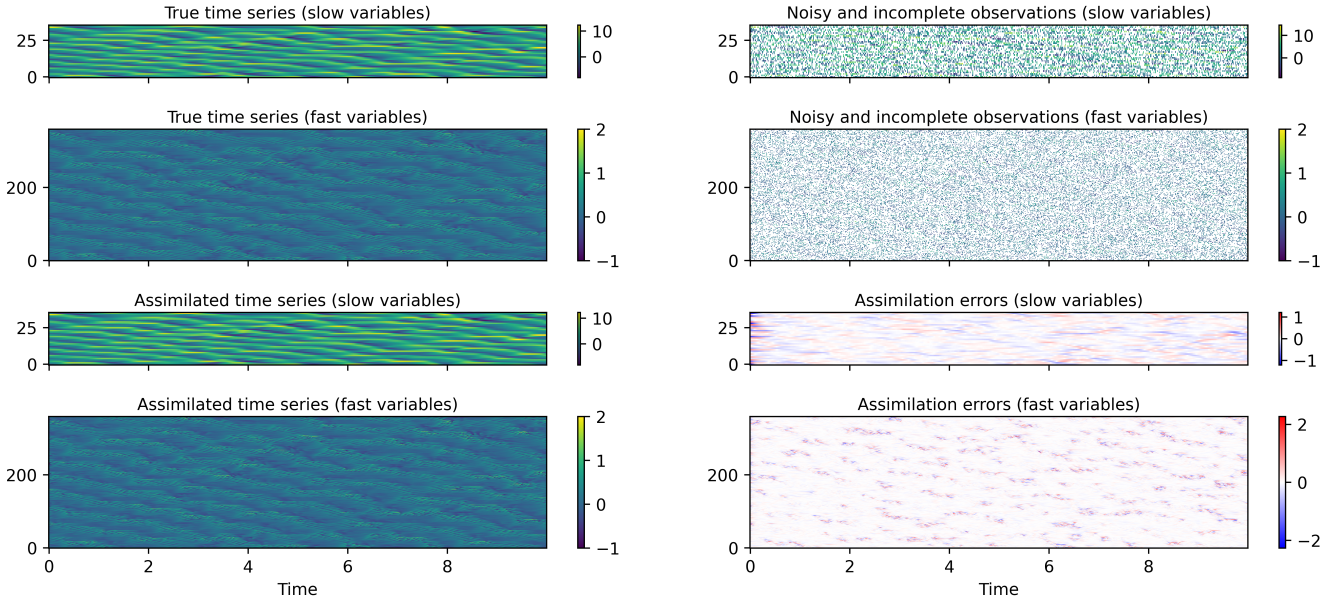

Figure 7. Visualization of the true values, sparse and noisy observations, assimilated time series (with weak-constraint 4D-Var) and assimilation errors for the two-timescale Lorenz-96 system, using observations from both the slow and fast variables.

$n = 36$ and $J = 10$, yielding $n = 36$ slow variables and $nJ = 360$ fast variables. We use the 4th order Runge-Kutta scheme, with a time step of 0.01 and a 1000-step assimilation window.

The two-timescale Lorenz-96 system is challenging to incorporate in DA systems that require a fixed spatial grid for the system states, since here the state consists of two periodic fields of different sizes, one of them being one-dimensional and the other two-dimensional. This state can, however, be conveniently represented in ADDA with a single `State` object, using the structure described in Sect. 3.1. Likewise, a binary masking observation operator with different observation proportions and noise standard deviations for the two fields can be defined straightforwardly by defining the observation proportion and noise amplitudes as dictionaries containing one floating value for each field. While still a simple system, this illustrates how ADDA handles systems with heterogeneous state structures, which is necessary to accommodate realistic geophysical problems.

We apply weak-constraint 4D-Var in 2 observation scenarios. In Sect. 4.3.1, sparse and noisy observations are available for both the slow and the fast variables. In Sect. 4.3.2 and 4.3.3, we have access to sparse and noisy observations of only the slow variables. The fast variables are ignored by the DA procedure in Sect. 4.3.2 (effectively acting as unresolved sub-grid-scale processes) but explicitly inferred in Sect. 4.3.3.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

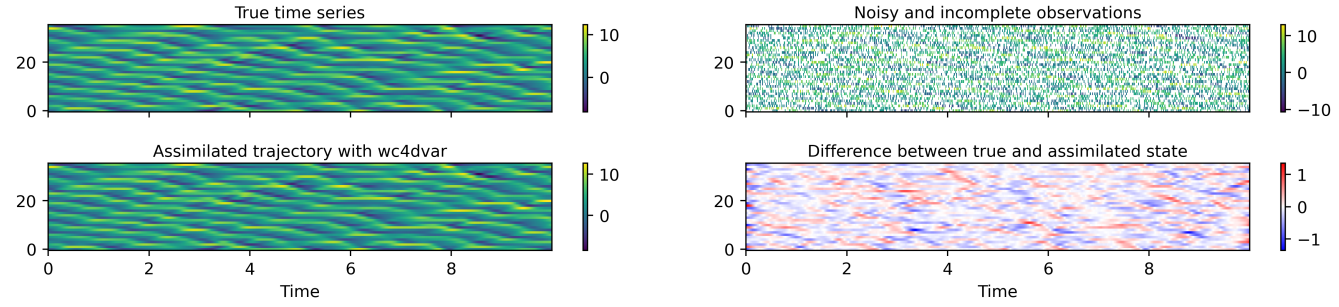

Figure 8. Visualization of the true values, sparse and noisy observations, assimilated time series (with weak-constraint 4D-Var) and assimilation errors for the slow variables of the two-timescale Lorenz-96 system when the influence of the fast variables is completely ignored.

4.3.1 Assimilation of jointly observed slow and fast variables

Here, we randomly mask 27 of the 36 slow variables and 334 of the 360 fast variables at each time step, so that $75\,\%$ of the slow and $90\,\%$ of the fast variables are unobserved. For the unmasked variables, we add a Gaussian random noise with a standard deviation of 1 for the slow variables and 0.1 for the fast variables.

We solve this assimilation problem with weak-constraint 4D-Var, initialized with nearest-neighbor interpolation in time. The model error distribution is defined as two separate diagonal Gaussian distributions for the slow and fast variables. Since we know that the fast variables evolve more rapidly than the slow ones and are observed more sparsely, their model error standard deviation is set to 5 times higher than that of the slow variables. This ratio must be selected empirically based on DA performance in cases such as this where model dynamics are precisely known. The results of DA (Fig. 7) show that the reconstructions are qualitatively accurate. Notably, the assimilation errors on the fast variables mostly spike in regions of rapid evolution and remain small elsewhere, while the errors on slow variables always remain relatively small.

4.3.2 Assimilation of the slow variables, ignoring the effect of the fast ones

We next assimilate the observations of the slow variables under the same conditions as in the previous example, but the fast variables are unobserved and entirely unaccounted for. While groundtruth system states were generated using dynamics $\mathcal{M}$ from Eq. (16)-(17), we assimilate using $\mathcal{M}$ for the one-scale Lorenz system (Eq. (15)), neglecting the last term of the true state equation (16). Since the dynamical model is significantly misspecified in this case, the model error that we use has a much higher standard deviation than the one used in Sect. 4.2.1 and 4.3.1, where the model was actually correct. Specifically, we set it to a Gaussian with mean 0 and covariance $10^{-2}\mathbf{I}_n$.

DA results are shown in Fig. 8. Although the reconstruction remains qualitatively accurate, a comparison with Fig. 7 shows that, as expected, ignoring the influence of the fast variables on the slow ones leads to a significant loss of accuracy with respect to the case of a joint assimilation of the two scales.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

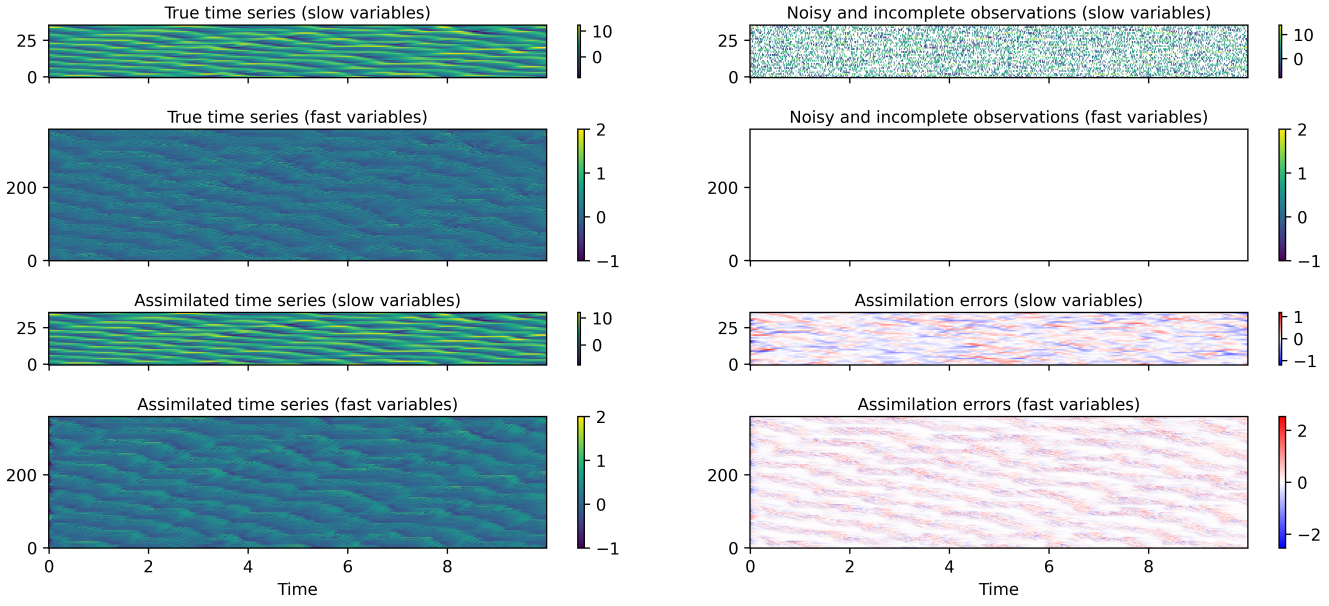

Figure 9. Visualization of the true values, sparse and noisy observations, assimilated time series (with weak-constraint 4D-Var) and assimilation errors for all variables of the two-timescale Lorenz-96 system when the fast variables are not observed but still inferred and taken into account for the assimilation of the system.

4.3.3 Assimilation of the slow variables with joint inference of the unobserved fast ones

Finally, we consider an intermediate case in which the fast variables are completely unobserved, as in the previous experiment, but their influence on the slow ones is still accounted for, and their values are jointly inferred with the slow variables. Thus, this example combines the assimilation method from Sect. 4.3.1 with the more challenging observation scenario of Sect. 4.3.2. Here, we initialize unobserved fast variables to 0 across the observation window, as this value lies within their stationary distribution.

The result of the optimization is displayed in Fig. 9. The reconstruction is qualitatively accurate for both the observed slow variables and the inferred fast variables. Comparison with Fig. 8 reveals that taking the influence of the fast variables on the slow ones into account, even with no additional observations, enables a substantial improvement in the accuracy of the assimilation for the slow variables. Besides, a comparison with Fig. 7 shows that, as expected, the absence of observations for the fast variables is still detrimental to the reconstruction quality, for the fast but also for the slow variables.

4.4 A larger-scale example with multi-layer quasi-geostrophic equations

As a larger-scale example, we consider the multi-layer quasi-geostrophic equations, as described by Hogg et al. (2005). Since differentiable implementations of these equations are still uncommon, we used (and slightly modified) the implementation proposed by Thiry et al. (2023), with equations for pressure and potential vorticity over three layers with background thick-

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

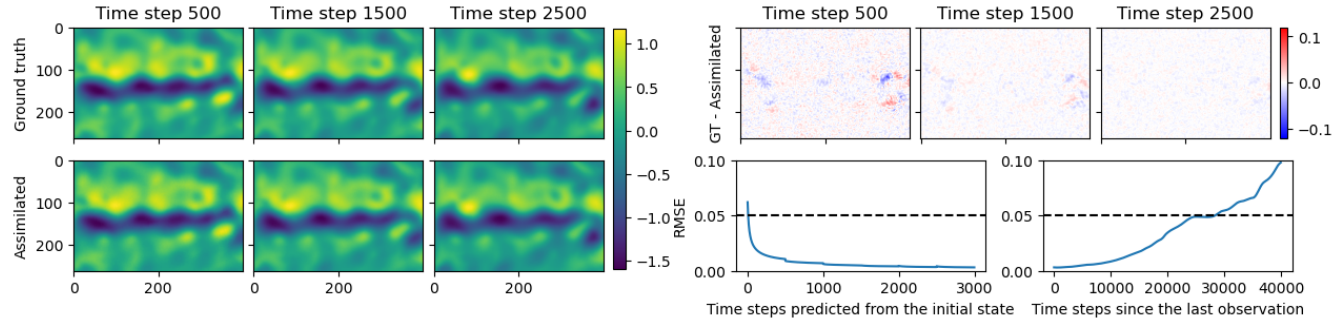

Figure 10. Summary of our data assimilation results on the three-layer quasi-geostrophic equations. In the left part of the figure, we show the true and assimilated pressure fields of a central part of the middle layer, for three time steps contained in the observation domain. The corresponding difference fields are displayed in the top-right panel of the figure. On the bottom right part, we plot the global assimilation root mean squared errors as a function of time, both inside of the observation domain (left) and beyond it (right). The dashed horizontal lines correspond to the amplitude of the noise on sparse observations.

nesses 350 m, 750 m and 2900 m. We use an eddy-resolving parameterization on a rectangular domain with a horizontal size of $3.84 \cdot 10^6$ m (width) by $4.8 \cdot 10^6$ m (height). This domain is discretized on a 769×961 grid at 5 km resolution. The time step of 600 seconds respects the Courant–Friedrichs–Lewy (CFL) condition. The state is defined by the three pressure fields of the vertical layers. Further information on the system and its implementation can be found in Thiry et al. (2023).

We work on an observation domain of 3000 time steps, corresponding to approximately 21 days. As in Sect. 4.2, we generate synthetic observations from the true state trajectory with random masking and Gaussian noise: $0.5\,\%$ of the variables are observed, with a noise standard deviation of 0.05. To avoid having to split the simulated state trajectory across multiple GPUs (we used one A100 with 80 GB of memory), we use sliding-window strong-constraint 4D-Var with sub-windows of 500 time steps. We initialize each element of the initial state to its first observed value.

For the first sub-window, our background prior distribution is a spherical Gaussian centered on the initial guess and with a standard deviation of 0.075 (i.e. 1.5 times the standard deviation of the observations). For the following sub-windows, using the `covariance_factor` argument described in Sect. 3.5.3, we decrease the background prior s.d. by a factor of 4. For each sub-window, we minimize the 4D-Var cost with 40 Adam optimizer steps using a triangular cyclic learning rate scheduler that varies the learning rate from 2×10^{-3} to 5×10^{-3} with 20 iterations per cycle. As explained in Sect. 3.5.4, this gradient descent strategy is readily supported by ADDA, which accepts any combination of PyTorch optimizers and schedulers with prescribed parameters.

Comparing the true and assimilated states at several time points (Fig. 10, left), we observe differences 1-2 orders of magnitude smaller than the states themselves (Fig. 10, top right). RMSE begins slightly above the level of observation noise (dashed black line), and decreases steadily over time (Fig. 10, bottom right). The discontinuities visible in the RMSE every 500 time steps correspond to the transitions between sub-windows of the assimilation.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Finally, to evaluate the robustness of our assimilation, we compare forecasts beyond the final observation to true states. The RMSE gradually increases with the forecast horizon (Fig. 10), but takes more than 20000 time steps (significantly longer than the observation domain) to exceed the noise amplitude. An animated version of these DA results is available in the ADDA repository.

4.5 Vertical diffusion and advection of an inert tracer

To demonstrate how ADDA can incorporate time-varying inputs to the assimilated system, we perform single-window strong- and weak-constraint 4D-Var for the vertical dynamics of an inert tracer in a 1D water column. We consider a horizontally homogeneous water system with time-varying free surface height, such as an estuary subject to tidal cycles. Since all quantities, including horizontal velocities, temperature and salinity, are assumed to be horizontally homogeneous, we can describe this system in a semi-one-dimensional way where all variables are resolved in only one spatial dimension, but horizontal processes are taken into account when modeling turbulence. Specifically, we decompose the vertical fluid motion into a mean part which, in our case, is characterized by tidal dynamics, and a fluctuating part that describes turbulent mixing (Reynolds decomposition). This turbulent mixing can effectively be modeled as a form of diffusion.

Now, we want to perform data assimilation on the time-dependent concentrations of a tracer suspended in the water. The tracer does not interact with other tracers, is neither created nor destroyed, and does not influence the dynamics of the water, but has a constant sinking velocity and otherwise follows the motion of the water. We use the transport equation

$$\dot{c} = \underbrace{-w_s \frac{\partial c}{\partial z}}_{\text{advection (sinking)}} + \underbrace{\frac{\partial}{\partial z}\left(\nu \frac{\partial c}{\partial z}\right)}_{\text{diffusion (turbulence)}} \tag{18}$$

with tracer concentration $c(z,t)$, constant sinking velocity w_s and turbulent diffusivity $\nu(z,t)$. Note that the vertical fluid velocity due to tidal elevation is neglected; instead, the grid moves with the tidal elevation. There is a one-way dependence of the tracer dynamics on the turbulent mixing, which enables us to pre-compute ν and then use it as a forcing input to our dynamical model for c, greatly reducing computational expenses during data assimilation. We perform the turbulence computation with the General Ocean Turbulence Model (GOTM) introduced by Burchard et al. (1999), specifically with a modified version of the `estuary` test case provided with the model. For further information on this type of scenario, we refer to (Burchard and Hetland, 2010). For our actual dynamical model described by Eq. (18), we use custom differentiable operators based on the transport routines of GOTM, re-implemented in PyTorch. Our DA task consists of inferring tracer concentration over space and time, from sparse and noisy observations of the tracer, with both strong- and weak-constraint 4D-Var.

Figure 11 shows the application of strong- and weak-constraint 4D-Var to DA for tracer transport. Computation took 101 s for strong- and 158 s for weak-constraint 4D-Var on a 48-core Intel(R) Xeon(R) Platinum 8160 CPU @ 2.10GHz. Both methods reconstruct the tracer concentration time series $c(z,t)$ with high precision. Both strong- and weak-constraint 4D-Var have relatively high reconstruction error at the beginning of the time series, but not at the end. This is likely caused by the stability of the physical system: inaccuracies in the form of small-scale fluctuations in the initial state will disperse and thus become weaker due to diffusion, limiting the ability of sparse observations to constrain initial states. This inherent difficulty of

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

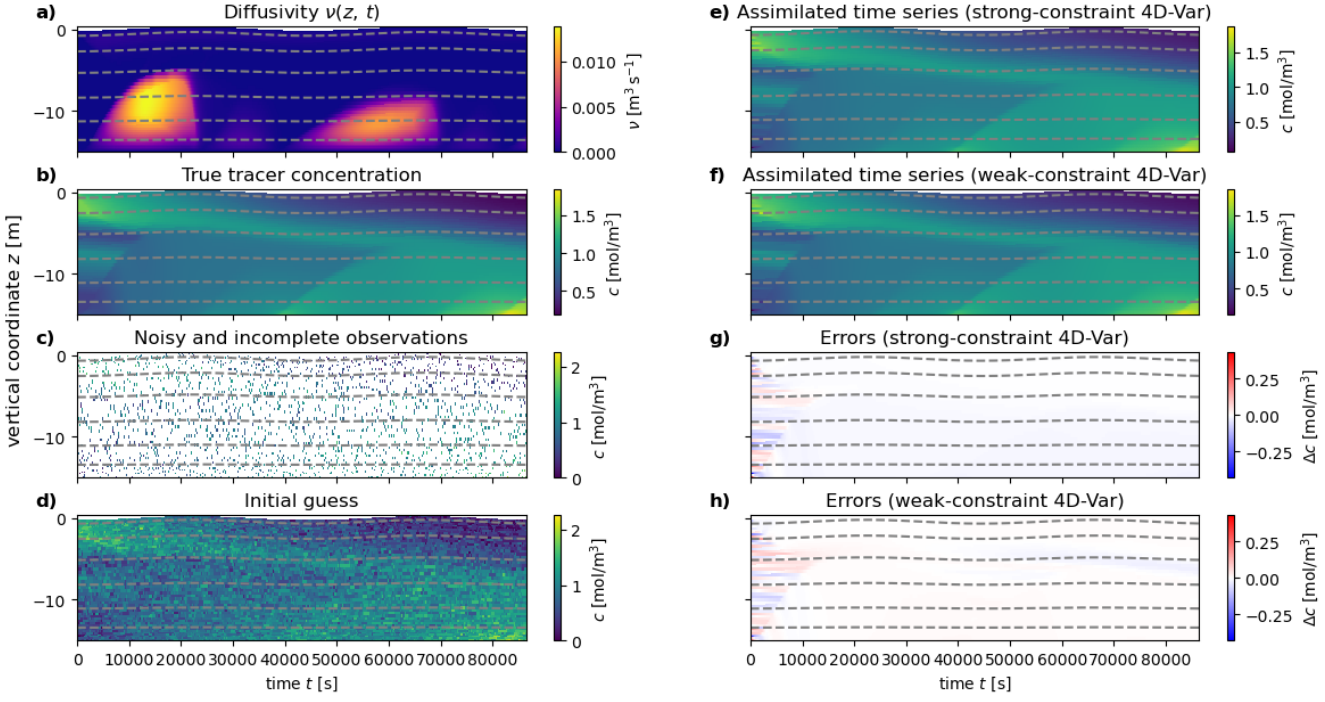

Figure 11. Data assimilation with 4D-Var on an inert tracer in a horizontally homogeneous setting with tides. The tracer concentration c follows the vertical transport equation (18). In all plots, the x-axis is time t in seconds and the y-axis is the vertical coordinate z in meters. **a)** Diffusivity $\nu(z,t)$ as a function of vertical position z and time t. **b)** The tracer concentration $c(z,t)$ simulated by GOTM (Burchard et al., 1999) acts as a known input to the assimilated system. **c)** Sparse and noisy pseudo-observations of tracer concentrations with white noise (s.d. 0.2, 10 % of values observed). **d)** Initial guess of the tracer concentrations over space and time using the next observed value. **e, f)** Assimilated trajectories using strong- (e) and weak-constraint (f) 4D-Var. Strong-constraint 4D-Var was performed using L-BFGS with a learning rate of 0.5 for 5 steps. Weak-constraint 4D-Var used a diagonal model error covariance with diagonal coefficients 10^{-6}, and leveraged L-BFGS with a learning rate of 1.0, for 100 steps. **g, h)** Errors of the assimilated trajectories for weak- (g) and strong-constraint (h) 4D-Var w.r.t. the true simulated time series.

initial state estimation of diffusive systems could be addressed by imposing a prior distribution on the initial state. On the other hand, as the initial inaccuracies disappear after a short interval, the objective of state estimation is fulfilled sufficiently well.

4.6 Assimilation with a neural emulator of Kolmogorov flow

Neural emulators offer a powerful alternative to classical numerical solvers in modeling physical systems and their improved cost/accuracy tradeoff is demonstrated on large-scale geophysical applications (Kochkov et al., 2023; Lam et al., 2023; Watt-Meyer et al., 2024; Dheeshjith et al., 2025). Unlike classical solvers, neural emulators are implemented in automatic differentiation frameworks that natively provide input/output gradients. This makes them more favorable for variational data assimilation than existing complex solvers that lack autodiff capabilities, which would require manual implementation of adjoint routines.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

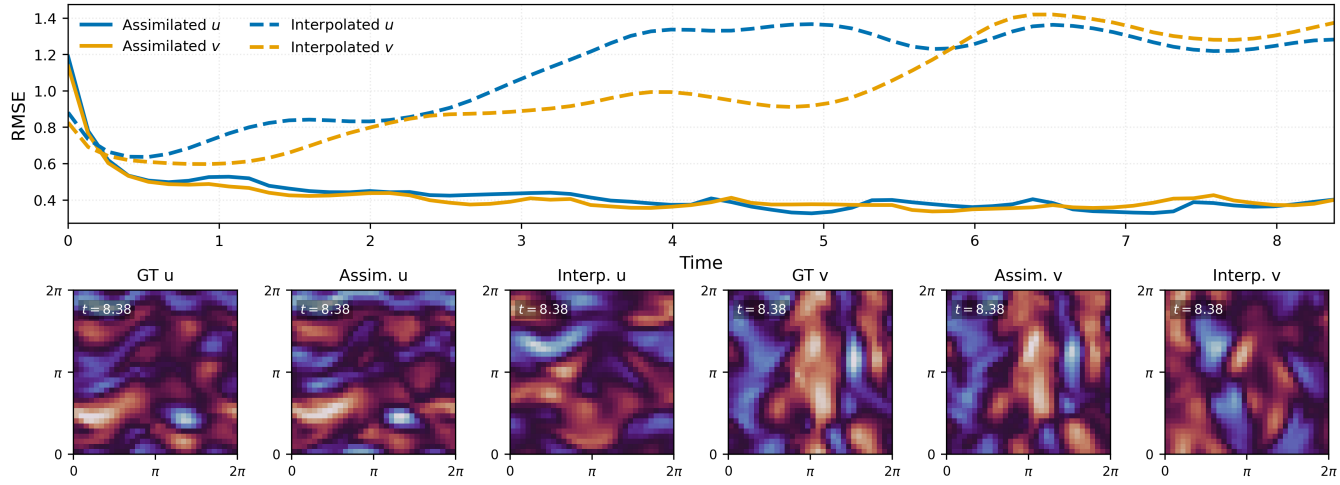

Figure 12. Results of sliding-window strong-constraint 4D-Var on the Kolmogorov flow emulator task. Assimilated states display a lower RMSE and stay closer to the groundtruth trajectory throughout the simulation. Visual inspection of the predicted fields at the end of the assimilation window also demonstrates the improvement.

By training on system states generated by existing solvers, neural emulators circumvent this need entirely (Nonnenmacher and Greenberg, 2021; Hatfield et al., 2021; Frezat et al., 2024), and integrate naturally into ADDA's framework, which can assimilate any system for which differentiable time stepping is available in PyTorch. To demonstrate this, we train a Fourier neural operator (Li et al., 2021) following the training procedure described in Bekar et al. (2025) to emulate Kolmogorov flow (Boffetta and Ecke, 2012), and use it as the forward model in strong-constraint 4D-Var data assimilation. Kolmogorov flow is governed by the 2D incompressible Navier-Stokes equations with periodic boundary conditions and a forcing consisting of a sinusoidal and a damping term,

$$\partial_t \mathbf{u} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \frac{1}{\mathrm{Re}} \Delta \mathbf{u} + \frac{1}{\rho} \nabla p = \mathbf{f} \qquad \text{in } \Omega \tag{19}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \tag{20}$$

$$\mathbf{f} = \sin(4y)\hat{\mathbf{e}}_1 - 0.1\mathbf{u} \tag{21}$$

where $\Omega = [0, 2\pi]^2$, $\mathbf{u} = \{u, v\}$ is the velocity field, Re is the Reynolds number, ρ is the density, p is the pressure field and $\mathbf{f}$ is the forcing. We set $(\rho, \mathrm{Re}, \nu) = (1, 10^3, 10^{-3})$. This system generates statistically stationary flow fields. The direct numerical solver resolves the smallest scale necessary ($\Delta t = 2.1 \times 10^{-4}$, $\Delta x = 3 \times 10^{-3}$) to simulate the turbulent physics with high fidelity (Kochkov et al., 2021) while the emulator is trained only on the spatiotemporally coarsened system states $\mathbf{u}$. The training data are temporally coarsened by a factor of approximately 600 and spatially coarsened by a factor of 64 along both spatial axes, and do not include forcing inputs.

For the assimilation task, we observe 5 % of state variables for 64 time steps, with Gaussian observation noise ($\sigma = 1$). We apply strong-constraint 4D-Var with non-overlapping 8-step sub-windows (using ADDA's named input `window_shift`). We

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

initialize each element of the system state to its first observed value, and use a spherical Gaussian background prior ($\sigma = 2$). We use the same optimizer and scheduler settings as in Sect. 4.4 but, for each sub-window, perform 100 optimization steps with 10 iterations per cycle. Figure 12 compares the assimilated trajectory to the initialization. DA produces system states that closely follow the groundtruth velocity field, while the initialization exhibits steadily increasing errors and is no longer correlated with the groundtruth fields at the end of the observation domain. This example demonstrates the ease of incorporating neural emulators into ADDA methods, as well as the effectiveness of neural emulators in enabling variational DA for simulations whose gradients are unavailable.

4.7 Latent data assimilation on the Kuramoto-Sivashinsky system

In high-dimensional geophysical problems, such as numerical weather prediction, construction of the background error covariance matrix may be computationally prohibitive (Xiao et al., 2024b; Fan et al., 2026). While common approaches like localization might mitigate this, likelihood calculations remain costly (see e.g. Carrassi et al. (2018) appendix A). Latent data assimilation (LDA) is a promising approach to addressing this challenge, in which a latent representation of the state space is learned and DA is carried out to identify the latent representations given observations. ADDA's routines for strong-constraint 4D-Var can be used for LDA simply by providing an optional named input `c_proj` of type `callable`. This input receives the decoder of a trained (variational) autoencoder, which maps latent states $\mathbf{z}$ onto physical states $\mathbf{x}_t$.

To demonstrate ADDA's straightforward incorporation of LDA, we implement the approach from (Fan et al., 2026), using the 1D Kuramoto-Sivashinsky (KS) dynamical system (Weinan and Liu, 2002) as a test case. This system evolves with the following PDE, boundary and initial conditions, resulting in chaotic dynamics:

$$\frac{\partial u}{\partial t} = -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} - \frac{1}{2}\left(\frac{\partial u}{\partial x}\right)^2, \qquad u(0,t) = u(L,t), \qquad u(x,0) = \sin(16\pi(x-\theta)/L), \qquad \theta \sim U[0,L]. \tag{22}$$

Initial conditions are sine waves with wavenumber 8 and uniform random phases. We solve the KS equation using a pseudo-spectral solver with an ETDRK2 (Cox and Matthews, 2002) time integrator implemented in PyTorch following the approach of Koehler et al. (2024). The domain length is set to $L = 100$ with spatial resolution $\Delta x \approx 0.78$ and time step $\Delta t = 0.1$. As we aim to carry out DA for chaotic trajectories from the stationary distribution on the KS system's attractor, we generate groundtruth trajectories by running the solver for 1600 steps and discarding the first 800. We implement the strong-constraint 4D-Var LDA formulation from Fan et al. (2026),

$$\min_{\mathbf{z}\in\mathbb{R}^s} J(\mathbf{z}) = \frac{1}{2}||\mathbf{z} - \mathbf{z}_B||^2_{\mathbf{\Sigma}^{-1}_{B_\mathbf{z}}} + \frac{1}{2}\sum_{t=0}^{T} ||\mathbf{y}_t - \mathcal{H}_t(\mathcal{M}^{(t)}(\mathcal{D}(\mathbf{z})))||^2_{\mathbf{\Sigma}^{-1}_{\eta_t}}, \tag{23}$$

where $\mathbf{z}$ is the initial latent state, $\mathbf{\Sigma}_{B_\mathbf{z}}$ is the background error covariance matrix in the latent space and $\mathcal{D}$ is the decoder mapping latent states to physical states. As argued in Fan et al. (2026), constructing the background error covariance matrix in latent space is more tractable than in data space, and they empirically show that it becomes approximately diagonal. They specify this matrix manually in the latent space of a regular auto-encoder, but we take a different approach and train a β-VAE instead (Xiao et al., 2024b). Since the VAE objective explicitly regularizes the latent variable toward a normal distribution (Higgins et al.,

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

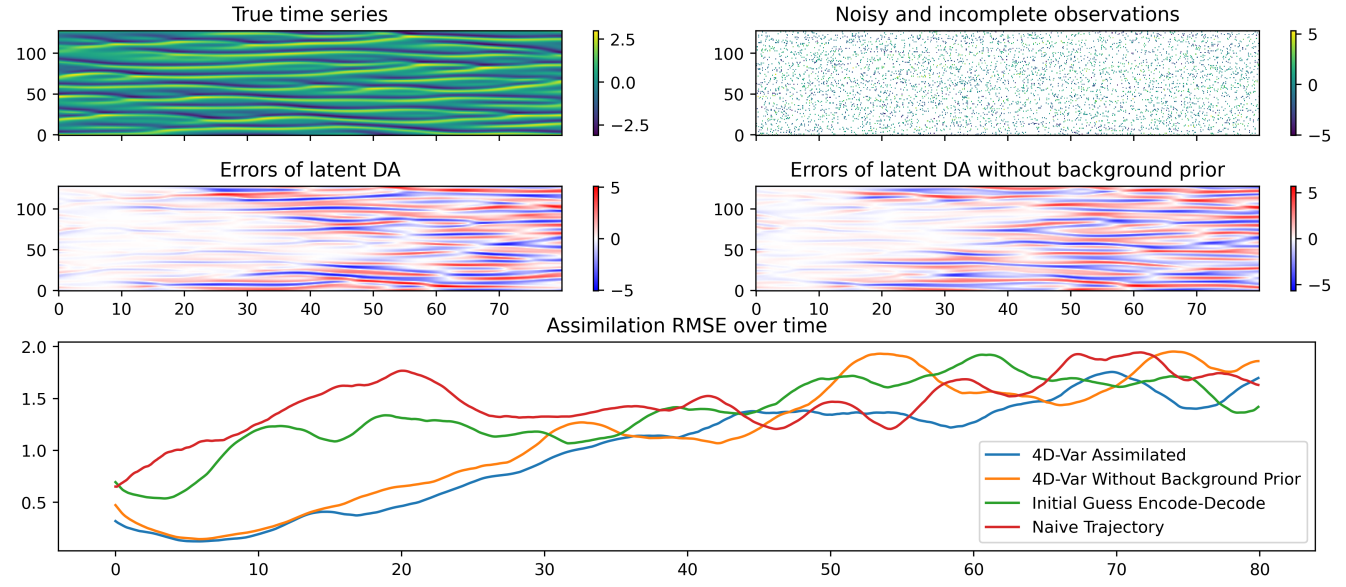

Figure 13. Assimilated results with strong-constraint 4D-Var with latent data assimilation on the Kuramoto-Sivashinsky system. Using a background prior in the latent space results in more accurate assimilation.

2017) and our training data are drawn from the stationary regime of the KS system, we define the background prior on the latent states also as a normal distribution, i.e., $\mathbf{\Sigma}_{B_\mathbf{z}}$ an identity matrix and $\mathbf{z}_B$ a vector of zeros.

We assimilate over a window of 100 time steps, i.e. 10 time units, observing $5\,\%$ of the state variables, with Gaussian white noise of s.d. 1.0. The latent state $\mathbf{z}$ is initialized by encoding a state $\mathbf{x}_0$ obtained by nearest-neighbor interpolation over time. Optimization is performed using L-BFGS for 20 steps with a learning rate of 0.1.

We compare four configurations: LDA with background prior, LDA without background prior, rolling out the encode-decode initialization of $\mathbf{x}_0$ without assimilation, and rolling out the naive initialization without assimilation. Note that even without an explicit prior term, the trained decoder provides an implicit (or "soft") prior: because it has learned to represent the training data distribution, optimizing through the decoder naturally biases the solution toward the learned data manifold rather than arbitrary states. As shown in Fig. 13, LDA with background prior performs the best, while the second best model is the LDA without background prior, followed by the encode-decode baseline. We believe that encoding and decoding act as a projection of the noisy input onto the manifold of the stationary dynamics. The nearest-neighbor-in-time initialization performs the worst. All the trajectories diverge after 400 steps due to the chaotic nature of the KS system.

4.8 Strong-constraint 4D-Var on the 2D Kuramoto-Sivashinsky system with a JAX-Torch bridge

JAX (Bradbury et al., 2018) has a growing ecosystem of differentiable scientific software (Kochkov et al., 2021; Bezgin et al., 2023; Holl and Thuerey, 2024; Koldunov et al., 2026), while PyTorch offers a more established deep learning ecosystem. We

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

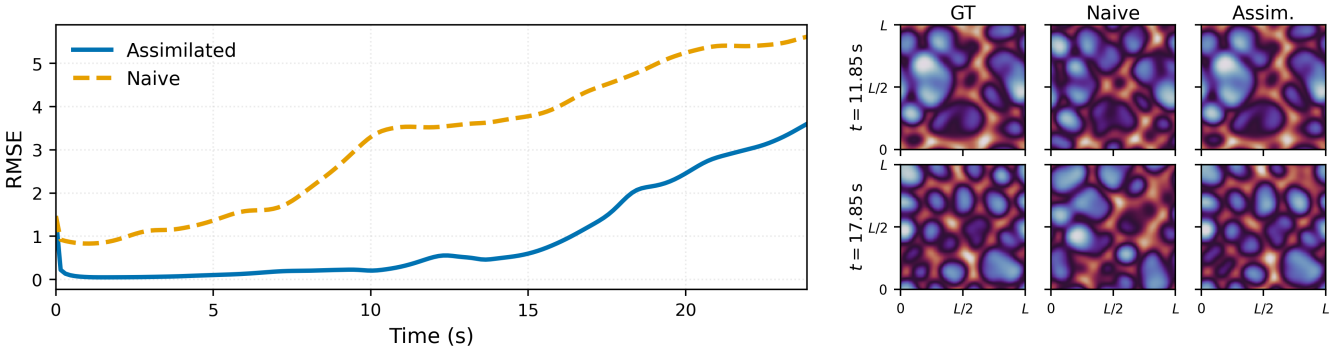

Figure 14. Assimilated results with strong-constraint 4D-Var on the 2D KS system. The RMSE of the predictions over time (left) and snapshots of the field variable at $t = 11.85s$ and $t = 17.85s$ (right). The predictions from the assimilated initial condition remain lower throughout the prediction window.

implement ADDA in PyTorch to leverage this maturity and its wider support for deep learning-based data assimilation methods. Since both frameworks support automatic differentiation, we bridge a JAX simulation code with ADDA for variational data assimilation.

The bridge is implemented as a custom `torch.autograd.Function`, where `jax.vjp` passes gradients from JAX to PyTorch, while dlpack performs zero-copy conversion of arrays between the two frameworks. This introduces computational overhead relative to a native PyTorch implementation, but avoids redundant re-implementation of scientific software that already supports automatic differentiation. Coupling frameworks in this way might be a practical alternative to re-implementation; it extends ADDA to the growing body of JAX-based differentiable solvers and increases its practicality as research software. To demonstrate this, we use Exponax (Koehler et al., 2024), a differentiable spectral solver library built in JAX, with the 2D-KS equation as a benchmark:

$$\frac{\partial u}{\partial t} + \frac{1}{2}\|\nabla u\|_2^2 + \Delta u + \Delta^2 u = 0, \qquad u(0,y,t) = u(L_x,y,t), \qquad u(x,0,t) = u(x,L_y,t). \tag{24}$$

At this domain size, the system exhibits chaotic dynamics. The initial condition is sampled as $u(x,y,0) \sim \mathcal{N}(0,1)$, with domain length $L_x = L_y = 30$, spatial resolution $\Delta_x = \Delta_y = 0.3$, and time step $\Delta_t = 0.15$, integrated using the ETDRK2 solver. The initial 1000 transient time steps are discarded, and the following 160 time steps form the dataset. We observe $25\,\%$ of state variables for 16 time steps with Gaussian observation noise ($\sigma = 1$) and apply strong-constraint 4D-Var, initializing each state element to its first observed value. We use the same optimizer and scheduler settings as in Sect. 4.7, optimizing for 5 steps. Figure 14 shows the RMSE of the assimilated versus initialized trajectories across 160 time steps and contours at two time points; the assimilated states exhibit lower prediction error throughout the prediction window and predictions from the assimilated initial condition remain close to the ground truth.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

5 Discussion

We consider this work to mark a shift in the relative technical difficulties of implementing variational and ensemble-based data assimilation methods. Historically, 4D-Var has been considered more powerful but also more difficult to develop and maintain than the ensemble Kalman filter and smoother (see e.g. Altaf et al. (2013); Carrassi et al. (2018)) due to the necessity of deriving an adjoint model of the studied dynamical system to compute its gradients. It has long been envisioned that automatic differentiation could be a crucial tool for overcoming this difficulty (e.g. Daescu et al. (2000); Castaings et al. (2006)), yet a flexible and extensive software basis for performing data assimilation with the help of automatic differentiation was still lacking. Another important advantage of automatic differentiation for data assimilation is the use of GPUs for efficient computation, which is straightforward in autodiff frameworks such as PyTorch. We hope that the present work will contribute to a large shift of data assimilation practice towards automatic differentiation. Concretely, the main difficulty that remains unaddressed by our work is that many of the geoscientific dynamical systems on which data assimilation is performed operationally rely on complex implementations in old but computationally efficient programming languages such as Fortran, which do not enable automatic differentiation. This difficulty can be tackled with two different strategies: first, one can directly re-implement the systems in an automatic differentiation framework. This approach, known as differentiable physics, has attracted increasing attention (Shen et al., 2023) and has already achieved significant success in a range of applications (Gelbrecht et al., 2023; Sapienza et al., 2025). A popular alternative is to train an emulator, i.e., a neural network designed to approximate the system dynamics, typically at a significantly lower cost and allowing for differentiation by design (Hatfield et al., 2021; Maulik et al., 2022; Seabra et al., 2024).

The potential extensions of our work are numerous. First, it should be noted that the field of data assimilation is extremely extensive and that many popular methods, either classical (such as various ensemble Kalman filter variants and weak-constraint 4D-Var schemes) or machine/deep learning-based (e.g., analog data assimilation (Lguensat et al., 2017), 4DVarNet (Fablet et al., 2021) and score-based data assimilation (Rozet and Louppe, 2023)), are not yet implemented in ADDA. Support for these techniques could open the way to establishing an extensive benchmark of data assimilation methods. Besides data assimilation itself, the problem of joint state and parameter estimation (discussed in e.g. Bocquet and Sakov (2013); Zinchenko and Greenberg (2024)) is a natural and straightforward extension of our work, as automatic differentiation could certainly be beneficial in this type of problem as well.

6 Conclusions

We have introduced ADDA, a software package for performing data assimilation in a large variety of contexts. Our numerous examples showcase the versatility of ADDA, which can handle e.g. states with heterogeneous fields, dynamics with forcing inputs and observations irregularly sampled in time. The modular aspect of ADDA further enables easy adaptation to new use cases, including custom dynamics, observation operators and probability distributions. A core feature of ADDA is that it leverages automatic differentiation in PyTorch, while also offering compatibility with an extensive corpus of existing differentiable simulation in JAX. ADDA fulfills a longstanding need to perform variational data assimilation without going through the

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

tedious task of implementing an adjoint model for every dynamical model in addition to the implementation of the model itself. Now, one only needs to provide a PyTorch or JAX implementation of the dynamical model, and the adjoint is then obtained through automatic differentiation. In addition, recent methods involving neural networks, such as latent data assimilation and 4D-Var with a deep emulator, are straightforward with ADDA, as shown in our experiments. We expect this work to enable a more widespread usage of variational data assimilation with differentiable physics simulators as well as deep emulators, and extensive comparisons between different dynamical models and optimization methods.

Code availability. A frozen archive of all of our code, including illustrative notebooks, is publicly available at https://zenodo.org/records/22127623 (Frion et al., 2026).

Author contributions. Conceptualization: AF, DG, MN and VZ. Methodology: AF, DG, MN, VZ, AB, and PN. Software: AF, DG, MN, VZ, AB, and PN. Resources: DG. Data curation: AF, AB, PN, and VZ. Writing – Original Draft: AF, AB, PN, and DG. Writing – Review and Editing: AF, AB, PN, DG, and MN. Visualization: AF, AB, and PN. Supervision: DG. Project administration: DG. Funding acquisition: DG.

Competing interests. None of the authors have any competing interests.

Acknowledgements. Our work has benefited from the support of the Helmholtz association through the HClimRep project. We also acknowledge support from two Horizon Europe research and innovation programmes: NECCTON under Grant Number 101081273 and EDITO-MODELLAB under Grant Number 101093293.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

References

Aharon, L., Lee, K., Sikka, K., Chettih, S., Hurwitz, C., Paninski, L., and Whiteway, M. R.: An uncertainty-aware framework for data-efficient multi-view animal pose estimation, https://doi.org/10.48550/arXiv.2510.09903, arXiv:2510.09903 [cs], 2025.

Ahmed, S. E., Pawar, S., and San, O.: PyDA: A Hands-On Introduction to Dynamical Data Assimilation with Python, Fluids, 5, 225, https://doi.org/10.3390/fluids5040225, 2020.

Aksoy, A., Zhang, F., and Nielsen-Gammon, J. W.: Ensemble-Based Simultaneous State and Parameter Estimation in a Two-Dimensional Sea-Breeze Model, Monthly Weather Review, 134, 2951–2970, https://doi.org/10.1175/MWR3224.1, 2006.

Altaf, M. U., El Gharamti, M., Heemink, A. W., and Hoteit, I.: A reduced adjoint approach to variational data assimilation, Computer Methods in Applied Mechanics and Engineering, 254, 1–13, https://doi.org/10.1016/j.cma.2012.10.003, 2013.

Amendola, M., Arcucci, R., Mottet, L., Casas, C. Q., Fan, S., Pain, C., Linden, P., and Guo, Y.-K.: Data Assimilation in the Latent Space of a Neural Network, https://doi.org/10.48550/arXiv.2012.12056, arXiv:2012.12056 [cs], 2020.

Anderson, J., Hoar, T., Raeder, K., Liu, H., Collins, N., Torn, R., and Avellano, A.: The Data Assimilation Research Testbed: A Community Facility, Bulletin of the American Meteorological Society, 90, 1283–1296, https://doi.org/10.1175/2009BAMS2618.1, 2009.

Anderson, J. L.: An Ensemble Adjustment Kalman Filter for Data Assimilation, Monthly Weather Review, 129, 2884–2903, https://doi.org/10.1175/1520-0493(2001)129<2884:AEAKFF>2.0.CO;2, 2001.

Anderson, J. L.: A Quantile-Conserving Ensemble Filter Framework. Part I: Updating an Observed Variable, Monthly Weather Review, 150, 1061–1074, https://doi.org/10.1175/MWR-D-21-0229.1, 2022.

Anderson, J. L. and Anderson, S. L.: A Monte Carlo Implementation of the Nonlinear Filtering Problem to Produce Ensemble Assimilations and Forecasts, Monthly Weather Review, 127, 2741–2758, https://doi.org/10.1175/1520-0493(1999)127<2741:AMCIOT>2.0.CO;2, 1999.

Andry, G., Lewin, S., Rozet, F., Rochman, O., Mangeleer, V., Pirlet, M., Faulx, E., Grégoire, M., and Louppe, G.: Appa: Bending Weather Dynamics with Latent Diffusion Models for Global Data Assimilation, Machine Learning and the Physical Sciences Workshop (NeurIPS), https://arxiv.org/abs/2504.18720, 2025.

Balsamo, G., Rabier, F., Balmaseda, M., Bauer, P., Brown, A., Dueben, P., English, S., McNally, T., Pappenberger, F., Sandu, I., Thepaut, J.-N., and Wedi, N.: Recent progress and outlook for the ECMWF Integrated Forecasting System, Tech. Rep. EGU23-13110, Copernicus Meetings, https://doi.org/10.5194/egusphere-egu23-13110, conference Name: EGU23, 2023.

Bannister, R. N.: Elementary 4d-var, 2001.

Bannister, R. N.: A review of operational methods of variational and ensemble-variational data assimilation, Quarterly Journal of the Royal Meteorological Society, 143, 607–633, https://doi.org/10.1002/qj.2982, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.2982, 2017.

Beauchamp, M., Febvre, Q., Georgenthum, H., and Fablet, R.: 4DVarNet-SSH: end-to-end learning of variational interpolation schemes for nadir and wide-swath satellite altimetry, Geoscientific Model Development, 16, 2119–2147, https://doi.org/10.5194/gmd-16-2119-2023, 2023.

Bekar, A. C., Agarwal, S., Hüttig, C., Tosi, N., and Greenberg, D. S.: Hybrid Latent Representations for PDE Emulation, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems, https://openreview.net/forum?id=Hh8ebJYQs3, 2025.

Bergou, E. H., Gratton, S., and Mandel, J.: On the convergence of a non-linear ensemble Kalman smoother, Applied Numerical Mathematics, 137, 151–168, https://doi.org/10.1016/j.apnum.2018.11.008, 2019.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Bezgin, D. A., Buhendwa, A. B., and Adams, N. A.: JAX-Fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows, Computer Physics Communications, 282, 108 527, https://doi.org/10.1016/j.cpc.2022.108527, 2023.

Bocquet, M. and Sakov, P.: Joint state and parameter estimation with an iterative ensemble Kalman smoother, Nonlinear Processes in Geophysics, 20, 803–818, https://doi.org/10.5194/npg-20-803-2013, 2013.

Bocquet, M. and Sakov, P.: An iterative ensemble Kalman smoother, Quarterly Journal of the Royal Meteorological Society, 140, 1521–1535, https://doi.org/10.1002/qj.2236, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.2236, 2014.

Bocquet, M., Farchi, A., and Malartic, Q.: Online learning of both state and dynamics using ensemble Kalman filters, Foundations of Data Science, 3, 305–330, https://doi.org/10.3934/fods.2020015, arXiv:2006.03859 [stat], 2021.

Boffetta, G. and Ecke, R. E.: Two-Dimensional Turbulence, Annual Review of Fluid Mechanics, 44, 427–451, https://doi.org/10.1146/annurev-fluid-120710-101240, 2012.

Bou, A., Bettini, M., Dittert, S., Kumar, V., Sodhani, S., Yang, X., De Fabritiis, G., and Moens, V.: TorchRL: A data-driven decision-making library for PyTorch, International Conference on Learning Representations, 2024, 1778–1811, https://proceedings.iclr.cc/paper_files/paper/2024/hash/07bc8125400bf4b140c332010756bd9b-Abstract-Conference.html, 2024.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Katariya, Y., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q.: JAX: composable transformations of Python+NumPy programs, http://github.com/jax-ml/jax, 2018.

Brajard, J., Carrassi, A., Bocquet, M., and Bertino, L.: Combining data assimilation and machine learning to infer unresolved scale parametrization, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 379, 20200 086, https://doi.org/10.1098/rsta.2020.0086, 2021.

Burchard, H. and Hetland, R. D.: Quantifying the Contributions of Tidal Straining and Gravitational Circulation to Residual Circulation in Periodically Stratified Tidal Estuaries, Journal of Physical Oceanography, 40, 1243–1262, https://doi.org/10.1175/2010JPO4270.1, 2010.

Burchard, H., Bolding, K., and Villarreal, M. R.: GOTM, a general ocean turbulence model: Theory, implementation and test cases, Space Applications Institute, 1999.

Burgers, G., Leeuwen, P. J. v., and Evensen, G.: Analysis Scheme in the Ensemble Kalman Filter, Monthly Weather Review, 126, 1719–1724, https://doi.org/10.1175/1520-0493(1998)126<1719:ASITEK>2.0.CO;2, 1998.

Carrassi, A., Bocquet, M., Bertino, L., and Evensen, G.: Data assimilation in the geosciences: An overview of methods, issues, and perspectives, WIREs Climate Change, 9, e535, https://doi.org/10.1002/wcc.535, _eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wcc.535, 2018.

Carrió, D. S., Mazzarella, V., and Ferretti, R.: High-resolution data assimilation for two maritime extreme weather events: a comparison between 3D-Var and EnKF, Natural Hazards and Earth System Sciences, 25, 2999–3026, https://doi.org/10.5194/nhess-25-2999-2025, 2025.

Castaings, W., Dartus, D., Honnorat, M., Dimet, F.-X. L., Loukili, Y., and Monnier, J.: Automatic Differentiation: A Tool for Variational Data Assimilation and Adjoint Sensitivity Analysis for Flood Modeling, in: Automatic Differentiation: Applications, Theory, and Implementations, edited by Bücker, M., Corliss, G., Naumann, U., Hovland, P., and Norris, B., pp. 249–262, Springer, Berlin, Heidelberg, ISBN 978-3-540-28438-3, https://doi.org/10.1007/3-540-28438-9_22, 2006.

Chen, S., Jia, Y., Qu, Q., Sun, H., and Fessler, J. A.: FlowDAS: A Stochastic Interpolant-based Framework for Data Assimilation, https://openreview.net/forum?id=1nWqhiulqD, 2025.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Cheng, S., Quilodrán-Casas, C., Ouala, S., Farchi, A., Liu, C., Tandeo, P., Fablet, R., Lucor, D., Iooss, B., Brajard, J., Xiao, D., Janjic, T., Ding, W., Guo, Y., Carrassi, A., Bocquet, M., and Arcucci, R.: Machine Learning With Data Assimilation and Uncertainty Quantification for Dynamical Systems: A Review, IEEE/CAA Journal of Automatica Sinica, 10, 1361–1387, https://doi.org/10.1109/JAS.2023.123537, 2023.

Cheng, S., Zhuang, Y., Kahouadji, L., Liu, C., Chen, J., Matar, O. K., and Arcucci, R.: Multi-domain encoder–decoder neural networks for latent data assimilation in dynamical systems, Computer Methods in Applied Mechanics and Engineering, 430, 117 201, https://doi.org/10.1016/j.cma.2024.117201, 2024.

Cheng, S., Min, J., Liu, C., and Arcucci, R.: TorchDA: A Python package for performing data assimilation with deep learning forward and transformation functions, Computer Physics Communications, 306, 109 359, https://doi.org/10.1016/j.cpc.2024.109359, 2025.

Cho, K. H., Pachepsky, Y., Ligaray, M., Kwon, Y., and Kim, K. H.: Data assimilation in surface water quality modeling: A review, Water Research, 186, 116 307, https://doi.org/10.1016/j.watres.2020.116307, 2020.

Cosme, E., Verron, J., Brasseur, P., Blum, J., and Auroux, D.: Smoothing Problems in a Bayesian Framework and Their Linear Gaussian Solutions, Monthly Weather Review, 140, 683–695, https://doi.org/10.1175/MWR-D-10-05025.1, 2012.

Cox, S. M. and Matthews, P. C.: Exponential time differencing for stiff systems, Journal of Computational Physics, 176, 430–455, 2002.

Daescu, D., Carmichael, G. R., and Sandu, A.: Adjoint Implementation of Rosenbrock Methods Applied to Variational Data Assimilation Problems, Journal of Computational Physics, 165, 496–510, https://doi.org/10.1006/jcph.2000.6622, 2000.

Dheeshjith, S., Subel, A., Adcroft, A., Busecke, J., Fernandez-Granda, C., Gupta, S., and Zanna, L.: Samudra: An AI Global Ocean Emulator for Climate, Geophysical Research Letters, 52, e2024GL114 318, https://doi.org/10.1029/2024GL114318, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2024GL114318, 2025.

Dorffer, C., Jourdin, F., Nguyen, T. T. N., Devillers, R., Mouillot, D., and Fablet, R.: Observation-Only Deep Learning for Gappy Satellite-Derived Ocean Color Data Using 4DVarNet, IEEE Transactions on Geoscience and Remote Sensing, 63, 1–12, https://doi.org/10.1109/TGRS.2025.3624465, 2025.

Dowd, M., Jones, E., and Parslow, J.: A statistical overview and perspectives on data assimilation for marine biogeochemical models, Environmetrics, 25, 203–213, https://doi.org/10.1002/env.2264, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/env.2264, 2014.

Drillet, Y., Martin, M., Miyazawa, Y., Bell, M., Chassignet, E., and Ciliberti, S.: Core services: an introduction to global ocean forecasting, State of the Planet Discussions, 2024, 1–14, 2024.

Durand, C., Finn, T. S., Farchi, A., Bocquet, M., Brajard, J., and Bertino, L.: Four-dimensional variational data assimilation with a sea-ice thickness emulator, The Cryosphere, 19, 5613–5637, https://doi.org/10.5194/tc-19-5613-2025, 2025.

Efron, B.: Tweedie's Formula and Selection Bias, Journal of the American Statistical Association, 106, 1602–1614, https://doi.org/10.1198/jasa.2011.tm11181, _eprint: https://doi.org/10.1198/jasa.2011.tm11181, 2011.

Evensen, G.: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, Journal of Geophysical Research: Oceans, 99, 10 143–10 162, https://doi.org/10.1029/94JC00572, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/94JC00572, 1994.

Evensen, G. and Leeuwen, P. J. v.: An Ensemble Kalman Smoother for Nonlinear Dynamics, Monthly Weather Review, 128, 1852–1867, https://doi.org/10.1175/1520-0493(2000)128<1852:AEKSFN>2.0.CO;2, 2000.

Evensen, G., Vossepoel, F. C., and van Leeuwen, P. J.: Weak Constraint 4DVar, in: Data Assimilation Fundamentals: A Unified Formulation of the State and Parameter Estimation Problem, edited by Evensen, G., Vossepoel, F. C., and van Leeuwen, P. J., pp. 49–61, Springer International Publishing, Cham, ISBN 978-3-030-96709-3, https://doi.org/10.1007/978-3-030-96709-3_5, 2022.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Fablet, R., Chapron, B., Drumetz, L., Mémin, E., Pannekoucke, O., and Rousseau, F.: Learning Variational Data Assimilation Models and Solvers, Journal of Advances in Modeling Earth Systems, 13, e2021MS002 572, https://doi.org/10.1029/2021MS002572, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002572, 2021.

Fablet, R., Chapron, B., Le Sommer, J., and Sévellec, F.: Inversion of Sea Surface Currents From Satellite-Derived SST-SSH Synergies With 4DVarNets, Journal of Advances in Modeling Earth Systems, 16, e2023MS003 609, https://doi.org/10.1029/2023MS003609, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2023MS003609, 2024.

Fan, H., Bai, L., Fei, B., Xiao, Y., Chen, K., Liu, Y., Qu, Y., Ling, F., and Gentine, P.: Physically consistent global atmospheric data assimilation with machine learning in latent space, Science Advances, https://www.science.org/doi/10.1126/sciadv.aea4248, 2026.

Febvre, Q., Le Sommer, J., Ubelmann, C., and Fablet, R.: Training Neural Mapping Schemes for Satellite Altimetry With Simulation Data, Journal of Advances in Modeling Earth Systems, 16, e2023MS003 959, https://doi.org/10.1029/2023MS003959, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2023MS003959, 2024.

Frerix, T., Kochkov, D., Smith, J., Cremers, D., Brenner, M., and Hoyer, S.: Variational Data Assimilation with a Learned Inverse Observation Operator, in: Proceedings of the 38th International Conference on Machine Learning, pp. 3449–3458, PMLR, ISSN 2640-3498, https://proceedings.mlr.press/v139/frerix21a.html, 2021.

Frezat, H., Fablet, R., Balarac, G., and Sommer, J. L.: Gradient-free online learning of subgrid-scale dynamics with neural emulators, https://doi.org/10.48550/arXiv.2310.19385, arXiv:2310.19385 [physics], 2024.

Frion, A. and Greenberg, D. S.: Uncertainty-aware data assimilation through variational inference, https://doi.org/10.48550/arXiv.2510.17268, arXiv:2510.17268 [cs], 2026.

Frion, A., Drumetz, L., Dalla Mura, M., Tochon, G., and Bey, A. A. E.: Neural Koopman Prior for Data Assimilation, IEEE Transactions on Signal Processing, 72, 4191–4206, https://doi.org/10.1109/TSP.2024.3416828, 2024.

Frion, A., Drumetz, L., Mura, M. D., Tochon, G., and Bey, A. A. E.: Augmented Invertible Koopman Autoencoder for long-term time series forecasting, Transactions on Machine Learning Research, https://openreview.net/forum?id=o6ukhJLzMQ, 2025.

Frion, A., Nguyen-Thanh, V. M., Bekar, A. C., Nimtz, P. R., Zinchenko, V., and Greenberg, D. S.: ADDA: a Modular Framework for Representing, Simulating and Assimilating Dynamics with End-to-end Differentiability, https://doi.org/10.5281/zenodo.22127623, 2026.

Gelbrecht, M., White, A., Bathiany, S., and Boers, N.: Differentiable programming for Earth system modeling, Geoscientific Model Development, 16, 3123–3135, https://doi.org/10.5194/gmd-16-3123-2023, 2023.

Ghorbani, A., Ghorbani, V., Nazari-Heris, M., and Asadi, S.: Data Assimilation for Agent-Based Models, Mathematics, 11, 4296, https://doi.org/10.3390/math11204296, 2023.

Grande, D., Buizza, R., and Storto, A.: Machine learning in ocean data assimilation: Advances, gaps and the road to operations, Ocean Modelling, 200, 102 678, https://doi.org/10.1016/j.ocemod.2026.102678, 2026.

Greybush, S. J., Kalnay, E., Miyoshi, T., Ide, K., and Hunt, B. R.: Balance and Ensemble Kalman Filter Localization Techniques, Monthly Weather Review, 139, 511–522, https://doi.org/10.1175/2010MWR3328.1, 2011.

Gustafsson, N., Janjić, T., Schraff, C., Leuenberger, D., Weissmann, M., Reich, H., Brousseau, P., Montmerle, T., Wattrelot, E., Bučánek, A., Mile, M., Hamdi, R., Lindskog, M., Barkmeijer, J., Dahlbom, M., Macpherson, B., Ballard, S., Inverarity, G., Carley, J., Alexander, C., Dowell, D., Liu, S., Ikuta, Y., and Fujita, T.: Survey of data assimilation methods for convective-scale numerical weather prediction at operational centres, Quarterly Journal of the Royal Meteorological Society, 144, 1218–1256, https://doi.org/10.1002/qj.3179, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3179, 2018.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Hatfield, S., Chantry, M., Dueben, P., Lopez, P., Geer, A., and Palmer, T.: Building Tangent-Linear and Adjoint Models for Data Assimilation With Neural Networks, Journal of Advances in Modeling Earth Systems, 13, e2021MS002 521, https://doi.org/10.1029/2021MS002521, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002521, 2021.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., De Chiara, G., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J.-N.: The ERA5 global reanalysis, Quarterly Journal of the Royal Meteorological Society, 146, 1999–2049, https://doi.org/10.1002/qj.3803, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3803, 2020.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A.: β-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK, 2017.

Hogg, A. M. C., Killworth, P. D., Blundell, J. R., and Dewar, W. K.: Mechanisms of Decadal Variability of the Wind-Driven Ocean Circulation, Journal of Physical Oceanography, 35, 512–531, https://doi.org/10.1175/JPO2687.1, 2005.

Holl, P. and Thuerey, N.: Phiflow: Differentiable Simulations for PyTorch, TensorFlow and Jax, in: Forty-first international conference on machine learning, 2024.

Houtekamer, P. L. and Mitchell, H. L.: Data Assimilation Using an Ensemble Kalman Filter Technique, Monthly Weather Review, 126, 796–811, https://doi.org/10.1175/1520-0493(1998)126<0796:DAUAEK>2.0.CO;2, 1998.

Huang, L., Gianinazzi, L., Yu, Y., Dueben, P. D., and Hoefler, T.: DiffDA: a Diffusion Model for Weather-scale Data Assimilation, https://doi.org/10.48550/arXiv.2401.05932, arXiv:2401.05932 [cs], 2024.

Jin, X., Kumar, L., Li, Z., Feng, H., Xu, X., Yang, G., and Wang, J.: A review of data assimilation of remote sensing and crop models, European Journal of Agronomy, 92, 141–152, https://doi.org/10.1016/j.eja.2017.11.002, 2018.

Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems, Journal of Basic Engineering, 82, 35–45, https://doi.org/10.1115/1.3662552, 1960.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S.: Machine learning–accelerated computational fluid dynamics, Proceedings of the National Academy of Sciences, 118, https://doi.org/10.1073/pnas.2101784118, _eprint: https://www.pnas.org/content/118/21/e2101784118.full.pdf, 2021.

Kochkov, D., Yuval, J., Langmore, I., Norgaard, P., Smith, J., Mooers, G., Lottes, J., Rasp, S., Düben, P., Klöwer, M., Hatfield, S., Battaglia, P., Sanchez-Gonzalez, A., Willson, M., Brenner, M. P., and Hoyer, S.: Neural General Circulation Models, http://arxiv.org/abs/2311.07222, 0 citations (Semantic Scholar/arXiv) [2023-12-01] arXiv:2311.07222 [physics], 2023.

Koehler, F. and Thuerey, N.: Neural Emulator Superiority: When Machine Learning for PDEs Surpasses its Training Data, https://doi.org/10.48550/arXiv.2510.23111, arXiv:2510.23111 [cs], 2026.

Koehler, F., Niedermayr, S., Westermann, R., and Thuerey, N.: APEBench: A Benchmark for Autoregressive Neural Emulators of PDEs, https://doi.org/10.48550/arXiv.2411.00180, 0 citations (Semantic Scholar/arXiv) [2024-11-16] arXiv:2411.00180, 2024.

Koldunov, N. V., Danilov, S., Cheedela, S., Sidorenko, D., Beyer, S., Scholz, P., Kuznetsov, I., Streffing, J., Koldunov, A., Pantiukhin, D., and others: FESOM2-JAX v1. 0: a differentiable shadow of the ocean-sea-ice model FESOM2, cast onto GPUs, arXiv preprint arXiv:2608.01546, 2026.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Laloyaux, P., Bonavita, M., Chrust, M., and Gürol, S.: Exploring the potential and limitations of weak-constraint 4D-Var, Quarterly Journal of the Royal Meteorological Society, 146, 4067–4082, https://doi.org/10.1002/qj.3891, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3891, 2020.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., Merose, A., Hoyer, S., Holland, G., Vinyals, O., Stott, J., Pritzel, A., Mohamed, S., and Battaglia, P.: Learning skillful medium-range global weather forecasting, Science, 382, 1416–1421, https://doi.org/10.1126/science.adi2336, 2023.

Lguensat, R., Tandeo, P., Ailliot, P., Pulido, M., and Fablet, R.: The Analog Data Assimilation, Monthly Weather Review, 145, 4093–4107, https://doi.org/10.1175/MWR-D-16-0441.1, 2017.

Li, Y., Han, W., Li, H., Duan, W., Chen, L., Zhong, X., Wang, J., Liu, Y., and Sun, X.: FuXi-En4DVar: An Assimilation System Based on Machine Learning Weather Forecasting Model Ensuring Physical Constraints, Geophysical Research Letters, 51, e2024GL111 136, https://doi.org/10.1029/2024GL111136, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2024GL111136, 2024.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A.: Fourier Neural Operator for Parametric Partial Differential Equations, https://doi.org/10.48550/arXiv.2010.08895, arXiv:2010.08895 [cs], 2021.

Liu, C., Xiao, Q., and Wang, B.: An Ensemble-Based Four-Dimensional Variational Data Assimilation Scheme. Part I: Technical Formulation and Preliminary Test, Monthly Weather Review, 136, 3363–3373, https://doi.org/10.1175/2008MWR2312.1, 2008.

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

Lorenz, E. N.: Designing Chaotic Models, Journal of the Atmospheric Sciences, 62, 1574–1587, https://doi.org/10.1175/JAS3430.1, 2005.

Lorenz, E. N. and Emanuel, K. A.: Optimal Sites for Supplementary Weather Observations: Simulation with a Small Model, Journal of the Atmospheric Sciences, 55, 399–414, https://doi.org/10.1175/1520-0469(1998)055<0399:OSFSWO>2.0.CO;2, 1998.

Maulik, R., Rao, V., Wang, J., Mengaldo, G., Constantinescu, E., Lusch, B., Balaprakash, P., Foster, I., and Kotamarthi, R.: Efficient high-dimensional variational data assimilation with machine-learned reduced-order models, Geoscientific Model Development, 15, 3433–3445, https://doi.org/10.5194/gmd-15-3433-2022, 2022.

Melinc, B. and Zaplotnik, Z.: 3D-Var data assimilation using a variational autoencoder, Quarterly Journal of the Royal Meteorological Society, 150, 2273–2295, https://doi.org/10.1002/qj.4708, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.4708, 2024.

Meunier, E., Ouala, S., Frezat, H., Sommer, J. L., and Fablet, R.: Towards fully differentiable neural ocean model with Veros, https://doi.org/10.48550/arXiv.2511.17427, arXiv:2511.17427 [cs], 2025.

Nadler, P., Arcucci, R., and Guo, Y.-K.: Data Assimilation for Parameter Estimation in Economic Modelling, in: 2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 649–656, https://doi.org/10.1109/SITIS.2019.00106, 2019.

Navon, I. M.: Data Assimilation for Numerical Weather Prediction: A Review, in: Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications, edited by Park, S. K. and Xu, L., pp. 21–65, Springer, Berlin, Heidelberg, ISBN 978-3-540-71056-1, https://doi.org/10.1007/978-3-540-71056-1_2, 2009.

Nerger, L.: PDAF V2.1, https://doi.org/10.5281/zenodo.7861829, 2023.

Niu, S., Luo, Y., Dietze, M. C., Keenan, T. F., Shi, Z., Li, J., and Iii, F. S. C.: The role of data assimilation in predictive ecology, Ecosphere, 5, art65, https://doi.org/10.1890/ES13-00273.1, _eprint: https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.1890/ES13-00273.1, 2014.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Nonnenmacher, M. and Greenberg, D. S.: Deep Emulators for Differentiation, Forecasting, and Parametrization in Earth Science Simulators, Journal of Advances in Modeling Earth Systems, 13, e2021MS002 554, https://doi.org/10.1029/2021MS002554, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002554, 2021.

Pandya, D., Vachharajani, B., and Srivastava, R.: A review of data assimilation techniques: Applications in engineering and agriculture, Materials Today: Proceedings, 62, 7048–7052, https://doi.org/10.1016/j.matpr.2022.01.122, 2022.

Pasmans, I., Chen, Y., Sebastian Finn, T., Bocquet, M., and Carrassi, A.: Ensemble Kalman filter in latent space using a variational autoencoder pair, Quarterly Journal of the Royal Meteorological Society, 152, e70 070, https://doi.org/10.1002/qj.70070, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.70070, 2026.

Penny, S. G., Smith, T. A., Chen, T.-C., Platt, J. A., Lin, H.-Y., Goodliff, M., and Abarbanel, H. D. I.: Integrating Recurrent Neural Networks With Data Assimilation for Scalable Data-Driven State Estimation, Journal of Advances in Modeling Earth Systems, 14, e2021MS002 843, https://doi.org/10.1029/2021MS002843, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002843, 2022.

Peyron, M., Fillion, A., Gürol, S., Marchais, V., Gratton, S., Boudier, P., and Goret, G.: Latent space data assimilation by using deep learning, Quarterly Journal of the Royal Meteorological Society, 147, 3759–3777, https://doi.org/10.1002/qj.4153, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.4153, 2021.

Qu, Y., Nathaniel, J., Li, S., and Gentine, P.: Deep Generative Data Assimilation in Multimodal Setting, pp. 449–459, https://openaccess.thecvf.com/content/CVPR2024W/EarthVision/html/Qu_Deep_Generative_Data_Assimilation_in_Multimodal_Setting_CVPRW_2024_paper.html, 2024.

Raanes, P. N., Chen, Y., and Grudzien, C.: DAPPER: Data Assimilation with Python: a Package forExperimental Research, Journal of Open Source Software, 9, 5150, https://doi.org/10.21105/joss.05150, 2024.

Rabier, F., Järvinen, H., Klinker, E., Mahfouf, J.-F., and Simmons, A.: The ECMWF operational implementation of four-dimensional variational assimilation. I: Experimental results with simplified physics, Quarterly Journal of the Royal Meteorological Society, 126, 1143–1170, https://doi.org/10.1002/qj.49712656415, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.49712656415, 2000.

Raoult, N., Douglas, N., MacBean, N., Kolassa, J., Quaife, T., Roberts, A. G., Fisher, R., Fer, I., Bacour, C., Dagon, K., Hawkins, L., Carvalhais, N., Cooper, E., Dietze, M. C., Gentine, P., Kaminski, T., Kennedy, D., Liddy, H. M., Moore, D. J. P., Peylin, P., Pinnington, E., Sanderson, B., Scholze, M., Seiler, C., Smallman, T. L., Vergopolan, N., Viskari, T., Williams, M., and Zobitz, J.: Parameter Estimation in Land Surface Models: Challenges and Opportunities With Data Assimilation and Machine Learning, Journal of Advances in Modeling Earth Systems, 17, e2024MS004 733, https://doi.org/10.1029/2024MS004733, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2024MS004733, 2025.

Rauch, H. E., Tung, F., and Striebel, C. T.: Maximum likelihood estimates of linear dynamic systems, AIAA Journal, 3, 1445–1450, https://doi.org/10.2514/3.3166, 1965.

Rayner, P. J., Michalak, A. M., and Chevallier, F.: Fundamentals of data assimilation applied to biogeochemistry, Atmospheric Chemistry and Physics, 19, 13 911–13 932, https://doi.org/10.5194/acp-19-13911-2019, 2019.

Rozet, F. and Louppe, G.: Score-based Data Assimilation, Advances in Neural Information Processing Systems, 36, 40 521–40 541, https://proceedings.neurips.cc/paper_files/paper/2023/hash/7f7fa581cc8a1970a4332920cdf87395-Abstract-Conference.html, 2023.

Rozet, F., Andry, G., Lanusse, F., and Louppe, G.: Learning Diffusion Priors from Observations by Expectation Maximization, Advances in Neural Information Processing Systems, 37, 87 647–87 682, https://doi.org/10.52202/079017-2783, 2024.

Sakov, P., Oliver, D. S., and Bertino, L.: An Iterative EnKF for Strongly Nonlinear Systems, Monthly Weather Review, 140, 1988–2004, https://doi.org/10.1175/MWR-D-11-00176.1, 2012.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Sakov, P., Haussaire, J.-M., and Bocquet, M.: An iterative ensemble Kalman filter in the presence of additive model error, Quarterly Journal of the Royal Meteorological Society, 144, 1297–1309, https://doi.org/10.1002/qj.3213, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3213, 2018.

Sapienza, F., Bolibar, J., Schäfer, F., Groenke, B., Pal, A., Boussange, V., Heimbach, P., Hooker, G., Pérez, F., Persson, P.-O., and Rackauckas, C.: Differentiable Programming for Differential Equations: A Review, https://doi.org/10.48550/arXiv.2406.09699, arXiv:2406.09699 [math], 2025.

Seabra, G. S., Mücke, N. T., Silva, V. L. S., Voskov, D., and Vossepoel, F. C.: AI enhanced data assimilation and uncertainty quantification applied to Geological Carbon Storage, International Journal of Greenhouse Gas Control, 136, 104 190, https://doi.org/10.1016/j.ijggc.2024.104190, 2024.

Shen, C., Appling, A. P., Gentine, P., Bandai, T., Gupta, H., Tartakovsky, A., Baity-Jesi, M., Fenicia, F., Kifer, D., Li, L., Liu, X., Ren, W., Zheng, Y., Harman, C. J., Clark, M., Farthing, M., Feng, D., Kumar, P., Aboelyazeed, D., Rahmani, F., Song, Y., Beck, H. E., Bindas, T., Dwivedi, D., Fang, K., Höge, M., Rackauckas, C., Mohanty, B., Roy, T., Xu, C., and Lawson, K.: Differentiable modelling to unify machine learning and physical models for geosciences, Nature Reviews Earth & Environment, 4, 552–567, https://doi.org/10.1038/s43017-023-00450-9, 2023.

Shoji, Y., Arisaka, S., Ono, E., and Mihara, K.: Poster Abstract: Data Assimilation for HVAC Simulations in Koopman-Invariant Subspace using Kalman Filter, in: Proceedings of the 12th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys '25, pp. 306–307, Association for Computing Machinery, New York, NY, USA, ISBN 979-8-4007-1945-5, https://doi.org/10.1145/3736425.3772109, 2025.

Shysheya, A., Diaconu, C., Bergamin, F., Perdikaris, P., Hernández-Lobato, J. M., Turner, R. E., and Mathieu, E.: On conditional diffusion models for PDE simulations, Advances in Neural Information Processing Systems, 37, 23 246–23 300, https://doi.org/10.52202/079017-0732, 2024.

Singh, A., Singh, A., Imbiriba, T., Erdogmus, D., and Borsoi, R.: KODA: A Data-Driven Recursive Model for Time Series Forecasting and Data Assimilation using Koopman Operators, https://doi.org/10.48550/arXiv.2409.19518, arXiv:2409.19518 [cs], 2024.

Solvik, K., Penny, S. G., and Hoyer, S.: 4D-Var Using Hessian Approximation and Backpropagation Applied to Automatically Differentiable Numerical and Machine Learning Models, Journal of Advances in Modeling Earth Systems, 17, e2024MS004 608, https://doi.org/10.1029/2024MS004608, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2024MS004608, 2025.

Sparrow, C.: The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors, vol. 41 of *Applied Mathematical Sciences*, Springer, New York, NY, ISBN 978-0-387-90775-8 978-1-4612-5767-7, https://doi.org/10.1007/978-1-4612-5767-7, 1982.

Stammer, D., Balmaseda, M., Heimbach, P., Köhl, A., and Weaver, A.: Ocean Data Assimilation in Support of Climate Applications: Status and Perspectives, Annual Review of Marine Science, 8, 491–518, https://doi.org/10.1146/annurev-marine-122414-034113, 2016.

Sun, J.-A., Fan, H., Gong, J., Fei, B., Chen, K., Ling, F., Zhang, W., Xu, W., Yan, L., Gentine, P., and Bai, L.: Align-DA: Align Score-based Atmospheric Data Assimilation with Multiple Preferences, https://doi.org/10.48550/arXiv.2505.22008, arXiv:2505.22008 [physics], 2025.

Sun, L., Seidou, O., Nistor, I., and Liu, K.: Review of the Kalman-type hydrological data assimilation, Hydrological Sciences Journal, 61, 2348–2366, https://doi.org/10.1080/02626667.2015.1127376, _eprint: https://doi.org/10.1080/02626667.2015.1127376, 2016.

Thiry, L., Li, L., and Mémin, E.: Modified (hyper-)viscosity for coarse-resolution ocean models, vol. 10, pp. 273–285, https://doi.org/10.1007/978-3-031-18988-3_17, arXiv:2204.13914 [physics], 2023.

Tippett, M. K., Anderson, J. L., Bishop, C. H., Hamill, T. M., and Whitaker, J. S.: Ensemble Square Root Filters, Monthly Weather Review, 131, 1485–1490, https://doi.org/10.1175/1520-0493(2003)131<1485:ESRF>2.0.CO;2, 2003.

https://doi.org/10.5194/egusphere-2026-5088
Preprint. Discussion started: 17 September 2026

Tong, X. T., Wang, Y., and Yan, L.: Latent Autoencoder Ensemble Kalman Filter for Nonlinear Data assimilation, https://doi.org/10.48550/arXiv.2603.06752, arXiv:2603.06752 [cs.LG], 2026.

Trémolet, Y.: Model-error estimation in 4D-Var, Quarterly Journal of the Royal Meteorological Society, 133, 1267–1280, https://doi.org/10.1002/qj.94, _eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.94, 2007.

Wang, W., Ren, K., Duan, B., Zhu, J., Li, X., Ni, W., Lu, J., and Yuan, T.: A Four-Dimensional Variational Constrained Neural Network-Based Data Assimilation Method, Journal of Advances in Modeling Earth Systems, 16, e2023MS003 687, https://doi.org/10.1029/2023MS003687, _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2023MS003687, 2024.

Watt-Meyer, O., Henn, B., McGibbon, J., Clark, S. K., Kwa, A., Perkins, W. A., Wu, E., Harris, L., and Bretherton, C. S.: ACE2: Accurately learning subseasonal to decadal atmospheric variability and forced responses, https://doi.org/10.48550/arXiv.2411.11268, arXiv:2411.11268, 2024.

Weinan, E. and Liu, D.: Gibbsian Dynamics and Invariant Measures for Stochastic Dissipative PDEs, Journal of Statistical Physics, 108, 1125–1156, https://doi.org/10.1023/A:1019747716056, 2002.

Wills, A. G. and Schön, T. B.: Sequential Monte Carlo: A Unified Review, Annual Review of Control, Robotics, and Autonomous Systems, 6, 159–182, https://doi.org/10.1146/annurev-control-042920-015119, 2023.

Xiao, Y., Bai, L., Xue, W., Chen, K., Han, T., and Ouyang, W.: FengWu-4DVar: Coupling the Data-driven Weather Forecasting Model with 4D Variational Assimilation, https://doi.org/10.48550/arXiv.2312.12455, arXiv:2312.12455 [physics], 2024a.

Xiao, Y., Jia, Q., Chen, K., Bai, L., and Xue, W.: VAE-Var: Variational Autoencoder-Enhanced Variational Methods for Data Assimilation in Meteorology, https://openreview.net/forum?id=utz99dx2RN, 2024b.

Xiao, Y., Fan, H., Chen, K., Cao, Y., Fei, B., Xue, W., and Bai, L.: LoRA-EnVar: Parameter-Efficient Hybrid Ensemble Variational Assimilation for Weather Forecasting, https://openreview.net/forum?id=zhMl4Smau7, 2025.

Xu, Q., Li, B., McRoberts, R. E., Li, Z., and Hou, Z.: Harnessing data assimilation and spatial autocorrelation for forest inventory, Remote Sensing of Environment, 288, 113 488, https://doi.org/10.1016/j.rse.2023.113488, 2023.

Xu, X., Sun, X., Han, W., Zhong, X., Chen, L., Gao, Z., and Li, H.: FuXi-DA: a generalized deep learning data assimilation framework for assimilating satellite observations, npj Climate and Atmospheric Science, 8, 156, https://doi.org/10.1038/s41612-025-01039-3, 2025.

Ying, Y. M.: nansencenter/NEDAS, https://doi.org/10.5281/zenodo.16585914, 2025.

Zhou, A., Hawkins, L., and Gentine, P.: Proof-of-concept: using ChatGPT to translate and modernize an Earth system model from Fortran to Python/JAX, arXiv preprint arXiv:2405.00018, 2024.

Zinchenko, V. and Greenberg, D. S.: Combined Optimization of Dynamics and Assimilation with End-to-End Learning on Sparse Observations, https://doi.org/10.48550/arXiv.2409.07137, arXiv:2409.07137 [cs], 2024.

Zupanski, D.: A General Weak Constraint Applicable to Operational 4DVAR Data Assimilation Systems, Monthly Weather Review, 125, 2274–2292, https://doi.org/10.1175/1520-0493(1997)125<2274:AGWCAT>2.0.CO;2, 1997.