https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Advancing data assimilation with the renewed Parallel Data Assimilation Framework (PDAF V3.1)

Lars Nerger[1], Yumeng Chen[2,3], Armin Corbin[4], and Johannes Keller[5,6]

[1]Alfred-Wegener-Institut, Helmholtz-Zentrum für Polar-und Meeresforschung (AWI), 27570 Bremerhaven, Germany
[2]Department of Meteorology, University of Reading, Reading RG6 6ET, UK
[3]National Centre for Earth Observation, University of Reading, Reading RG6 6ET, UK
[4]Institute for Geodesy and Geoinformation, University of Bonn, 53115 Bonn, Germany
[5]Agrosphere Institute (IBG-3), Forschungszentrum Jülich GmbH, 52425 Jülich, Germany
[6]Centre for High-Performance Scientific Computing in Terrestrial Systems: HPSC TerrSys, Geoverbund ABC/J, Leo-Brandt-Strasse, 52425 Jülich, Germany

Correspondence: Lars Nerger (lars.nerger@awi.de)

Abstract. The Parallel Data Assimilation Framework (PDAF) is a widely used open-source software for data assimilation (DA) with complex, high-dimensional Earth system models and other applications for research and operational use. PDAF is structured to provide a framework for ensemble integrations and to assimilate observations. For the DA, it provides ensemble Kalman filters and smoothers, nonlinear ensemble DA methods, and 3D variational methods. PDAF's observation module interface (PDAF-OMI) further provides a system for structured observation handling, enabling the management of large numbers of different observation types. With the recent upgrade to version 3, PDAF underwent significant code modernization and functionality enhancements to unify more than 20 years of developments since its first release. This study provides a comprehensive overview of PDAF's functionality and concepts, including the new features introduced with the major revision. These are, in particular, a new universal interface that allows for the application of any ensemble filter and smoother method, or any 3D variational method, without changes to the source code. Further, a class of ensemble Kalman filters that processes observations serially, new diagnostics for the ensemble and observations, model-independent support for incremental analysis updating (IAU), and an explicit mode for file-based offline coupling between the model and the DA were added in the major revision. PDAF can apply the same DA methods to idealized toy models as well as realistic high-dimensional models without recoding. This speeds up the development and testing of new DA methods. The implementation of a DA system with PDAF is performed utilizing a set of template files. These allow for the implementation in a very limited time, supported by extensive documentation, as is demonstrated by an example implementation with a toy model. For high-dimensional applications, PDAF allows developers to start the implementation of a new DA system with low complexity and to extend its functionality stepwise, enabling an easy start with DA. Couplings to more than 30 models have been implemented by the PDAF developers and the user community and many of them are publicly available. Further, PDAF's internal interface to DA methods provides a clear approach to add new algorithms that can leverage PDAF's framework functionality for observations, localization, and state vector handling. This allows developers of DA methods to make their methods available to the PDAF community.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

1 Introduction

Data assimilation (DA) is applied to combine observations with numerical models. This combination is computed taking the errors, as well as error correlations of both the observations and model into account. DA is commonly used to estimate model fields for initializing a model forecast, e.g., for forecasting the atmosphere or ocean (see, e.g., Valmassoi et al., 2023; Drevillon et al., 2025). Another common application is the estimation of the model state over longer periods, which yields so-called reanalysis (e.g., Hersbach et al., 2020; Baatz et al., 2021; Storto et al., 2019). Reanalyses products are usually provided operationally (for the ocean, see e.g., Cirano et al., 2025). Further applications are the estimation of model process parameters, e.g., for ocean-biogeochemical or hydrological modeling (e.g., Mamnun et al., 2025; Zhao et al., 2025), assessing model errors (Zupanski and Zupanski, 2006), and identifying observations which are relevant in improving a model state, which can help to design observation campaigns (e.g., Bishop et al., 2001). While DA is often applied in environmental sciences, it is also applied in engineering (e.g., Ben Ali et al., 2022; Hong et al., 2025; Bakhshaei et al., 2025). In general, DA can be applied in all cases where observations over time and a corresponding numerical model are available. Here, the observations do not need to be model variables as long as one can define a function, i.e., the observation operator, that computes a model equivalent to the observations from the model variables. These features of DA offer a wide range of applications.

The DA research community has introduced numerous DA methods over the past three decades. These methods rely generally on either statistical estimation, i.e., estimating the probability distribution of the model fields or parameters, or on optimization methods, which minimize the misfit between observations and model, utilizing the uncertainties of both to weight their influence. Commonly used estimation methods are ensemble-based Kalman filters (EnKFs), of which many different variants exist (see, e.g., Vetra-Carvalho et al., 2018), and sometimes related smoothers. For optimization methods, 3-dimensional variational (3D-Var, optimizing sequentially in time) and 4-dimensional variational (4D-Var, optimizing over a time window) optimization are commonly used (see, e.g., Bannister, 2017). To advance variational methods, hybrid approaches combining them with ensembles have also been developed. Nonlinear ensemble DA methods are expected to improve the DA prediction skill, compared to methods assuming Gaussian error distributions like EnKFs. The nonlinear methods are typically particle filters utilizing resampling or explicit transformations of ensemble states (see, e.g., van Leeuwen et al., 2019). More recent developments in nonlinear DA methods are based on iterative flows or transports (e.g., Pulido and van Leeuwen, 2019; Hu and van Leeuwen, 2021). Overall, there is a large number of algorithms from which one can choose to apply data assimilation, and there is, so far, no consensus in the research community on whether optimization-based or statistical DA is preferable. EnKFs appear to be particularly easy to use because the covariance matrix representing the uncertainties is generated by the ensemble, so no additional empirical operators are required. However, these methods require large compute resources due to the need to integrate an ensemble of typically between 20 and 250 model states. Furthermore, these methods require tuning DA parameters for optimal performance. For variational DA methods without ensembles, complex covariance operators have to be developed that represent uncertainties and the cross-covariance between different model variables (e.g., Weaver et al., 2016; Mirouze et al., 2016), and much of the development work is focused on this aspect. When using ensembles in variational DA, aspects of ensemble integration, localization, and tuning appear as for EnKFs, with the additional aspect of possible preconditioning

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

to ensure a good convergence rate of the solver algorithm computing the optimization. Due to such difficulties across different classes of DA methods, most users focus on a single class, and, in general, there is little transition between the two algorithmic classes.

The implementation of a pure DA algorithm like the original Ensemble Kalman filter (Evensen, 1994; Burgers et al., 1998) can usually be accomplished with a few lines of code (e.g., 12 lines in Algorithm 4 in the Appendix of Vetra-Carvalho et al., 2018). However, implementing DA with realistic models comes with a certain complexity: one needs to combine the different (typically spatially resolved) model fields that should be included in the DA process into a single one-dimensional state vector. One also needs to process the different types of observations that one intends to assimilate. Thus, one needs to read observation information, store their values, coordinates, and uncertainties, and implement the observation operator that computes the model equivalent to an observation. Since many of the DA methods utilize so-called 'localization', a distance-based tapering of covariances or weighting of observations, one also needs to implement the computation of distances between locations on the model grid and locations of observations. The state vector (or an ensemble of state vectors) and the observations, their respective coordinates, and observation operators are then used in the actual DA algorithm. For variational DA without ensembles, the covariance operators must also be developed and implemented. Apart from these operations, the actual DA algorithm has to be implemented. For high-dimensional cases, one needs to implement the DA method and the operations on the state vector and observations with parallelization, as otherwise the computations will be too slow or the computer's memory might be insufficient. Given that for complex models. e.g. simulating the ocean or atmosphere, the number of model grid points can be of $O(10^6)$ and orders of magnitude larger, and the number of assimilated observation can also be of $O(10^5)$ and beyond, an efficient workflow has to be developed for all the operations mentioned before.

To couple the DA code with the model, the general possibilities are 1) to keep the DA and model programs separate and to utilize restart files of the model to exchange information between both programs or 2) to integrate the DA code with the model so that data between the model and DA code are exchanged in memory (for example by using pointers to model field arrays, copying between model fields and DA state vectors, or using parallel communication). We refer to the use of separate programs as 'offline coupling', while the integration of DA code with the model is referred to as 'online coupling'. Offline coupling appears to be easier to implement than online coupling because one does not need to modify the model's source code. However, offline coupling still requires model-specific functionality in the DA code to read model restart files and subsequently handle model field data for use in the data assimilation algorithm. Online coupling avoids the need to write and read restart files and model restarts at the frequency of the observation intervals and is hence computationally more efficient. However, one needs to modify the model source code, but, with a good strategy, these changes can be very limited, as discussed in Nerger et al. (2005a).

The operations described before have often been implemented specifically for the selected model (e.g., Zhang et al., 2006; Moore et al., 2011; Barker et al., 2012; Eicker et al., 2014) or a specific selection of models (e.g., Buehner et al., 2025). While this allows developers to account for model specifics, they need to develop their own data-handling workflows and implement their own functionality, including parallelization. Also, such code cannot easily take up new developments from the DA community, because they need to be (re)coded in the particular system. To circumvent the requirement of implementing the

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

DA as part of a specific model, different model-agnostic open-source frameworks for DA have been developed. Widely used open-source frameworks are the DA Research Testbed (DART Anderson et al., 2009) and the Parallel DA Framework (PDAF, Nerger et al., 2005b; Nerger and Hiller, 2013). Another open-source software with a longer development history is EnKF-c (Sakov, 2014), while the Joint Effort for DA Integration (JEDI, e.g., Liu et al., 2022) is a newer development. Instead of an independent DA software, Sun et al. (2021) integrated data assimilation functionality into a model coupling software, while Friedemann and Raffin (2022) combined PDAF with a separate ensemble framework with the aim of improving the parallel performance of the ensemble integration. These systems are all implemented using a compiled programming language (Fortran, C, or C++) to obtain optimal compute performance. More recently, Python was used to implement the framework NEDAS (NERSC ensemble data assimilation system, Ying, 2025). While all these systems aim to perform data assimilation across different models and applications, they differ in their software design, targeted application types and sizes, and functionality. Nonetheless, the design of the systems is based on the experience of the DA researchers developing them.

Here, we provide an overview of PDAF with a focus on the features of the revision 3.1 (Nerger, 2025). PDAF revision 3.0 introduced a new universal interface and additional DA methods and functionality, while revision 3.1 refined some interface functions and added functionality. We will in short write PDAF3 to refer to the new features of these revisions. While aspects of PDAF have been discussed in some earlier publications, e.g. Nerger et al. (2020) discussing the use of PDAF for coupled DA, or Chen et al. (2025) in the context of the pyPDAF software that allows the use of PDAF by implementing all case-specific code in Python, the changes introduced with PDAF3 have not yet been published. Thus, this work is intended to provide a comprehensive overview of the structure and features of PDAF to be useful for researchers who might consider using PDAF for their application, or DA software developers to obtain an overview of how PDAF, as a generic framework for high-dimensional DA applications, is structured. The paper is organized as follows. Section 2 provides a general overview of the PDAF history, use, and structure of PDAF. The components of PDAF are described in Sec. 3. Section 4 discusses an application example demonstrating a code structure that follows the implementation of PDAF with complex model, while Sec. 5 concludes the study.

2 General concept and structure of PDAF

PDAF was originally developed to compare different DA methods, in particular ensemble-based Kalman filters (Nerger et al., 2005a). Such a comparison remains relevant today when developing new DA methods, as it requires applying these methods under identical conditions. PDAF was initially used in the context of oceanography, with initial applications to a finite element ocean model with unstructured grids (Nerger et al., 2006, 2007). These applications motivated PDAF to flexibly accommodate arbitrary model grids. PDAF was open-source since its initial code release v1.0 in 2004, and its concepts were described in Nerger et al. (2005b). To obtain an optimal compute performance a direct coupling between the models and the PDAF code was used, where the complexity of ocean models motivated a coupling strategy that avoids major changes in the model code (Nerger et al., 2005b). These original design concepts are still followed in PDAF3. Significant enhancements over time were the addition of further advanced ensemble-based Kalman filters, ensemble smoother, nonlinear particle filters, and hybrid

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

nonlinear-Kalman filters. Further, support for the file-based offline-coupling of model and PDAF, and structured observation handling (the PDAF Observation Module Interface, OMI) were introduced. With PDAF v2.0, 3-dimensional variational and ensemble-variational DA methods were added. While these enhancements have been used in many studies, most have not been explicitly published. PDAF3, discussed here, is a major upgrade in which the complete code base of PDAF was modernized and reconciled to better combine the components that were added since the initial release.

PDAF has been applied in a wide range of application areas including the ocean physics (e.g. Losa et al., 2014; Androsov et al., 2019), marine biogeochemistry (e.g. Nerger and Gregg, 2007; Pradhan et al., 2020), sea ice (e.g. Yang et al., 2015), atmosphere (e.g. Pardini et al., 2020; Corbin and Kusche, 2022; Shao and Nerger, 2024), air quality (Li et al., 2024), hydrology (e.g. Gerdener et al., 2023; Tang et al., 2024), glaciology (e.g. Cook et al., 2023; Gillet-Chaulet, 2020), solid Earth (e.g. Fournier et al., 2013; Sanchez et al., 2020; Schachtschneider et al., 2022), climate prediction (e.g. Brune et al., 2015; Polkova et al., 2019; Düsterhus and Brune, 2024), paleo climate (e.g. Masoum et al., 2024), and the exploration of coupled DA (e.g. Goodliff et al., 2019; Tang et al., 2020). Next to research applications, PDAF is used operationally for ocean, climate, and sea ice forecasting (Tuomi et al., 2018; Bruening et al., 2021; Liang et al., 2020). PDAF was also used as a tool to assist in the development of new DA methodologies and algorithms, e.g., the error subspace transform Kalman filter (ESTKF, Nerger et al., 2012b), ensemble smoothers (Nerger et al., 2014; Kirchgessner et al., 2017), regulated and adaptive localization (Nerger et al., 2012a; Kirchgessner et al., 2014), the nonlinear ensemble transform filter (NETF, Tödter, 2015), and the local hybrid Kalman-nonlinear ensemble transform filter (LKNETF Nerger, 2022). In addition to applications of PDAF and methodological research, the coupling of PDAF to different models was published, e.g., for weather research and forecast model (WRF, Shao and Nerger, 2024), the Earth System Modeling Framework with application to SCHISM (Yu et al., 2025), the terrestrial systems modeling platform, TSMP (Kurtz et al., 2016), the Community Land Model version 5.0 (Strebel et al., 2022), the ensemble and assimilation tool, EAT (Bruggeman et al., 2024), and the AWI climate model (AWI-CM-PDAF, Nerger et al., 2020). Further applications and methodological developments can be found in a full list of the more than 150 publications and 17 PhD theses, provided on the PDAF website (PDAF, 2026).

PDAF is designed as a software library. Thus, its functionality is accessed via calls to procedures (subroutines in Fortran or functions when using PDAF with languages like C or Python; below, we use the general term ‘function’ to refer to procedures, even though the Fortran code of PDAF mainly uses subroutines). PDAF provides a well-defined interface for calling its functions. Further, there is an interface for user-supplied functions, which are called by PDAF to perform operations that are specific to a model or the observations (denoted ‘callback function’ or ‘callback’ in short). An internal interface to the DA methods enables a unified structure, simplifying the addition of DA methods utilizing the PDAF functionality to handle state vectors and observations.

As a library, PDAF does not provide features for reading or writing files. Also, the configuration of PDAF is performed via function calls. Reading files containing model field information, configurations, or observations have to be implemented by the end user using the templates provided with PDAF.

The PDAF releases provide the PDAF library with different components:

- an interface for offline and online DA coupling

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

- an ensemble integration framework
- a collection of DA methods, callable through universal interfaces
- a structured functionality for observation handling (PDAF-OMI)
- interfaces to functions for covariance matrix handling in variational DA methods
- a collection of functions for ensemble and observation diagnostics, and for ensemble generation
- a functionality to generate synthetic observations for twin experiments

The structure of PDAF is modular to ensure separation of concerns, allowing different components of the DA system to evolve independently. In particular, this concerns the development of DA methods and diagnostics, the implementation of support for additional observation types and additional observation operators, and the implementation of covariance operators for variational DA methods. Also, the development of models is independent of that of the PDAF components. The separation of concerns leverages the defined interfaces of PDAF, allowing the addition of new functionality without other code changes.

Next to the PDAF library, the releases provide source codes to utilize the PDAF library:

- a strategy to handle complex state vectors containing model fields of different sizes
- template and tutorial codes, which form the basis for model-specific implementations
- command-line parsing to read configuration parameters at run-time
- full implementations of PDAF with chaotic test models.

Related to the template and tutorial codes, the PDAF website (PDAF, 2026) provides tutorials on using and implementing DA with PDAF. The PDAF core developers focus on the core functionality of the PDAF libraray and provide the templates and examples. By now, more than 30 models have been coupled with PDAF (a list is provided on the PDAF website), partly by the PDAF core developers or its user community. These model couplings are managed in separate code repositories, and it is voluntary for PDAF users whether they make their coupling code public.

3 PDAF components for DA

In this section, we describe the different components of PDAF. For brevity, we omit an introduction to DA algorithms and the required operations. For an overview of ensemble-based filter and smoother methods, including localization and inflation methods, we refer to Vetra-Carvalho et al. (2018); for variational DA methods, to Bannister (2017).

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

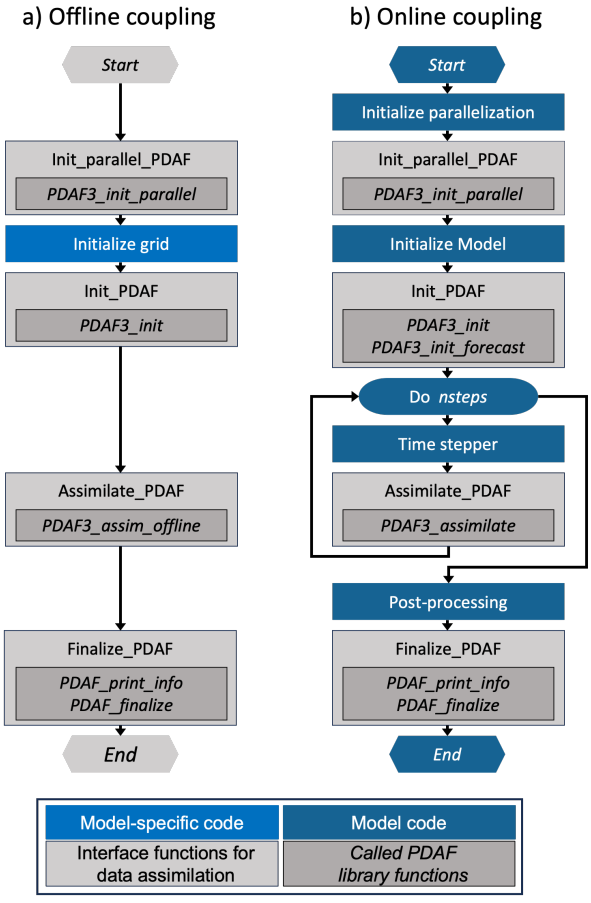

Figure 1. Left: Structure of the data assimilation program for offline-coupled DA computing a single analysis step; right: Structure of the model program with additions for cycled online-coupled DA. The interface functions for PDAF are marked light grey, while dark grey shows the PDAF library functions. For offline coupling, model-specific grid information are initialized (light blue), while for online coupling, dark blue boxes mark the blocks of model functionality. For offline coupling the parallelization is initialized in *P*DAF3_init_parallel

3.1 Interface for offline and online DA coupling

The coupling of PDAF with a model can be performed as 'online coupling' by directly inserting interface functions for PDAF into the model code, or as 'offline coupling' by using an assimilation program that is executed separately from the model and exchanges data with it via the model restart files. Figure 1 shows the code structure with PDAF for both cases. To enable an easy transition between offline and online coupling, the overall structure is the same for both approaches. Four functions (light grey boxes in Fig. 1) are used to perform the interfacing between the model and PDAF, i.e., they are called by the model code for online coupling or the main program in case of offline coupling and contain the actual calls to PDAF library functions (dark grey boxes), and the related declarations. First, the function *Init_parallel_PDAF* calls the PDAF function *PDAF3_init_parallel*

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

to initialize the parallelization for the DA (see Sec. 3.2). Subsequently, *Init_PDAF* initializes the structure of the state vector (see Sec. 3.9.1) and calls the PDAF initialization function *PDAF3_init*, which also initializes the ensemble by executing a callback function. For online coupling, the forecasting is initialized by calling *PDAF3_init_forecast*. *Assimilate_PDAF* calls an *assimilate*-function of PDAF to perform the assimilation using the chosen DA method (see Sec. 3.4). In case of online coupling, it is placed inside the model's time stepping loop and the called function *PDAF3_assimilate* also manages the ensemble forecasting (see Sec. 3.3). For offline coupling, the function *PDAF3_assim_offline* omits the ensemble forecast management. Finally, the function *Finalize_PDAF* calls PDAF functions to display information on the required memory and run time, and for deallocating PDAF's internal data structures. The PDAF library functions prefixed with *PDAF3_* build the modernized interface that was introduced with PDAF3.

While the overall code structure is designed so that the codes that need to be implemented for offline and online coupling modes are mostly identical, there are some differences. For the offline coupling, the DA program is separate from the model program. In this case, the model grid information must be read from a model-specific file (blue box *Initialize_grid*). The parallelization can be either set up to follow a domain-decomposition of the model, or to simply split each field in the state vector into parts of approximately equal size. For offline coupling, PDAF3 introduced a new function, *PDAF3_assim_offline*, that directly performs the analysis step, whereas in previous versions of PDAF, the functionality for ensemble integrations had to be explicitly switched off for the offline mode.

The online coupling of PDAF was already introduced in Nerger et al. (2005b). However, it used a more complex structure than that shown in Fig. 1 with an additional loop around the model time stepping loop ('flexible parallel' setup, see also Nerger and Hiller (2013)). This approach was motivated by the much smaller computers of that time, which typically did not allow for integrating all ensemble members at once in parallel. The right side of Fig. 1 shows the structure of the model with PDAF online coupling for the case that the parallelization is configured so that all ensemble states can be integrated at once ('fully parallel' setup). The interface functions for PDAF are inserted into the model code at well-defined places, e.g., directly after the model parallelization, directly after the model initialization, or at the very end of the time stepping loop. The fully parallel setup only needs minimal changes to the model code because the time stepping loop is not modified apart from inserting a single function call to *Assimilate_PDAF*. This function will be invoked at each time step during the model integration. This allows PDAF to also perform operations during the forecast phase, e.g., to apply incremental analysis updating (IAU). The 'flexible parallel' setup is still supported by PDAF, and used, e.g., in the implementation with the Earth system modeling framework (Yu et al., 2025). It was revised in PDAF3 to also support, e.g., IAU, and is explained in the Appendix.

An important aspect of PDAF's online coupling is that the model time stepping code does not need to be implemented as a separate, callable function. This is because PDAF never invokes the model, but the model calls PDAF functions. Further, with PDAF one can execute the model in the same way as without DA, but with additional options that control the DA. These features support a fast coupling between a model and PDAF, and an easy use of the online-coupled DA program. The four added function calls enable the parallel ensemble integration framework of PDAF, which allows the assimilative model to perform an ensemble integration and to apply the DA at specified intervals.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

3.2 Parallelization

The parallelization approach of PDAF has been discussed in Nerger et al. (2005b, 2020); Yu et al. (2025). Here, we only describe the main aspects. PDAF is parallelized utilizing the Message Passing Interface standard (MPI, Message Passing Interface Forum, 2025). The DA methods in PDAF are parallelized to operate on distributed state vectors usually following a domain decomposition as typically used in complex models. The domain-localized ensemble DA methods are further parallelized using the shared-memory parallelization standard OpenMP (OpenMP Architecture Review Board, 2024). Since OpenMP does not require explicitly distributed arrays, the implementation can be easier than using MPI for parallelization. Thus, OpenMP can be particularly attractive for offline-coupled DA if the problem size allows computing the DA on a single node of a computer. In addition to the DA methods, PDAF supports parallelization of ensemble forecasting in the online-coupled mode.

As mentioned before, the function *Init_parallel_PDAF* invokes *PDAF3_init_parallel* to initialize the MPI parallelization for offline coupling, or to modify the parallelization of the model for online coupling. For offline coupling, the parallelization is used to decompose the state vector. For online coupling, a two-level parallelization is used. First, each model instance, denoted as 'model task', can be parallelized. Second, the ensemble parallelization allows for computing multiple parallelized model tasks in parallel. MPI uses the concept of 'communicators' to group processes that can exchange data, i.e. communicate, with each other. The standard configuration of the communicators for online coupling is sketched in Fig. 2 for n model tasks using a decomposition over m processes each. All processes performing model integrations (communicator 'ENSEMBLE') are reconfigured. The processes for each model task form a 'MODEL' communicator allowing communication within each model task (columns in Fig. 2). An assimilation communicator ('ASSIM'; leftmost column) allows the processes that compute the DA analysis step to communicate with each other. Finally, a set of coupling communicators ('COUPLE'; rows in Fig. 2) allows PDAF to collect the distributed ensemble states on the 'ASSIM'-processes. Thus, for computing the analysis step, each of the processes in the leftmost column obtains a complete ensemble on a sub-domain of the model grid, which is required by most DA methods. After computing the analysis step, PDAF distributes the ensemble states again over all model tasks. Note that this standard configuration of computing the DA analysis only by the processes that compute the first model task can be changed by adapting the communicator or by replacing the functions performing the ensemble collection and distribution. For offline coupling only the first column of the parallelization ('ASSIM') exists.

PDAF also supports to perform the parallel ensemble integration with a fraction of total processes, e.g., if a so-called input/output server (io-server) is used. In this case, some processes are reserved for the the io-server ('OTHER' in Fig. 2. Only the processes computing the model integrations ('ENSEMBLE') are then provided to PDAF to be reconfigured for the ensemble integration. The provided template function *Init_parallel_PDAF* is generic, and one can use it without modifications when implementing PDAF with various models.

3.3 Ensemble integration framework

When not using a DA algorithm, the PDAF online coupling (right side of Fig. 1) reduces to an ensemble integration framework, facilitated by the inserted functions. In particular, the modified parallelization permits to compute n parallel model

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

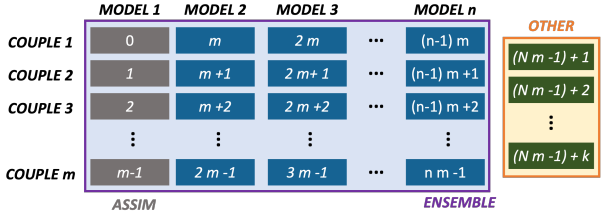

Figure 2. Standard MPI layout of the online-coupled PDAF implementation with n model tasks each distributed over m physical processor cores. The box 'ENSEMBLE' marks the communicator of all processes performing the model integrations, while 'OTHER' is an optional group of k processes that might perform, e.g. file operations. Each small box represents an MPI process (rank). For model processes this corresponds to a subdomain. In the box 'ENSEMBLE', each column corresponds to a model task integrating one or multiple ensemble states. Each row corresponds to a subdomain of the model. Each row and column has its own MPI communicator, 'COUPLE' or 'MODEL'. In addition, the processes in the first column, i.e. first model task, are grouped in the assimilation communicator ('ASSIM') that computes the analysis step.

tasks. Each model task is executed independently, and *Init_PDAF* will also be executed with ensemble parallelization. This changed parallelization does not need further modifications to the model[1]. Unlike the model, PDAF is aware of ensemble parallelization and can perform parallel communication across different model tasks. This allows the functions *PDAF3_init* and *PDAF3_init_forecast* to initialize ensemble handling. Then, the model integration loop is computed in the ensemble-parallel form, and *PDAF3_assimilate* completes the counting of time steps and ensemble states and to apply DA at the specified time step. For ensemble integration without DA, PDAF provides a function *PDAF3_prepost* that provides the forecast ensemble to a callback function, without requiring any functions for the analysis step, such as observation initialization. This callback function gives users the option to manipulate or save the ensemble states at prescribed intervals. The ensemble management is hidden inside the executed PDAF library functions.

3.4 DA methods

PDAF provides a wide range of DA methods, including EnKFs, nonlinear and hybrid ensemble filters, and 3D-Var schemes, see Table 1. All DA methods are parallelized. For EnKFs, there are variants which process observations at once or, added with PDAF3, serially. Localization is available for all ensemble methods except the particle filter and parameterized 3D-Var. The localization approaches vary between methods that apply localization to the state covariance matrix ('covariance localization') and those that perform analyses on local domains, combined with a distance-dependent weighting of observations ('observation localization' see, e.g., Janjić et al., 2011). For the latter variants, separate global and localized variants are implemented because localization causes overhead from generating local state vectors and computing separate updates for each local domain, even when these variants become equivalent to the global filters when all observations are assimilated for each

[1]If the model is implemented to use the predefined MPI communicator 'MPI_COMM_WORLD', it needs to be replaced with a communicator variable to allow splitting into model tasks

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Table 1. Overview of DA methods provided in PDAF v3.1. An 'x' in the column 'global' indicates whether an explicit formulation without localization is provided, the column 'local' indicates whether localization is provided, with 'c' denoting covariance localization, and 'do' denoting domain localization, including observation localization. An 'x' in the column 'smoother' indicates that a smoother is also included with the DA method.

DA method	global	local	smoother	
EnKF	x	c	x	Ensemble KF, Evensen (1994)
ETKF	x	do	x	Ensemble transform KF, Bishop et al. (2001); Hunt et al. (2007)
ESTKF	x	do	x	Error subspace transform KF, Nerger et al. (2012b)
SEIK	x	do	x	Singular evolutive interpolated KF, Pham (2001)
EnSRF	x	c		Ensemble square-root filter with serial obs. processing, Whitaker and Hamill (2002)
EAKF	x	c		Ensemble adjustment KF with serial local least-squares update, Anderson (2003)
NETF	x	do	x	Nonlinear ensemble transform filter, Tödter and Ahrens (2015); Tödter et al. (2016)
PF	x			Particle filter, see van Leeuwen et al. (2019)
KNETF		do		Hybrid Kalman-nonlinear transform filter, Nerger (2022)
3D-Var	x			3D-Var with parameterized covariances, see Bannister (2017)
ensemble 3D-Var	x	do		Ensemble 3D-Var , see Bannister (2017)
hybrid 3D-Var	x	do		Hybrid 3D-Var, see Bannister (2017)

local domain. For most of the ensemble filters, a smoother functionality is also implemented, which allows for updating past estimates from future observations. In addition, for 3D-Var methods, PDAF provides a variant that uses covariance operators representing parameterized covariances, but also schemes using an ensemble or a hybrid combination of ensemble and parameterized covariance operators. These implementations follow Bannister (2017). To the authors' knowledge, PDAF is currently the only DA framework that provides the full range of DA methods, including ensemble Kalman methods with one-step and serial processing of observations, nonlinear ensemble filters and smoothers, and 3D-Var methods.

3.5 Operations in the assimilation step

The implementation of DA methods in PDAF follows a uniform scheme. This leads to a universal interface in which all ensemble-based filters and smoothers can be used from a single function call (*PDAF3_assimilate* for online or *PDAF3_assim_offline* for offline coupling, see Figure 1). Likewise, all 3D-Var methods can be used from a single function call (*PDAF3_assimilate_3dvar_all* for online or *PDAF3_assim_offline_3dvar_all* for offline coupling). The calls for 3D-Vars are different from those for ensemble-based methods because 3D-Var requires additional callback functions. The universal assimilation functions provide flexibility in testing different DA methods without recoding.

The general flow of operations inside the universal functions used for all DA methods is shown in Figure 3. Here, the grey fields mark functions provided by PDAF, while the colored fields in the left column are callback functions that perform

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

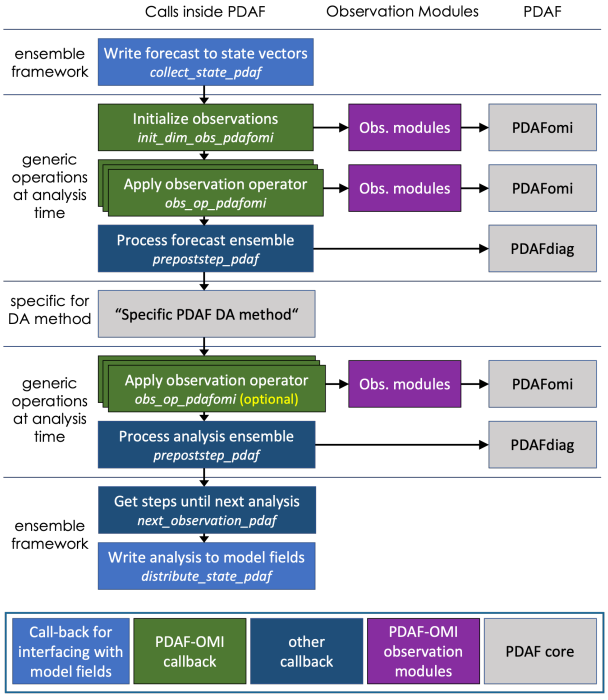

Figure 3. Processing steps applied in the assimilation functions of PDAF for all DA methods. The left column contains functions inside PDAF; the center column contains observation modules with observation-specific operations; the right column shows calls to PDAF functions. There are generic steps to prepare observations and process the ensemble before and after the specific DA method, which computes the update to the ensemble or the single-state estimate in the case of 3D-Var. Names in italics denote the default names of the callback functions. The observation operator is applied to all ensemble states, indicated by a stack of boxes.

operations specific to the model or observations. On the level of the ensemble framework, thus only with online coupling, the forecast fields are written into the ensemble of state vectors before the analysis step, and afterwards the state vectors are written back into the model fields. Further, the number of time steps for the following ensemble forecast is determined. Before and after the function computing the specific DA analysis update, operations are performed independent of the specific DA method. Before the analysis update, the observation information is initialized and the observation operator is applied to all ensemble state vectors. This functionality uses PDAF-OMI as indicated by the calls to the observation modules (purple fields in Figure 3), see Sec. 3.7 for details. Further, the callback function *prepoststep_pdaf* is called to provide the user code access to the forecast ensemble. This function is used, e.g., to compute diagnostics on the forecast or analysis ensemble, or to write the ensemble or its mean state to files. After computing the analysis update, the observation operator can optionally be applied again before *prepoststep_pdaf* is called for the analysis ensemble. The order of these operations is similar to that described in Chen et al. (2025), but in PDAF3, the observation initialization was moved out of the specific DA analysis step. This simplified the PDAF-internal code, and also allowed the introduction of PDAF-OMI diagnostics on the observed ensemble (see Sec. 3.7) before and after computing the analysis update.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

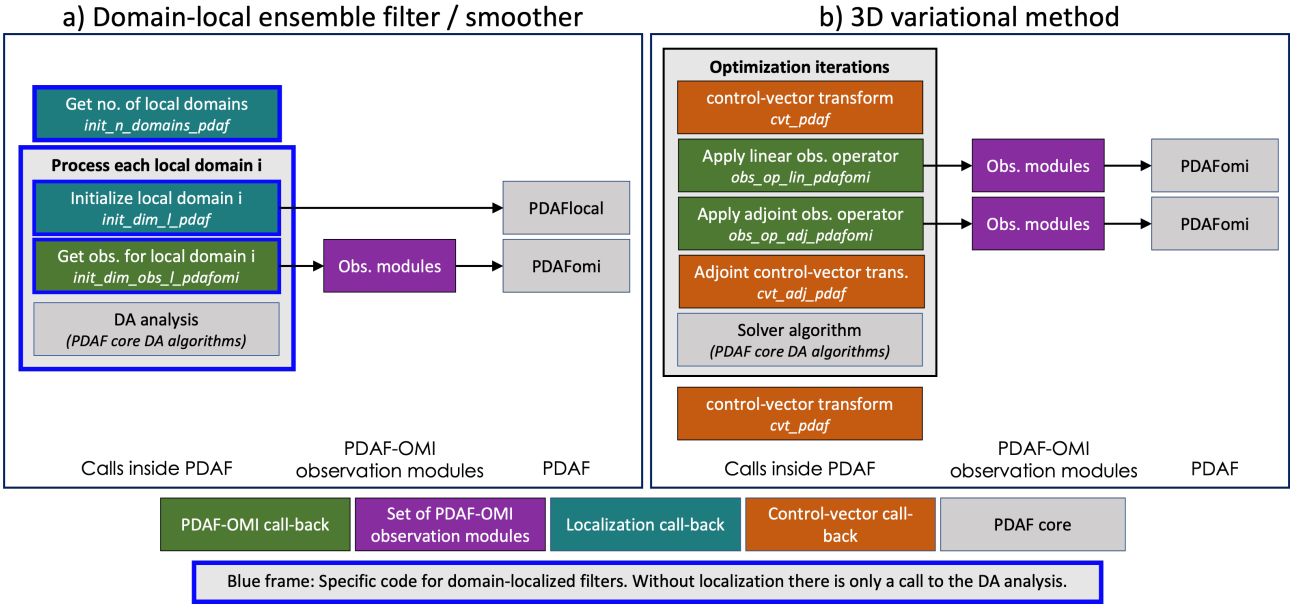

Figure 4. Specific processing steps for the actual DA analysis. The left shows the steps for ensemble methods, also indicating the additional operations for domain localization. The right shows the steps for the 3D-Var update, which adds the control-vector transform operations in which, e.g., the covariance operators are applied to convert from the state vector to the control vector.

The specific DA method usually executes additional callback functions, as is exemplified in Figure 4 for a domain-localized filter and smoother and for 3D-Var. For domain-localized filters and smoothers (Figure 4a), a sequence of operations have to be performed: 1) the number of local analysis domains has to be determined; 2) in a loop over these local analysis domains, the local domain information has to be initialized, which are the coordinates of the local domain, the size of the local state vector, and an index vector that maps the full state vector to the local state vector; 3) the observations to be assimilated in a local analysis domain have to be determined, which is done by PDAF-OMI through the observation modules (see Sec. 3.7); 4) the actual local analysis update for the filter and smoother are then computed by the PDAF core DA algorithm. For global filters, only the function for the DA analysis is called, as there is no local analysis loop and no additional calls to callback functions. This also holds for the filters with covariance localization (EnKF, EnSRF, EAKF).

For 3D-Var (Figure 4b), there is a sequence of operations within each optimization iteration. The application of a linearized and an adjoint observation operator required by 3D-Var are handled via the observation modules for PDAF-OMI. In addition, there are callback functions which transform between the state vector and the control vector computed in each iteration. The control vector is the vector for which the actual optimization is computed (see Bannister, 2017). In these functions, covariance operators (see Sec. 3.8) are applied. They are called during the iterations when computing the cost function of 3D-Var, and once afterwards to compute the final optimized state vector. For ensemble and hybrid 3D-Var, the call structure is analogous, but there are additional functions to compute the control vector transformation associated with the ensemble-represented covariance

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

matrix. In addition, the optimization step for the ensemble and hybrid 3D-Var will be followed by an EnKF step that updates the ensemble spread.

3.6 Callback functions

For operations specific to the model or to observations, PDAF uses callback functions. They are provided to PDAF with the user code. PDAF generally follows the approach of pulling information from model- or observation-specific operations. Thus, when calling PDAF assimilation functions, PDAF executes a callback function at the time when some information is needed that is specific to the model or the observations. This approach provides a clear code structure with callback functions that focus on a certain operation, e.g., writing model fields into a state vector or applying an observation operator to a state vector. Further, it keeps the PDAF core code agnostic of the particular model and observations. The approach also ensures that the required memory can be kept low, as e.g. information on local analysis domains are only stored for one domain at a time instead of all local domains at once. In the case of online coupling, the implementation is done within the context of the model, such that, e.g., shared variables of the model can be utilized, and the code structure of the model can be followed. An overview of the callback functions relating to the model is provided in Table 2.

With the callback functions, PDAF also follows the strategy to implement operations directly, rather than providing generic code for which a user would need to provide a description of data to use this code. An example is the function to write model forecast fields into a state vector (*collect_state_pdaf* in Fig. 3). Models use vastly different data structures to store a model field. For example, different ocean models store fields as one-, two-, three-, or even four-dimensional arrays, and one array index could describe a horizontal or vertical direction, a general grid-point counter, or the variable index. Writing a field from such arrays into the state vector is usually just a nested loop. DA software could provide a generic function for this operation for the user to specify the underlying data structure, but internally it would need to handle many different cases. This approach would certainly introduce overhead. A design decision for PDAF was to let the user implement the nested loop according to the model's data structure, rather than providing a generic function for such an elementary operation. An exception to this design is the observation handling by PDAF-OMI.

3.7 Observation handling with PDAF-OMI

The DA algorithms require different information about the observations. Next to the vector of observation values, $\mathbf{y}$, specifications of observation errors are required. Here, usually some operation involving the observation error covariance matrix $\mathbf{R}$, is computed. To apply localization, the coordinates of each observation are required, such that distances can be computed. Finally, the observation operator $\mathcal{H}$ needs to be defined. The observation operators are applied analogously for all DA methods, but the operations involving the matrix $\mathbf{R}$ are distinct for different DA methods. For example, the ETKF and ESTKF, but also particle filters, need to multiply some matrix by the inverse of $\mathbf{R}$. In contrast, the original EnKF (Evensen, 1994) adds $\mathbf{R}$ to some matrix. Distinctively, the methods with serial observation processing (EnSRF, EAKF) need to assess every single element of $\mathbf{y}$, $\mathcal{H}(\mathbf{x})$, and the corresponding entry from the diagonal of $\mathbf{R}$, which is always assumed to be diagonal for these filters. The observational information must be managed across multiple observation types (e.g., measurements of different physical variables

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Table 2. Overview of callback functions relating to the model fields and model grid. The callback functions relating to observations are described in Sec. 3.7, while callback functions used specifically for 3D-Var methods are described in Sec. 3.8

Function	Description of functionality
collect_state_pdaf	Fill a state vector array provided by PDAF with information from model fields
distribute_state_pdaf	Write values from a state vector array provided by PDAF into model field arrays
init_ens_pdaf	Fill the ensemble array provided by PDAF with the initial ensemble states
next_observation_pdaf	Provide PDAF with the number of time steps of the following forecast phase
prepoststep_pdaf	Perform operations on the ensemble array provided by PDAF
init_n_domains_pdaf	For domain-localized DA methods: Provide PDAF with the number of local analysis domains
init_dim_l_pdaf	For domain-localized DA methods: Provide PDAF with size of the local state vector and an array of the indices of elements of the local state vector in the global state vector; further initialize the coordinates of the local analysis domain.

from satellites or other instruments), each with different characteristics. For example, remote-sensing satellite observations of processing level 3 or 4 are usually available on some regular grid, while in situ measurements are often irregularly distributed. In addition, the management of observational information must be aware of the parallelization. For example, the observation operator is applied as $\mathcal{H}(\mathbf{x})$ to some model state $\mathbf{x}$, which can be distributed in memory due to a domain decomposition of the model. When a DA algorithm then computes the innovation vector $\mathbf{y} - \mathcal{H}(\mathbf{x})$, one has to ensure that both vectors are sorted identically.

To provide a structured way to implement the observation-related operations described above, the PDAF observation module infrastructure (PDAF-OMI) was developed. The use of PDAF-OMI is optional but strongly recommended, as it simplifies the implementation. With PDAF-OMI, each observation type is implemented independently, and PDAF-OMI assembles actual multivariate observation vectors, ensuring a consistent parallelization.

The design of PDAF-OMI is motivated by object-oriented programming (OOP), but for the sake of simplicity, it avoids abstractions and inheritance features of OOP and relies on encapsulation in Fortran modules and data types. For each observation type, one implements an observation module that provides the actual operations. For ensemble DA methods, there are three functions; for 3D-Var, one needs five functions, as shown in Figs. 3 and 4 and listed in Table 3. Each observation module thus encapsulates one observation type, ensuring that each observation module can be developed independently. In each observation module, the data for the specific observation is stored in a Fortran derived data type, denoted *thisobs*. This allows PDAF-OMI to access this data via pointers. Apart from one exception, all functions in the observation modules are very short and only call a function of PDAF-OMI, providing it with the variable *thisobs*. Only the function for the general initialization of the observation information (*init_dim_obs_TYPE*) is longer because it reads the observation data from a file, performs checks for valid observations, and initializes arrays holding the observation values, coordinates, and errors. These arrays are then provided to

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

PDAF-OMI in a function call. In addition, an index array is initialized to describe the elements of the state vector required for the observation operator. If an observation operator with interpolation is used, also an array holding interpolation coefficients can be initialized. For the DA methods using covariance localization, a PDAF library function is called to provide PDAF-OMI with the coordinates for the state vector elements.

The index array initialized in *init_dim_obs_TYPE* allows for an implementation of the observation operators that is independent of the particular model. PDAF currently provides observation operators for interpolation in regular and unstructured grids, for observations that are model state values at grid points, and for observations for which the observation operator is applied outside of the PDAF functions, e.g., during the model integration. In addition, a particular observation operator is provided for strongly-coupled DA in which the different model components are executed on separate processes (see Tang et al., 2021; Nerger et al., 2020). Further observation operators can be implemented by users based on a template (see, e.g., the observation operators for Global Navigation Satellite System (GNSS) observations in Shao and Nerger, 2024).

The universal assimilation functions introduced with PDAF3 (see Sec. 3.5) make essential use of PDAF-OMI because it internally provides the functionality for many observation-related operations that are specific to a DA method. This also reduces the number of required functions that need to be implemented for observation handling to just two for ensemble-based filters with global analysis or covariance localization, three for domain-localized filters, and five functions for 3D-Var as described in Table 3. With PDAF-OMI, the callback functions called by PDAF are generic functions that call the observation-specific functions in the observation modules (middle column in Figs. 3 and 4).

Each function in the observation modules calls one or several functions of PDAF-OMI. These allow PDAF-OMI to also provide advanced functionality like the exclusion of observations that are potential outliers if the difference of their value from the ensemble mean exceeds a specified factor of the observation error variance. PDAF-OMI also handles localization with an option for isotropic or non-isotropic localization, as well as spatially varying localization distances. The standard mode of PDAF-OMI considers uncorrelated observation errors (a diagonal matrix $\mathbf{R}$), but functions to handle correlated observation errors are also provided. In addition, one can enable a debug mode to print values from the different arrays handled by PDAF-OMI, e.g., for a single local analysis domain. While PDAF-OMI uses derived-type variables of Fortran, it can also be used with other programming languages like C or Python as described by Chen et al. (2025).

PDAF-OMI stores the observational information internally. To access these, e.g., in the callback function *prepoststep_pdaf*, PDAF provides functions to retrieve the

- vector of observations and array of related coordinates (*PDAFomi_diag_get_obs*)
- observed ensemble mean vector (*PDAFomi_diag_get_HXmean*)
- the observed ensemble (*PDAFomi_diag_get_HX*)
- the inverse observation error variances. (*PDAFomi_diag_get_ivar*)

These functions provide the information separately for each observation type. In addition, PDAF-OMI provides observation-related diagnostics, see Sec. 3.9.3.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Table 3. Overview of functions for PDAF-OMI, which are implemented for each observation type. They are called by the generic callback routines of PDAF-OMI. Here *TYPE* is a placeholder for the observation type.

Function	Description
init_dim_obs_TYPE	Overall initialization function. It reads the data for one observation type from a file, applies quality checks, and initializes the observation values, errors, coordinates, and an index array that serves as input to the observation operator. Then, it returns the number of observations directly to PDAF. (all DA methods)
obs_op_TYPE	Function for observation operator; it receives a state vector from PDAF and returns the observed state vector. Short code mainly calling the selected observation operator function. (all DA methods)
init_dim_obs_l_TYPE	Function to initialize a local observation vector for domain-localized DA methods. Short code calling PDAF-OMI function searching observations within some distance. It returns the number of local observations directly to PDAF. (domain-localization only)
obs_op_lin_TYPE	Function for linearized observation operator. It receives a state increment vector from PDAF and returns the observed state vector. Short code mainly calling the selected linear observation operator function. (3D-Var only)
obs_op_lin_TYPE	Function for adjoint observation operator. It receives an observed state vector from PDAF and returns the state vector. Short code mainly calling the selected adjoint observation operator function. (3D-Var only)

3.8 Covariance operations in variational DA

3D-var methods use a so-called control-vector transformation to map the state vector $\mathbf{x}$ to the control vector $\mathbf{c}$. Typically, this transformation involves a square root of the state error covariance matrix in a factorized form represented by covariance operators. Such operators describe variances and covariances between different locations and also balance different model fields for multivariate DA updates. The transformation can also involve variable transformations separating unbalanced variables (e.g. Weaver et al., 2005). PDAF implements the control vector transformations in the form of two callback functions (see Figure 4), each providing the transformation in one of the directions. The functions provide maximum freedom in implementing the operations, including the option to use external libraries.

3.9 Further functionality for DA systems

3.9.1 Handling of multivariate state vectors

The PDAF library functions treat state vectors as an abstract vector without knowing what variables are stored in it, because this allows the DA methods to be implemented in a generic form. In contrast, the case-specific user code needs to construct the state vector, filling it with the data from different model fields. An efficient approach for this is provided by the template codes

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

in the PDAF release and in the example code used in Sec. 4. The approach uses a derived type variable *id* in which the indices of each field in the state vector are stored. This variable is adapted by the user to include the fields of a specific model. Using *id*, one can access a field in the state vector by this name (e.g., writing *id%temp* in Fortran code to access the temperature field) while one can also implement a loop over all fields as numbered indices. In addition, another derived type variable *sfields* stores the dimension of a field in the state vector, its offset from the beginning of the vector, and optionally further information like the field's name. This variable is allocated as a vector with the length given by the number of fields, so that the information for each field can be stored. One can use this variable to, for example, access the offset of the temperature field in the state vector with *sfields(id%temp)%off*, or one can implement a loop over all variables in the state vector, e.g., for file output. Note that these variables are only used in the user code and do not require any additional functions to access their information, as they are directly accessible.

3.9.2 Ensemble generation

All ensemble DA methods require an initial ensemble of N model state realizations, which represents the initial state estimate and its uncertainty. There are different ways to generate the initial ensemble. A very simple method is to use model states, which are selected randomly at different points in time from a previous model simulation. As the mean of these states is unlikely to represent the best initial estimate, one can replace the ensemble mean state with a chosen estimate. The random selection can be easily implemented, but it usually leads to a slow convergence proportional to $1/\sqrt{N}$ to the actual covariance matrix. Thus, a rather large ensemble might be necessary.

A more sophisticated ensemble initialization method is to use empirical orthogonal functions (EOFs) computed from a previous model run and then apply second-order exact sampling (Pham, 2001), below denoted as O(2)-sampling, which creates a random ensemble that exactly samples the covariance matrix represented by the EOFs. This approach leads to a faster convergence of the covariance matrix according to the spectrum of singular values of the EOFs and allows applications to run with rather small ensembles between 20 and 50 states, as demonstrated by various applications (e.g. Goodliff et al., 2019; Tang et al., 2021; Yu et al., 2025). In practice, the sampling method proceeds in two steps. In the step, fields are read from a model trajectory and computes a singular values decomposition to generate the EOFs and related weights (the singular values). In the second step, the EOFs and weights are provided to the O(2)-sampling method. For ensemble size N, the O(2)-sampling uses the leading $N-1$ EOFs. It multiplies these by the square root of their weight and then computes the ensemble perturbations by multiplying the EOFs with an orthogonal random matrix that is constructed to preserve the mean and covariances (see, e,g., the appendix of Nerger et al., 2012b). PDAF provides functions and example codes for both the computation of EOFs and the O(2)-sampling.

Another possibility is to generate ensemble spread by perturbing process parameters or external forcing. This approach is often employed for convergent systems in which perturbed initial states progressively approach the same state in the absence of perturbations. Examples are the Earth's upper atmosphere (e.g., Lee et al., 2012; Corbin and Kusche, 2022) or ocean circulation (e.g., Storto and Andriopoulos, 2021), where a single atmospheric forcing would lead to convergent dynamics. Using perturbations to generate representative initial spread starting from a single unperturbed model state may require integrating the

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

model for a long period. As such, this method is primarily used to maintain the ensemble spread after an ensemble is initialized using the methods described above. Perturbing process parameters is also common in DA for ocean-biogeochemical models, which involve many parameters that control different modeled processes (e.g., Pradhan et al., 2019; Mamnun et al., 2025). Here, the biogeochemical processes respond rapidly to perturbations, e.g., the onset of a phytoplankton bloom, so that the perturbation can result in sufficient spread. Further, hydrological DA applications use perturbed parameter fields because the DA is commonly used to estimate parameters describing subsurface features such as hydraulic conductivity and soil porosity (e.g., Li et al., 2025). For parameter perturbations, PDAF provides the function *PDAF_generate_rndvec* to generate random vectors according to a specified distribution like normal, log-normal, or uniform.

3.9.3 Ensemble and observation diagnostics

Different diagnostics are used in DA. These either refer to the model state and ensemble or to the assimilated observations. For example, one can quantify the quality of the ensemble, such as its skill in estimating uncertainty and its representativeness with respect to observations. Table 4 provides an overview of the available diagnostic functions in PDAF. Most of the functions can be generally used with PDAF, while three diagnostic functions are specific to PDAF-OMI. These functions provide statistics separately for each assimilated observation type. For example, the computed correlation, centered root mean square difference (RMSD), and standard deviation are quantities that can be displayed in a Taylor diagram (Taylor, 2004).

3.9.4 Generating synthetic observations

Synthetic observations are used for twin experiments or observation system simulation experiments (OSSEs, e.g. Prive et al., 2023). PDAF provides functionality for generating synthetic observations using PDAF-OMI. The functionality is provided as an alternative to PDAF's *assimilate*-functions following the same code structure. PDAF-OMI is used to first initialize the observation information from the real observations. This information is then used by PDAF to apply the observation operator to a model state, followed by the addition of random observation errors of the specified variance. This synthetic observation vector is then provided to a callback function, where it can be written into a file. In an OSSE, the observations from this file can be read in the observation module for the actual DA. The template code of PDAF provides writing and reading routines utilizing the binary netCDF file format. The synthetic observations can be generated by executing PDAF's online-coupled mode with ensemble size one. In offline mode, one can generate observations by reading model fields from the outputs of a prior model run. Overall, the observation generation uses the same code structure as the actual DA and can hence be implemented with minimal coding effort.

3.9.5 Incremental Analysis Updates

Incremental analysis updates (IAU, Bloom et al., 1996) distribute the increment computed by the DA method over a number of time steps during the model integration. IAU can reduce initialization shocks, e.g., resulting from imbalances in the analysis states, if the increment is added instantaneously. PDAF3 provides a model-agnostic functionality to apply IAU in which the

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Table 4. Overview of diagnostic functionality of PDAF. Functions prefixed with *PDAF_diag* are generally usable, while those starting with *PDAFomi_diag* are part of PDAF-OMI and operate on the observations.

Function	Computed quantity
PDAF_diag_ensmean	Ensemble mean state vector
PDAF_diag_variance	Ensemble variance vector (e.g. for plotting like the state vector)
PDAF_diag_stddev	Mean ensemble standard deviation (separately for each model field in state vector)
PDAF_diag_rmsd	Root mean square difference (RMSD) of two vectors
PDAF_diag_diffstats	Statistics on the difference between two vectors, e.g. observation and model state: Correlation, centered RMSD, mean bias, mean absolute difference, variance of observations and of observed ensemble mean
PDAF_diag_effsample	Effective sample size (often used for nonlinear DA)
PDAF_diag_compute_moments	Statistical moments of the ensemble (mean, variance, skewness, and excess kurtosis)
PDAF_diag_CRPS	Continuous ranked probability score with decomposition in reliability and resolution (CRPS, Hersbach, 2000)
PDAF_diag_histogram	Rank histograms (see, e.g. Hamill et al., 2001)
PDAF_diag_reliability_bidget	Reliability budget (Rodwell et al., 2016).
PDAFomi_diag_rmsd	RMSD between observation and model state separately for each observation type
PDAFomi_diag_CRPS	CRPS analogous to *PDAF_diag_CRPS* but specific for observations and separately for each observation type
PDAFomi_diag_diffstats	Statistics analogous to *PDAF_diag_diffstats* but specific for observations and separately for each observation type

actual functionality to add the increment over time is included in the *assimilate*-functions of PDAF which are called at each time step. The IAU is activated by a function call (*PDAF_iau_init*) that sets the number of time steps over which IAU should act and the IAU weight function. Available weight functions are a uniform weight and the triangular shape with maximum weight in the middle of the IAU period, as described by (Bloom et al., 1996), but the user can also provide a vector describing another function.

3.10 Model coupling codes and documentations

3.10.1 Template and tutorial codes

The implementation of a DA system with PDAF consists of the user-provided source code for the coupling to the model, either online or offline. This code calls the PDAF library functions. In addition, there are callback functions and PDAF-OMI observation modules. The coupling code only needs to be implemented once, while later code changes mainly involve

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

implementing the assimilation of additional observation types or adding further diagnostic calculations. Existing public codes for different models (see the PDAF website, PDAF, 2026)) can be used as the basis for one's own implementation. For models that have not yet been coupled to PDAF, the PDAF releases provide an implementation tutorial and templates to help with the implementation. The template codes are instrumented to guide through the implementation by extensive documentation inside the source code, and by screen outputs that notify about functionality that still needs to be completed. The tutorial codes are accompanied by slide sets, available from the website, that explain the code structure and implementation steps. Section 4 discusses a tutorial code for online and offline coupled DA and shows some properties of ensemble DA.

3.10.2 Full implementations with test models

PDAF was used in several studies that developed new DA methods or assessed properties of existing DA methods (e.g., Nerger et al., 2012a, b, 2014; Kirchgessner et al., 2014, 2017; Nerger, 2022). These studies typically used low-dimensional chaotic models to characterize the methods' behavior. The PDAF releases provide full implementations of the three-variable model by Lorenz (1963), the Lorenz-96 model (Lorenz, 1996), and two variants of the more advanced models by Lorenz (2005). The properties of established DA methods using these have been described in many publications, which can serve as references for one's own studies. For studies that used PDAF, one can also reproduce the published results using the options documented on the PDAF website (PDAF, 2026).

4 Application and code example

Different applications of PDAF have already been discussed in the literature. Recent open-source examples are Chen et al. (2025) using the Modular Arbitrary-Order Ocean-Atmosphere Model (MAOOAM) with both Fortran and Python, Nerger et al. (2020) for using the Alfred-Wegener-Institute Climate Model (AWI-CM), and Yu et al. (2025) using the Earth System Modeling Framework (ESMF) and the ocean model SCHISM (Semi-implicit Cross-scale Hydroscience Integrated System Model). In addition, the parallel performance of DA applications with PDAF have been assessed in different studies (e.g., Nerger and Hiller, 2013; Nerger et al., 2020; Kurtz et al., 2016; Yu et al., 2025). Although these studies did not use PDAF3, the codes are still usable and a similar parallel performance is expected with PDAF3.

Instead of the more complex models used in these application examples, we discuss here a particularly simple but illustrative application example. The model domain is rectangular with directions x and y. The model represents two fields, denoted as 'field A' and 'field B'. The model dynamics simply shift the model field by one grid point upward in the y-direction. The coordinates are given by the grid point indices in both directions. The size of the model grid can be freely chosen, a larger grid with, e.g., size 512×2048 can be useful to assess run times of DA methods, while smaller grid, e.g. 18×36 grid points used here, allows one to quickly perform many experiments with varying DA parameter values to assess the behavior of DA methods. Analogous to many models in Earth sciences, the model is coded in Fortran and uses a typical structure of initialization, time stepping, and post-processing. The model fields are declared in a Fortran module, and the model uses the widely used binary file format NetCDF. The model is parallelized using MPI with a domain decomposition along the x-direction. Note, that the

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

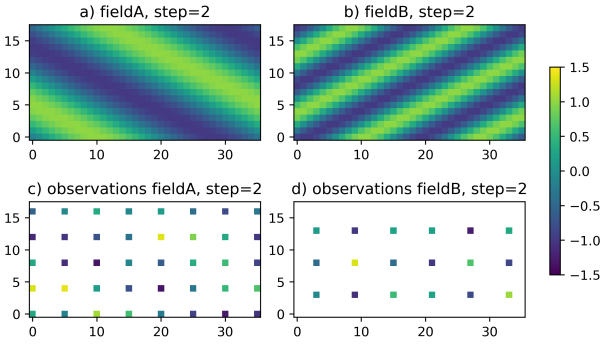

Figure 5. Model and observations at time step 2. Shown are the a) field A, b) field B, c) observations of field A, d) observations of field B.

model is not intended to represent particular physics, but is chosen to illustrate the effects of DA with different DA methods and of PDAF functionalities, e.g., localization or inflation. Even more than showing these effects, the example source code demonstrates a realistic implementation of a PDAF-based DA system. The implementation for a real 2-dimensional numerical model with a rectangular regular grid would be analogous. The provided code includes a detailed documentation for compiling and running experiments in the form of a README file. Furthermore, the code includes extensive inline documentation explaining the functionality of each file and function.

For DA, the example includes implementations of online and offline coupled ensemble filter methods, supporting all ensemble-based filters listed in Table 1. The ensemble can be initialized by reading model fields or by O(2)-sampling as described in Sec. 3.9.2. When reading model fields, one has the option to replace the mean of these fields with the mean from the precomputed covariance matrix file. This ensures the use of the same ensemble mean state as for O(2)-sampling. In real applications, one might use the best available state estimate as the initial ensemble mean state. To assess the DA process, the function *prepoststep_pdaf* (see Fig. 3) computes the ensemble standard deviation and diagnostics for the difference between observed ensemble mean and observations. For completeness, the code also includes the options to use IAU and PDAF's functionality to generate synthetic observations. For using this functionality, refer to the README file and the PDAF online documentation.

For generating input files for the truth, which the DA is intended to estimate, the initial ensemble states and the observations, the example includes a Python script in which both the grid size and the density and error level of the observations can be configured.

4.1 Model and DA setup

The grid size is set to 36 points in the x-direction and 18 points in the y-direction. The two fields are initialized with a sine wave of different orientation and wavelength. The fields serving as 'truth' at time step 2 are shown in the upper row of Fig. 5. Observations are generated from the truth fields by selecting the model values on a regular coarse grid and perturbing them with uncorrelated Gaussian noise with standard deviation of 0.5 for field A and 0.25 for field B. The observation grid is kept fixed over time. The lower row of Figure 5 shows an example of observations at model time step 2. Assimilation experiments are

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

performed with the global and localized variants of the ESTKF for ensemble sizes between 3 and 20. The experiments run over 20 time steps with assimilation at every second step. A multiplicative inflation is varied in terms of the forgetting factor (see, e.g., Nerger et al., 2012b) in the range between 0.4 and 1.0 in steps of 0.1 to find an optimal inflation. The localization is applied using the finite 5th-order polynomial from Gaspari and Cohn (1999). The localization radius refers to the support radius of the function and is varied here between 5 and 90 grid point units.

Only observations of field A are assimilated and a discussion of the multivariate effect is omitted. This is because the two model fields are not dynamically linked due to the simplified dynamics. The assimilation performance will be assessed by computing the root-mean square error (RMSE) of the state estimate compared to the true model state at time step 20. The experiments focus on the influence of localization and on the effect of different ensemble initializations. In particular, the experiments compare the initialization by reading prepared ensemble files and directly using their mean state as initial state estimate (initialization type ENS1); the initialization by reading ensemble files and replacing their mean by the mean state of 20 ensemble states (ENS2), and the initialization by O(2)-sampling using the mean state of all 20 ensemble states as central state of the ensemble (ENS3). The ensembles are chosen such that for ensemble size 20, all three initializations produce the same result. With this, the experiments will assess the effect of the ensemble initializations when using smaller ensembles, i.e., $N \leq 20$. In addition, the experiments allow us to study the influence of localization in comparison to a global filter for different ensemble sizes.

For ENS1 and ENS2, a set of 20 ensemble states of field A was generated by rotating the sine wave of field A and varying its wavelength. Thus, it is assumed that the orientation and wavelength of the wave might be unknown. An example of 6 ensemble states for ENS1 is shown in Figure 6a-f. The rotation is chosen so that the first 11 out of the 20 ensemble states do not match the orientation of the true state. Only for larger ensembles, an approximately correct orientation is included. Note, that the ensemble is chosen systematically in ENS1 and ENS2 to demonstrate effects of the ensemble states, while in real cases one would rather use random choices.

For the O(2)-sampling in ENS3, a covariance matrix has been generated from the 20 ensemble states used in ENS1 and ENS2, which is used to sample the chosen number of ensemble states as described in Sec. 3.9.2. The ensemble states of type ENS3 for ensemble size 6 are shown in Figure 6g-l. Here, each ensemble member includes the information of the leading 5 EOFs and the shapes of the ensemble states of type ENS3 look very different from those of type ENS1. The O(2)-sampling compresses the ensemble information according to the EOF weights. The accumulated sum of the leading 9 weights is about 96% of the total sum. Thus, DA with an ensemble of size $N = 10$ of ENS3 is expected to yield a result similar to using ENS1 or ENS2 with $N = 20$. Note, that the big difference in the shapes of the ensemble states from ENS1 and ENS3 is also due to the fact that the initial mean state is very close to zero, so that the shape of each ensemble state is dominated by the computed perturbation.

The experiments are performed here with very small ensembles. This is possible because the model setup has a low dimension (the sine wave is essentially one-dimensional) and the ensemble is chosen in a way that the approximately correct orientation of the one-dimensional wave is included if the ensemble is large enough. High-dimensional geophysical models in general require larger ensemble sizes and localization. However, by using the ensemble type ENS3, successful DA has been

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

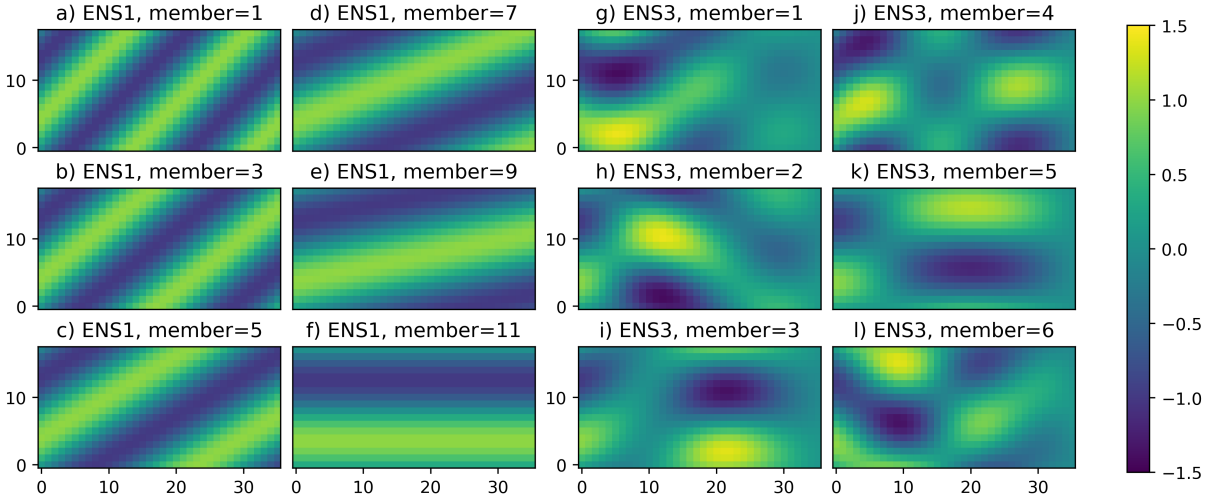

Figure 6. Example of ensemble states of field A. Left (a-f): for ensemble initialization type ENS1; right (g-l): for type ENS3 based on O(2)-sampling for ensemble size $N = 6$.

performed in different ocean DA applications with ensemble sizes between 20 and 46 (e.g. Goodliff et al., 2019; Tang et al., 2021). In a lock-exchange example, which is geometrically still simple but more realistic than the example here, Yu et al. (2025) have shown successful DA with ensemble size 8.

4.2 Assimilation results

Figure 7 shows the RMSE for the different experiments relative to the RMSE of the free-running ensembles (rRMSE). The experiments use the ESTKF and the LESTKF with all three ensemble initializations. Shown is the minimum error over all cases varying the forgetting factor and, for LESTKF, the localization radius.

The DA experiments for the cases ENS1 and ENS2 behave analogously, with ENS2 showing a more systematic decrease of the rRMSE for $N < 12$. In the following we focus on comparing ENS2 with ENS3. The effect of localization is particularly visible. As expected from the previous discussion on the chosen ensemble, the global ESTKF needs a relatively large ensemble to obtain a stronger reduction of the RMSE. Up to $N = 11$ the rRMSE is only reduced to about 90%. When the ensemble size is further increased, the RSME is strongly reduced. At $N = 15$, the relative error is reduced to 20%, and it is minimal at $N = 19$ with about 9%. With localization, the error is strongly reduced for smaller ensembles: Even for $N = 3$ the error is reduced to about 50%. For $N = 6$ the relative error is 37% for the LESTKF, while for the ESTKF it is 97%. The estimated state for the ESTKF and LESTKF for $N = 6$ is shown in Fig. 8a,b. While the ESTKF is not able to generate an estimate with a realistic pattern, the localization allows the LESTKF to obtain an estimate close to the correct pattern of the true state (see Fig. 5), but with a too high amplitude in comparison to the best estimate shown in Fig. 8d. For $N = 14$, the ESTKF shows a comparable error of 37% and the state in Fig. 8c indicates the correct pattern, but not the correct amplitude. The ESTKF and LESTKF obtain the minimum RMSE for ENS2 for the same ensemble size $N = 19$, which is due to the particular choice of

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

ensemble states. However, for small ensembles, the LESTKF computes local DA increments, e.g., using a localization radius of 10 grid points for $N = 6$, and the local ensemble is more representative than the global ensemble states. This enables the DA process to obtain estimates with the correct orientation even with ensemble states that do not include this orientation. This aspect of localization is consistent with the discussions in Kirchgessner et al. (2014) and Perianez et al. (2014). However, it is different from the more common discussion of using the localization to reduce spurious correlations due to sampling errors. Nonetheless, one can also consider correlations resulting from the incorrect orientation of the ensemble states as spurious correlations.

Comparing the previous cases with the experiments using ENS3 shown in Fig. 7c demonstrates the advantage of using the O(2)-sampling. The errors decrease faster as the ensemble size increases. The ESTKF with $N = 6$ shows already partly the correct pattern (see Fig. 8e), in particular in the left half of the domain. The relative error is lower with 70% compared to 97% for ENS2. The LESTKF with $N = 6$ and ENS3 (Fig. 8f) shows generally the correct pattern and a smoother field with smaller RMSE compared to using ENS2 in (Fig. 8b). The smoother field is obtained because a larger localization radius of 25 grid points yields the minimum RMSE in ENS3, compared to 10 grid points with ENS2. With ENS3, the minimum error is already obtained for $N = 10$ for the ESTKF, which is consistent with the expectation discussed in the previous section. The LESTKF shows again lower errors than the ESTKF for the same ensemble size for $N \leq 9$ and reaches the minimum error for $N \geq 8$. Figure 8h shows the optimal state estimate for the LESTKF with ENS3 for $N = 8$. The estimate for the ESTKF with $N = 8$ in Fig. 8g is almost identical, but has a slightly larger error. Overall, the experiments demonstrate that using O(2)-sampling allows for applying the DA with smaller ensembles than using model states as ensemble input.

For all ensemble initialization types, the minimum RMSE is obtained for ensembles of less than 20 states. This effect can be due to the fact that the number of observations is low, so that sampling errors, e.g., due to the random perturbations applied for generating the observations, are not negligible. More extensive studies would use repeated experiments with varying random numbers to obtain the mean effect. We omit such experiments here due to the illustrative aim of the experiments.

4.3 Code features and recommendations

While we omit a discussion of the actual source code, we point to specific aspects of the application example and include some recommendations for using such code with real models.

The example code illustrates the handling of multivariate state vectors as described in Sec. 3.9.1. The code uses a separate module (*statevector_pdaf_mod*) to initialize the arrays *id* and *sfields*. *id* holds the indices of both model fields in the state vector. *sfields* stores the dimension of each field, the number of dimensions, the field's name used in the program, and its name as used in file output. This information is used, e.g., to produce a list that provides an overview of each field in the state vector. It is also used in the file reading and writing functions, as it allows a generic code that loops over all fields. For clarity, the initialization of *id* and *sfields* is separated into two functions, which are adapted to include the fields of the specific model. Both functions are called by the general function for initializing the state vector.

Analogous to a real model, the example model uses a Fortran module (*model_mod*) to store variables like the model fields, grid dimensions, or coordinates. To make these variables accessible to the DA functions by the Fortran use-include approach,

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

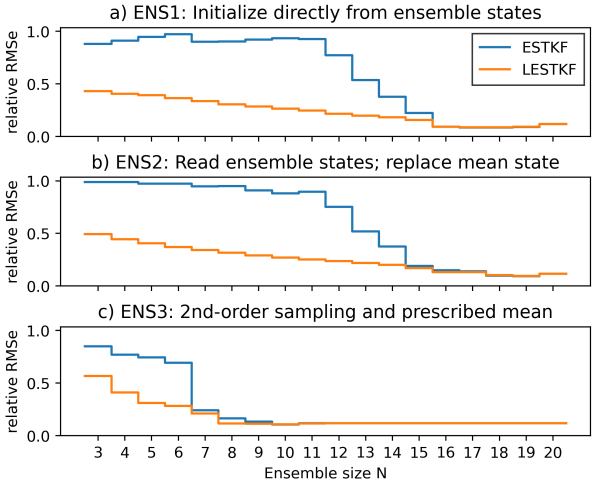

Figure 7. RMS errors for the ESTKF and LESTKF relative to the RMSE of the free running ensemble in dependence on ensemble size. Shown are the case of ensemble initialization using type ENS1: reading ensemble files, ENS2: reading ensemble files and replacing the mean state of the ensemble, and ENS3: using second-order exact sampling and the same mean state as ENS2.

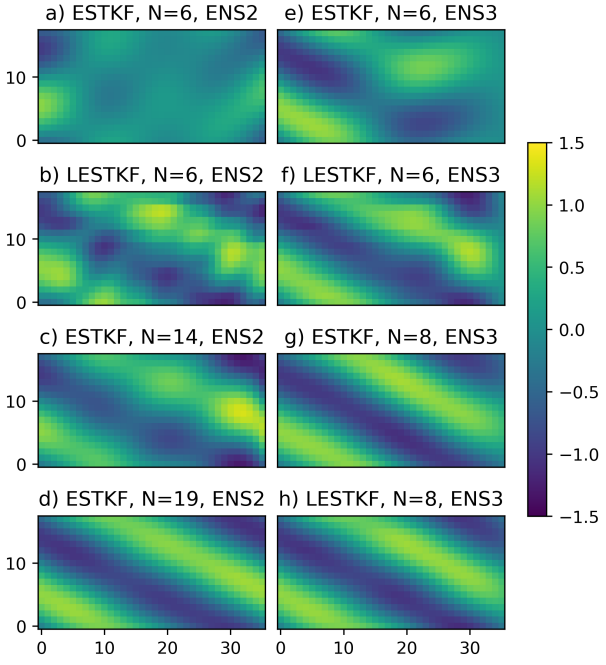

Figure 8. Examples of the state estimates at time step 20 for using the ensemble initialization type ENS2 (left) and ENS3 (right). The related RMS errors are displayed in Figure 7.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

the example code uses a single interfacing module (*model_pdaf_mod*). Thus, only this single module includes information directly from the model, while all other case-specific functions access the information by including this module. These functions use the variables independently of where they have been declared or initialized. This allows to use most of the functions for both online and offline coupled DA without changes. In the case of online coupling, *model_pdaf_mod* includes the variables directly from *model_mod*. The model source code, and hence *model_mod*, are not present in the case of offline coupling. In this case, one directly declares the variables in *model_pdaf_mod* and initializes them in the function *initialize_grid* (see Fig. 1a). The example code for the offline-coupled DA, uses 20 source code files, of which 12 files are identical to the files of the online coupled case. This approach allows for an easy transition between the offline and online coupled cases. Note, that the example code uses the model-specific variable names, e.g., *nx* for the number of grid points in x-direction. However, one could use *model_pdaf_mod* to rename the model variables into generic names, e.g., using *nx* always for the number of grid points in x, or longitudinal, direction. In this way, the functions using these variables could be applied also for other models independently of the specific name conventions of the model.

An important consideration for implementing the data assimilation for a particular model is the required effort. The example code is organized in 22 files for online coupling, excluding the model files, and 21 files for offline coupling. As described in Sec. 3.10.1, one can base a PDAF implementation on template code files. Ten of the template files are generic and can be used without changes, but need to be compiled with the other user code files. This is because they include the module *assimilation_pdaf_mod*, which declares configuration options and might be amended with case-specific variables, or because they perform file operations. For offline coupling, the main program file is generic, whereas for online coupling, the main file is the model's main program file. Some of the files provide optional features. For example, we included the functionality to generate files containing synthetic observations from a model run and to later use these. Also, using the command line parser in *init_pdaf_parse* is optional. Most source code files are short with less than 50 lines of code and most coding effort is in the observation modules. A more detailed description of the length of code in the example files is provided in Appendix B.

For complex models, a good computational performance can be obtained when computing the online-coupled ensemble DA using a separate directory for each model task (see discussion in Nerger et al., 2020). This approach allows the model tasks to write files into separate directories and hence to avoid possible conflicts in file writing. One can also directly use the model's restart files for ensemble restarting, since each model writes its restart files in a separate directory and reads them at restart time. To illustrate this approach, the example code contains a shell script for performing such a distributed DA run. The script generates the different ensemble directories, copies the online-coupled model into the directories, generates a suitable multi-program execution line, and then starts the parallel DA run.

5 Conclusions

This study describes the Parallel Data Assimilation Framework, PDAF, as of its recent revision 3.1, which includes the major advancements and code modernizations that were introduced when upgrading PDAF from the previous PDAF revision 2.3.1. Starting with the general concept, structure, and some historical context of PDAF's development, its components are described.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

PDAF provides an online-coupled mode with a framework for ensemble integrations and data assimilation (DA), enabling DA without avoidable overhead from model restarts. This approach can be implemented with minimal changes to a model code (usually by adding four function calls to interface functions) and enables the creation of an assimilative model with high computational performance. In contrast, PDAF's offline-coupled mode allows computing the DA in a separate program, even for models for which the source code is not available. These features enable the rapid development of a working DA program or assimilative model, allowing researchers to focus on applying DA. PDAF was also already coupled by the user community to a large range of models, and one can utilize existing couplings for DA applications.

PDAF provides a wide range of DA methods, including ensemble-based Kalman filters (EnKFs), nonlinear particle filters, and 3-dimensional variational (3D-Var) DA methods. The EnKFs include global and localized filters. For EnKFs, the different variants assimilate observations either serially (e.g., the EAKF method) or all at once (like the widely used LETKF and LESTKF methods and the original perturbed-observations ensemble Kalman filter). For 3D-Var methods, variants using parameterized covariance operators or ensembles are included. The wide range of DA methods makes PDAF usable to many applications. Because PDAF is not explicitly designed for Earth sciences, it can be used for data assimilation in any application where a dynamical model and observations over time are available.

With an internal interface to DA methods, PDAF also provides an environment for developing and implementing DA methods. These can leverage PDAF's framework functionality for observation handling, localization, and treatment of state vectors. A DA method implemented in PDAF can readily be applied to both toy models and real high-dimensional model setups without the need for recoding. This enables a quick transition from method development to real applications. Implementing a DA method in PDAF further allows developers of DA methods to make their developments readily available to a large user community through a PDAF release.

There are already many applications of PDAF in different models of Earth system components, as well as applications of coupled DA. This large user base also ensures an extensively tested code base in which many combinations of DA methods and hyper-parameters were applied.

While PDAF is a well-established framework with a long history, the framework is still under active ongoing development with about two code releases per year. A particular aspect is the addition of recent nonlinear DA methods using iterative flows or transport. Given recent developments in computer technology, support for graphics processing units (GPUs) will also become relevant. Another development will be the combination with machine-learning (ML) DA methods. Here, the linkage between the Fortran-coded PDAF library and ML codes in Python will become relevant, and pyPDAF can provide a basis for this. Using PDAF with ML-based models is straightforward with pyPDAF. As an open-source community framework, further community contributions, e.g., new DA methods, additional observation operators, covariance operators, and couplings to other models, are welcome contributions that both support the individual developers by reaching a large user community and the community by having access to state-of-the-art developments. As such PDAF is a tool to enable better and easier DA.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Code availability. The PDAF releases are available from the repository at https://github.com/PDAF/PDAF and also archived at Zenodo (https://doi.org/10.5281/zenodo.17016069, Nerger, 2025). The example code for Sec. 4 is available at https://github.com/PDAF/2dmodel_pdaf and also archived at Zenodo (https://doi.org/10.5281/zenodo.19709940, Nerger, 2026a). The Python scripts for generating Figures 5 to 8 are archived at Zenodo (https://doi.org/10.5281/zenodo.19809457, Nerger, 2026b).

Data availability. No data sets were used in this article.

Appendix A: Flexible-parallel mode

Section 3.1 discussed the fully-parallel implementation mode of PDAF. Here we explain the flexible-parallel mode. The flexible-parallel mode was the original parallelization variant of PDAF (Nerger et al., 2005b; Nerger and Hiller, 2013), but was revised for PDAF3. The flexible-parallel mode allows users to compute the ensemble forecast using fewer model tasks than ensemble states. It even allows users to use a single model task, i.e. the same number of processes as for a single model, to compute the full ensemble forecast. This implies that less processes are required than for the fully-parallel mode so that the online coupled DA can be computed on a smaller computer compared to the fully parallel mode. However, it will require more time to perform the computations. The flexible-parallel mode has been implemented, e.g., in Earth system modeling framework (Yu et al., 2025) and the PDAF release includes a template code demonstrating the implementation.

Figure A1 shows the program structure for the flexible-parallel mode. Compared to the fully parallel mode in Fig. 1, the flexible-parallel mode requires an additional control structure (purple in Fig. A1). An unconstrained loop is added that encloses the time stepping loop of the model. Inside this loop, the PDAF function *PDAF3_get_fcst_info* is called to retrieve the information on the number of time steps that have to be computed in the forecast phase (*isteps*), an integer flag indicating whether to exit the outer loop, and a time value indicating at which model time the forecast has to start. Note that in the flexible-parallel mode, the number of time steps in the model loop is the number of time steps in the forecast phase (*isteps*), while in the fully-parallel mode the overall number of time steps in the DA experiment (*nsteps*) is used. Apart from this number of time steps, both parallelization modes become equivalent if the flexible-parallel mode is executed with as many model tasks as there are ensemble states. In both parallelization modes, the function *PDAF3_assimilate* is called at each model time step so that it can, e.g., apply IAU.

Important for the flexible-parallel mode is that a model task has to step back in time when forecasting more than one ensemble state. For a model this is a nonphysical situation, because in reality time does not step back. Thus, one has to check a model code, whether such stepping back in time can be done consistently, and one might need to adapt the model code if this is not the case. For example, for ocean models the atmospheric forcing has to be handled consistently.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

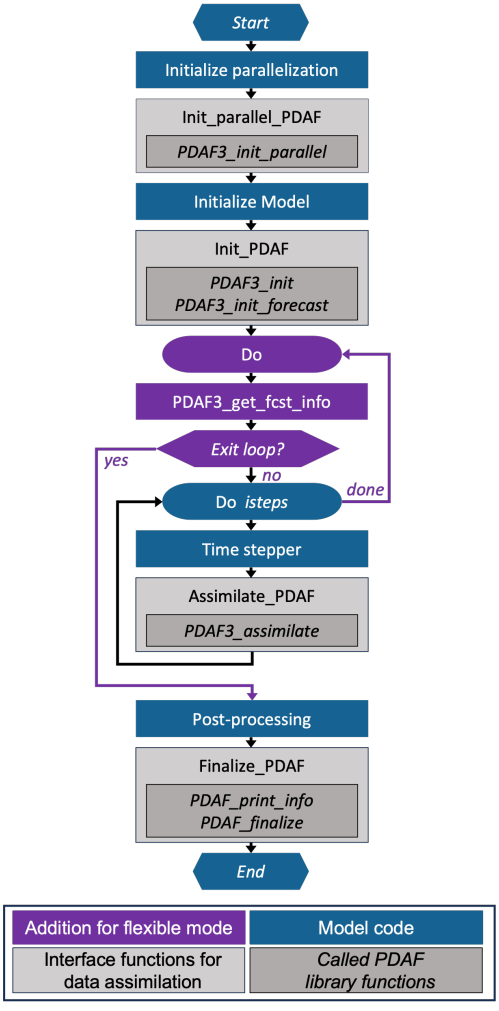

Figure A1. Structure of the online-coupled data assimilation program with PDAF's flexible-parallel mode. Compared to the fully-parallel mode (see Figure 1), there is an additional loop and control structure (purple). This allows each model task to compute multiple ensemble integrations.

Appendix B: Length of example codes

Here, we quantify the effort of implementing the DA in terms of the number of source code lines. Table B1 shows the count of functional code lines, i.e., source code without blank lines or comments, in the files of the online and online coupled example DA codes. The files holding the model source code are omitted. For the online coupling, we inserted 14 code lines into the model files (including preprocessor directives that allow us to compile the model without the coupling to PDAF; the actual calls to interface functions for PDAF are four lines).

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Table B1. Number of functional code lines in the files of the example codes for the online and offline coupled DA. The file names omit the suffix '.F90'. There are two columns for each coupling mode: The first shows the total number of lines, the second the count of lines specific to the example. The difference arises from the generic code in the templates. Italic font marks files in which the code is fully generic.

File *.F90	online		offline	
	total	specific	total	specific
Model interface functions and module				
init_parallel_pdaf	*27*	*0*	*18*	*0*
init_pdaf	88	22	81	22
assimilate_pdaf	*36*	*0*	*33*	*0*
finalize_pdaf	*10*	*0*	*identical to online*	
model_pdaf_mod	8	4	11	8
initialize_grid_mod	-	-	42	31
Call-back functions				
callback_obs_pdafomi	41	18	identical to online	
collect_state_pdaf	24	16	-	-
distribute_state_pdaf	24	16	-	-
get_obs_pdaf	*9*	*0*	*identical to online*	
init_dim_l_pdaf	26	10	identical to online	
init_ens_pdaf	*64*	*0*	*identical to online*	
init_n_domains_pdaf	8	3	identical to online	
next_observation_pdaf	24	9	-	-
prepoststep_pdaf	*64*	*0*	*identical to online*	
PDAF-OMI observation modules				
obs_A_pdafomi	140	59	identical to online	
obs_B_pdafomi	140	59	identical to online	
Other functions and modules				
main	*model-provided*		*26*	*0*
assim_pdaf_mod	*51*	*0*	*identical to online*	
init_pdaf_parse	108	16	identical to online	
io_pdaf_mod	234	170	identical to online	
parallel_pdaf_mod	*16*	*0*	*identical to online*	
statevector_pdaf_mod	77	13	identical to online	
synobs_pdaf_mod	*121*	*0*	*identical to online*	

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Most of the code is provided by the templates into which specific code for a particular model or observations is added. Ten of the provided files are generic and do not need changes when implementing the code for a particular model. Only the remaining 13 files for online coupling or 11 files for offline coupling need to be edited. The total number of specific code lines is 409 for offline and 415 for online-coupled DA, respectively. However, eight of the non-generic files with a total of 348 specific code lines are identical for both coupling cases. Only 61 lines of code had to be implemented specifically for the offline case, and 67 lines for the online case. Thus, when one has implemented one of the coupling cases, the transition to the other coupling approach can be performed with very little coding effort. The difference is mainly that the offline case requires *initialize_grid_mod* to initialize the model grid information, whereas the online case requires transferring between model fields and the state vector (*collect_state_pdaf* and *distribute_state_pdaf*) and specifying the number of time steps in the forecast phase (*next_observation_pdaf*). While the templates for *init_PDAF* are different for the online and offline coupling, the added specific code lines are identical.

The code in the different files is usually short: Only 5 of the files have more than 100 total lines of code, and only one file has more than 100 lines of specific code. Two files have 59 lines of specific code while all other files have specific code of 22 or less lines. The callback functions have at most 64 code lines with at most 18 lines of specific code. The longest file (*io_pdaf_mod.F90*, 234 lines) holds the functions to perform the file reading and writing including reading from the covariance matrix file. While this code was newly implemented for the example, it can now also serve as a template and is generally usable for any rectangular 2-dimensional model domain with the domain decomposition used in the example. The specific code in (*io_pdaf_mod.F90* are 170 lines, which is nearly half of the overall specific code in the example. Also, the two observation modules are longer files, each with 56 lines of specific code. The 22 lines of specific code in *init_pdaf.F90* and *init_pdaf_offline.F90* are the specification of default values for some of the observation-related variables and the initialization of coordinate vectors for using localization in the serial-observation processing filters. The file *init_pdaf_parse.F90*, with 16 specific code lines, contains command-line parsing functionality, e.g. to choose at run-time which observations are assimilated. With a more realistic model, one might want to replace the command-line reading with configuration functionality compatible with the model, e.g., Fortran namelists, which are often used by models of Earth system components.

Author contributions. LN coded PDAF and the example model. YC, AC, and JK contributed functionality and bug fixes to PDAF, as well as additions to its documentation. YC and AC also provided inputs to the example model. LN wrote the initial version the paper; all authors contributed to its structure and to revising it.

Competing interests. The contact author has declared that none of the authors has any competing interests.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Acknowledgements. YC is supported by the UK Natural Environment Research Council's support for the National Centre for Earth Observation (Contract Number PR140015). AC is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – FOR 5405: Magnetosphere, Ionosphere, Plasmasphere and Thermosphere, as a coupled system (MIPT) – project number 462853228.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

References

Anderson, J., Hoar, T., Raeder, K., Liu, H., Collins, N., Torn, R., and Arellano, A.: The Data Assimilation Research Testbed: A Community Facility, Bull. Am. Meteorol. Soc., 90, 1283–1296, 2009.

Anderson, J. L.: A Local Least Squares Framework for Ensemble Filtering, Mon. Wea. Rev., 131, 634–642, 2003.

Androsov, A., Nerger, L., Schnur, R., Schröter, J., Albertella, A., Rummel, R., Savcenko, R., Bosch, W., Skachko, S., and Danilov, S.: On the assimilation of absolute geodetic dynamics topography in a global ocean model: Impact on the deep ocean state, J. Geodesy, 93, 141–157, 2019.

Baatz, R., Hendricks Franssen, H. J., Euskirchen, E., Sihi, D., Dietze, M., Ciavatta, S., Fennel, K., Beck, H., De Lannoy, G., Pauwels, V. R. N., Raiho, A., Montzka, C., Williams, M., Mishra, U., Poppe, C., Zacharias, S., Lausch, A., Samaniego, L., Van Looy, K., Bogena, H., Adamescu, M., Mirtl, M., Fox, A., Goergen, K., Naz, B. S., Zeng, Y., and Vereecken, H.: Reanalysis in Earth System Science: Toward Terrestrial Ecosystem Reanalysis, Reviews of Geophysics, 59, e2020RG000 715, 2021.

Bakhshaei, K., Salavatidezfouli, S., Stabile, G., and Rozza, G.: Stochastic parameter prediction in cardiovascular problems, Computer Methods in Biomechanics and Biomedical Engineering, pp. 1–25, 2025.

Bannister, R. N.: A rview of operational methods of variational and ensemble-variational data assimilation, Q. J. R. Meteorol. Soc., 143, 607–633, 2017.

Barker, D., Huang, X.-Y., Liu, Z., Auligne, T., Zhang, X., Rugg, S., Ajjaji, R., Bourgeois, A., Bray, J., Chen, Y., Demirtas, M., Guo, Y.-R., Henderson, T., Huang, W., Lin, H.-C., Michalakes, J., Rizvi, S., and Zhang, X.: The Weather Research and Forecasting Model's Community Variational/Ensemble Data Assimilation System: WRFDA, Bulletin of the American Meteorological Society, 93, 831–843, 2012.

Ben Ali, M., Leon, O., Donjat, D., Bezard, H., Laroche, E., Mons, V., and Champagnat, F.: Data assimilation for aerothermal mean flow reconstruction using aero-optical observations: a synthetic investigation, in: 56th 3AF International Conference on Applied Aerodynamics, 28 — 30 March 2022, Toulouse – France, 2022.

Bishop, C. H., Etherton, B. J., and Majumdar, S. J.: Adaptive Sampling with the Ensemble Transform Kalman Filter. Part I: Theoretical Aspects, Mon. Wea. Rev., 129, 420–436, 2001.

Bloom, S. C., Takacs, L. L., A, M. D. S., and Ledvina, D.: Data assimilation using incremental analysis updates, Mon. Wea. Rev., 124, 1256–1271, 1996.

Bruening, T., Li, X., and ad I. Lorkowski, F. S.: An operational, assimilative model system for hydrodynamic and biogoechemical application for German coastal waters, Hydrographische Nachrichten, 118, 6–15, 2021.

Bruggeman, J., Bolding, K., Nerger, L., Teruzzi, A., Spada, S., Skakala, J., and Ciavatta, S.: EAT v1.0.0: a 1D test bed for physical-biogeochemical data assimialtion in natural waters, Geosci. Model Dev., 17, 5619–5639, 2024.

Brune, S., Nerger, L., and Baehr, J.: Assimilation of oceanic observations in a global coupled Earth system model with the SEIK filter, Oce. Mod., 96, 254–264, 2015.

Buehner, M., Caron, J.-F., Lapalme, E., Caya, A., Du, P., Rochon, Y., Skachko, S., Bani Shahabadi, M., Heilliette, S., Deshaies-Jacques, M., Chang, W., and Sitwell, M.: The Modular and Integrated Data Assimilation System at Environment and Climate Change Canada (MIDAS v3.9.1), Geoscientific Model Development, 18, 1–18, https://doi.org/10.5194/gmd-18-1-2025, 2025.

Burgers, G., van Leeuwen, P. J., and Evensen, G.: On the Analysis Scheme in the Ensemble Kalman Filter, Mon. Wea. Rev., 126, 1719–1724, 1998.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Chen, Y., Nerger, L., and Lawless, A. S.: A Python interface to the Fortran-based parallel data assimilation framework: pyPDAF v1.0.2, Geosci. Model Dev., 18, 8235–8252, 2025.

Cirano, M., Alvarez-Fanjul, E., Capet, A., Ciliberti, S., Clementi, E., Dewitte, B., Dinápoli, M., Serafy, G. E., Hogan, P., Joseph, S., Miyazawa, Y., Montes, I., Narvaez, D. A., Regan, H., Simionato, C. G., Smith, G. C., Staneva, J., Tanajura, C. A. S., Thupaki, P., Urbano-Latorre, C., Veitch, J., and Hidalgo, J. Z.: A description of existing operational ocean forecasting services around the globe, State of the Planet, 5-opsr, 5, 2025.

Cook, S., Gillet-Chaulet, F., and Fuerst, J.: Robust reconstruction of glacier beds using transient 2D assimilation with Stokes, J. Glacioloy, 69, 1393–1402, 2023.

Corbin, A. and Kusche, J.: Improving the estimation of thermospheric neutral density via two-step assimilation of in situ neutral density into a numerical model, Eaeth, Planeets and Space, 74, 183, 2022.

Drevillon, M., Chassignet, E., Traon, P.-Y. L., Bayler, E., Davidson, F., Smith, G., Vinayachandran, P. N., Schiller, A., Fanjul, E. A., Wilmer-Becker, K., Akella, S., Cuven, S., Bahurel, P., Brassington, G., Byun, D.-S., Calewaert, J.-B., Campuzano, F., Cho, Y.-K., Choi, B.-J., Christensen, K. H., Ciavatta, S., Ciliberti, S., Clementi, E., Conchon, A., Cossarini, G., Crespin, J., Cucurull, L., Mey-Frémaux, P. D., Derval, C., Drillet, Y., Entel, M., Faugère, Y., Federico, I., Fennel, K., Fujii, Y., amd Stephanie Guinehut, G. G., Harris, C., Hasson, A., Hernandez, F., Heslop, E., Hill, K., Horemans, D., Juza, M., Kerry, C., Kourafalou, V., Kurapov, A., Lellouche, J.-M., Levier, B., Liu, Y., Liu, G., Makrygianni, N., Martin, M., Balbontin, G. M., Masina, S., Mogensen, K., Moore, A., Nerger, L., Paquin, J.-P., Paul, A., Pavanathara, F., Pearlman, J., Pelletier, C., Post, J., Buil, M. P., Regan, H., Remy, E., Remya, P. G., Ruggiero, G., Samuelsen, A., Skakala, J., Sperrevik, A. K., Staneva, J., Storto, A., Tanajura, C. A. S., Teruzzi, A., Thur, S., Tonani, M., Veitch, J., , and Yu, L.: OceanPredict'24 Symposium: advancing ocean prediction science for societal benefit, Bulletin of the American Meteorological Society, 106, E2479–E2489, 2025.

Düsterhus, A. and Brune, S.: Decadal Predictability of Seasonal Temperature Distributions, Geophys. Res. Lett., 51, e2023GL107 838, 2024.

Eicker, A., Schumacher, M., Kusche, J., Döll, P., and Müller Schmied, H.: Calibration/Data Assimilation Approach for Integrating GRACE Data into the WaterGAP Global Hydrology Model (WGHM) Using an Ensemble Kalman Filter: First Results, Surveys in Geophysics, 35, 1285–1309, 2014.

Evensen, G.: Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, J. Geophys. Res., 99, 10 143–10 162, 1994.

Fournier, A., Nerger, L., and Aubert, J.: An ensemble Kalman filter for the time-dependent analysis of the geomagnetic field, Geochemistry Geophysics Geosystems, 14, 4035–4043, 2013.

Friedemann, S. and Raffin, B.: An elastic framework for ensemble-based large-scale data assimilation, International Journal of High Performance Computing Applications, 36, 543–563, 2022.

Gaspari, G. and Cohn, S. E.: Construction of Correlation Functions in Two and Three Dimensions, Q. J. Roy. Meteor. Soc., 125, 723–757, 1999.

Gerdener, H., Kusche, J., Schulze, K., Döll, P., and Klos, A.: The Global Land Water Storage Data Set Release 2 (GLWS2.0) Derived via Assimilating GRACE and GRACE-FO Data into a Global Hydrological Model, Journal of Geodesy, 97, 73, https://doi.org/10.1007/s00190-023-01763-9, 2023.

Gillet-Chaulet, F.: Assimilation of surface observations in a transient marine ice sheet model using an ensemble Kalman filter, The Cryosphere, 14, 811–832, 2020.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Goodliff, M., Bruening, T., Schwichtenberg, F., Li, X., Lindenthal, A., Lorkowski, I., and Nerger, L.: Temperature assimilation into a coastal ocean-biogeochemical model: Assessment of weakly- and strongly-coupled data assimilation, Oce. Dyn., 69, 1217–1237, 2019.

Hamill, T. M., Whitaker, J. S., and Snyder, C.: Distance-Dependent Filtering of Background Error Covariance Estimates in an Ensemble Kalman Filter, Mon. Wea. Rev., 129, 2776–1790, 2001.

Hersbach, H.: Decomposition of the Continuous Ranked Probability Score for ensemble prediction systems, Weather and Forecasting, 15, 559–570, 2000.

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, Q. J. R. Meteorol. Soc., 146, 1999–2049, 2020.

Hong, Y., Ma, Y., Wu, Y., Sun, S., Wen, S., and Sun, Z.: Rapid non-invasive measurement of the time-dependent heat flux and temperature distribution in participating medium under non-Gaussian noise, International Journal of Thermal Sciences, 217, 110 112, 2025.

Hu, C.-C. and van Leeuwen, P. J.: A particle flow filter for high-dimensional system applications, Q. J. R. Meteorol. Soc., 147, 2352–2374, 2021.

Hunt, B. R., Kostelich, E. J., and Szunyogh, I.: Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter, Physica D, 230, 112–126, 2007.

Janjić, T., Nerger, L., Albertella, A., Schröter, J., and Skachko, S.: On domain localization in ensemble based Kalman filter algorithms, Mon. Wea. Rev., 139, 2046–2060, 2011.

Kirchgessner, P., Nerger, L., and Bunse-Gerstner, A.: On the Choice of an Optimal Localization Radius in Ensemble Kalman Filter Methods, Mon. Wea. Rev., 142, 2165–2175, 2014.

Kirchgessner, P., Toedter, J., Ahrens, B., and Nerger, L.: The smoother extension of the nonlinear ensemble transform filter, Tellus A, 69, 1327 766, 2017.

Kurtz, W., He, G., Kollet, S. J., Maxwell, R. M., Vereecken, H., and Franssen, H.-J. H.: TerrSysMP-PDAF (version 1.0): a modular high-performance data assimilation framework for an integrated land surface-subsurface model, Geosci. Model. Dev., 9, 1341–1360, 2016.

Lee, I. T., Matsuo, T., Richmond, A. D., Liu, J. Y., Wang, W., Lin, C. H., Anderson, J. L., and Chen, M. Q.: Assimilation of FORMOSAT-3/COSMIC Electron Density Profiles into a Coupled Thermosphere/Ionosphere Model Using Ensemble Kalman Filtering, Journal of Geophysical Research: Space Physics, 117, https://doi.org/10.1029/2012JA017700, 2012.

Li, F., Bogena, H. R., Keller, J., Bayat, B., Raj, R., and Hendricks-Franssen, H.-J.: A new approach for joint assimilation of cosmic-ray neutron soil moisture and groundwater level data into an integrated terrestrial model, Hydrology and Earth System Sciences, 29, 6419–6443, 2025.

Li, H., Yang, T., Nerger, L., Zhang, D., Zhang, D., Tang, G., Wang, H., Sun, Y., Fu, P., Su, H., and Wang, Z.: NAQPMS-PDAF v2.0: a novel hybrid nonlinear data assimilation system for improved simulation of PM2.5 chemical components, Geosci. Model Dev., 17, 8495–8519, 2024.

Liang, X., Zhao, F., Li, C., Zhang, L., and Bingrui, L.: Evaluation of ArcIOPS sea ice forecasting products during the ninth CHINARE-Arctic in summer 2018, Adv. Polar Science, 31, 14–25, 2020.

Liu, Z., Snyder, C., Guerrette, J. J., Jung, B.-J., Ban, J., Vahl, S., Wu, Y., Trémolet, Y., Auligné, T., Ménétrier, B., Shlyaeva, A., Herbener, S., Liu, E., Holdaway, D., and Johnson, B. T.: Data assimilation for the Model for Prediction Across Scales – Atmosphere with the Joint

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Effort for Data assimilation Integration (JEDI-MPAS 1.0.0): EnVar implementation and evaluation, Geosci. Model Dev., 15, 7859–7878, 2022.

Lorenz, E. N.: Deterministic nonperiodic flow, J. Atmos. Sci., 20, 131–141, 1963.

Lorenz, E. N.: Predictability - A problem partly solved, in: Proceedings Seminar on Predictability, pp. 1–18, ECMWF, Reading, UK, 1996.

Lorenz, E. N.: Designing Chaotic Models, J. Atm. Sci., 62, 1574–1587, 2005.

Losa, S. N., Danilov, S., Schröter, J., Janjić, T., Nerger, L., and Janssen, F.: Assimilating NOAA SST data into BSH operational circulation model for the North and Baltic Seas: Part 2. Sensitivity of the forecast's skill to the prior model error statistics, J. Mar. Syst., 129, 259–270, 2014.

Mamnun, N., Voelker, C., Vrekoussis, M., and Nerger, L.: Spatially Varying Biogeochemical Parameter Estimation in a Global Ocean Model, J. Geophys. Res. Oceans, 130, e2025JC022 752, 2025.

Masoum, A., Nerger, L., Willeit, M., Gnopolski, A., and Lohmann, G.: Paleoclimate data assimilation with CLIMBER- X: An ensemble Kalman filter for the last deglaciation, PLOS ONE, 19, e0300 138, 2024.

Message Passing Interface Forum: MPI: A Message-Passing Interface Standard Version 5.0, https://www.mpi-forum.org/docs/mpi-5.0/mpi50-report.pdf, 2025.

Mirouze, I., Blockley, E. Q., Lea, D. J., Martin, M. J., and Bell, M. J.: A multiple length scale correlation operator for ocean data assimilation, Tellus A, 68, 29 744, 2016.

Moore, A., Arango, H., Broquet, G., Powell, B. S., Weaver, A., and Zavala-Garay, J.: The Regional Ocean Modeling System (ROMS) 4-dimensional variational data assimilations systems, Part I - System overview and formulation, Prog. Ocean., 91, 34–49, 2011.

Nerger, L.: Data assimilation for nonlinear systems with a hybrid nonlinear Kalman ensemble transform filter, Q. J. R. Meteorol. Soc., 148, 620–640, 2022.

Nerger, L.: PDAF (Parallel Data Assimilation Framwork), Zenodo, https://doi.org/10.5281/zenodo.17016069, 2025.

Nerger, L.: PDAF/2dmodel_pdaf, Zenodo, https://doi.org/10.5281/zenodo.19709940, 2026a.

Nerger, L.: Plot scripts for manuscript on PDAF 3.1 in GMD, Zenodo, https://doi.org/10.5281/zenodo.19809457, 2026b.

Nerger, L. and Gregg, W. W.: Assimilation of SeaWiFS data into a global ocean-biogeochemical model using a local SEIK filter, J. Mar. Syst., 68, 237–254, 2007.

Nerger, L. and Hiller, W.: Software for Ensemble-based Data Assimilation Systems - Implementation Strategies and Scalability, Computers & Geosciences, 55, 110–118, 2013.

Nerger, L., Hiller, W., and Schröter, J.: A Comparison of Error Subspace Kalman Filters, Tellus, 57A, 715–735, 2005a.

Nerger, L., Hiller, W., and Schröter, J.: PDAF - The Parallel Data Assimilation Framework: Experiences with Kalman filtering., in: Use of High Performance Computing in Meteorology - Proceedings of the 11. ECMWF Workshop, edited by Zwieflhofer, W. and Mozdzynski, G., pp. 63–83, World Scientific, 2005b.

Nerger, L., Danilov, S., Hiller, W., and Schröter, J.: Using sea level data to constrain a finite-element primitive-equation ocean model with a local SEIK filter, Ocean Dynamics, 56, 634–649, 2006.

Nerger, L., Danilov, S., Kivman, G., Hiller, W., and Schröter, J.: Data Assimilation with the Ensemble Kalman Filter and the SEIK Filter applied to a Finite Element Model of the North Atlantic, J. Mar. Syst., 65, 288–298, 2007.

Nerger, L., Janjić, T., Schröter, J., and Hiller, W.: A regulated localization scheme for ensemble-based Kalman filters, Q. J. Roy. Meteor. Soc., 138, 802–812, 2012a.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Nerger, L., Janjić, T., Schröter, J., and Hiller, W.: A unification of ensemble square root Kalman filters, Mon. Wea. Rev., 140, 2335–2345, 2012b.

Nerger, L., Schulte, S., and Bunse-Gerstner, A.: On the influence of model nonlinearity and localization on ensemble Kalman smoothing, Q. J. Roy. Meteor. Soc., 140, 2249–2259, 2014.

Nerger, L., Tang, Q., and Mu, L.: Efficient ensemble data assimilation for coupled models with the Parallel Data Assimilation Framework: example of AWI-CM (AWI-CM-PDAF 1.0), Geosci. Model Dev., 13, 4305–4321, 2020.

OpenMP Architecture Review Board: OpenMP Application Programming Interface, https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-6-0.pdf, 2024.

Pardini, F., Corradini, S., Costa, A., Ongari, T. E., Merucci, L., Neri, A., Stelitano, D., and deḾichieli Vitturi, M.: Ensemble-Based Data Assimilation of Volcanic Ash Clouds from Satellite Observations: Application to the 24 December 2018 Mt. Etna Explosive Eruption, Atmosphere, 11, 359, 2020.

PDAF: The Parallel Data Assimilation Framework, https://pdaf.awi.de, 2026.

Perianez, A., Reich, H., and Potthast, R.: Optimal localization for ensemble Kalman filter systems, J. Meteorol. Soc. Jpn., 92, 585–597, 2014.

Pham, D. T.: Stochastic Methods for Sequential Data Assimilation in Strongly Nonlinear Systems, Mon. Wea. Rev., 129, 1194–1207, 2001.

Polkova, I., Brune, S., Kadow, C., Romanova, V., Gollan, G., Baehr, J., Glowienka-Hense, R., Greatbatch, R. J., Hense, A., Köhl, A., Kröger, J., Müller, W. A., Pankatz, K., and Stammer, D.: Initialization and ensemble generation for decadal climate predictions: a comparison of different methods, J. Adv. Model. Earth Syst., 11, 149–172, 2019.

Pradhan, H. K., Voelker, C., Losa, S. N., Bracher, A., and Nerger, L.: Assimilation of global total chlorophyll OC-CCI data and its impact on individual phytoplankton fields, J. Geophys. Res. Oceans, 124, 470–490, 2019.

Pradhan, H. K., Voelker, C., Losa, S. N., Bracher, A., and Nerger, L.: Global assimilation of ocean-color data of phytoplankton functional types: Impact of different data sets, J. Geophys. Res. Oceans, 125, e2019JC015 586, 2020.

Prive, N. C., MacGrath-Spangler, E. L., Carvalho, D., Karpovicz, B. M., and I, . M.: Robustness of Observing system simulation experiments, Tellus A, 75, 309–333, 2023.

Pulido, M. and van Leeuwen, P. J.: Sequential Monte Carlo with kernel embeddeed mappings: The mapping particle filter, J. Comp. Phys., 396, 400–415, 2019.

Rodwell, M. J., Lang, S. T. K., Ingleby, N. B., Bormann, N., Holm, E., Rabier, F., Richardson, D. S., and Yamaguchi, M.: Reliability in ensemble data assimilation, Q. J. R. Meteorol. Soc., 142, 443–454, 2016.

Sakov, P.: EnKF-C user guide, Tech. Rep. doi:10.48550/arXiv.1410.1233, Bureau of Meteorology, 2014.

Sanchez, S., Wicht, J., and Bärenzung, J.: Predictions of the geomagnetic secular variation based on the ensemble sequential assimilation of geomagnetic field models by dynamo simulations, Earth, Planets, and Space, 72, 157, 2020.

Schachtschneider, R., Saynisch-Wagner, J., Klemann, V., Bagge, M., and Thomas, M.: An approach for constraining mantle viscosities through assimilation of palaeo sea level data into a glacial isostatic adjustment model, Nonlin. Proc. Geophys., 29, 53–75, 2022.

Shao, C. and Nerger, L.: WRF-PDAF v1.0: implementation and application of an online localized ensemble data assimilation framework, Geosci. Model Dev., 17, 4433–4445, 2024.

Storto, A. and Andriopoulos, P.: A new stochastic ocean physics package and its application to hybrid-covariance data assimilation, Q. J. R. Meteorol. Soc., 147, 169101 725, 2021.

Storto, A., Alvera-Azcárate, A., Balmaseda, M. A., Barth, A., Chevallier, M., Counillon, F., Domingues, C. M., Drevillon, M., Drillet, Y., Forget, G., Garric, G., Haines, K., Hernandez, F., Iovino, D., Jackson, L. C., Lellouche, J.-M., Masina, S., Mayer, M., Oke, P. R., Penny,

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

S. G., Peterson, K. A., Yang, C., and Zuo, H.: Ocean Reanalyses: Recent Advances and Unsolved Challenges, Frontiers in Marine Science, 6, 418, 2019.

Strebel, L., Bogena, H. R., Vereecken, H., and Hendricks Franssen, H.-J.: Coupling the Community Land Model version 5.0 to the parallel data assimilation framework PDAF: description and applications, Geoscientific Model Development, 15, 395–411, https://doi.org/10.5194/gmd-15-395-2022, 2022.

Sun, C., Liu, L., Li, R., Yu, X., Yu, H., Zhao, B., Wang, G., Liu, J., Qiao, F., and Wang, B.: Developing a common, flexible and efficient framework for weakly coupled ensemble data assimilation based on C-Coupler2.0, Geosci. Model Dev., 14, 2635–2657, 2021.

Tang, Q., Mu, L., Sidorenko, D., Goessling, H., Semmler, T., and Nerger, L.: Improving the ocean and atmosphere in a coupled ocean-atmosphere model by assimilating satellite sea surface temperature and subsurface profile data, Q. J. R. Meteorol. Soc., 146, 4014–4029, 2020.

Tang, Q., Mu, L., Goessling, H. F., Semmler, T., and Nerger, L.: Strongly coupled data assimilation of ocean observations into an ocean-atmosphere model, Geophys. Res. Lett., 48, e2021GL094 941, 2021.

Tang, Q., DElottier, H., Kurtz, W., Nerger, L., Schilling, O. S., and Brunner, P.: HGS-PDAF (version 1.0): a modular data assimilation framework for an integrated surface and subsurface hydrological model, Geosci. Model Dev., 17, 3559–3578, 2024.

Taylor, K. E.: Summarizing multiple aspects of model performance in a single diagram, J. Geophys. Res., 106(D7), 7183–7192, 2004.

Tödter, J.: Derivation and Characterization of a new filter for nonlinear high-dimensional data assimilation, Ph.D. thesis, University of Frankfurt, http://d-nb.info/1074192494/34, 2015.

Tödter, J. and Ahrens, B.: A second-order exact ensemble square root filter for nonlinear data assimilation, Mon. Wea. Rev., 143, 1347–1367, 2015.

Tödter, J., Kirchgessner, P., Nerger, L., and Ahrens, B.: Assessment of a nonlinear ensemble transform filter for high-dimensional data assimilation, Mon. Wea. Rev., 144, 409–427, 2016.

Tuomi, L., She, J., Lorkowski, I., Axell, L., Lagemaa, P., Schwichtenberg, F., and Huess, V.: Overview of CMEMS BAL MFC Service and Developments, in: Proceedings of the Eigth EuroGOOS International Conference, 3–5 October 2017, Bergen, pp. 261–268, 2018.

Valmassoi, A., Keller, J. D., Kleist, D. T., English, S., Ahrens, B., Ďurán, I. B., Bauernschubert, E., Bosilovich, M. G., Fujiwara, M., Hersbach, H., Lei, L., Löhnert, U., Mamnun, N., Martin, C. R., Moore, A., Niermann, D., Ruiz, J. J., , and Scheck, L.: Current Challenges and Future Directions in Data Assimilation and Reanalysis, Bull. Am. Meteorol. Soc., 104, E756–E767, 2023.

van Leeuwen, P. J., Künsch, H. R., Nerger, L., Potthast, R., and Reich, S.: Particle filters for high-dimensional geoscience applications: a review, Q. J. R. Meteorol. Soc., 145, 2335–2365, 2019.

Vetra-Carvalho, S., van Leeuwen, P. J., Nerger, L., Barth, A., Altaf, M. U., Brasseur, P., Kirchgessner, P., and Beckers, J.-M.: State-of-the-art stochastic data assimilation methods for high-dimensional non-Gaussian problems, Tellus A, 70, 1445 364, 2018.

Weaver, A. T., Deltel, C., Machu, E., Ricci, S., and Daget, N.: A multivariate balance operator for variational ocean data assimilation, Q. J. R. Meteorol. Soc., 131, 3605–3625, 2005.

Weaver, A. T., Tshimanga, J., and Piacentini, A.: Correlation operators based on an implicitly formulated diffusion equation solved with the Chebyshev iteratio, Q. J. R. Meteorol. Soc., 142, 455–471, 2016.

Whitaker, J. S. and Hamill, T. M.: Ensemble Data Assimilation without Perturbed Observations, Mon. Wea. Rev., 130, 1913–1927, 2002.

Yang, Q., Losa, S. N., Losch, M., Liu, J., Zhang, Z., Nerger, L., and Yang, H.: Assimiliating summer sea-ice concentration into a coupled ice-ocean model using a LSEIK filter, Ann. Glaciology, 56, 38–44, 2015.

Ying, Y. M.: NEDAS, Zenodo, 2025.

https://doi.org/10.5194/egusphere-2026-2412
Preprint. Discussion started: 20 May 2026

Yu, H.-C., Zhang, Y. J., Nerger, L., Lemmen, C., Yu, J. C., Chou, T.-Y., Chu, C.-H., and Terng, C.-T.: Development of a flexible data assimilation system for a 3D unstructured-grid ocean model under Earth System Modeling Framework, Ocean Modeling, 196, 102 546, 2025.

Zhang, F., Meng, Z., and Aksoy, A.: Tests of an Ensemble Kalman Filter for Mesoscale and Regional-Scale Data Assimilation. Part I: Perfect Model Experiments, Mon. Wea. Rev., 134, 722–736, 2006.

Zhao, H., Montzka, C., Keller, J., Li, F., Vereecken, H., and Hendricks Franssen, H.-J.: How Does Assimilating SMAP Soil Moisture Improve Characterization of the Terrestrial Water Cycle in an Integrated Land Surface-Subsurface Model?, Water Resources Research, 61, e2024WR038 647, 2025.

Zupanski, D. and Zupanski, M.: Model Error Estimation Employing an Ensemble Data Assimilation Approach, Mon. Wea. Rev., 134, 1337–1354, 2006.