https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

Versatile Earth-systems COupleR (VerCOR v.0.4.6): Quick and easy coupling and prototyping of Earth system model components in Python and JAX.

Roman Nuterman[1], Isak Rockström[1], Vivek Veluvali[1], Etienne Meunier[2,1], Aster Lei Stoustrup[1], and Markus Jochum[1]

[1]University of Copenhagen, Niels Bohr Institute, Jagtvej 155 A, 2200 Copenhagen, Denmark
[2]National Institute for Research in Digital Science and Technology, 48 Rue Barrault, 75013 Paris, France

Correspondence: Roman Nuterman (nuterman@nbi.ku.dk)

Abstract. To speed up model development, a JAX-based library is developed that allows uncomplicated coupling of Earth system model components with each other and with data models. It consists of two components: a simple but fully differentiable coupler that organizes the data flow, and a Bayesian optimizer that automatically tunes the desired model configuration. The coupler is described in detail, and the workflow is illustrated by coupling an ocean to an atmosphere model and tuning the result. There is also a demonstration provided that this coupled model is fully differentiable, and gradient based methods can be used for parameter optimization as well. This technique, however, is still in its infancy, and we hope that with the present library the community is enabled to determine how to best combine gradient-based and gradient-free optimization methods.

1 Introduction

The first coupled ocean-atmosphere model was built in 1969 (Manabe and Bryan, 1969), and since then general circulation models have become increasingly sophisticated and realistic. Today's Earth System Models (ESMs) not only capture the physical climate system, but also simulate terrestrial and marine biogeochemistry (reference). Over time, many ESM biases have been reduced by increasing resolution (e.g., Jochum et al., 2012), or improving parameterizations (e.g., Gent et al., 1995). Some, however, appear to be difficult to remove (e.g., tropical precipitation, Segura et al., 2025). To overcome this impasse, experimentation needs to become faster and cheaper. To this end, we developed the Versatile Earth System COupleR (VerCOR v0.4.6) as a platform for coupled JAX based VERsatile Earth Simulators (VERES). What follows is a detailed description of the coupler (Section 2) and two applications: the ocean model Veros is coupled to the atmosphere model JCM and the resulting model is calibrated with the Bayesian optimizer VerOpt (Stoustrup et al. 2026, Section 3). This section will also explicitly motivate the calibration targets and final parameter choices. Section 4 explores the potential of auto-differentiability as a second method to calibrate model parameters in the coupled framework. We target two parameters of the ocean mixed layer model, and an indentical-twin experiment shows that the autodiff functionality of JAX does indeed recover the proper parameter values. Roll-outs of more than 10 days currently pose technical challenges that we describe and quantify. Section 5 then provides summary and outlook.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

2 Versatile Earth System Coupler

2.1 Software design and implementation

The Versatile Earth System COupleR (VerCOR) v0.4.6 is a Python library developed for constructing coupled Earth System Models (ESMs) from both prognostic and data-based Earth system components. VerCOR provides the infrastructure to define component interfaces, facilitate field transfers between component grids, and advance components in time. VerCOR is built around the popular high-performance JAX library, which supports just-in-time (JIT) compilation, hardware-agnostic execution on CPUs, GPUs, and TPU devices, and automatic differentiation (Bradbury et al., 2018). Hence, an ESM configuration built entirely from JAX-based components can both simulate the evolution of the Earth system and be treated as a differentiable numerical program. Although VerCOR is optimized for JAX-based components, it also supports non-JAX components within the same library.

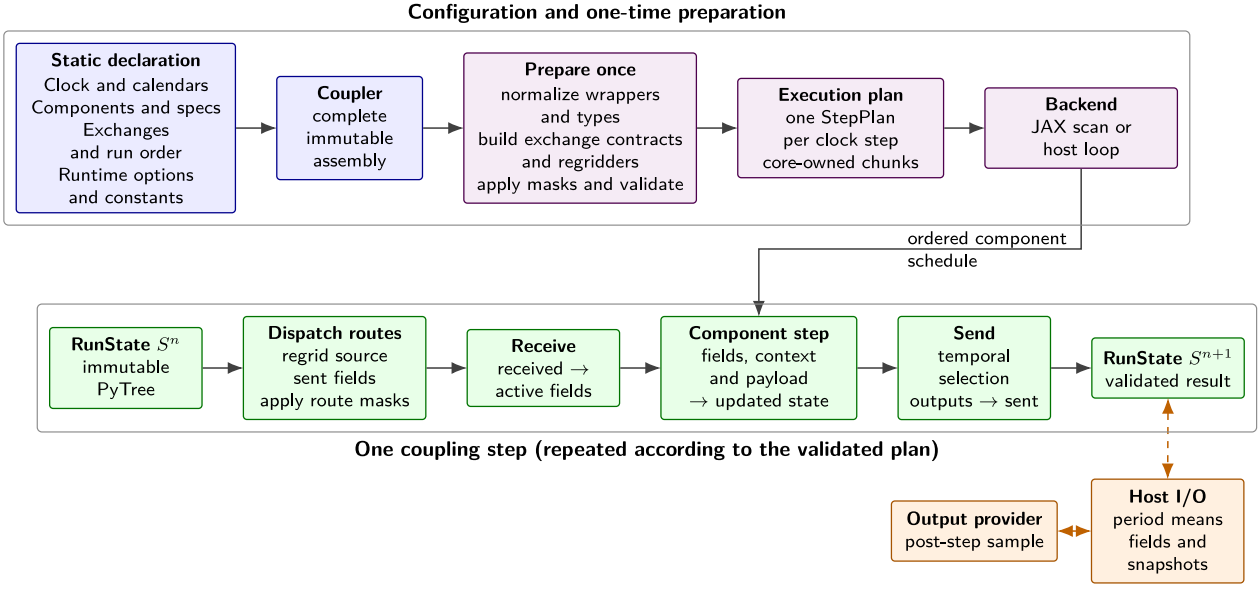


Figure 1. Software architecture and data flow in VerCOR v0.4.6 . A complete static declaration is represented as one prepared coupling and a validated execution plan. Each scheduled component is advanced through the same dispatch-receive-step-send sequence. An immutable `RunState` is passed between operations. The backend executes a plan defined by the core without changing the coupling behavior. The dashed branch represents an optional host-side output and is omitted from output-free runs that are JIT-compiled or differentiated.

VerCOR is built in a way that it follows a three-stage workflow (Fig. 1):

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

- **Declaration** of a coupling configuration, which includes components, exchange routes between them, a clock with a calendar, and runtime options.
- **Preparation** of the declaration involves normalizing each component's state, such as checking field shapes and casting fields to the configured precision. It also constructs exchange contracts and regridders. The contracts represent how data move between components, i.e., which model owns each field and which model receives it. Field ownership and transfer direction are derived from the exchange routes. VerCOR also applies topology policies, which specify where each exchange route is active and what fraction of each target grid cell participates in the exchange. Finally, it validates the resulting state.
- **Execution** with a selected backend and following an explicit plan. This advances over time an immutable `RunState` that represents the entire coupled Earth system.

Such separation makes the definition of coupling explicit. It also ensures that the setup procedure and host-side effects (such as I/O operations) are outside the time-stepping function.

2.2 Design objectives and public architecture

In order to support reproducibility along with extensibility, VerCOR follows four design objectives:

- Numerical state is represented as an immutable JAX PyTree. This makes state advancement traceable, compilable, and differentiable.
- Component wrapping (integration) does not require inheritance from a VerCOR base class. Instead, VerCOR accepts any object that provides the required structure, i.e., component name, grid, specification, and step function. This allows models with different internal designs to be integrated.
- Each kind of configuration is stored in its own immutable object. For example, physical constants are stored in `PhysicalConstants` and component declarations in `ComponentSpec`.
- User- or adapter-provided objects (wrappers), like components or exchanges, are validated at every stage where they enter the coupler.

To assemble a coupled model, the library provides a single top-level path: `Coupler`. Its constructor receives the `Clock`, all components with `Exchange` routes, and the run order of registered components. It also receives a `RuntimeOptions` object, physical constants, and logging options. To change the coupler configuration, one must construct a new coupler instance because it copies component and collections of component-field routes into internal immutable objects. A request for an initial coupler state or the start of a coupled run triggers evaluation of coupler preparation, which happens at most once per coupler instance. Table 1 summarizes the public API.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

Table 1. Public API in VerCOR v0.4.6 . Component, output, regridding, workflow, and masks are exported from the library root and provide the minimal assembly interface.

Object	Configuration or state	Runtime role
`Clock`	Start date, coupling time step, number of steps, and calendar	Supplies absolute step indices and model times in Gregorian, no-leap, or 360-day calendars.
`RectilinearGrid`	Longitude and latitude coordinates and an optional binary mask	Defines the horizontal shape of component fields and the source and target geometry for regridding.
`Component` and `ComponentSpec`	Component identity, grid, inputs, outputs, initial fields, execution capability, lifecycle, transfer and output policies	Convert received fields and the component payload into declared updates to fields and payload.
`Exchange`	Source, target, field names, route identifier, and regridder factory	Transfers scalar and vector fields from the source's sent store to the target's received store.
`Coupler` and `RuntimeOptions`	Complete coupling graph, run order, precision, workflow, backend, and topology policy	Prepare and validate the graph, select the backend, execute the plan, and coordinate optional output.
`RunState`	Coupled component fields, received and sent exchange stores, payloads, grids, and route masks	Is the immutable PyTree passed between steps and returned to the caller.

2.3 Component wrapper

Every VerCOR component follows a structural protocol with four members: a unique `name`, a `RectilinearGrid`, an immutable specification `ComponentSpec`, and a `step` method that advances the component in time. The specification defines the fields, which the component reads (`inputs`), the fields it may update (`outputs`), and initial values for declared fields. It also specifies the component execution backend (JAX or Python host), its lifecycle, how it transfers time-dependent data from `DataComponent`, and its output options. `CallableComponent` adapts a standard Python function, typically a wrapped prognostic component, according to this protocol. In contrast, `DataComponent` represents static or time-dependent forcing data.

The component lifecycle has three functions. The first initializes fields and an arbitrary component payload and is executed once during preparation. The second initializes exchange fields after all component and exchange contracts have been defined. Finally, the third performs various checks of the complete component state before execution. VerCOR also prepares numerical values by expanding initial values across the entire component grid axes and making sure all numerical values comply with the selected precision.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

The `step` method receives component fields, a `StepContext`, and an optional payload. During execution with JAX backend, the context object contains the coupling interval, step index, physical constants, precision information, and logger. For other execution backend, the context object may also include model time. Declared output fields are replaced if the `step` method simply returns fields, but the existing payload is retained. However, returning a `StepResult` can replace both output fields and payload. In a functional form, the advancement of component c can be represented as

$$\left(\boldsymbol{x}_c^{n+1}, p_c^{n+1}\right) = F_c(\boldsymbol{x}_c^n, \boldsymbol{r}_c^n, p_c^n; \Delta t, \boldsymbol{\theta}), \tag{1}$$

where $\boldsymbol{x}_c$ is component fields, $\boldsymbol{r}_c$ is the fields received through exchanges, p_c is the component's payload, Δt is the coupling interval, and $\boldsymbol{\theta}$ is the physical constants. The coupler copies $\boldsymbol{r}_c^n$ into the relevant input fields before evaluating F_c.

A model's prognostic state, forcing cursor, or internal carry are stored in the payload. At every time step, the component returns a new payload instead of modifying its definition. During compiled JAX execution, component payloads are replaced after each time advancement. The new payloads must maintain their PyTree structure, including the shapes and data types of their leaves. Because the host component operates outside the compiled JAX scan function, it may restructure its payload. In both scenarios, the component definition stays a static configuration, enabling independent runs to start from independent states.

2.4 Core data structures and field exchange

2.4.1 Grids, field dimensions, and state

Each component uses a `RectilinearGrid` with one-dimensional longitude and latitude coordinates. It has optional coordinate edges for conservative remapping and an optional two-dimensional binary mask, which defines the active domain. VerCOR supports exchange fields with two, three, and four dimensions. All field dimensions follow the standard axis order, leading with time (n_{time}), then level (n_{lev}), and finally horizontal latitude (n_{lat}) and longitude (n_{lon}). Such a broad convention prevents silent broadcasting of incompatible field dimensions.

The public immutable `RunState`, which is a JAX PyTree object, stores component working fields, fields received through exchange routes, fields to be sent to other components, and the component payload. One can construct modified initial conditions with `RunState.replace_fields`, which returns a new state while preserving its entire structure.

2.4.2 Exchange routes, regridding, and topology/masks

To define/instantiate an `Exchange` route between two components, one must provide either scalar or vector field names and a regridding method. Multiple routes between the same components have distinct `route_id` identifiers. During the preparation phase, VerCOR validates the component names, the uniqueness of `route_id` identifiers, the source-target declarations, and the capabilities of the constructed regridder. This validation makes certain that two different routes to the same component cannot overwrite the same target field.

VerCOR provides two-dimensional bilinear interpolation and conservative remapping for rectilinear grids. Bilinear interpolation supports scalar fields and horizontal vector components, whereas conservative remapping supports only scalar fields.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

Geometric weights of regridding objects are constructed during the preparation phase, outside the time loop. A topology policy (exchange-mask rules) can also generate binary and fractional masks for each exchange route. These masks determine where transferred data applies and what fraction of each target grid cell receives it.

For data components with time-indexed forcing data, `TransferPolicy` controls which value is placed in the sent store. Currently, the coupler supports the current time-step value, linear interpolation between adjacent monthly mean values, and selection of daily mean values. The coupler clock precomputes calendar-dependent indices and interpolation weights. Field and mask validation enforces the required target shape and precision. NaN values and infinite values are always rejected at the preparation, exchange, component step, and send stages.

2.5 Runtime preparation, scheduling, and execution

2.5.1 Preparation and validation

The first call to `initial_state` or `run` creates a prepared coupling with an immutable state. During the preparation phase, the program calls component setup methods/functions. It also ensures that the specified precision is used across all grids, fields, physical constants, and components. In addition, it builds exchange contracts and creates regridders and masks. The system then allocates and prefills stores for component working fields and for received and sent data. Then it validates the complete `RunState`. The validation procedure checks component names, execution order, and exchange routes between them. It also checks grid coordinates and associated masks, along with array shapes, fields and masks finiteness, and the payload structure.

A start date, a coupling interval, a calendar and a total number of time steps define the clock. Currently, Gregorian, no-leap, and 360-day calendars are supported. Runtime precision of floating-point types in the entire VerCOR is controlled by the `RuntimeOptions`.

2.5.2 Execution plan and coupling logic

Before the simulation begins and the state is advanced, a fixed schedule for the entire run, called `ExecutionPlan`, is constructed. It contains exactly one `StepPlan` for each time step. The `StepPlan` contains information about all components to be run and the order in which they run. The coupler also validates the plan before advancing any component.

For each scheduled component, four operations are performed in the following order: (1) all fields are dispatched and interpolated to the target component; (2) received fields are copied into the component's input fields; (3) the component is advanced by one time step; and (4) selected outgoing fields are copied to the sent store, making them available for other components. The result is then written to a new, immutable coupled state. A component can use the current-step output of a source component that advanced earlier in the order. However, no feedback route to a component advanced earlier is allowed.

2.5.3 JAX, automatic differentiation and custom backends

By default, VerCOR chooses JAX backend when all registered components are JAX-based. Otherwise, it uses the Python host driver. For JAX execution, consecutive coupling time steps that run the same components in the same order are combined into a

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

single execution block, called a chunk. Each of these chunks is compiled with `jax.jit` and is run with `jax.lax.scan`, carrying `RunState` between iterations. This block division is needed to support periodic output and checkpointing (see subsection 2.7 for details). For an output-free JAX execution, VerCOR supports forward-mode (first-order Jacobian-vector products) and reverse-mode (vector-Jacobian products) gradients with respect to physically valid continuous inputs. We use `jax.checkpoint` during gradient calculations to save memory at the cost of some additional computation. This function recalculates temporary results when needed instead of storing them from every coupler time step.

A custom backend follows the same execution plan but through a pure Python loop. This execution plan lets VerCOR run components written in non-JAX libraries (such as PyTorch). Such components could have mutable state or other host-side behavior. The program becomes non-differentiable in this case; however, VerCOR still controls how fields are received and sent, how components advance in time, output handling, and so on.

2.6 Bundled components and model integration

VerCOR has small slab components for atmosphere, ocean, land, and sea ice. These are very simple models used for examples and rapid construction of coupling graphs. The library also includes data components based on ERA-Interim (Dee et al., 2011) and ERA5 (Hersbach et al., 2020) reanalysis data. We also include wrappers for traditional atmospheric and ocean models. The JAX-GCM (JCM) is the intermediate-complexity differentiable atmospheric model described by Davenport et al. (2026). The Veros is the JAX-based ocean general circulation model (Häfner et al., 2018, 2021). There is also a wrapper for CAMulator, the machine-learned CAM6 atmosphere emulator (Chapman et al., 2025) developed within the MILES-CREDIT project (Schreck et al., 2025).

These wrappers adapt components' fields and array shapes plus initialize model-specific payloads to fit VerCOR conventions. They also advance each model for the coupled time interval. The library converts bundled components into a VerCOR-conventional internal representation. After that, the library applies the usual settings for routes, state, workflow, backend, and output configurations.

2.7 Output

To perform output, both an individual component wrapper and the coupler must enable it. A component's `OutputSpec` defines how its results are sampled, for example, an averaging period, and whether to turn model-specific snapshots on or off. After the component completes a time step, it returns a data record containing named variables, coordinates, dimension labels, and the metadata required for NetCDF files. VerCOR can write averaged fields after every step or at the end of each day, month, or year. A central output coordinator selects variables, checks their dimensions and metadata, accumulates averaged fields, assigns calendar times, creates paths that avoid filename conflicts, and writes the NetCDF files.

One can also run VerCOR in an output-free way. Such mode allows gradient calculations for the complete coupled simulation with JAX backend. Input and output (I/O) operations are not allowed while JAX is tracing the simulation state because Python performs ordinary file operations through the computer's operating system. When JAX traces the state, it examines the

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

calculations to build a compiled program (the sequence of operations), where arrays are placeholders for future values. However, I/O requires actual data rather than a mathematical operation that JAX can compile or differentiate.

3 Tuning of a new coupled model

In a first application we use VerCOR to couple the JAX-based ocean model Veros (Häfner et al., 2021) to the JAX-based atmosphere model JCM/Speedy-v2 (Davenport et al., 2026). This VERES combines JCM at T31 spectral truncation with 8 vertical layers and Veros at a horizontal resolution of 4° with 15 vertical layers. Land albedo, soil moisture, and snow cover are based on monthly climatology obtained from ERA-Interim reanalysis over the period of 1979-2008; there are no rivers. North of $80^\circ N$ albedo, temperature and cover of sea-ice are based on ERA-Interim too, and surface salinity is restored to the climatological levels from the World Ocean Atlas 2005 (Antonov and Levitus, 2006). We plan to soon couple a sea-ice and land model, and target a 1 degree versions of VERES, but for now the goal is to build a tool with which to explore tropical convection and its optimization in a coupled ocean-atmosphere configuration.

3.1 Strategy

Veros and JCM have been developed and optimized independently of each other, and after the technical challenge of coupling has been overcome (previous section), model parameters need to be adjusted to yield a reasonable model climate (Hourdin et al., 2017). As it is, there is too much equatorial precipitation and the precipitation center is shifted south in all three basins (Fig. 2). Furthermore, the SST in the Southern Ocean and the coastal upwelling regions off South America and Africa is too warm, and in the northern mid-latitudes too cold. The structure of these biases is similar to other GCMs, although the amplitude is certainly larger (e.g., Small and coauthors (2014)). In combination with steep orographic features, the spectral representation of the atmosphere causes spectral ringing, which is expressed most strongly by the concentric rainbands emerging from the Andes. This feature will become weaker with higher resolutions and more sophisticated dynamical cores, and will not be addressed here.

There are dozens of adjustable parameters and it is not obvious how to proceed with tuning the model. Furthermore, it is important to note that this tuning has to be done in a coupled framework; optimizing individual components with prescribed forcing will provide poor results (Stoustrup et al., 2026). We provide here a recipe for VERES, with the goal of making the tuning process transparent and reproducible:

1) the goal: We plan to use our VERES configuration to develop ML-based closures for tropical convection, and to teach ENSO dynamics to students. Thus, for the calibration we target the tropical mean climate.

2) the process: Tropical convection is the single most important process in shaping not only tropical but also global climate, and tropical precipitation is a well observed indicator of the quality of its representation. This is not a law of nature, but an empirical finding (e.g., Barsugli and Sardeshmukh (2002), Stoustrup et al. 2026). Thus, we will target tropical precipitation as target for VERES optimization.

3) calibration: There are numerous parameters that affect precipitation, and they may not be independent of each other. Thus,

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

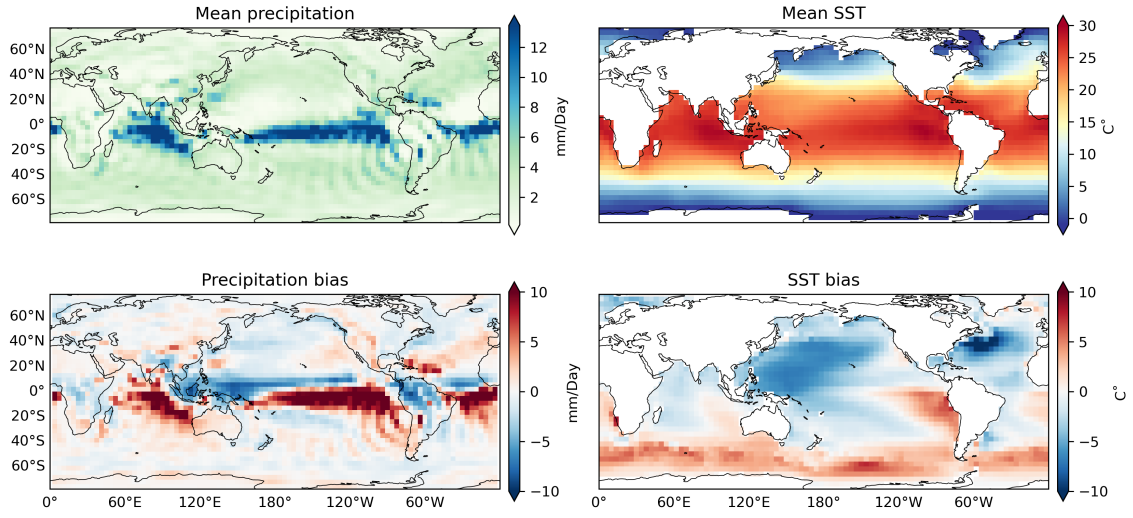


Figure 2. 20 year mean of the VERES precipitation (upper left), its bias (lower left), SST (upper right), and its bias (lower right).

we use Bayesian optimization (see below) to find their optimal combination, and we chose tropical Pacific precipitation as target for the calibration.

4) the parameters: Expert opinion will provide the list of parameters that most likely affect tropical precipitation. It will also provide physically meaningful uncertainty ranges.

5) validation & selection: Validation is based on tropical precipitation in the other two basins, to avoid overfitting, and on the net top-of-the-atmosphere radiation. From the remaining parameter combinations, we chose the one with the best SST and zonally averaged surface wind stress. Again, other choices are possible, but ours are natural choices for final goals: a new convection closure and a better understanding of ENSO.

3.2 Bayesian optimization

For Bayesian optimization we use the Python package VerOpt (Versatile Optimizer, Stoustrup et al. (2026)). It provides efficient gradient-free optimization of a computationally expensive model by utilizing Gaussian Process Regression to construct a surrogate model of the loss (or objective) function. Objective function estimates and uncertainties given by the surrogate model are utilized to identify promising regions in the parameter space by maximizing an acquisition function (Mrozowska et al. (2025).

Coupled climate models have, across decades of development, consistently presented tropical rainfall biases most prominently characterized by a double Inter-Tropical Convergence Zone (ITCZ) bias over the Pacific Ocean (Zhang et al. (2019)). To target both the magnitude and spatial distribution of the bias, model-observation precipitation $RMSE$ between $160°E$ and $100°W$ is minimized in a Northern box $(0-20°N)$ and a Southern box $(0-20°S)$ simultaneously:

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

$$\mathcal{L}_N(\mathbf{X}) = \sqrt{\sum_{i\in\mathcal{N}} \frac{w_i(P_i(\mathbf{X}) - O_i)^2}{\sum_{j\in\mathcal{N}} w_j}} \qquad \mathcal{L}_S(\mathbf{X}) = \sqrt{\sum_{i\in\mathcal{S}} \frac{w_i(P_i(\mathbf{X}) - O_i)^2}{\sum_{j\in\mathcal{S}} w_j}}$$

where $\{\mathcal{N},\mathcal{S}\}$ denote the North and South box domains, and $w_i = cos(\varphi)$ denotes grid cell weights by latitude. P denotes model precipitation values computed as the mean of the final 20 years of each 30-year model integration (the uncertainty of the annual mean precipitation for these twenty years is 0.3 mm.day), and O denote the target precipitation values computed as the 1980-2020 temporal mean of climatology given by the Global Precipitation Project Version 2.3 (GPCP; Adler et al. (2003, 2018)). Previous studies highlight sub-grid scale parameterizations of convective and condensation processes, as well as biased ocean-atmosphere interactions, as mechanisms underlying the double ITCZ bias (Zhang et al. (2019)). This study explores a 9-dimensional atmosphere model parameter set described in Table 2.

Scheme	Parameter	Default value	Bounds
Surface flux	`vgust` (gust wind speed)	5.0	[2.5, 7.5]
	`cds` (ocean/atmosphere drag coefficient)	$1 \cdot 10^{-3}$	$[5 \cdot 10^{-4}, 1.5 \cdot 10^{-3}]$
Convection	`rhbl` (relative humidity threshold, boundary layer)	0.9	[0.45, 1.0]
	`entmax` (maximum entrainment rate)	0.5	[0.25, 0.75]
	`trcnv` (convective relaxation timescale)	6.0	[3.0, 9.0]
	`rhil` (relative humidity threshold, cloud layer)	0.7	[0.35, 1.0]
Condensation	`rhlsc` (RH threshold, large-scale condensation)	0.9	[0.45, 1.0]
	`rhblsc` (RH threshold, BL large-scale condensation)	0.95	[0.475, 1.0]
	`trlsc` (large-scale condensation relaxation timescale)	4.0	[2.0, 6.0]

Table 2. Optimized parameters with default values and search bounds. Bounds are linear and span $\pm 50\%$ of each default.

The optimization study uses a total of 250 model integrations, the first 50 of which are randomly sampled initial points (each point has as coordinates values of the nine parameters in Table 2) to condition the surrogate function and the remaining 200 are Bayesian points selected by maximizing the acquisition function. To simultaneously pursue the North and South box objectives, this study uses a qLogEHVI acquisition function which quantifies the expected improvement in the hyper-volume dominated by the Pareto front.

3.3 Calibration & validation

The upper left panel illustrates the central working mechanism of Veropt: an initial random seeding of points yields results in the left or central part of the panel. The default precipitation (marked with a star) is worse than most randomly selected points. As VerOpt learns the topology of the loss function it finds more and more points closer to the upper right corner, which minimize the precipitation bias. Marked in color is net TOA radiation: clearly, there is some trade-off between TOA radiation and quality of tropical precipitation. The default has a net heat gain of 15 W/m2, so we decided to only analyze points with an imbalance of

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

less than 20 W/m2. This is a subjective choice; clearly even 2 W/m2 is too much of an imbalance for long integrations. The current goal, however, is to provide a tool to develop ML-based closures for tropical convection, so that we decided to focus on
240 TOA radiation only after the sea-ice and land model have been added. For clarity, the remaining panels have these points with excessive TOA imbalances removed, and it can be seen that points that yield good precipitation in the tropical Pacific, also lead to improvements in the Atlantic without much deterioration in the Indic or the global SST. Thus, we chose the rightmost of the points that have less than 20 W/m2 TOA inbalance (marked as hexagon): it more than halves the SH precipitation biases without much loss in other key metrics.

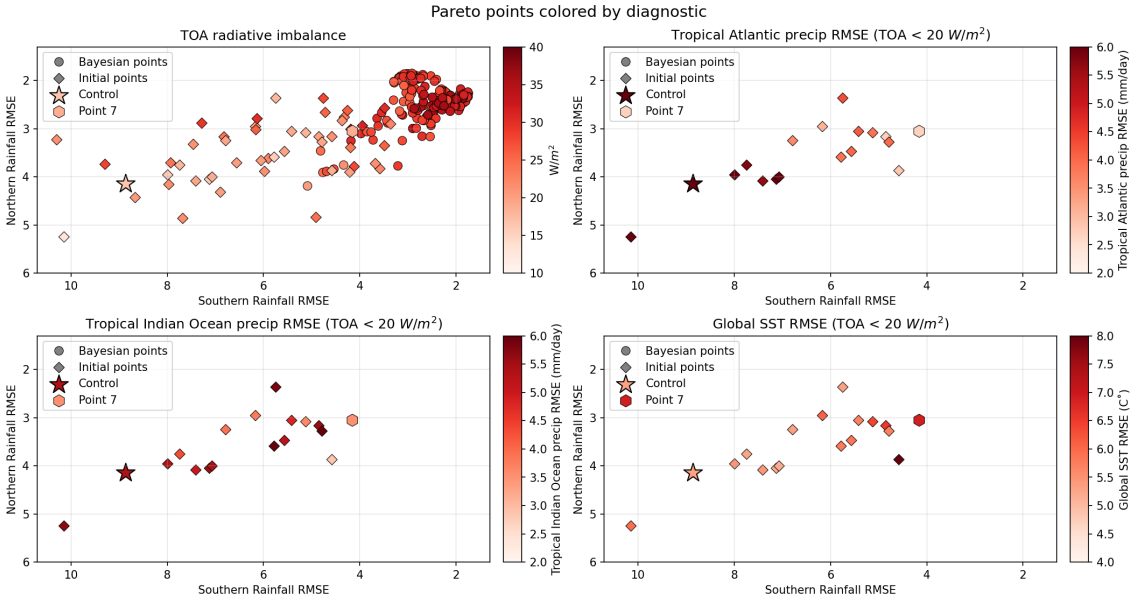


Figure 3. Evaluation of the precipitation biases for the individual points (parameter combinations). Every simulation has a precipitation bias in the southern and the northern box (x and y-axis, respectively). In color, the four different panels each show a validation metric that has not been part of the objective function: net TOA radiation (upper left), tropical Atlantic (upper right) & Indic precipitation bias (lower left), and global SST (lower right).

245 Despite the large improvement in precipitation, the resulting parameter values are in general not far off from the default (Table 3), which speaks to the sensitivity of precipitation to parameterized atmosphere processes.

Table 3. Parameter values and objectives for point 7 and the control run.

	vgust	rhbl	rhlsc	cds	entmax	trcnv	rhil	rhblsc	trlsc	North obj.	South obj.	Total obj.
Point 7	5.39	0.49	0.96	$9.7 \cdot 10^{-4}$	0.32	6.18	0.88	0.75	3.17	3.05	4.16	7.21
Control	5.00	0.90	0.90	$1.0 \cdot 10^{-3}$	0.50	6.00	0.70	0.95	4.00	4.15	8.86	13.01

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

A permutation importance algorithm using a random forest regressor model was used through the Sci-Kit Learn to identify parameter importance in objective function value. The algorithm essentially fits the objective function values and parameters using a random forest regressor, permutes individual parameters, and measures how much the model weakens. By identifying which parameters the model relies on the most, their importance to the objective function value can be measured. Note that this importance value does not reflect relation to high-quality objective values *per se*, but rather how sensitive the model is to changes to a particular parameter (positively or negatively). Table 4 shows permutation importance values for optimized parameters. Importantly, parameters have different importance for each objective function. Gust wind speed emerges as the most important parameter for the south-box objective, with other parameter values being much less important. For the north-box objective, relative humidity threshold and boundary-layer large-scale condensation realitive-humidity threshold, alongside gust wind-speed, appear as the most important parameters.

Table 4. Permutation importance of optimized parameters (mean $\pm$ standard deviation) for each objective box. Importance values are absolute drop in random forest regressor score (R^2) when parameter is permuted.

Parameter	South box objective	North box objective
`surface_flux.vgust`	1.319 ± 0.225	0.365 ± 0.086
`surface_flux.cds`	0.131 ± 0.029	0.039 ± 0.031
`condensation.rhblsc`	0.062 ± 0.017	0.471 ± 0.098
`convection.rhbl`	0.055 ± 0.014	0.707 ± 0.154
`condensation.trlsc`	0.054 ± 0.024	-0.002 ± 0.008
`convection.rhil`	0.021 ± 0.014	0.038 ± 0.018
`convection.trcnv`	-0.001 ± 0.004	0.017 ± 0.006
`convection.entmax`	-0.011 ± 0.007	-0.011 ± 0.008
`condensation.rhlsc`	-0.030 ± 0.031	0.029 ± 0.011

Comparing the precipitation and SST fields between the control and the optimized setting shows that the precipitation biases have indeed been reduced throughout, but their spatial pattern has remained (Fig. 4). Furthermore, as a result of the generally larger cloud cover (not shown), the SST is cooler almost everywhere, which is an improvement in the SH and a deterioration in the NH. There has been little change in surface winds (not shown), apart from a slight improvement of the SH trades.

A full validation would require, of course, an analysis of the subsurface ocean and long-term trends, which is postponed until the land and sea-ice models have been coupled as well. The main purpose here is to provide a publicly available platform to develop new subgrid closures and ESM components and illustrate the workflow. VerOpt is gradient free and works well for problems with less than 20 dimensions. On 48 cores the present 30 year integrations take approximately a day, so that with 8 nodes the optimization discussed above takes 3 weeks. Thus, longer integrations with full one-degree ESMs can easily be done at the large GPU-based HPC centers. However, spatially varying parameters or neural-network based subgrid closures can easily have thousands of parameters, so that it would be helpful to have techniques other than Bayesian optimization available. The

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

next section describes the use of a JAX functionality that creates special demands at the model architecture, but promises to be much more efficient.

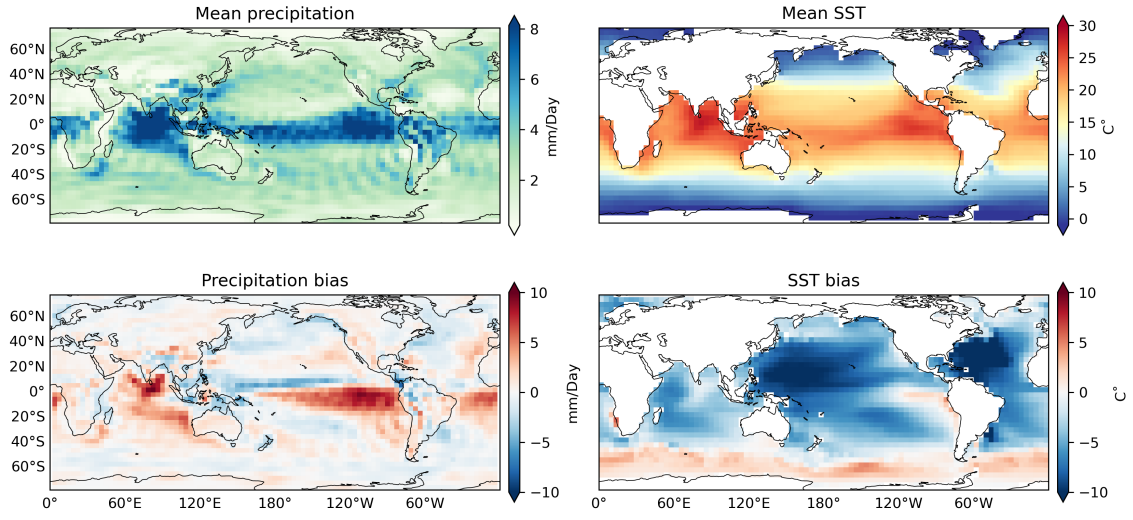


Figure 4. 20 year mean of the VERES precipitation (upper left), its bias (lower left), SST (upper right), and its bias (lower right) for the chosen best parameter combination.

4 Gradient based optimization with JAX autodiff

Computing the gradient of a model output with respect to its inputs or parameters is central to many tasks in Earth system modelling, as for instance parameter estimation (Navon, 1998; Gelbrecht et al., 2023), variational data assimilation (Le Dimet and Talagrand, 1986; Courtier et al., 1994), or the training of data-driven corrections and closures (Ramadhan et al., 2020; Frezat et al., 2022; Kochkov et al., 2024). Adjoint models have long been used for this purpose in ocean and atmosphere models, written either by hand or with automatic source translation tools (Errico, 1997; Giering and Kaminski, 1998; Heimbach et al., 2005; Hascoët and Pascual, 2013). However, developing and maintaining an adjoint remains demanding, since it has to be updated each time the forward code changes. The development of deep learning libraries such as PyTorch (Paszke et al., 2019) and JAX (Bradbury et al., 2018) has popularised automatic differentiation (AD), allowing gradients of numerical computations to be evaluated without manually implementing their derivatives. Several AD strategies exist, i.e., forward or reverse mode, continuous or discrete adjoints, and they are reviewed in Sapienza et al. (2024). Each of them comes with its own trade-off in terms of computational cost, memory, numerical stability and ease of implementation. In this work, we apply reverse-mode AD (backpropagation) directly on the discretised solver, i.e., we differentiate the discrete forward program (discretise-then-differentiate). This choice is the easiest to implement in our case, since it leaves the forward model unchanged, while still providing in one backward pass the gradient with respect to a large number of parameters.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

To this end, we adapted Veros so that it is differentiable end-to-end (Meunier et al., 2025). The main modifications consist in writing the time step as a functionally pure function and in handling functions with singular gradients. Thus, we are able to compute gradients through a full coupled simulation and use them to calibrate the model parameters. VerCOR and JCM are both fully differentiable, too, and we can perform the calibration in VERES. We chose c_k and c_ϵ, two key parameters of the vertical-mixing parameterization, which controls the ocean's stratification (see Mrozowska et al. (2025) for the Bayesian optimization of the same two parameters).

The calibration metric is the near-surface vertical structure of temperature and salinity. Let $z_1 > z_2 > z_3$ denote the depths of the three uppermost ocean levels (z_1 the surface level), and for a field $X \in \{T, S\}$ define its difference to the surface level as

$$\Delta_k X = X(z_k) - X(z_1), \qquad k \in \{2,3\}. \tag{2}$$

This quantity is designed to evaluate the stratification of the field. A first simulation is run with the true (unknown to the optimizer) parameters to generate a synthetic target, which we then try to recover through optimization from four different random initial parameter sets. Optimization is done using gradient descent using gradients obtained by reverse-mode automatic differentiation (backpropagation) through the entire coupled rollout. The loss is

$$\mathcal{L}(\tilde{c}_k, \tilde{c}_\epsilon) = \sum_{i\in\Omega} \sum_{X\in\{T,S\}} \sum_{k\in\{2,3\}} w_{X,k} \left(\Delta_k X(i;\tilde{c}_k,\tilde{c}_\epsilon) - \Delta_k X(i;c_k,c_\epsilon)\right)^2, \tag{3}$$

where Ω is the spatial domain, $\tilde{c}_k, \tilde{c}_\epsilon$ are the parameters being optimized, c_k, c_ϵ the true (target) parameters, and $w_{X,k}$ is the inverse spatial variance of the target field $\Delta_k X(\cdot; c_k, c_\epsilon)$, which makes the temperature and salinity terms comparable in scale.

The model is integrated for 10 days, the gradients are saved for every time step, and the tuning target is the model state averaged over days 8-10. Four combinations of c_k and c_ϵ are chosen that are different from the default, and backpropagation should be able to recover the default values. To visualize the procedure, we computed the topology of the loss function, and superimposed on it the trajectory of the evolving parameter values (Fig. 5). All four initial parameter combinations are much improved, but only one recovers the default values, the other three end up in a relative minimum. Interestingly, the red-point trajectory crosses the purple-point's, but ends up in a different minimum. This is due to the discrete value of the individual parameter improvements; the behavior of these trajectories suggests that the topology of the loss function varies on smaller scales than is accounted for in the learning rate.

The mixed-layer example provided here is a proof of concept to show that the coupled model is differentiable and that gradient-based optimization can lead to meaningful results. However, we chose the ocean mixed layer, because from previous experience (Mrozowska et al. 2025) we know that the loss function is well behaved and adjustments happen fast. Decades or century long integrations will face significant memory needs and high demands on the accuracy of the gradients (Rehmann et al., 2026). Furthermore, atmospheric circulation is turbulent even at low resolutions so that a single backpropagation may be quite misleading, even without the relative-minimum problem. These challenges will be overcome eventually, and we expect that a library like VERES will be helpful along the way.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

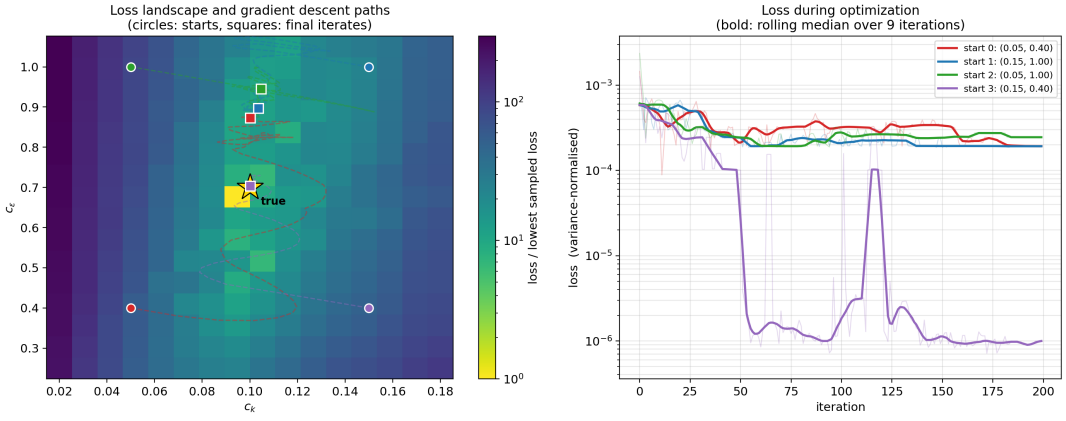


Figure 5. Top left: loss landscape over the two parameters, with descent paths from four random initializations. Top right: loss evolution during optimization, showing the presence of a local minimum.

5 Summary and outlook

Combining different ESM components and tuning of the resulting coupled model is a major bottleneck in ESM development. Especially in coupled configurations parametric uncertainty shows up in the most unexpected places (e.g., Jochum et al. 2008), and can mask or falsely indicate the presence of a structural model flaw, something that needs to be addressed with new or improved physics. The goal of VerCOR is to provide an environment where coupling and tuning of new model elements is quick and easy, to allow the scientist to focus on new physics. Thus, VerCOR is based on Python/JAX, which makes it accessible to even BSc students and allows it to exploit the rapidly growing amount of GPU resources. We demonstrate the usefulness of this simple & differentiable coupler by coupling a new ocean to a new atmosphere model and calibrating the result with the Bayesian optimizer VerOpt. Furthermore, we demonstrate that the coupled model is fully differentiable, which provides an alternative way to calibrate model parameters.

We expect to use the present coarse resolution ocean/atmosphere configuration in teaching, and for research into the application of auto-differentiability to model development. Using our access to EuroHPC's Lumi, we will repeat the tuning exercise described in section 3 to a T106/1 degree version of JCM/Veros, and then explore the online tuning of data-driven representations of tropical convection. Furthermore, we will provide an interface to a JAX-based atmospheric emulator (CAMulator, Chapman et al. 2026), a sea-ice model (Veris, Gärtner et al. (2026)), as well as a JAX version of the marine biogeochemistry model MOBI (Muglia et al. (2017)). The goal is not an IPCC-class ESM, rather it is to have a lightweight library with which researchers can explore coupled phenomena across different Earth system components without having to rely on the Fortran-based all-or-nothing environment of the established ESMs.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

Code availability. The version of VerCOR v0.4.6 used in this study has been archived on Zenodo at https://doi.org/10.5281/zenodo.23594053 (Nuterman et al., 2026). The project source code is also available on GitHub: https://github.com/nutrik/vercor.

Data availability. The data used to produce the results in this paper are available at: https://doi.org/10.17894/ucph.d6e0dff5-7103-4c20-9fc4-b15ce1cd3b0a

Author contributions. RN developed VerCOR and wrote the description of its architecture. AS developed VerOpt, and IR & VV applied it. EM performed the gradient based optimization, and MJ designed calibration and optimization experiments and wrote the manuscript. All authors reviewed the manuscript and were involved in the selection of the results presented in this paper.

Competing interests. The authors declare no competing interests.

Acknowledgements. The present work is unthinkable without the relentless drive for perfection at the world's climate modelling centers, and we are indebted to generations of software engineers and climate scientists. We hope that translating a small part of their work into Python/JAX will entrain future students of climate and computer scientists into the field, and open up new avenues of discovery.

The authors thank Peter Lauritzen (NCAR, Boulder, Colorado) for valuable discussions and insights into the implementation details of CESM (Community Earth System Model) coupler, which helped inform the development of Vercor. We are also grateful to Prof. Brian Vinter (Aarhus University) for the original inspiration and to Dr. Dion Häfner (Pasteur Labs) for his continued support.

The authors also acknowledge the use of OpenAI Codex (OpenAI) for AI-assisted software development. A substantial part of the source code was generated using Codex based on algorithmic specifications, numerical formulations, and instructions provided by the authors.

The funding for VERES was provided by several agencies: RN was supported through the PRECISE (Prediction of Ice Sheets on Earth) project of Novo Nordisk Foundation (NNF Challenge, grant no. NNF23OC0081251), IR and VV through CINCH (Causal Inference for Climate Change), Villum Foundation, EM through Institute Pierre-Simone Laplace, AS through ClimTip, an EU-Horizon project, and MJ through the Niels Bohr Institute. We are grateful for the compute provided by the Danish Center of Climate Computing and the high-performance computing support from the Derecho (doi:10.5065/qx9a-pg09) and Casper (https://ncar.pub/casper) systems provided by the NSF National Center for Atmospheric Research (NCAR). Furthermore, we are indebted to Ellen Davenport, Tien-Yiao Hsu and Duncan Watson-Parris for numerous fruitful discussions.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

References

Adler, R. F., Huffman, G. J., Chang, A., Ferraro, R., Xie, P., Janowiak, J., Rudolf, B., Schneider, U., Curtis, S., Bolvin, D., Gruber, A., Susskind, J., Arkin, P., and Nelkin, E.: The Version-2 Global Precipitation Climatology Project (GPCP) Monthly Precipitation Analysis (1979–Present), Journal of Hydrometeorology, 4, 1147–1167, https://doi.org/10.1175/1525-7541(2003)004<1147:TVGPCP>2.0.CO;2, 2003.

Adler, R. F., Sapiano, M. R. P., Huffman, G. J., Wang, J.-J., Gu, G., Bolvin, D., Chiu, L., Schneider, U., Becker, A., Nelkin, E., Xie, P., Ferraro, R., and Shin, D.-B.: The Global Precipitation Climatology Project (GPCP) Monthly Analysis (New Version 2.3) and a Review of 2017 Global Precipitation, Atmosphere, 9, 138, https://doi.org/10.3390/atmos9040138, 2018.

Antonov, J. and Levitus, S.: World ocean atlas 2005. Vol. 2, Salinity, Report, NOAA, https://repository.library.noaa.gov/view/noaa/1127, 2006.

Barsugli, J. J. and Sardeshmukh, D.: Global atmosphere sensitivity to tropical SST anomalies throughout the Indo-Pacific basin., J. Climate, 15, 3427–3441, https://doi.org/10.1175/1520-0442(2002)015<3427:GASTTS>2.0.CO;2, 2002.

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

Chapman, W. E., Schreck, J. S., Sha, Y., Gagne II, D. J., Kimpara, D., Zanna, L., Mayer, K. J., and Berner, J.: CAMulator: Fast Emulation of the Community Atmosphere Model, arXiv, https://doi.org/10.48550/arXiv.2504.06007, 2025.

Courtier, P., Thépaut, J.-N., and Hollingsworth, A.: A strategy for operational implementation of 4D-Var, using an incremental approach, Quarterly Journal of the Royal Meteorological Society, 120, 1367–1387, https://doi.org/10.1002/qj.49712051912, 1994.

Davenport, E. H., Madan, J. V., Gjini, R., Brzenski, J., Crawford, B., Ho, N., Hsu, T.-Y., Liang, Y., Liu, Z., Manivannan, V., Pham, E., Vutukuru, R., Williams, A. I. L., Yang, Z., Yu, R., Lutsko, N. J., Hoyer, S., and Watson-Parris, D.: JCM v1.1: A Differentiable, Intermediate-Complexity Atmospheric Model, Geoscientific Model Development, 19, 6451–6466, https://doi.org/10.5194/gmd-19-6451-2026, 2026.

Dee, D. P., Uppala, S. M., Simmons, A. J., Berrisford, P., Poli, P., Kobayashi, S., Andrae, U., Balmaseda, M. A., Balsamo, G., Bauer, P., Bechtold, P., Beljaars, A. C. M., van de Berg, L., Bidlot, J., Bormann, N., Delsol, C., Dragani, R., Fuentes, M., Geer, A. J., Haimberger, L., Healy, S. B., Hersbach, H., Hólm, E. V., Isaksen, L., Kållberg, P., Köhler, M., Matricardi, M., McNally, A. P., Monge-Sanz, B. M., Morcrette, J.-J., Park, B.-K., Peubey, C., de Rosnay, P., Tavolato, C., Thépaut, J.-N., and Vitart, F.: The ERA-Interim reanalysis: configuration and performance of the data assimilation system, Quarterly Journal of the Royal Meteorological Society, 137, 553–597, https://doi.org/10.1002/qj.828, 2011.

Errico, R. M.: What is an adjoint model?, Bulletin of the American Meteorological Society, 78, 2577–2591, https://doi.org/10.1175/1520-0477(1997)078<2577:WIAAM>2.0.CO;2, 1997.

Frezat, H., Le Sommer, J., Fablet, R., Balarac, G., and Lguensat, R.: A posteriori learning for quasi-geostrophic turbulence parametrization, Journal of Advances in Modeling Earth Systems, 14, e2022MS003 124, https://doi.org/10.1029/2022MS003124, 2022.

Gärtner, J. P., Losch, M., Cheedela, S. K., Jochum, M., and Nuterman, R.: Veris: fast & efficient sea-ice modeling in Python with GPU acceleration, Geoscientific Model Development, 19, 5225–5236, https://doi.org/10.5194/gmd-19-5225-2026, 2026.

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

Gent, P. R., Willebrand, J., McDougall, T. J., and McWilliams, J. C.: Parameterizing Eddy-Induced Tracer Transports in Ocean Circulation Models, Journal of Physical Oceanography, 25, 463 – 474, https://doi.org/10.1175/1520-0485(1995)025<0463:PEITTI>2.0.CO;2, 1995.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.

Giering, R. and Kaminski, T.: Recipes for adjoint code construction, ACM Transactions on Mathematical Software, 24, 437–474, https://doi.org/10.1145/293686.293695, 1998.

Häfner, D., Jacobsen, R. L., Eden, C., Kristensen, M. R. B., Jochum, M., Nuterman, R., and Vinter, B.: Veros v0.1 - A Fast and Versatile Ocean Simulator in Pure Python, Geoscientific Model Development, 11, 3299–3312, https://doi.org/10.5194/gmd-11-3299-2018, 2018.

Häfner, D., Nuterman, R., and Jochum, M.: Fast, Cheap, and Turbulent - Global Ocean Modeling with GPU Acceleration in Python, Journal of Advances in Modeling Earth Systems, 13, e2021MS002 717, https://doi.org/10.1029/2021MS002717, 2021.

Hascoët, L. and Pascual, V.: The Tapenade automatic differentiation tool: principles, model, and specification, ACM Transactions on Mathematical Software, 39, 20:1–20:43, https://doi.org/10.1145/2450153.2450158, 2013.

Heimbach, P., Hill, C., and Giering, R.: An efficient exact adjoint of the parallel MIT General Circulation Model, generated via automatic differentiation, Future Generation Computer Systems, 21, 1356–1371, https://doi.org/10.1016/j.future.2004.11.010, 2005.

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

Hourdin, F., Mauritsen, T., Gettelman, A., Golaz, J.-C., Balaji, V., Duan, Q., Folini, D., Ji, D., Klocke, D., Qian, Y., Rauser, F., Rio, C., Tomassini, L., Watanabe, M., and Williamson, D.: The Art and Science of Climate Model Tuning, Bulletin of the American Meteorological Society, 98, 589 – 602, https://doi.org/10.1175/BAMS-D-15-00135.1, 2017.

Jochum, M., Jahn, A., Peacock, S., Bailey, D. A., Fasullo, J. T., Kay, J., Levis, S., and Otto-Bliesner, B.: True to Milankovitch: Glacial inception in the new Community Climate System Model, J. Climate, 25, 2226–2239, https://doi.org/10.1175/JCLI-D-11-00044.1, 2012.

Kochkov, D., Yuval, J., Langmore, I., Norgaard, P., Smith, J., Mooers, G., Klöwer, M., Lottes, J., Rasp, S., Düben, P., Hatfield, S., Battaglia, P., Sanchez-Gonzalez, A., Willson, M., Brenner, M. P., and Hoyer, S.: Neural general circulation models for weather and climate, Nature, 632, 1060–1066, https://doi.org/10.1038/s41586-024-07744-y, 2024.

Le Dimet, F.-X. and Talagrand, O.: Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects, Tellus A, 38, 97–110, https://doi.org/10.3402/tellusa.v38i2.11706, 1986.

Manabe, S. and Bryan, K.: Climate Calculations with a Combined Ocean-Atmosphere Model, Journal of Atmospheric Sciences, 26, 786 – 789, https://doi.org/10.1175/1520-0469(1969)026<0786:CCWACO>2.0.CO;2, 1969.

Meunier, E., Ouala, S., Frezat, H., Le Sommer, J., and Fablet, R.: Towards Fully Differentiable Neural Ocean Model with Veros, in: Differentiable Systems and Scientific Machine Learning Workshop at EurIPS 2025, Copenhagen, Denmark, https://doi.org/10.48550/arXiv.2511.17427, 2025.

Mrozowska, M. A., Avery, J., Stoustrup, A., Nuterman, R., Johnsen, C.-J., Thormann, A., and Jochum, M.: Bayesian Optimization With GPU Acceleration for Ocean Models, Journal of Geophysical Research: Machine Learning and Computation, 2, e2024JH000 517, https://doi.org/10.1029/2024JH000517, 2025.

Muglia, J., Somes, C. J., Nickelsen, L., and Schmittner, A.: Combined effects of atmospheric and seafloor iron fluxes to the glacial ocean., Paleoceanography, 32, 1204–1218, https://doi.org/10.1002/2016PA003077, 2017.

Navon, I. M.: Practical and theoretical aspects of adjoint parameter estimation and identifiability in meteorology and oceanography, Dynamics of Atmospheres and Oceans, 27, 55–79, https://doi.org/10.1016/S0377-0265(97)00032-8, 1998.

https://doi.org/10.5194/egusphere-2026-6549
Preprint. Discussion started: 27 November 2026
© Author(s) 2026. CC BY 4.0 License.



Nuterman, R., Rockström, I., Veluvali, V., Meunier, E., Stoustrup, A., and Jochum, M.: Versatile Earth-systems COupleR v0.4.6, Zenodo [code], https://doi.org/10.5281/zenodo.23594053, 2026.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, vol. 32, https://dl.acm.org/doi/10.5555/3454287.3455008, 2019.

Ramadhan, A., Marshall, J., Souza, A., Lee, X. K., Piterbarg, U., Hillier, A., Wagner, G. L., Rackauckas, C., Hill, C., Campin, J.-M., et al.: Capturing missing physics in climate model parameterizations using neural differential equations, arXiv preprint arXiv:2010.12559, https://doi.org/10.48550/arXiv.2010.12559, 2020.

Rehmann, A., Zimmermann, H., and Haefner, D.: Mosaic: a benchmark suite for differntiable physics solvers., https://arxiv.org/abs/2606.27895, 2026.

Sapienza, F., Bolibar, J., Schäfer, F., Groenke, B., Pal, A., Boussange, V., Heimbach, P., Hooker, G., Pérez, F., Persson, P.-O., and Rackauckas, C.: Differentiable programming for differential equations: a review, https://doi.org/10.48550/arXiv.2406.09699, 2024.

Schreck, J. S., Sha, Y., Chapman, W., Kimpara, D., Berner, J., McGinnis, S., Kazadi, A., Sobhani, N., Kirk, B., Becker, C., Gantos, G., and Gagne II, D. J.: Community Research Earth Digital Intelligence Twin: a scalable framework for AI-driven Earth System Modeling, npj Climate and Atmospheric Science, 8, 239, https://doi.org/10.1038/s41612-025-01125-6, 2025.

Segura, H., Bayley, C., Fievét, R., Glöckner, H., Günther, M., Kluft, L., Naumann, A. K., Ortega, S., Praturi, D. S., Rixen, M., Schmidt, H., Winkler, M., Hohenegger, C., and Stevens, B.: A Single Tropical Rainbelt in Global Storm-Resolving Models: The Role of Surface Heat Fluxes Over the Warm Pool, Journal of Advances in Modeling Earth Systems, 17, e2024MS004 897, https://doi.org/https://doi.org/10.1029/2024MS004897, e2024MS004897 2024MS004897, 2025.

Small, R. J. and coauthors: A new synoptic scale resolving global climate simulation using the Community System Earth System Model., J. Adv. in Mod. Earth Sys., 6, 1065–1094, https://doi.org/10.1002/2014MS000363, 2014.

Stoustrup, A., Nuterman, R., Mrozowska, M., and Jochum, M.: Bayesian optimization for Earth system models., J. Adv. Mod. Earth System, submitted, 2026.

Zhang, G. J., Song, X., and Wang, Y.: The double ITCZ syndrome in GCMs: A coupled feedback problem among convection, clouds, atmospheric and ocean circulations, Atmospheric Research, 229, 255–268, https://doi.org/10.1016/j.atmosres.2019.06.023, published 15 November 2019, 2019.