

<https://doi.org/10.1038/s42005-024-01880-7>

Online calibration of deep learning sub-models for hybrid numerical modeling systems

Check for updates

Said Ouala¹ ✉, Bertrand Chapron², Fabrice Collard³, Lucile Gaultier³ & Ronan Fablet¹

Defining end-to-end (or online) training schemes for the calibration of neural sub-models in hybrid systems requires working with an optimization problem that involves the solver of the physical equations. Online learning methodologies thus require the numerical model to be differentiable, which is not the case for most modeling systems. To overcome this, we present an efficient and practical online learning approach for hybrid systems. The method, called EGA for Euler Gradient Approximation, assumes an additive neural correction to the physical model, and an explicit Euler approximation of the gradients. We demonstrate that the EGA converges to the exact gradients in the limit of infinitely small time steps. Numerical experiments show significant improvements over offline learning, highlighting the potential of end-to-end learning for hybrid modeling.

High-fidelity simulations of physical phenomena require intense modeling and computing efforts. Prohibitive computational costs often lead to only resolving certain scales and the resulting scale truncation may question the accuracy and reliability of the simulations. For example, in ocean, atmosphere and climate systems, many physical, biological, and chemical phenomena happen at scales finer than the discretization of numerical models. These unresolved processes generally encompass dissipation effects, but are also responsible for some energy redistribution and backscattering¹⁻³. In practice, parameterizations or sub-models of these phenomena are coupled to the numerical models. These sub-models introduce a significant source of uncertainty and might limit the predictability of large-scale models.

The design of hybrid modeling approaches, based on the combination of a physical core and a deep learning sub-model is under active development⁴⁻¹². Most of these advances are based on offline learning strategies in which the deep learning sub-model is trained independently without interacting with the dynamics. Offline learning is simple to implement and to test, however several works have shown that this strategy can lead to unphysical behaviours. The success of end-to-end learning strategies in pure surrogate modeling applications in weather forecasting¹³⁻¹⁸ suggests that building hybrid models that can be trained end-to-end (or online, as shown in Fig. 1) is likely to become a revolution in numerical modeling and might solve most of the issues with hybrid models that are trained offline. A first example of this revolution is developed in¹², where a full set of primitive equations are implemented in JAX resulting in a fully differentiable Neural Global Circulation Model.

Most state-of-the-art operational numerical models are built on tools that do not support automatic differentiation. For instance, regional ocean

modeling is based on the Regional Oceanic Modeling System (ROMS) and a large community in weather and atmosphere relies on the Weather Research and Forecasting Model (WRF) for regional applications. These tools are popular and have large communities, and it is very unlikely that such community will transition, at least in near future, to new models that support automatic differentiation. In this situation, where the physical core is not differentiable, how to optimize hybrid models end-to-end? In this work, we derive an easy-to-use workflow to perform end-to-end optimization while bypassing the differentiability bottleneck of physical models. Our methodology relies on the additive decomposition of the neural sub-model and on an explicit Euler approximation of the gradient. We prove that the proposed Euler Gradient Approximation (EGA) converges to the exact gradients as the time step tends to zero. We validate our contribution through several numerical experiments, including ocean-atmosphere prototypical eddy-resolved simulations.

Results

Problem formulation

Hereafter, the physical system of consideration is governed by a differential equation of the following form:

$$\partial_t u^\dagger(t, \cdot) = \mathcal{N}(u^\dagger(t, \cdot)) + \mathcal{G}(\cdot) \quad (1)$$

where the state $u_t^\dagger \triangleq u^\dagger(t, \cdot) \in \mathcal{U}^\dagger$ belongs to an appropriate function space \mathcal{U}^\dagger for all times $t \geq 0$. The operator \mathcal{N} represents the natural variability of the system and \mathcal{G} is an external forcing.

This equation describes how a real physical system changes over time. For complex systems such as the atmosphere and the ocean, such

¹IMT Atlantique, Lab-STICC, 29200 Brest, France. ²Ifremer, LOPS, 29200 Brest, France. ³ODL, 29200 Brest, France. ✉e-mail: said.ouala@imt-atlantique.fr

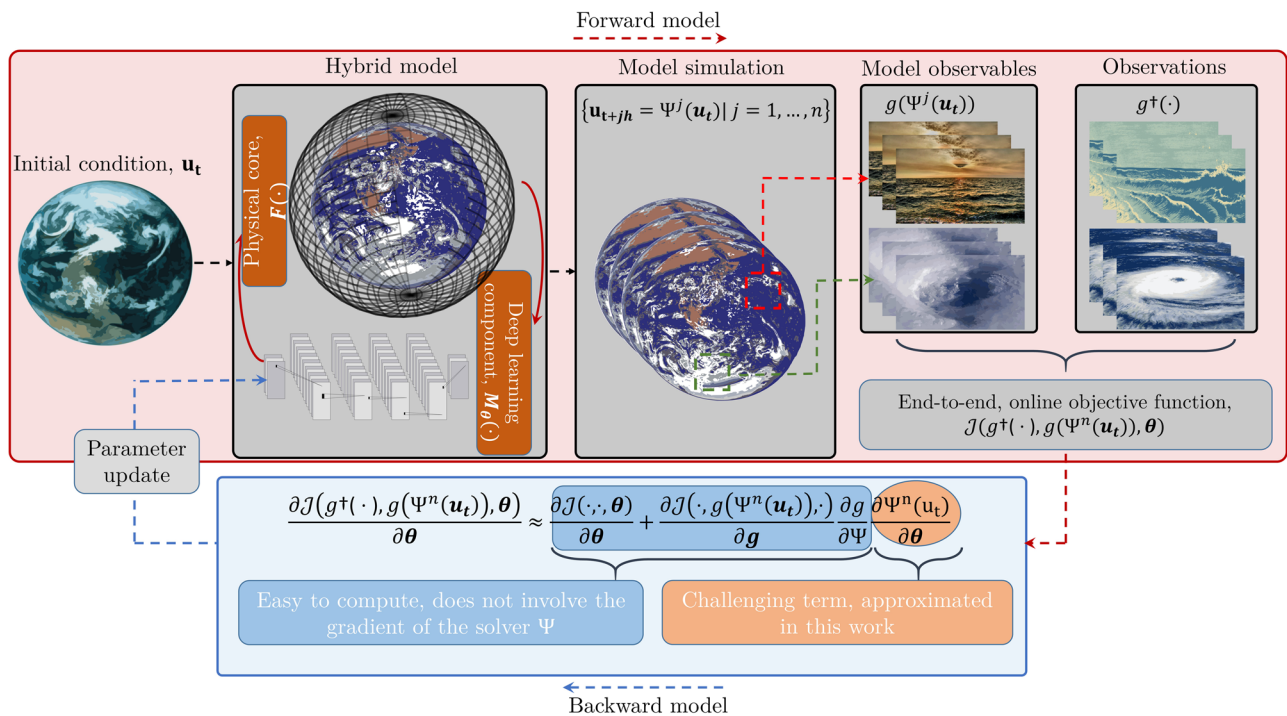


Fig. 1 | Global overview of the online learning problem and link to the proposed EGA. The aim of the proposed framework is to provide a simple and easy-to-use workflow to solve online optimization problems of hybrid models that contain both a physical core and a deep learning component. In online learning, a hybrid model is initialized, and a number of simulation timesteps are computed using a numerical solver. Model observables are then derived from the simulation and

compared to ground truth data. Optimizing the error between these two quantities using gradient-based methods requires solving an optimization problem that involves the gradient of the numerical model, which is unavailable when the model lacks automatic differentiation capabilities. This work proposes practical and efficient methods for approximating this gradient, enabling online learning for hybrid models that are not IA-Native.

models are not available. As an alternative, we must rely on a physical core \mathcal{F} , that is coupled with a number of heuristic process representations, encoded in a sub-model \mathcal{M}_θ ¹⁹. This sub-model \mathcal{M}_θ depends on a vector of parameters $\theta \in \mathbb{R}^a$ that need to be tuned using observables of the true state u^\dagger . Both \mathcal{F} and \mathcal{M}_θ are (potentially) non-linear operators, constrained to produce well-posed solutions of the state $u_t \triangleq u(t, \mathbf{x}) \in \mathcal{U}$ in some appropriate function space \mathcal{U} . The time evolution of this state is described by the following Partial Differential Equation (PDE):

$$\begin{cases} \partial_t u(t, \mathbf{x}) &= \mathcal{F}(u(t, \mathbf{x})) + \mathcal{M}_\theta(u(t, \mathbf{x})) \\ u(0, \mathbf{x}) &= u_0(\mathbf{x}) \end{cases} \quad (2)$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$ is the space variable. The PDE (2) should also be supplied with appropriate boundary conditions on $\partial\Omega$.

Defining and calibrating sub-models is a crucial step in physical simulations. Physics-based sub-models rely on first principles to describe the operator \mathcal{M}_θ . Typical examples can be found in the context of subgrid-scale representations that are based on eddy viscosity assumptions. In such representations, missing scales of motion are assumed to be mainly diffusive and the operator \mathcal{M}_θ becomes a diffusion operator^{20–24}.

Physics-based sub-models have the advantage of being expressed in a continuous form, enabling theoretical validation and ensuring the equations are well-posed under these sub-models. In contrast, machine learning solutions often rely on discrete versions of the system (2), which can pose challenges for theoretical validation. However, empirical evidence continuously demonstrates the interest in using machine learning models to enhance current state-of-the-art physical simulations.

Deep learning and sub-model calibration

For most standard deep learning models, the models and calibration techniques are built after discretizing the governing equations (2):

$$\begin{cases} \dot{\mathbf{u}}_t &= \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t) \\ \mathbf{u}(0) &= \mathbf{u}_0 \end{cases} \quad (3)$$

where $\mathbf{u}_t \triangleq \mathbf{u}(t) \in L \subset \mathbb{R}^{d_u}$ is a discretized version of u_t and \mathbf{F} the corresponding vector field. The sub-model \mathbf{M}_θ is a deep neural network with parameters $\theta \in \mathbb{R}^a$. The boundary conditions of the problem are dropped for simplicity. From this equation, we also define a flow:

$$\mathbf{u}_{t_j} = \phi_{t_j}(\mathbf{u}_t) = \mathbf{u}_t + \int_t^{t_j} (\mathbf{F} + \mathbf{M}_\theta)(\mathbf{u}_w) dw \quad (4)$$

Let us also assume that the numerical integration of (4) is performed using a numerical scheme Ψ that runs for the sake of simplicity with a fixed step size h :

$$\mathbf{u}_{t+nh} = \Psi^n(\mathbf{u}_t) = \underbrace{\Psi \circ \Psi \circ \dots \circ \Psi}_{n \text{ times}}(\mathbf{u}_t) \quad (5)$$

where n is the number of grid points such as $t_j = t + nh$. We assume that the solver Ψ as well as the time step h are defined by some stability and performance criteria that make the resolution of the equation (5) converge to the true solution (4).

In this work, we discuss possible optimization strategies for the calibration of θ . The sub-model \mathbf{M}_θ is assumed to be a deep learning model, and we focus on gradient-based optimization strategies for calibrating its parameters. Note that calibrating the parameters of \mathbf{M}_θ using gradient-free techniques is also possible. For instance by defining a state space model on

the parameters θ and running an ensemble Kalman inversion^{25–27}. However, we focus on gradient based techniques, as they are the most used optimization techniques for calibrating deep learning models. We also assume that there is a single sub-model \mathbf{M}_θ , and that its impact on \mathbf{F} is additive. These assumptions are realistic in several simulation based scenarios, and we provide in the discussion section, possible extensions for non additive sub-models.

Offline learning strategy

The common strategy for optimizing deep learning-based sub-models is to formulate the optimization problem as a supervised regression problem. Provided a reference $\mathbf{R}_t \in \mathbb{R}^{d_u}$ that represents an ideal sub-model response to the input \mathbf{u}_t , the offline learning strategy can be written as the emulation of \mathbf{R}_t using the deep learning model \mathbf{M}_θ . This translates into a supervised regression problem where the objective function can be written as:

$$\mathcal{Q}(\mathbf{R}_t, \mathbf{M}_\theta, \theta) \tag{6}$$

with \mathcal{Q} , a cost function that depends implicitly on θ through \mathbf{M}_θ , and possibly, explicitly when, for instance, regularization constraints on the weights of the sub-models are used. The parameters are typically optimized using a stochastic gradient descent algorithm, and the gradients of the model are computed using automatic differentiation.

Example 1. (Offline learning of subgrid scale sub-models). In subgrid scale modeling, the sub-model \mathbf{M}_θ accounts for unresolved processes due to limited resolution when discretizing the equations^{8,28,29}. In this context, \mathbf{R}_t is defined as the difference between a high-resolution equation and a low-resolution one i.e., assuming some reference discretization at $\mathbb{R}^{d_{u^\dagger}}$ with $d_{u^\dagger} \gg d_u$ in which the continuous equations are perfectly resolved, \mathbf{R}_t is the subgrid-scale term that is defined as follows

$$\mathbf{R}_t = \tau(\mathbf{F}_{d_{u^\dagger}}(\mathbf{u}_t^\dagger)) - \mathbf{F}(\tau(\mathbf{u}_t^\dagger))$$

where \mathbf{u}^\dagger is the true, high resolution, state, $\mathbf{F}_{d_{u^\dagger}}$ the equations defined on the high-resolution grid and τ is a projection from the high-resolution space to the low resolution one. This operation typically includes a filtering of the finer scales in $\mathbb{R}^{d_{u^\dagger}}$ and a coarse-graining to the grid in \mathbb{R}^{d_u} . Given a dataset of N snapshot pairs $\{(\mathbf{R}_{t_k}, \tau(\mathbf{u}_{t_k}^\dagger)) | k = 1, \dots, N\}$, the cost function can be written as:

$$\mathcal{Q}(\mathbf{R}_t, \mathbf{M}_\theta, \theta) = \frac{1}{N} \sum_k \left\| \mathbf{R}_{t_k} - \mathbf{M}_\theta(\tau(\mathbf{u}_{t_k}^\dagger)) \right\|^2 + \mathcal{R}(\cdot) \tag{7}$$

where $\mathcal{R}(\cdot)$ is a regularization term and $\| \cdot \|$ is some appropriate norm (typically L^2) in \mathbb{R}^{d_u} . The optimization problem (7) is typically solved using a stochastic gradient descent algorithm.

The offline optimization problem aims to minimize the discrepancy between the sub-model \mathbf{M}_θ and a reference dataset \mathbf{R}_{t_k} that is considered an ideal response. For several applications that involve making numerical models closer to historical observations, such as weather forecasting, for instance, the complexity of the underlying dynamics and our lack of prior knowledge makes challenging the definition and construction of an offline reference dataset \mathbf{R}_{t_k} . Furthermore, several works^{9,30} showed that deep learning sub-models that are trained offline perform poorly in online tests. These models can lead to numerical instabilities or physically unrealistic flows that can lead to a blow-up of the simulation.

Online learning strategy

In online learning strategies, the sub-model must minimize the cost between a numerical integration of the model (5) and some reference. Formally, let us assume $\mathbf{g}^\dagger \in \mathcal{F} : \mathcal{U}^\dagger \rightarrow \mathbb{R}^m$ to be a real vector valued observable of the true dynamics described in (1). This observable might correspond to real observations of a geophysical state or to some observable of an idealized experiment for which (1) is assumed to be known. To this observable, we

associate a real vector-valued observable of the dynamical system (3) $\mathbf{g} \in \mathcal{F} : L \rightarrow \mathbb{R}^m$. The online learning problem aims at matching these two quantities using an objective function of the following form:

$$\mathcal{J}(\mathbf{g}^\dagger(u_{t+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta) \tag{8}$$

where n is a hyperparameter that corresponds to the number of integration time steps of (5) used in the optimization. Here, the cost function \mathcal{J} is given in discrete time and depends on the solution of the dynamical model (5).

Example 2. (Online learning of subgrid-scale sub-models). Similarly to the previous example, the sub-model \mathbf{M}_θ is assumed to account for unresolved processes due to limited resolution after discretizing the equations. In this context, $\mathbf{u}^\dagger \in \mathbb{R}^{d_h}$ is the true reference high-resolution simulation, and \mathbf{g}^\dagger is some observable on this high-resolution simulation. A natural choice of this observable, e.g.³¹, is a filtering and coarse graining operation, i.e. $\mathbf{g}^\dagger = \tau$. The observable on the low-resolution model \mathbf{g} is a full state observable, i.e., $\mathbf{u} = \mathbf{g}(\mathbf{u})$, and the online loss function can be simplified and defined as the difference between a high-resolution equation and the low-resolution one. Given a dataset of N pairs of time series $\{(\tau(u_{t_k+jh}^\dagger), \tau(u_{t_k}^\dagger)) | k = 1 \dots N \text{ and } j = 1 \dots n\}$, the cost function can write:

$$\mathcal{J}(\cdot) = \frac{1}{N} \sum_k \frac{1}{n} \sum_{j=1}^n \left\| \tau(u_{t_k+jh}^\dagger) - \Psi^j(\tau(u_{t_k}^\dagger)) \right\| + \mathcal{R}(\cdot)$$

The resolution of the optimization problem (8) becomes challenging, as it now involves the resolution of the whole dynamical system (4). Specifically, when solving the online optimization problem, i.e., $\arg \min_\theta \mathcal{J}$, using gradient descent, it is necessary to compute the gradient of the loss (8) with respect to the parameters of the sub-model:

$$\frac{\partial \mathcal{J}}{\partial \theta}(\mathbf{g}^\dagger(u_{t+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta) = \frac{\partial \mathcal{J}(\cdot, \cdot, \theta)}{\partial \theta} + \frac{\partial \mathcal{J}(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \cdot)}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \Psi} \frac{\partial \Psi^n(\mathbf{u}_t)}{\partial \theta} \tag{9}$$

The gradient of the solver Ψ^n must thus be evaluated for every n with respect to the parameters of the sub-model:

$$\frac{\partial \Psi^n(\mathbf{u}_t)}{\partial \theta} = \frac{\partial}{\partial \theta} \Psi \circ \Psi \circ \dots \circ \Psi(\mathbf{u}_t) \tag{10}$$

Computing this gradient requires the solver Ψ to be differentiated with respect to the parameters of the sub-model, which critically limits the use of the online approach in practice. As discussed in the introduction, most large-scale forward solvers in Earth system models (ESM), and digital twin frameworks in Computational Fluid Dynamics (CFD) applications rely on high-performance languages and tools that do not embed automatic differentiation (AD). We discuss in the following section how to approximate (10) in order to solve the online learning problem.

Related works on online learning for hybrid systems

Differentiable emulators In the context of online learning, differentiable emulators³² are considered to both approximate the physical part of (3) and the numerical solver Ψ . These models were successfully tested on data assimilation toy problems³², and can be adapted to the online learning of sub-models^{33,34}.

Let \mathbf{G}_ϕ be a deep learning approximation of \mathbf{F} such that $\hat{\mathbf{u}}_t \approx \mathbf{G}_\phi(\mathbf{u}_t) + \mathbf{M}_\theta$ and let Ψ_α be the numerical scheme with parameters α . The solver Ψ_α can be based on a deep learning model. For the sake of simplicity, the solver Ψ_α is assumed to be an explicit single-step scheme, discretized with a fixed step size h' . The emulator aims at approximating the true solver

(5) i.e.:

$$\begin{aligned} \mathbf{u}_{t_f} = \mathbf{u}_{t_0+nh} &= \underbrace{\Psi \circ \Psi \circ \dots \circ \Psi}_{n \text{ times}}(\mathbf{u}_{t_0}) \\ &\approx \underbrace{\Psi_\alpha \circ \Psi_\alpha \circ \dots \circ \Psi_\alpha}_{r \text{ times}}(\mathbf{u}_{t_0}) \end{aligned} \quad (11)$$

Assuming that r, h', \mathbf{G}_ϕ and Ψ_α are correctly calibrated, the gradients of the true solution with respect to the parameters (10) are replaced by the ones of (11). In practice, \mathbf{G}_ϕ (and occasionally Ψ_α) as well as the sub-model \mathbf{M}_θ can be trained sequentially. For complex models, these emulators, and their solvers, require careful tuning to ensure that the training of the sub-model is not biased by the uncertainty of the emulator³³.

Optimal control and definition of solver-specific backward models

The online optimization problem can be written as an optimal control problem with the following optimization problem:

$$\begin{cases} \arg \min_{\theta} \mathcal{J} \\ \text{s.t.} & \dot{\mathbf{u}}_t = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t) \end{cases} \quad (12)$$

A Lagrangian of this optimization problem can then be written as:

$$\mathcal{L} = \mathcal{J} + \int_t^{t+nh} \lambda_t (\dot{\mathbf{u}}_t - \mathbf{F}(\mathbf{u}_t) - \mathbf{M}_\theta(\mathbf{u}_t)) dt \quad (13)$$

where λ_t is a time-dependent Lagrange multiplier. The gradients of the online objective function with respect to the parameters are identical to the ones of the Lagrangian. The optimal control problem is here formulated in continuous time, but a corresponding discrete time control can also be defined based on the discretized dynamics (5).

Interpreting this machine learning problem as an optimal control provides multiple methods to build solutions of the online learning problem. Overall, optimal control methods can be divided into direct and indirect approaches³⁵. Direct approaches, *discretize then optimize* methods, are based on a discrete dynamical model constraint. Indirect approaches, *optimize then discretize* techniques, work on the continuous optimal control problem (12). The latter technique was advertised in³⁶ as a potentially efficient methodology for the optimization of neural ODE models.

In practice, one significant drawback of employing optimal control methods lies in the complexity of designing the adjoint model. It necessitates a domain-specific treatment based on the underlying dynamics of the physical equations, which might limit the use of this strategy in the context of online learning of deep learning sub-models.

Online learning using gradient-free methods The computational burden associated with evaluating gradients of numerical solvers has motivated the development of gradient-free optimization methods for addressing online learning problems in hybrid numerical models. These methods are based on the definition of a state-space model in the parameter space. Parameter estimation is then formulated as an inverse problem and estimates are derived using standard inversion techniques based on observations. Among these methods, Ensemble Kalman Inversion (EKI) is a widely used, state-of-the-art technique that has been explored in recent related works^{37,38}.

Gradient-free methods are particularly appealing since they avoid the need to compute gradients of the hybrid model when solving online learning problems. However, they require careful tuning of the inversion scheme, a task that becomes increasingly complex as the number of parameters grows, which is often the case when using deep learning-based sub-models.

Euler gradient approximation (EGA) for the online learning problem

We aim to define a relevant, easy-to-use workflow for solving the online optimization problem in hybrid modeling systems where the physical model is not differentiable. In order to do so, we introduce the EGA, which relies on

an explicit Euler discretization for the computation of the gradients, i.e. the backward pass in deep learning.

Recall that the online cost function (8):

$$\mathcal{J}(\mathbf{g}^\dagger(u_{t_0+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta)$$

depends explicitly on the non-differentiable solver Ψ , that runs (for the sake of simplicity) with a fixed step size h :

$$\mathbf{u}_{t_f} = \mathbf{u}_{t_0+nh} = \Psi^n(\mathbf{u}_t) = \underbrace{\Psi \circ \Psi \circ \dots \circ \Psi}_{n \text{ times}}(\mathbf{u}_t)$$

As discussed in the previous section, the resolution of the online optimization problem requires the computation of the gradient of the online objective with respect to the parameters of the sub-model. By using the chain rule, it is trivial to show that this gradient depends on the gradient of the solver Ψ i.e.:

$$\frac{\partial \mathcal{J}}{\partial \theta}(\mathbf{g}^\dagger(u_{t_0+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta) = \frac{\partial \mathcal{J}(\cdot, \cdot, \theta)}{\partial \theta} + \frac{\partial \mathcal{J}(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \cdot)}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \Psi} \frac{\partial \Psi^n(\mathbf{u}_t)}{\partial \theta}$$

In order to solve the online optimization problem, we aim to find an efficient approximation of the gradient of the solver Ψ with respect to the parameters of the sub-model. Let us consider an explicit Euler solver Ψ_E , a single integration step of (3) using Ψ_E can be written as:

$$\mathbf{u}_{t+h} = \Psi_E(\mathbf{u}_t) \quad (14)$$

where $\Psi_E(\mathbf{u}_t) = \mathbf{u}_t + h(\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t))$.

We aim at approximating the gradient of the solver Ψ , using the ones of the Euler solver. In this context, and in order to keep track of the order of convergence of the gradient, we introduce the following proposition.

Proposition 1. Assuming that the solver Ψ has order $p \geq 1$, we have for any initial condition \mathbf{u}_t :

$$\begin{aligned} \mathbf{u}_{t+h} &= \Psi(\mathbf{u}_t) \\ &= \Psi_E(\mathbf{u}_t) + O(h^2) \end{aligned} \quad (15)$$

In particular, every solution at an arbitrary time $t_f = t + nh$ can be written as:

$$\begin{aligned} \mathbf{u}_{t_f} = \mathbf{u}_{t_0+nh} &= \Psi^n(\mathbf{u}_t) \\ &= \Psi_E(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \end{aligned} \quad (16)$$

By using the proposition 1, we can show that the gradient of the solver Ψ can be written as a function of the gradient of the Euler solver plus a bounded term.

Theorem 1. (EGA): Under the same conditions as proposition 1, and assuming that the number of time steps n is fixed and corresponds to a hyperparameter of the online learning problem, the gradient of the solver Ψ can be written as follows:

$$\begin{aligned} \frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) &= \sum_{j=1}^{n-1} \prod_{i=1}^{i=n-j} \underbrace{\frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)}}_{\text{Jacobian of the flow}} h \underbrace{\frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t))}_{\text{Gradient of the sub-model}} \\ &+ h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \end{aligned} \quad (17)$$

Corollary 1.1. Under the same conditions as proposition 1, and if we assume that the online learning problem is defined for a given initial and finite times t_0 and t_f such that $n = \frac{t_f - t_0}{h}$, then the convergence of the EGA

becomes linear in h i.e.:

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \quad (18)$$

The difference between the EGA formulation of Theorem 1 and the one of Corollary 1.1 lies in the convergence rate of the accumulated error in the gradient approximation. Specifically, we show in Supplementary Note A.2 (see equation (S.I. 5)) that this error accumulation term is bounded by $nO(h^2)$. In Theorem 1, the number of time steps n is assumed to be fixed, resulting in the error accumulation term being bounded by $O(h^2)$. In Corollary 1.1, n increases as h decreases, i.e., $n = (t_f - t_0)/h$, so the error accumulation is only bounded by $O(h)$.

Here, we assume for simplicity that the Euler solver runs at the same time step as the one of the solver Ψ . In practice and as shown in the experiments, these solvers might have different time steps and our methodology only requires training trajectories of the solver Ψ to be sampled at the time step of the Euler solver.

Both theorem 1 and corollary 1.1 show that the gradient of the solver Ψ can be approximated by knowing two terms, the Jacobian of the flow as well as the gradient of the sub-model with respect to the parameters θ at some simulation time step of the solver $\mathbf{u}_{t_k+jh} = \Psi^j(\mathbf{u}_t)$ with $j = 1, \dots, n - 1$. While the gradient of the sub-model with respect to the parameters can be computed for every input \mathbf{u}_{t_k+jh} using automatic differentiation, the Jacobian of the flow needs to be specified or approximated.

EGA Jacobian approximation

The Jacobian in the EGA equations (17) and (18) need to be provided in order to evaluate the gradients of the solver Ψ with respect to the parameters. In practice, several techniques can be used, we discuss in this work three techniques based on a static formulation of the Jacobian, on the use of a tangent linear model and on an ensemble approximation.

Static formulation (Static-EGA). A static formulation of the Jacobian corresponds to approximating the Jacobian term in (17) and (18) by the identity matrix. It can be shown that a static formulation of the Jacobian matrix implies corollary 1.2.

Corollary 1.2. (Static-EGA). Under the same conditions as proposition 1, and assuming that the number of time steps n is fixed, and corresponds to a hyperparameter of the online learning problem, the gradient of the solver Ψ can be written as the one of the Euler solver as follows:

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + O(h^2) \quad (19)$$

If we assume that the number of time steps varies with, h i.e., that the online learning problem is defined for a given initial and finite times t_0 and t_f such that $n = \frac{t_f - t_0}{h}$ that the error term bound becomes is linear in h i.e.:

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + O(h) \quad (20)$$

This static approximation provides a simple, easy-to-use, approximation of the gradient of the solver with respect to the parameters θ . Furthermore, and similarly to the gradients in (17) and (18), the order of convergence of the gradients in (19) and (20) is quadratic and linear, respectively. However, the constant of convergence of this approximation is larger than the approximations in (17) and (18) due to the presence of additional quadratic terms in the $O(h^2)$. In this context, improving the

precision of the gradient approximation requires either reducing the time step h , or providing a better approximation of the Jacobian matrix.

Tangent linear model (TLM) approach (TLM-EGA). The rise of variational data assimilation techniques motivated the development of Tangent Linear Models for several physical models. This TLM corresponds to the Jacobian of the solver Ψ that operates only on the physical core \mathbf{F} , without a sub-model \mathbf{M}_θ . Let us call this solver Ψ_o and it corresponds to the time discretization of the following integral:

$$\mathbf{u}_{t_f} = \Psi_o^n(\mathbf{u}_{t_0}) \approx \mathbf{u}_t + \int_t^{t_f} \mathbf{F}(\mathbf{u}_w) dw \quad (21)$$

The tangent linear model of Ψ_o can be defined simply, for every initial condition \mathbf{u}_t as the Jacobian of Ψ_o i.e.:

$$TLM_{\Psi_o}(\mathbf{u}_t) = \frac{\partial \Psi_o(\mathbf{u}_t)}{\partial \mathbf{u}_t} \quad (22)$$

In order to use this TLM in the gradient defined in (17) and (18), we need to add the variation that is due to the sub-model \mathbf{M}_θ . Since the sub-model is additive, we can write the following result.

Corollary 1.3. (Extended TLM for Hybrid systems). Under the same conditions of proposition 1, and assuming that the order of both solvers Ψ and Ψ_o is p , and given the TLM of Ψ_o . We can write the Jacobian of Ψ as follows:

$$TLM_{\Psi}(\mathbf{u}_t) = \frac{\partial \Psi(\mathbf{u}_t)}{\partial \mathbf{u}_t} = TLM_{\Psi_o}(\mathbf{u}_t) + \frac{\partial}{\partial \mathbf{u}_t} \sum_{k=1}^{k=p} \frac{h^k}{k!} \mathbf{M}_\theta(\mathbf{u}_t)^{(k-1)} + O(h^{p+1}) \quad (23)$$

Since all the reminder terms in the results showed in (17) and (18) are bounded by at most $O(h^2)$, a sufficient approximation of the Jacobian term based on the TLM of Ψ_o requires evaluating the Jacobian of \mathbf{M}_θ only up to $k = 1$ i.e.:

$$TLM_{\Psi}(\mathbf{u}_t) = \frac{\partial \Psi(\mathbf{u}_t)}{\partial \mathbf{u}_t} = TLM_{\Psi_o}(\mathbf{u}_t) + h \frac{\partial \mathbf{M}_\theta(\mathbf{u}_t)}{\partial \mathbf{u}_t} + O(h^2) \quad (24)$$

We recall that the Jacobian of \mathbf{M}_θ can be computed efficiently using automatic differentiation.

Ensemble tangent linear model (ETLM) approximation (ETLM-EGA). The Jacobian of the flow in both (17) and (18) can be approximated by using an ensemble. This formulation is widely employed in data assimilation, and it is documented in numerous state-of-the-art works. For completeness, we provide a brief overview here, which is largely based on the work of³⁹.

For every initial condition \mathbf{u}_b , we start by constructing a K -member ensemble i.e. $\{\mathbf{u}_t^i | i = 1 \dots K\}$. These initial conditions are then propagated by the solver Ψ to compute the ensemble prediction:

$$\mathbf{u}_{t+h}^i = \Psi(\mathbf{u}_t^i) \text{ for } i = 1, \dots, K \quad (25)$$

Perturbations are constructed relative to the ensemble mean (indicated by $\bar{\mathbf{u}}_t$), $\delta \mathbf{u}_t^i = \mathbf{u}_t^i - \bar{\mathbf{u}}_t$ and $\delta \mathbf{u}_{t+h}^i = \mathbf{u}_{t+h}^i - \bar{\mathbf{u}}_{t+h}$, and these are assembled into $d_u \times K$ matrices, $\delta \mathbf{U}_t$ and $\delta \mathbf{U}_{t+h}$, representing ensemble perturbations listed column-wise for times t and $t + h$, respectively. Based on these ensemble perturbations, the Jacobian matrix can be approximated as the best linear fit between $\delta \mathbf{U}_t$ and $\delta \mathbf{U}_{t+h}$:

$$\frac{\partial \Psi(\mathbf{u}_t)}{\partial \mathbf{u}_t} \approx ETLM = \delta \mathbf{U}_{t+h} \delta \mathbf{U}_t^T [\delta \mathbf{U}_t \delta \mathbf{U}_t^T]^{-1} \quad (26)$$

where T denotes the matrix transpose, and -1 represents the matrix inverse.

The ensemble approximation of the Jacobian is particularly valuable when missing a tangent linear model, as it provides a simple data-driven approximation of the sensitivity of the flow to initial conditions. However, When utilizing an ensemble approximation, a particularly challenging aspect arises when dealing with high-dimensional systems, where the system dimension d_u , is significantly larger than the number of ensemble members K . In such situations, a good calibration of the initial perturbation is mandatory in order to produce an informative ensemble.

Implementation in deep learning frameworks

Based on the above formulations, one can derive approximations for the gradient of the online cost (8). This involves substituting the gradient of the solver in (9) with any of the provided methods and neglecting the $O(\cdot)$ term.

$$\frac{\partial \mathcal{J}}{\partial \theta} (\mathbf{g}^\dagger(u_{t+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta) = \mathbf{v} + \mathbf{w} \mathbf{A}_{l,p} + O(h^p) \quad (27)$$

where $\mathbf{v} = \frac{\partial \mathcal{J}(\cdot, \theta)}{\partial \theta}$ represents the gradient of the regularization, $\mathbf{w} = \frac{\partial \mathcal{J}(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \cdot)}{\partial \mathbf{g}}$ is the gradient of the online cost with respect to the solver and $\mathbf{A}_{l,p}$ is the approximation of the gradient of the solver with respect to the parameters. The subscript p represents the order of approximation of the gradient, it also implicitly specifies the training methodology adopted for the number of time steps n . In particular, if $p = 2$, the number of steps n is fixed and is considered as a training hyperparameter. If $p = 1$, it means that n is deduced from a given initial and finite time. The subscript l represents the methodology used to compute the Jacobian of the flow:

$$\mathbf{A}_{l,p} = \sum_{j=1}^{j=n-1} \mathbf{J}_{j,l} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t))$$

where $\mathbf{J}_{j,l} = (\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{i-1}(\mathbf{u}_t))}{\partial \Psi^{i-1}(\mathbf{u}_t)})$ is defined from a Jacobian approximation as follows:

$$\frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} = \begin{cases} \frac{\Psi(\Psi^{n-1}(\mathbf{u}_t))}{\Psi^{n-1}(\mathbf{u}_t)}, & \text{if } l = 1 \text{ (EGA)} \\ \mathbf{I}, & \text{if } l = 2 \text{ (Static - EGA)} \\ TLM_\Psi, & \text{if } l = 3 \text{ (TLM - EGA)} \\ ETLM, & \text{if } l = 4 \text{ (ETLM - EGA)} \end{cases} \quad (28)$$

The implementation of (27) in programming languages that support automatic differentiation requires evaluation of the gradient of the sub-model \mathbf{M}_θ for inputs $\Psi^j(\mathbf{u}_t)$ that are issued from the non-differentiable solver Ψ . These gradients are multiplied with the Jacobian of the flow $\mathbf{J}_{j,l}$ to produce an estimate of the gradient of the solver. The evaluation of the gradient can be based on composable function transforms⁴⁰, or on the modification of the backward call in standard automatic differentiation languages. In Supplementary Note D, we provide algorithms that illustrate practical implementations of these two methodologies.

EGA for an improved computational cost in training long roll-outs

Increasing the number of training steps n strongly influences the inference accuracy of both surrogate physical simulators^{41,42} and hybrid models^{12,31,43}. When large number of time steps are required, the EGA can be used to reduce the computational complexity of the computational graph in the training phase. For instance, if we assume that the numerical solver requires N Number of Function Evaluations (NFE) per time step, the EGA can be used with a first order Jacobian approximation and would require only backpropagation through a single function evaluation. This allows to have larger roll-outs at a smaller computational cost. We evaluate the ability of using the EGA on fully differentiable hybrid models to reduce the time and memory complexity of the backward pass in experimental subsection.

Experiments

Numerical experiments are performed to evaluate the proposed online learning techniques. We first validate numerically the order of convergence of some results developed in the previous section. We also evaluate the performance of the EGA when compared to fully differentiable hybrid models in terms of memory and time complexity of the backward pass. We then evaluate sub-models trained online with the proposed gradient approximations. These sub-models are benchmarked against state-of-the-art training techniques, including online learning with the exact gradient (when assuming that the gradient of the solver is available). The comparison of the sub-models is done principally on online metrics, where we evaluate the hybrid system (the physical core coupled to the deep learning sub-model) in simulating trajectories that are realistic with respect to some reference data. We also compare, when relevant, offline metrics, where only the output of the sub-model is evaluated (without considering the coupling to the physical core). We consider two case studies on the Lorenz-63 and quasi-geographic dynamics. Additional experiments on the multiscale Lorenz 96 system⁴⁴, as well as the details on the parameterization of the deep learning sub-models, training data, objective functions, and baseline methods are given in Methods.

Lorenz 63 system

The Lorenz 63 dynamical system is a 3-dimensional model of the form:

$$\begin{aligned} \dot{u}_{t,1}^\dagger &= \sigma(u_{t,2}^\dagger - u_{t,1}^\dagger) \\ \dot{u}_{t,2}^\dagger &= \rho u_{t,1}^\dagger - u_{t,2}^\dagger - u_{t,1}^\dagger u_{t,3}^\dagger \\ \dot{u}_{t,3}^\dagger &= u_{t,1}^\dagger u_{t,2}^\dagger - \beta u_{t,3}^\dagger \end{aligned} \quad (29)$$

Under parametrization $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$, this system exhibits chaotic dynamics with a strange attractor⁴⁵.

We assume that we are provided with \mathbf{F} , an imperfect version of the Lorenz system (29) that does not include the term $\beta u_{t,3}^\dagger$. This physical core is supplemented with a sub-model \mathbf{M}_θ as follows:

$$\dot{\mathbf{u}}_t = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t) \quad (30)$$

where $\mathbf{u}_t = [u_{t,1}, u_{t,2}, u_{t,3}]^T$ and $\mathbf{F} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is given by:

$$\begin{aligned} F_1(\mathbf{u}_t) &= \sigma(u_{t,2} - u_{t,1}) \\ F_2(\mathbf{u}_t) &= \rho u_{t,1} - u_{t,2} - u_{t,1} u_{t,3} \\ F_3(\mathbf{u}_t) &= u_{t,1} u_{t,2} \end{aligned} \quad (31)$$

The sub-model \mathbf{M}_θ is a fully connected neural network with parameters θ .

Analysis of the order of convergence. Here, we present a numerical validation of the convergence order for both the EGA and Static-EGA results developed in the previous section. We specifically focus on the results of the theorem 1 equation (17) and the one of the corollary 1.2 equation (19). In this experiment, we evaluate the error of the proposed gradient approximations when the time step h varies from 10^{-1} to 10^{-4} . This time step will correspond to the one used by the Euler approximation of the gradient, and not to the time step of the forward solver Ψ . Regarding the solver Ψ , we use in this experiment a differentiable adaptive step size solver Ψ (DOPRI8 used in³⁶). This solver allows us to compute the exact gradient of the online cost function using automatic differentiation.

Figure 2 shows the value of the gradient error for different values of h . Overall, the error of the proposed approximations behave as second order. This result validates numerically the development of theorem 1 and, importantly, the result of the Static-EGA (corollary 1.2 equation (19)), that will be used in the following experiments.

Analysis of the performance of the EGA. We present an evaluation of the memory and time complexity of the EGA with respect to backpropagation through the entire computational graph of the numerical solver. We assume here that the numerical solver Ψ is a Runge-Kutta 4 (RK4) solver (Please refer to Supplementary Note C for a similar analysis

on the DOPRI8 solver), and we compare the time and memory usage of the EGA with respect to standard backpropagation through Ψ , for an increasing number of solver steps n . We study the performance of both the EGA formulations defined in equation (17) (with a first order Jacobian) and the Static EGA defined in equation (19).

The panel (a) of Fig. 3 shows that the EGA leads to an average speedup in the computation time of the gradients by factor of 4 with respect to the full computational graph. The size of the computational graph given in panel (b) of Fig. 3 is also divided roughly by the same factor. Panel (c) of Fig. 3 presents the gradient error of the EGA relative to the absolute value. This error indicates that the static EGA, which misses Jacobian information, fails to provide accurate gradients after $n = 10$ simulation steps, with errors exceeding 50% of the gradient value. However, the EGA with Jacobian information maintains error levels below 33% across all simulation steps. These experiments highlight the potential of EGA for supporting the training of hybrid models or surrogates for physical simulators over long simulation times.”

Submodel performance. In this experiment, we evaluate the performance of the proposed online learning techniques in deriving a relevant sub-model M_θ . We assume that we are provided with a dataset of the true system in (29), $\mathcal{D}_h = \{(u_{t_k+jh_i}^\dagger, u_{t_k}^\dagger) \mid k = 1 \dots N \text{ and } j = 1 \dots n\}$ and we aim at optimizing the parameters of the sub-model M_θ to minimize the cost of the online objective function (8). We evaluate two of the proposed online learning schemes, the Static-EGA given in Eq. (19) and the ETLM-EGA in which the Jacobian is approximated using an ensemble (26). These schemes are compared to the simulations based on a sub-model that is calibrated using the exact gradient of the online objective. We also compare these models to the physical core, that we run without any correction.

A qualitative analysis of both the simulation and short-term forecasting performance of the tested models is given in Fig. 4. Overall, all the tested models are able to significantly improve the physical core, leading to a significant decrease in the forecasting error (as shown in panel (b)), while also reproducing the Lorenz 63 attractor (as highlighted in panel (a)). This

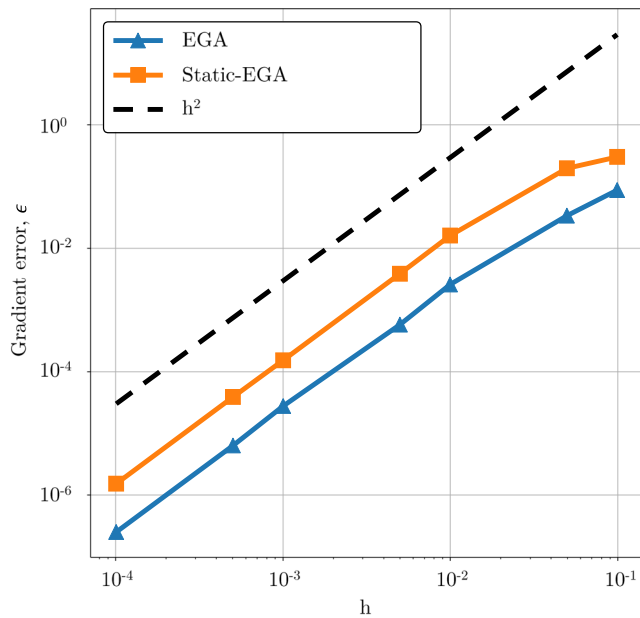


Fig. 2 | Order of convergence of the proposed Euler gradient approximations. We provide experimental validation of the gradient based on both the EGA (17) and Static-EGA (19) formulas. The error is computed by comparing these gradients to the exact ones, returned using automatic differentiation of the solver Ψ . We recall that in the EGA, the Jacobian of the flow is computed exactly using automatic differentiation. The dashed line correspond to the h^2 slop and indicates the order of convergence of the proposed methods.

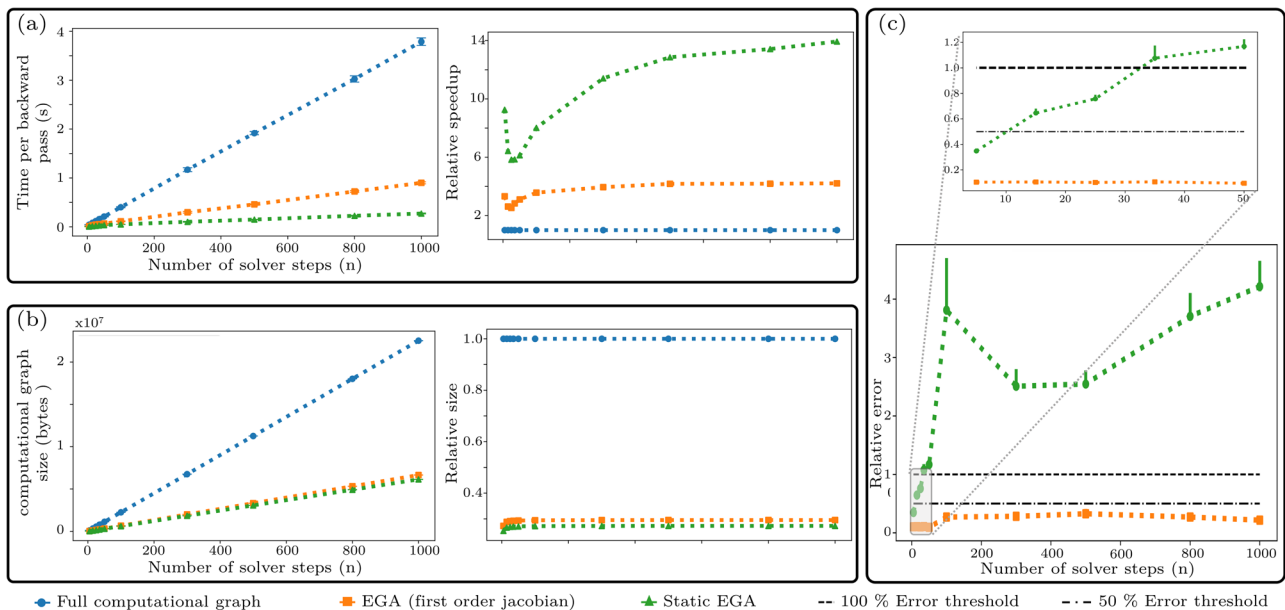


Fig. 3 | Analysis of the computational complexity of the EGA with respect to backpropagation through the RK4 numerical solver. **a** Wall time per backward pass and speedup relative to the standard backpropagation through the numerical solver. **b** Size of the computational graph and relative size with respect to standard backpropagation. **c** Relative error of the EGA with respect to backpropagation through the numerical solver. The inset provides a zoomed-in view of the relative

error for smaller values of n (up to $n = 50$), highlighting the behavior of the methods within the 100% and 50% error thresholds. The Error is averaged over the training data samples and error bars correspond to the standard deviation (size of the error bars of the figures in panel c was divided by 50). This benchmark was executed on a single NVIDIA GeForce GTX 1080 Ti GPU.

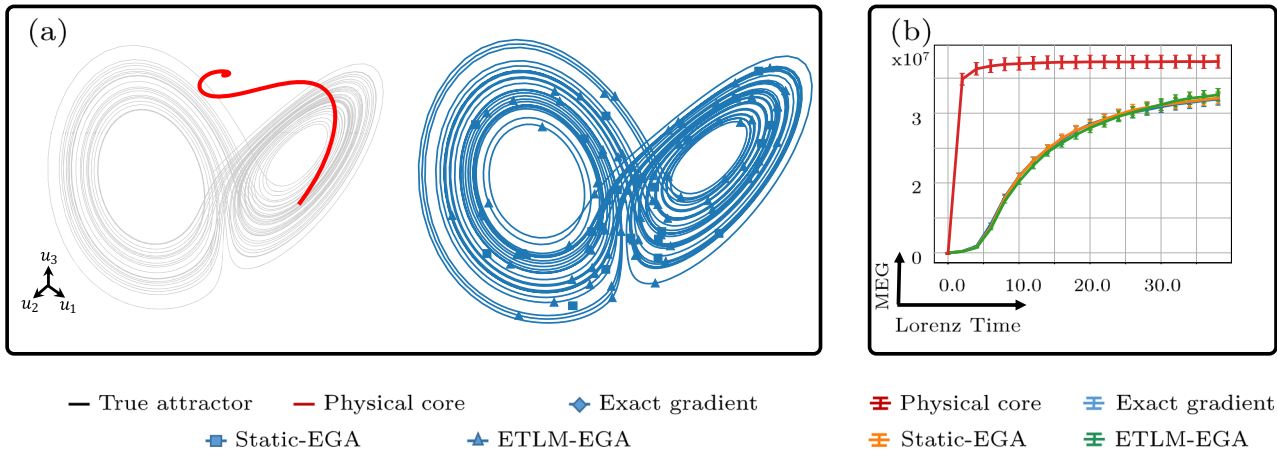


Fig. 4 | Qualitative analysis of the tested models in the Lorenz 63 experiment. **a** The attractor of the true Lorenz 63 model is compared to both the physical core (without the neural network correction term M_θ) and to the hybrid models (with the neural network corrections). The sub-models are optimized online using stochastic gradient descent, and the gradients of the online function are computed either

exactly (using automatic differentiation) or using some of the proposed approximations. **b** Mean error growth (MEG) of the tested models. The mean and standard deviation of the MEG are computed based on an ensemble of 20 trajectories issued from 5 different runs. The error bars are scaled by 1/20.

Table 1 | Simulation performance the data-driven models

Model	Exponents	Dimension
True gradient	(0.90 -0.01 -14.57) ± (0.02 0.01 0.02)	2.061 ± 0.001
Static approximation	(0.91 -0.01 -14.57) ± (0.02 0.01 0.02)	2.061 ± 0.001
Ensemble approximation	(0.91 -0.01 -14.56) ± (0.02 0.01 0.02)	2.061 ± 0.001
Physical core	(-0.03 -4.99 -5.98) ± (0.01 1.43 1.42)	0

full Lyapunov spectrum and Lyapunov dimension of the tested models. The Lyapunov spectrum of the true Lorenz 63 system is (0.91, 0.0, -14.57) and it's dimension is estimated to be 2.064⁵⁷. The Lyapunov spectrum was estimated using the Gram-Schmidt orthonormalization technique, starting from initial conditions in the test set. The reported values correspond to the average Lyapunov spectrum after convergence, with the errors representing the standard deviation.

finding is also validated through the computation of the Lyapunov spectrum and the Lyapunov dimension of the tested models, Table 1. All the training schemes examined in this study yield simulations that closely align with the true underlying dynamics, which confirms the effectiveness of the proposed Euler approximations, for solving the online learning problem.

Quasi-geostrophic turbulence

Quasi-geostrophic dynamics. In this second experiment, we consider Quasi-Geostrophic (QG) dynamics. QG theory is a workhorse to study geophysical fluid dynamics, relevant when the fluid follows an hydrostatic assumption and for which the Coriolis acceleration balances the horizontal pressure gradients.

Over a doubly periodic (x, y) square domain with length $L = 2\pi$, the dimensionless governing equations in the vorticity (ω_t) and streamfunction (ψ_t) are:

$$\frac{\partial \omega_t}{\partial t} + \mathcal{A}(\omega_t, \psi_t) = \frac{1}{\text{Re}} \nabla^2 \omega_t - f - r \omega_t \tag{32a}$$

$$\nabla^2 \psi_t = -\omega_t \tag{32b}$$

where, $\mathcal{A}(\omega_t, \psi_t)$ represents the nonlinear advection term:

$$\mathcal{A}(\omega_t, \psi_t) = \frac{\partial \psi_t}{\partial y} \frac{\partial \omega_t}{\partial x} - \frac{\partial \psi_t}{\partial x} \frac{\partial \omega_t}{\partial y},$$

and f represents a deterministic forcing⁴⁶:

$$f(x, y) = k_f \left[\cos(k_f x) + \cos(k_f y) \right].$$

Large eddy simulation setting. In this experiment, we study the development of a sub-model that accounts for unresolved subgrid scale effects on the coarsened resolution of the QG equations (32). We are specifically interested in a Large Eddy Simulation (LES) setting in which the vorticity and streamfunction are filtered using a Gaussian filter⁴⁷, denoted by $(\bar{\cdot})$. Applying this filter to Eqs. (32a)-(32b) yields:

$$\frac{\partial \bar{\omega}_t}{\partial t} + \mathcal{A}(\bar{\omega}_t, \bar{\psi}_t) = \frac{1}{\text{Re}} \nabla^2 \bar{\omega}_t - \bar{f} - r \bar{\omega}_t + \underbrace{\mathcal{A}(\bar{\omega}_t, \bar{\psi}_t) - \overline{\mathcal{A}(\omega_t, \psi_t)}}_{\Pi_t \approx M_\theta} \tag{33a}$$

$$\nabla^2 \bar{\psi}_t = -\bar{\omega}_t \tag{33b}$$

When compared to the direct numerical simulation (DNS) of equations (32a)-(32b), the LES can be solved at a coarser resolution. Yet, the term Π_t encoding the Sub Grid Scale (SGS) variability requires a closure procedure. In this experiment, the proposed online learning techniques must recover a sub-model that accounts for this SGS term. This sub-model is a deep learning Convolutional Neural Network (CNN) $M_\theta(\bar{\psi}_t, \bar{\omega}_t)$ that takes as inputs both the vorticity and streamfunction fields $(\bar{\psi}_t, \bar{\omega}_t)$.

We use different random vorticity fields as initial conditions to generate 14 Direct Numerical Simulation (DNS) trajectories. These DNS data are then filtered to the resolution of the LES simulation and used as training, validation, and testing datasets. In this experiment, we compare the proposed online learning strategy with a static approximation of the Jacobian to both online learning with an exact gradient and offline learning schemes. These learning-based approaches are also compared to a standard, physics-based, dynamic Smagorinsky (DSMAG) parameterization²¹, in which the diffusion coefficient is constrained to be positive³¹ in order to avoid numerical instabilities related to energy backscattering^{7,31}.

Table 2 | Correlation coefficients between the predicted and true subgrid scale term

Online, true gradient	Online, Static-EGA	Offline	DSMAG
$0.883 \pm 8.550 \times 10^{-4}$	$0.881 \pm 5.650 \times 10^{-3}$	$0.930 \pm 2.009 \times 10^{-3}$	0.243

The correlation coefficients of the learning-based methods was averaged over 5 model runs.

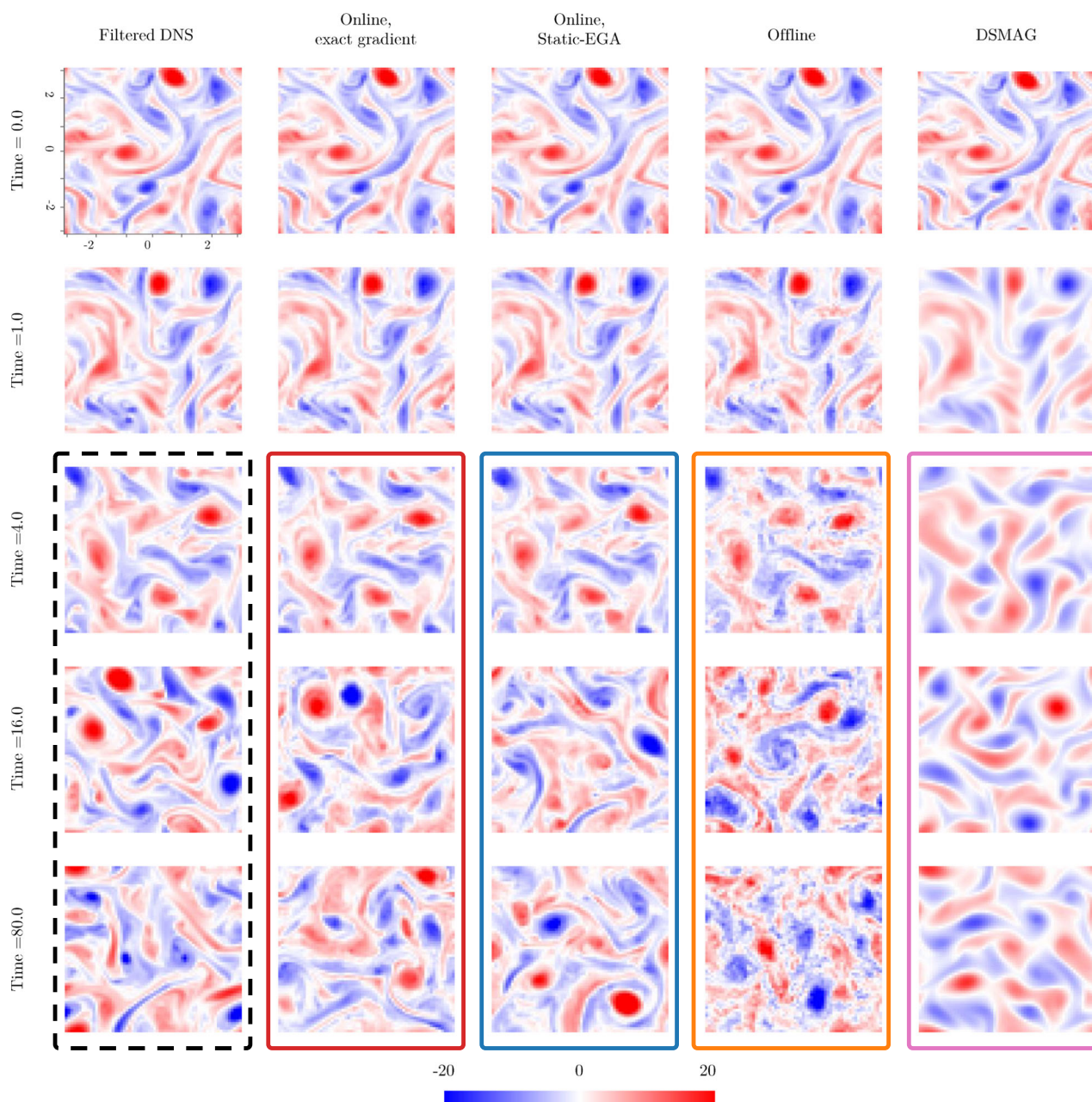


Fig. 5 | Vorticity field simulation example for the different models. Time evolution of the vorticity field for the different models, starting from the same initial condition.

Offline analysis. We first examine the accuracy of the deep learning sub-models in predicting the subgrid-scale term Π_t for never-seen-before samples of $(\bar{\psi}_t, \bar{\omega}_t)$ within the testing set. We use a commonly used metric^{8,31}, the correlation coefficient c between the modeled $M_\theta(\bar{\psi}_t, \bar{\omega}_t)$ and true Π_t SGS terms.

Table 2 shows the correlation coefficients, averaged over two model runs and 1000 testing samples, for both the deep learning sub-models and the DSMAG baseline. Consistent with the previous findings³¹, offline tests show that the data-driven SGS models substantially outperform DSMAG with a correlation coefficient c above 0.8. We also validate, similarly to

previous works³¹ that offline learning performs better on offline metrics than online learning-based models.

Online analysis. Here, we evaluate the ability of the trained hybrid models to reproduce the dynamics of the filtered DNS simulation. Figure 5 shows a simulation example from an initial condition in the test set. A visual analysis reveals that the DSMAG scheme (purple panel in Fig. 5) smooths out fine-scale structures. This model is intrinsically built on a diffusion assumption which enables the system to sustain large-scale variability at the cost of excessively smoothing small-scale features.

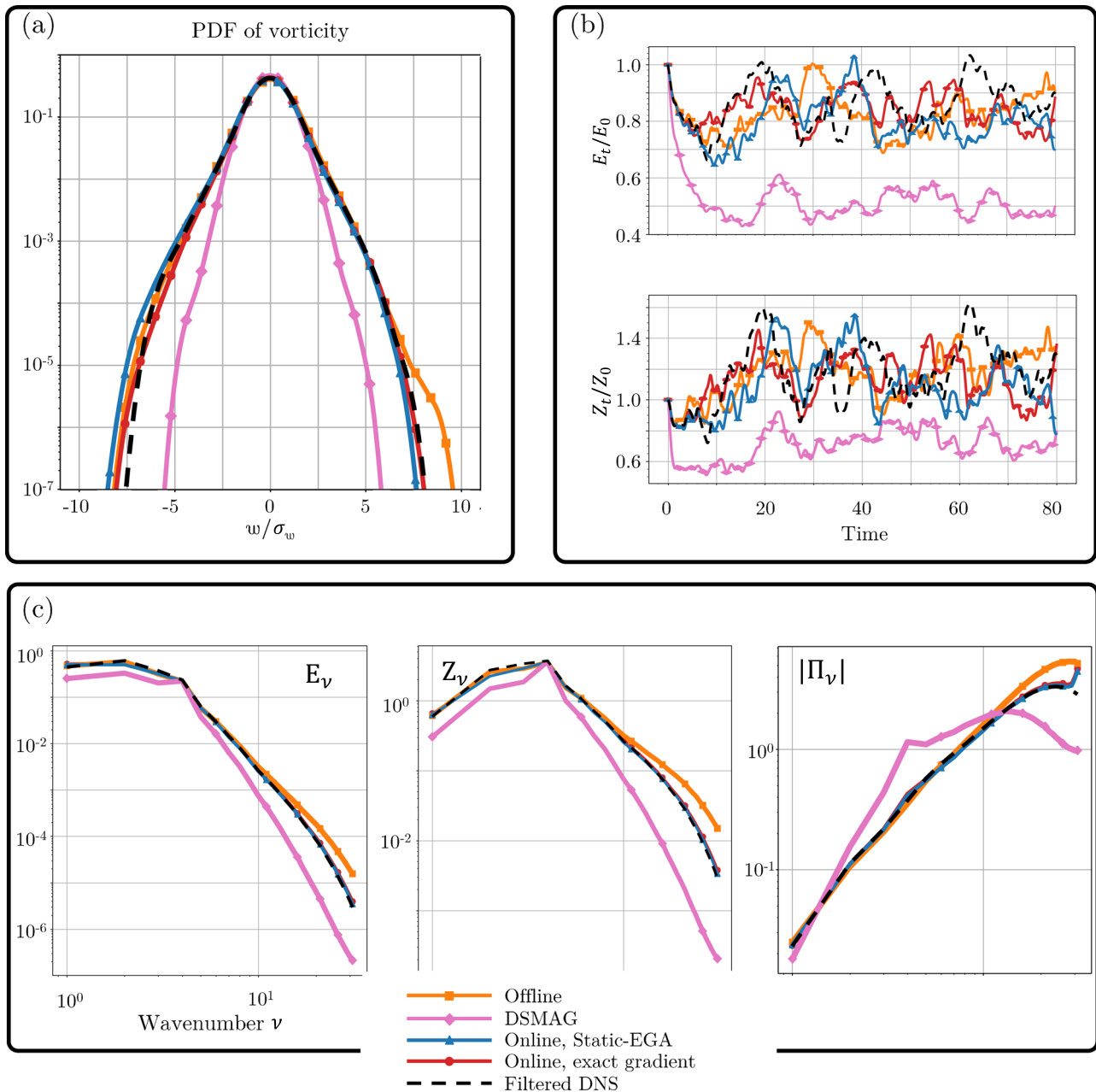


Fig. 6 | Statistical evaluation of the different models. **a** Probability density function of the vorticity field **(b)** Time evolution of the kinetic energy $E_t = \frac{1}{2} \langle \overline{v}_t \overline{w}_t \rangle$ and enstrophy $Z_t = \frac{1}{2} \langle \overline{\omega}_t^2 \rangle$ normalized by the energy and enstrophy of the initial condition (of the filtered DNS) E_0 and Z_0 respectively. **c** Time averaged kinetic energy

spectra E_ν , enstrophy spectra Z_ν , and power spectrum of the SGS term $|\Pi_\nu|$. The PDF is computed using a kernel density estimator. The result highlighted in this figure correspond to colored boxes in Fig. 5.

Although deep learning-based sub-models calibrated offline demonstrate superior offline performance, indicated by a high correlation coefficient (refer to Table 2), their coupling with the solver results in an unphysical behavior within the coupled hybrid dynamical system. Specifically, the online performance of the offline sub-model shows, in the orange panel in Fig. 5, sustained high-resolution variability, that is not present in the filtered DNS. The results of online learning with the exact gradient show more realistic flows, that are visually comparable to the the filtered DNS. The blue panel in Fig. 5, demonstrates that the proposed Euler approximation of the online learning problem also provides a flow close to the filtered DNS, without relying on a differentiable solver.

We draw similar conclusions from the statistical properties of the simulated flows reported in Fig. 6. Overall, the analysis of the Probability

Density Function (PDF) of the vorticity field in Fig. 6 confirms that online learning schemes display the closest statistical properties to the filtered DNS field. The DSMAG sub-model does not reproduce the extreme events of both positive and negative vorticity, and the offline learning-based model exhibits a skewed distribution tail, which is predominantly biased towards positive vorticity values. Likely, the time series of kinetic energy E_t and enstrophy Z_t demonstrate that the DSMAG model, characterized by excessive diffusivity and the absence of backscattering, results in a substantial reduction in both Z_t and E_t . The analysis of the time-averaged spectra also clearly highlights the robustness of the sub-models that are trained online. Both the one optimized with the exact gradient and the proposed approximation accurately reproduce small and large scales of the flow.

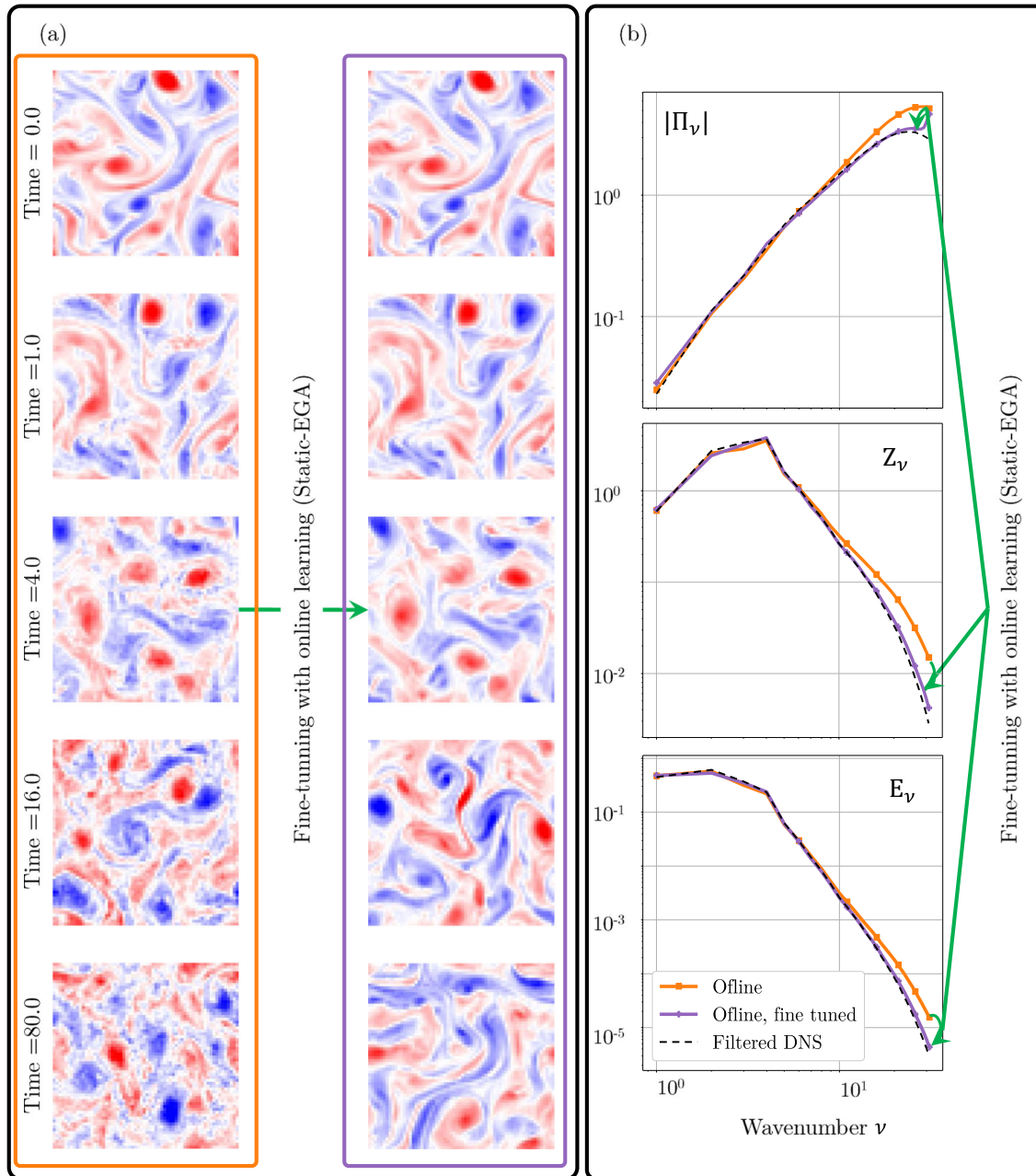


Fig. 7 | Vorticity field and time-averaged spectrums of the offline model before and after fine-tuning. **a** The same offline model that leads to an unphysical vorticity field in Fig. 5 is fine-tuned using the proposed static approximation of the online learning problem, leading to a realistic vorticity simulation. **b** Time-averaged

kinetic energy spectra E_v , enstrophy spectra Z_v , and power spectrum of the SGS term $|\Pi_v|$ of the offline model before and after the online fine-tuning. The green arrows in both panels **a** and **b** highlight the impact of the models trained online.

Fine-tuning offline sub-models. The proposed online optimization approach can be beneficial in situations where a sub-model, calibrated offline, displays a nonphysical or unstable behavior when coupled with the solver. Such a sub-model can be significantly improved by fine-tuning its parameters using an online optimization scheme and our proposed gradient approximation allows achieving this fine-tuning step, without requiring access to the exact gradient of the solver. Figure 7 displays the improvements of the offline model discussed above when fine-tuned (for two epochs) using the proposed static approximation of the online learning. A visual inspection of the vorticity field in Fig. 7, shows how this fine-tuning step successfully removes the unphysical behavior of the offline model. This fine-tuning step also brings significant improvement to the time-averaged spectra, shown in Fig. 7, now aligning closely with the filtered DNS spectra.

Discussion

Nowadays, the high-fidelity simulation and forecasting of physical phenomena rely on dynamical cores derived from well-understood physical principles, along with sub-models that approximate the impacts of certain phenomena that are either unknown or too expensive to be resolved explicitly. Recently, deep learning techniques brought attention and potential to the definition and calibration of these sub-models. In particular, online learning strategies provide appealing solutions to improve numerical models and make them closer to actual observations.

In this work, we develop the EGA, an easy-to-use workflow that allows for the online training of sub-models of hybrid numerical modeling systems. It bypasses the differentiability bottleneck of the physical models and converges to the exact gradients as the time step tends to zero. We stress the robustness and efficiency of the proposed online learning scheme on

realistic case studies, including Quasi-Geostrophic dynamics. Overall, we report significant improvements when compared to standard offline learning schemes and achieve a performance that is similar to solving the exact online learning problem.

Our future work will explore the proposed methodology for large-scale realistic models such as the ones used in atmosphere and ocean simulations. Besides algorithm developments, it will require technical efforts to synchronize the PDE solvers, usually written in high-performance languages such as FORTRAN, to the deep learning-based sub-models that are usually in languages and packages that support automatic differentiation (Pytorch, Jax, Tensorflow). We will particularly focus on the end-to-end calibration of hybrid systems with observation-driven constraints and expect to improve forecasting performance when compared to standard physical models and to full data-driven forecasting surrogates as the ones developed in¹³⁻¹⁸. Scaling the demonstrations to realistic use cases introduces various challenges. In the following, we present some of these challenges along with initial solutions for addressing them.

The sub-model \mathbf{M}_θ in (3) may often account for various unresolved processes (e.g., passive and active tracer transport, biogeochemistry, convection), which are typically governed by different underlying dynamics. To calibrate a number of q sub-models, and assuming that these models are additive, equation (3) becomes:

$$\begin{cases} \dot{\mathbf{u}}_t &= \mathbf{F}(\mathbf{u}_t) + \sum_{i=1}^q \mathbf{M}_\theta^i(\mathbf{u}_t) \\ \mathbf{u}(0) &= \mathbf{u}_0 \end{cases} \quad (34)$$

Jointly calibrating these sub-models using an online learning methodology should certainly be considered with care to avoid mixing the individual contribution of every sub-model. To circumvent such a difficulty, it is necessary to define and include offline costs for each sub-model as a regularization of the online learning objective function. For every sub-model \mathbf{M}_θ^i , we need to define, as discussed in the offline learning section of Results, a reference dataset \mathbf{R}_i^r that represents an ideal response to the input \mathbf{u}_t . The new online cost function that takes into account these offline regularizations can be written as:

$$\mathcal{J}_{combined} = \mathcal{J}(\mathbf{g}^\dagger(u_{t+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta) + \sum_{i=1}^q \mathcal{Q}^i(\mathbf{R}_i^r, \mathbf{M}_\theta^i, \theta)$$

where $\mathcal{J}(\cdot)$ is the online cost function. Minimizing each individual regularization term within this learning methodology ensures that every sub-model accurately represents a specified underlying process. Simultaneously, the online objective function guarantees the correct interaction among these sub-models and with the physical core \mathbf{F} .

In the present development, interactions between the resolved flow component and the unresolved components are formulated as correction terms that are constructed based on the resolved components of the flow. Likely, we are missing several degrees of freedom that represent the independent or generic fine-scale variability. This missing variability generates uncertainty. Taking into account and modeling this uncertainty is mandatory in applications that require probabilistic forecasting, such as data assimilation⁴⁸. Models that encode uncertainty can consistently define Stochastic versions of the original equations and can ensure that certain quantities, such as integrals of functions over a spatial volume like momentum, mass, matter, and energy, are conserved at every time step^{49,50}. The proposed online learning methodology can allow for the calibration of such stochastic sub-models for various configurations regarding the model class and the objective function formulation.

The proposed Euler gradient approximations are based on the assumption that the sub-model in (3) is additive. Precisely, this assumption allows us to express the gradient of the Euler solver, for a given initial condition, in terms of the gradient of the sub-model. The latter is then easily evaluated using automatic differentiation. Additive sub-models are

commonly found in physical simulations, enabling the proposed framework to cover a wide range of case studies. Yet, the proposed methodology can be extended to non-additive sub-models by converting them into additive ones. This conversion can be achieved by explicitly coding the non-additive physical contribution into the sub-model \mathbf{M}_θ .

Understanding the mechanisms responsible for the success of the deep learning model will improve the reliability of such representations. Interpreting deep learning sub-models might also guide the analysis of the interaction between the large-scale physical core and the heuristic processes that are represented by the sub-model. In particular, the sub-model \mathbf{M}_θ can be decomposed into a combination of known expected physical contributions. In the context of subgrid-scale modeling, these contributions might include the dissipative effects acting on the smallest scales, and some advection corrections and backscattering effects acting on the transport of the resolved quantities. New decompositions can then be considered to, for instance, more effectively include the dissipation explicitly and/or anticipated advection corrections in \mathbf{F} .

In addition to subgrid-scale modeling and interpretation, extracting other sources of information including errors, due for instance to the presence of systematic errors, biases, or overall incompatibility of \mathbf{F} in the modeling of some observable $g(\cdot)$, could also be addressed. It is a challenging problem and must require capabilities to disentangle all the contributions from the sub-model \mathbf{M}_θ , while taking into account the possible errors and the overall (positive or negative) impact of \mathbf{F} into the modeling of $g(\cdot)$. For future investigations, we believe that valuable insights on this problem can be obtained by developing ensembles of simulations, performed at various spatial scales, to subsequently analyze the inferred different sub-models.

Our future work will explore the proposed methodology for large-scale realistic models such as the ones used in atmosphere and ocean simulations. Besides algorithm developments, it will require technical efforts to synchronize the PDE solvers, usually written in high-performance languages such as FORTRAN, to the deep learning-based sub-models that are usually in languages and packages that support automatic differentiation (Pytorch, Jax, Tensorflow). We will particularly focus on the end-to-end calibration of hybrid systems with observation-driven constraints and expect to improve forecasting performance when compared to standard physical models and to full data-driven forecasting surrogates as the ones developed in¹³⁻¹⁸.

Methods

Training configuration in the Lorenz 63 experiment

Training data. In the first experiment given in Results, with the Lorenz 63 system, we use multiple datasets $\mathcal{D}_h = \{(u_{t_k+jh}^\dagger, u_{t_k}^\dagger) \mid k = 1 \dots N \text{ and } j = 1 \dots n\}$ that are sampled as the same time step as the Euler approximation. These datasets are generated using the LOSDA ODE solver⁵¹. The number of training samples N is equal to 100 time steps and the number of simulation time steps n is fixed to 10.

The dataset used in the remaining Lorenz 63 experiments, $\mathcal{D}_h = \{(u_{t_k+jh}^\dagger, u_{t_k}^\dagger) \mid k = 1 \dots N \text{ and } j = 1 \dots n\}$ is sampled at $h = 0.01$. The same sampling rate was used in multiple works⁵²⁻⁵⁴ that involve the data-driven identification of the Lorenz 63 system. This dataset is also generated using the LOSDA ODE solver⁵¹. The number of training samples N is equal to 5000 time steps and the number of simulation time steps n is fixed to 10.

Training criterion and numerical solver. In both the Lorenz 63 experiments, the online objective function corresponds to the mean squared error between the true Lorenz 63 state and the numerical integration of the model (30) over $n = 10$ time steps. The cost function can be written as:

$$\mathcal{J} = \frac{1}{N} \sum_k \frac{1}{n} \sum_{j=1}^n \| u_{t_k+jh}^\dagger - \Psi^j(u_{t_k}^\dagger) \|_2^2 \quad (35)$$

where $\| \cdot \|_2$ is the L2 norm. The solver Ψ used in these experiments is a differentiable DOPRI8 solver, developed in³⁶.

Table 3 | Parameters of the Flow Configurations

DNS grid ($N_{\text{DNS}} \times N_{\text{DNS}}$)	LES grid ($N_{\text{LES}} \times N_{\text{LES}}$)	Scale (δ)	DNS time step (h_{DNS})	LES time step (h_{LES})	Re	r	k_f
1024 × 1024	64 × 64	16	5×10^{-5}	8×10^{-4}	20000	0.1	4

Both the DNS and LES systems share the same parameters, except for the grid size and the integration time step. The grid size of the LES system is reduced by a factor of $\delta = 16$. The time step of the LES system corresponds to that of the DNS multiplied by δ .

Parameterization of the deep learning sub-model and Baseline. The sub-model employed in the Lorenz 63 experiments consists of a fully connected neural network with two hidden layers, each with three neurons. The activation function utilized for these hidden layers is the hyperbolic tangent. Below is a detailed description of the models tested in the Lorenz 63 experiment given in Results:

- **No calibration and only using the physical core:** In this experiment we only run the physical core given by $\dot{\mathbf{u}}_t = \mathbf{F}(\mathbf{u}_t)$ i.e., by removing the sub-model \mathbf{M}_θ in (30).
- **Online calibration with exact gradient:** The sub-model is trained by utilizing the exact gradient of the online cost (35). To compute the gradient of the solver, we rely on the fact that the solver used in this experiment is implemented in a differentiable language, allowing for automatic differentiation.
- **Online calibration with a static approximation:** The sub-model is trained by utilizing the proposed Euler formulation of the gradient of the online cost (35), in which we use a static approximation of the Jacobian as described in (19).
- **Online calibration with an Ensemble approximation:** Similarly to above, the sub-model is trained by utilizing the proposed Euler formulation of the gradient of the online cost (35), but the Jacobian is approximated using an ensemble as described in (26). The size of the ensemble is set to 5 members.

Evaluation criteria. In Fig. 2, the gradient error is computed as the mean absolute error between the exact gradient (computed using automatic differentiation) and the one returned by one of the proposed approximations. If we use equation (27) to express this error and assuming that the exact gradient is $\frac{\partial \mathcal{J}}{\partial \theta}$, the error depicted in Fig. 2 is computed as:

$$\epsilon = \frac{1}{a} \left\| \frac{\partial \mathcal{J}}{\partial \theta} - \mathbf{v} + \mathbf{w} \mathbf{A}_{l,p}(\mathbf{u}_t) \right\|_1 \quad (36)$$

where $\|\cdot\|_1$ is the L1 norm and a is the number of parameters (we recall that $\theta \in \mathbb{R}^a$). In this experiment, the tested gradients correspond to $p = 2$ (since we use a fixed number of simulation steps $n = 10$), and to $l = 1, 2$ (which corresponds to the EGA and Static-EGA formulas in (28)). The exact Jacobian is in this experiment evaluated using automatic differentiation.

In Fig. 4, the MEG is simply the mean squared error at lead time $t_0 + ih$ normalized by the initial error at $t_0 + h$. It can be written as:

$$\text{MEG}(t_0 + ih) = \frac{\|u_{t_0+ih}^\dagger - \Psi^i(u_{t_0}^\dagger)\|_2^2}{\|u_{t_0+h}^\dagger - \Psi(u_{t_0}^\dagger)\|_2^2} \quad (37)$$

The Lyapunov spectrum in Table 1 is computed using the Gram-Schmidt orthonormalization technique⁵⁵, and the Lyapunov dimension is deduced from the spectrum as given in⁵⁶.

Training configuration in the QG experiment

Training data. We run the QG equations with the flow configuration given in Table 3 starting from 14 different initial random fields of vorticity on a high-resolution grid to generate direct numerical simulation (DNS) data. These runs are used to generate 14 different initial conditions that are in the statistical equilibrium regime. The new initial conditions are then used to generate DNS data of 2 million time steps that correspond to 4000 eddy turnover times. These data are then filtered to the resolution of the LES simulation and sampled every h_{LES} . From the 14 runs, we used 8

datasets for training one for validation, and the remaining 5 datasets for testing and evaluation of the models. Regarding the training and validation datasets, they are written as $\{(\bar{\omega}_{t_k+jh_{\text{LES}}}, \bar{\psi}_{t_k})\}$ with $k = 1 \dots N$ and $j = 1 \dots n$ where $N = 2000 \times 8$ (where 8 is the number of training datasets) and $n = 10$ for the online learning experiments and as $\{(\Pi_{t_k}, (\bar{\omega}_{t_k}, \bar{\psi}_{t_k}))\}$ with $k = 1 \dots N$.

Training criterion. The online objective function in the QG experiment corresponds to the means squared error between the true vorticity field and the one issued from the numerical integration of the model (33) over $n = 10$ time steps. The cost function can be written as:

$$\mathcal{J} = \frac{1}{N} \sum_k \frac{1}{n} \sum_{j=1}^n \|\bar{\omega}_{t_k+jh} - \Psi^j(\bar{\omega}_{t_k})\| \quad (38)$$

The solver Ψ depends solely on the vorticity field $\bar{\omega}$ as all the other variables can be deduced from $\bar{\omega}$.

The offline objective function is the mean squared error between the output of the sub-model \mathbf{M}_θ and the reference subgrid scale term Π . It can be written as:

$$\mathcal{J} = \frac{1}{N} \sum_k \frac{1}{n} \sum_{j=1}^n \|\Pi_{t_k} - \mathbf{M}_\theta(\bar{\omega}_{t_k}, \bar{\psi}_{t_k})\| \quad (39)$$

Numerical solver of the QG system. QG system is solved using a code adapted from³¹. This solver is written in a differentiable language which allows the comparison of our proposed approximation to models that are optimized online with the true gradient of the solver. It relies on a pseudo-spectral solver and a classical fourth-order Runge-Kutta time integration scheme.

Filtering and coarse graining operation. The QG equations are defined on a double periodic squared domain $\Omega \in [-\pi, \pi]^2$. The DNS solution is constructed on a regular $N_{\text{DNS}} \times N_{\text{DNS}}$ grid with a uniform spacing $\Delta_{\text{DNS}} = 2\pi N_{\text{DNS}}^{-1}$. The LES system of equations is obtained by projecting the DNS states through a convolution with a spatial kernel G , followed by a discretization on the reduced grid, with larger spacing $\Delta_{\text{LES}} = \delta \Delta_{\text{DNS}}$. We use in this experiment a Gaussian filter that can be defined in spectral space as:

$$G_\nu = \exp\left(-\frac{\nu^2 \Delta_f^2}{24}\right),$$

where Δ_f is the filter size, which is taken to be $\Delta_f = 2\Delta_{\text{LES}}$ to yield sufficient resolution⁸. This filtering/coarse-graining operation can be written as (taking here as an example the DNS vorticity field ω):

$$\bar{\omega}_\nu = (\omega * G)(|\nu| < \pi \Delta_{\text{LES}}^{-1}).$$

Regarding numerical aspects, we can solve the time integration of the LES system with a larger time-step by a factor corresponding to the grid size ratio, that is, $\Delta t_{\text{LES}} = \delta \Delta t_{\text{DNS}}$.

Parameterization of the deep learning sub-model and baseline. The sub-model used in the QG experiments is a Convolutional Neural Network (CNN) that has the same architecture as the one used in⁷. The inputs of the CNN are the vorticity and streamfunction fields $(\bar{\omega}_{t_k}, \bar{\psi}_{t_k})$. The convolutional layers have the same dimension (64×64) as that of the

input and output layers. All layers are initialized randomly. The number of channels is set to 64 and the filter size is (5×5) . The activation function of each layer is ReLu (rectified linear unit) except for the last one, which is a linear map. Similarly to³¹ We use periodic padding.

Below is a detailed description of the models tested in the QG experiment:

- **Online calibration with exact gradient:** In this calibration scheme, we assume that the solver of (33) is differentiable and we optimize the parameters of the sub-model with the exact gradient of the online objective function. This experiment was already studied in³¹ and shows better stability performance than offline calibration schemes.
- **Online calibration with a static approximation:** In this experiment, we evaluate the performance of the proposed online learning scheme in which the gradient of the online loss function is approximated based on (19). In this experiment, the solver of (33) is not assumed to be differentiable.
- **Offline calibration:** We also compare the proposed static approximation to a simple offline learning strategy in which the CNN is calibrated to reproduce the subgrid-scale term Π_r .
- **Physical SGS model with dynamic Smagorinsky (DSMAG):** We also evaluate and compare the proposed approximation schemes with respect to classical physics-based parameterization given by the dynamic Smagorinsky (DSMAG) model. In this model, the impact of the unresolved scales on the dynamics is assumed to be diffusive with a diffusion constant that is computed automatically. The dynamic Smagorinsky model has been widely used as a baseline in many studies. For a detailed explanation, please refer to, for example⁷.

Data availability

The data used in this study can be generated using the code available at <https://github.com/saidOUALA/EGA>.

Code availability

The code used in this study is available at <https://github.com/saidOUALA/EGA>.

Received: 27 May 2024; Accepted: 18 November 2024;

Published online: 18 December 2024

References

1. Frederiksen, J. S. & Davies, A. G. Eddy viscosity and stochastic backscatter parameterizations on the sphere for atmospheric circulation models. *J. Atmos. Sci.* **54**, 2475–2492 (1997).
2. Shutts, G. A kinetic energy backscatter algorithm for use in ensemble prediction systems. *Q. J. R. Meteorol. Soc.: A J. Atmos. Sci., Appl. Meteorol. Phys. Oceanogr.* **131**, 3079–3102 (2005).
3. Juricke, S., Danilov, S., Koldunov, N., Oliver, M. & Sidorenko, D. Ocean kinetic energy backscatter parametrization on unstructured grids: Impact on global eddy-permitting simulations. *J. Adv. Model. Earth Syst.* **12**, e2019MS001855 (2020).
4. Jiang, S., Zheng, Y. & Solomatine, D. Improving ai system awareness of geoscience knowledge: Symbiotic integration of physical approaches and deep learning. *Geophys. Res. Lett.* **47**, e2020GL088229 (2020).
5. Camps-Valls, G., Reichstein, M., Zhu, X. & Tuia, D. Advancing deep learning for earth sciences: From hybrid modeling to interpretability. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, 3979–3982 (IEEE, 2020).
6. Brajard, J., Carrassi, A., Bocquet, M. & Bertino, L. Combining data assimilation and machine learning to infer unresolved scale parametrization. *Philos. Trans. R. Soc. A* **379**, 20200086 (2021).
7. Guan, Y., Chattopadhyay, A., Subel, A. & Hassanzadeh, P. Stable a posteriori les of 2d turbulence using convolutional neural networks: Backscattering analysis and generalization to higher re via transfer learning. *J. Comput. Phys.* **458**, 111090 (2022).
8. Guan, Y., Subel, A., Chattopadhyay, A. & Hassanzadeh, P. Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate les. *Phys. D: Nonlinear Phenom.* **443**, 133568 (2023).
9. Jakhhar, K. et al. Learning closed-form equations for subgrid-scale closures from high-fidelity data: Promises and challenges. *arXiv preprint arXiv:2306.05014* (2023).
10. Farchi, A., Chrust, M., Bocquet, M., Laloyaux, P. & Bonavita, M. Online model error correction with neural networks in the incremental 4d-var framework. *J. Adv. Model. Earth Syst.* **15**, e2022MS003474 (2023).
11. Shen, C. et al. Differentiable modelling to unify machine learning and physical models for geosciences. *Nat. Rev. Earth Environ.* **4**, 552–567 (2023).
12. Kochkov, D. et al. Neural general circulation models for weather and climate. *Nature* 1–7 (2024).
13. Pathak, J. et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214* (2022).
14. Lam, R. et al. Learning skillful medium-range global weather forecasting. *Science* **382**, 1416–1421 (2023).
15. Dueben, P. D. et al. Challenges and benchmark datasets for machine learning in the atmospheric sciences: Definition, status, and outlook. *Artif. Intell. Earth Syst.* **1**, e210002 (2022).
16. Chen, K. et al. Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv preprint arXiv:2304.02948* (2023).
17. Bi, K. et al. Accurate medium-range global weather forecasting with 3d neural networks. *Nature* **619**, 533–538 (2023).
18. Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J. K. & Grover, A. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343* (2023).
19. Hourdin, F. et al. The art and science of climate model tuning. *Bull. Am. Meteorol. Soc.* **98**, 589–602 (2017).
20. Smagorinsky, J. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather Rev.* **91**, 99–164 (1963).
21. Germano, M., Piomelli, U., Moin, P. & Cabot, W. H. A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A: Fluid Dyn.* **3**, 1760–1765 (1991).
22. Moin, P., Squires, K., Cabot, W. & Lee, S. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Phys. Fluids A: Fluid Dyn.* **3**, 2746–2757 (1991).
23. Spalart, P. & Allmaras, S. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, 439 (1992).
24. Wilcox, D. C. Formulation of the kw turbulence model revisited. *AIAA J.* **46**, 2823–2838 (2008).
25. Iglesias, M. A., Law, K. J. & Stuart, A. M. Ensemble kalman methods for inverse problems. *Inverse Probl.* **29**, 045001 (2013).
26. Kovachki, N. B. & Stuart, A. M. Ensemble kalman inversion: a derivative-free technique for machine learning tasks. *Inverse Probl.* **35**, 095005 (2019).
27. Böttcher, L. Gradient-free training of neural ODEs for system identification and control using ensemble Kalman inversion. *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, <https://openreview.net/forum?id=opPP65bJIN> (2023).
28. Bose, R. & Roy, A. M. Accurate deep learning sub-grid scale models for large eddy simulations. *arXiv preprint arXiv:2307.10060* (2023).
29. Ross, A., Li, Z., Perezhugin, P., Fernandez-Granda, C. & Zanna, L. Benchmarking of machine learning ocean subgrid parameterizations in an idealized model. *J. Adv. Model. Earth Syst.* **15**, e2022MS003258 (2023).
30. Chattopadhyay, A. & Hassanzadeh, P. Long-term instability of deep learning-based digital twins of the climate system: Cause and solution. *Bulletin of the American Physical Society* (2023).

31. Frezat, H., Le Sommer, J., Fablet, R., Balarac, G. & Lguensat, R. A posteriori learning for quasi-geostrophic turbulence parametrization. *J. Adv. Model. Earth Syst.* **14**, e2022MS003124 (2022).
32. Nonnenmacher, M. & Greenberg, D. S. Deep emulators for differentiation, forecasting, and parametrization in earth science simulators. *J. Adv. Model. Earth Syst.* **13**, e2021MS002554 (2021).
33. Frezat, H., Fablet, R., Balarac, G. & Sommer, J. L. Gradient-free online learning of subgrid-scale dynamics with neural emulators (2023). 2310.19385.
34. Pedersen, C., Zanna, L., Bruna, J. & Perezhogin, P. Reliable coarse-grained turbulent simulations through combined offline learning and neural emulation. *1st Workshop on the Synergy of Scientific and Machine Learning Modeling @ ICML2023*, <https://openreview.net/forum?id=VNxsoKLpiQ> (2023).
35. Andersson, J. A., Gillis, J., Horn, G., Rawlings, J. B. & Diehl, M. Casadi: a software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* **11**, 1–36 (2019).
36. Chen, T. Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 6571–6583 (2018).
37. Lopez-Gomez, I. et al. Training physics-based machine-learning parameterizations with gradient-free ensemble kalman methods. *J. Adv. Model. Earth Syst.* **14**, e2022MS003105 (2022).
38. Pahlavan, H. A., Hassanzadeh, P. & Alexander, M. J. Explainable offline-online training of neural networks for parameterizations: A 1d gravity wave-qbo testbed in the small-data regime. *Geophys. Res. Lett.* **51**, e2023GL106324 (2024).
39. Allen, D. R., Hodyss, D., Hoppel, K. W. & Nedoluha, G. E. The ensemble tangent linear model applied to nonorographic gravity wave drag. *Monthly Weather Rev.* **151**, 193–210 (2023).
40. Horace He, R. Z. functorch: Jax-like composable function transforms for pytorch. <https://github.com/pytorch/functorch> (2021).
41. Brandstetter, J., Worrall, D. E. & Welling, M. Message passing neural PDE solvers. *International Conference on Learning Representations*, <https://openreview.net/forum?id=vSix3HPYKSU> (2022).
42. Lippe, P., Veeling, B., Perdikaris, P., Turner, R. & Brandstetter, J. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Adv. Neural Inform. Process. Syst.* **36** (2024).
43. List, B., Chen, L.-W., Bali, K. & Thuerey, N. How temporal unrolling supports neural physics simulators. *arXiv preprint arXiv:2402.12971* (2024).
44. Lorenz, E. N. Predictability: A problem partly solved. In *Proc. Seminar Predict.* **1** (1996).
45. Lorenz, E. N. Deterministic Nonperiodic Flow. *J. Atmos. Sci.* **20**, 130–141 (1963).
46. Chandler, G. J. & Kerswell, R. R. Invariant recurrent solutions embedded in a turbulent two-dimensional kolmogorov flow. *J. Fluid Mech.* **722**, 554–595 (2013).
47. Sagaut, P. *Large eddy simulation for incompressible flows: an introduction* (Springer Science & Business Media, 2005).
48. Zhen, Y., Resseguier, V. & Chapron, B. Physically constrained covariance inflation from location uncertainty. *EGUsphere* **2023**, 1 (2023).
49. Mémin, E. Fluid flow dynamics under location uncertainty. *Geophys. Astrophys. Fluid Dyn.* **108**, 119–146 (2014).
50. Holm, D. D. Variational principles for stochastic fluid dynamics. *Proc. R. Soc. A: Math., Phys. Eng. Sci.* **471**, 20140963 (2015).
51. Hindmarsh, A. C. ODEPACK, a systematized collection of ODE solvers. *IMACS Trans. Sci. Comput.* **1**, 55–64 (1983).
52. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl Acad. Sci.* **113**, 3932–3937 (2016).
53. Lguensat, R., Tandeo, P., Ailliot, P., Pulido, M. & Fablet, R. The Analog Data Assimilation. *Monthly Weather Review* <http://journals.ametsoc.org/doi/10.1175/MWR-D-16-0441.1> (2017).
54. Ouala, S. et al. Learning latent dynamics for partially observed chaotic systems. *Chaos: Interdiscip. J. Nonlinear Sci.* **30**, 103121 (2020).
55. Parker, T. S. & Chua, L. O. *Stability of Limit Sets* (pp. 57–82. Springer New York, New York, NY, 1989).
56. Parker, T. S. & Chua, L. O. *Dimension* (pp. 167–199. Springer New York, New York, NY, 1989).
57. Sprott, J. C. *Chaos and Time-Series Analysis*. (Oxford University Press, Inc., New York, NY, USA, 2003).

Acknowledgements

This work was funded by the ERC Synergy project 856408-STUOD, the French ANR Oceanix (ANR-19-CHIA-0016), and the EU Horizon Europe projects EDITO Model Lab (Grant 101093293) and AI4PEX (Grant 101137682). It also benefited from HPC and GPU resources provided by GENCI-IDRIS (Grant 2021-101030) and the CPER AIDA GPU cluster, supported by the Regional Council of Brittany and FEDER.

Author contributions

S.O. conducted the research; S.O., B.C., F.C., L.G., and R.F. designed the research and analyzed the data; S.O., B.C., and R.F. wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s42005-024-01880-7>.

Correspondence and requests for materials should be addressed to Said Ouala.

Peer review information *Communications Physics* thanks the anonymous reviewers for their contribution to the peer review of this work. A peer review file is available.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2024