



HAL
open science

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

Said Ouala, Bertrand Chapron, Fabrice Collard, Lucile Gaultier, Ronan Fablet

► **To cite this version:**

Said Ouala, Bertrand Chapron, Fabrice Collard, Lucile Gaultier, Ronan Fablet. Online calibration of deep learning sub-models for hybrid numerical modeling systems. *Communications Physics*, 2024, 7 (1), pp.402. 10.1038/s42005-024-01880-7. hal-04907210

HAL Id: hal-04907210

<https://hal.science/hal-04907210v1>

Submitted on 30 Jan 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ONLINE CALIBRATION OF DEEP LEARNING SUB-MODELS FOR HYBRID NUMERICAL MODELING SYSTEMS

A PREPRINT

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

(1) IMT Atlantique; Lab-STICC, 29200 Brest, France

(2) Ifremer, LOPS, 29200 Brest, France

(3) ODL, 29200 Brest, France

{said.ouala, ronan.fablet}@imt-atlantique.fr

Bertrand.Chapron@ifremer.fr,

{dr.fab, lucile.gaultier}@oceandatalab.com

November 20, 2023

ABSTRACT

Artificial intelligence and deep learning are currently reshaping numerical simulation frameworks by introducing new modeling capabilities. These frameworks are extensively investigated in the context of model correction and parameterization where they demonstrate great potential and often outperform traditional physical models. Most of these efforts in defining hybrid dynamical systems follow offline learning strategies in which the neural parameterization (called here sub-model) is trained to output an ideal correction. Yet, these hybrid models can face hard limitations when defining what should be a relevant sub-model response that would translate into a good forecasting performance. End-to-end learning schemes, also referred to as online learning, could address such a shortcoming by allowing the deep learning sub-models to train on historical data. However, defining end-to-end 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 difficulty and bypass the differentiability challenge of physical models, 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 are performed on various case studies, including prototypical ocean-atmosphere dynamics. Results show significant improvements over offline learning, highlighting the potential of end-to-end online learning for hybrid modeling.

Keywords Hybrid models, Online learning, Learning simulation, Numerical models, Gradient approximation

1 Introduction

High-fidelity simulations of physical phenomena require intense modeling and computing efforts. Prohibitive computational costs often lead to only resolving certain spatiotemporal 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 backscattering Frederiksen and Davies [1997], Shutts [2005], Juricke et al. [2020]. 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.

In recent years, scientific computing problems have been explored from the perspective of machine learning and Artificial Intelligence (AI). The combination of AI with computational sciences has given rise to a wide spectrum of methodological questions, articulated in the framework of Scientific Machine Learning (SciML). These questions encompass various aspects, from model acceleration through AI solvers Raissi et al. [2019], Ouala et al. [2021], Partee et al. [2022], Li et al. [2023], to the learning of model correction terms or sub-models in numerical simulations Maulik et al. [2019], Zanna and Bolton [2020], Gupta and Lermusiaux [2021], Charalampopoulos and Sapsis [2022], Bonnet et al. [2022], Subel et al. [2023]. Additionally, Scientific Machine Learning has significantly advanced the development of state-of-the-art surrogate modeling techniques Wang et al. [2018], Brunton and Kutz [2019], Ouala et al. [2020], Hasegawa et al. [2020], Ouala et al. [2023a], Serrano et al. [2023]. These models show particular interest for complex systems where physical models do not exist, or for the replacement of complex representations of physical systems (e.g., computationally expensive computer models) with simpler, computationally-efficient counterparts, capable of capturing some of the underlying dynamics.

Surrogate modeling finds applications in forecasting Ouala et al. [2018a], Cheng et al. [2022], Migus et al. [2023], Serrano et al. [2023], simulation Nguyen et al. [2019], Brajard et al. [2019], Bocquet et al. [2020], Ouala et al. [2020, 2023b], data assimilation Ouala et al. [2018b], Cheng et al. [2023], Ouala et al. [2023c], Cheng et al. [2024] and control Kaiser et al. [2018], Brunton [2021]. In the context of Earth system predictability, AI has exhibited remarkable potential and has rapidly gained momentum in leveraging atmosphere-reanalyzed data to provide efficient forecasting systems that are purely data-driven. By learning from relatively coarse-grained, but fully consistent, space-time atmosphere parameter fields, AI techniques show large promises and often outperform advanced numerical weather forecast models when focusing on specific processes or variables Pathak et al. [2022], Lam et al. [2022], Dueben et al. [2022], Chen et al. [2023], Bi et al. [2023], Nguyen et al. [2023].

To scale up to entire systems and coupled systems, the design of hybrid modeling approaches is also actively under development, building on the combination of physical equations and deep learning components Jiang et al. [2020], Camps-Valls et al. [2020], Brajard et al. [2021], Guan et al. [2022, 2023], Jakhar et al. [2023], Farchi et al. [2023], Shen et al. [2023]. Such strategies require two main aspects. The availability of a *representative enough* database and the ability to translate the problem into a relevant objective function that can be optimized with respect to the deep learning model. From a machine learning point of view, the ability to define end-to-end learning strategies is a key element that allowed deep learning models to advance several signal processing fields. Specifically, questions such as how to train end-to-end a large language model and how to define an objective function on the *usually intractable* likelihood of the data in generative modeling is one of the most important ingredients in the design and success of deep learning architectures. Recent studies, referred to here as online learning schemes, have explored such a philosophy in the calibration of hybrid numerical models that contain both physical and deep learning components Um et al. [2020], Frezat et al. [2022, 2023]. This learning strategy requires, in general, having access to the gradient of the physical model with respect to the parameters of the deep learning component. However, this gradient is usually unavailable for most state-of-the-art numerical models, which explains why offline learning strategies have been mostly considered in hybrid numerical modeling systems Frezat et al. [2020], Zanna and Bolton [2020], Ross et al. [2023], Guan et al. [2023].

In this work, we address the online learning of hybrid models. We derive an easy-to-use workflow to perform online 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. Overall, we show significant improvements compared to standard offline learning methods, achieving performance similar to solving the exact online learning problem. We also show that the proposed methodology can be used to improve sub-models that are trained offline, in producing more realistic simulations.

The paper is organized as follows. We start in section 2 by giving a general introduction to the parameterization problem in earth system modeling and we discuss and situate some state-of-the-art approaches with respect to deep learning models. We then introduce both offline and online learning of sub-models and present some state-of-the-art solutions to solve the online learning problem including the use of differentiable emulators Nonnenmacher and Greenberg [2021], and the definition of appropriate adjoints in *optimize than discretize* learning strategies Chen et al. [2018]. Section 3 presents the proposed framework, followed by the experiments and results in Section 4. We discuss perspectives for future works in Section 5 and close the paper with a conclusion in Section 6.

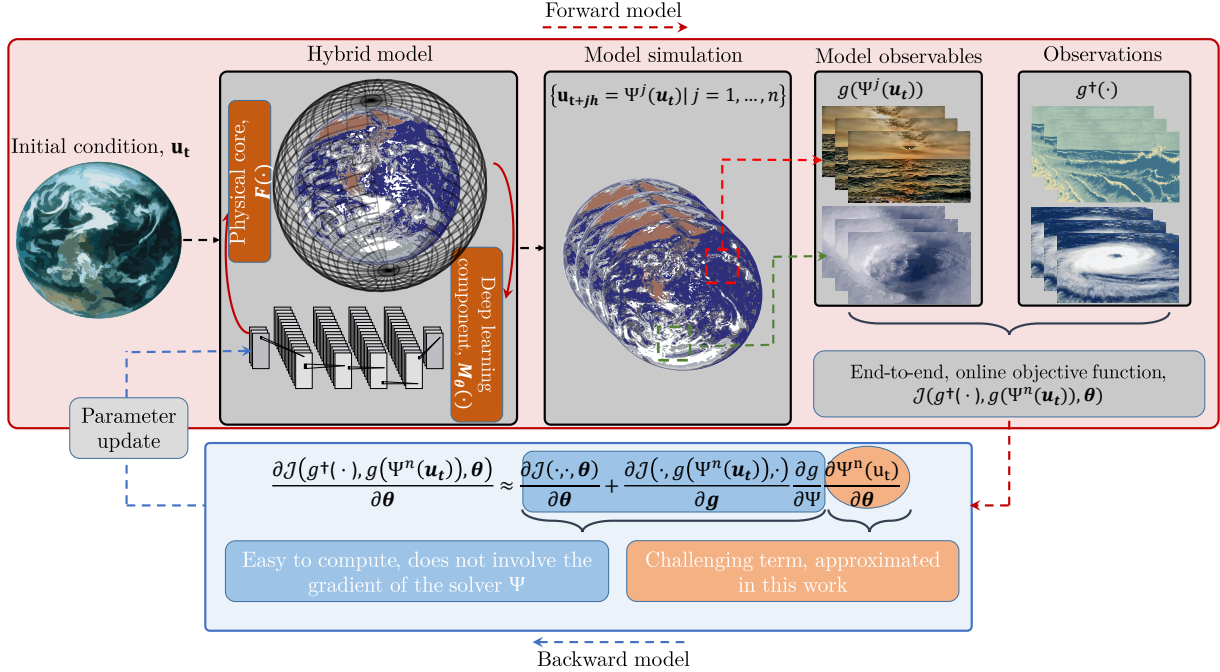


Figure 1: **Global overview of the online learning problem and link to the proposed EGA.** The aim of the proposed framework is to provide a simple and easy-to-use workflow to solve online optimization problems of hybrid models that contain both a physical core and a deep learning component.

2 Problem formulation and related works

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 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 Hourdin et al. [2017]. 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 Smagorinsky [1963], Germano et al. [1991], Moin et al. [1991], Spalart and Allmaras [1992], Wilcox [2008]. In the same context, stochastic subgrid scale sub-models, like those developed under the Location Uncertainty Mémin [2014] (LU) or Stochastic Advection by Lie Transport (SALT) Holm [2015] paradigms, also rely on physical principles. These sub-models encode within \mathcal{M}_θ a diffusive effect of large-scale components by small-scale velocity, small-scale energy backscattering, and a modified advection term related to the turbophoresis phenomenon Resseguier et al. [2017a,b,c].

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. Despite recent progress in learning neural operators Li et al. [2023], Lanthaler and Stuart [2023], validating these models remains more complex when compared to physics-based approaches. However, empirical evidence continuously demonstrates the interest in using machine learning models to enhance current state-of-the-art physical simulations.

2.1 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:

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

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 $t_f = t + nh$. 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) convergent 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 Iglesias et al. [2013], Kovachki and Stuart [2019], Böttcher [2023]. 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 discuss in section 5, how they can be relaxed.

2.2 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:

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

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 2.1 (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 Bose and Roy [2023], Guan et al. [2023], Ross et al. [2023]. 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*

$$\mathbf{R}_t = \tau(\mathbf{F}_{d_{u^\dagger}}(\mathbf{u}_t^\dagger)) - \mathbf{F}(\tau(\mathbf{u}_t^\dagger)) \quad (7)$$

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:

$$\mathcal{Q}(\mathbf{R}_t, \mathbf{M}_\theta, \theta) = \frac{1}{N} \sum_k \left\| \mathbf{R}_{t_k} - \mathbf{M}_\theta(\tau(\mathbf{u}_{t_k}^\dagger)) \right\|^2 + \mathcal{R}(\cdot) \quad (8)$$

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 (8) 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 that is considered an ideal response. However, several works showed that, while deep learning models show great success in achieving better offline results than state-of-the-art physical parameterizations, these models might perform poorly in online tests, in which the data-driven sub-model is coupled to the numerical solver of the physical model. These models can show pathological behaviors such as numerical instabilities or physically unrealistic flows that can lead to a blow-up of the simulation Jakhar et al. [2023], Chatopadhyay and Hassanzadeh [2023]. Discussing and accessing the performance of offline closures in online tests is mandatory for the progress and reliability of such tools, and several works have shown that offline sub-models can be constrained to deliver stable long-term online simulations. For instance, Guan et al. [2022] showed that offline subgrid scale sub-models can produce realistic and stable simulations by using large datasets, and Frezat et al. [2020], Guan et al. [2023] studied the positive impact of including physical constraints on offline closures to allow for realistic long time simulations.

Another problem with offline optimization is that 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} . In such scenarios, relying on end-to-end learning is appealing and has already motivated several works that rely on pure data-driven surrogate models Pathak et al. [2022], Lam et al. [2022], Dueben et al. [2022], Chen et al. [2023], Bi et al. [2023], Nguyen et al. [2023]. In the context of hybrid models, a major drawback of such optimization strategies is the inclusion of the numerical solver within the optimization problem, which makes the resolution more challenging. In the rest of this paper, we briefly discuss various online optimization strategies and propose an easy-to-use workflow for learning online sub-models.

2.3 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:

$$\mathcal{J}(\mathbf{g}^\dagger(u_{t+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta) \quad (9)$$

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).

The online learning strategy significantly widens the range of metrics that can be considered to calibrate the sub-model parameters. It can indeed link the training of the sub-model parameters to any end-to-end constraint related to the (short-term) simulation of the dynamical model (3).

Example 2.2 (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. Frezat et al. [2022], 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:*

$$\mathcal{J}(\cdot) = \frac{1}{N} \sum_k \frac{1}{n} \sum_{j=1}^n \|\tau(u_{t_k+jh}^\dagger) - \Psi^j(\tau(u_{t_k}^\dagger))\| + \mathcal{R}(\cdot) \quad (10)$$

The resolution of the optimization problem (9) 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 (9) with respect to the parameters of the sub-model:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}}(\mathbf{g}^\dagger(u_{t+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \boldsymbol{\theta}) = \frac{\partial \mathcal{J}(\cdot, \cdot, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{J}(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \cdot)}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \Psi} \frac{\partial \Psi^n(\mathbf{u}_t)}{\partial \boldsymbol{\theta}} \quad (11)$$

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

$$\frac{\partial \Psi^n(\mathbf{u}_t)}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \Psi \circ \Psi \circ \dots \circ \Psi(\mathbf{u}_t) \quad (12)$$

Computing this gradient requires the solver in Ψ to be differentiated with respect to the parameters of the sub-model, which critically limits the use of the online approach in practice. 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). They can not provide adjoint models that evaluate gradients with respect to deep learning parameters. This largely explains the extensive emphasis on offline calibration schemes, even though, the aim of a sub-model is always to have a good online performance.

2.4 Emulators

In our context, differentiable emulators Nonnenmacher and Greenberg [2021] are considered to both approximate the physical part of (3) and the numerical solver Ψ . These models were successfully tested on data assimilation toy problems Nonnenmacher and Greenberg [2021], and can be adapted to the online learning of sub-models Frezat et al. [2023].

Let \mathbf{G}_ϕ be a deep learning approximation of \mathbf{F} such that $\dot{\mathbf{u}}_t \approx \mathbf{G}_\phi(\mathbf{u}_t) + \mathbf{M}_\theta$ and let Ψ_α be the numerical scheme with parameters α that solves this approximate model. The solver Ψ_α can be based on a deep learning model. For the sake of simplicity of presentation, 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.*:

$$\begin{aligned} \mathbf{u}_{t_f} = \mathbf{u}_{t_0+nh} &= \underbrace{\Psi \circ \Psi \circ \dots \circ \Psi}_{n \text{ times}}(\mathbf{u}_{t_0}) \\ &\approx \underbrace{\Psi_\alpha \circ \Psi_\alpha \circ \dots \circ \Psi_\alpha}_{r \text{ times}}(\mathbf{u}_{t_0}) \end{aligned} \quad (13)$$

Assuming that r , h' , \mathbf{G}_ϕ and Ψ_α are correctly calibrated, the gradients of the true solution with respect to the parameters are replaced by the ones of (13). 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 emulator Frezat et al. [2023].

2.5 Link to optimal control and gradient descent by the shooting method

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

$$\begin{cases} \arg \min_{\boldsymbol{\theta}} \mathcal{J} \\ \text{s.t. } \dot{\mathbf{u}}_t = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t) \end{cases} \quad (14)$$

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

$$\mathcal{L} = \mathcal{J} + \int_t^{t+nh} \lambda_t (\dot{\mathbf{u}}_t - \mathbf{F}(\mathbf{u}_t) - \mathbf{M}_\theta(\mathbf{u}_t)) dt \quad (15)$$

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 adjoint models of (5) with respect to the parameters of the deep learning sub-model. Overall, optimal control methods can be divided into direct and indirect approaches Andersson [2013], Andersson et al. [2019]. 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 (14). The latter technique was advertised in Chen et al. [2018] 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.

3 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 (9):

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

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

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

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.*:

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

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:

$$\mathbf{u}_{t+h} = \Psi_E(\mathbf{u}_t) \quad (16)$$

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

$$\begin{aligned} \mathbf{u}_{t+h} &= \Psi(\mathbf{u}_t) \\ &= \Psi_E(\mathbf{u}_t) + O(h^2) \end{aligned} \quad (17)$$

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

$$\begin{aligned} \mathbf{u}_{t_f} = \mathbf{u}_{t+nh} &= \Psi^n(\mathbf{u}_t) \\ &= \Psi_E(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \end{aligned} \quad (18)$$

By using the proposition 3.1, 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 3.1 (EGA). *: Under the same conditions as proposition 3.1, 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:*

$$\frac{\partial}{\partial \boldsymbol{\theta}} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{n-1} \left(\underbrace{\prod_{i=1}^{n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)}}_{\text{Jacobian of the flow}} \right) h \underbrace{\frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t))}_{\text{Gradient of the sub-model}} + h \frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \quad (19)$$

Corollary 3.1.1. *Under the same conditions as proposition 3.1, 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.*:*

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

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 3.1 and corollary 3.1.1 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 $\boldsymbol{\theta}$ 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+jh} using automatic differentiation, the Jacobian of the flow needs to be specified or approximated.

3.1 EGA Jacobian approximation

The Jacobian in the EGA equations (19) and (20) 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.

3.1.1 Static formulation (Static-EGA)

A static formulation of the Jacobian corresponds to approximating the Jacobian term in (19) and (20) by the identity matrix. It can be shown that a static formulation of the Jacobian matrix implies corollary 3.1.2.

Corollary 3.1.2 (Static-EGA). *Under the same conditions as proposition 3.1, 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:*

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + O(h^2) \quad (21)$$

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.:

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + O(h) \quad (22)$$

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 (19) and (20), the order of convergence of the gradients in (21) and (22) is quadratic and linear, respectively. However, the constant of convergence of this approximation is larger than the approximations in (19) and (20) 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.

3.1.2 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 ¹. Let us call this solver Ψ_o and it corresponds to the time discretization of the following integral:

$$\mathbf{u}_{t_f} = \Psi_o^n(\mathbf{u}_{t_0}) \approx \mathbf{u}_t + \int_t^{t_f} \mathbf{F}(\mathbf{u}_w) dw \quad (23)$$

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

$$TLM(\mathbf{u}_t) = \frac{\partial \Psi_o(\mathbf{u}_t)}{\partial \mathbf{u}_t} \quad (24)$$

In order to use this TLM in the gradient defined in (19) and (20), 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 3.1.3 (Extended TLM for Hybrid systems). *Under the same conditions of proposition 3.1, 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:*

$$\frac{\partial \Psi(\mathbf{u}_t)}{\partial \mathbf{u}_t} = TLM(\mathbf{u}_t) + \frac{\partial}{\partial \mathbf{u}_t} \sum_{k=1}^{k=p} \frac{h^k}{k!} \mathbf{M}_\theta(\mathbf{u}_t)^{(k-1)} + O(h^{p+1}) \quad (25)$$

Since all the reminder terms in the results showed in (19) and (20) 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.:

$$\frac{\partial \Psi(\mathbf{u}_t)}{\partial \mathbf{u}_t} = TLM(\mathbf{u}_t) + h \frac{\partial \mathbf{M}_\theta(\mathbf{u}_t)}{\partial \mathbf{u}_t} + O(h^2) \quad (26)$$

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

¹Here, we assume for simplicity that the tangent linear model does not include the variation from any physical sub-model.

3.1.3 Ensemble Tangent Linear Model (ETLM) approximation (ETLM-EGA)

The Jacobian of the flow in both (19) and (20) 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 of Allen et al. [2023].

For every initial condition \mathbf{u}_t , 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:

$$\mathbf{u}_{t+h}^i = \Psi(\mathbf{u}_t^i) \text{ for } i = 1, \dots, K \quad (27)$$

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 $d_{\mathbf{u}} \times K$ 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 $\delta\mathbf{U}_{t+h}$:

$$\frac{\partial\Psi(\mathbf{u}_t)}{\partial\mathbf{u}_t} \approx ETLM = \delta\mathbf{U}_{t+h}\delta\mathbf{U}_t^T[\delta\mathbf{U}_t\delta\mathbf{U}_t^T]^{-1} \quad (28)$$

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 $d_{\mathbf{u}}$, 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.

3.2 Implementation in deep learning frameworks

Based on the above formulations, one can derive approximations for the gradient of the online cost (9). This involves substituting the gradient of the solver in (11) with any of the provided methods and neglecting the $O(\cdot)$ term.

$$\begin{aligned} \frac{\partial\mathcal{J}}{\partial\theta}(\mathbf{g}^\dagger(\mathbf{u}_{t+nh}^\dagger), \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta) &= \frac{\partial\mathcal{J}(\cdot, \cdot, \theta)}{\partial\theta} + \frac{\partial\mathcal{J}(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \cdot)}{\partial\mathbf{g}} \frac{\partial\mathbf{g}}{\partial\Psi} \frac{\partial\Psi^n(\mathbf{u}_t)}{\partial\theta} \\ &= \mathbf{v} + \mathbf{w}\mathbf{A}_{l,p} + O(h^p) \end{aligned} \quad (29)$$

where $\mathbf{v} = \frac{\partial\mathcal{J}(\cdot, \cdot, \theta)}{\partial\theta}$ represents the gradient of the regularization, $\mathbf{w} = \frac{\partial\mathcal{J}(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \cdot)}{\partial\mathbf{g}} \frac{\partial\mathbf{g}}{\partial\Psi}$ 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 $\mathbf{J}_{j,l} = \left(\prod_{i=1}^{i=n-j} \frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)} \right)$ is defined from a Jacobian approximation as follows:

$$\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)} = \begin{cases} \frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}, & \text{if } l = 1 \text{ (EGA)} \\ \mathbf{I}, & \text{if } l = 2 \text{ (Static-EGA)} \\ TLM, & \text{if } l = 3 \text{ (TLM-EGA, Eqs. (26))} \\ ETLM, & \text{if } l = 4 \text{ (ETLM-EGA, Eqs. (28))} \end{cases} \quad (30)$$

The implementation of (29) 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 transforms Bradbury et al. [2018], Horace He [2021], or on the modification of the backward call in standard automatic differentiation languages. In the appendix (see Appendix F), we provide algorithms that illustrate practical implementations of these two methodologies.

4 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 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 system Lorenz [1996], as well as the details on the parameterization of the deep learning sub-models, training data, objective functions, and baseline methods are given in the appendix.

4.1 Lorenz 63 system

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

$$\begin{aligned} \dot{u}_{t,1}^\dagger &= \sigma(u_{t,2}^\dagger - u_{t,1}^\dagger) \\ \dot{u}_{t,2}^\dagger &= \rho u_{t,1}^\dagger - u_{t,2}^\dagger - u_{t,1}^\dagger u_{t,3}^\dagger \\ \dot{u}_{t,3}^\dagger &= u_{t,1}^\dagger u_{t,2}^\dagger - \beta u_{t,3}^\dagger \end{aligned} \quad (31)$$

Under parameterization $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$, this system exhibits chaotic dynamics with a strange attractor Lorenz [1963].

We assume that we are provided with \mathbf{F} , an imperfect version of the Lorenz system (31) 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:

$$\dot{\mathbf{u}}_t = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t) \quad (32)$$

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:

$$\begin{aligned} F_1(\mathbf{u}_t) &= \sigma(u_{t,2} - u_{t,1}) \\ F_2(\mathbf{u}_t) &= \rho u_{t,1} - u_{t,2} - u_{t,1} u_{t,3} \\ F_3(\mathbf{u}_t) &= u_{t,1} u_{t,2} \end{aligned} \quad (33)$$

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

4.1.1 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 3.1 equation (19) and the one of the corollary 3.1.2 equation (21). 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 in Chen et al. [2018]). 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 3.1 and, importantly, the result of the Static-EGA (corollary 3.1.2 equation (21)), that will be used in the following experiments.

4.1.2 Sub-Model performance

In this experiment, we evaluate the performance of the proposed online learning techniques in deriving a relevant sub-model \mathbf{M}_θ . We assume that we are provided with a dataset of the true system in (31), $\mathcal{D}_h = \{(u_{t_k+jh_i}^\dagger, u_{t_k}^\dagger) \mid \text{with } k = 1 \dots N \text{ and } j = 1 \dots n\}$ and we aim at optimizing the parameters of the sub-model \mathbf{M}_θ to minimize the cost of the online objective function (9). We evaluate two of the proposed online learning schemes, the Static-EGA given in Eq. (21) and the ETLM-EGA in which the Jacobian is approximated using an ensemble (28). 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.

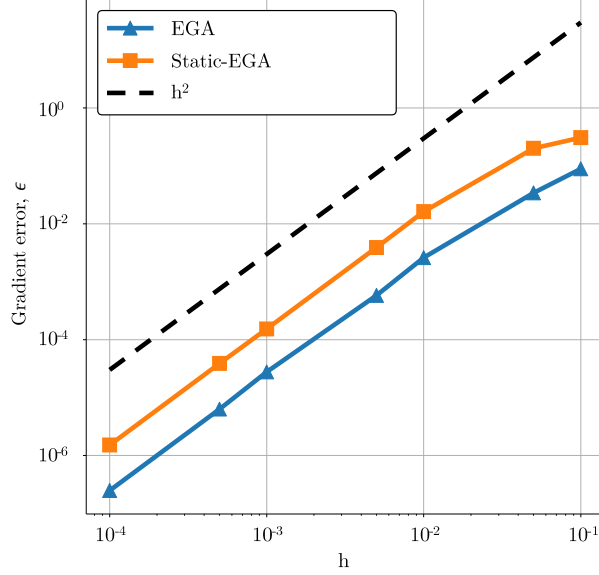


Figure 2: **Order of convergence of the proposed Euler gradient approximations.** We provide experimental validation of the gradient based on both the EGA (19) and Static-EGA (21) formulas. The error is computed by comparing these gradients to the exact ones, returned using automatic differentiation of the solver Ψ . We recall that in the EGA, the Jacobian of the flow is computed exactly using automatic differentiation.

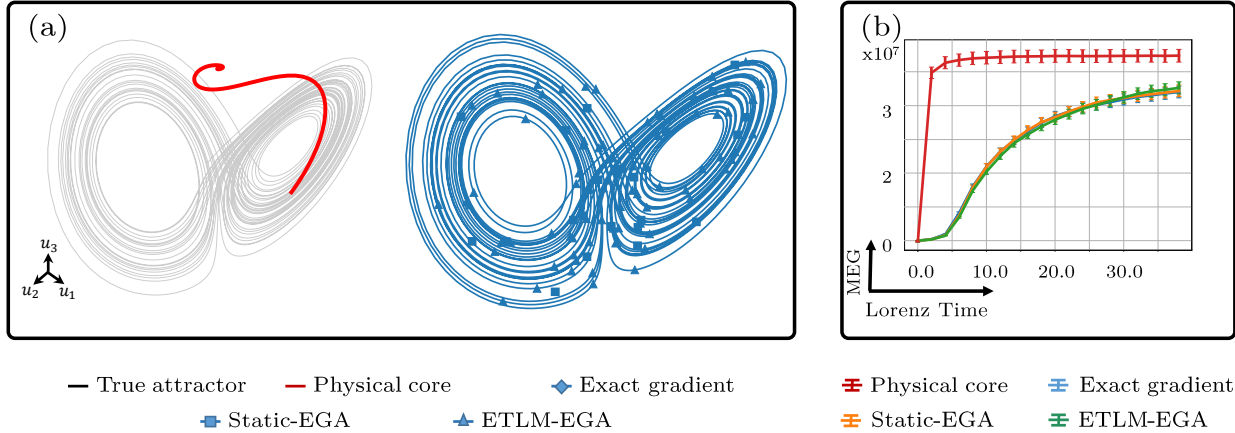


Figure 3: **Qualitative analysis of the tested models in the Lorenz 63 experiment.** (a) The attractor of the true Lorenz 63 model is compared to both the physical core (without the neural network correction term M_θ) and to the hybrid models (with the neural network corrections). The sub-models are 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 $1/20$.

A qualitative analysis of both the simulation and short-term forecasting performance of the tested models is given in Figure 3. 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 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.

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

Table 1: **Simulation performance the data-driven models**: 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 2.064 Sprott [2003].

4.2 Quasi-Geostrophic turbulence

4.2.1 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:

$$\frac{\partial \omega_t}{\partial t} + \mathcal{A}(\omega_t, \psi_t) = \frac{1}{\text{Re}} \nabla^2 \omega_t - f - r\omega_t \quad (34a)$$

$$\nabla^2 \psi_t = -\omega_t \quad (34b)$$

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 forcing Chandler and Kerswell [2013]:

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

4.2.2 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 (34). We are specifically interested in a Large Eddy Simulation (LES) setting in which the vorticity and streamfunction are filtered using a Gaussian filter Sagaut [2005], denoted by $\overline{(\cdot)}$. Applying this filter to Eqs. (34a)-(34b) yields:

$$\frac{\partial \bar{\omega}_t}{\partial t} + \mathcal{A}(\bar{\omega}_t, \bar{\psi}_t) = \frac{1}{\text{Re}} \nabla^2 \bar{\omega}_t - \bar{f} - r\bar{\omega}_t + \underbrace{\mathcal{A}(\bar{\omega}_t, \bar{\psi}_t) - \overline{\mathcal{A}(\omega_t, \psi_t)}}_{\Pi_t \approx \mathbf{M}_\theta} \quad (35a)$$

$$\nabla^2 \bar{\psi}_t = -\bar{\omega}_t \quad (35b)$$

When compared to the direct numerical simulation (DNS) of equations (34a)-(34b), 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) $\mathbf{M}_\theta(\bar{\psi}_t, \bar{\omega}_t)$ 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) parameterization Germano et al. [1991], in which the diffusion coefficient is constrained to be positive Frezat et al. [2022] in order to avoid numerical instabilities related to energy backscattering Guan et al. [2022], Frezat et al. [2022].

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

 Table 2: *Correlation coefficients between the predicted and true subgrid scale term.*

4.2.3 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 $(\psi_t, \bar{\omega}_t)$ within the testing set. We use a commonly used metric Frezat et al. [2022], Guan et al. [2023], the correlation coefficient c between the modeled $\mathbf{M}_\theta(\psi_t, \bar{\omega}_t)$ 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 findings Frezat et al. [2022], 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 works Frezat et al. [2022] that offline learning performs better on offline metrics than online learning-based models.

4.2.4 Online analysis

Here, we evaluate the ability of the trained hybrid models to reproduce the dynamics of the filtered DNS simulation. Figure 4 shows a simulation example from an initial condition in the test set. A visual analysis reveals that the DSMAG scheme (purple panel in Fig. 4) 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. 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. 4, 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 filtered DNS. The blue panel in Figure 4, 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 Figure 5. Overall, the analysis of the Probability Density Function (PDF) of the vorticity field in Fig. 5 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.

4.2.5 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 6 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 Figure 6, 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 6, now aligning closely with the filtered DNS spectra.

5 Outlook

With these promising results, we foresee future investigations to scale such online learning schemes to higher-complexity simulation frameworks. Challenges include accounting in the proposed approach for multiple sub-models that might not be additive. Extensions to stochastic sub-models are also appealing to address probabilistic forecasting and data assimilation challenges. Finally, we discuss the decomposition of the learned sub-models into a combination of interpretable physical processes.

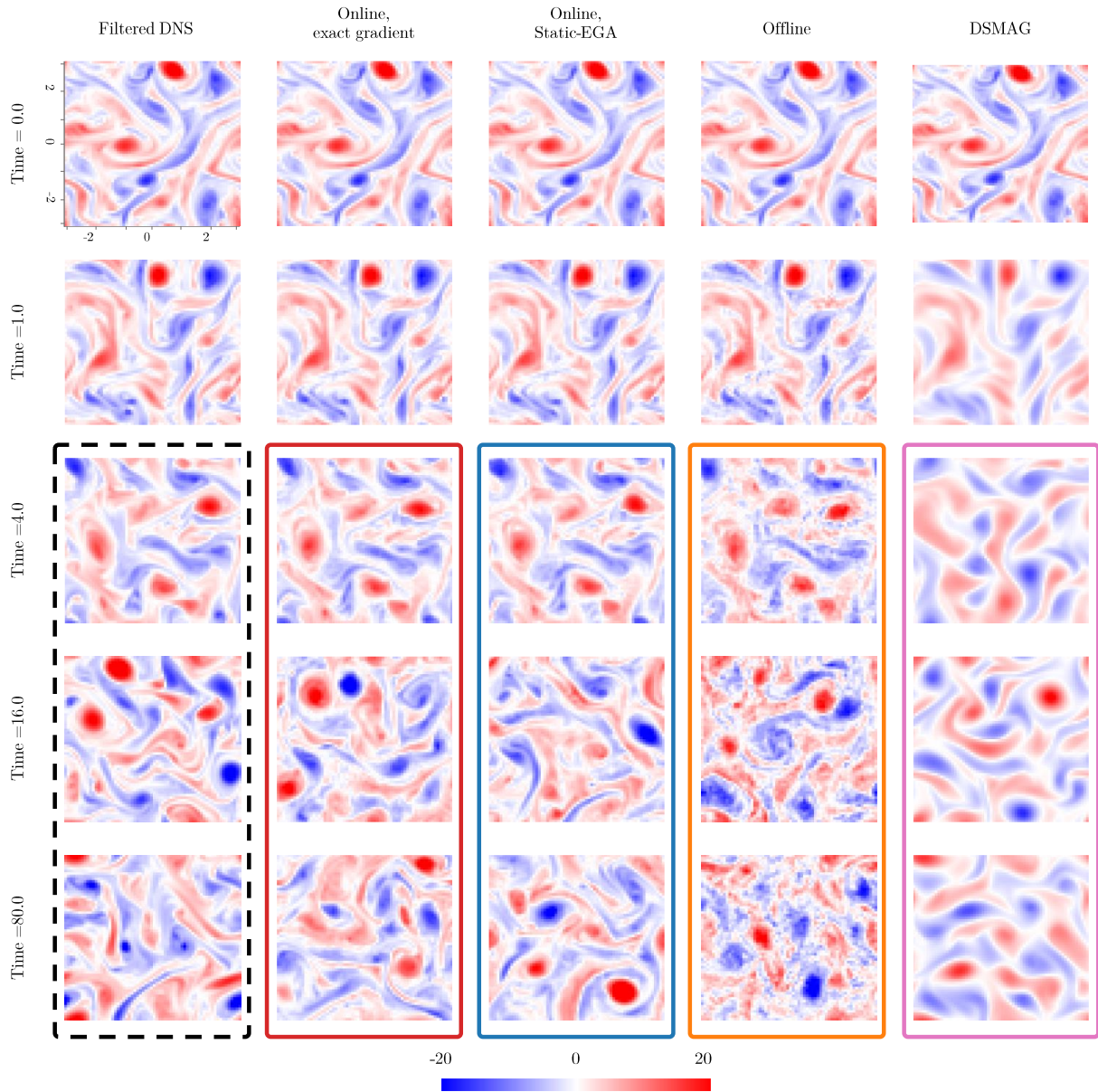


Figure 4: *Vorticity field simulation example for the different models.* Spectral analysis of the simulated trajectories, as well as the time series of the energy and enstrophy fields are given in Figure 5.

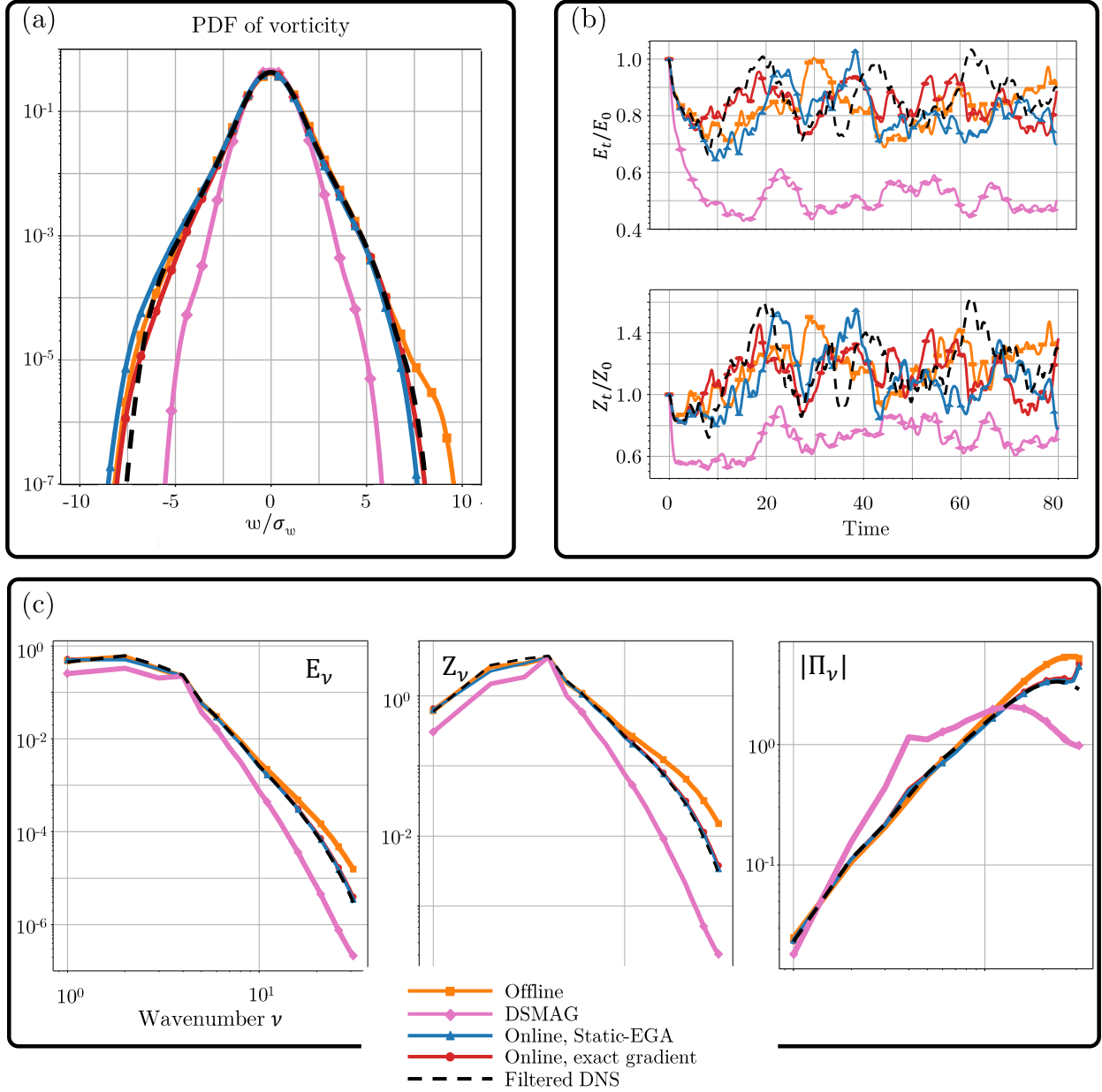


Figure 5: **Statistical evaluation of the different models.** (a) Probability density function of the vorticity field (b) Time evolution of the kinetic energy $E_t = \frac{1}{2} \langle \overline{\psi_t \overline{\omega}_t} \rangle$ and enstrophy $Z_t = \frac{1}{2} \langle \overline{\omega_t^2} \rangle$ 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.

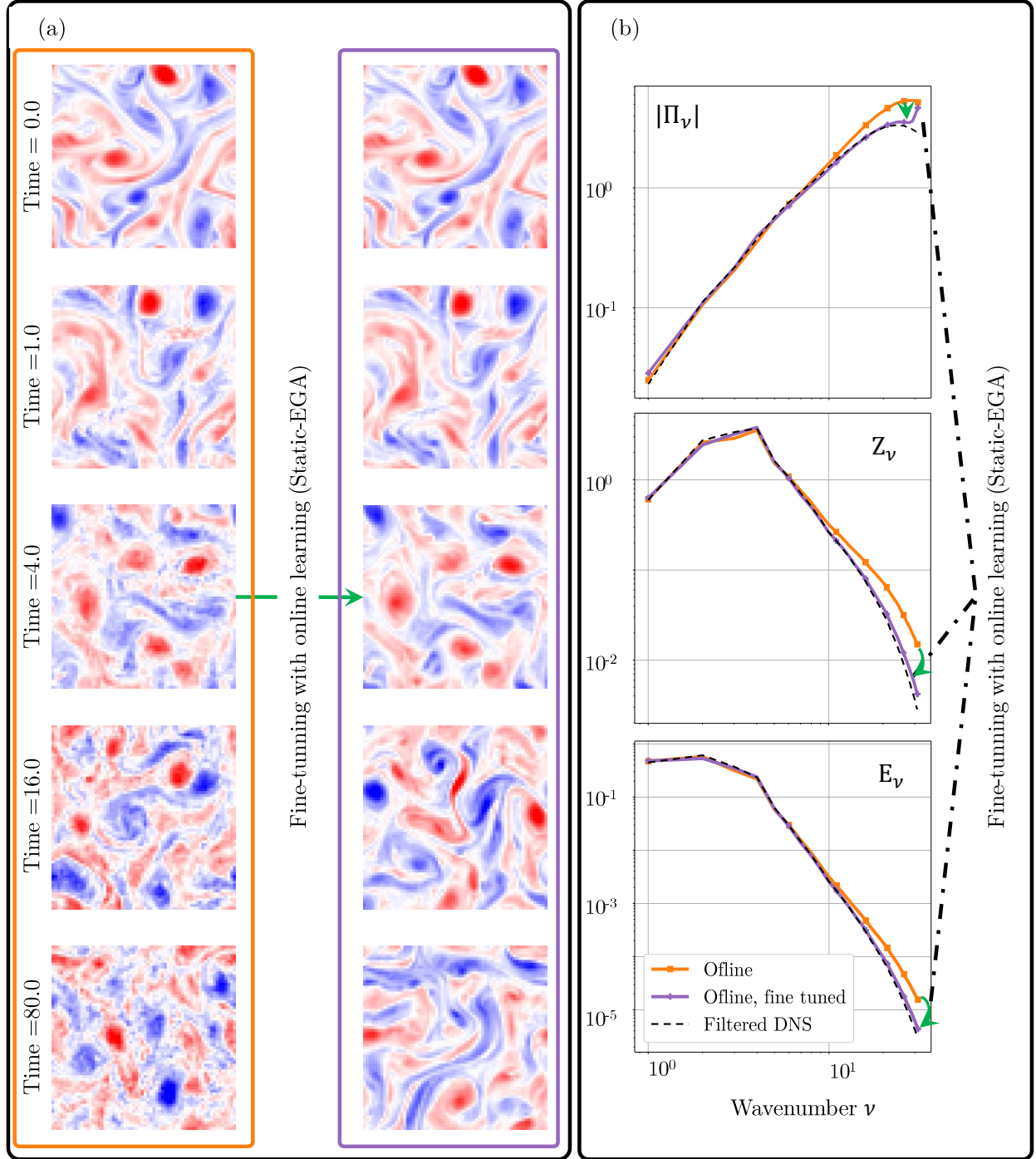


Figure 6: *Vorticity field and time-averaged spectrums of the offline model before and after fine-tuning.* (a) The same offline model that leads to an unphysical vorticity field in Figure 4 is fine-tuned using the proposed static approximation of the online learning problem, leading to a realistic vorticity simulation. (b) Time-averaged kinetic energy spectra E_ν , enstrophy spectra Z_ν and power spectrum of the SGS term $|\Pi_\nu|$ of the offline model before and after the online fine-tuning.

5.1 Extension to multiple sub-models

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:

$$\begin{cases} \dot{\mathbf{u}}_t &= \mathbf{F}(\mathbf{u}_t) + \sum_{i=1}^q \mathbf{M}_\theta^i(\mathbf{u}_t) \\ \mathbf{u}(0) &= \mathbf{u}_0 \end{cases} \quad (36)$$

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 2.2, a reference dataset \mathbf{R}_t^i 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:

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

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} .

5.2 Stochastic hybrid models

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 assimilation Zhen et al. [2023]. 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 Mémin [2014], Holm [2015]. 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.

5.3 Beyond additive sub-models

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 .

5.4 Towards explainable sub-models

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.

6 conclusion

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 in Pathak et al. [2022], Lam et al. [2022], Dueben et al. [2022], Chen et al. [2023], Bi et al. [2023], Nguyen et al. [2023].

Acknowledgement

This work was supported by the ERC Synergy project 856408-STUOD, Labex Cominlabs (grant SEACS), CNES (grant OSTST-MANATEE), Microsoft (AI EU Ocean awards), ANR Projects Melody and OceaniX. It benefited from HPC and GPU resources from Azure (Microsoft EU Ocean awards) and GENCI-IDRIS (Grant 2020-101030).

References

- Jorgen S Frederiksen and Antony G Davies. Eddy viscosity and stochastic backscatter parameterizations on the sphere for atmospheric circulation models. *Journal of the atmospheric sciences*, 54(20):2475–2492, 1997.
- Glenn Shutts. A kinetic energy backscatter algorithm for use in ensemble prediction systems. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(612):3079–3102, 2005.
- Stephan Juricke, Sergey Danilov, Nikolay Koldunov, Marcel Oliver, and Dmitry Sidorenko. Ocean kinetic energy backscatter parametrization on unstructured grids: Impact on global eddy-permitting simulations. *Journal of Advances in Modeling Earth Systems*, 12(1):e2019MS001855, 2020.
- Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
- Said Ouala, Laurent Debreu, Ananda Pascual, Bertrand Chapron, Fabrice Collard, Lucile Gaultier, and Ronan Fablet. Learning runge-kutta integration schemes for ode simulation and identification. *arXiv preprint arXiv:2105.04999*, 2021.
- Sam Partee, Matthew Ellis, Alessandro Rigazzi, Andrew E Shao, Scott Bachman, Gustavo Marques, and Benjamin Robbins. Using machine learning at scale in numerical simulations with smartsim: An application to ocean climate modeling. *Journal of Computational Science*, 62:101707, 2022.
- Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *J. Mach. Learn. Res.*, 24(89):1–97, 2023.
- Romit Maulik, Omer San, Jamey D Jacob, and Christopher Crick. Sub-grid scale model classification and blending through deep learning. *Journal of Fluid Mechanics*, 870:784–812, 2019.
- Laure Zanna and Thomas Bolton. Data-driven equation discovery of ocean mesoscale closures. *Geophysical Research Letters*, 47(17):e2020GL088376, 2020.

- Abhinav Gupta and Pierre FJ Lermusiaux. Neural closure models for dynamical systems. *Proceedings of the Royal Society A*, 477(2252):20201004, 2021.
- Alexis-Tzianni G Charalampopoulos and Themistoklis P Sapsis. Machine-learning energy-preserving nonlocal closures for turbulent fluid flows and inertial tracers. *Physical Review Fluids*, 7(2):024305, 2022.
- Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.
- Adam Subel, Yifei Guan, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Explaining the physics of transfer learning in data-driven turbulence modeling. *PNAS nexus*, 2(3):pgad015, 2023.
- Zheng Wang, Dunhui Xiao, Fangxin Fang, Rajesh Govindan, Christopher C Pain, and Yike Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.
- Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.
- S Ouala, D Nguyen, L Drumetz, B Chapron, A Pascual, F Collard, L Gaultier, and R Fablet. Learning latent dynamics for partially observed chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(10):103121, 2020.
- Kazuto Hasegawa, Kai Fukami, Takaaki Murata, and Koji Fukagata. Machine-learning-based reduced-order modeling for unsteady flows around bluff bodies of various shapes. *Theoretical and Computational Fluid Dynamics*, 34: 367–383, 2020.
- Said Ouala, Bertrand Chapron, Fabrice Collard, Lucile Gaultier, and Ronan Fablet. Extending the extended dynamic mode decomposition with latent observables: the latent edmd framework. *Machine Learning: Science and Technology*, 4(2):025018, 2023a.
- Louis Serrano, Leon Migus, Yuan Yin, Jocelyn Ahmed Mazari, and Patrick Gallinari. Infinity: Neural field modeling for reynolds-averaged navier-stokes equations. *arXiv preprint arXiv:2307.13538*, 2023.
- Said Ouala, Cedric Herzet, and Ronan Fablet. Sea surface temperature prediction and reconstruction using patch-level neural network representations. *arXiv:1806.00144 [cs, stat]*, may 2018a. URL <http://arxiv.org/abs/1806.00144>. arXiv: 1806.00144.
- Sibo Cheng, I Colin Prentice, Yuhan Huang, Yufang Jin, Yi-Ke Guo, and Rossella Arcucci. Data-driven surrogate model with latent data assimilation: Application to wildfire forecasting. *Journal of Computational Physics*, 464: 111302, 2022.
- Léon Migus, Julien Salomon, and Patrick Gallinari. Stability of implicit neural networks for long-term forecasting in dynamical systems. *arXiv preprint arXiv:2305.17155*, 2023.
- Duong Nguyen, Said Ouala, Lucas Drumetz, and Ronan Fablet. Em-like learning chaotic dynamics from noisy and partial observations. *SciRate*, Mar. 2019. URL <https://scirate.com/arxiv/1903.10335>.
- Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the lorenz 96 model. *Geoscientific Model Development Discussions*, pages 1–21, 2019.
- Marc Bocquet, Julien Brajard, Alberto Carrassi, and Laurent Bertino. Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization. *arXiv preprint arXiv:2001.06270*, 2020.
- Said Ouala, Steven L Brunton, Bertrand Chapron, Ananda Pascual, Fabrice Collard, Lucile Gaultier, and Ronan Fablet. Bounded nonlinear forecasts of partially observed geophysical systems with physics-constrained deep learning. *Physica D: Nonlinear Phenomena*, 446:133630, 2023b.
- Said Ouala, Ronan Fablet, Cédric Herzet, Bertrand Chapron, Ananda Pascual, Fabrice Collard, and Lucile Gaultier. Neural network based kalman filters for the spatio-temporal interpolation of satellite-derived sea surface temperature. *Remote Sens.*, 10(12):1864, Nov 2018b. ISSN 2072-4292. doi:10.3390/rs10121864. URL <http://dx.doi.org/10.3390/rs10121864>.
- Sibo Cheng, César Quilodrán-Casas, Said Ouala, Alban Farchi, Che Liu, Pierre Tandeo, Ronan Fablet, Didier Lucor, Bertrand Iooss, Julien Brajard, et al. Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review. *IEEE/CAA Journal of Automatica Sinica*, 10(6):1361–1387, 2023.
- Said Ouala, Pierre Tandeo, Bertrand Chapron, Fabrice Collard, and Ronan Fablet. End-to-end kalman filter in a high dimensional linear embedding of the observations. *Stochastic Transport in Upper Ocean Dynamics*, page 211, 2023c.

- Sibo Cheng, Che Liu, Yike Guo, and Rossella Arcucci. Efficient deep data assimilation with sparse observations and time-varying sensors. *Journal of Computational Physics*, 496:112581, 2024.
- Eurika Kaiser, Marek Morzyński, Guillaume Daviller, J Nathan Kutz, Bingni W Brunton, and Steven L Brunton. Sparsity enabled cluster reduced-order models for control. *Journal of Computational Physics*, 352:388–409, 2018.
- Steven L Brunton. Machine learning of dynamics with applications to flow control and aerodynamic optimization. In *Advances in Critical Flow Dynamics Involving Moving/Deformable Structures with Design Applications: Proceedings of the IUTAM Symposium on Critical Flow Dynamics involving Moving/Deformable Structures with Design applications, June 18-22, 2018, Santorini, Greece*, pages 327–335. Springer, 2021.
- Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.
- Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.
- Peter D Dueben, Martin G Schultz, Matthew Chantry, David John Gagne, David Matthew Hall, and Amy McGovern. Challenges and benchmark datasets for machine learning in the atmospheric sciences: Definition, status, and outlook. *Artificial Intelligence for the Earth Systems*, 1(3):e210002, 2022.
- Kang Chen, Tao Han, Junchao Gong, Lei Bai, Fenghua Ling, Jing-Jia Luo, Xi Chen, Leiming Ma, Tianning Zhang, Rui Su, et al. Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv preprint arXiv:2304.02948*, 2023.
- Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023.
- Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023.
- Shijie Jiang, Yi Zheng, and Dimitri Solomatine. Improving ai system awareness of geoscience knowledge: Symbiotic integration of physical approaches and deep learning. *Geophysical Research Letters*, 47(13):e2020GL088229, 2020.
- Gustau Camps-Valls, Markus Reichstein, Xiaoxiang Zhu, and Devis Tuia. Advancing deep learning for earth sciences: From hybrid modeling to interpretability. In *IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium*, pages 3979–3982. IEEE, 2020.
- Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to infer unresolved scale parametrization. *Philosophical Transactions of the Royal Society A*, 379(2194): 20200086, 2021.
- Yifei Guan, Ashesh Chattopadhyay, Adam Subel, and Pedram Hassanzadeh. Stable a posteriori les of 2d turbulence using convolutional neural networks: Backscattering analysis and generalization to higher re via transfer learning. *Journal of Computational Physics*, 458:111090, 2022.
- Yifei Guan, Adam Subel, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate les. *Physica D: Nonlinear Phenomena*, 443:133568, 2023.
- Karan Jakhar, Yifei Guan, Rambod Mojjani, Ashesh Chattopadhyay, Pedram Hassanzadeh, and Laura Zanna. Learning closed-form equations for subgrid-scale closures from high-fidelity data: Promises and challenges. *arXiv preprint arXiv:2306.05014*, 2023.
- Alban Farchi, Marcin Chrust, Marc Bocquet, Patrick Laloyaux, and Massimo Bonavita. Online model error correction with neural networks in the incremental 4d-var framework. *Journal of Advances in Modeling Earth Systems*, 15(9): e2022MS003474, 2023.
- Chaopeng Shen, Alison P Appling, Pierre Gentine, Toshiyuki Bandai, Hoshin Gupta, Alexandre Tartakovsky, Marco Baity-Jesi, Fabrizio Fenicia, Daniel Kifer, Li Li, et al. Differentiable modelling to unify machine learning and physical models for geosciences. *Nature Reviews Earth & Environment*, 4(8):552–567, 2023.
- Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33: 6111–6122, 2020.
- Hugo Frezat, Julien Le Sommer, Ronan Fablet, Guillaume Balarac, and Redouane Lguensat. A posteriori learning for quasi-geostrophic turbulence parametrization. *Journal of Advances in Modeling Earth Systems*, 14(11): e2022MS003124, 2022.

- Hugo Frezat, Ronan Fablet, Guillaume Balarac, and Julien Le Sommer. Gradient-free online learning of subgrid-scale dynamics with neural emulators, 2023.
- Hugo Frezat, Guillaume Balarac, Julien Le Sommer, Ronan Fablet, and Redouane Lguensat. Physical invariance in neural networks for subgrid-scale scalar flux modeling. *arXiv preprint arXiv:2010.04663*, 2020.
- Andrew Ross, Ziwei Li, Pavel Perezhogin, Carlos Fernandez-Granda, and Laure Zanna. Benchmarking of machine learning ocean subgrid parameterizations in an idealized model. *Journal of Advances in Modeling Earth Systems*, 15(1):e2022MS003258, 2023.
- Marcel Nonnenmacher and David S Greenberg. Deep emulators for differentiation, forecasting, and parametrization in earth science simulators. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002554, 2021.
- Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pages 6571–6583, 2018.
- Frédéric Hourdin, Thorsten Mauritsen, Andrew Gettelman, Jean-Christophe Golaz, Venkatramani Balaji, Qingyun Duan, Doris Folini, Duoying Ji, Daniel Klocke, Yun Qian, et al. The art and science of climate model tuning. *Bulletin of the American Meteorological Society*, 98(3):589–602, 2017.
- Joseph Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164, 1963.
- Massimo Germano, Ugo Piomelli, Parviz Moin, and William H Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, 1991.
- Parviz Moin, Kyle Squires, W Cabot, and Sangsan Lee. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Physics of Fluids A: Fluid Dynamics*, 3(11):2746–2757, 1991.
- Philippe Spalart and Steven Allmaras. A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439, 1992.
- David C Wilcox. Formulation of the kw turbulence model revisited. *AIAA journal*, 46(11):2823–2838, 2008.
- Etienne Mémin. Fluid flow dynamics under location uncertainty. *Geophysical & Astrophysical Fluid Dynamics*, 108(2): 119–146, 2014.
- Darryl D Holm. Variational principles for stochastic fluid dynamics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2176):20140963, 2015.
- Valentin Resseguier, Etienne Mémin, and Bertrand Chapron. Geophysical flows under location uncertainty, part i random transport and general models. *Geophysical & Astrophysical Fluid Dynamics*, 111(3):149–176, 2017a.
- Valentin Resseguier, Etienne Mémin, and Bertrand Chapron. Geophysical flows under location uncertainty, part ii quasi-geostrophy and efficient ensemble spreading. *Geophysical & Astrophysical Fluid Dynamics*, 111(3):177–208, 2017b.
- Valentin Resseguier, Etienne Mémin, and Bertrand Chapron. Geophysical flows under location uncertainty, part iii sqg and frontal dynamics under strong turbulence conditions. *Geophysical & Astrophysical Fluid Dynamics*, 111(3): 209–227, 2017c.
- Samuel Lanthaler and Andrew M Stuart. The curse of dimensionality in operator learning. *arXiv preprint arXiv:2306.15924*, 2023.
- Marco A Iglesias, Kody JH Law, and Andrew M Stuart. Ensemble kalman methods for inverse problems. *Inverse Problems*, 29(4):045001, 2013.
- Nikola B Kovachki and Andrew M Stuart. Ensemble kalman inversion: a derivative-free technique for machine learning tasks. *Inverse Problems*, 35(9):095005, 2019.
- Lucas Böttcher. Gradient-free training of neural odes for system identification and control using ensemble kalman inversion. *arXiv preprint arXiv:2307.07882*, 2023.
- Rikhi Bose and Arunabha M Roy. Accurate deep learning sub-grid scale models for large eddy simulations. *arXiv preprint arXiv:2307.10060*, 2023.
- Ashesh Chattopadhyay and Pedram Hassanzadeh. Long-term instability of deep learning-based digital twins of the climate system: Cause and solution. *Bulletin of the American Physical Society*, 2023.
- Joel Andersson. A general-purpose software framework for dynamic optimization (een algemene softwareomgeving voor dynamische optimalisatie). 2013.
- Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019.

- Douglas R Allen, Daniel Hodyss, Karl W Hoppel, and Gerald E Nedoluha. The ensemble tangent linear model applied to nonorographic gravity wave drag. *Monthly Weather Review*, 151(1):193–210, 2023.
- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.
- Richard Zou Horace He. functorch: Jax-like composable function transforms for pytorch. <https://github.com/pytorch/functorch>, 2021.
- Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996.
- Edward N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, March 1963. ISSN 0022-4928. doi:10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2. URL [http://journals.ametsoc.org/doi/abs/10.1175/1520-0469\(1963\)020%3C0130:DNF%3E2.0.CO;2](http://journals.ametsoc.org/doi/abs/10.1175/1520-0469(1963)020%3C0130:DNF%3E2.0.CO;2).
- Julien Clinton Sprott. *Chaos and Time-Series Analysis*. Oxford University Press, Inc., New York, NY, USA, 2003. ISBN 0198508409.
- Gary J Chandler and Rich R Kerswell. Invariant recurrent solutions embedded in a turbulent two-dimensional kolmogorov flow. *Journal of Fluid Mechanics*, 722:554–595, 2013.
- Pierre Sagaut. *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media, 2005.
- Yicun Zhen, Valentin Resseguier, and Bertrand Chapron. Physically constrained covariance inflation from location uncertainty. *EGUsphere*, 2023:1, 2023.
- A. C. Hindmarsh. ODEPACK, a systematized collection of ODE solvers. *IMACS Transactions on Scientific Computation*, 1:55–64, 1983.
- Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016. ISSN 0027-8424, 1091-6490. doi:10.1073/pnas.1517384113. URL <http://www.pnas.org/lookup/doi/10.1073/pnas.1517384113>.
- Redouane Lguensat, Pierre Tandeo, Pierre Ailliot, Manuel Pulido, and Ronan Fablet. The Analog Data Assimilation. *Monthly Weather Review*, aug 2017. ISSN 0027-0644, 1520-0493. doi:10.1175/MWR-D-16-0441.1. URL <http://journals.ametsoc.org/doi/10.1175/MWR-D-16-0441.1>.
- Thomas S. Parker and Leon O. Chua. *Stability of Limit Sets*, pages 57–82. Springer New York, New York, NY, 1989a. ISBN 978-1-4612-3486-9. doi:10.1007/978-1-4612-3486-9_3. URL https://doi.org/10.1007/978-1-4612-3486-9_3.
- Thomas S. Parker and Leon O. Chua. *Dimension*, pages 167–199. Springer New York, New York, NY, 1989b. ISBN 978-1-4612-3486-9. doi:10.1007/978-1-4612-3486-9_7. URL https://doi.org/10.1007/978-1-4612-3486-9_7.
- E.N. Lorenz. *Predictability: a problem partly solved*. PhD thesis, Shinfield Park, Reading, 1995 1995.
- R. Fablet, S. Ouala, and C. Herzet. Bilinear residual neural network for the identification and forecasting of geophysical dynamics. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1477–1481, Sep. 2018. doi:10.23919/EUSIPCO.2018.8553492.

A Proof of the proposition 3.1

Since the solver Ψ is of order p , and the explicit Euler scheme has order 1 it is trivial to prove proposition (3.1) from the Taylor expansion of Ψ . We include the proof here for completeness. We can write as h approaches zero:

$$\begin{aligned}
 \mathbf{u}_{t+h} &= \Psi(\mathbf{u}_t) \\
 &= \mathbf{u}_t + \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t))^{k-1} + O(h^{p+1}) \\
 &= \underbrace{\mathbf{u}_t + h(\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t))}_{\Psi_E(\mathbf{u}_t)} + \sum_{k=2}^p h^k \frac{1}{k!} (\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t))^{k-1} + O(h^{p+1}) \\
 &= \Psi_E(\mathbf{u}_t) + \underbrace{\sum_{k=2}^p h^k \frac{1}{k!} (\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t))^{k-1}}_{<Ch^2} + O(h^{p+1}) \\
 &= \Psi_E(\mathbf{u}_t) + O(h^2)
 \end{aligned} \tag{37}$$

The second part of the proposition can be proven similarly by replacing \mathbf{u}_t by $\Psi^{n-1}(\mathbf{u}_t)$.

B Proof of theorem 3.1

Notice that for every n , the following holds:

$$\begin{aligned}
 \frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) &= \frac{\partial}{\partial \theta} \Psi \circ \Psi^{n-1}(\mathbf{u}_t) \\
 &= \underbrace{\frac{\partial \Psi(\Psi^{n-1}(\mathbf{u}_t))}{\partial \Psi^{n-1}(\mathbf{u}_t)} \frac{\partial \Psi^{n-1}(\mathbf{u}_t)}{\partial \theta}}_{\text{Variation due to the initial condition}} + \underbrace{\frac{\partial \Psi(\Psi^{n-1}(\mathbf{u}_t))}{\partial \theta}}_{\text{Variation of the gradient of the solver given the initial condition}}
 \end{aligned} \tag{38}$$

We use (38) to construct the following:

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) \frac{\partial}{\partial \theta} \Psi(\Psi^{j-1}(\mathbf{u}_t)) + \frac{\partial}{\partial \theta} \Psi(\Psi^{n-1}(\mathbf{u}_t)) \tag{39}$$

where all the derivatives with respect to θ are taken assuming that the initial condition is fixed. The proof of (39) is conducted by recurrence, and is given in appendix J.

To prove theorem 3.1, we simply replace in (39) the solver Ψ by its Euler approximation given in the proposition 3.1:

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) \frac{\partial}{\partial \theta} (\Psi_E(\Psi^{j-1}(\mathbf{u}_t)) + O(h^2)) + \frac{\partial}{\partial \theta} (\Psi_E(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2)) \tag{40}$$

if we develop the expression of the explicit Euler solver, we have:

$$\begin{aligned}
 \frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) &= \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) \frac{\partial}{\partial \theta} (\Psi^{j-1}(\mathbf{u}_t) + h(\mathbf{F}(\Psi^{j-1}(\mathbf{u}_t)) + \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t))) + O(h^2)) \\
 &+ \frac{\partial}{\partial \theta} (\Psi^{n-1}(\mathbf{u}_t) + h(\mathbf{F}(\Psi^{n-1}(\mathbf{u}_t)) + \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t))) + O(h^2)) \\
 &= \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) O(h^2) \\
 &+ h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \\
 &= \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + nO(h^2)
 \end{aligned} \tag{41}$$

The leading order term in $\sum_{j=1}^{j=n-1} (\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)}) O(h^2)$ is $O(h^2)$. Furthermore, and since n is fixed, $nO(h^2)$ is simply equal to $O(h^2)$. Reporting this in equation (41) completes the proof as follows:

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \quad (42)$$

C Proof of the corollary 3.1.1

We prove corollary (3.1.1), we start from (41) and replace n by $\frac{t_f - t_0}{h}$. This makes the convergence linear in h :

$$\frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + O(h) \quad (43)$$

D Proof of the Corollary 3.1.2

In order to prove corollary 3.1.2, we write the Jacobian of the Taylor expansion of the solver Ψ and we keep the zero order term.

$$\begin{aligned} \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \Psi(\Psi(\mathbf{u}_t)) &= \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \left(\Psi(\mathbf{u}_t) + \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{F}(\Psi(\mathbf{u}_t)) + \mathbf{M}_\theta(\Psi(\mathbf{u}_t)))^{k-1} + O(h^{p+1}) \right) \\ &= \mathbf{I} + \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{F}(\Psi(\mathbf{u}_t)) + \mathbf{M}_\theta(\Psi(\mathbf{u}_t)))^{k-1} + O(h^{p+1}) \\ &= \mathbf{I} + O(h) \end{aligned} \quad (44)$$

If we replace the Jacobian in (19) by the approximation in (44):

$$\begin{aligned} \frac{\partial}{\partial \theta} \Psi^n(\mathbf{u}_t) &= \sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \mathbf{I} + O(h) \right) h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \\ &= \sum_{j=1}^{j=n-1} (\mathbf{I} + O(h)) h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \\ &= \sum_{j=1}^{j=n-1} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + \sum_{j=1}^{j=n-1} O(h) h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + O(h^2) \\ &= \sum_{j=1}^{j=n-1} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t)) + nO(h^2) \\ &= \sum_{j=1}^{j=n} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + O(h^2) \end{aligned} \quad (45)$$

The second part of the corollary, *i.e.* assuming that $n = \frac{t_f - t_0}{h}$ can be proven similarly.

E Proof of the Corollary 3.1.3

$$\begin{aligned}
 \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \Psi(\Psi(\mathbf{u}_t)) &= \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \left(\Psi(\mathbf{u}_t) + \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{F}(\Psi(\mathbf{u}_t)) + \mathbf{M}_\theta(\Psi(\mathbf{u}_t)))^{k-1} + O(h^{p+1}) \right) \\
 &= \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \left(\Psi(\mathbf{u}_t) + \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{F}(\Psi(\mathbf{u}_t)))^{k-1} + \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{M}_\theta(\Psi(\mathbf{u}_t)))^{k-1} + O(h^{p+1}) \right) \\
 &= \underbrace{\frac{\partial}{\partial \Psi(\mathbf{u}_t)} \left(\Psi(\mathbf{u}_t) + \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{F}(\Psi(\mathbf{u}_t)))^{k-1} + O(h^{p+1}) \right)}_{\frac{\partial}{\partial \Psi(\mathbf{u}_t)} \Psi_o(\Psi(\mathbf{u}_t))} + \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{M}_\theta(\Psi(\mathbf{u}_t)))^{k-1} \\
 &= \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \Psi_o(\Psi(\mathbf{u}_t)) + \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{M}_\theta(\Psi(\mathbf{u}_t)))^{k-1} \\
 &= TLM(\Psi(\mathbf{u}_t)) + \frac{\partial}{\partial \Psi(\mathbf{u}_t)} \sum_{k=1}^p h^k \frac{1}{k!} (\mathbf{M}_\theta(\Psi(\mathbf{u}_t)))^{k-1}
 \end{aligned} \tag{46}$$

F Algorithms of the proposed online learning methodology

F.1 Gradient evaluation using composable function transforms

A direct evaluation of (29) can be based on composable function transforms of modern languages such as JAX Bradbury et al. [2018] or PYTORCH (based on the functorch tool). These tools allow to evaluate vector valued gradients (not only vector Jacobian products), and it can be adapted to the computation of (29). Algorithm 1 highlights how this can be achieved.

Algorithm 1 Gradient computation based on composable function transforms

Input:

- Ψ : Non-differentiable solver of the hybrid system (5)
- \mathbf{M}_θ : Deep learning based sub-model
- \mathbf{u}_t : Initial condition
- n, h : Number of simulation steps and time step
- \mathcal{J}_{online} : Online cost function
- l : Approximation scheme for the Jacobian of the flow

return

- ▷ Iterate through the solver Ψ
 - for** $j \leftarrow 1$ **to** n **do**
 - $\mathbf{u}_{t+jh} = \Psi^j(\mathbf{u}_t)$
 - end for**
 - ▷ Precompute the Jacobians $\mathbf{J}_{j,l}$
 - for** $j \leftarrow 1$ **to** $n-1$ **do**
 - $\mathbf{J}_{j,l} = \prod_{i=1}^{i=n-j} \partial \Psi(\Psi^{n-i}(\mathbf{u}_t)) / \partial \Psi^{n-i}(\mathbf{u}_t)$
 - end for**
 - ▷ Compute the vector valued gradients of the sub-model
 - for** $j \leftarrow 1$ **to** n **do**
 - $\frac{\partial \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t))}{\partial \theta}$ ▷ This can be done, for example, using functorch
 - end for**
 - ▷ Compute the Euler approximation of the gradient of the solver
 - $\mathbf{A}_{l,p} = \sum_{j=1}^{j=n-1} \mathbf{J}_{j,l} h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{j-1}(\mathbf{u}_t)) + h \frac{\partial}{\partial \theta} \mathbf{M}_\theta(\Psi^{n-1}(\mathbf{u}_t))$
 - ▷ Compute the remaining gradients and evaluate the gradient of the online cost
 - $\mathbf{v} = \partial Q(\cdot, \cdot, \theta) / \partial \theta$
 - $\mathbf{w} = \partial Q(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \cdot) / \partial \mathbf{g} \frac{\partial \mathbf{g}}{\partial \Psi}$
 - $\partial \mathcal{J}_{online} / \partial \theta = \mathbf{v} + \mathbf{w} \mathbf{A}_{l,p}(\mathbf{u}_t)$
-

E.2 Modification of the backward call

The gradient of the online cost function can be computed by a modification of the backward call of modern automatic differentiation languages. The idea is to construct a ResNet-like computational graph using the sub-model \mathbf{M}_θ . We modify the gradient of the output of each ResNet block using a hook to include the information of the Jacobian of the non-differentiable solver. And the gradient of each block will correspond to $\partial \mathbf{M}_\theta(\cdot) / \partial \theta$. We provide an implementation of this technique, inspired by the syntax of PyTorch, in Algorithm 2.

Algorithm 2 Gradient computation based on backward call modification

Input:

- Ψ : Non-differentiable solver of the hybrid system (5)
- \mathbf{M}_θ : Deep learning based sub-model
- \mathbf{u}_t : Initial condition
- n, h : Number of simulation steps and time step
- \mathcal{J}_{online} : Online cost function
- l : Approximation scheme for the Jacobian of the flow

return

- ▷ Backward hook function
 - def** HOOK(\mathbf{z}_{t+jh}, j)
 - ▷ $\text{grad}(\cdot)$ refers to a modification of the gradient
 - $\text{grad}(\mathbf{z}_{t+jh}) = h \cdot \text{grad}(\mathbf{z}_{t+nh}) \cdot \mathbf{J}_{j,l}$ ▷ notice that $\mathbf{w} = \text{grad}(\mathbf{z}_{t+nh})$
 - end def**
 - ▷ Iterate through the solver Ψ
 - for** $j \leftarrow 1$ **to** n **do**
 - $\mathbf{u}_{t+jh} = \Psi^j(\mathbf{u}_t)$
 - end for**
 - ▷ Precompute the Jacobians $\mathbf{J}_{j,l}$
 - for** $j \leftarrow 1$ **to** $n - 1$ **do**
 - $\mathbf{J}_{j,l} = \prod_{i=1}^{i=n-j} \partial \Psi(\Psi^{n-i}(\mathbf{u}_t)) / \partial \Psi^{n-i}(\mathbf{u}_t)$
 - end for**
 - ▷ Generate a ResNet like computational graph
 - $\mathbf{z}_t = \mathbf{u}_t$ ▷ Initialize the ResNet state
 - for** $j \leftarrow 1$ **to** n **do**
 - $\mathbf{z}_{t+jh} = \mathbf{M}_\theta(\mathbf{z}_{t+(j-1)h})$
 - $\text{data}(\mathbf{z}_{t+jh}) = \mathbf{u}_{t+jh}$ ▷ $\text{data}(\cdot)$ refers to a modification of the value
 - if** $j \neq n$ **then**
 - ▷ Modify the gradient of the ResNet State
 - HOOK(\mathbf{z}_{t+jh}, j)
 - end if**
 - end for**
 - ▷ Compute the online objective function
 - $Q(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta) = Q(\cdot, \mathbf{g}(\mathbf{z}(\mathbf{u}_{t+nh})), \theta)$ ▷ The simulated states now have a computational graph
 - Backward($Q(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \theta)$) ▷ Run a backward call
-

G Training configuration in the Lorenz 63 experiment

G.1 Training data

In the first experiment with the Lorenz 63 system 4.1.1, 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 solver Hindmarsh [1983]. 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 second Lorenz 63 experiment 4.1.2, $\mathcal{D}_h = \{(u_{t_k+jh_i}^\dagger, u_{t_k}^\dagger) \mid k = 1 \dots N \text{ and } j = 1 \dots n\}$ is sampled at $h = 0.01$. The same sampling rate was used in multiple works Brunton et al. [2016], Lguensat et al. [2017], Ouala et al. [2020] that involve the data-driven identification of the Lorenz 63 system. This dataset is also

generated using the LOSDA ODE solver Hindmarsh [1983]. The number of training samples N is equal to 5000 time steps and the number of simulation time steps n is fixed to 10.

G.2 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 (32) over $n = 10$ time steps. The cost function can be written as:

$$\mathcal{J} = \frac{1}{N} \sum_k \frac{1}{n} \sum_{j=1}^n \|u_{t_k+jh}^\dagger - \Psi^j(u_{t_k}^\dagger)\|_2^2 \quad (47)$$

where $\|\cdot\|_2$ is the L2 norm. The solver Ψ used in these experiments is a differentiable DOPRI8 solver, developed in Chen et al. [2018].

G.3 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 experiment 4.1.2:

- **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 (32).
- **Online calibration with exact gradient:** The sub-model is trained by utilizing the exact gradient of the online cost (47). 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 (47), in which we use a static approximation of the Jacobian as described in (21).
- **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 (47), but the Jacobian is approximated using an ensemble as described in 28. The size of the ensemble is set to 5 members.

G.4 Evaluation criteria

In Figure 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 (29) to express this error and assuming that the exact gradient is $\frac{\partial \mathcal{J}}{\partial \theta}$, the error depicted in Figure 2 is computed as:

$$\epsilon = \frac{1}{a} \left\| \frac{\partial \mathcal{J}}{\partial \theta} - \mathbf{v} + \mathbf{w} \mathbf{A}_{l,p}(\mathbf{u}_t) \right\|_1 \quad (48)$$

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 (30)). The exact Jacobian is in this experiment evaluated using automatic differentiation.

In Figure 3, 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:

$$\text{MEG}(t_0 + ih) = \frac{\|u_{t_0+ih}^\dagger - \Psi^i(u_{t_0}^\dagger)\|_2^2}{\|u_{t_0+h}^\dagger - \Psi(u_{t_0}^\dagger)\|_2^2} \quad (49)$$

The Lyapunov spectrum in Table 1 is computed using the Gram-Schmidt orthonormalization technique Parker and Chua [1989a], and the Lyapunov dimension is deduced from the spectrum as given in Parker and Chua [1989b].

H Training configuration in the QG experiment

H.1 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

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

Table 3: *Parameters of the DNS Flow Configurations.* 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 .

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 $\{(\bar{\omega}_{t_k+jh_{\text{LES}}}, (\bar{\omega}_{t_k}, \bar{\psi}_{t_k})) \mid \text{with } k = 1 \dots N \text{ and } j = 1 \dots n\}$ where $N = 2000 \times 8$ (where 8 is the number of training datasets) and $n = 10$ for the online learning experiments and as $\{(\Pi_{t_k}, (\bar{\omega}_{t_k}, \bar{\psi}_{t_k})) \mid \text{with } k = 1 \dots N\}$.

H.2 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 (35) over $n = 10$ time steps. The cost function can be written as:

$$\mathcal{J} = \frac{1}{N} \sum_k \frac{1}{n} \sum_{j=1}^n \|\bar{\omega}_{t_k+jh} - \Psi^j(\bar{\omega}_{t_k})\| \quad (50)$$

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:

$$\mathcal{J} = \frac{1}{N} \sum_k \frac{1}{n} \sum_{j=1}^n \|\Pi_{t_k} - \mathbf{M}_\theta(\bar{\omega}_{t_k}, \bar{\psi}_{t_k})\| \quad (51)$$

H.3 Numerical Solver of the QG system

QG system is solved using a code adapted from Frezat et al. [2022]. 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.

H.4 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 resolution Guan et al. [2023]. This filtering/coarse-graining operation can be written as (taking here as an example the DNS vorticity field ω):

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

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, $\Delta t_{\text{LES}} = \delta \Delta t_{\text{DNS}}$.

I 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 in Guan et al. [2022]. 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 to Frezat et al. [2022] 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 (35) 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 in Frezat et al. [2022] 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 (21). In this experiment, the solver of (35) 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_t .
- **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 example, Guan et al. [2022].

I.1 Additional experiment on the two scale Lorenz 96

The two scale L96 system describes a coupled system of equations Lorenz [1995] with S slow variables, $u_t^\dagger = [u_{t,1}^\dagger, u_{t,2}^\dagger, \dots, u_{t,S}^\dagger]^T$ each of which is coupled to B fast variables $(y_{t,1,s}, y_{t,2,s}, \dots, y_{t,B,s})$:

$$\begin{aligned} \dot{u}_{t,s}^\dagger &= -u_{t,s-1}^\dagger (u_{t,s-2}^\dagger - u_{t,s+1}^\dagger) - u_{t,s}^\dagger + A + R_{t,s} \\ \dot{y}_{t,b,s} &= -c\gamma y_{t,b+1,s} (y_{t,b+2,s} - y_{t,b-1,s}) - cy_{t,b,s} + \frac{dc}{\gamma} u_{t,s}^\dagger \end{aligned} \quad (52)$$

where $R_{t,s} = -\left(\frac{dc}{\gamma}\right) \sum_{b=1}^B y_{t,b,s}$

In this experiment, We assume that the physical core \mathbf{F} represents the equations that govern the slow variables and we use a sub-model \mathbf{M}_θ to mimic the impact of the fast variables $y_{t,b,s}$:

$$\dot{\mathbf{u}}_t = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_\theta(\mathbf{u}_t) \quad (53)$$

where $\mathbf{u}_t = [u_{t,1}, u_{t,1}, \dots, u_{t,S}]^T \in \mathbb{R}^S$ and \mathbf{M}_θ is a fully connected neural network with parameters θ . The physical core $\mathbf{F} = [F_1, F_2, \dots, F_S]^T$ is given by:

$$\dot{u}_{t,s} = F_s = -u_{t,s-1}^\dagger (u_{t,s-2}^\dagger - u_{t,s+1}^\dagger) - u_{t,s}^\dagger + A \quad (54)$$

The goal of this experiment is to evaluate the proposed Euler approximation of the online learning of the sub-model \mathbf{M}_θ (based on a static approximation of the Jacobian of the flow) on a multiscale dynamical systems, for varying time steps of the Euler approximation. We set the number of slow variables $S = 8$ and the number of fast variables $B = 5$. Regarding the values of the parameters of the equation, we use the following configuration: $A = 8$, $d = 1$, $\gamma = 10$, and $c = 10$. This simulation configuration yields chaotic dynamics, where the statistical properties can not be solely explained by the slow model (54) (this can be visualized in Figure 7, where the PDF of the physical core is given with respect to the one of the multiscale system). We assume here that the number of time steps n is fixed, and we run a series of two experiments for which the time step of the Euler approximation of the gradients decreases from $h = 0.1$ to $h = 0.05$.

We compare the proposed Euler approximation to both offline and online calibration techniques. The online calibration is carried using an exact gradient and also with an emulator. The emulator is trained sequentially to the sub-model \mathbf{M}_θ

as discussed in section (2.4). In the online learning experiments, the training datasets correspond to time series of the full Lorenz 96 system $\{(u_{t_k+jh_i}^\dagger, u_{t_k}^\dagger)\}$ with $k = 1 \dots N$ and $j = 1 \dots n$ and in the offline learning experiment, the training data corresponds to $\{(u_{t_k}^\dagger, \mathbf{R}_{t_k} = [R_{t_k,1}, R_{t_k,2}, \dots, R_{t_k,S}]^T)\}$ with $k = 1 \dots N$. The number of simulation steps is set to $n = 10$ and the size of the dataset N is equal to 20000.

Overall, the following models are tested:

- **Online calibration with exact gradient:** In this calibration scheme, we implement the solver of (53) in a differentiable language and we optimize the parameters of the sub-model with the exact gradient of the online objective function.
- **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 the Static-EGA (21). In this experiment, the solver of (53) is not assumed to be differentiable.
- **Online calibration with an emulator:** We also evaluate and compare the proposed approximation schemes to online learning with emulators as presented in section 2.4. We recall that in this experiment, physical core \mathbf{F} in (53) is replaced (in the training phase) by a neural network that is trained sequentially with \mathbf{M}_θ . This network is a linear quadratic model, similar to the one discussed in Fablet et al. [2018].
- **Offline calibration:** We also compare the proposed static approximation to a simple offline learning strategy. In this experiment, the parameters of the sub-model are optimized to minimize the following offline objective function:

$$\mathcal{Q} = \frac{1}{N} \sum_k \left\| \mathbf{R}_{t_k} - \mathbf{M}_\theta(u_{t_k}^\dagger) \right\|^2 \quad (55)$$

Besides the offline calibration scheme, the online cost function used in this experiment is the mean squared error of the numerical integration of (53) with respect to the true two-scale Lorenz 96 sequence. All the tested experiments share the same parameterization of the sub-model \mathbf{M}_θ , which is a fully connected neural network with 6 hidden layers, each with 100 neurons and a hyperbolic tangent activation.

I.1.1 Performance of the learnt sub-model

We plot the performance of both online and offline optimization approaches in the Lorenz 96 case study in Figure 7. We evaluate the approaches with respect to the true Lorenz 96 simulation and also with respect to a simulation issued from the physical core. Overall, we notice that both offline and online schemes noticeably outperform the physical core which highlights the relevance of using such corrections. Regarding the short-term prediction performance, panel (c) of Figure 7 shows that all models are able to provide better predictions than the physical core. In this experiment, the online learning with true gradient is able to provide the best prediction performance. When evaluating the qualitative properties of the simulation of the models highlighted for instance in panels (a) and (b) in Figure 7, We also found that the proposed approximate gradient for the online learning scheme provides a very nice correction that is on the same level as online optimization with the true gradient. The proposed approximation also improves when we reduce the time step of the Euler approximation of the gradient from $h = 0.1$ to $h = 0.05$. This experiment also reveals that online learning with an emulator can be challenging. Specifically, the qualitative and quantitative comparison of the PDF of the models trained online with an emulator in Figure 7 shows that the sub-model calibration can be highly sensitive and shows that two different model initializations can lead to very distinct sub-models.

J Proof of the equation (39)

The proof of (39) is conducted by recurrence. For $n = 1$ we have:

$$\frac{\partial}{\partial \theta} \Psi^1(\mathbf{u}_t) = \frac{\partial \Psi(\Psi^0(\mathbf{u}_t))}{\partial \theta} \quad (56)$$

which is by inspection of (38) true.

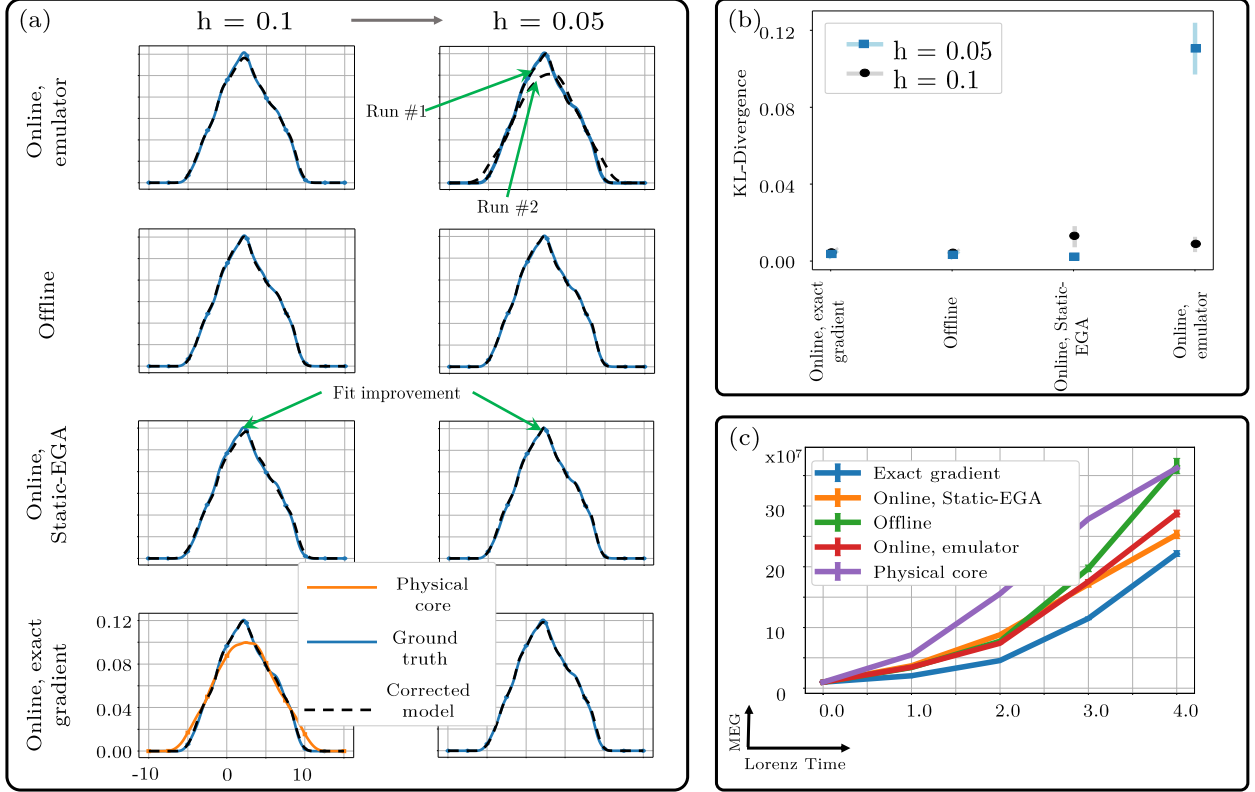


Figure 7: *Simulation example of the tested models in the Lorenz 96 experiment.* We compare a multiscale simulation of the true system (52) to the physical core in (32) (without the neural network sub-model) and to corrected models where the correction term M_θ is calibrated both online and offline. We compare the PDF of the first state of the tested models with respect to the one computed from a simulation of the true system given by (52) both qualitatively in (a) and quantitatively in (b). (c) Mean error growth of the tested models. We plot both the mean and standard deviation that was computed based on an ensemble of 20 trajectories issued from 5 different learning realizations. The error bars are scaled by 1/20.

Assume that (39) is true for all n , for $n + 1$ we have:

$$\begin{aligned}
 \frac{\partial}{\partial \theta} \Psi^{n+1}(\mathbf{u}_t) &= \sum_{j=1}^{j=n} \left(\prod_{i=1}^{i=n+1-j} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \right) \frac{\partial}{\partial \theta} \Psi(\Psi^{j-1}(\mathbf{u}_t)) + \frac{\partial}{\partial \theta} \Psi(\Psi^n(\mathbf{u}_t)) \\
 &= \prod_{i=1}^{i=n} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^0(\mathbf{u}_t)) \\
 &\quad + \prod_{i=1}^{i=n-1} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^1(\mathbf{u}_t)) \\
 &\quad \vdots \\
 &\quad + \prod_{i=1}^{i=2} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^{n-2}(\mathbf{u}_t)) \\
 &\quad + \prod_{i=1}^{i=1} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^{n-1}(\mathbf{u}_t)) \\
 &\quad + \frac{\partial}{\partial \theta} \Psi(\Psi^n(\mathbf{u}_t))
 \end{aligned} \tag{57}$$

if we take $\frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)}$ as a common factor we have:

$$\begin{aligned}
 \frac{\partial}{\partial \theta} \Psi^{n+1}(\mathbf{u}_t) &= \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \prod_{i=2}^{i=n} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^0(\mathbf{u}_t)) \\
 &+ \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \prod_{i=2}^{i=n-1} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^1(\mathbf{u}_t)) \\
 &\vdots \\
 &+ \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \prod_{i=2}^{i=2} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^{n-2}(\mathbf{u}_t)) \\
 &+ \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^{n-1}(\mathbf{u}_t)) \\
 &+ \frac{\partial}{\partial \theta} \Psi(\Psi^n(\mathbf{u}_t))
 \end{aligned} \tag{58}$$

Factorization of $\frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)}$:

$$\begin{aligned}
 \frac{\partial}{\partial \theta} \Psi^{n+1}(\mathbf{u}_t) &= \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \left(\prod_{i=2}^{i=n} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^0(\mathbf{u}_t)) \right. \\
 &+ \prod_{i=2}^{i=n-1} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^1(\mathbf{u}_t)) \\
 &\vdots \\
 &+ \prod_{i=2}^{i=2} \frac{\partial \Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial \Psi^{n+1-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^{n-2}(\mathbf{u}_t)) \\
 &+ \frac{\partial}{\partial \theta} \Psi(\Psi^{n-1}(\mathbf{u}_t)) \\
 &+ \left. \frac{\partial}{\partial \theta} \Psi(\Psi^n(\mathbf{u}_t)) \right)
 \end{aligned} \tag{59}$$

Which is conveniently written as:

$$\begin{aligned}
 \frac{\partial}{\partial \theta} \Psi^{n+1}(\mathbf{u}_t) &= \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \left(\prod_{i=1}^{i=n-1} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^0(\mathbf{u}_t)) \right. \\
 &+ \prod_{i=1}^{i=n-2} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^1(\mathbf{u}_t)) \\
 &\vdots \\
 &+ \prod_{i=1}^{i=1} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \frac{\partial}{\partial \theta} \Psi(\Psi^{n-2}(\mathbf{u}_t)) \\
 &+ \frac{\partial}{\partial \theta} \Psi(\Psi^{n-1}(\mathbf{u}_t)) \\
 &+ \left. \frac{\partial}{\partial \theta} \Psi(\Psi^n(\mathbf{u}_t)) \right)
 \end{aligned} \tag{60}$$

Equation (60) can be written in a more compact form as:

$$\begin{aligned}
 \frac{\partial}{\partial \boldsymbol{\theta}} \Psi^{n+1}(\mathbf{u}_t) &= \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \underbrace{\left(\sum_{j=1}^{j=n-1} \left(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \right) \frac{\partial}{\partial \boldsymbol{\theta}} \Psi(\Psi^{j-1}(\mathbf{u}_t)) \right)}_{= \frac{\partial}{\partial \boldsymbol{\theta}} \Psi^n(\mathbf{u}_t)} + \frac{\partial}{\partial \boldsymbol{\theta}} \Psi(\Psi^{n-1}(\mathbf{u}_t)) \\
 &+ \frac{\partial}{\partial \boldsymbol{\theta}} \Psi(\Psi^n(\mathbf{u}_t)) \\
 &= \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \frac{\partial \Psi^n(\mathbf{u}_t)}{\partial \boldsymbol{\theta}} + \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \boldsymbol{\theta}}
 \end{aligned} \tag{61}$$

which is by inspection of (38) true. This completes the proof of the formula (39).

K Data and code availability

The code developed for this work is written in Python using Pytorch and is available at <https://github.com/CIA-Oceanix/EGA>.