Learning sub-grid dynamics in idealized turbulent systems

Hugo Frezat

Université Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Mécanique des Fluides**

Arrêté ministériel : 25 mai 2016

Présentée par

Hugo Frezat

Thèse dirigée par **Guillaume Balarac,** Professeur des Universités, Grenoble INP, Université Grenoble Alpes
codirigée par **Julien Le Sommer,** Directeur de recherche, CNRS Délégation Alpes
et **Ronan Fablet,** Professeur des Universités, IMT Atlantique

préparée au sein du **Laboratoire des Ecoulements Géophysiques et Industriels**
dans **l'École Doctorale IMEP²**

Learning sub-grid dynamics in idealized turbulent systems

Apprentissage de la dynamique sous-maille dans des systèmes turbulents idéalisés

Thèse soutenue publiquement le **9 décembre 2022**,
devant le jury composé de :

Monsieur Thomas DUBOS
Professeur, Ecole Polytechnique, Rapporteur
Monsieur Nicolas THOME
Professeur, Sorbonne Université, Rapporteur
Monsieur Eric BLAYO
Professeur, Université Grenoble-Alpes, Président
Madame Laure ZANNA
Professeur, Courant Institute, NYU, Examinateur
Monsieur Etienne MEMIN
Directeur de Recherche, INRIA Rennes, Examinateur

Abstract

Climate predictions and weather forecasting strongly rely on simulations of the Earth's oceans and atmosphere turbulent dynamics. But the simulation of turbulent processes is so computationally expansive that it is only possible to resolve the largest physical scales. The representation of unresolved scales in these simulations is therefore a key source of uncertainty and its modeling is still an open problem. Recently, machine learning techniques have been receiving growing attention for the design of parametrizations and subgrid-scale models. In this thesis, we explore the impact of explicitly embedding law invariances in neural networks trained to represent the small-scale dynamics of a scalar quantity advected by a turbulent flow. We also propose a new training algorithm inspired by the end-to-end approach applied to turbulence modeling, where the loss can be optimized on so-called "a posteriori" metrics. While the strategy gives promising results, it requires a differentiable numerical solver during the learning phase. We try to address this limitation with an additional step during which we train a differentiable emulator of the resolved dynamics. The error patterns in the emulator are shown to be propagated as a correction bias in the subgrid-scale model, limiting its performance. However, regularizing the model loss enable stable simulations and brings "a posteriori" learning benefits in non-differentiable solvers. Our results show that neural networks can respect physical principles and outperform classical models in long-term stable simulations. Their implementation in realistic solvers is expected to improve climate understanding and turbulence in general.

Keywords : subgrid modeling, machine learning, turbulence

Résumé

Les prédictions climatiques et prévisions météorologiques reposent fortement sur des simulations de la dynamique turbulente des oceans et de l'atmosphere Terrestre. Mais la simulation de ces processus turbulents est si coûteuse en calcul qu'il n'est possible que de résoudre les plus grandes échelles physiques. La représentation des échelles non-résolues dans ces simulations est donc une source d'incertitude majeure et sa modélisation est encore un problème ouvert. Récemment, les techniques d'apprentissage automatique ont reçu une attention grandissante pour la construction de paramétrisations et modèles sous-maille. Dans cette thèse, nous explorons l'impact de l'incorporation explicite de lois d'invariances dans des réseaux de neurones entrainés pour représenter la dynamique à petite échelle d'une quantité scalaire advectée par un écoulement turbulent. Nous proposons aussi un nouvel algorithme d'apprentissage inspiré par l'approche bout-à-bout appliqué à la modélisation de la turbulence, où la fonction de coût peut être optimisée sur des métriques dites "a posteriori". Si cette stratégie donne des résultats prometteurs, elle requiert néanmoins un code numérique différentiable pendant la phase d'apprentissage. Nous essayons d'adresser cette limitation avec une étape additionnelle durant laquelle nous entrainons un émulateur différentiable de la dynamique résolue. Les motifs d'erreur générés par l'émulateur se révèlent être propagés en tant que bias correctif dans le modèle sous-maille, ce qui limite sa performance. Cependant, régulariser la fonction de coût du modèle permet d'effectuer des simulations stables et apporte les bénéfices de l'apprentissage "a posteriori" dans des codes non-differentiables. Nos résultats montrent que les réseaux de neurones peuvent respecter des principes physiques et surpasser les modèles classiques dans des simulations stables de longue durée. Leur implémentation dans des codes réalistes devrait améliorer notre compréhension du climat et de la turbulence en général.

Mots-clés : modélisation sous-maille, apprentissage automatique, turbulence

Acknowledgements

After three years of passionating research, I was privileged to have Laure Zanna, Etienne Memin, and Eric Blayo participating in my jury as well as Thomas Dubos and Nicolas Thome reviewing my manuscript. I am very grateful for the comments and discussions that happened during the defense.

Coming from three different fields, working with a turbulence physicist, an oceanographer, and an applied mathematician might appear complicated, and indeed, Guillaume Balarac, Julien Le Sommer, and Ronan Fablet were always ready for debates about how the same thing should be called during meetings.

I have always been really looking forward to each discussion and new ideas that came out of it would always spark something in me that I had to try right away. I'm sure that future collaborations will be possible and that there are many great research topics to investigate, at the edge of multidisciplinarity.

I learned a lot from the one-week trip to the US where we visited Columbia, Princeton, and NYU with so many collaborators that I could not list them here. The recent effort leading progress for climate science between applied mathematicians and physicists is truly inspiring.

Obviously, I'd like to mention my colleagues from the MOST team with whom I had many discussions about mountain sports during coffee breaks. Training models and crunching numbers would not have been possible without Patrick Begou who provided top-tier GPUs, all for myself.

Finally, the closest support came from my family; mom and grand-parents always curious to understand the essence of what my work is, and of course, Elora, who shares this journey with me.

Contents

Chapter 1

Introduction

The study of the climate system is an active field of research, especially in the context of the ongoing anthropogenic impacts and the potential to cause substantial climate changes during the coming decades and centuries (Houghton et al., 2001). Modern computational approaches to understanding this complex system are based on a combination of models and observations, both in-situ and satellite (see Fig. 1.1 for example). In practice, observational data is used to correct the error made by numerical models, primarily driven by the dynamics of the ocean, atmosphere, and their coupled interaction (Bauer et al., 2015; Kharin and Zwiers, 2000). These corrections (Laloyaux et al., 2016) can be partially regarded as taking into account factors that would not be captured by a differential equation. Moreover, numerical simulations used for climate projections are similar to weather forecast models, except that they are typically run on much larger spatiotemporal scales. These global models are thus extremely expansive from a computational point-of-view and simplifications leading to larger sources of error are mandatory (Stevens and Bony, 2013). Indeed, simulations at full resolution are not tractable with the current computer resources yet and are likely to remain the case for the foreseeable future (Raäisaänen, 2007; Schneider et al., 2017b; Wiens et al., 2009). Numerically, the spatial domain is discretized onto a grid, for which local information is only known either at the corners or centers of the grid cell. The required grid cell resolution should typically be in the order of the smallest dynamical structure, which goes all the way down to the dissipative scale in turbulent flows. Global climate models are still far from reaching this resolution since they are only able to resolve horizontal grid spacing ranging from ∼100–300 km (Eyring et al., 2016) with ∼10–50 vertical levels from top to bottom of the ocean and atmosphere (Stewart et al., 2017). The dynamics happening in-between these grid cells, also called *subgrid-scale* is not resolved but remains an essential component of these models (Schneider et al., 2017a). Modeling the small-scale dynamics has also been important in different applications, for example, to improve renewable energies or to study astrophysical flows.

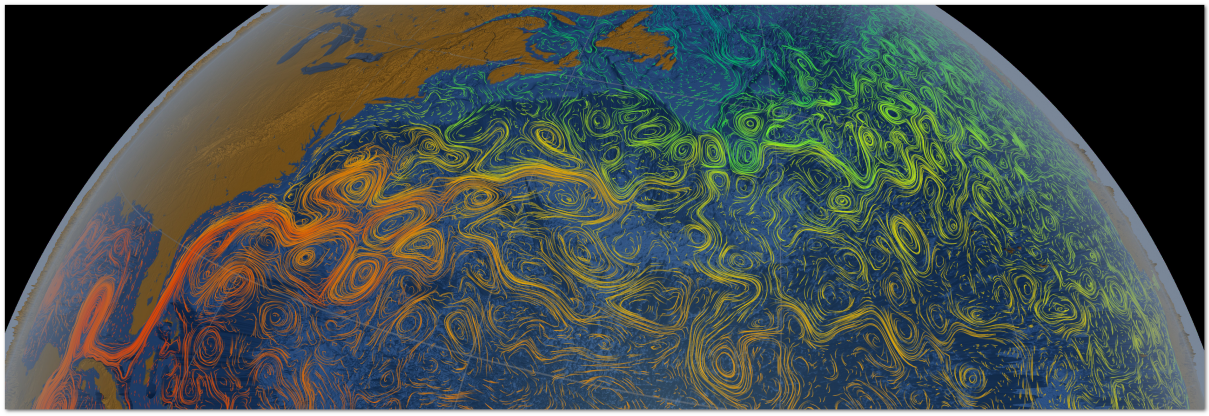

Figure 1.1: Ocean flows colored with sea surface temperature in the Gulf stream stretching from the Gulf of Mexico towards Western Europe. The data was produced by the ECCO2 joint project at MIT/JPL and uses MITgcm with satellite and in-situ data of the global ocean and sea-ice. *Credits: NASA/Goddard Space Flight Center Scientific Visualization Studio.*

Models of unresolved turbulent motions are historically based on first principles, physics, idealized experiments, and high-resolution observations. Their design involves a mixture of empirical and process-based modeling. Process-based models are formulated, tested, and calibrated with experiments performed in the field, in the laboratory, or with computers (Stensrud, 2009). On this basis, the actual model estimates a tendency term for the target subgrid term from its resolved variables. The underlying conceptual framework can rely on some ensemble averaging procedure, so that the model intends to capture the statistical effect of unresolved processes, for instance for turbulence models (Mellor, 1985). Alternatively, one may consider a spatial filtering procedure from which the model can exploit the scale-invariant properties – or hypotheses – of turbulence, as in Large Eddy Simulation (LES) (Lesieur et al., 2005). The advantage of this approach is that only large-scale processes are resolved, which relaxes the grid resolution restrictions. However, small-scale dynamics remain to be modeled from resolved quantities. This has been an open challenge for a long time, as it mathematically translates to a high dimensional non-linear inverse problem (Piomelli, 2014). Typical applications are found with the incompressible Navier-Stokes equations which give an accurate representation of the fluid motions that drive the oceans and the atmosphere's dynamics (Fox-Kemper and Menemenlis, 2008; Mason, 1989). In practice, the turbulent dynamics of these geophysical processes is also driven by multiple factors, including thermodynamics, magnetism, or chemical reactions. This makes the design of subgrid models highly convoluted with complex interactions that can not always be analyzed individually. While process isolation eliminates some difficulties, there are still numerical errors related to the spatial discretization of the governing equations (Fornberg, 1990).

In this thesis, we will be first interested in the subgrid process responsible for the mixing of a passive scalar in a turbulent flow (e.g. a contaminant), and the evolution of quasi two-dimensional geophysical fluid motions, separately. To further focus on the *subgrid* modeling problem, we use pseudo-spectral methods, which reduce the numerical error at the expanse of a limited geometry such as symmetric domains with periodic boundaries. Studying turbulent flows in these *idealized* configurations is primordial to understanding subgrid processes without interacting with other sources of error. These "theoretical" applications are expected to provide a basis for complex processes and geometries.

Recently, the use of machine learning (ML) to improve subgrid models has gained momentum. Calibrating subgrid models against observations with ML and emulators is for instance becoming common practice (Ollinaho et al., 2013). ML also provides new means to design subgrid models from high-resolution simulations that resolve all the physical scales in a restricted, smaller configuration (Gamahara and Hattori, 2017; Maulik et al., 2019). In this context, ML may learn a mapping that predicts the tendency term due to unresolved subgrid effects from resolved quantities available in a target model. These methodologies have been applied to many types of flows, with applications to ocean (Bolton and Zanna, 2019) and atmosphere (Rasp et al., 2018) dynamics. However, it is important to remember that ML models are purely mathematical tools. Besides their ability to universally represent non-linear functions, they are not constrained to follow physical laws or invariances by design. Even if their performance on the specific task they were training for is remarkable, they often lack generalization and break whenever they are used in a different context. These models are thus difficult to interpret, which limits their widespread among physicists. It has become quite clear (Battaglia et al., 2018) that these models are powerful and could benefit from domain-specific knowledge to improve both their interpretability and generalization performance. These recent ideas build on the rise of scientific machine learning (SciML) (Innes et al., 2019) and its broad application to physical sciences. SciML is an emerging field, which bridges scientific computing and machine learning. Some recent key developments in the field have been motivated both *from* physical insights and *for* their applications to physical sciences, especially in fluid dynamics (Carleo et al., 2019). The conceptual research in ML motivated by applications to problems governed by partial differential equations (Long et al., 2018; Raissi et al., 2019; Sirignano and Spiliopoulos, 2018) have for instance gradually freed ML from its black-box reputation. It is also now possible to directly benefit from ML for system identification and equation discovery (Brunton et al., 2016). At this state, first ideas are emerging with applications in simple benchmark systems such as the Lorenz (Dueben and Bauer, 2018). Moving to more complex dynamics in two and three dimensions is however the next step towards the acceptability and usability of a new generation of ML-based subgrid models in climate models.

1.1 Thesis aims and structure

We are still a long way to solving the full range of dynamics in ocean and atmosphere models. While recent efforts have made a lot of progress to the subgrid modeling problem, in particular, using data-driven techniques, challenges remain. We will see in the next chapter that existing data-driven models are not particularly using the previously established understanding of the underlying dynamics. In order to see practical use in global climate models (GCMs), these models need to get away from their "black-box" reputation, gain some interpretability and significantly improve from existing alternatives.

From these observations, some questions naturally arise concerning the future of subgrid models using data-driven methods:

1. How to embed well-known analytical identities into modern machine learning models to build an understood basis for subgrid modeling?
2. Can we formulate a learning strategy able to capture complex geophysical phenomena with long-term stability?
3. Is it possible to train precise subgrid models from any numerical solver, even in the presence of errors?

Thesis structure. The necessary background about the equations of motion that govern fluid dynamics and the subgrid problem will be presented in Chapter 2 with an overview of the existing literature. We also include an introduction to the machine learning models used in this thesis, along with the numerical methods that solve the governing equations.

Investigating the above questions starts with Chapter 3, where we describe different approaches to include law invariances in neural network architectures. Their application to the subgrid modeling of advection-diffusion (transport) equations shows performance and generability improvements.

Realizing the limitations of the previous chapter to properties based on group actions such as invariances, we question the "static" nature of the training process used for current data-driven models in Chapter 4 and propose a differentiable strategy to improve the long-term stability in two-dimensional flows.

Building upon the results of the previous chapter, we explore in Chapter 5 a solution to overcome the differentiability requirement using a temporary data-driven emulation of the numerical solver.

This thesis ends with Chapter 6, where we summarize the results and discuss their implications, together with future challenges that are yet to tackle.

Parts of the results have been published in the following journals:
Frezat et al. (2021). in *Physical Review Fluids*
Frezat et al. (2022). in *Journal of Advances in Modeling Earth Systems (JAMES)*

Chapter 2

Background

This chapter describes the necessary background used in the following content chapters. First, we will introduce the partial differential equations (PDEs) governing the flow dynamics and the goal of this thesis, i.e. the subgrid modeling problem in a discrete numerical perspective. Related to this, we will discuss some performance characteristics as well as classical baseline subgrid-scale (SGS) models with their limitations. We will give the basis of the numerical methods that are used throughout this thesis to solve the PDEs introduced previously, with their advantages and drawbacks. Finally, the concept of data-driven methods is outlined, and recent machine learning (ML) models are reviewed, with a focus on state-of-the-art neural networks (NN) that will be used as the computational basis of this thesis.

Here are the major references for each subsection :

1. *Pope (2000)*
2. *Canuto et al. (2012)*
3. *Goodfellow et al. (2016)*

Contents

2.1 Turbulent flows phenomenology

The motion of continuous fluids encountered in oceans and atmospheres is governed by the Navier-Stokes (NS) equations,

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u}, \qquad \nabla \cdot \mathbf{u} = 0 \tag{2.1}$$

where $\mathbf{u}$ is the velocity vector, p the pressure and ν the kinematic viscosity. In this study, we will only consider "incompressible" flows, where density ρ is constant, i.e. $\partial\rho/\partial t = 0$. When studying flows governed by the NS equations, it has been long shown important to adimensionalize the system in order to perform reproducible experiments. In a coordinate system $\mathbf{x}$ at rest in an inertial frame, we can use characteristic length scale L and velocity U to define,

$$\mathbf{x}' = \frac{\mathbf{x}}{L}, \qquad U' = \frac{\mathbf{u}}{U}, \qquad t' = \frac{tU}{L}, \qquad P' = \frac{p}{\rho \mathbf{u}^2}. \tag{2.2}$$

From these quantities, the non-dimensional form of the NS and continuity equations (2.1) can be obtained,

$$\frac{\partial U_i'}{\partial t'} + U_j' \frac{\partial U_i'}{\partial x_j'} = -\frac{\partial P'}{\partial x_i'} + \frac{1}{\mathrm{Re}} \frac{\partial^2 U_i'}{\partial x_j'^2} \tag{2.3}$$

$$\frac{\partial U_i'}{\partial x_i'} = 0 \tag{2.4}$$

with explicit partial derivatives w.r.t. spatial coordinates for simplicity. Now, the equation is only parametrized by the Reynolds number Re and we can identify two types of flows depending on their value. When Re is small, the non-linearities of the inertial effects can be neglected and the flow is often said to be laminar. When Re is large, the flow is highly affected by complex non-linear interactions, which creates a large range of structures and chaotic motion, i.e. the flow is then said to be turbulent. We have,

$$\mathrm{Re} = \frac{UL}{\nu}. \tag{2.5}$$

2.1.1 Turbulent scales

In this thesis, we will only be interested in homogeneous turbulence with high Reynold numbers, such that Kolmogorov (1941)'s theory applies. In order to understand the complexity of numerically solving the NS equations, we briefly introduce the turbulent scales based on the Richardson (1922)'s energy cascade. First, let's consider turbulence to be composed of *eddies* of different sizes ℓ described by their characteristic,

$$\underbrace{\tau(\ell)}_{\text{timescale}} = \ell / \underbrace{u(\ell)}_{\text{velocity}}. \tag{2.6}$$

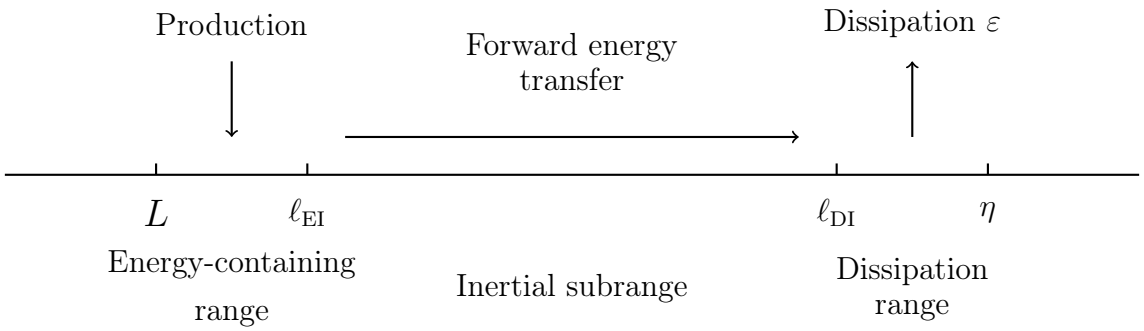

Figure 2.1: Sketch of the energy cascade at high Reynolds number with the various lengthscales and ranges for an eddy size ℓ.

We can by extension define the largest eddies by their lengthscale ℓ_0 which is comparable to the flow scale L, and their characteristic velocity $u_0 = u(\ell_0)$, again comparable to the flow velocity U. In the energy cascade, large eddies tend to transfer their energy to smaller eddies, which again undergo a similar break-up process transferring their energy down to the smallest eddies. This process continues until $\mathrm{Re}(\ell) = u(\ell)\ell/\nu$ is sufficiently small and molecular viscosity is effective in dissipating the kinetic energy. Since these eddies have energy of order u_0^2 and timescale $\tau_0 = \ell_0/u_0$, their dissipation rate is supposed to scale as $\varepsilon \sim u_0^2/\tau_0 = u_0^3/\ell_0$. Now, important hypotheses drawn by Kolmogorov (1962) states that turbulent flows at sufficiently high Reynolds number have universal small-scale statistical motions that are uniquely determined by ν and ε. It follows the Kolmogorov's unique length, velocity, and time scales,

$$\eta = \left(\nu^3/\varepsilon\right)^{1/4}, \quad u_\eta = (\varepsilon\nu)^{1/4}, \quad \tau_\eta = (\nu/\varepsilon)^{1/2}. \tag{2.7}$$

One can identify the *universal equilibrium range* $\ell < \ell_{\mathrm{EI}} \sim \frac{1}{6}\ell_0$ as being splitted into two subranges: the *inertial subrange* ($\ell_{\mathrm{EI}} > \ell > \ell_{\mathrm{DI}}$) and the *dissipation range* ($\ell < \ell_{\mathrm{DI}} \sim 60\eta$). The various lengthscales and ranges are sketched in Fig. 2.1. From the scaling $\varepsilon \sim u_0^3/\ell_0$, we can determine the ratios between the largest and smallest scales in a flow at a certain Re, we have,

$$\eta/\ell_0 \sim \mathrm{Re}^{-3/4}, \tag{2.8}$$

$$u_\eta/u_0 \sim \mathrm{Re}^{-1/4}, \tag{2.9}$$

$$\tau_\eta/\tau_0 \sim \mathrm{Re}^{-1/2}. \tag{2.10}$$

When solving the NS equations, direct numerical simulation (DNS) must resolve the entire range of scales present in the flow, which goes from ℓ_0 to η. However, geophysical flows span very large Reynolds numbers (e.g., $\sim 10^{11}$ in the Gulf Stream, $\sim 10^{12}$ for a tropical cyclone) which lead to a tiny ratio η/ℓ_0. The numerical consequence is the requirement of a fine grid such that the spacing Δ between two grid points is in the order of the smallest eddies η.

2.1.2 Reduced-order modeling via filtering

In practice, DNS is not possible with the available computational resources. We know however that most of the energy and anisotropy are contained in the larger scales of motion, but nearly all of the computational effort in DNS is spent on the smallest and dissipative motions. It is also accepted that eddies in their expected universal equilibrium range can be statistically represented by a model. In a large-eddy simulation (LES), the large-scale dynamics is explicitly resolved, while the influence of the smaller scales is modeled. From an analytic modeling point-of-view, a *filtering* operation is defined to perform this scale separation, i.e. to decompose a vector field $\mathbf{y}(\mathbf{x}, t)$ into the sum of a filtered (representing the large scales) component $\bar{\mathbf{y}}(\mathbf{x}, t)$ and a residual (for the small scales) $\mathbf{y}'(\mathbf{x}, t)$, such that $\mathbf{y}(\mathbf{x}, t) = \bar{\mathbf{y}}(\mathbf{x}, t) + \mathbf{y}'(\mathbf{x}, t)$. The numerical interest of the method is that the filtered term $\bar{\mathbf{y}}(\mathbf{x}, t)$ can be resolved on a coarser grid, compared to the initial, high-resolution field $\mathbf{y}(\mathbf{x}, t)$. On a coarser grid, the residual term $\mathbf{y}'(\mathbf{x}, t)$ is now referred to as the "subgrid"-scale (SGS) term. The gain in computational complexity comes both from the smaller requirements on the grid and temporal spacing, i.e.,

$$\bar{\Delta} \gg \Delta \tag{2.11}$$

$$\overline{\Delta t} \gg \Delta t. \tag{2.12}$$

These quantities are proportional to the filter width (or ratio) $\Delta' = \bar{\Delta}/\Delta$ which would, ideally, be located close to the smallest energy-containing motions ℓ_{EI}. The (low-pass) filtering operation (Leonard, 1975) over the entire domain is defined by,

$$\bar{\mathbf{y}}(\mathbf{x}, t) = \int G(\mathbf{r}, \mathbf{x}) \mathbf{y}(\mathbf{x} - \mathbf{r}, t) \mathrm{d}\,\mathbf{r} \tag{2.13}$$

where filter kernel G should satisfy some properties to be applicable to partial differential equations (PDEs),

$$\overline{\frac{\partial \mathbf{y}}{\partial t}} = \frac{\partial \bar{\mathbf{y}}}{\partial t}, \qquad \text{commutes with partial derivatives,} \tag{2.14}$$

$$\overline{c(\mathbf{y}_1 + \mathbf{y}_2)} = c\bar{\mathbf{y}}_1 + c\bar{\mathbf{y}}_2, \qquad \text{linearity,} \tag{2.15}$$

$$\bar{c} = c, \qquad \text{conservation of constants.} \tag{2.16}$$

To better understand the grid coarsening operation, we have studied the filtered vector fields in wavenumber space. Let us denote the Fourier transform of $\mathbf{y}(\mathbf{x})$ by,

$$\hat{\mathbf{y}}(k) = \mathcal{F}\{\mathbf{y}(\mathbf{x})\}. \tag{2.17}$$

We can expand the filtered field and kernel in wavenumber space, using $\hat{G}(k) = 2\pi\mathcal{F}\{G(\mathbf{x})\}$ and the convolution theorem,

$$\begin{aligned} \hat{\bar{\mathbf{y}}}(k) &= \mathcal{F}\{\bar{\mathbf{y}}(\mathbf{x})\} \\ &= 2\pi\mathcal{F}\{G(\mathbf{x})\}\mathcal{F}\{\mathbf{y}(\mathbf{x})\} \\ &= \hat{G}(k)\hat{\mathbf{y}}(k). \end{aligned} \tag{2.18}$$

In this thesis, we consider a statistically homogeneous, periodic field $\mathbf{y}(\mathbf{x})$ in the interval $0 \leq \mathbf{x} \leq L$ completely resolved on N nodes of a uniform grid of spacing $\Delta = L/N$. In this context, $\mathbf{y}(\mathbf{x})$ can be written in the form of an inverse Fourier series,

$$\mathbf{y}(\mathbf{x}) = \sum_{n=1-\frac{1}{2}N}^{\frac{1}{2}N} \hat{\mathbf{y}}(k_n) e^{ik_n \mathbf{x}} \tag{2.19}$$

where k_n is the nth wavenumber obtained from the exponential notation in the real transform,

$$k_n = \frac{2\pi n}{L} \tag{2.20}$$

It is possible to provide a relationship between the grid spacing and the number of required wavenumbers in spectral space,

$$\Delta = \frac{\pi}{k_{\max}} = \frac{\pi}{k_{N/2}}. \tag{2.21}$$

Now, the coarse grid resolution can be determined by choosing a cutoff wavenumber $k_c < k_{\max}$, so that the number of nodes on the coarse grid is,

$$\bar{N} = \bar{\Delta} L = \frac{\pi}{k_c} L. \tag{2.22}$$

The filtered field can finally be reconstructed as an inverse Fourier series, we obtain values on the coarse grid,

$$\bar{\mathbf{y}}(\mathbf{x}) = \sum_{n=1-\frac{1}{2}\bar{N}}^{\frac{1}{2}\bar{N}} \hat{G}(k_n) \hat{\mathbf{y}}(k_n) e^{ik_n \mathbf{x}}. \tag{2.23}$$

Cutoff kernel. Note that when G is a cutoff kernel, the operation of grid coarsening is the same as the filtering operation. Indeed, the cutoff kernel is defined in spectral space as $\hat{G}(k) = H(k_c - |k|)$, i.e.

$$\hat{G}(k)\hat{\mathbf{y}}(k) = \begin{cases} \hat{\mathbf{y}}(k), & \text{for } |k| < k_c \\ \\ 0, & \text{for } |k| \geq k_c. \end{cases} \tag{2.24}$$

In this case, coarsening (2.23) can be applied directly to the original field without loss of information,

$$\bar{\mathbf{y}}(\mathbf{x}) = \sum_{n=1-\frac{1}{2}\bar{N}}^{\frac{1}{2}\bar{N}} \hat{\mathbf{y}}(k_n) e^{ik_n \mathbf{x}}. \tag{2.25}$$

The impact of the kernel choice on the subgrid (residual) term will be briefly mentioned in the following chapters.

2.1.3 Instantaneous and statistical characterization

While the filtered fields can be resolved on a coarse grid, the remaining objective of this reduced-modeling approach is thus to model the SGS component $\mathbf{y}'(\mathbf{x})$. Going back to the NS (2.4), applying the filtering operation gives,

$$\frac{\partial \bar{U}_i}{\partial t} + \bar{U}_j \frac{\partial \bar{U}_i}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + \nu \frac{\partial^2 \bar{U}_i}{\partial x_j^2} + \frac{\partial}{\partial x_j}\left(\bar{U}_i \bar{U}_j - \overline{U_i U_j}\right) + F_{\bar{U}} \tag{2.26}$$

$$\frac{\partial \bar{U}_i}{\partial \bar{x}_i} = 0. \tag{2.27}$$

Here, the residual term to be modeled (or equivalently the difference with the non-filtered NS equation) appears as the divergence *residual stress tensor* τ_{ij},

$$\left(\frac{\partial U_i}{\partial t}\right)' = \frac{\partial}{\partial x_j}\left(\bar{U}_i \bar{U}_j - \overline{U_i U_j}\right) = \frac{\partial}{\partial x_j}\tau_{ij}. \tag{2.28}$$

Note that the same methodology can be applied to other differential equations, as we will see with the barotropic quasi-geostrophic equation and the advection-diffusion of a scalar field. The equations for $\bar{U}$ can thus be solved if a model for the residual term is provided. In practice, most of the physical models are producing an approximation of the *residual stress tensor*, since its divergence can be accurately computed from the spatial discretization scheme, i.e.,

$$\left(\frac{\partial U_i}{\partial t}\right)' \equiv \frac{\partial}{\partial x_j} \underbrace{\mathcal{M}_{ij}}_{\approx \tau_{ij}}. \tag{2.29}$$

In the machine learning community, however, this residual model $\mathcal{M}$ often directly approximates the divergence of the *residual stress tensor* as a function of filtered velocity field $\bar{U}(\mathbf{x}, t)$, coarse grid spacing $\bar{\Delta}$ and kernel operator G,

$$\left(\frac{\partial U_i}{\partial t}\right)' \equiv \underbrace{\mathcal{M}(\bar{U}(\mathbf{x}, t), \bar{\Delta}, G)}_{\approx \frac{\partial}{\partial x_j}\tau_{ij}}. \tag{2.30}$$

The consensus in turbulence modeling is that there are two different ways to evaluate the performance of a SGS model $\mathcal{M}$:

In the ***a priori*** tests, we measure the quality of the instantaneous prediction of the SGS model when compared to τ_{ij}^{DNS}, the residual term extracted from a direct simulation.

In the ***a posteriori*** tests, the SGS model is used to perform a new simulation and the subsequent quality of the model is defined by flow (spatiotemporal) statistics.

In general, while *a priori* tests are useful to first quantify the validity of the model, it will only be useful if it performs well in *a posteriori* tests, by e.g. stability and invariants conservation.

2.1.4 Existing models

Designing subgrid models has been an important topic in turbulence modeling, and there exists a strong theoretical basis behind state-of-the-art physical models, i.e., without machine learning. Here, we only introduce well-known models that we use as baselines to benchmark the performance of proposed ML alternatives. Depending on the objective, models are commonly classified into two categories (Sagaut, 2006).

Structural modeling. The objective of structural models is to best reproduce the subgrid term, without prior knowledge of the interactions between resolved and modeled scales. A first category of developments was made from formal series expansions, trying to approximate filtered terms in the NS equations. Those models are often based on approximate deconvolution, which aims at reconstructing the unfiltered field from the filtered one,

$$U \equiv G^{-1} * \bar{U}. \tag{2.31}$$

Conversely, taking the Taylor series expansion of some filtered velocity field $\bar{U}$ at order δ, we obtain,

$$\bar{U}(\mathbf{x}) \approx \sum_{i=0}^{\delta} \frac{(-1)^i}{i!} \bar{\Delta}^i \mu_i(\mathbf{x}) \frac{\partial^i U}{\partial x^i}(\mathbf{x}) \tag{2.32}$$

where μ_i is the ith-order moment of the kernel G. Inverting (2.32) can be approximated by an iterative deconvolution (Stolz and Adams, 1999) or by a truncated Taylor expansion. We get,

$$U(\mathbf{x}) \approx \left(\sum_{i=0}^{\delta} \frac{(-1)^i}{i!} \bar{\Delta}^i \mu_i(\mathbf{x}) \frac{\partial^i}{\partial x^i}(\mathbf{x}) \right)^{-1} \bar{U}(\mathbf{x}) \tag{2.33}$$

$$= \left(1 - \frac{1}{2} \bar{\Delta}^2 \mu_2(\mathbf{x}) \frac{\partial^2}{\partial x^2} + \cdots \right) \bar{U}(\mathbf{x}). \tag{2.34}$$

Now, applying this procedure to the subgrid term τ_{ij}, we obtain the general form of the Gradient model (Carati et al., 1999; Clark et al., 1979),

$$\tau_{ij} = \bar{U}_i \bar{U}_j - \overline{U_i U_j} = \sum_{l,m=0} C_{lm}(G) \frac{\partial^l \bar{U}_i}{\partial x^l} \frac{\partial^m \bar{U}_j}{\partial x^m} \tag{2.35}$$

for which the coefficients $C_{lm}(G)$ depend explicitly on the filter. In practice, an order $\delta = 2$, i.e. $l = m = 2$ expansion is used. We also mention non-linear models (Lund and Novikov, 1993) which are based on the properties of the subgrid tensors and scale similarity models (Bardina et al., 1980) based on the hypothesis that filtered terms have the same statistical structure as those at the smallest resolved scales. ML-based models can be also classified as structural since they minimize a mismatch in the subgrid term. Overall, these structural models yield the best performance in *a priori* tests but are known to be unstable in long-term evolution due to incorrect predictions of the subgrid dissipation.

Functional modeling. The objective of functional models is to reproduce the action of the subgrid term on the resolved fields. In this regard, these models depend on the dynamics of the considered flows. Using hypotheses from energy transfer mechanisms in turbulence, it is possible to derive some understanding of the interactions between scales in the context of homogeneous isotropic turbulence, which is also the main focus of this thesis. A large category of models is dedicated to the forward energy cascade process appearing in three-dimensional NS equations as shown in Fig. 2.1. The energy transfer from resolved to subgrid scales is known to be analogous to molecular diffusion, hence a large category of models based on the concept of subgrid viscosity ν_{sgs}. This diffusion term can be written as,

$$\tau_{ij} = \nu_{\text{sgs}} \bar{S}_{ij} \tag{2.36}$$

where $\bar{S}_{ij}$ is the resolved strain rate tensor. Using resolved scales only, the objective is to approximate the instantaneous energy flux $\tilde{\epsilon}$ going through the cutoff, we have in particular,

$$\int_0^{k_c} 2k^2 E(k) dk = \langle 2\bar{S}_{ij}\bar{S}_{ij} \rangle = \langle |\bar{S}| \rangle. \tag{2.37}$$

Using the local equilibrium hypothesis in homogeneous turbulence, we arrive at an exact expression for the average energy flux,

$$\langle \tilde{\varepsilon} \rangle = \langle -\bar{S}_{ij}\tau_{ij} \rangle = \langle -\bar{S}_{ij}\nu_{\text{sgs}}\bar{S}_{ij} \rangle. \tag{2.38}$$

Reformulating this averaged relationship as a local one lead to the Smagorinsky model (Smagorinsky, 1963),

$$\nu_{\text{sgs}} = (C_S \bar{\Delta})^2 |\bar{S}|^{1/2} \tag{2.39}$$

The constant C_S, while theoretically evaluated at $C_S \approx 0.148$ has been subject to modifications that improved results in simulations. For e.g., (Métais and Lesieur, 1992) used $C_S = 0.18$ in isotropic homogeneous turbulence while (Deardorff, 1970) adjusted to $C_S \approx 0.1$ for a channel flow and similarly (Meneveau, 1994) found $C_S \approx 0.1 - 0.12$. Many other categories of models are based on spectral formulations of the energy spectrum using results of the Eddy Damped Quasi-Normal Markovian (EDQNM) models (Chollet and Lesieur, 1981). An effective viscosity ν_{e} can be determined from the shape of the spectrum in the inertial range and the filtering kernel type and derived from the subgrid transfer term,

$$T_{\text{sgs}}(k \,|\, k_c) = -2k^2 E(k) \nu_{\text{e}}(k \,|\, k_c) \tag{2.40}$$

with ν_{e} proportional to the asymptotics value of the effective viscosity at wavenumbers $k < k_c$ and local variations of the effective viscosity around k_c. Taking advantage of both *structural* and *functional* modeling, some have proposed mixed models (Bardina, 1983).

Dynamic procedure. In order to adapt the coefficient of SGS viscosity ν_{sgs} to the local structure of the flow, Germano et al. (1991) proposed an algorithm that computes a coefficient $C_{\text{sgs}}(\mathbf{x}, t)$ which is local in space and time. The idea is to write the Germano identity with a second filter of associated length $\tilde{\bar{\Delta}}$ so that $\tilde{\bar{\Delta}} > \bar{\Delta}$. In practice, $\tilde{\bar{\Delta}} = 2\bar{\Delta}$ is often used,

$$L_{ij} = T_{ij} - \tilde{\tau}_{ij} \tag{2.41}$$

where T_{ij} is the SGS tensor corresponding to the second filter, obtained as

$$T_{ij} = \tilde{\bar{U}}_i \tilde{\bar{U}}_j - \widetilde{\overline{U_i U_j}}. \tag{2.42}$$

Now, assuming that both τ and T can be modeled by the same constant C_{sgs}, we can use the following relations,

$$\tau_{ij} = C_{\text{sgs}} \alpha^{\tau}_{ij}, \tag{2.43}$$

$$T_{ij} = C_{\text{sgs}} \alpha^{T}_{ij} \tag{2.44}$$

where α^{τ} and α^{T} correspond to a given model without its fixed coefficient evaluated from the first and second filters, respectively. Assuming that C_{sgs} is constant over the test filter length, we can write the Germano identity as a linear system,

$$L_{ij} = C_{\text{sgs}} \alpha^{T}_{ij} - \widetilde{C_{\text{sgs}} \alpha^{\tau}_{ij}} = C_{\text{sgs}} \alpha^{T}_{ij} - C_{\text{sgs}} \tilde{\alpha}^{\tau}_{ij}. \tag{2.45}$$

However, the linear system is overconstrained (multiple equations for 1 unknown), and Lilly (1992) proposed to solve for C_{sgs} using the least-square formulation

$$C_{\text{sgs}} = \frac{M_{ij} L_{ij}}{M_{ij} M_{ij}} = \frac{(\alpha^{T}_{ij} - \tilde{\alpha}^{\tau}_{ij})(C_{\text{sgs}} \alpha^{T}_{ij} - C_{\text{sgs}} \tilde{\alpha}^{\tau}_{ij})}{(\alpha^{T}_{ij} - \tilde{\alpha}^{\tau}_{ij})^2}. \tag{2.46}$$

In practice, however, the constant can locally take negative values and thus produce backward energy transfers. This is known to create numerical instabilities in the simulations, which can be mitigated by averaging the numerator and denominator of (2.46),

$$C_{\text{sgs}} = \frac{\langle M_{ij} L_{ij} \rangle}{\langle M_{ij} M_{ij} \rangle}. \tag{2.47}$$

ML perspective. Interestingly, only simple models are used in modern solvers since incremental improvements have been limited to specific use cases and more complex models often lead to robustness issues. ML-based models are expected to be universal approximations of the SGS term but we pointed out that *structural*-based models are often unstable in simulations. They also suffer from a similar limitation, i.e., the lack of generalization to conditions outside of the training data. Still, an interesting goal would be to combine these parametric models with knowledge from *functional* modeling in order to improve both numerical stability and capability to reproduce the transfers between subgrid and resolved scales.

2.2 Specific applications

It is important to remember that model universality is only a hypothesis. The intuition motivating ML models is that we should be able to learn a "generalized" model that performs relatively well under the same cascade hypothesis circumstances in the following applications.

The NS equations are general equations used to describe fluid motion. In order to study some particular phenomena, they are often coupled with other sets of PDEs that may interact,

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} = -\nabla p + \nu\nabla^2\mathbf{u} + \overleftarrow{f}\,(\mathbf{y}_1,\cdots,\mathbf{y}_n), \quad \nabla\cdot\mathbf{u} = 0 \\ \dfrac{\partial \mathbf{y}_1}{\partial t} = \overrightarrow{f_1}(\mathbf{u},\mathbf{y}_1,\cdots,\mathbf{y}_n) \\ \cdots \\ \dfrac{\partial \mathbf{y}_n}{\partial t} = \overrightarrow{f_n}(\mathbf{u},\mathbf{y}_1,\cdots,\mathbf{y}_n). \end{cases} \tag{2.48}$$

Some examples include thermal convection when coupled with an equation state, or magnetohydrodynamics (MHD) when coupled with Maxwell's equations.

2.2.1 Scalar field modeling

In this thesis, both of the specific applications involve the evolution of a scalar quantity, i.e., concentration C and vorticity ω. Their filtered form also involves a subgrid term, which is identically defined for any scalar quantity y,

$$\left(\frac{\partial y}{\partial t}\right)' = \frac{\partial}{\partial x_i}\left(\bar{U}_i\bar{y} - \overline{U_i y}\right) \tag{2.49}$$

$$= \frac{\partial}{\partial x_i}\tau_i. \tag{2.50}$$

In this context, functional models have a slightly different form and rely on the gradient of the considered scalar quantity,

$$\tau_i = \nu_{\text{sgs}}\frac{\partial \bar{y}}{\partial x_i}. \tag{2.51}$$

Now, the definition of ν_{sgs} depends on functional consideration, as described in section 2.1.4 depending on the transfer term,

$$T_{\text{sgs}}(\bar{y}) = \bar{y}\frac{\partial \tau_i}{\partial x_i}, \tag{2.52}$$

$$T_{\text{sgs}}(\bar{U}) = \bar{S}_{ij}\tau_{ij}. \tag{2.53}$$

In general, the scaling is different between two-dimensional and three-dimensional turbulent systems.

2.2.2 Advection-diffusion

In this thesis, we study how a scalar field C, in particular concentrations of (bio)chemical species, can be transported by a fluid. The quantity C is said to be passive if it does not have any dynamical effect on the fluid motion itself. The dynamics of the passive scalar field is governed by the advection-diffusion equation,

$$\frac{\partial C}{\partial t} + (\mathbf{u} \cdot \nabla)C = \kappa \nabla^2 C + F_C \tag{2.54}$$

where κ is the diffusivity and F_C again, an external force. Note that even if this equation is linear in C, the evolution of a passive scalar by a turbulent field has been shown to be a difficult problem. The dynamics produced by the passive scalar is subject to chaos and a large range of scales, but also displays characteristics completely different from those of the advecting velocity field. The non-dimensional version of the (2.54) using quantities (2.2) is given by,

$$\frac{\partial C}{\partial t'} + (U' \cdot \nabla)C = \frac{1}{\mathrm{Pe}} \nabla^2 C + F_C \tag{2.55}$$

where $\mathrm{Pe} = \mathrm{Sc}\,\mathrm{Re}$ is the Peclet number. Here, Re is a parameter of the NS equation, and the advection-diffusion is controlled by the ratio of diffusive and viscous effects,

$$\mathrm{Sc} = \frac{\nu}{\kappa}. \tag{2.56}$$

For more details about passive scalars, we refer the reader to the review book (Warhaft, 2000).

2.2.3 The barotropic quasi-geostrophic approximation

In general, the NS are studied in three-dimensional space, where $\mathbf{u} = (u, v, w) \in \mathbb{R}^3$, but the study of their two-dimensional counterpart is also of interest, primarily motivated by the understanding of geophysical flows, for which the third dimension can be neglected. Although they follow the same equations, two-dimensional and three-dimensional flows are very different in nature. In particular, kinetic energy is transferred forward (from the largest toward the smallest scales) in three dimensions, whereas it is transferred backward (from the smallest towards the largest scales) in two dimensions. One consequence is that the two-dimensional flow dynamics is dominated by large-scale coherent structures such as vortices or jets, while the three-dimensional dynamics tends to transfer energy to the viscous scale where it is dissipated as heat. It is particularly interesting to notice that the curl of flow velocity, also referred to as the vorticity ω is a scalar quantity in two dimensions, i.e. $\omega = \nabla \times \mathbf{u} = \partial v/\partial x - \partial u/\partial y$. We can then reformulate the NS equations

as a conservation law for the vorticity. Indeed, taking the curl of (2.1) gives

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\, \omega = \nu \nabla^2 \omega + F_\omega, \tag{2.57}$$

$$\mathbf{u} = \mathbf{e}_z \times \nabla \psi, \tag{2.58}$$

$$\omega = \nabla^2 \psi \tag{2.59}$$

where the velocity is expressed as the curl of a streamfunction ψ and the out-of-plane unit vector $\mathbf{e}_z = (0, 0, 1)$. Also note that for convenience, we can transform the non-linear advection term on the left-hand side of (2.57) as the Jacobian determinant N_ω of ω and ψ,

$$\begin{aligned}(\mathbf{u} \cdot \nabla)\, \omega &= (\mathbf{e}_z \times \nabla \psi \cdot \nabla)\, \omega \\ &= \partial_x \psi \partial_y \omega - \partial_y \psi \partial_x \omega \\ &= N_\omega. \end{aligned} \tag{2.60}$$

The quasi-geostrophic equations describe a flow in a rotating frame but use the *beta-plane* approximation and the *geostrophic* balance (Bouchet and Venaille, 2012) that are acceptable when describing mid and high-latitude ocean and atmosphere flows. Here, applying the simplifications gives,

$$\frac{\partial q}{\partial t} + N_q = \nu \nabla^2 q - \mu q - \beta \partial_x \psi - \frac{1}{R^2} \psi + F_q \tag{2.61}$$

$$q = \nabla^2 \psi - \psi / R^2. \tag{2.62}$$

where q is called potential vorticity, μ is a large-scale drag coefficient, β is the Earth rotation vector approximation by beta-plane and R is the Rossby radius of deformation. This coefficient plays a central role in geostrophic dynamics as it describes the length scale at which rotational effects become important in the flow. However, we will here consider purely barotropic dynamics, i.e., in the limit of an infinite Rossby radius of deformation $R = \infty$. We finally arrive to the barotropic QG equations as a function of vorticity $q = \nabla^2 \psi = \omega$,

$$\frac{\partial \omega}{\partial t} + N_\omega = \nu \nabla^2 \omega - \mu \omega - \beta \partial_x \psi + F_\omega. \tag{2.63}$$

Note also that two-dimensional systems can have multiple vertical layers (Shevchenko and Berloff, 2015) corresponding for example to higher and lower parts of the ocean or atmosphere, with different sets of parameters. In two-dimensional flows, functional modeling has to take a different approach since the dynamics is quite different, i.e. we have a forward enstrophy cascade. To account for this difference, the Leith model (Leith, 1996) is based on the local gradient of vorticity $\bar{\omega}$,

$$\nu_{\text{sgs}} = (C_L \bar{\Delta})^3 |\nabla \bar{\omega}| \tag{2.64}$$

For more details about geophysical flows, we refer the reader to the review book (Majda and Wang, 2006).

2.3 Numerical solver

In order to run direct simulations (Orszag and Patterson Jr, 1972; Rogallo, 1981) of the equations described above, we need a discretize the temporal and spatial variables. In this thesis, we are interested in studying turbulence models in idealized configurations, i.e. where the geometry is simple and the error related to discretizations is minimal. This relaxed constraint allows us to use a pseudo-spectral method to discretize the space on a grid. In all the simulations performed here, we use a Fourier discretization, from which the domain boundaries are always periodic and grid spacing is uniform over the domain. An advantage over the finite(-differences, volumes, elements) family of numerical methods is that the precision when computing partial derivatives converge much faster w.r.t. the grid resolution. In the pseudo-spectral method, trial functions are represented by infinitely differentiable global functions, often as orthogonal polynomials. In this context, equations are solved in spectral space, except for the non-linear terms. There exist many orthogonal polynomials which all have specific constraints about domain boundaries and grid spacing. Let us recall the Fourier transform of grid quantity,

$$\hat{\mathbf{y}}(k) = \mathcal{F}\{\mathbf{y}(\mathbf{x})\}. \tag{2.65}$$

In spectral space, computing a spatial derivative corresponds to a multiplication by a wavenumber *coefficients* (2.20),

$$\frac{\partial \hat{\mathbf{y}}}{\partial x} = ik_x \hat{\mathbf{y}}. \tag{2.66}$$

A non-linear multiplication in spectral space involves a convolution, which is expansive in practice and should be avoided. It is thus common to compute non-linear products $\mathbf{y}_i\mathbf{y}_j$ in physical space.

Dealiasing. This non-linear product in discrete space leads to an aliasing error for unresolved values at high wavenumbers. In practice, it is possible to reduce this error by truncating wavenumbers that are impacted by the aliasing, using the so-called "p/q" rule,

$$\mathcal{F}\{\mathbf{y}_i\mathbf{y}_j\}(k) = 0, \quad \forall |k| \geq \frac{qk_{\max}}{p}. \tag{2.67}$$

It is also possible to remove entirely the aliasing error by computing the non-linear product on a grid containing "q/p" more wavenumbers. The most used region in the literature is found at $p, q = 2, 3$ with $p < q$.

Time advancement. A PDE discretized with the pseudo-spectral method can be time-stepped forward in spectral space as a composition of a linear $\mathbf{L}$ and non-linear $\mathbf{N}$ parts,

$$\hat{\mathbf{y}}^{(n+1)} = \hat{\mathbf{y}}^{(n)} + \Delta t \left(\hat{\mathbf{L}}\hat{\mathbf{y}} + \hat{\mathbf{N}}(\hat{\mathbf{y}}) \right). \tag{2.68}$$

In practice, we use Runge Kutta (RK) schemes or exponential variants (Kassam and Trefethen, 2005) of different orders.

2.4 Data-driven models

Data-driven approaches, and particularly machine learning are now commonplace in the scientific community (Rackauckas et al., 2020). The popularity of machine learning is obviously related to the amount of available data (Zhou et al., 2017) and computational resources (Waldrop, 2016) particularly from GPUs and derived TPUs (Tensor Processing Units) (Wang et al., 2019). In this thesis, we will apply our methodologies to deep learning (Goodfellow et al., 2016), which is also now widely understood as the state-of-the-art approach for any computer vision, speech recognition, or natural language processing tasks and gaining interest in the scientific community as well. The three types of machine learning algorithms are reinforcement, unsupervised, and supervised. In *reinforcement learning*, the model is constantly improving based on feedback from experiences. The model can perform actions and receive a corresponding score that it will try to maximize. In *unsupervised learning*, only inputs $\mathbf{y}$ are known, without any information about the expected labels $\mathbf{z}$. The goal of the algorithm is then to discover patterns from the data classified into clusters (e.g. K-means clustering) or transformed using dimensionality reduction (e.g. PCA). In *supervised learning*, inputs $\mathbf{y}$ and labels $\mathbf{z}$ are known and the goal is thus to construct a function f that accurately maps $\mathbf{y}$ to $\mathbf{z}$. The problem of SGS modeling can be defined as a supervised learning algorithm if we know how to compute the exact SGS term, or a reinforcement learning problem if this is not the case. In this thesis, we will assume that direct simulations are possible and that we can indeed generate data for labels $\mathbf{z}$.

Regression. Learning to reproduce the SGS dynamics is equivalent to a regression problem, i.e. the labels $\mathbf{z}$ and predictions $\mathring{\mathbf{z}}$ are continuous real values with $f : \mathbb{R}^m \to \mathbb{R}^n$. To be trainable, the function f also belongs to a family of parametric functions, i.e. it is built from independent variables called parameters, and denoted θ. We write a trainable function as,

$$\mathring{\mathbf{z}} = f(\mathbf{y} \,|\, \theta). \tag{2.69}$$

Now, to find the parameters θ, it is required to define a loss function $\mathcal{L}$ that quantifies the mismatch between labels $\mathbf{z}$ and model prediction $\mathring{\mathbf{z}}$. The mean-squared-error (MSE) is a commonly used loss function for regression tasks,

$$\mathcal{L} := \sum_{i=1}^{S} (\mathbf{z}^{(i)} - \mathring{\mathbf{z}}^{(i)})^2 \tag{2.70}$$

over S samples from the training data, containing an ensemble of pairs $(\mathbf{y}_i, \mathbf{z}_i)$. Learning the parameters θ can be carried as a mathematical optimization problem, where the objective is to minimize the loss function $\mathcal{L}$,

$$\arg\min_{\theta} \mathcal{L}(\mathbf{z}, f(\mathbf{y} \,|\, \theta)). \tag{2.71}$$

In the general case, the function f is non-convex and it is not guaranteed to find a global minimum. The list of optimized used in machine learning applications is large (Jain et al., 2017), but most state-of-the-art algorithms are derived from gradient descent, from which one step is given as,

$$\theta^{(n+1)} = \theta^{(n)} - \eta \nabla_\theta \mathcal{L}(\mathbf{z}, f(\mathbf{y} \,|\, \theta)) = \theta^{(n)} - \eta \frac{1}{S} \sum_{i=1}^{S} \frac{\partial \mathcal{L}(\mathbf{z}^{(i)}, f(\mathbf{y}^{(i)} \,|\, \theta))}{\partial \theta} \tag{2.72}$$

where η is the learning rate and n the current iteration of the algorithm. Trying to find an optimal local minimum is often dealt with using *stochastic* gradient descent (SGD), where the loss is computed on random subsets $S_{\text{SGD}} < S$ of the training data. Among more involved variants, the Adam optimizer (Kingma and Ba, 2015) is the most popular algorithm to train neural networks.

Building blocks. As mentioned above, the function f is a parametric function. To be precise, f is often defined as successive operations, or building blocks that themselves contain (trainable) *parameters*. For one-dimensional problems, it is common to define trainable operations as dense matrix-vector multiplications,

$$\text{Dense}(\mathbf{y} \,|\, W_\theta, b_\theta) \equiv W_\theta \mathbf{y} + b_\theta, \qquad \mathbf{y} \in \mathbb{R}^m \tag{2.73}$$

with $W_\theta \in \mathbb{R}^{m \times n}$ is a weight matrix and $b_\theta \in \mathbb{R}^n$ is a bias vector. The architectures using those blocks are called fully-connected neural networks. In multi-dimensional problems, taking advantage of the spatial representation of the data is possible using convolutions,

$$\text{Conv}(\mathbf{y} \,|K_\theta, b_\theta) \equiv K_\theta * \mathbf{y} + b_\theta, \qquad \mathbf{y} \in \mathbb{R}^{m_1 \times \cdots \times m_D} \tag{2.74}$$

where $*$ is a discrete convolution operator. The neighborhood of the convolution is controlled by the kernel K_θ. Now, functions built from dense matrix-vector multiplications and convolutions are not able to reproduce non-linear maps, since these operations are linear. To introduce non-linearity in the function, it is possible to combine the linear basis functions with non-linear *activation* functions. One popular example introduced by Rosenblatt (1958) is the rectified linear unit (Relu),

$$\text{Relu}(\mathbf{y}) = \max(0, \mathbf{y}). \tag{2.75}$$

Many more building blocks provide benefits in training generalization, performance, or convergence speed, but we won't discuss them in this thesis. Our goal here is really to combine our knowledge from the physics to improve parametric functions f at equal complexity, i.e. an equivalent number of trainable parameters and evaluation time.

Chapter 3

Invariances for subgrid models with machine learning

This chapter describes the application of different sets of assumptions (or inductive biases) to the problem of a scalar concentration transported by a turbulent flow. **Approximate** and **exact** methods are first discussed with existing applications to fluid dynamics problems. The unphysical behavior of a trained neural network is demonstrated when the testing regime differs from the one used during training (for example by changing the external forcing form). In addition to providing an interpretable framework, reducing the dimensionality of the problem using physical knowledge is shown to improve the performance and generalization of the trained model. Some unaddressed limitations and possible future avenues, in particular to phenomena that can not be expressed as law invariances are discussed at the end of the chapter.

The results presented in this chapter have been published *in Frezat et al. (2021).*

Contents

3.1 Embedding physical knowledge in machine learning models

Following the recent advances in deep learning for turbulence modeling (Duraisamy et al., 2019), neural networks (NNs) are expected to be good candidates for formulating subgrid models and offer a promising alternative to physics-based models. Initial works in this direction were initiated by the modeling of SGS momentum in both incompressible (Gamahara and Hattori, 2017; Xie et al., 2020) and compressible (Xie et al., 2019) flows. These data-driven approaches have also been shown to outperform classical models in specific applications, for example in the estimation of the subgrid-scale reaction rates (Lapeyre et al., 2019) or the subgrid modeling of ocean dynamics (Bolton and Zanna, 2019). However, these models do lack embedded physical laws and are often thought of as "black-box" in which our comprehension is limited. This has two major implications for the model; first, the subgrid model is only able to predict a narrow range of flows that have been used during training and suffer from rather limited generalization (or extrapolation) capabilities. Then, and most importantly, the subgrid model will situationally predict unrealistic terms that do not follow the expected behavior from well-known physical laws. Knowing that NNs are universal approximators (Hornik et al., 1989), we still have to realize that this only means that we *can* represent a wide variety of functions *but* we do not have any universal way to construct the weights that lead to this conclusion. In this regard, one of the benefits of embedding physical knowledge in machine learning models is to reduce the dimensionality of the inverse problem we are considering, which has the additional potential to speed up the learning phase and reduce the quantity of required training data. The first successful explicit embedding of known invariance has shown great success when applied to Reynolds Averaged Navier Stokes (RANS) simulations (Ling et al., 2016a,b). We describe in the following subsections different techniques (see Table 3.1) to embed invariances in a machine learning framework.

Subsection	Technique	Nature	Explicit examples
3.1.1	Data manipulation	Both	Bolton and Zanna (2019); Kim and Lee (2020); Lapeyre et al. (2019)
3.1.2	Loss function	Approx.	Bode et al. (2021); Charalampopoulos and Sapsis (2022)
3.1.3	Architecture	Exact	Beucler et al. (2021a); Ling et al. (2016a,b); Mohan et al. (2020)

Table 3.1: Identified techniques for physical invariances embedding with explicit examples of applications to subgrid modeling.

3.1.1 Data manipulation

Pre-processing. The first applications with machine learning (ML) are using pointwise-based architectures in which only local information is known. Careful choice of the input parameters of the model is thus required because of the inherent non-local nature of the considered PDEs. In a geometrical space $\mathcal{G}$, inputs of the model are compositions f of spatial derivatives of model states $\mathbf{y}$ (or filtered velocities $\mathbf{u}$ in the case of SGS momentum models), such that

$$\left\{ f\left(\frac{\partial y_p}{\partial x_p}\right)\right\}, x_p \in \mathcal{G}^p. \tag{3.1}$$

Interestingly, providing gradients to the model is already exploiting some useful properties of the input. For example, the gradient is linear for any vector field at a point $a \in \mathbb{R}^p$ for any constant α,

$$\nabla(\alpha \mathbf{y})(a) = \alpha \nabla \mathbf{y}(a). \tag{3.2}$$

In practice, at most second-order partial derivatives are provided to the model in order to avoid both computational complexity and extensive required memory storage.

Post-processing. While not tied to machine learning methods, post-processing the output of the model is an efficient way to constrain the prediction distribution to be in an acceptable range for subsequent use. For any trained model $\mathcal{M}$, it is possible to apply a transformation h after training,

$$\mathbf{z}' = h(\mathcal{M}(\mathbf{y})). \tag{3.3}$$

It has been particularly useful to preserve momentum when predicting subgrid dynamics (Bolton and Zanna, 2019) using separate models for each spatial direction. The authors demonstrated that removing the spatial mean from the recombined predictions is indeed enforcing global momentum conservation, drastically improving the performance of the model without sacrificing accuracy.

Augmentation. If we want a model to reproduce a particular invariance, one option would be to expose the corresponding data during the training process. The idea behind data augmentation is to include new (input, output) pairs that verify some identities from the initial data,

$$\{\mathbf{y}, \varphi(\mathbf{y})\} \rightarrow \{\mathbf{z}, \varphi'(\mathbf{z})\} \tag{3.4}$$

for which there exists an identity between φ and φ' that we wish to learn with the model. Also, note that data augmentation increases the amount of data which helps reduce overfitting. This has been used in turbulence modeling for rotations (Lapeyre et al., 2019) and for the statistical symmetry between (x, y, z) and $(x, y, -z)$ in a channel flow (Kim and Lee, 2020).

3.1.2 Penalizing loss function

Machine learning models are trained by optimization algorithms that seek to minimize a loss function. In the case of NNs, the employed optimization algorithm is based on some form of gradient descent, for which the loss function is required to be scalar-valued $\ell : \mathbb{R}^n \to \mathbb{R}$, commonly chosen in regression problems to be defined as a norm between the target and the model prediction. For example, to train a model to predict a quantity $\mathbf{z}$ from inputs $\mathbf{y}$, we would define a loss function of the form

$$\ell(\mathbf{z}, \mathring{\mathbf{z}}) := ||\mathbf{z} - \mathring{\mathbf{z}}||_p = \left(\sum_{i \in B} |\mathbf{z}_i - \mathring{\mathbf{z}}_i|^p \right)^{1/p} \tag{3.5}$$

where $\mathring{\mathbf{z}} = \mathcal{M}(\mathbf{y})$ is a model prediction from given inputs and B is a finite set corresponding to a training batch (i.e. a subset of the entire dataset). The problem with L^p norms can tend towards zero and still break some symmetries about the trained functional. This means that even if the optimization algorithm completes with $\ell = 0$, the only guarantee is given by the loss, which most of the time is not a direct quality indicator for the model in terms of the underlying physics. In order to guide the training, it is common to have multiple weighted terms in the loss function that directly corresponds to some identities that we want to verify, for example,

$$\ell(\mathbf{z}, \mathring{\mathbf{z}}) := \alpha_1 \ell_1(\mathbf{z}, \mathring{\mathbf{z}}) + \cdots + \alpha_n \ell_n(\mathbf{z}, \mathring{\mathbf{z}}). \tag{3.6}$$

The challenge with this type of composite loss is in the choice of the weights $\alpha_n \in \mathbb{R}$, which gives a sense of importance to the corresponding term. This can quickly get complicated if the losses are defined as different types of norms and/or if they map to values that have different orders of magnitude.

In practice. A 4-term loss function was used by Bode et al. (2021) to model the subgrid term in turbulent reactive flows. The architecture used in this work is known as Generative Adversarial Network (GAN) and requires an adversarial term in the loss. Additionally, authors have used the MSE of the subgrid term and its gradients with a L_1 norm for the continuity part of the incompressible Navier Stokes equations that constrain the divergence of the velocity field to be zero. It is also important to mention that all loss weights α_n were taken equivalent and non-dimensional, provided that the operators and input fields are also non-dimensionalized.

Recently, a 2-term loss function was used by Charalampopoulos and Sapsis (2022) in order to constrain the energy conservation property of the non-linear advection terms for both momentum and inertial tracers in two-dimensional jets. The adoption of this constraint was shown to not only increase the accuracy of the model but also to improve its stability.

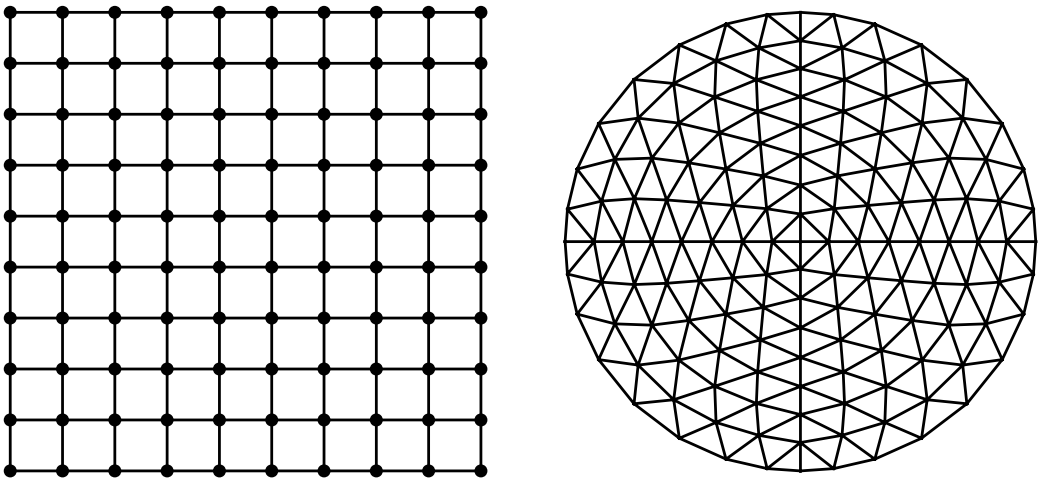

Figure 3.1: Two-dimensional examples of structured (left) and unstructured (right) geometries. CNNs are particularly used to represent Cartesian geometries on structured grids, while GNNs can represent complex shapes on unstructured meshes composed of polygons (usually triangles or quadrilaterals).

3.1.3 Architecture embedding

Architecture families. Many studies have investigated Convolutional Neural Networks (CNNs) to the application of the subgrid problem (Beck et al., 2019; Pawar et al., 2020). Being based on convolutions kernels, they operate on spatial neighborhoods and are particularly suited for data having a geometrical extent in a two or three-dimensional space represented as a structured grid (i.e. with equivalent spacing Δx_p between each grid point and in every dimension (see Fig. 3.1, left). Using this type of architecture already guarantees translation equivariance, which is a desired property of any equations of motion. More recently, Graph Neural Networks (GNNs) became quite popular, with a few applications in the context of turbulence modeling (Belbute-Peres et al., 2020). This family of architecture opens the possibility for unstructured geometries (see Fig. 3.1, right) which is in fact the standard type of geometry used in realistic solvers for both geophysical and industrial flows. The graph layout also guarantees permutation invariance, i.e. the outputs should depend on the set of inputs and not a specific ordering of its elements.

Functional blocks. Remember that neural networks are built from a series of (differentiable) blocks that have trainable parameters. It is possible to combine these blocks with operations that do not have trainable components but provide some identities. Typically, one can describe a model

$$\mathcal{M}(\mathbf{y} \,|\, \theta) := h_1 \circ \mathcal{M}_1(\theta_1) \circ \cdots \circ h_n \circ \mathcal{M}_n(\theta_n) \circ h_{n+1} \tag{3.7}$$

where θ are the trainable parameters and h arbitrary differentiable operations that map

the input of one layer to another. As a simple example, it is possible to guarantee that our model outputs positive values if the last operation of the neural network is defined as $h_{n+1} : \mathbb{R}^n \to \mathbb{R}^n_+$. This approach is one of the most interpretable techniques to embed direct identities into the trained model since the expected outcome is analytically encoded and can be verified at machine precision.

Specific architectures in turbulence modeling. As mentioned in the introductory part of this chapter, the use of specific functional blocks was first described by Ling et al. (2016b) which proposed a new architecture that guarantees Galilean invariance through a tensor basis multiplicative layer. Later, Mohan et al. (2020) was able to embed the notion of incompressibility in a neural network for the task of coarse-graining turbulent velocity fields using a Helmholtz decomposition of the flow. As a general method, Beucler et al. (2021a) augments a predefined neural network with fixed conservation layers that sequentially enforce the constraint expressed as a linear system. This last approach was applied in the context of climate modeling, systematically reducing the error on the constrained variables.

3.2 Symmetries in advection-diffusion equations

Recall that from (2.54) the dynamics of a scalar transported in a turbulent flow arise as a key issue in many applications. We assume in this chapter that the Navier-Stokes equations are completely resolved (using DNS) so that velocity fields can be obtained at any time in both fine and coarse resolutions. The objective is now to model the evolution of a scalar concentration C on a coarse grid $\bar{\Omega}$ (see Fig. 3.2), such that

$$\begin{cases} \dfrac{\partial C}{\partial t} + (\mathbf{u}\cdot\nabla)C = \kappa\nabla^2 C + F_C, & \mathbf{u}, C \in \Omega \\ \dfrac{\partial \bar{C}}{\partial t} + (\bar{\mathbf{u}}\cdot\nabla)\bar{C} = \kappa\nabla^2 \bar{C} + F_{\bar{C}} + \underbrace{\nabla\cdot\left(\bar{\mathbf{u}}\bar{C} - \overline{\mathbf{u}C}\right)}_{\tau_C}, & \bar{\mathbf{u}}, \bar{C} \in \bar{\Omega} \end{cases} \tag{3.8}$$

where F_C is a time-dependent forcing term and the projection from fine to coarse grid is defined by a spatial operator $\mathcal{T} : \Omega \to \bar{\Omega}$. On coarse grid $\bar{\Omega}$, the equation can be solved except for τ_C which depends on fine grid variables $\mathbf{u}$ and C. This leads to the following inverse problem,

$$\tau_C \approx \mathcal{M}(\bar{\mathbf{u}}, \bar{C}). \tag{3.9}$$

In the particular setting of NNs, this comes to learning a model $\mathcal{M}$ that approximates the SGS scalar term τ_C and has already been explored in (Portwood et al., 2021; Vollant et al., 2017). However, as illustrated in 3.3.1, such NN models do not convey expected physical properties (e.g., invariance properties), which greatly impact their actual range of applicability. In this section, we choose four non-exhaustive physical relationships that

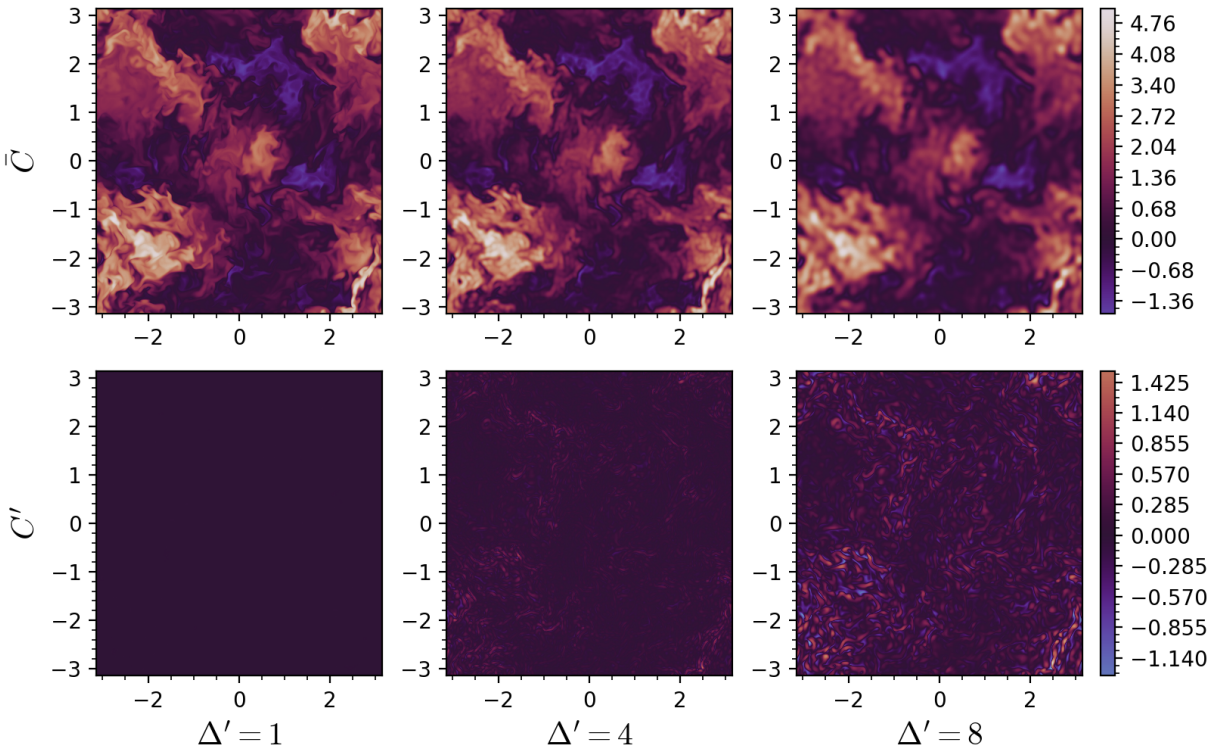

Figure 3.2: Data slices of filtered scalar concentration $\bar{C}$ and corresponding subgrid contribution $C' = C - \bar{C}$ at different grid ratios $\Delta' = \Delta/\bar{\Delta}$, filtered using a cut-off kernel. Note that a value of $\Delta' = 1$ indicates DNS resolution, i.e. $C = \bar{C}$ and $C' = 0$.

should be verified by any SGS scalar model and show how to include them within a NN as **exact** and **approximate** constraints.

3.2.1 Frame symmetry

The first three constraints are based on frame symmetries (Berselli et al., 2006) that involve translation, rotation, and Galilean invariances. These properties are directly related to the NS equations by which the scalar concentration is transported. In particular, for all solutions $\mathbf{u}$, we say that $g\mathbf{u}$ is also a solution for any $g \in G$ where G is the symmetry group of transformations acting on space-time functions $\mathbf{u}(\mathbf{x}, t)$ (Frisch and Kolmogorov, 1995). In the context of SGS modeling, we will refer to the invariant properties derived by Oberlack (1997) using the symmetries of the original NS equations.

Translation invariance. Translation invariance states that the considered SGS model shall not be location-dependent. In other words, model $\mathcal{M}$ shall satisfy the following constraint,

$$\forall \delta \in \mathbb{R}, \quad \mathcal{M}(T_\delta \bar{\mathbf{u}}, T_\delta \bar{C}) = T_\delta \mathcal{M}(\bar{\mathbf{u}}, \bar{C}) \tag{3.10}$$

where T_δ is a translation operator for spatial displacement δ, such that $T_\delta \mathbf{y}(\mathbf{x}) = \mathbf{y}(\mathbf{x}+\delta)$. We recall that in the deep learning literature, there exists a class of architecture called

CNNs that are described by a combination of convolution layers. Since the convolution commutes with translation, this type of architecture provides a built-in invariance to spatial displacements, which approximates (3.10). Convolutional architectures also appear as a natural choice when dealing with structure tensors (i.e. time, space, and space-time processes) as they relate to local filtering operations with respect to tensor dimensions. For instance, convolutional layers embed all discretized filtering operations, such as first-order or higher-order derivatives of the considered variables. This may provide the basis for some physical interpretations of CNN architectures. Importantly, CNNs greatly outperform fully-connected architectures and are currently the state-of-the-art schemes for a wide range of machine learning applications with one-dimensional (Ince et al., 2016), two-dimensional (Krizhevsky et al., 2012) or three-dimensional (Kamnitsas et al., 2017) states.

Rotation invariance. The next invariance is related to reflections and rotations. The coarse equation (3.8) holds under the rotation of the coordinate system and velocity vector,

$$\mathcal{M}(A_{ij}\bar{\mathbf{u}}_j, \bar{C}) = \mathcal{M}(\bar{\mathbf{u}}, \bar{C}) \tag{3.11}$$

for any rotation matrix A with $A^T A = AA^T = I$ in a rotated frame $\mathbf{y}_i = A_{ij}\mathbf{y}_j$. Ensuring rotation invariance in a NN is a difficult problem since convolutions kernels are not imposed but rather learned by the optimization algorithm. Some progress has been made to exploit these symmetries (Cohen and Welling, 2016; Weiler et al., 2018) but are still limited to finite subsets of rotations with a computational tradeoff between accuracy and computational complexity. A common and flexible way to approximate this invariance is to use data augmentation, i.e. extending the dataset with new rotated velocity fields $\bar{\mathbf{u}}$ mapping to the same non-rotated SGS term τ_C. We used permutations of our initial data that verify (3.11) with angles of 90 degrees.

Galilean invariance. The last fundamental frame invariance defines that turbulence is required to be the same in all inertial frames of reference. This property has been extensively discussed in the context of SGS modeling (Speziale, 1985). It writes easily for our problem as

$$\forall \beta \in \mathbb{R}, \quad \mathcal{M}(\bar{\mathbf{u}} + \beta, \bar{C}) = \mathcal{M}(\bar{\mathbf{u}}, \bar{C}). \tag{3.12}$$

We propose to ensure this invariance by reducing the velocity to a standardized quantity, i.e. with zero mean $\langle(\bar{\mathbf{u}} + \beta)\rangle = 0$ for each instantaneous sample. Let us denote the standardized operator $\cdot'$, we have

$$(\bar{\mathbf{u}} + \beta)' = \bar{\mathbf{u}} + \beta - \langle \bar{\mathbf{u}} + \beta \rangle \tag{3.13}$$

$$= \bar{\mathbf{u}} - \langle \bar{\mathbf{u}} \rangle \tag{3.14}$$

$$= \bar{\mathbf{u}}'. \tag{3.15}$$

Note that this assumption is true only if bias is removed from all convolution layers. We include this standardization step directly on the inputs of the model.

3.2.2 Scalar concentration linearity

The invariances described previously directly relate to the NS equation for the transport of scalar concentration. Now, the linearity of the advection-diffusion equations is a fundamental property that shall also be verified by the proposed SGS model. Formally, it comes to verify the following constraint,

$$\forall \lambda \in \mathbb{R}, \quad \mathcal{M}(\bar{\mathbf{u}}, \lambda \bar{C}) = \lambda \mathcal{M}(\bar{\mathbf{u}}, \bar{C}). \tag{3.16}$$

This relationship can not be verified within a classical CNN since the model contains non-linearities introduced by the activation functions. We propose to split the model into two sub-models $\mathcal{V}$ and $\mathcal{T}$ that take as inputs the velocities and the transported scalar concentration respectively. $\mathcal{V}$ involves a classic CNN architecture with activations. By contrast, to conform to the linearity property, model $\mathcal{T}$ applies a single convolution layer with no bias and non-linear functions. The resulting SGS model, called $\mathcal{M}_{\text{SGTNN}}$ for "**S**ub**G**rid **T**ransport **N**eural **N**etwork" is given by

$$\mathcal{M}_{\text{SGTNN}}(\bar{\mathbf{u}}, \bar{C}) = [\mathcal{V}(\bar{\mathbf{u}}) \times \mathcal{T}(\bar{C})] \circ \text{Conv}_+ \tag{3.17}$$

where $\times$ is a term-by-term matrix product and Conv_+ is the convolution kernel with unitary width $K = 1$. We sketch the resulting architecture in Fig. 3.3. The common pieces of the model are described in Sec. 3.3. Here, the particularities related to scalar concentration linearity lie in the application in parallel of the operators $\mathcal{V}$ and $\mathcal{T}$ and the concatenation of their $128 \times 128 \times 128$ tensors outputs as a term-by-term product. The last convolutional layer maps the 128-dimensional representation to a scalar field. We can now show that (3.16) holds,

$$\mathcal{M}_{\text{SGTNN}}(\bar{\mathbf{u}}, \lambda \bar{C}) = \sum_{i=1}^{d} \mathcal{V}(\bar{\mathbf{u}})^{(i)} \lambda \mathcal{T}(\bar{C})^{(i)} \circ \text{Conv}_+^{(i)} = \lambda \sum_{i=1}^{d} \mathcal{V}(\mathbf{u})^{(i)} \mathcal{T}(\bar{C})^{(i)} \circ \text{Conv}_+^{(i)} \tag{3.18}$$

where d is the input dimension of Conv_+ (here $d = 128$) and linearity of $\mathcal{T}$ is implied from its convolution. Finally,

$$\mathcal{M}_{\text{SGTNN}}(\bar{\mathbf{u}}, \lambda \bar{C}) = \lambda([\mathcal{V}(\bar{\mathbf{u}}) \times \mathcal{T}(\bar{C})] \circ \text{Conv}_+) \tag{3.19}$$

$$= \lambda \mathcal{M}_{\text{SGTNN}}(\bar{\mathbf{u}}, \bar{C}) \tag{3.20}$$

which validates that the architecture has a linear path on $\bar{C}$.

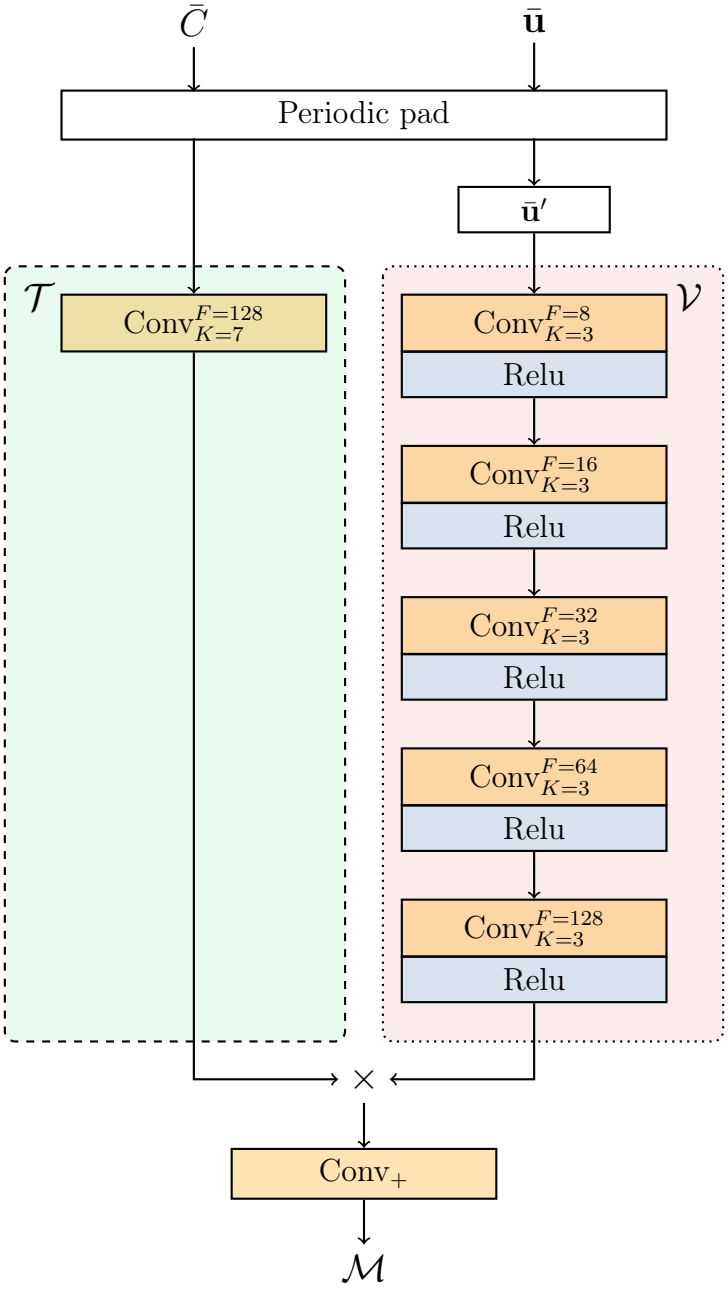

Figure 3.3: Sketch of the "**S**ub**G**rid **T**ransport **N**eural **N**etwork" architecture (SGTNN). At the top, two non-learnable steps are applied; the periodic pad replicates boundaries according to the width of convolution kernels and $\bar{\mathbf{u}}'$ standardizes the velocities to ensure Galilean invariance. Then, the two sub-models $\mathcal{V}$ (right) for the velocities and $\mathcal{T}$ (left) for the scalar concentration are represented in dotted and dashed lines, respectively. Convolutions are defined by kernel width K and number of features F, with last convolution $\mathrm{Conv}_+ \equiv \mathrm{Conv}_{K=1}^{F=1}$ applied to the combined result of the sub-models.

3.3 Numerical experiments

NN SGS models require the generation of a training dataset, usually coming from DNS, from which model parameters are learned. This dataset should contain a range of dynamics that is expected to be reproduced by the SGS model. When dealing with SGS scalar turbulence, we follow the hypothesis of Batchelor (1959), which states that the small scales of a scalar concentration in a turbulent flow at a high Reynolds number are statistically similar. Thus, the SGS dataset shall comprise all the scales up to the limit between the inertial subrange and the energy-containing range. To be precise, in the scalar-variance spectrum (Corrsin, 1951; Obukhov, 1970) given by

$$E_C(k) = C_* \chi \epsilon^{-1/3} k^{-5/3} \tag{3.21}$$

where C_* is the Oboukou-Corrsin constant, χ and ϵ are the scalar variance and kinetic energy dissipation rates respectively in spectral space at wavenumber k.

Data generation. The data used in this chapter are generated from a DNS of homogeneous isotropic turbulence. A classical pseudospectral code as described in 2.3 with second-order explicit Runge Kutta time advancement is used to solve the equations (2.1) and (2.54) in a triply periodic domain $[0, 2\pi]^3$. The size of the computational domain is larger than four times the integral length scale to ensure that the largest flow structures are not affected, and the domain is discretized using 512^3 grid points. The simulation parameters are chosen such that $k_{\max}\eta_K > 1.5$ and $k_{\max}\eta_B > 1.5$, where $k_{\max}$ is the maximum wavenumber in the domain, and η_K and η_B are the Kolmogorov and Batchelor scales, respectively. The Reynolds number based on the Taylor microscale Re_λ is around 160 and the turbulent molecular Schmidt number Sc is set to 0.7. The initial random velocity fields are generated in Fourier space as a solenoidal isotropic field with random phases and a prescribed energy spectrum as in (Rosales and Meneveau, 2005), for example. The initial scalar concentration is constructed from a large-scale random field according to the procedure described in (Eswaran and Pope, 1988). Statistical stationarity is obtained using a random forcing located at small wavenumbers $k_F \in [2, 3]$, for both velocities (Alvelius, 1999) and scalar concentration (da Silva and Pereira, 2007) fields. Applying the scalar spectral forcing only on the smallest wavenumbers allows us to contain its effect on the resolved scales of the simulation. We extract the SGS term and the non-residual input quantities (see Fig. 3.4) using spatial filtering described in 2.1.2, i.e.,

$$\{\bar{u}_x, \bar{u}_y, \bar{u}_z, \bar{C}\} \rightarrow \tau_C. \tag{3.22}$$

To avoid large correlations in the data, we ensure that every sample is separated by almost one large eddy turnover time $t_L \approx L/U \approx (L^2/\epsilon)^{1/3}$, with L the integral scale. We construct a dataset from 80 equally-spaced samples taken from 40000 temporal integration. The learning dataset is then split into two parts so that the first 60 samples are

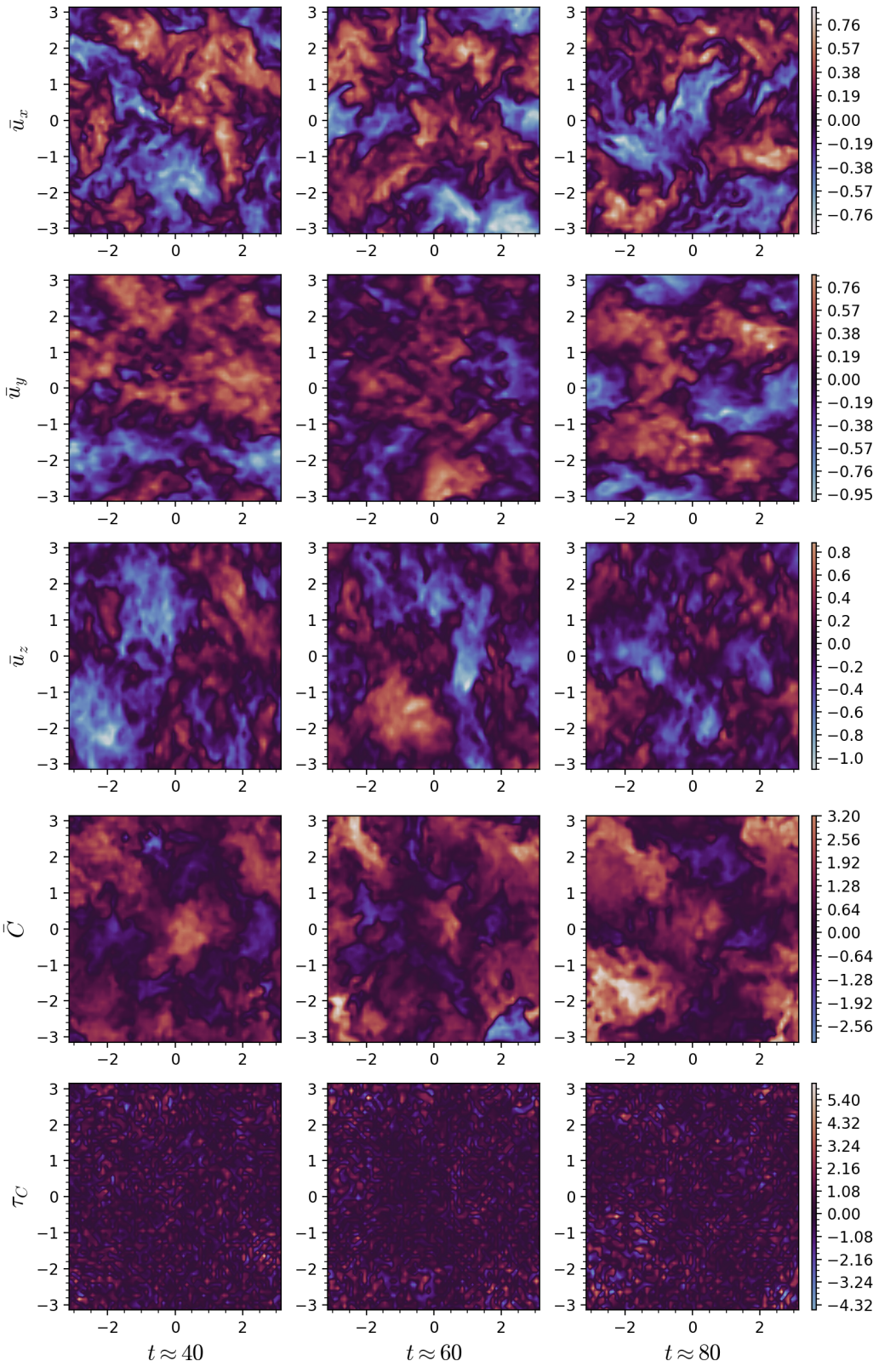

Figure 3.4: Data slices extracted during the simulation for the training dataset. $\bar{u}$ are the velocities in each spatial directions, $\bar{C}$ is the scalar concentration and τ_C is the SGS term. The simulation was run with $N = 512$ grid points and filtered at $N = 64$, i.e. $\Delta' = 8$. Note that the training dataset only contains statistically developed turbulence states.

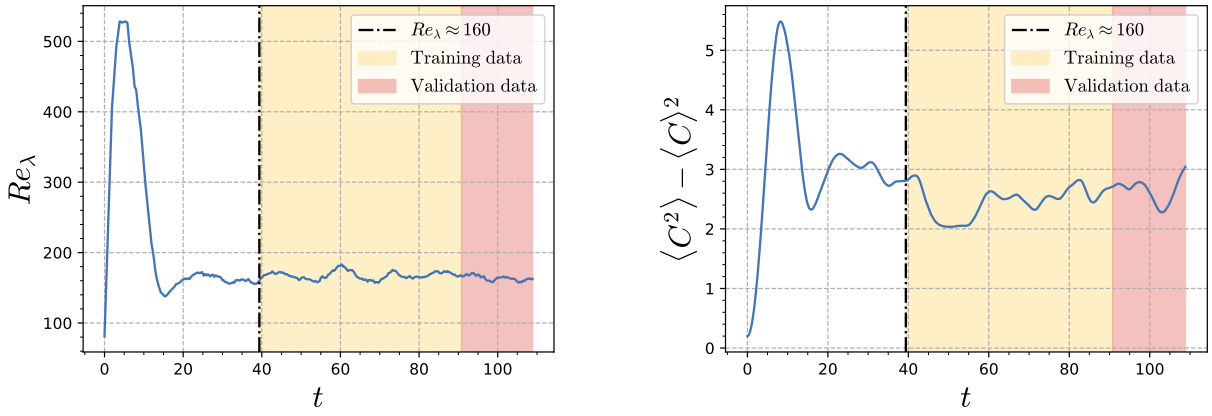

Figure 3.5: Evolution of the Reynolds number on the Taylor microscale (left) and scalar variance (right). The flow is considered in an established turbulence regime at $t \approx 40$. After convergence, the first 60 samples are used for training and the remaining 20 samples are used for validation.

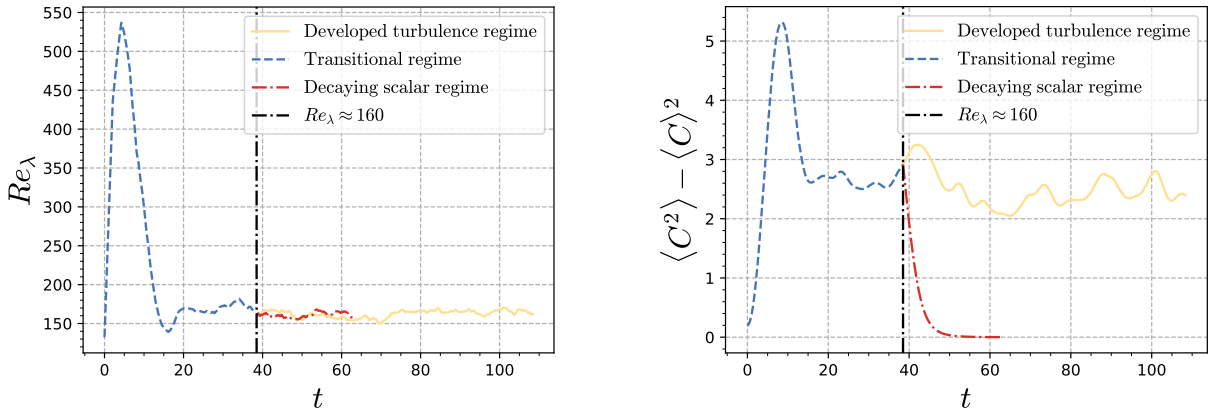

Figure 3.6: Evolution of the Reynolds number on the Taylor microscale (left) and scalar variance (right) in different regimes (developed turbulence, transitional and scalar decay) for the testing data.

dedicated to training and the remaining 20 samples are used to validate the model, which is particularly useful to monitor the behavior of the learning phase. For the test dataset, we follow the same methodology and extract three different flow regimes used for the *a priori* evaluation; (1) a developed turbulence statistically similar to the training data, where both velocity and scalar forcing terms are active; (2) a forced transitional regime to turbulence with fields initialized at large scales and (3) a scalar decay driven by a forced turbulent flow. Reynolds number and scalar variance evolution as well as data separation from the datasets are shown in Figs. 3.5 and 3.6 for training, validation, and testing, respectively.

NN architectures and learning schemes. In this chapter, we considered three different types of NNs. First, we replicate the baseline NN SGS from Portwood et al. (2021),

which relies on a Multilayer Perceptron (MLP) to predict the residual flux. This model uses the gradients of grid quantities $\bar{\mathbf{u}}$ and $\bar{C}$ as inputs and optimizes 8 dense layers Dense composed of 64 neurons with $\text{Relu}(x) = \max(0, x)$ non-linear activation and a final 512−neurons dense layer before the output layer. The MLP model writes

$$\begin{aligned}\mathcal{M}_{\text{MLP}}(\nabla\bar{\mathbf{u}}, \nabla\bar{C} \,|\, \theta) := &\,\text{Dense}^{(64)}(\theta_1) \circ \text{Relu} \circ \cdots \circ \text{Dense}^{(64)}(\theta_6) \circ \text{Relu} \\ &\circ \text{Dense}^{(512)}(\theta_7) \circ \text{Relu} \\ &\circ \text{Dense}^{(3)}(\theta_8).\end{aligned} \tag{3.23}$$

Note that the last layer maps to three values, since the original model was trained to predict SGS fluxes $\overline{\mathbf{u}C} - \bar{\mathbf{u}}\bar{C}$. We compute the divergence of this term exactly after training. We also compared a classical CNN composed of six non-linear convolution units, i.e. applying Relu after Conv with kernels K of size $3 \times 3 \times 3$ and increasing number of filters F from 8 to 128. Since our simulations are done in a periodic domain, we can replicate periodically our input at the boundaries (Pad_P operator) without introducing any error. This allows us to remove any padding inside the NN. The CNN model writes

$$\begin{aligned}\mathcal{M}_{\text{CNN}}(\bar{\mathbf{u}}, \bar{C} \,|\, \theta) := &\,\text{Pad}_P \circ \text{Conv}_{K=3}^{F=8}(\theta_1) \circ \text{Relu} \circ \cdots \circ \text{Conv}_{K=3}^{F=128}(\theta_5) \circ \text{Relu} \\ &\circ \text{Conv}_{K=1}^{F=1}(\theta_6).\end{aligned} \tag{3.24}$$

Finally, the transformation-invariant model $\mathcal{M}_{\text{SGTNN}}$ has the same structure as $\mathcal{M}_{\text{CNN}}$ except for the symmetry specificities, as described in 3.2. The optimization problem is formulated as the minimization of a loss function $\mathcal{L}$ which in our case will be defined as a simple mean-squared-error (L_2^2) on the SGS prediction. For a batch of S samples, we have

$$\mathcal{L}(\mathcal{M}) := \frac{1}{S}\sum_{i=1}^{S} ||\tau_C^{(i)} - \mathcal{M}(\bar{\mathbf{u}}^{(i)}, \bar{C}^{(i)})||_2^2. \tag{3.25}$$

Note that this loss is slightly different for the MLP since it has different input and output quantities. In practice, the minimization is carried out using the Adam optimizer (Kingma and Ba, 2015) for 1000 epochs. We use an adaptive step-based scheduler, which decreases the learning rate from $1e^{-4}$ by a factor of $\rho = 0.75$ every 250 epoch. As a pre-processing step, we normalize the input data to zero mean and unitary variance (or standardize), which has been shown to improve the convergence speed of gradient descent steps (Ioffe and Szegedy, 2015).

3.3.1 Unphysical behaviors across different forcing regimes

In practice, scalar concentration C is diffused over time, its dynamics can be controlled by the forcing term F_C. As shown in the temporal evolution in Figs. 3.5, the models are trained on simulation data in a turbulent equilibrium state. To maintain this state,

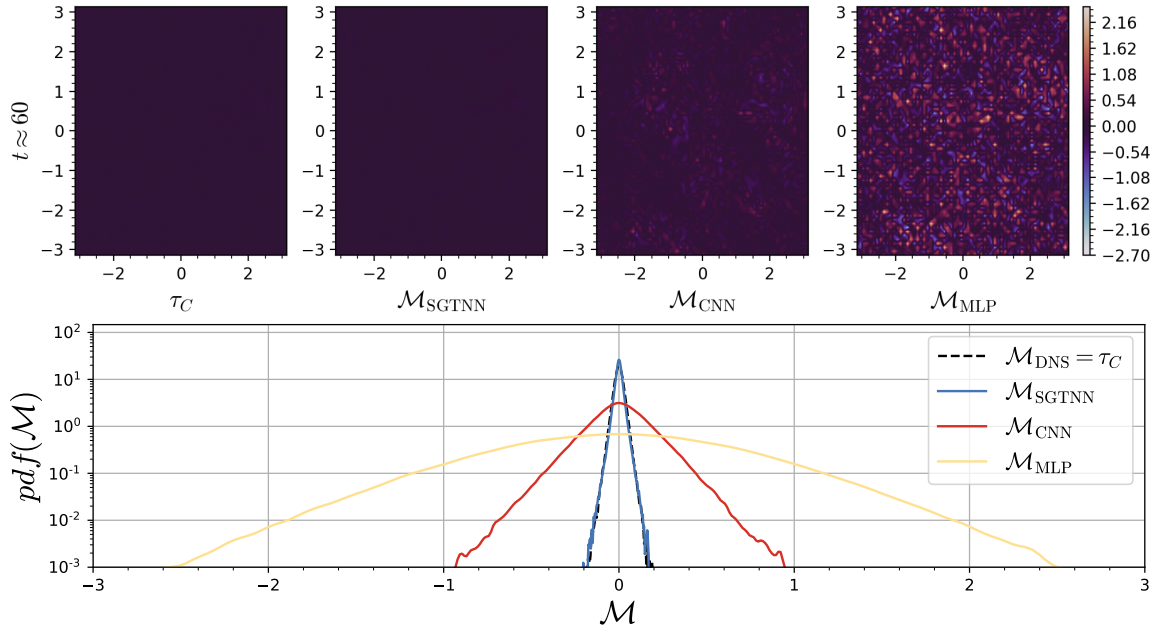

Figure 3.7: Data slices (top) and PDF (bottom) of the subgrid term τ_C and predictions by the different NN-based models at the end of the decay $t \approx 60$ when scalar variance is almost zero.

energy is injected at large scale (or small wavenumbers k_F), i.e.,

$$F_C(k) = 0, \quad \forall k \notin k_F \tag{3.26}$$

$$F_{\mathbf{u}}(k) = 0, \quad \forall k \notin k_F. \tag{3.27}$$

Remember also that the subgrid term τ_C is representing unresolved dynamics, which corresponds to a range of wavenumbers in the inertial range of the spectrum. In order to be independent of the effect of the source term, we must ensure that the unresolved dynamics do not overlap with the forcing wavenumbers k_F. In our experiments, DNS resolution is set at 512^3 and coarse resolution is downsampled on a 64^3 grid. In the wavenumber space, our simulations are defined on $|\mathbf{k}| = \frac{N}{2} + 1$ wavenumbers. The unresolved range in our numerical setup is then $k_{\bar{C}} \in [33, 257]$, which does not contains the forcing wavenumber $k_F \in [2, 3]$. Now, since modeling the subgrid term is in direct correlation with the definition of the forcing term, the models are expected to perform well in different forcing regimes. However, as we can see in Figs. 3.7, models trained without physical invariances ($\mathcal{M}_{\text{MLP}}$ and $\mathcal{M}_{\text{CNN}}$) are predicting a non-zero subgrid term τ_C when the scalar concentration has been completely diffused, which is not expected from a physical point of view. The model with invariances $\mathcal{M}_{\text{SGTNN}}$ on the other hand, is accurately reproducing the correct behavior, mostly due to the scalar concentration linearity constraint (3.16).

Physical invariances impact. We can also look more closely at the *a priori* performance of the different NN-based models exposed to the known invariances described in section

	$\mathcal{M}_{\text{MLP}}$		$\mathcal{M}_{\text{CNN}}$		$\mathcal{M}_{\text{SGTNN}}$	
Invariance	$\text{E}[\ell_{\text{rms}}]$	$\text{Var}[\ell_{\text{rms}}]$	$\text{E}[\ell_{\text{rms}}]$	$\text{Var}[\ell_{\text{rms}}]$	$\text{E}[\ell_{\text{rms}}]$	$\text{Var}[\ell_{\text{rms}}]$
(3.10)	1.499	0.47	1.028	0.11	**0.949**	**0.04**
(3.11)	1.229	1.81×10^{-8}	0.873	1.45×10^{-7}	**0.834**	$\mathbf{2.04 \times 10^{-8}}$
(3.12)	**1.229**	**0.00**	0.922	1.42×10^{-3}	**0.835**	**0.00**
(3.16)	2.724	5.81	1.122	0.28	**0.835**	**0.00**

Table 3.2: *a priori* evaluation of the four physical invariances discussed in the chapter on the testing developed turbulence data. We show the expectation and variance of the NRMSE on the SGS term τ_C predicted by NN-based models for each invariance.

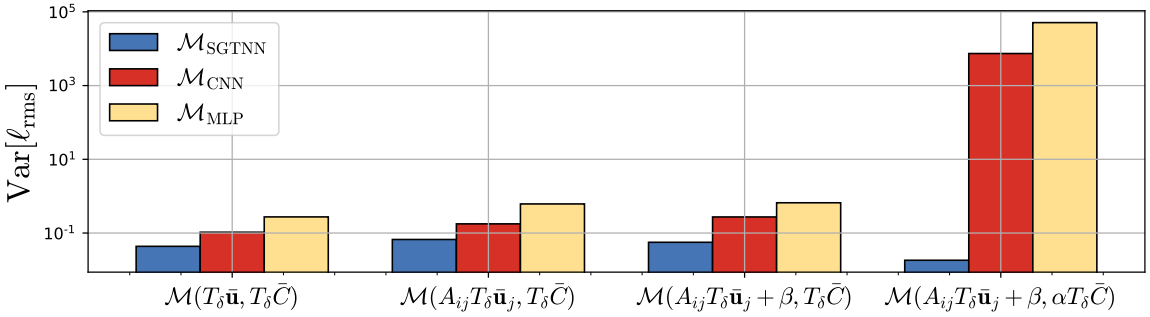

Figure 3.8: Variance of the NRMSE as the different invariances are accumulated (from left to right).

3.2. In Table 3.2, we evaluate the normalized root-mean-squared error (NRMSE, see 3.3.2) ℓ_{rms} expectation and variance of τ_C with uniformly sampled values of δ, λ, β and the 6 possible permutations on the testing dataset. As expected, the translation equivariance built-in convolutions lead to improved results for the related invariance, which has a variance 4 times smaller for $\mathcal{M}_{\text{CNN}}$ and 15 times smaller for $\mathcal{M}_{\text{SGTNN}}$ compared to the baseline $\mathcal{M}_{\text{MLP}}$. For the rotation invariance, data augmentation has helped by a relatively small margin compared to the other models. Still, note that variance for $\mathcal{M}_{\text{MLP}}$ is one order of magnitude smaller than $\mathcal{M}_{\text{CNN}}$ for the rotation invariance. Galilean invariance is verified for both $\mathcal{M}_{\text{MLP}}$ using the gradients of input quantities and $\mathcal{M}_{\text{SGTNN}}$ using the standardization. Finally, scalar concentration linearity is only verified by $\mathcal{M}_{\text{SGTNN}}$, and is not a property that can be implicitly discovered by the other models. We can see in Fig. 3.8 that translation is the most penalizing invariance for the $\mathcal{M}_{\text{SGTNN}}$, and special treatment beyond convolution equivariance would be beneficial.

3.3.2 Results

For benchmarking purposes, we consider two physical models in addition to the three NN-based architectures. In particular, we chose a functional $\mathcal{M}_{\text{DynSmag}}$ and a structural $\mathcal{M}_{\text{DynRG}}$ model, adapted to SGS scalar turbulence from 2.1.4.

The dynamic eddy diffusivity model (Moin et al., 1991) is an extension of the Smagorinsky model (Germano et al., 1991) expressed as

$$\mathcal{M}_{\text{DynSmag}} := \nabla \cdot \left(\underbrace{C_{\text{S}} \bar{\Delta}^2 |\bar{S}| \frac{\partial \bar{C}}{\partial x_i}}_{\phi_{\text{DynSmag}}} \right). \tag{3.28}$$

This model is commonly used in practice because it is stable when $C_{\text{S}} \in \mathbb{R}_+^*$, i.e. scalar variance transfers are always from large to small scale (forward).

The dynamic regularized gradient model (Balarac et al., 2013) can also be extended in a similar manner,

$$\mathcal{M}_{\text{DynRG}} := \nabla \cdot \left(\underbrace{C_{\text{G}} \bar{\Delta}^2 \bar{S}^{\ominus} \frac{\partial \bar{C}}{\partial x_i}}_{\phi_{\text{DynRG}}} \right). \tag{3.29}$$

This model is expected to be more stable than the simple gradient model since its strain-rate component only contains forward energy transfers.

Metrics. Most *a priori* evaluations in the literature characterize the performance of a model based on a structural metric which is defined by the error on the SGS term, and a functional metric that operates on the SGS scalar residual flux ϕ_C. In particular, it is common to define the *a priori* performance of a given model by its ability to reproduce the distribution of SGS scalar dissipation, i.e. the transfer of energy between resolved and subgrid scales, given by $\phi_C \cdot \nabla \bar{C}$. In this chapter however, we focus on the modeling of the SGS term $\tau_C := \nabla \cdot \phi_C$, and since the divergence operator is not invertible, our proposed models do not have access to the residual flux ϕ_C. Therefore, we perform an extensive study of the predicted SGS term (or structural performance) statistical properties. As discussed in (Meneveau and Katz, 2000), it is suggested that structural performance describes the short-term time evolution of a model and is of most importance to the understanding of local and instantaneous errors in a model. In this study, we evaluate three different types of metrics based on structural, statistical, and functional considerations.

Structural performance is measured using a normalized root-mean-squared error (NRMSE) and correlation coefficient $r(\mathcal{M})$ between the SGS term and the model prediction,

$$\ell_{\text{rms}}(\mathcal{M}) = \frac{\sqrt{\left\langle (\tau_C - \mathcal{M})^2 \right\rangle}}{\sigma(\tau_C)}, \tag{3.30}$$

$$r(\mathcal{M}) = \frac{\left\langle (\tau_C^{(i)} - \langle \tau_C \rangle)(\mathcal{M}^{(i)} - \langle \mathcal{M} \rangle) \right\rangle}{\sigma(\tau_C)\sigma(\mathcal{M})}. \tag{3.31}$$

	$\downarrow \ell_{\text{rms}}$	$\uparrow r$	$\downarrow D_{\text{JS}}$	$\downarrow I_\epsilon$
$\mathcal{M}_{\text{DynSmag}}$	0.940	0.360	0.289	0.177
$\mathcal{M}_{\text{DynRG}}$	0.875	0.497	0.354	**0.062**
$\mathcal{M}_{\text{MLP}}$	1.229	0.136	**0.045**	0.089
$\mathcal{M}_{\text{CNN}}$	0.872	0.560	0.090	0.101
$\mathcal{M}_{\text{SGTNN}}$	**0.835**	**0.612**	0.110	0.118

Table 3.3: A priori evaluation of the SGS term in developed turbulence regime using testing data with the metrics described in 3.3.2.

Statistical performance evaluates how consistent the PDF of the prediction is w.r.t. the true one. The Jensen-Shannon (JS) distance D_{JS} (Endres and Schindelin, 2003) is a metric that measures the similarity between two probability distributions defined on the same probability space and built on the Kullback-Leiber (KL) divergence D_{KL}, or relative entropy,

$$D_{\text{KL}}(P_{\tau_C}||P_{\mathcal{M}}) = \sum_{i=1}^{N} P_{\tau_C}^{(i)} \ln\left(\frac{P_{\tau_C}^{(i)}}{P_{\mathcal{M}}^{(i)}},\right) \tag{3.32}$$

$$D_{\text{JS}}(P_{\tau_C}||P_{\mathcal{M}}) = \sqrt{\frac{1}{2}D_{\text{KL}}(P_{\tau_C}||D_{\text{A}}) + \frac{1}{2}D_{\text{KL}}(P_{\mathcal{M}}||D_{\text{A}})}, \tag{3.33}$$

$$D_{\text{A}} = \frac{1}{2}(P_{\tau_C} + P_{\mathcal{M}}). \tag{3.34}$$

Functional performance is systematically defined as the error on the integral dissipation I_ϵ, which can be obtained from the divergence of the SGS residual flux in a periodic domain V as

$$\begin{aligned} I_\epsilon(\mathcal{M}) &= -\int \phi_{\mathcal{M}} \cdot \nabla\bar{C} \, \mathrm{d}V + \int \phi_C \cdot \nabla\bar{C} \, \mathrm{d}V \\ &= -\int \bar{C}\nabla \cdot \phi_{\mathcal{M}} \, \mathrm{d}V + \int \bar{C}\nabla \cdot \phi_C \, \mathrm{d}V \\ &= -\int \bar{C}\mathcal{M} \, \mathrm{d}V + \int \bar{C}\tau_C \, \mathrm{d}V. \end{aligned} \tag{3.35}$$

3.3.2.1 *A priori* – developed turbulence regime

We first perform the evaluation on the developed turbulence regime of the testing dataset (see Fig. 3.6) which is the closest to the training data, and where NN-based models are expected to perform the best. It is important to note that some metrics such as the NRMSE and correlation coefficient strongly depend on the type of kernel used to filter out the small scales features (Fabre and Balarac, 2011). In Table 3.3, we show the results of the *a priori* metrics with the same range of values as the training data, i.e. $\delta = 0$, $\beta = 0$ and $\lambda = 1$. The invariant model $\mathcal{M}_{\text{SGTNN}}$ gives the most consistent structural results compared to the DNS. In particular, we observe a substantial improvement on

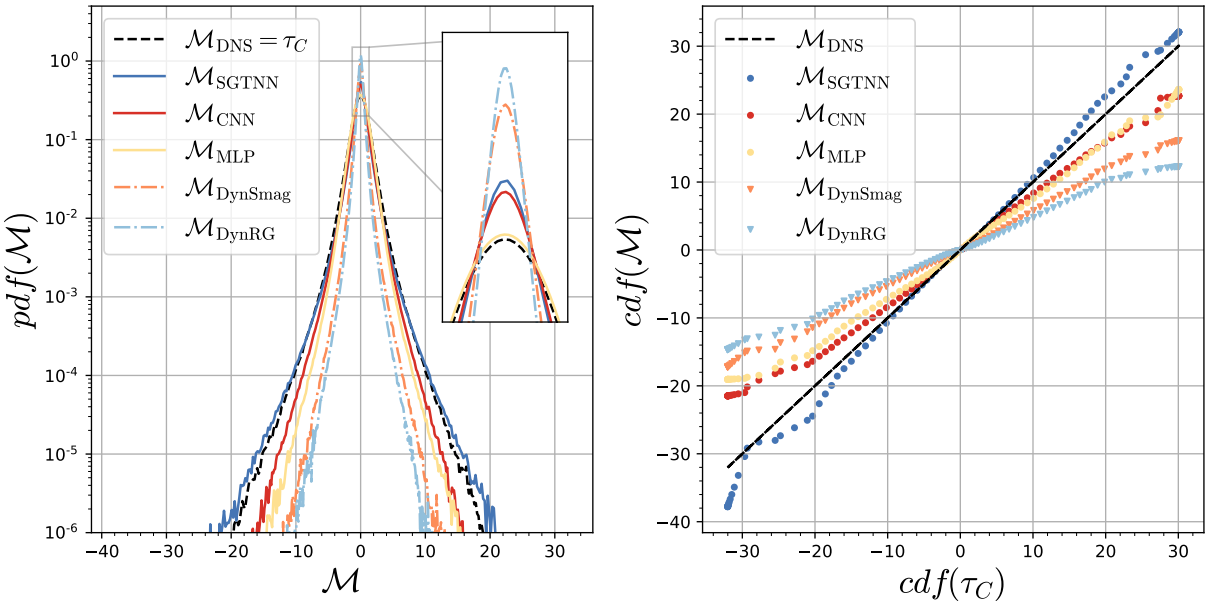

Figure 3.9: Probability distribution function (left) and quantiles-quantiles (right) of the SGS term in developed turbulence regime over the entire testing dataset.

ℓ_{rms} and r compared to $\mathcal{M}_{\mathrm{CNN}}$. On the statistical metric, it is clear from the inset in Fig. 3.9 that $\mathcal{M}_{\mathrm{MLP}}$ is better at reproducing mean values (around $\tau_C = 0$), which tends to lead to smaller distance D_{JS}. However, $\mathcal{M}_{\mathrm{SGTNN}}$ is more accurate on the tails of the distribution compared to the other models. This is particularly emphasized by the quantiles-quantiles (QQ) plot, which draws the theoretical quantiles from DNS on the x-axis and the predicted ones on the y-axis. Again, it is found that $\mathcal{M}_{\mathrm{SGTNN}}$ is in best agreement with the theoretical quantiles, for which a perfectly reproduced distribution would fit a straight line. Recall that the scalar concentration linearity shows the most noticeable impact on *a priori* performance. In this regime, scalar variance remains almost constant, and NN-models are thus expected to perform optimally in terms of their loss function $\mathcal{L}$. We see however that $\mathcal{M}_{\mathrm{MLP}}$ has poor structural performance, which could be explained by the fact that it is minimizing error on the residual flux rather than the SGS term itself. The next two regimes will test the ability of the NN-based models to generalize to flows that were not part of the training data.

3.3.2.2 *A priori* – transitional regime

In this regime, we look at a forced flow that transitions from a newly initialized to a turbulent state (see Fig. 3.6). In the *a priori* tests, we consider both velocities and scalar concentration in a transitional regime. Fig. 3.10 shows that ℓ_{rms} is large for the first sample predictions of NN-based models but rapidly decreases to a similar magnitude than in developed turbulence. The correlation given by r is also quite low during the first iterations but quickly reaches the statistical mean at $t \approx 10$, as well as metric D_{JS} from

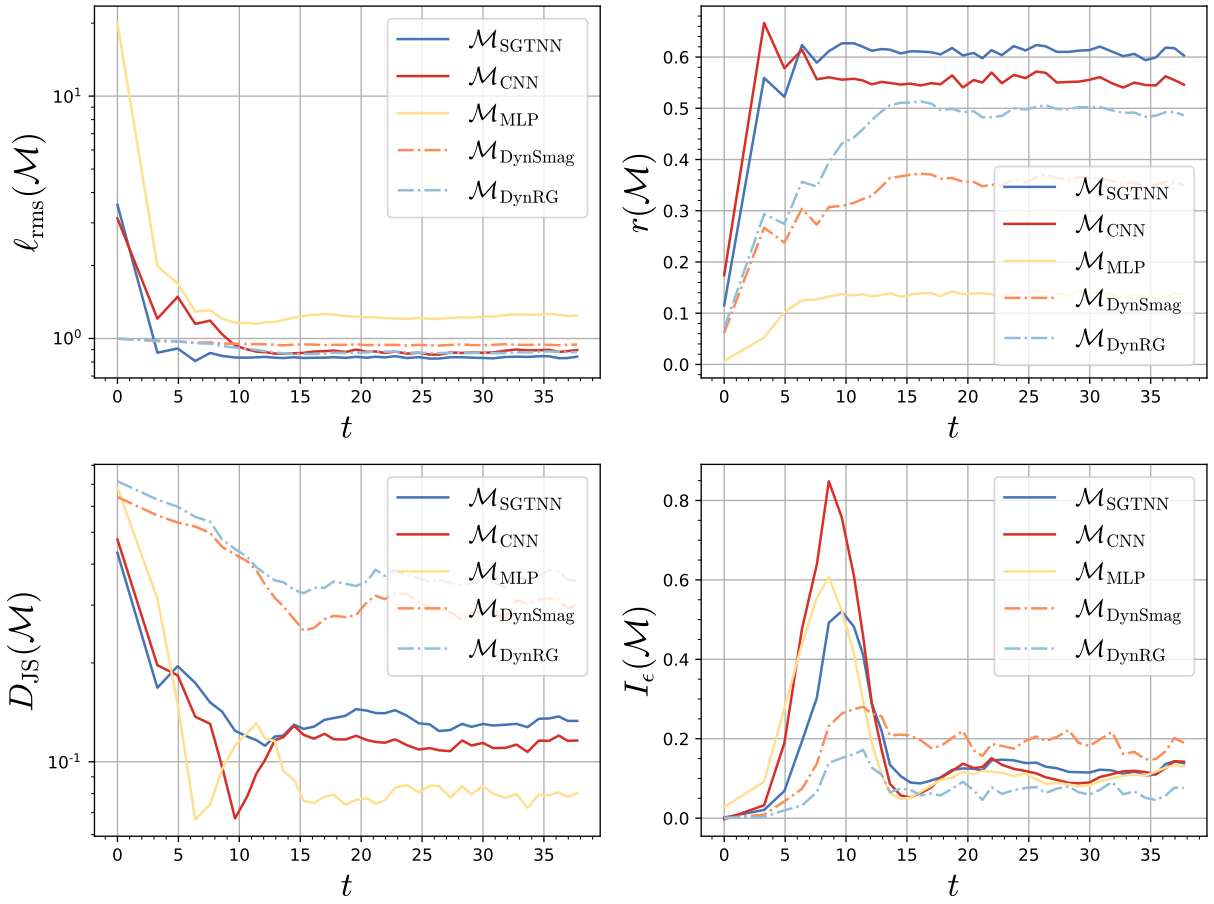

Figure 3.10: Evaluation of the different metrics with respect to time in transitional regime. NRMSE (top left), correlation coefficient (top right), Jensen-Shannon distance (bottom left) and integral dissipation error (bottom right).

which a large distance is observed. The decrease in performance temporally coincides with the unphysical spin-up phase, and the prediction error gradually improves as the regime gets closer to the training one. We note that these models could benefit from a loss regularization or constraint on the transfers, which would better capture the transitional phases. e.g., Vollant et al. (2017) and Charalampopoulos and Sapsis (2022) proposed to build a model based on multi-objective minimization, or regularization of the structural term with a functional term based on scalar variance and energy transfers in three and two dimensions, respectively.

3.3.2.3 *A priori* – decaying regime

In the last *a priori* regime, we remove the scalar forcing, i.e. $F_C = 0$ while maintaining the velocity field in turbulent motion (again, see Fig. 3.6). This results in a decay of the scalar concentration within the domain. This regime is a difficult case for the NN-based models presented in this chapter and asses their capacity to generalize to a different dynamic of the flow, which has not been seen in the training data. Both $\mathcal{M}_{\mathrm{MLP}}$

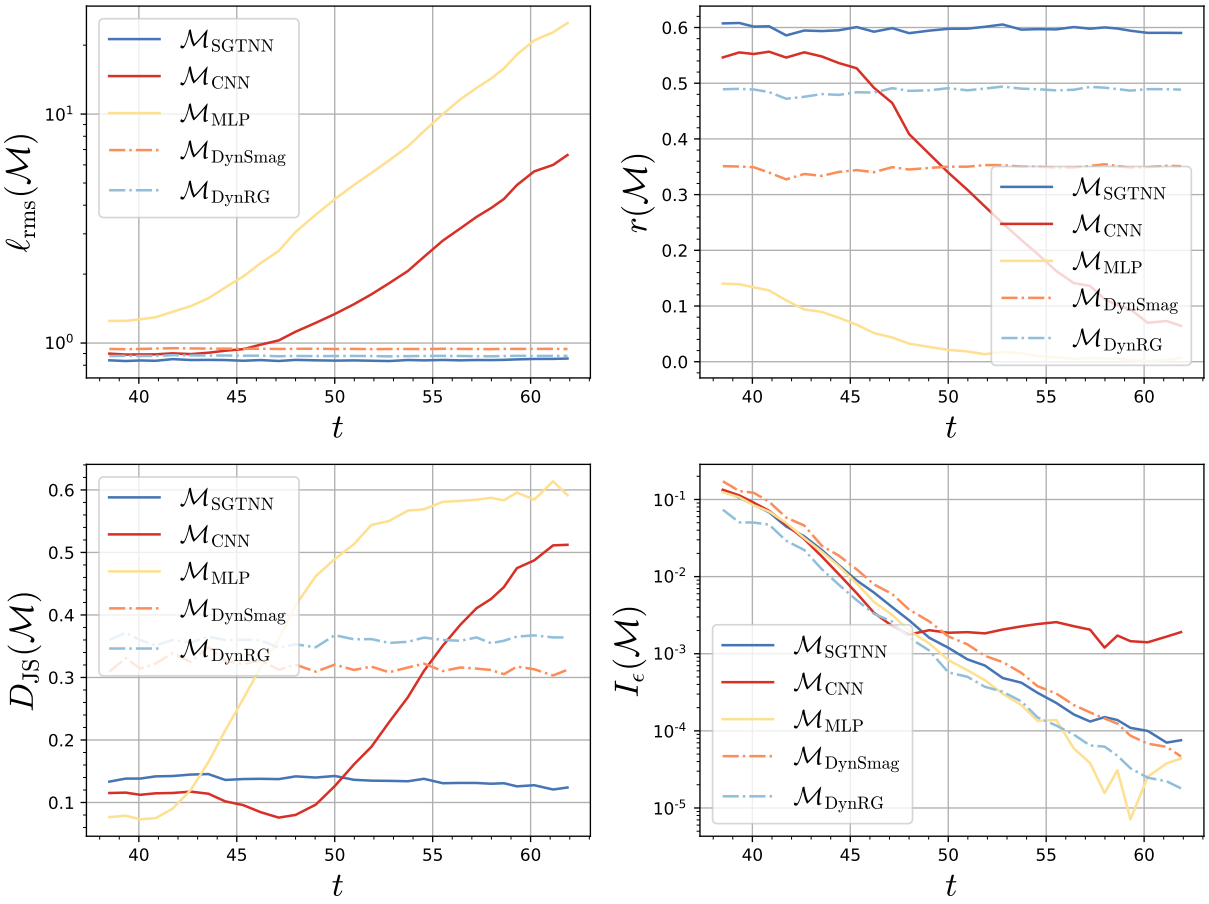

Figure 3.11: Evaluation of the different metrics with respect to time in decaying regime. NRMSE (top left), correlation coefficient (top right), Jensen-Shannon distance (bottom left) and integral dissipation error (bottom right).

and $\mathcal{M}_{\text{CNN}}$ see their structural and statistical performance drastically decreasing after a short time, at $t \approx 40$, as see in Fig. 3.11 for NRMSE ℓ_{rms}, correlation r and statistical distance D_{JS}. Interestingly, we observe that $\mathcal{M}_{\text{CNN}}$ is producing an error on the integral dissipation which does not tend toward zero compared to the other models. Overall, when comparing the NN-based models, we can say that the imposed invariances act as physical regularizers since they help the generalization while ensuring short-time coherent behavior across different regimes.

Simulations. We run new simulations using the NN-based models to compute the SGS term at every iteration. These *a posteriori* tests are complementary to the *a priori* tests because they show the behavior and time evolution of the SGS models in practice. In the following tests, we use a hybrid DNS-LES approach in order to isolate the scalar concentration modeling from other sources of error. To achieve this, we keep the velocity fields at DNS resolution, i.e. 512^3, while the scalar concentration fields of the different models evolve at the same resolution as the filtered training data, i.e. 64^3. At every sub-step in the time integration, the velocity fields are extrapolated from the DNS to the LES

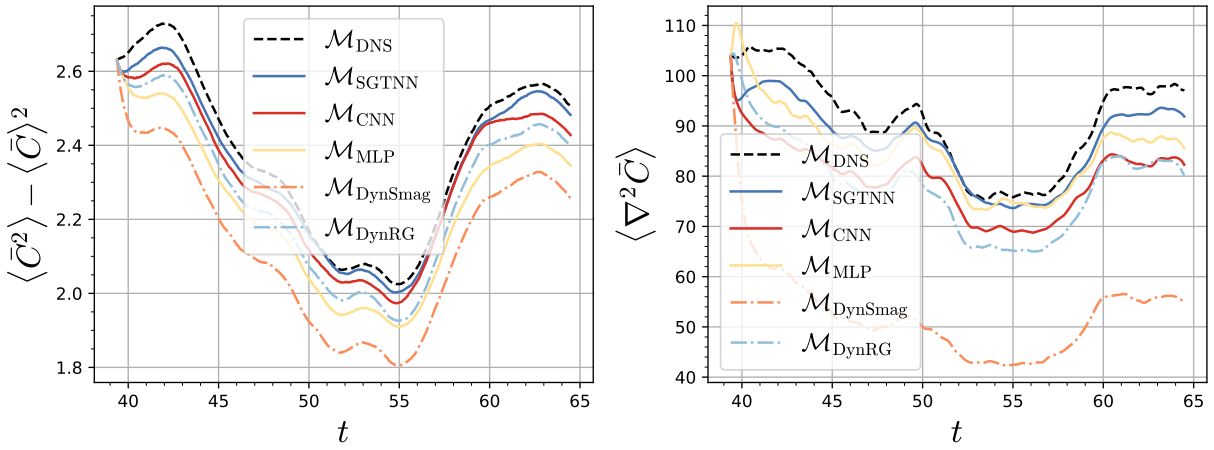

Figure 3.12: Integrated statistics of simulation in developed turbulence regime starting from the training data: scalar variance (left) and resolved scalar enstrophy (right).

grid in spectral space using a cut-off filter. Note that only data on the LES grid are used to obtain SGS term predictions from the different models. The advantage of this approach is that there is no modeling error on the velocity field used in the scalar equation and SGS prediction. Thus, when the modeled scalar concentrations are compared with the filtered DNS, the difference will only be due to the SGS term. Now, we look at the temporal evolution of different statistical metrics such as the filtered scalar variance $\langle \bar{C}^2 \rangle - \langle \bar{C} \rangle^2$ and the resolved scalar enstrophy $\langle \nabla^2 \bar{C} \rangle$ which is particularly interesting to characterize the energy transfers in the smallest scales (San et al., 2015), from which the SGS model is the most important. For each regime, we also plot the scalar energy spectrum at the intermediate and final simulation time.

3.3.2.4 *A posteriori* – developed turbulence regime

In the first two simulations, we start in a developed turbulence regime after 20000 temporal iterations from the training data (see Fig. 3.5) and testing data (see Fig. 3.6) respectively with the same velocity and scalar concentration forcing as those used in the datasets. Using the same starting point as the training data is useful in order to consider the accumulation of errors during the simulation since the models only learned from a finite number of sub-samples of the exact simulation. We observe in Fig. 3.12 that the scalar variance (left) produced by $\mathcal{M}_{\text{SGTNN}}$ and $\mathcal{M}_{\text{CNN}}$ are the closest to the DNS except after long integration time at $t \approx 60$ where $\mathcal{M}_{\text{CNN}}$ starts accumulating error. The behavior of the model on the smallest scales is shown by the resolved scalar enstrophy (right). Here, $\mathcal{M}_{\text{SGTNN}}$ and $\mathcal{M}_{\text{MLP}}$ are the most consistent, while $\mathcal{M}_{\text{DynSmag}}$ is too diffusive, as expected. One can look more closely at the energy spectra in Fig. 3.13 that the model with invariances is the most accurate on the smallest wavenumbers, which is particularly visible at $k \geq 10$. It is interesting to note that we can draw similar conclusions in this

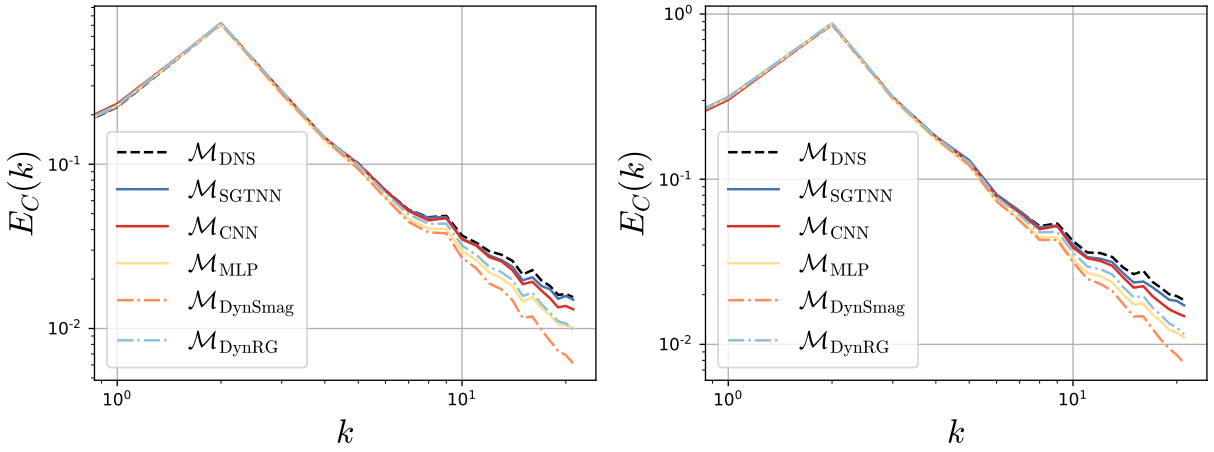

Figure 3.13: Scalar energy spectrum of simulation in developed turbulence regime starting from the training data at $t \approx 50$ (left) and $t \approx 65$ (right).

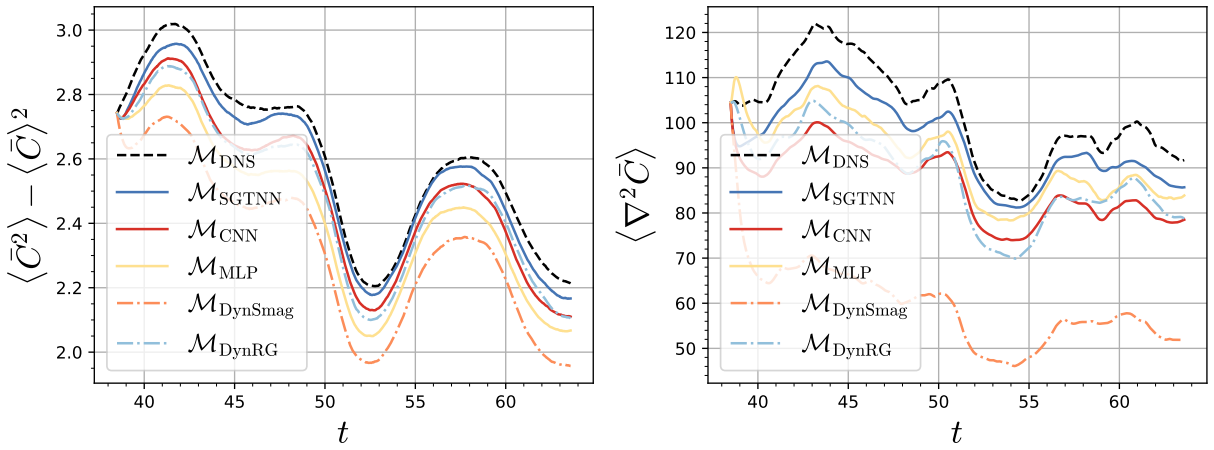

Figure 3.14: Integrated statistics of simulation in developed turbulence regime starting from the testing data: scalar variance (left) and resolved scalar enstrophy (right).

evaluation compared to the *a priori* structural results, i.e. $\mathcal{M}_{\mathrm{SGTNN}}$ slightly improves from $\mathcal{M}_{\mathrm{CNN}}$ which also performs better than $\mathcal{M}_{\mathrm{MLP}}$. Now, the same conclusions can be drawn in the developed turbulence regime with the testing data starting point, as depicted in Figs. 3.14 and 3.15. We can already see that a CNN without physical constraints is not able to reproduce the dynamics of the SGS term in statistically similar conditions to those of the training data. Indeed, it still performs at the same accuracy for the scalar variance but worse than some physical models for the resolved scalar enstrophy. The effect of the physical invariances is shown to be important and gives the ability to get the best performances from the model.

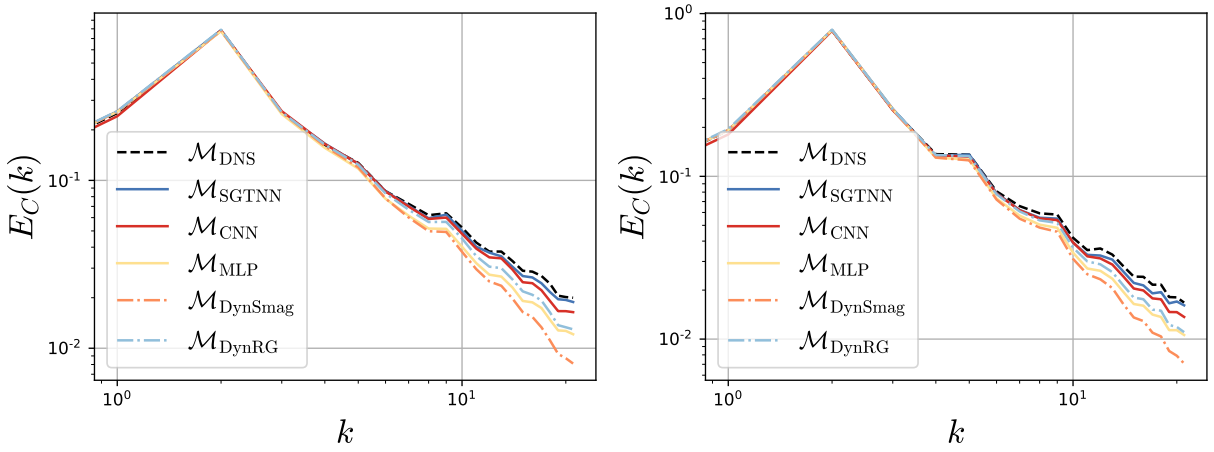

Figure 3.15: Scalar energy spectrum of simulation in developed turbulence regime starting from the testing data at $t \approx 50$ (left) and $t \approx 65$ (right).

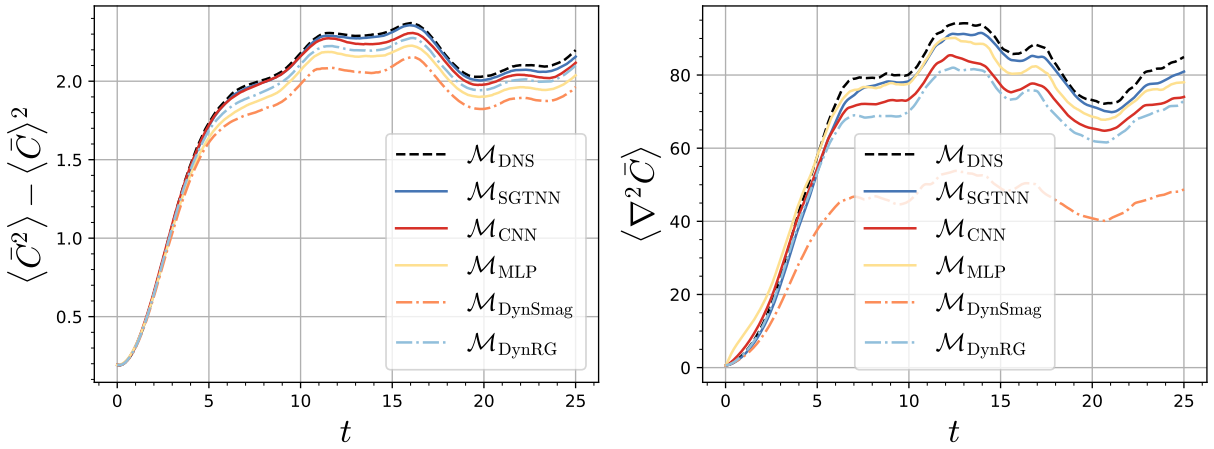

Figure 3.16: Integrated statistics of simulation in scalar transition regime starting from the testing data: scalar variance (left) and resolved scalar enstrophy (right).

3.3.2.5 *A posteriori* – scalar transition regime

The transitional regime is already testing the extrapolation capabilities of the models. In this regime, we start from a velocity field already in turbulent motion and a scalar concentration initialized at large scales and forced with the previously described scheme. The difficulty of this regime resides in the transition of the scalar field to turbulent advection, which then comes down to a developed regime after $t \approx 15$. Both scalar variance and scalar enstrophy are well reproduced by $\mathcal{M}_{\mathrm{SGTNN}}$ (see Fig. 3.16) during and after the transition. The hypothesis drawn from the *a priori* evaluation of the similar regime seems to correlate with the *a posteriori* statistics, where both $\mathcal{M}_{\mathrm{CNN}}$ and $\mathcal{M}_{\mathrm{MLP}}$ produce a relatively large error during the first transitional iterations, while the invariant NN model is more consistent. Still, Fig. 3.17 indicates that the spectrum produced by the baseline NN-based models maintained a performance similar to the previous regime.

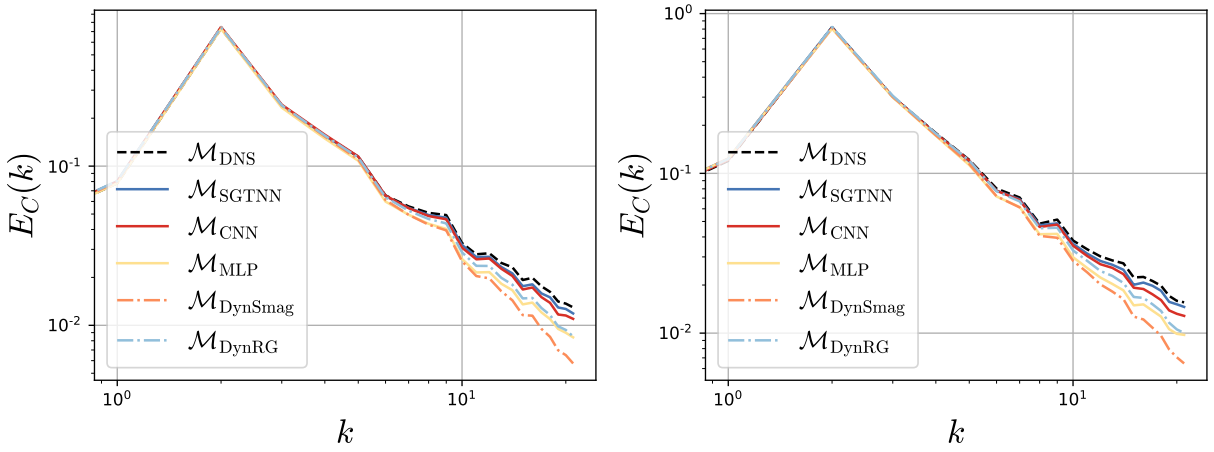

Figure 3.17: Scalar energy spectrum of simulation in scalar transition regime starting from the testing data at $t \approx 6$ (left) and $t \approx 25$ (right).

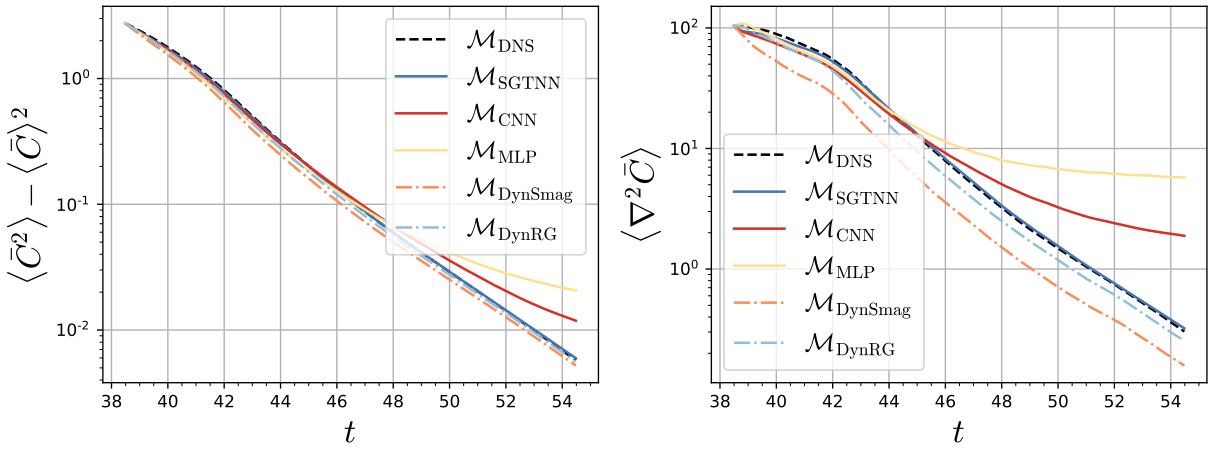

Figure 3.18: Integrated statistics of simulation in full decay regime starting from the testing data: scalar variance (left) and resolved scalar enstrophy (right).

Interestingly, the mismatch does not increase a lot between the middle of the transition and the end of the transition, which means that the models without invariances are still able to generalize well to this regime.

3.3.2.6 *A posteriori* – full decay regime

In the *a priori* evaluation, it was clear that $\mathcal{M}_{\text{SGTNN}}$ gave the best performance in the scalar decay regime while the other NN-based models were not able to generalize. In this *a posteriori* regime, we go one step further and completely remove the source terms for both velocity and scalar concentration fields, which results in a full decay. Here, we also observe in Fig. 3.18 that the scalar variance and resolved scalar enstrophy stopped decreasing for $\mathcal{M}_{\text{CNN}}$ and $\mathcal{M}_{\text{MLP}}$, which indicates that both models are still producing energy when the scalar concentration is almost completely decayed. We also see in the

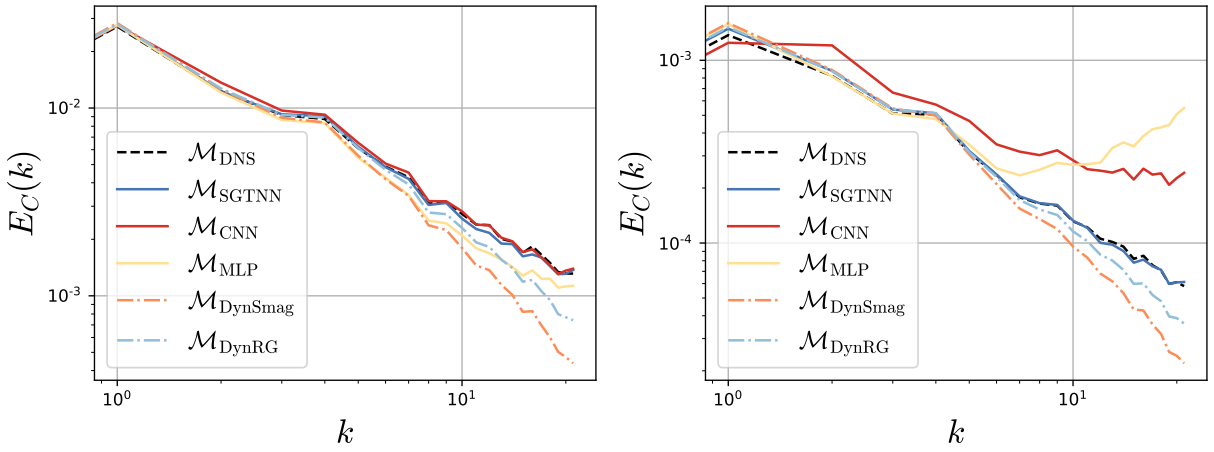

Figure 3.19: Scalar energy spectrum of simulation in full decay regime starting from the testing data at $t \approx 46$ (left) and $t \approx 55$ (right).

spectrum (Fig. 3.19) that the NN-based models without invariances are not able to diffuse the small scales of the SGS term which incrementally lead to a simulation blowup at $t \approx 55$. $\mathcal{M}_{\mathrm{SGTNN}}$ however is stable and still in excellent agreement with the DNS and out-performs significantly the physical models, even in this regime that has a different dynamics not visible in the training data.

Generalization capabilities. The *a posteriori* regimes discussed above are statistically different from the data used to train the NN-based models. To evaluate the robustness of the models and their ability to extrapolate to different regimes, we compute a locally normalized largest absolute error (ℓ_∞) of the *a posteriori* metrics, namely scalar variance and resolved scalar enstrophy, during their time evolution,

$$\ell_\infty(m, \mathring{m}) = \max_i \left\{ \left| \frac{1}{m_i}(m_i - \mathring{m}_i) \right| \right\} \tag{3.36}$$

where m and $\mathring{m}$ refer to metrics computed from the DNS and LES simulations, respectively. The results are shown in Table 3.4 where we put the emphasis on the error increase compared to the reference (or base) given by the developed turbulence regime starting from training data for each model. The models without invariances show increasing errors in scalar transition and full decay. This increase is particularly visible in the resolved scalar enstrophy, for which the error of $\mathcal{M}_{\mathrm{MLP}}$ increases to one order of magnitude higher than $\mathcal{M}_{\mathrm{CNN}}$. $\mathcal{M}_{\mathrm{SGTNN}}$ is consistent across the different regimes, with an error than does not increase more than 1.5 times compared to the base regime. Note that the scalar variance error decreases in the scalar transition regime for $\mathcal{M}_{\mathrm{SGTNN}}$ and $\mathcal{M}_{\mathrm{CNN}}$, which can be explained by the fact that this regime does not exhibit complex dynamics at large scale. The increase of scalar variance error is only visible in full decay, driven by the small-scale numerical instabilities of the models, which are then propagated to the largest scales. For $\mathcal{M}_{\mathrm{SGTNN}}$, the largest error increase of resolved scalar enstrophy occurs in the

$\langle\bar{C}^2\rangle - \langle\bar{C}\rangle^2$	ℓ_∞ (base)	ℓ_∞	ℓ_∞	ℓ_∞
$\mathcal{M}_{\text{MLP}}$	0.072 (×1)	0.084 (×1.168)	0.073 (×1.017)	2.598 (×36.146)
$\mathcal{M}_{\text{CNN}}$	0.040 (×1)	0.058 (×1.427)	0.039 (×0.959)	1.064 (×26.247)
$\mathcal{M}_{\text{SGTNN}}$	**0.026** (×1)	**0.027** (×**1.009**)	**0.019** (×**0.727**)	**0.038** (×**1.440**)
$\langle\nabla^2\bar{C}\rangle$				
$\mathcal{M}_{\text{MLP}}$	0.118 (×1)	0.127 (×**1.077**)	2.628 (×22.264)	17.968 (×152.239)
$\mathcal{M}_{\text{CNN}}$	0.175 (×1)	0.189 (×1.082)	0.494 (×2.827)	5.215 (×29.860)
$\mathcal{M}_{\text{SGTNN}}$	**0.082** (×1)	**0.091** (×1.104)	**0.112** (×**1.357**)	**0.093** (×**1.127**)
	Developed (train)	Developed (test)	Scalar transition	Full decay

Table 3.4: Locally normalized maximum absolute error ℓ_∞ on scalar variance (first three rows) and resolved scalar enstrophy (last three rows) of the different NN-based models in each *a posteriori* regime. Relative error compared to the base regime, i.e. developed turbulence starting from training data is shown as ℓ_∞/ℓ_∞ (base).

scalar transition regime, which is not surprising and can be due to the unrealistic nature of this regime.

3.4 Summary and discussion

In this chapter, we described a new NN-based SGS model that explicitly implements invariances for the advection-diffusion of a scalar concentration driven by an incompressible turbulent flow. The model is shown to predict a realistic behavior w.r.t. different forcing regimes, while classical NN-based models produce energy transfers when the scalar concentration is completely diffused. In addition to the generalization capabilities, the model also outperforms physical models and improves on NN-based models that do not embed physical knowledge. Due to the small-scale statistical universality of turbulence (Schumacher et al., 2014), it is quite important for a model to be independent of the specific flow regime. We have demonstrated some degree of extrapolation to different regimes, but limitations are likely to exist for regimes that drastically differ from the training data. We can note for example domains with closed geometries such as wall-bounded flows or regimes with different numerical parameters that impact the dynamics of the flow. These directions have been explored, i.e. many have proposed generalization schemes for different Reynolds numbers (Guan et al., 2022a; Zhu et al., 2021). While this could be explored in future work, a more challenging application is related to active scalars, such as temperature or reactive chemicals that directly interact with the momentum equation, and should probably be modeled together. Numerically, the ideal setting of spectral discretization is useful for planetary or stellar applications with spherical do-

mains but has serious limitations with respect to the geometry of the domain. Moving to stencil-based numerical methods such as finite volumes or finite differences constitutes a natural next exploratory step that comes with difficulties related to the separation of physical quantities and discretization errors introduced by the chosen scheme. We note that most ML frameworks are developed in Python, while large-scale models are based on high-performance languages such as Fortran or C. In this work, we used a simple wrapper that compiles the Python code into a library that can be called directly from its Foreign Function Interface (FFI). While this can be a temporary solution, the recent trend in ML-based physics has led to the development of new architectures that bridge the gap between ML models and HPC simulations (Partee et al., 2022). At this point, we believe that NN-based SGS modeling has significantly improved over classical baselines, and while data generation remains costly due to the complexity of DNS simulations, data availability from community datasets and model generalizability are slowly proving that this change in direction is possible.

Chapter 4

A posteriori learning for subgrid models

In this chapter, we describe a fully data-driven learning scheme that builds on the dynamical nature of differential equations, with an application to two-dimensional barotropic quasi-geostrophic turbulence. While the inductive biases presented in the previous chapter are powerful additions to NN-based models, the ability to formulate the invariance in an analytical form is required. The scheme can, however, reproduce complex dynamics that are not necessarily expressed explicitly. The trained models are shown to be long-term stable and outperform state-of-the-art baselines. Still, some limitations currently restrict their application to small solvers. These are discussed and potential solutions are proposed.

Some results presented in this chapter have been published *in Frezat et al. (2022).*

Contents

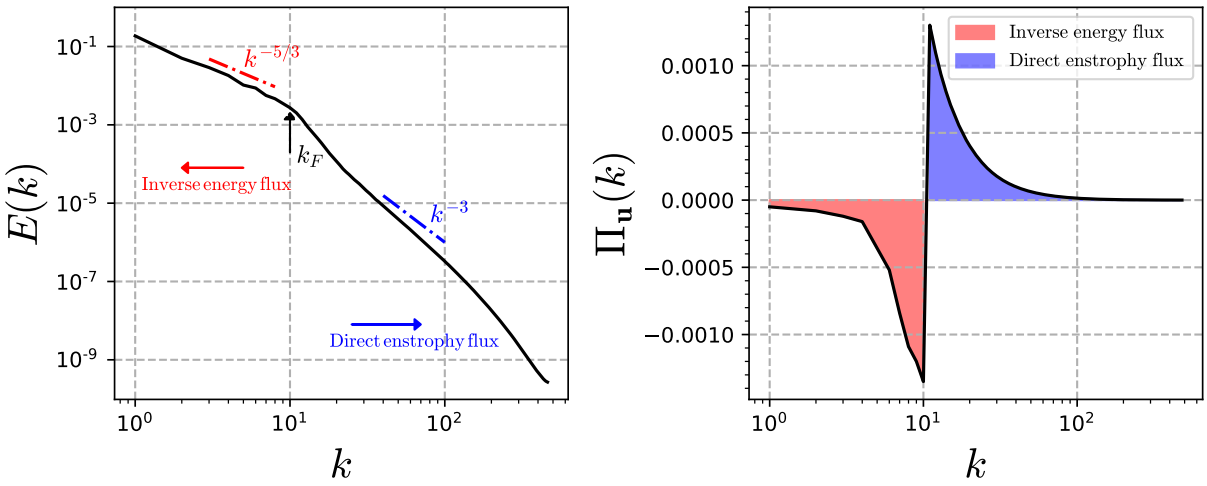

Figure 4.1: Energy spectrum in a direct numerical simulation of a two-dimensional flow (left) exhibiting the dual cascade with characteristic slopes and corresponding energy flux (right). Inverse energy flux is observed for wavenumbers smaller than the forcing scale $k < k_F$ and a direct enstrophy flux at $k > k_F$.

4.1 Representing complex dynamical phenomena

The physics that drives turbulent motion in oceans and atmospheres is complex. We have had long-time observations of their behavior and already some knowledge is known and can be explicitly expressed. For example, we have explored frame invariances in chapter 3, which consists in properties that a flow is recognized to follow. While this is important for a numerical model to always agree with analytical invariances, this does not always result in the desired behavior of the model on a "higher level" point-of-view, which is often either based on long-term or statistical metrics. In expansive numerical simulations, it is important for a model to be stable, which is difficult to define as a mathematical expression. It is possible to derive conditions for which models are found to be stable, for example, turbulence models with positive eddy viscosities (Meneveau et al., 1996; Trias et al., 2015) have been shown to fulfill this goal, at the cost of accuracy. This trade-off has been pushed by incremental modifications to these models, while still holding the stability condition, but they remain far from optimal, even when combined with NNs (Sarghini et al., 2003). Again, with the recent advances in deep learning, new turbulence models outperforming stable physical models have been designed (Vinuesa and Brunton, 2022). However, in some conditions, these models are particularly unstable and difficult to regularize. This phenomenon is stronger in two-dimensional flows (Maulik et al., 2019) for which the inverse energy cascade is dominating, and affects the models ability to reproduce transfers from small to large scales, also referred to as backscatter (Carati et al., 1995).

4.1.1 Backscatter in two-dimensional flows

In three-dimensional flows, the energy cascade hypothesis is directly related to the turbulent kinetic energy generated at large-scale and transferred to smaller scales until molecular viscous dissipation. While this is true in a statistical sense, the transfers are not locally behaving in a unidirectional way. The turbulent dissipation at the smallest scales is equivalent to the difference between forward-scatter, coinciding with the energy cascade, and backscatter in which energy is transferred from the small scales back to the large scales (Lesieur and Metais, 1996). Historically, developing SGS models that account for backscatter is a challenging task (Liu et al., 2011; Piomelli et al., 1991; Schumann, 1995). Indeed, an overprediction of backscatter that can not be compensated by eddy-viscosity will lead to an accumulation of small-scale energy causing simulations to become numerically unstable. In two-dimensional flows, we observe a dual cascade composed of "forward" enstrophy and "inverse" energy, again in a statistical sense (see Fig. 4.1). As a consequence, a large number of subgrid models have been proposed in particular for geophysical flows (see Danilov et al. (2019) for a review) with well-documented configurations and performance metrics (Graham and Ringler, 2013). To study scale interactions in the dual cascade regime, we can look at the temporal evolution of the enstrophy $Z(k)$ and kinetic energy $E(k)$ spectrum in spectral space,

$$\frac{\partial Z(k)}{\partial t} = T_\omega(k) + F_\omega(k) - D_\omega(k) \tag{4.1}$$

$$\frac{\partial E(k)}{\partial t} = T_{\mathbf{u}}(k) + F_{\mathbf{u}}(k) - D_{\mathbf{u}}(k) \tag{4.2}$$

where and the different terms of the right-hand side are related to various effects: dissipation D, external source F, and transfers between scales T. This first term writes

$$T_\omega(k) = \int_{|\mathbf{k}|=k} \Re\{\hat{\omega}^*(\mathbf{k})\hat{N}_\omega(\mathbf{k})\}\,\mathrm{d}S(\mathbf{k}) \tag{4.3}$$

$$T_{\mathbf{u}}(k) = \int_{|\mathbf{k}|=k} \Re\{\hat{\mathbf{u}}^*(\mathbf{k})\hat{N}_{\mathbf{u}}(\mathbf{k})\}\,\mathrm{d}S(\mathbf{k}) \tag{4.4}$$

where forward and backward transfers are indicated by the sign of T and hold for both enstrophy and kinetic energy (Boffetta and Ecke, 2012). We also define the enstrophy and kinetic energy fluxes for a wavenumber sphere of radius k_s,

$$\Pi_\omega(k_s) = -\int_0^{k_s} T_\omega(k')\,\mathrm{d}k' \tag{4.5}$$

$$\Pi_{\mathbf{u}}(k_s) = -\int_0^{k_s} T_{\mathbf{u}}(k')\,\mathrm{d}k' \tag{4.6}$$

with

$$\frac{d\Pi_\omega(k)}{d\Pi_{\mathbf{u}}(k)} = k^2. \tag{4.7}$$

Again, were are interested in modeling the unresolved dynamics of the two-dimensional NS equations on coarse grid $\bar{\Omega}$, defined in vorticity formulation such that

$$\begin{cases} \dfrac{\partial \omega}{\partial t} + N_\omega = \nu \nabla^2 \omega + F_\omega, & \omega \in \Omega \\ \dfrac{\partial \bar{\omega}}{\partial t} + N_{\bar{\omega}} = \nu \nabla^2 \bar{\omega} + F_{\bar{\omega}} + \underbrace{N_{\bar{\omega}} - \bar{N}_\omega}_{\tau_\omega}, & \bar{\omega} \in \bar{\Omega} \end{cases} \tag{4.8}$$

where F_ω is a time-dependent forcing term and the projection from fine to coarse grid is defined by a spatial operator $\mathcal{T} : \Omega \to \bar{\Omega}$. On coarse grid $\bar{\Omega}$, the equation can be solved except for τ_ω which depends on fine grid variables $\mathbf{u}$ and ω. Alternatively, the SGS term τ_ω can be expressed as the divergence of a flux,

$$\tau_\omega = \nabla \cdot (\bar{\mathbf{u}}\,\bar{\omega} - \overline{\mathbf{u}\,\omega}) \tag{4.9}$$

Using (2.60) only involves reduced vorticity $\bar{\omega}$ and streamfunction $\bar{\psi}$, since velocities can be obtained from the latter. In this case, we are interested in the following inverse problem,

$$\tau_\omega = \partial_x \bar{\psi} \partial_y \bar{\omega} - \partial_y \bar{\psi} \partial_x \bar{\omega} - \overline{\partial_x \psi \partial_y \omega + \partial_y \psi \partial_x \omega}. \tag{4.10}$$

Note that choosing to model the SGS term τ_ω instead of the SGS flux divergence is reducing the complexity of the NN prediction to a scalar field but it restricts the accessible quantities. When we study the effect of the SGS model on scale interactions (also see Fig. 4.2), it is possible to decompose the transfers into a resolved and a modeled part, i.e.,

$$T_\omega(k) = \int_{|\mathbf{k}|=k} \Re\{\underbrace{\hat{\bar{\omega}}^*(\mathbf{k}) \hat{N}_{\bar{\omega}}(\mathbf{k})}_{\text{resolved}} - \underbrace{\hat{\bar{\omega}}^*(\mathbf{k}) \hat{\tau}_\omega(\mathbf{k})}_{\text{modeled}}\} \, \mathrm{d}S(\mathbf{k}). \tag{4.11}$$

Equivalent quantities for the resolved enstrophy equilibrium in physical space can be expressed as

$$\frac{\partial Z(\mathbf{x})}{\partial t} = \bar{\omega} \nu \nabla^2 \bar{\omega} + \bar{\omega} F_{\bar{\omega}} - \bar{\omega} N_{\bar{\omega}} + \bar{\omega} \tau_\omega. \tag{4.12}$$

The last term can be further decomposed into a diffusion and a transfer term $T_\omega(\mathbf{x})$, from which the sign is also an indicator of forward and backward transfers. Using (4.9), it writes

$$\bar{\omega} \tau_\omega = \underbrace{\nabla \cdot (\bar{\omega}(\bar{\mathbf{u}}\,\bar{\omega} - \overline{\mathbf{u}\,\omega}))}_{\text{diffusion}} - \underbrace{(\bar{\mathbf{u}}\,\bar{\omega} - \overline{\mathbf{u}\,\omega}) \cdot \nabla \bar{\omega}}_{\text{transfers } T_\omega(\mathbf{x})}. \tag{4.13}$$

While $T_\omega(\mathbf{x})$ can not be computed using the modeling approach (4.10), it is still possible to study its global effect,

$$\int \bar{\omega} \tau_\omega \, \mathrm{d}S = - \int T_\omega \, \mathrm{d}S \tag{4.14}$$

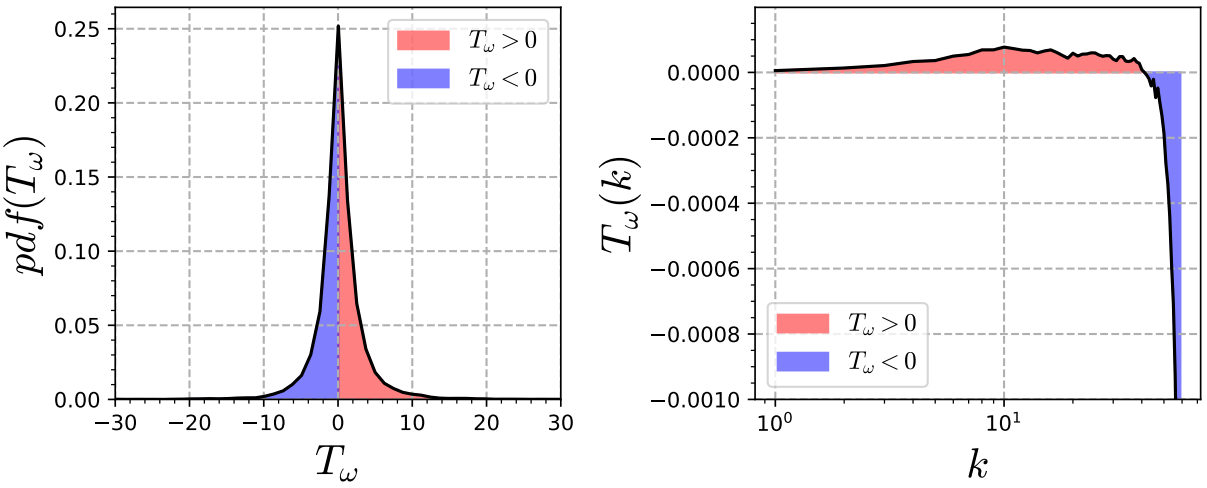

Figure 4.2: Enstrophy transfers in physical space (left) and spectral space (right) for the modeled part (contributions from τ_ω). Forward scatter corresponds to $T_\omega < 0$ and backscatter is represented in $T_\omega < 0$.

4.1.2 Trade-off solutions

As discussed in the previous section, models that mispredict backscatter are often subject to numerical instabilities by the accumulation of small-scale energy. The first example is the spatial averaging performed on the dynamical procedure (Germano et al., 1991) of physical models. Recall from 2.1.4 that this method can extract a scale-dependent coefficient field $C_{\text{sgs}} \in \mathbb{R}^2$ for two-dimensional flows. It is well known that using this spatially-varying coefficient lead to significant portions of negative eddy viscosities, destabilizing the simulations (Meneveau and Katz, 2000). Some form of spatial averaging is thus required, reducing the dynamic model to a global coefficient $C_{\text{sgs}} \in \mathbb{R}$. Using 2.46, positive clipping of the numerator can also be used in addition to spatial averaging,

$$\langle L_{ij} M_{ij} \rangle^+ = \frac{1}{2} \left(\langle L_{ij} M_{ij} \rangle + |\langle L_{ij} M_{ij} \rangle| \right). \tag{4.15}$$

This approach, however, is only relevant when estimating the undetermined coefficient of a given model. NN-based models on the other hand are commonly defined by their trainable parameters, which are then fixed during evaluation. In this context, positive clipping has been used as a post-processing step on the SGS term predicted by the NN-based model $\mathcal{M}$ (Maulik et al., 2019), also removing negative eddy viscosities,

$$\mathcal{M} = 0, \quad \forall \frac{\partial^2 \bar{\omega}}{\partial x_i^2} \mathcal{M} < 0. \tag{4.16}$$

The downside of performing this operation in post-processing is that the models are optimally trained to also produce negative eddy viscosities, consequently impacting their mean predictions. In practice, the resulting models are stable but also highly dissipative. An alternative would be to directly define this constraint in the NN architecture and train the model adequately.

Training distribution. Without restricting the NN-based model to positive eddy viscosities, using more training data has also been shown to improve numerical stability during simulations Guan et al. (2022a). This is an indirect consequence of sample size and variance impact on performance metrics of machine learning models, which has been known for a long time (Markham and Rakes, 1998). However, the motivation behind reduced modeling being primarily the computational cost of high-resolution simulations, generating data has a measurable limit. The problem is even more important since training algorithms are shown to converge better when the inter-sample correlation is low. Since simulation data is obtained by taking tiny temporal steps, integrating on large intervals and sub-sampling sparsingly is thus required to avoid highly correlated data. The saturating dataset size also depends on the model architecture, e.g., local inputs to train fully-connected architectures have been shown to require smaller datasets (Zhou et al., 2019). Recently, it has been shown that incorporating physical constraints (Guan et al., 2022b) using a similar approach described in chapter 3 promisingly improves the numerical stability of the SGS model.

4.2 A turbulence equivalent to end-to-end learning

Let us generalize the inverse problem involved in the time evolution of variables $\mathbf{y}(t)$. We assume an underlying differential equation to be known and defined by a direct operator $f(\mathbf{y})$. The aim is to solve an equivalent approximation for reduced variables $\bar{\mathbf{y}}(t)$ such that

$$\begin{cases} \dfrac{\partial \mathbf{y}}{\partial t} = f(\mathbf{y}), & \mathbf{y} \in \Omega \\ \dfrac{\partial \bar{\mathbf{y}}}{\partial t} = g(\bar{\mathbf{y}}) + \mathcal{M}(\bar{\mathbf{y}}), & \bar{\mathbf{y}} \in \bar{\Omega} \\ \mathcal{T}(\mathbf{y}) = \bar{\mathbf{y}} \end{cases} \tag{4.17}$$

where $\bar{\Omega} \subset \Omega$, g a reduced-order operator, $\mathcal{M}$ a reduced model and $\mathcal{T}$ a projection that maps direct variables to reduced ones. The objective in this inverse problem is to find an operator $\mathcal{M}$ such that the evolution of the reduced variables match the projection $\mathcal{T}(\mathbf{y})$ of the direct variables $\mathbf{y}$. In most situations, we have $f = g$ with variables existing on different spaces or dimensionalities. Note that this can be applied to any type of partial differential equation without any loss of generality. Within a learning framework, one states the identification of a general reduced term

$$\tau(\mathbf{y}) = \mathcal{T}(f(\mathbf{y})) - g(\mathcal{T}(\mathbf{y})) \approx \mathcal{M}(\bar{\mathbf{y}} \,|\, \theta) \tag{4.18}$$

where θ are trainable model parameters. Under the assumption that $\mathcal{T}$ commutes with partial derivatives, the classical approach comes to train a model $\mathcal{M}(\bar{\mathbf{y}} \,|\, \theta)$ as a functional

approximation of the missing term $\tau(\mathbf{y})$. This approach has been applied to SGS modeling, as described in chapter 3 for advection-diffusion equations, but has also been widely explored in the recent literature for NS equations (Beck and Kurz, 2021). It does not however constrain the trained model to behave as expected when implemented in the solver of the reduced-order system. In this respect, an end-to-end framework would appear as an appealing approach to explicitly state the SGS problem according to the optimal approximation of the projected direct variables. Such end-to-end approaches have shown many advantages in the approximation of partial differential equations in general (Bakarji and Tartakovsky, 2021; Chen et al., 2018; Fablet et al., 2021a). When applied to physical problems, they are often referred to as differentiable physics (de Avila Belbute-Peres et al., 2018; Holl et al., 2020; Um et al., 2020), since they require the gradient of all the considered operators and solvers to be available for the optimization algorithm. Overall, these two learning strategies differ in the space where the training is performed, similarly to the definition of *a priori* and *a posteriori* metrics (Pope, 2000) that evaluates the performance of SGS models. From now on, we will refer to *a priori* and *a posteriori* learning strategies for the classical and end-to-end approaches, respectively.

4.2.0.1 *a priori* learning

The *a priori* learning strategy comes to learn a SGS model using training metrics defined on instantaneous quantities, i.e. a direct measure of the accuracy of the model based on the predicted SGS term $\tau(\mathbf{y})$. The *a priori* loss $\mathcal{L}_<$ has the form

$$\mathcal{L}_<(\mathcal{M}) := \ell(\tau(\mathbf{y}), \mathcal{M}(\bar{\mathbf{y}} \,|\, \theta)) \tag{4.19}$$

where $\mathcal{M}$ is a given SGS model to be evaluated. The most common *a priori* metrics ℓ found in the physics community are mean squared and absolute errors between exact and predicted SGS terms. Training a NN-based SGS model according to the *a priori* strategy then comes to building a representative ground-truth dataset $\mathbb{D} := \{\bar{\mathbf{y}}\} \rightarrow \tau(\mathbf{y})$ of paired reduced variables and SGS terms and solve the following minimization problem w.r.t. model parameters θ,

$$\arg\min_\theta \mathcal{L}_<(\mathcal{M}) \equiv \arg\min_\theta \ell(\tau(\mathbf{y}), \mathcal{M}(\bar{\mathbf{y}} \,|\, \theta)), \quad \tau(\mathbf{y}), \bar{\mathbf{y}} \in \mathbb{D}. \tag{4.20}$$

Solving for (4.20) requires evaluating the partial derivatives of the *a priori* loss $\mathcal{L}_<$ which involves the gradient of the SGS model $\mathcal{M}$,

$$\frac{\partial \mathcal{L}_<}{\partial \theta} = \frac{\partial \mathcal{L}_<}{\partial \tau}\frac{\partial \tau}{\partial \theta} + \frac{\partial \mathcal{L}_<}{\partial \mathcal{M}}\frac{\partial \mathcal{M}}{\partial \theta} = \frac{\partial \mathcal{L}_<}{\partial \mathcal{M}}\frac{\partial \mathcal{M}}{\partial \theta}. \tag{4.21}$$

In practice, a NN-based model is implemented using a differentiable framework and (4.21) is automatically available.

Remark. We may also emphasize that, by construction, the *a priori* learning strategy shall lead to the optimal *a priori* results in the turbulence sense, which shall translate into a good instantaneous prediction of the SGS term according to metrics $\mathcal{L}_<$.

4.2.0.2 *a posteriori* learning

With the *a posteriori* learning strategy, the SGS problem is stated as the approximation of the temporal evolution projected direct variables w.r.t. some *a posteriori* metrics. This is important since it is possible for a model to perform well *a priori* but poorly in *a posteriori*, the most common factor being numerical instabilities. This is a direct motivation of the backscatter issue observed in two-dimensional flows described in section 4.1.1. Let us denote by Φ the flow operator that advances the reduced variables in time, i.e.,

$$\Phi_\theta^{t_1}(\bar{\mathbf{y}}(t_0)) = \bar{\mathbf{y}}(t_0) + \int_{t_0}^{t_1} g(\bar{\mathbf{y}}(t)) + \mathcal{M}(\bar{\mathbf{y}}(t) \,|\, \theta)\, \mathrm{d}t = \bar{\mathbf{y}}(t_1). \tag{4.22}$$

Numerically-speaking, flow operator Φ_θ involves a time integration scheme from start to end time t_0 and t_1, respectively. Following recent advances in neural integration schemes (Finlay et al., 2020; Ouala et al., 2021; Yan et al., 2019), we may consider here both explicit and adaptive schemes. The *a posteriori* loss $\mathcal{L}_>$ is now time-dependent and has the following form,

$$\mathcal{L}_>(\mathcal{M}) := \ell(\{\mathbf{y}(t)\}_{t\in[t_0,t_1]}, \{\Phi_\theta^t(\bar{\mathbf{y}}(t_0))\}_{t\in[t_0,t_1]}). \tag{4.23}$$

Now, the *a posteriori* minimization problem involves the time integration of Φ starting from $\bar{\mathbf{y}}(t_0) = \mathcal{T}(\mathbf{y}(t_0))$ on sub-intervals $[t_0, t_1]$. This requires a dataset $\mathbb{D}$ built from direct and reduced variables on continuous trajectories spanning temporal intervals, i.e. $\mathbb{D} := \{\mathbf{y}(t)\}_{t\in[0,T]}$. Note that since storing direct variables can be difficult due to their high resolution, one can instead precompute the reduced variables $\mathbb{D} := \{\mathcal{T}(\mathbf{y}(t))\}_{t\in[0,T]}$ and use the reduced-form loss accordingly,

$$\mathcal{L}_>(\mathcal{M}) := \ell(\{\mathcal{T}(\mathbf{y}(t))\}_{t\in[t_0,t_1]}, \{\Phi_\theta^t(\bar{\mathbf{y}}(t_0))\}_{t\in[t_0,t_1]}). \tag{4.24}$$

The minimization problem is now defined for every sub-interval $[t_0, t_1] \in [0, T]$ from the entire dataset,

$$\arg\min_\theta \mathcal{L}_> \equiv \arg\min_\theta \ell(\{\mathcal{T}(\mathbf{y}(t))\}_{t\in[t_0,t_1]}, \{\Phi_\theta^t(\bar{\mathbf{y}}(t_0))\}_{t\in[t_0,t_1]}), \qquad \{\mathbf{y}(t)\}_{t\in[t_0,t_1]} \in \mathbb{D}. \tag{4.25}$$

Updating model parameters θ requires the flow partial derivatives, i.e. expanding from (4.25) at t_1 gives

$$\frac{\partial \mathcal{L}_>}{\partial \theta} = \frac{\partial \mathcal{L}_>}{\partial \Phi} \left(\int_{t_0}^{t_1} \frac{\partial g(\bar{\mathbf{y}}(t))}{\partial \theta} + \frac{\partial \mathcal{M}(\bar{\mathbf{y}}(t) \,|\, \theta)}{\partial \theta}\, \mathrm{d}t \right). \tag{4.26}$$

This makes explicit that the gradient-based minimization of the *a posteriori* loss involves the computation of the gradient w.r.t. all the components of the reduced operator g as well as the considered integration scheme that discretizes Φ. The *a posteriori* learning strategy significantly widens the range of metrics that can be considered to train the SGS model. In the next section, we describe the specific points of the training algorithm along with the different options related to data sampling and loss continuity.

Remark. We may point out that similar differentiable models have recently been explored for temporally-developing plane turbulence jets (MacArt et al., 2021), mixed layer turbulence from KH instability (Stachenfeld et al., 2021) and the short-term simulation of two-dimensional flows (Kochkov et al., 2021). However, the behavior of the strategy on long-term stability is not explored, and evaluation against state-of-the-art models is not conducted.

4.2.1 Training algorithm

Following the continuous definition of the *a posteriori* strategy, we explain the discretization steps for practical use. For simplicity, we consider direct variables $\mathbf{y}$ spatially discretized on regular square grids, i.e. with $N_x = N_y = N \in \mathbb{R}^2$ grid points. Reduced variables $\bar{\mathbf{y}}$ are discretized on a coarser grid with $\bar{N} < N$, with reduced ratio $\Delta' = \Delta/\bar{\Delta}$, also called filter size, such that $\Delta'\bar{N} = N$. The temporal evolution of direct $\mathbf{y}$ and reduced $\bar{\mathbf{y}}$ variables can be integrated by taking small timesteps Δt and $\overline{\Delta t} = \Delta'\Delta t$, respectively. From spatial and temporal discretizations, we can define the discretized flow operator Φ,

$$\Phi_\theta^{t_1}(\bar{\mathbf{y}}(t_0)) = \bar{\mathbf{y}}(t_0) + \sum_{i=1}^{M} \overline{\Delta t} F_t \left[g(\bar{\mathbf{y}}(t_0 + i\overline{\Delta t})) + \mathcal{M}(\bar{\mathbf{y}}(t_0 + i\overline{\Delta t}) \,|\, \theta) \right] \tag{4.27}$$

where F_t is an arbitrary temporal discretization scheme, either explicit or implicit, and M the number of small steps required to integrate from t_0 to t_1, i.e. $M\overline{\Delta t} = t_1 - t_0$. As indicated in (4.24), we compare successive state variables in the loss using the recurrent relation

$$\Phi_\theta^{(M)} = \Phi_\theta^{t_0 + M\overline{\Delta t}}(\bar{\mathbf{y}}(t_0)) = \Phi_\theta^{(M-1)} + \overline{\Delta t} F_t \left[g(\Phi_\theta^{(M-1)} + \mathcal{M}(\Phi_\theta^{(M-1)} \,|\, \theta) \right]. \tag{4.28}$$

The sampling for the dataset is also arbitrary, one could skip γ samples to reduce the inter-sample correlation between consecutive initial training states,

$$\mathcal{L}_>(\mathcal{M}) := \ell(\{\mathcal{T}(\mathbf{y}(t))\}_{t\in\gamma[t_0,t_1]}, \{\Phi_\theta^t(\bar{\mathbf{y}}(\gamma t_0))\}_{t\in\gamma[t_0,t_1]}). \tag{4.29}$$

Also related to sampling, it is possible to only use independent sub-intervals that do not overlap, further reducing inter-sample correlation, i.e.,

$$\arg\min_\theta \mathcal{L}_> \equiv \arg\min_\theta \ell(\{\mathcal{T}(\mathbf{y}(t))\}_{t\in\gamma[t_0,t_1]}, \{\Phi_\theta^t(\bar{\mathbf{y}}(\gamma t_0))\}_{t\in\gamma[t_0,t_1]}), \tag{4.30}$$

$$\{\mathbf{y}(t)\}_{t\in\gamma[t_0,t_1]} \in \mathbb{D}, \bigcap_{d\in\mathbb{D}} [t_0,t_1]^d = \emptyset. \tag{4.31}$$

Algorithm 1 Training algorithm for SGS model $\mathcal{M}$ using the *a posteriori* strategy. Direct variables $\mathbf{y}$ are sampled randomly from dataset $\mathbb{D}$ which is only required to contain (reduced) direct states.

Require: dataset $\mathbb{D} := \{\mathcal{T}(\mathbf{y}(t))\}_{t\in[0,T]}$
Require: reduced system g, training model $\mathcal{M}(\bar{\mathbf{y}} \,|\, \theta)$
Require: number of iterations temporal M, number of epochs ϵ, starting time t_0
Require: loss function $\mathcal{L}_>$
for $i \leftarrow 1$ to ϵ **do**
  for all $\{\mathcal{T}(\mathbf{y}(t))\}_{t\in[t_0,t_1]} \in \mathbb{D}$ **do** ▷ Sample consecutive (reduced) direct states
    $\bar{\mathbf{y}}_0 \leftarrow \mathcal{T}(\mathbf{y}(t_0))$ ▷ Define initial reduced state from direct states
    for $j \leftarrow 1$ to M **do**
      $\bar{\mathbf{y}}_j \leftarrow \overline{\Delta t} F_t \left[(g(\bar{\mathbf{y}}_{j-1}) + \mathcal{M}(\bar{\mathbf{y}}_{j-1} \,|\, \theta))\right]$ ▷ Discrete temporal integration
    end for
    $L \leftarrow \mathcal{L}_>(\{\mathcal{T}(\mathbf{y}(t))\}_{t\in[t_0,t_1]}, \{\bar{\mathbf{y}}\}_{[0\cdots M]})$ ▷ Compute *a posteriori* loss
    $\theta \leftarrow \text{step}\left(\frac{\partial L}{\partial \theta}\right)$ ▷ Optimize model parameters
  end for
end for

The general algorithm to train a SGS model $\mathcal{M}$ using the *a posteriori* learning strategy is detailed in Algorithm 1.

Optimality limitations. In theory, one should expect optimal results if the training integration steps M correspond to the evaluation temporal horizon (in the order of thousands to millions of integration steps) since the SGS model would be optimized for the entire trajectory. In practice, ML use backpropagation (Goodfellow et al., 2016) to compute the gradient of the loss function $\mathcal{L}_>$ w.r.t. the weights θ of the trained SGS model. This algorithm is implemented using adjoint graphs, where the gradient of each operator is stored in reverse mode. The size of this graph can grow quickly to exceed GPU memory. If our reduced system g, SGS model $\mathcal{M}$ and temporal integration scheme F_t are composed of O_g, $O_\mathcal{M}$ and O_{F_t} operations, respectively, the graph size is defined as $O_{F_t} M(O_g + O_\mathcal{M})$. In practice, this has been the major limitation even for a simple toy system, i.e. O_g relatively small.

Convergence. The training process is required to perform some temporal integration of reduced (dynamical) system g, which can be a source of well-known numerical instabilities. If the reduced system g becomes unstable, the optimization algorithm will not be able to converge to an optimal solution. Issues related to temporal integration and expansive data generation may lead to practical difficulties. We consider additional key steps to Algorithm 1:

Algorithm 2 Discarding samples from unstable temporal integration, i.e. when CFL is above some safe threshold. This step can be inserted (between Line 6 and 7) in the general *a posteriori* learning Algorithm 1.

Require: Safety CFL threshold $C_{\max}$, similar to the one used in reduced solver g.

Require: Reduced candidate states $\bar{\mathbf{y}}_j$ from Algorithm 1 (Line 5).

1: **if** $\max_j \mathrm{CFL}(\bar{\mathbf{y}}_j) > C_{\max}$ **then** ▷ Discard batch if stability is not obtained

2: continue

3: **end if**

Algorithm 3 Linearly incrementing the number of integration steps M to avoid discarded samples due to CFL condition not respected (Algorithm 2). This step can replace the maximum M with a local number of integration steps M_{local} (Line 4) of the general *a posteriori* learning Algorithm 1.

Require: Maximum number of integration steps M.

Require: Current epoch i and number of training epochs ϵ.

1: $M_{\text{local}} \leftarrow i \left\lceil \frac{M}{\epsilon} \right\rceil$ ▷ Define a local number of iterations

1. Integration of the reduced system g must fulfill the Courant-Friedrichs-Lewy (CFL) criteria, typically used in the numerical solver. Incorrect predictions from the SGS model can impact the states of the simulation, creating numerical instabilities that can lead to a numerical blowup of the system and consequently exploding gradient for the minimization algorithm. The classical solution is to use an adaptative timestep such that CFL is fixed. In this context however, it would require computation of direct states on the fly in order to compare $\mathcal{T}(\mathbf{y}(t))$ and $\bar{\mathbf{y}}_j$ at the same temporalities, which would be extremely expansive. As a stability condition, we decide to withdraw samples with large CFL numbers, i.e. that do not guarantee numerical stability. Note that using temporal integration schemes with cheap interpolation capabilities could be an alternative to slightly adapt the timestep.

2. Large number of integration steps M may lead to more occurrences of large CFL and too many discarded samples, which in turn lead to slow convergence. Since the first SGS predictions are random guesses from model initialization, it is important to start the training process with M small and slowly increase to the desired temporal horizon as the model improves. In this chapter, we use a simple linear increment, but any increasing heuristic should work as long as the first critical epochs take a small number of iterations.

Stability and convergence steps 1. and 2. can be implemented using Algorithm 2 and 3, respectively.

4.2.2 Application to barotropic quasi-geostrophic turbulence

Geophysical turbulence is widely acknowledged to involve energy backscatter. This is caused by the relative dominance of the Coriolis force which creates vortical structures that appear two-dimensional. SGS modeling is also a key issue for the simulation of ocean and atmosphere dynamics because of the large range of motions involved (Frederiksen et al., 2012; Jansen et al., 2015; Juricke et al., 2020, 2019). As a case study framework, we consider barotropic QG flows. While providing an approximate yet representative model for rotating stratified flows found in the atmosphere and ocean dynamics, it involves relatively complex SGS features that make the learning problem non-trivial. As such, QG flows are regarded as an ideal playground to explore and assess the relevance of the *a priori* and *a posteriori* learning strategies for SGS models in geophysical turbulence. The evolution of reduced vorticity $\bar{\omega}$ for the QG equations is similar to (4.8) with additional terms related to various geophysical approximations already described in 2.2.3,

$$\begin{cases} \dfrac{\partial \omega}{\partial t} + N_\omega = \nu \nabla^2 \omega - \mu \omega - \beta \partial_x \psi + F_\omega, & \omega \in \Omega \\ \dfrac{\partial \bar{\omega}}{\partial t} + N_{\bar{\omega}} = \nu \nabla^2 \bar{\omega} - \mu \bar{\omega} - \beta \partial_x \bar{\psi} + F_{\bar{\omega}} + \underbrace{N_{\bar{\omega}} - \bar{N}_\omega}_{\tau_\omega}, & \bar{\omega} \in \bar{\Omega} \end{cases} \tag{4.32}$$

and SGS term τ_ω is equivalent to (4.10). In this formulation, the inverse problem can be expressed as a functional of reduced vorticity and streamfunction, i.e.,

$$\tau_\omega \approx \mathcal{M}(\bar{\omega}, \bar{\psi}). \tag{4.33}$$

In order to study the *a posteriori* learning strategy, we solve equations (4.32) using a *differentiable* pseudospectral code with full 3/2 dealiasing (Canuto et al., 2007) and a classical fourth-order explicit Runge Kutta time advancement. The system is defined in a squared domain $\Omega \in [-\pi, \pi]^2$, or domain length $L = 2\pi$ discretized with a Fourier basis, i.e. double-periodic boundary conditions $\partial\Omega$ on N grid points with uniform spacing $\Delta = LN^{-1}$. We extract the SGS term and the non-residual input quantities using spatial filtering described in 2.1.2, i.e.,

$$\{\bar{\omega}, \bar{\psi}\} \rightarrow \tau_\omega. \tag{4.34}$$

Note that τ_ω is only required for the *a priori* learning strategy. To generate the corresponding datasets, we subsample one direct state every Δ' iteration performed by f so that these states directly correspond to one iteration performed by g, we have,

$$\mathbb{D} := \{\bar{\omega}^{(i)}, \bar{\psi}^{(i)}\} \rightarrow \tau_\omega^{(i)} = \{\bar{\omega}(t), \bar{\psi}(t)\}_{t\in[0,T]} \rightarrow \tau_\omega(t)_{t\in[0,T]} \tag{4.35}$$

$$(i\overline{\Delta t}) \in [0, T] \tag{4.36}$$

$$t \in [0, T] \tag{4.37}$$

so that datasets for both *a priori* and *a posteriori* learning are equivalently composed of the same samples.

		DECAY	FORCED	BETA-PLANE
Length of the domain	L	2π	2π	2π
Linear drag	μ	0	2×10^{-2}	2×10^{-2}
Kinematic viscosity	ν	3.125×10^{-5}	1.025×10^{-5}	1.025×10^{-5}
Rossby parameter	β	0	0	2.195×10^{2}
Reynolds number	Re	3.2×10^{4}	2.2×10^{5}	3.4×10^{5}
Number of grid points	N	2048	2048	2048
Timestep	Δt	10^{-4}	10^{-4}	10^{-4}

Table 4.1: Parameters of the different DNS flow configurations. Note that reduced systems use the same parameters, except for grid resolution $\bar{N}$ and timestep $\overline{\Delta t}$, obtained from the spatial filter ratio. The quantities are given in numerical (unitless) as directly used in the solver for reproducibility.

Flow configurations. To evaluate the robustness of the training strategies, we design numerical experiments for three different configurations of QG flows. The numerical parameters of these configurations are detailed in Table 4.1.

DECAY (see Fig. 4.3). We first study decaying turbulence and reproduce the configuration described in previous works based on the *a priori* strategy (Guan et al., 2022a; Maulik et al., 2019). This type of flow is particularly interesting because of its non-stationary nature, i.e., the system's invariants are temporally varying. The initial vorticity fields are sampled randomly from a Gaussian distribution $\omega(t = 0) \sim \mathcal{N}(0, 1)$ at moderate wavenumbers $k \in [10, 32]$ and the system is integrated for 10000 iterations before reaching spectrum self-similarity (Batchelor, 1969).

FORCED (see Fig. 4.4). We then evaluate the SGS models on a more realistic wind-forced configuration representative of mesoscale oceanic simulation (Fox-Kemper and Menemenlis, 2008; Graham and Ringler, 2013). To reproduce an equilibrium solution, we use a linear drag $\mu > 0$ and initiate turbulence from a slowly varying in time circular source at large scale $k = 4$ with steady enstrophy rate injection $\langle F_\omega(t)^2 \rangle = 3$ such that,

$$F_\omega(t) = \cos(4y + \pi \sin(1.4t)) - \cos(4x + \pi \sin(1.5t)). \tag{4.38}$$

Stationary turbulent states are obtained from the same random initialization, but followed by a 500000 iterations spin-up on a smaller grid (1024^2) and energy propagation to the smallest scales of the direct grid (2048^2) in about 25000 iterations.

BETA-PLANE (see Fig. 4.5). Finally, we observe the impact of planetary rotation through the beta-plane effect on a mid-latitude geophysical flow, which creates jet-like structures typically seen in atmospheres. The initialization for this configuration is the same as the previous one.

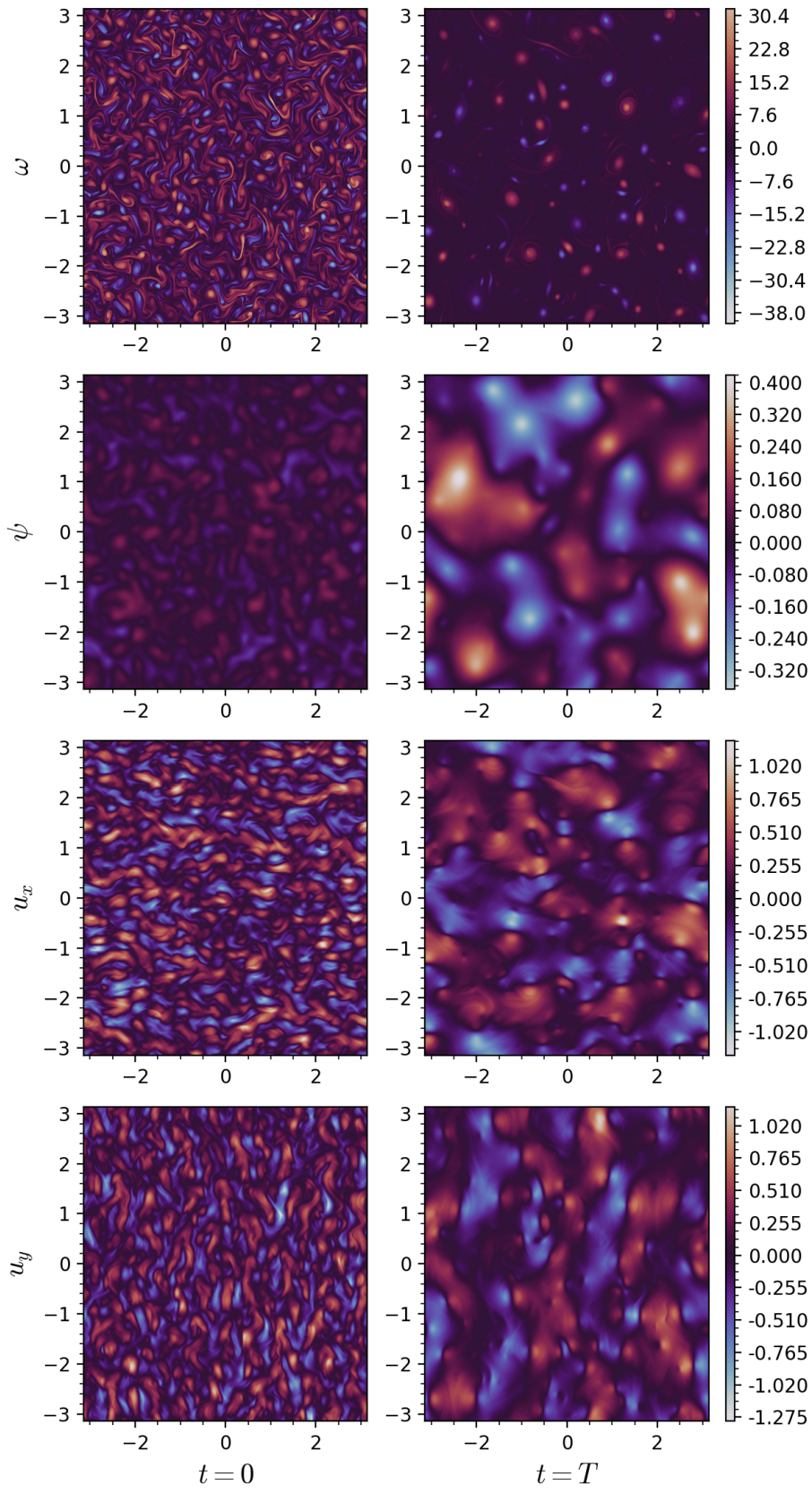

Figure 4.3: DNS fields at start time (left) and stop time (right) for a trajectory simulation in **Decay** QG turbulence. The simulation was run for 144000 temporal iterations, corresponding to approximately 120 eddy turnover times.

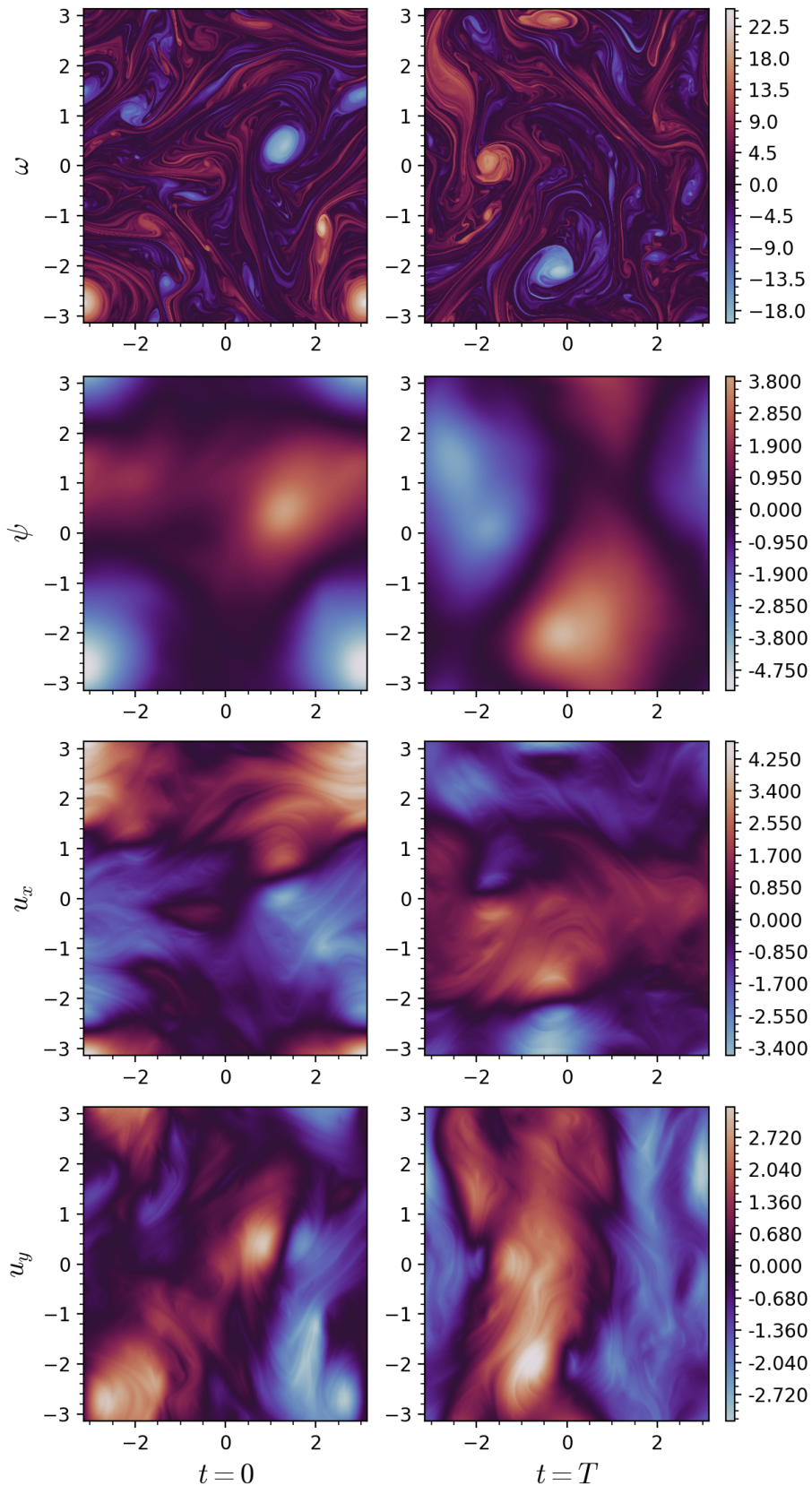

Figure 4.4: DNS fields at start time (left) and stop time (right) for a trajectory simulation in **Forced** QG turbulence. The simulation was run for 288000 temporal iterations, corresponding to approximately 60 eddy turnover times.

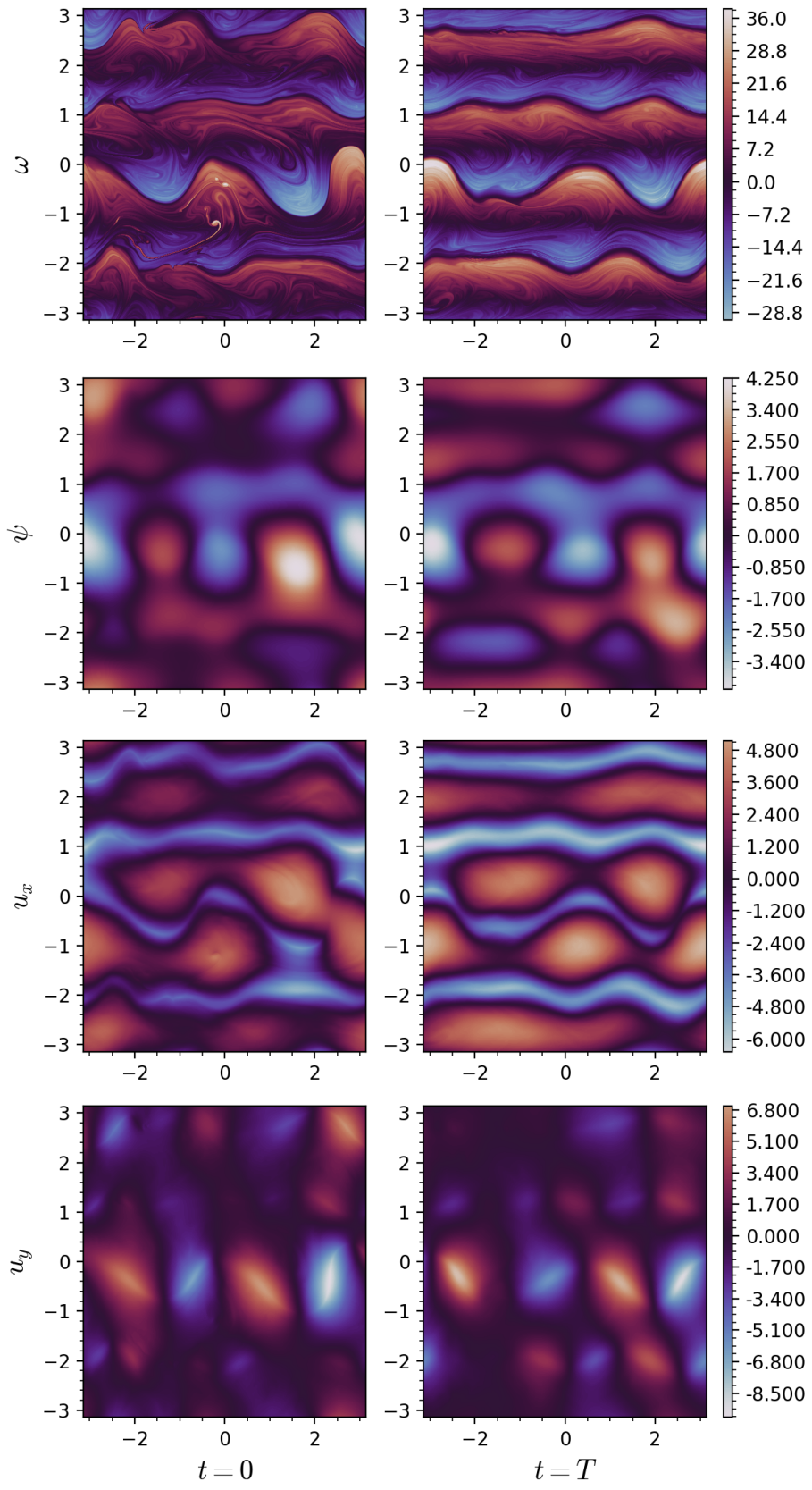

Figure 4.5: DNS fields at start time (left) and stop time (right) for a trajectory simulation in **Beta-plane** QG turbulence. The simulation was run for 288000 temporal iterations, corresponding to approximately 120 eddy turnover times.

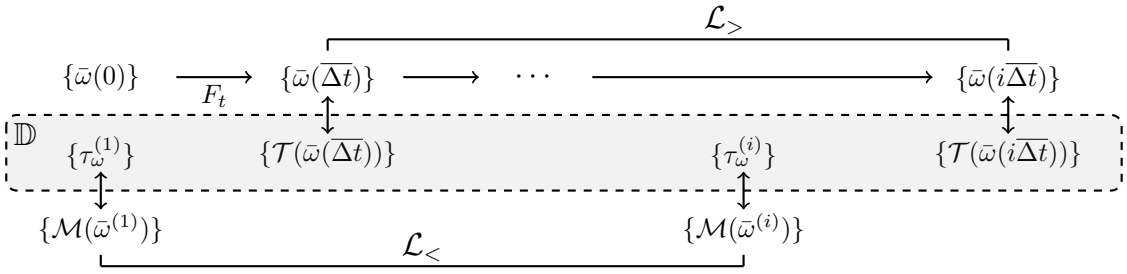

Figure 4.6: Loss generation for the *a priori* (bottom) and *a posteriori* (top) learning strategies from dataset $\mathbb{D}$.

NN architectures and learning schemes. The main focus of this chapter being the impact of the *a priori* and *a posteriori* learning strategies, we consider the same NN-based models $\mathcal{M}$. We use a convolutional architecture, particularly relevant for spatially structured problems, and already discussed in section 2.4. The architecture consists of 10 Conv layers with kernels K of size $5 \times 5 \times 5$ and 64 filters F each, followed by non-linear Relu activations. Again, input boundaries are replicated periodically given the geometry of the domain. It writes,

$$\begin{aligned}\mathcal{M}(\bar{\omega}, \bar{\psi} \,|\, \theta) := \mathrm{Pad}_P \circ \mathrm{Conv}_{K=5}^{F=64}(\theta_1) \circ \mathrm{Relu} \circ \cdots \circ \mathrm{Conv}_{K=5}^{F=64}(\theta_9) \circ \mathrm{Relu} \\ \circ \mathrm{Conv}_{K=1}^{F=1}(\theta_{10}).\end{aligned} \tag{4.39}$$

More involved architecture could further improve the overall performance and might be interesting to explore. However, we may point out that the goal is not to design an optimal NN-based architecture but rather to evaluate the training strategy at the same computational cost. Regarding the learning phase, the training loss for the *a priori* strategy (4.20) computes the MSE of the predicted term w.r.t. the SGS term produced by the direct simulation, on a batch of B samples,

$$\mathcal{L}_<(\mathcal{M}) := \frac{1}{B} \sum_{i=1}^{B} ||\tau_\omega^{(i)} - \mathcal{M}(\bar{\omega}^{(i)}, \bar{\psi}^{(i)})||_2^2. \tag{4.40}$$

For the *a posteriori* strategy (4.25), the choice of the training loss is more flexible, since we can explore spatiotemporal metrics for temporal batches of $M = (t_1 - t_0)/\overline{\Delta t}$ discrete integration steps. To illustrate the basics of the strategy, we use the MSE of the most important state of the QG system, i.e. the vorticity,

$$\mathcal{L}_>(\mathcal{M}) := \frac{1}{M} \sum_{i=1}^{M} ||\mathcal{T}(\omega(i\overline{\Delta t})) - \bar{\omega}(i\overline{\Delta t})||_2^2. \tag{4.41}$$

One may note that the losses are not computed on the same elements and corresponding temporalities (see Fig. 4.6). For the *a priori* strategy, the loss function is computed over batches of data B, randomly sampled from the dataset. For the *a posteriori* strategy, however, the loss function is based on M states integrated from randomly sampled starting fields $\bar{\omega}(t_0)$.

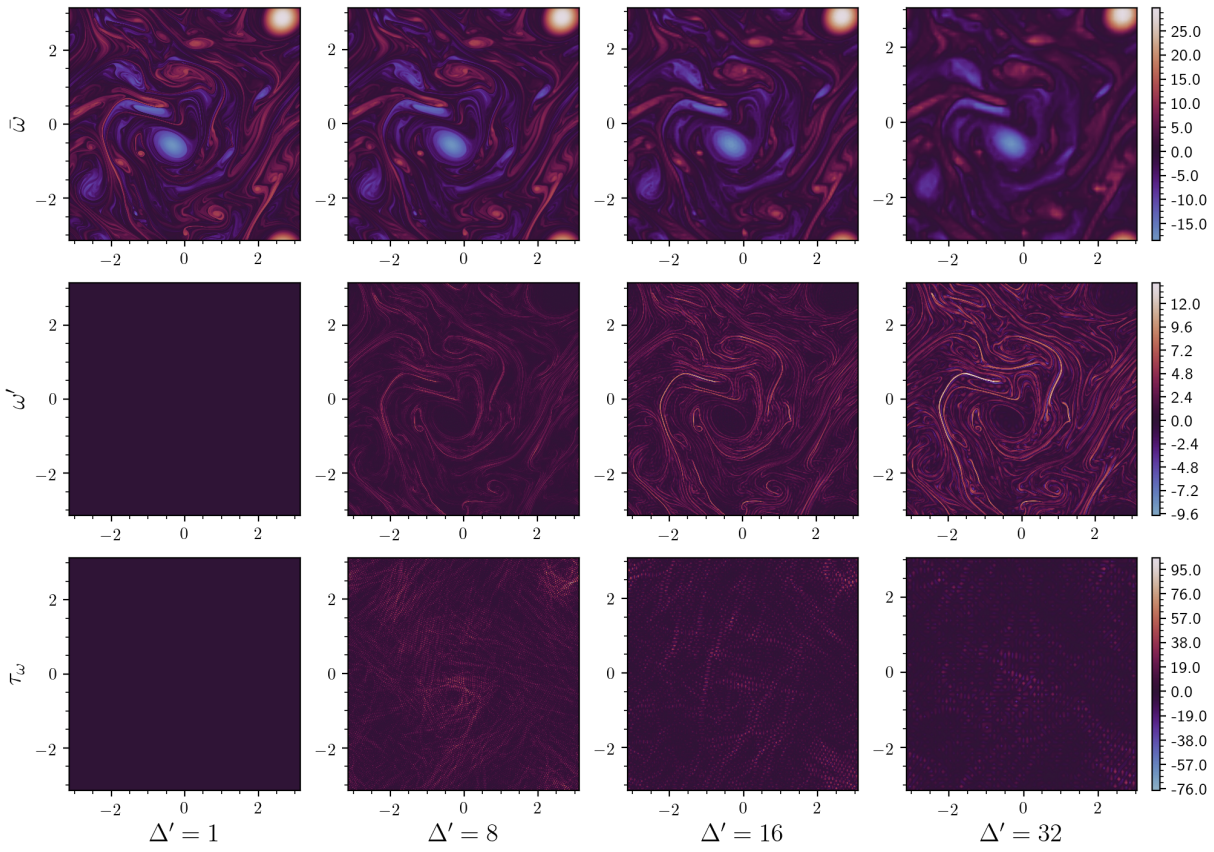

Figure 4.7: Example of reduced vorticity $\bar{\omega}$, corresponding subgrid contributions $\omega' = \omega - \bar{\omega}$ and SGS term τ_ω at different grid ratios Δ' filtered using a cut-off kernel in the **FORCED** configuration. Note that a value of $\Delta' = 1$ indicates direct simulation, i.e. $\omega = \bar{\omega}$ and $\omega' = \tau_\omega = 0$.

4.3 Results

In this section, we describe multiple experiments on the performance and applicability of the described learning strategies to predict subgrid-scale quantities τ_ω in reduced simulations (see Fig. 4.7). One major concern is long-term stability, in particular for stationary configurations. Reducing the amount of data used for training is also an important objective since direct simulations are extremely expensive, or even impossible. Through these experiments, we will also describe some degrees of generalization, i.e., avoiding generating new data and training new models when the simulation description changes. The models are trained with the Adam optimizer (Kingma and Ba, 2015) and simple learning rate schedulers, having minimal impact on the final performance. Empirically, we noted that 30 training epochs were necessary for the *a priori* strategy, while 5 are enough for the *a posteriori* strategy. However, the latter is consequently more expansive due to the solver steps involved in the training loop. As classical baselines, we also compare the performance of the dynamic versions of the Smagorinsky and Leith models described in 2.1.4.

		DECAY	FORCED	BETA-PLANE
Direct grid size	N	2048	2048	2048
Direct timestep	Δt	10^{-4}	10^{-4}	10^{-4}
Filter size	Δ'	16	16	16
Reduced grid size	$\bar{N}$	128	128	128
Reduced timestep	$\overline{\Delta t}$	16×10^{-4}	16×10^{-4}	16×10^{-4}
a priori batch size	B	25	25	25
a posteriori iterations	M	25	25	25
Training sample skip	γ	1	1	1
Training trajectories		10	10	10
Training samples		3000	3000	3000
Testing trajectories		5	5	5
a priori testing horizon		3000	3000	3000
a posteriori testing iterations		9000	18000	18000

Table 4.2: Training and testing setup for the long-term stability experiment. Direct data fields are projected onto a reduced grid using a cut-off filter, i.e. coarse-graining in spectral space (2.23). Training and testing samples as well as testing iterations are given for each trajectory.

4.3.1 Long-term stability

In this first experiment, we train models for both learning strategies in each turbulent configuration, i.e. three separate *a priori* and *a posteriori* models optimized from and evaluated in **DECAY**, **FORCED** and **BETA-PLANE** flows. The datasets contain 10 independent trajectories from simulations of 3000 snapshots reduced with $\Delta' = 16$ (see Fig. 4.7) each using different initial conditions, which gives datasets of 30000 samples. We take consecutive samples separated by $\overline{\Delta t}$, i.e. without skipping $\gamma = 1$ (4.29) in order to emulate large data availability at the expanse of a high inter-sample correlation (here ≈ 1). To enable a fair comparison of the two learning strategies, we choose an *a priori* batch size equal to the number of *a posteriori* iterations, i.e. $B = M = 25$ with non-overlapping temporal intervals (4.31). To evaluate the long-term stability and generalization, the temporal length of the evaluation is taken to be substantially larger than the training data, with $3\times$ longer in the **DECAY** configuration, limited by the non-stationary nature of the flow, and $6\times$ longer for the **FORCED** and **BETA-PLANE** stationary flows. Training and evaluation parameters are summarized in Table. 4.2. Before delving into the detailed analysis of both short-term *a priori* insights and long-term *a posteriori* simulation performance, we can start by observing vorticity fields $\bar{\omega}$ at the end of a testing horizon and a long-term trajectory (see Figs. 4.8 and 4.9).

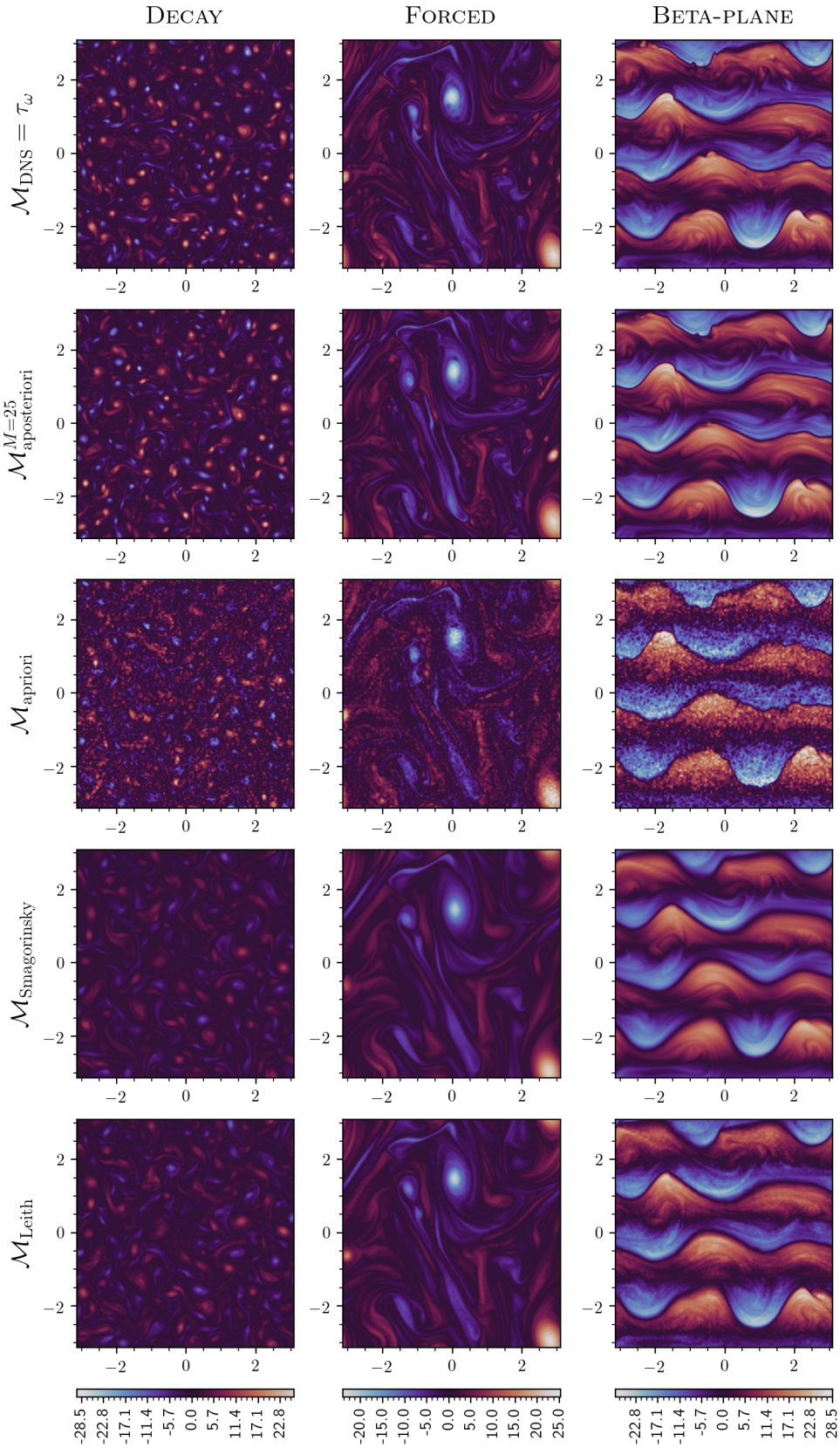

Figure 4.8: Reduced vorticity fields $\bar{\omega}$ at the end of a testing horizon from the direct simulation (τ_ω, top) and the different models for configurations described in Table 4.2.

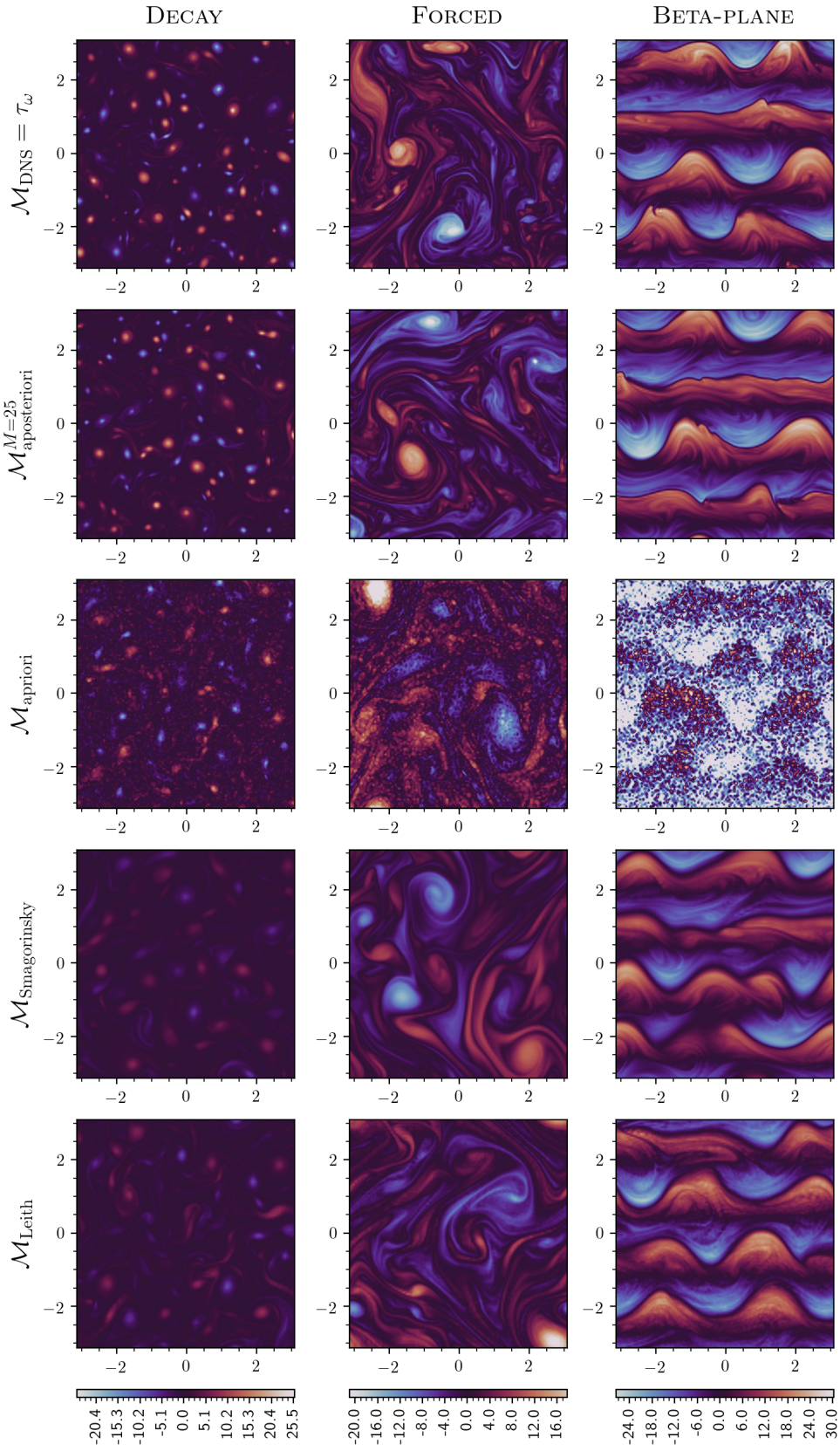

Figure 4.9: Reduced vorticity fields $\bar{\omega}$ at the end of a long-term trajectory from the direct simulation (τ_ω, top) and the different models for configurations described in Table 4.2.

	$\uparrow r$	$\downarrow D_{\mathrm{JS}}$	$\downarrow I_\epsilon$	$\uparrow r$	$\downarrow D_{\mathrm{JS}}$	$\downarrow I_\epsilon$	$\uparrow r$	$\downarrow D_{\mathrm{JS}}$	$\downarrow I_\epsilon$
$\mathcal{M}_{\mathrm{aposteriori}}$	**0.768**	**0.072**	**0.673**	0.454	**0.135**	**0.456**	0.480	**0.013**	**1.099**
$\mathcal{M}_{\mathrm{apriori}}$	0.736	0.658	-2.940	**0.821**	0.691	-0.794	**0.823**	0.736	-0.982
$\mathcal{M}_{\mathrm{Smagorinsky}}$	0.176	0.322	13.274	0.086	0.490	4.097	0.046	0.557	9.086
$\mathcal{M}_{\mathrm{Leith}}$	0.145	0.467	6.863	0.077	0.583	2.408	0.033	0.677	3.662
		Decay			**Forced**			**Beta-plane**	

Table 4.3: Short-term performance of the considered SGS models in the three different configurations.

4.3.1.1 Qualitative analysis

From those fields, it is clear that the *a posteriori* models lead to the closest results. In particular, we expect to see vortex pairing and the emergence of larger structures in **Decay**, but physical models are incorrectly dissipating relevant small scales due to their purely diffusive form. The NN-based model trained with the *a priori* strategy has accumulated small-scale enstrophy and is thus perturbed with noise coming from numerical instabilities. The model trained with the *a posteriori* strategy seems to be stable and retain small-scale features, even outside of the training regime, which supports some degree of generalization (or extrapolation), particularly difficult for non-stationary dynamics. In the **Forced** configuration, we see both large vortices generated by wind-forcing and small filaments in between. Overall, we draw similar conclusions, except that stability is preserved for a larger temporal horizon for the *a posteriori* model, while instabilities are exaggerated for the *a priori* model. Finally, **Beta-plane** has an important impact on the topology of the dynamics, as it creates high-velocity longitudinal jets. Note however that in this simple setting without topography (flat bottom layer), the system does not go through state transitions and remains in statistical equilibrium. The strong vorticity gradients between jets are predicted more accurately by the *a posteriori* model while the *a priori* model exhibits faster instabilities, Leith seems to start accumulating small-scale noise and Smagorinsky is still over-dissipating.

4.3.1.2 Short-term *a priori* performance

We first look at the performance of the different models on the same temporal horizon than used in the training data, i.e., 3000 samples each separated by $\overline{\Delta t}$. Here, it is interesting to quantify *a priori* performance in order to make potential connexions with long-term *a posteriori* performance. We extract in Table 4.3 similar metrics than described in 3.3.2, in particular *structural* correlation coefficient r, *statistical* Jensen-Shannon distance D_{JS} and *functional* error on the integral of the transfer term (4.14), $I_\epsilon = -\int \bar{\omega}\mathcal{M}\,\mathrm{d}S + \int T_\omega\,\mathrm{d}S$. Interestingly, the *a priori* model performs on-par or better on the *structural* metric for each configuration, particularly in stationary flows for which the other models

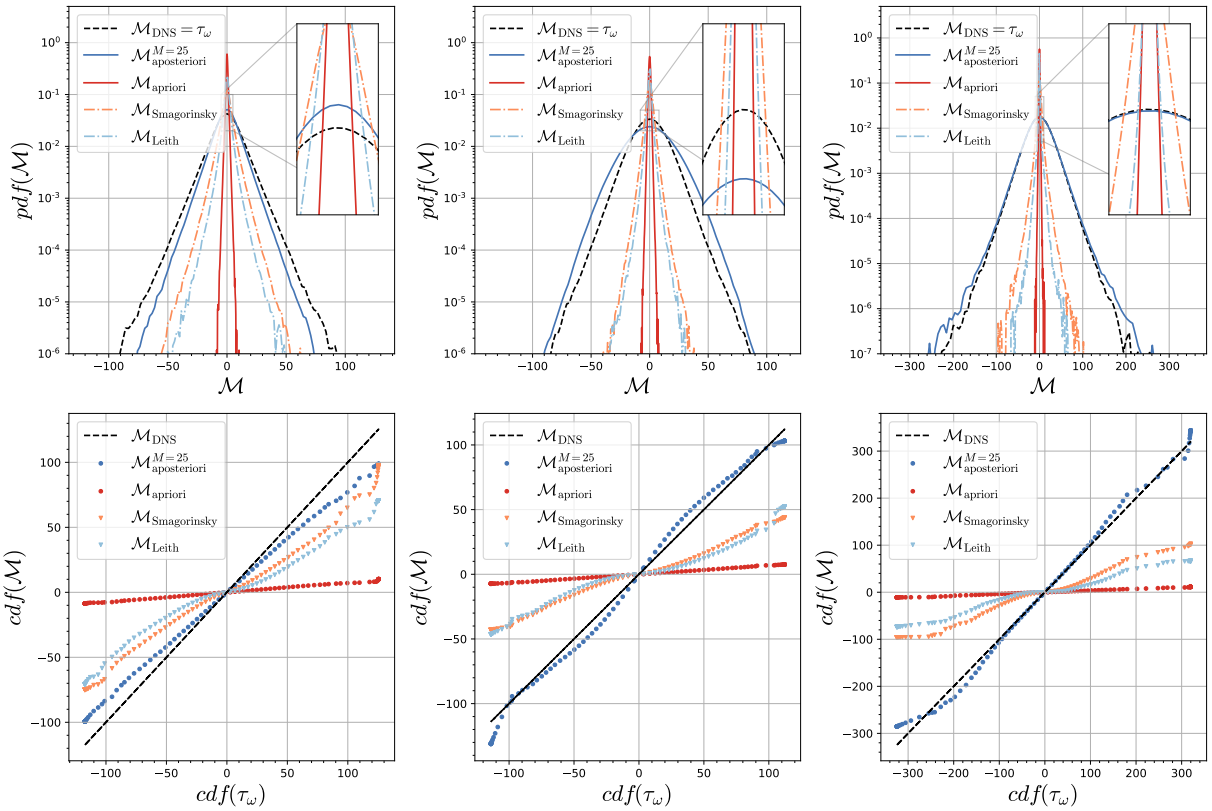

Figure 4.10: Probability distribution function (top) and quantiles-quantiles (bottom) of the SGS term in **Decay** (left), **forced** (center) and **Beta-plane** (right) configurations on the short-term testing dataset (3000 samples).

have correlation coefficients $r < 0.5$. We already know from the qualitative analysis that the *a priori* model is not stable in a simulation, which tends to indicate that the *structural* metric is not a good indicator of stability in *a posteriori* scenario. It may also point out that the *a priori* model trained on the MSE of the SGS term, a purely *structural* metric, can not be guaranteed to reach simulation stability. On the *statistical* metric, however, the *a priori* model performs worst, while the *a posteriori* model clearly outperforms the other models. To understand this large difference, we can look at the PDF of the SGS term and predictions (see Fig. 4.10). In **Decay** configuration, we see a better prediction of mean values for the *a posteriori* model while others are closer to a double-exponential, but also a closer representation of the distribution tails for the *a posteriori*. In **Forced** configuration, the *a posteriori* slightly underpredicts mean values and slightly overpredicts intermediate values but still remains in best agreement with the SGS term. Finally, in **Beta-plane** configuration, the *a posteriori* model is almost perfectly reproducing the PDF of τ_ω. This time, it is difficult to conclude about the relation between simulation stability and *statistical* metrics. Indeed, even if the unstable *a priori* model is particularly inefficient at reproducing the distribution of the predicted term, the physical models are not the closest either, but they are known to be particularly stable in practice. For the

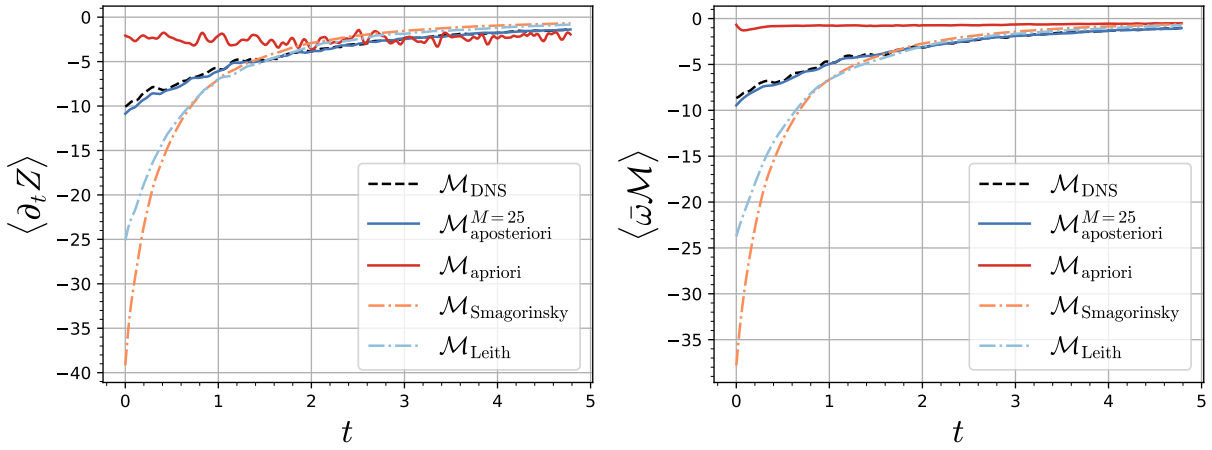

Figure 4.11: Evolution of domain-averaged enstrophy budget (left) and modeled transfer term (right) in **DECAY** configuration for 3000 reduced temporal steps.

functional metric, we observe the smallest absolute value for the *a posteriori* model and a large positive error for the physical models. On the other hand, the *a priori* model is always producing negative errors, which corresponds to an over-prediction of global enstrophy transfer compared to the SGS model. Here, it seems that models that do not dissipate enough enstrophy might indicate potential simulation instabilities, while highly dissipative physical models are known to be stable.

4.3.1.3 Long-term *a posteriori* performance

We now run new simulations for each configuration with the direct solver and the different reduced models for several reduced iterations as described in Table 4.2. First, we want to analyze the evolution of the transfer term on the same temporal horizon as in the *a priori* evaluation. We can also look at the two-dimensional enstrophy budget (4.12), which contains additional terms in QG equations,

$$\frac{\partial Z(\mathbf{x})}{\partial t} = \underbrace{\bar{\omega}\nu\nabla^2\bar{\omega}}_{\text{viscous}} - \underbrace{\bar{\omega}N_{\bar{\omega}}}_{\text{resolved}} + \underbrace{\bar{\omega}\tau_{\omega}}_{\text{modeled}} \tag{4.42}$$

$$- \underbrace{\bar{\omega}\mu\bar{\omega}}_{\text{linear drag}} \tag{4.43}$$

$$- \underbrace{\bar{\omega}\beta\partial_x\bar{\psi}}_{\text{beta-plane}} \tag{4.44}$$

$$+ \underbrace{\bar{\omega}F_{\bar{\omega}}}_{\text{external}} . \tag{4.45}$$

where resolved and modeled refer to non-linear terms. Then, we will discuss the evolution of the quadratic invariants of the system and spectral statistics to evaluate their long-term performance.

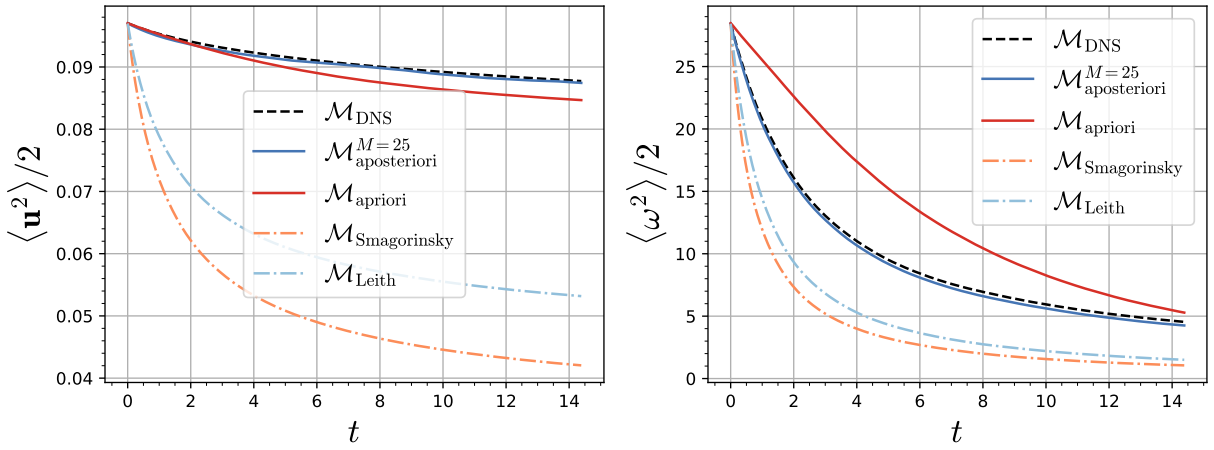

Figure 4.12: Evolution of domain-averaged energy (left) and enstrophy(right) in **Decay** configuration for 9000 reduced temporal steps.

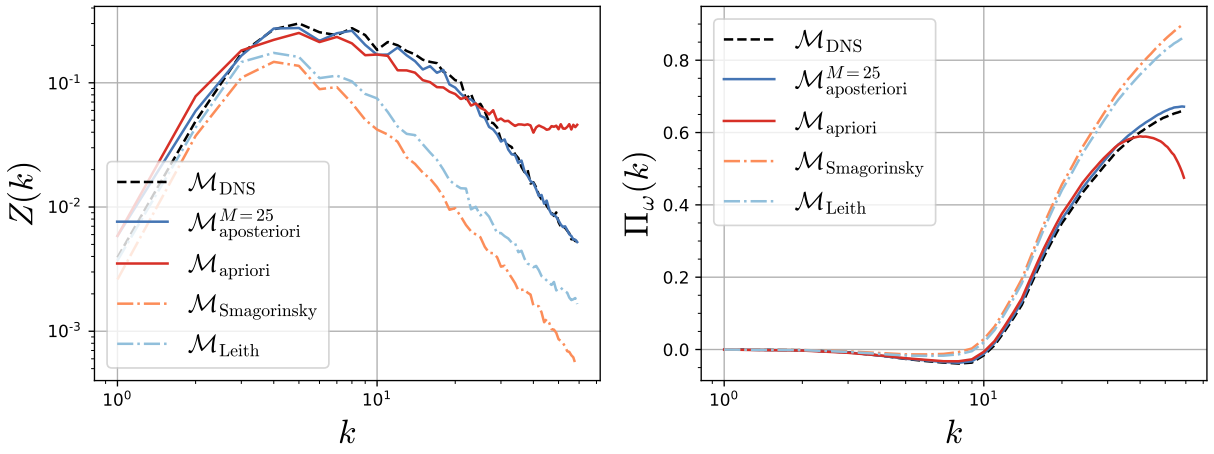

Figure 4.13: Final enstrophy spectrum (left) and time-averaged enstrophy flux (right) in **Decay** configuration for 9000 reduced temporal steps.

Decay. In this configuration, linear drag (4.43), beta-plane (4.44), and external (4.45) terms are equal to zero. We can see in Fig. 4.11 that the enstrophy budget is dominated by the modeled transfer term. We observe large values for the physical models at the beginning, which are effectively dissipating the small scales until none are left. The *a priori* model produces smaller global transfers compared to the direct simulation, which indeed was reflected by negative integral dissipation error I_ϵ in the *a priori* evaluation. On the other hand, modeled transfers are closely reproduced by the *a posteriori* model, resulting in a correct enstrophy budget. The evolution of the quadratic invariants in Fig. 4.12 confirms these observations with a large decrease of energy and enstrophy for the physical models due to excessive dissipation, while the *a priori* model correctly captures the energy decay but dissipates enstrophy too slowly compared to the direct simulation. The spectral statistics shown in Fig. 4.13 are also in close agreement with the direct simulation for the *a posteriori* model, in particular for the large wavenumbers of the

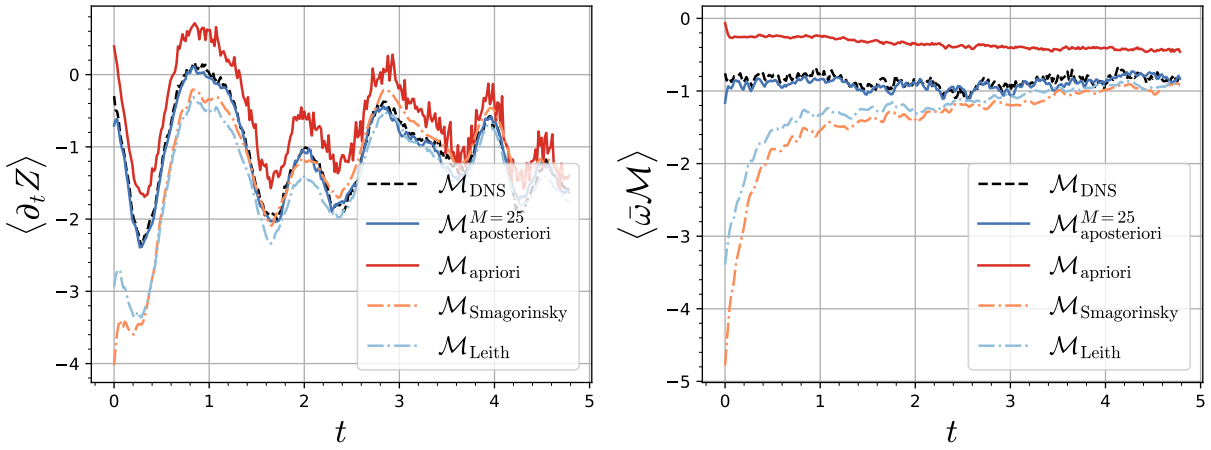

Figure 4.14: Evolution of domain-averaged total enstrophy budget (left) and modeled transfer term (right) in **FORCED** configuration for 3000 reduced temporal steps.

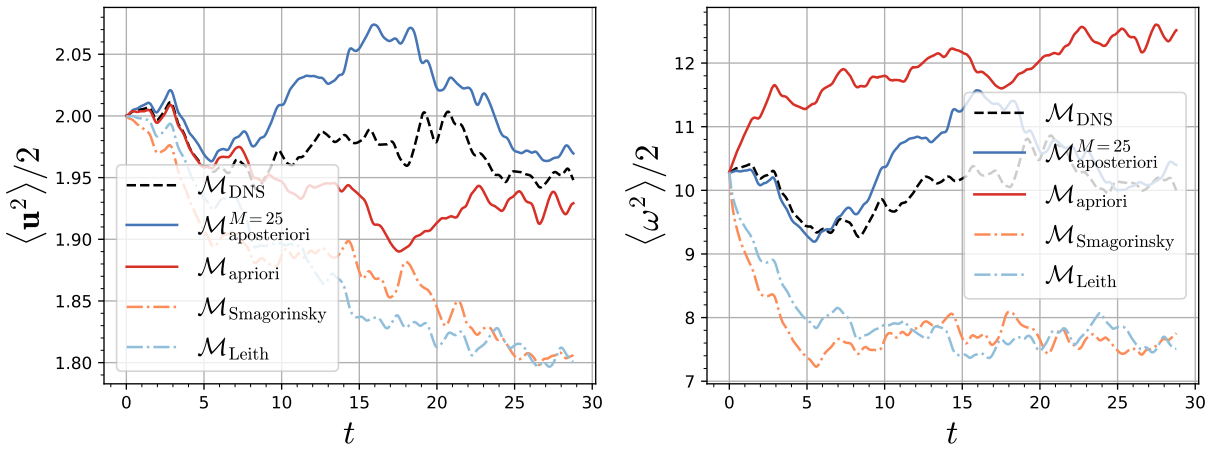

Figure 4.15: Evolution of domain-averaged energy (left) and enstrophy(right) in **FORCED** configuration for 18000 reduced temporal steps.

enstrophy spectrum $Z(k) = k^2 E(k)$ which particularly highlights the dynamics of the smallest resolved scales. We note that the unstable behavior of the *a priori* model is visible from large values in the enstrophy spectrum and small values in the enstrophy flux at the largest wavenumbers k, typical of numerical noise accumulation.

FORCED. In this configuration, only the beta-plane (4.44) term is equal to zero. The resulting dynamics of the enstrophy budget is quite different from the one in the previous configuration. The enstrophy budget in Fig. 4.14 is dominated by external forcing (not shown here), which is visible by the oscillating behavior driven by the trigonometric functions in (4.38). It is still easy to see that the *a posteriori* model is again the closest to the enstrophy budget of the direct simulation. Now, the modeled transfers are expected to remain constant in an equilibrium configuration, which is not the case for both physical models and the *a priori* model due to the same reasons already pointed out in the previous

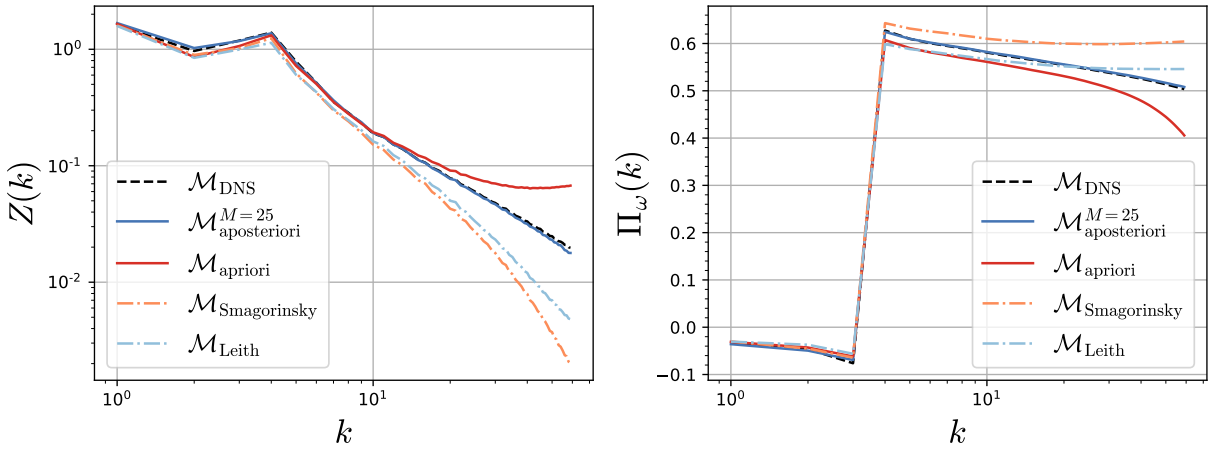

Figure 4.16: Time-averaged enstrophy spectrum (left) and enstrophy flux (right) in **Forced** configuration for 18000 reduced temporal steps.

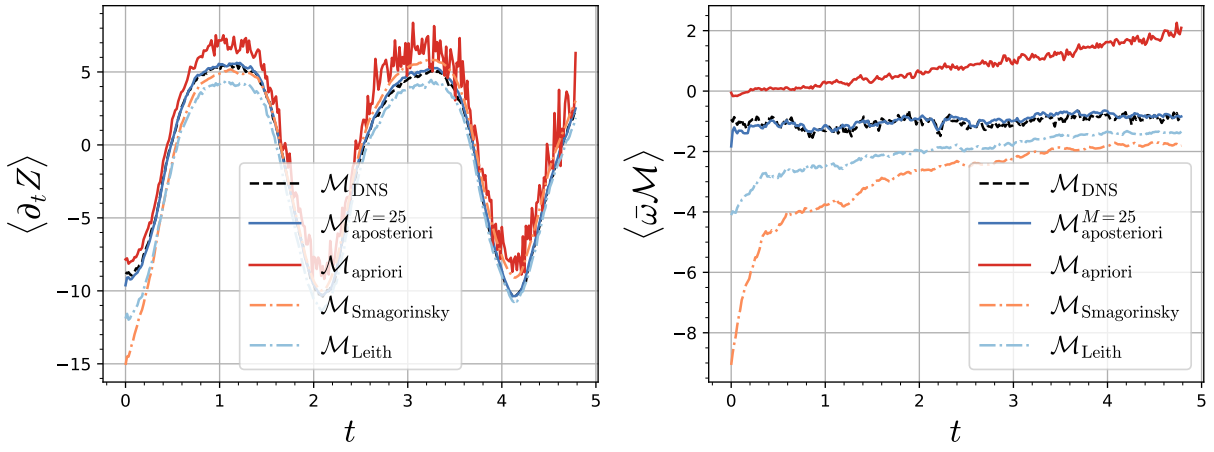

Figure 4.17: Evolution of domain-averaged total enstrophy budget (left) and modeled transfer term (right) in **Beta-plane** configuration for 3000 reduced temporal steps.

configuration. Here, we run simulations for at least 3 times longer than the complete decorrelation time of the system. We can indeed see large fluctuations of the quadratic integrals in Fig. 4.15 due to the chaotic nature of the flow, but we expect those quantities to remain approximately constant over time. This property is verified on the kinetic energy for both NN-based models, but the enstrophy of the *a priori* model increases over time, which indicates an unrealistic accumulation of enstrophy and may result in a potential future blow-up of the simulation. Additionally, the time-averaged enstrophy spectrum in Fig. 4.16 demonstrates the ability of the *a posteriori* model to reproduce accurately both the small scales and the largest scales of the simulation compared to the other models.

Beta-plane. In this configuration, all of the terms are non-zero and we now observe jet-like dynamics. The enstrophy budget in Fig. 4.17 is now dominated by the beta-plane

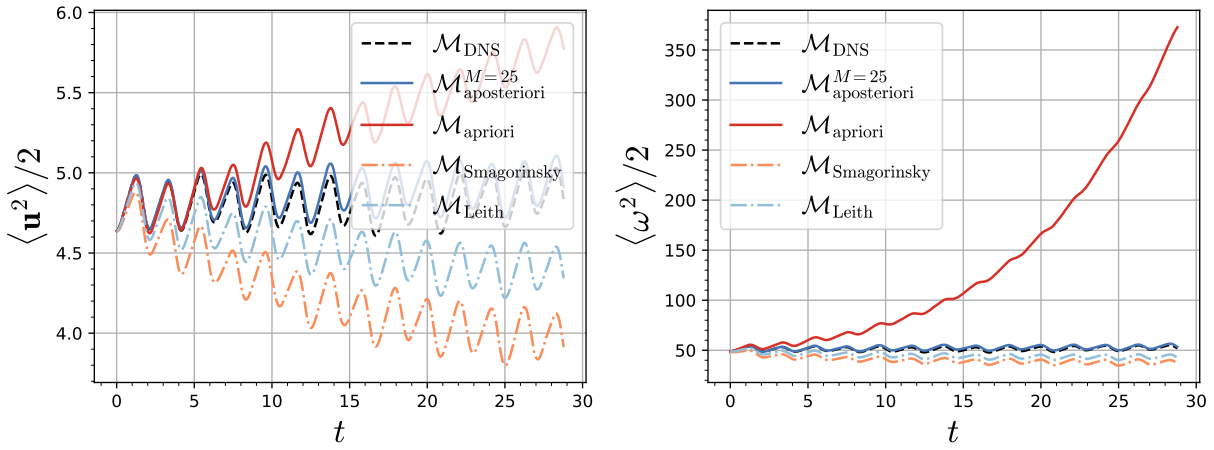

Figure 4.18: Evolution of domain-averaged energy (left) and enstrophy(right) in **BETA-PLANE** configuration for 18000 reduced temporal steps.

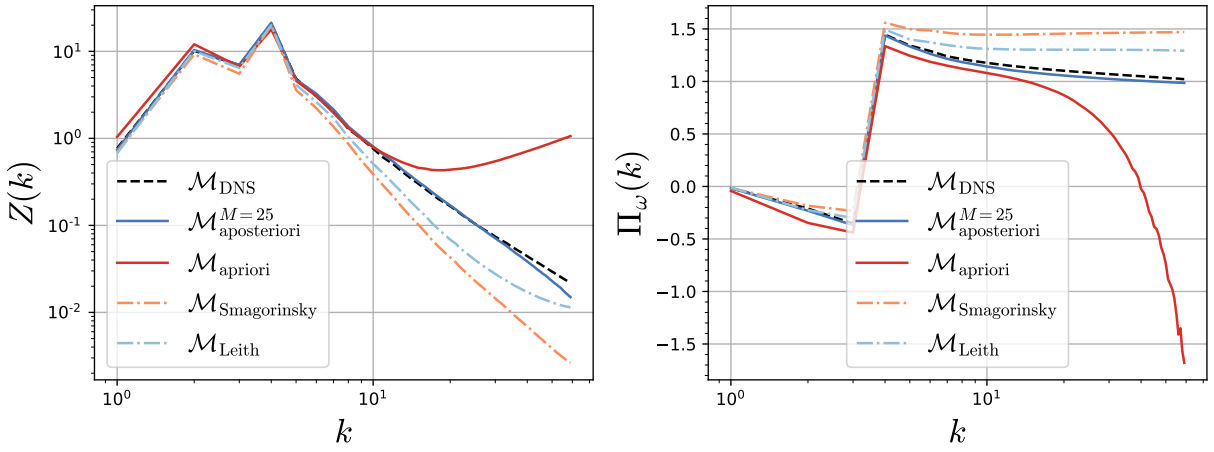

Figure 4.19: Time-averaged enstrophy spectrum (left) and enstrophy flux (right) in **BETA-PLANE** configuration for 18000 reduced temporal steps.

effect (also not shown here). While not visible on the enstrophy budget due to its small magnitude, the modeled transfers term is quickly increasing for the *a priori* model, which is a strong instability indicator. The instabilities are indeed growing throughout time, as seen for the energy and enstrophy in Fig. 4.18. On the other hand, the *a posteriori* model performs extremely well on this long-term simulation and accurately reproduces the jet fronts magnitude of the direct simulation. The enstrophy spectrum and flux in Fig. 4.19 are similar to those of the previous configuration, except that linear damping has a stronger impact due to the relative increase in velocity from the beta-plane effect. Overall, while the dynamics in this configuration is not dominated by the modeled effect, numerical accumulation of small-scale enstrophy can still quickly lead to simulation blowup.

We summarize these results in Table 4.4 with the ℓ_{rms} and ℓ_∞ of the global modeled enstrophy transfers and enstrophy fluxes between the direct simulation and the different models for each configuration.

	ℓ_{rms}	ℓ_∞	ℓ_{rms}	ℓ_∞	ℓ_{rms}	ℓ_∞
$\langle\bar{\omega}\mathcal{M}\rangle$						
$\mathcal{M}^{M=25}_{\text{aposteriori}}$	**0.093**	**0.107**	**0.956**	**0.532**	**0.877**	**0.964**
$\mathcal{M}_{\text{apriori}}$	1.549	0.921	6.526	0.912	11.071	4.111
$\mathcal{M}_{\text{Smagorinsky}}$	2.686	3.359	10.300	5.305	13.285	8.727
$\mathcal{M}_{\text{Leith}}$	1.738	1.738	7.153	3.475	6.522	3.667
$\Pi_\omega(k)$						
$\mathcal{M}^{M=25}_{\text{aposteriori}}$	**0.050**	**0.504**	**0.018**	**0.125**	**0.134**	**0.099**
$\mathcal{M}_{\text{apriori}}$	0.227	0.993	0.349	0.194	3.816	2.643
$\mathcal{M}_{\text{Smagorinsky}}$	0.609	4.399	0.479	0.200	1.236	0.438
$\mathcal{M}_{\text{Leith}}$	0.521	3.579	0.159	0.263	0.726	0.365
	Decay		**Forced**		**Beta-plane**	

Table 4.4: ℓ_{rms} and ℓ_∞ on global modeled enstrophy transfers (first four rows) and enstrophy fluxes (last four rows) of the different models in each configuration.

4.3.2 Interpreting grid resolution

The described NN-based SGS models are trained and evaluated on a single grid resolution $\bar{N} = 128$ in both x and y directions. In practice, it is often desired to run simulations at multiple degrees of precision, represented by coarser or finer grids that corresponds to different filter size Δ' in SGS modeling. Here, we evaluate the ability of the *a posteriori* model trained with $\Delta' = \Delta'_{\text{train}} = 16$ to generalize to larger and smaller filter sizes $\Delta' = 4, 8, 32, 64$ in the **Forced** configuration. We run 5 new short simulations on the same temporal interval $[0, 51200\Delta t]$. Remember that the timestep of the reduced model scales with the filter size, i.e. $\overline{\Delta t} = \Delta'\Delta t$ so that models reach the end of the simulation with fewer iterations as Δ' increases. The numerical parameters of the QG solver remain identical. The results in Fig. 4.20 shows that the *a posteriori* is able to maintain good performance for $\Delta' < \Delta'_{\text{train}}$. This is not surprising since small structures modeled on small filter sizes are already visible at Δ'_{train}, and the statistical structure of turbulence is thus included in the model, w.r.t. to the enstrophy spectrum. Also, there are more resolved structures on finer grids, which simplifies the SGS modeling problem. Still note that the opposite is observed on the enstrophy fluxes for the physical models, where the increase is due to more timesteps taken, leading to more dissipation. Now, when $\Delta' > \Delta'_{\text{train}}$, the model is expected to extrapolate to a range of dynamics that have not been seen during training. We can indeed see in the global enstrophy transfers that the error increases for the *a posteriori* models, while physical baselines are able to maintain the same mismatch independently of the filter size. It seems possible to use transfer learning (Subel et al., 2021) to improve the generalization of the model to different grid resolutions without re-training it entirely.

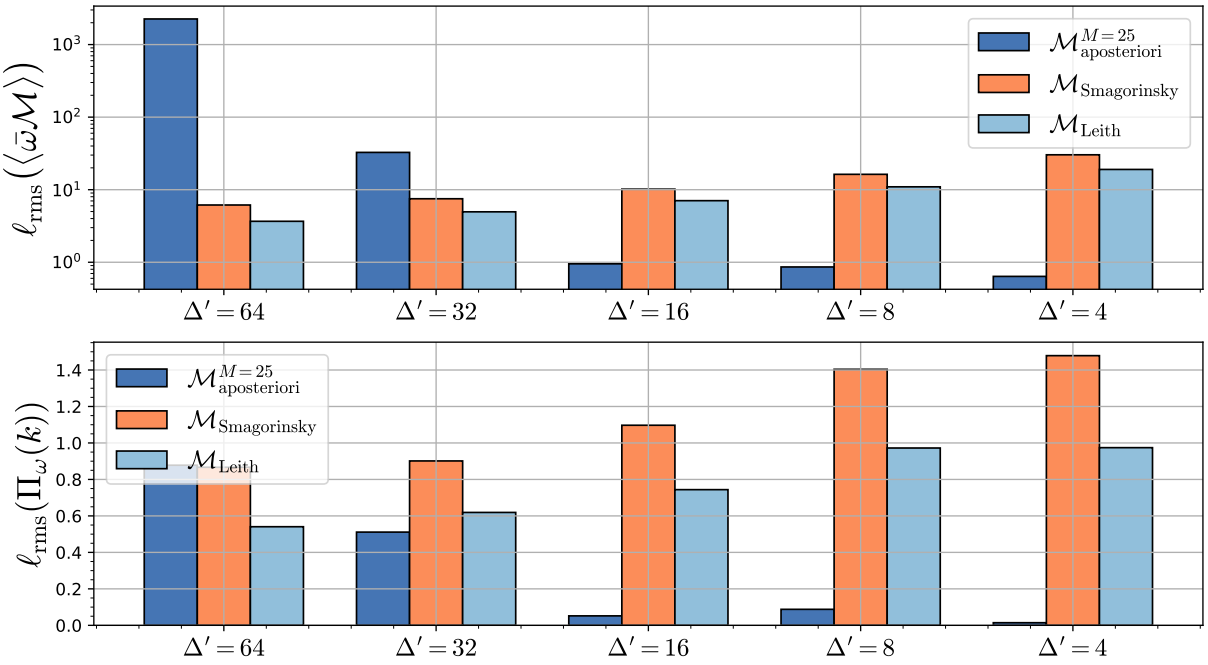

Figure 4.20: ℓ_{rms} on global modeled enstrophy transfers (top) and enstrophy fluxes (bottom) for different filter sizes Δ' in **FORCED** configuration.

Grid scaling hypothesis. The ability to scale to different filter sizes for baseline physical models has been derived by dimensional analysis, explicitly depending on the grid spacing Δ. We take inspiration and propose to formulate a similar model based on the previously trained *a posteriori* model,

$$\mathcal{M}^{\text{hypothesis}}_{\text{grid}}(\bar{\omega}, \bar{\psi}, \Delta) = \underbrace{\Delta^{\vartheta}}_{\text{grid scaling hypothesis}} \mathcal{M}^{M=25}_{\text{aposteriori}}(\bar{\omega}, \bar{\psi}) \tag{4.46}$$

where exponent $\vartheta \in \mathbb{R}$ is a trainable parameter. To make a connection with physical models, we have $\vartheta = 2$ for the Smagorinsky model and $\vartheta = 3$ for the Leith model. Note that the optimal correction for the base model will be $\Delta^{\vartheta} = 1$, which is only possible if $\vartheta = 0$. Instead, we can use the filter ratio in the model formulation,

$$\mathcal{M}^{\text{ratio}}_{\text{grid}}(\bar{\omega}, \bar{\psi}, \Delta) = \Delta_r^{\vartheta} \mathcal{M}^{M=25}_{\text{aposteriori}}(\bar{\omega}, \bar{\psi}) = \left(\frac{\Delta}{\Delta_{\text{train}}}\right)^{\vartheta} \mathcal{M}^{M=25}_{\text{aposteriori}}(\bar{\omega}, \bar{\psi}) \tag{4.47}$$

so that if $\Delta = \Delta_{\text{train}}$, then $\Delta_r^{\vartheta} = 1$ for any ϑ.

Arbitrary grid scaling. Since we do not know if the grid scaling hypothesis is valid in the context of NN-based models, we also formulate a generic grid scaling that non-linearly depends on the filter size. It writes,

$$\mathcal{M}^{\text{arbitrary}}_{\text{grid}}(\bar{\omega}, \bar{\psi}, \Delta) = \mathcal{M}_{\text{grid}}(\Delta \,|\, \theta) \mathcal{M}^{M=25}_{\text{aposteriori}}(\bar{\omega}, \bar{\psi}) \tag{4.48}$$

$$\mathcal{M}_{\text{grid}}(\Delta \,|\, \theta) := \text{Dense}^{(32)}(\theta_1) \circ \text{Relu} \circ \text{Dense}^{(64)}(\theta_2) \circ \text{Relu} \circ \text{Dense}^{(1)}(\theta_3).$$

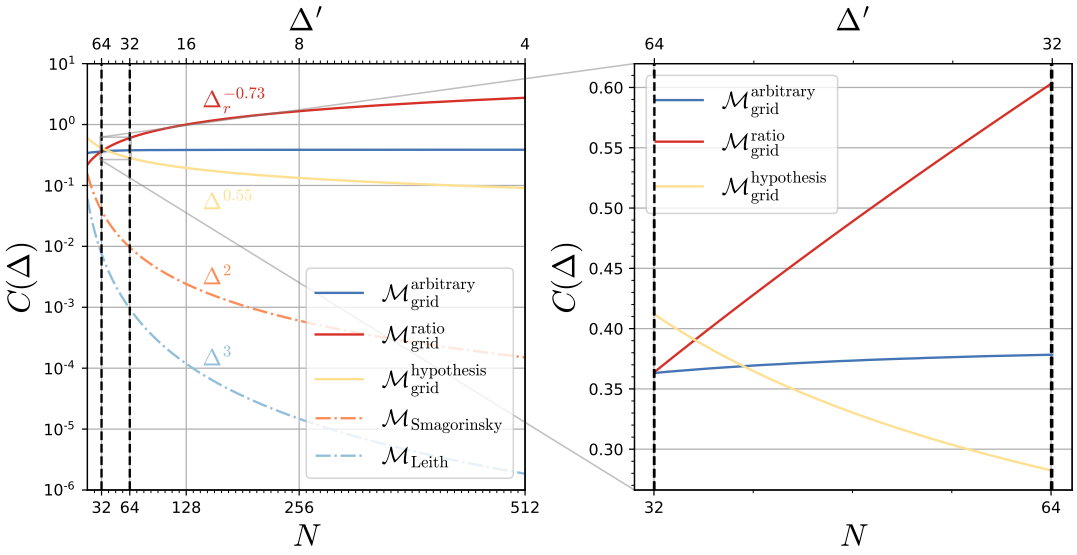

Figure 4.21: Evolution of the grid correction coefficient with respect to the number of grid points N (left) with a zoom on the specific NN-based proposed models (right).

Now, we can use the *a posteriori* learning strategy to optimize the trainable parameters of these three models. We used 3 trajectories from the difficult cases, i.e., coarse simulations with $\Delta' = 32$ and 64 so that the base case is not included and extrapolation to finer grids can be further verified. In each batch, we run the reduced models at different filter sizes in parallel and compute the loss based on their states. Again, we use $M = 25$ temporal steps during training and only 2 epochs were necessary for model convergence. The grid coefficients produced by the described variants are shown in Fig. 4.21 along with baseline physical models. Interestingly, the correction for NN-based models seems to converge between 0.3 and 0.6 at coarse grid sizes N, which is orders of magnitude larger than the scaling applied by physical models. Also, the contribution of the SGS model is expected to decrease as N increases and gets closer to the direct simulation resolution $\lim_{N\to\infty} \tau_\omega = 0$, but the opposite behavior is observed for the model based on Δ_r. While the constraint is verified, i.e., $C(\Delta) = 1$ when $\Delta = \Delta_{\text{train}}$, this forced the exponent to be negative in order to reach small values at a coarser grid resolution. We can see in Fig. 4.22 that the grid models make a large improvement when run on a coarser grid compared to the base *a posteriori* one. At $\Delta' = 64$, there is a slight advantage for the arbitrary model, for which the grid coefficient $C(\Delta) \approx 0.38$. On the other hand, the hypothesis model tends to accumulate some energy and may not be stable over longer timescales. It is also important to verify that these models are still stable and accurate on finer grids. As discussed above, we have seen that the base model already performs very well when $\Delta' < \Delta'_{\text{train}}$ and thus expect the coefficient $C(\Delta)$ predicted by the grid models to be close to 1. However, we see in Fig. 4.21 that the unconstrained models (hypothesis and arbitrary) are producing correction coefficients that are always smaller than 1. Surprisingly, Fig. 4.23 indicates

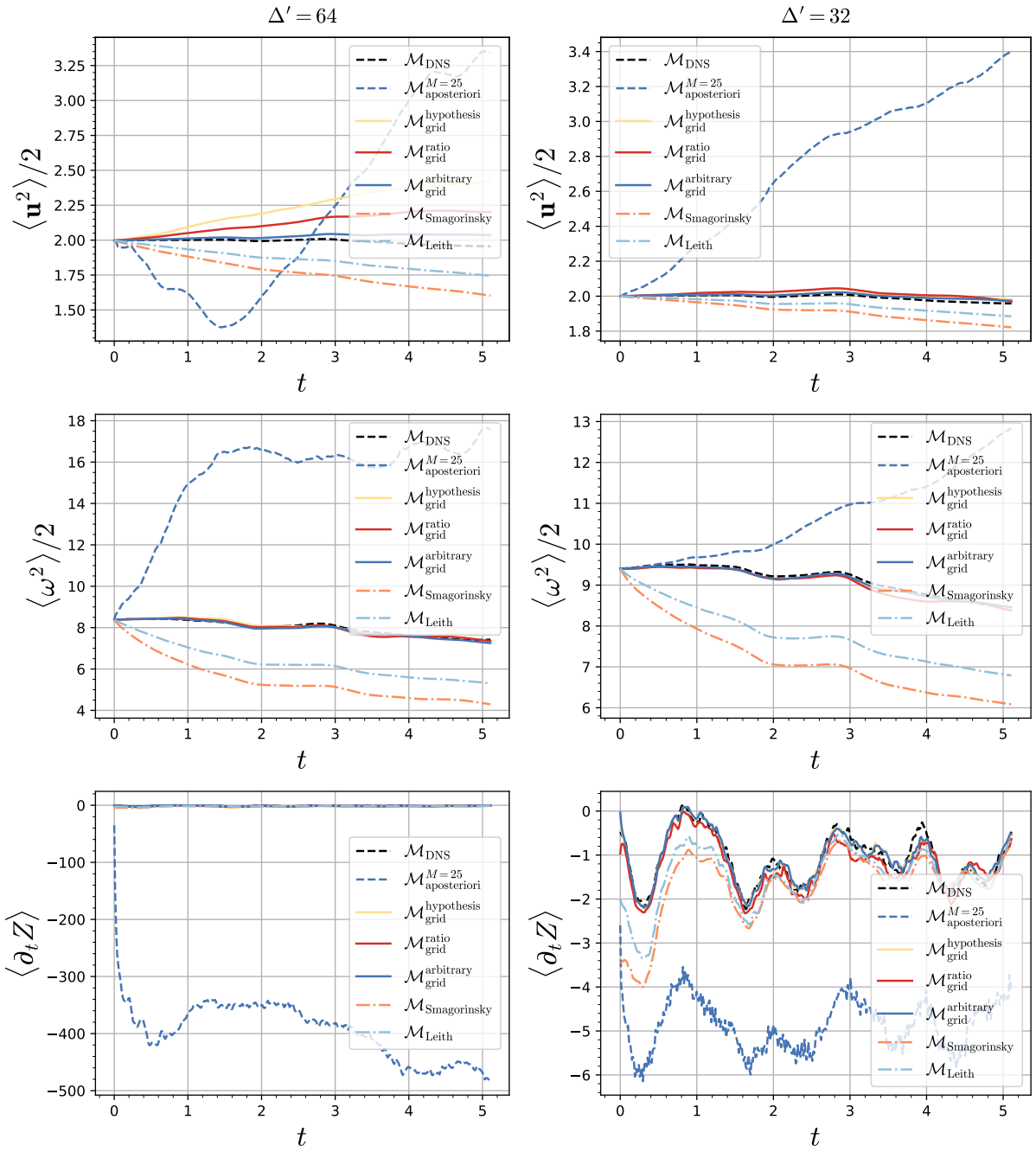

Figure 4.22: Evolution of domain-averaged energy (top), enstrophy (middle) and total enstrophy budget (bottom) on coarser grids $\Delta' > \Delta'_{\text{train}}$.

that $C(\Delta) < 1$ improves the performance of the models on the conservation of both quadratic invariants. On the other hand, a larger coefficient as produced by the ratio model is shown to be quite unstable, in particular on the finest grid at $\Delta' = 4$. These results seem to indicate that a simple constant is effectively able to "correct" the SGS dynamics already included in the base model. Also, and more importantly, extrapolation to unseen dynamics corresponding to larger scales of the turbulent spectrum is indeed possible, proportional to a corrective coefficient $C(\Delta)$. The improvement on finer grids

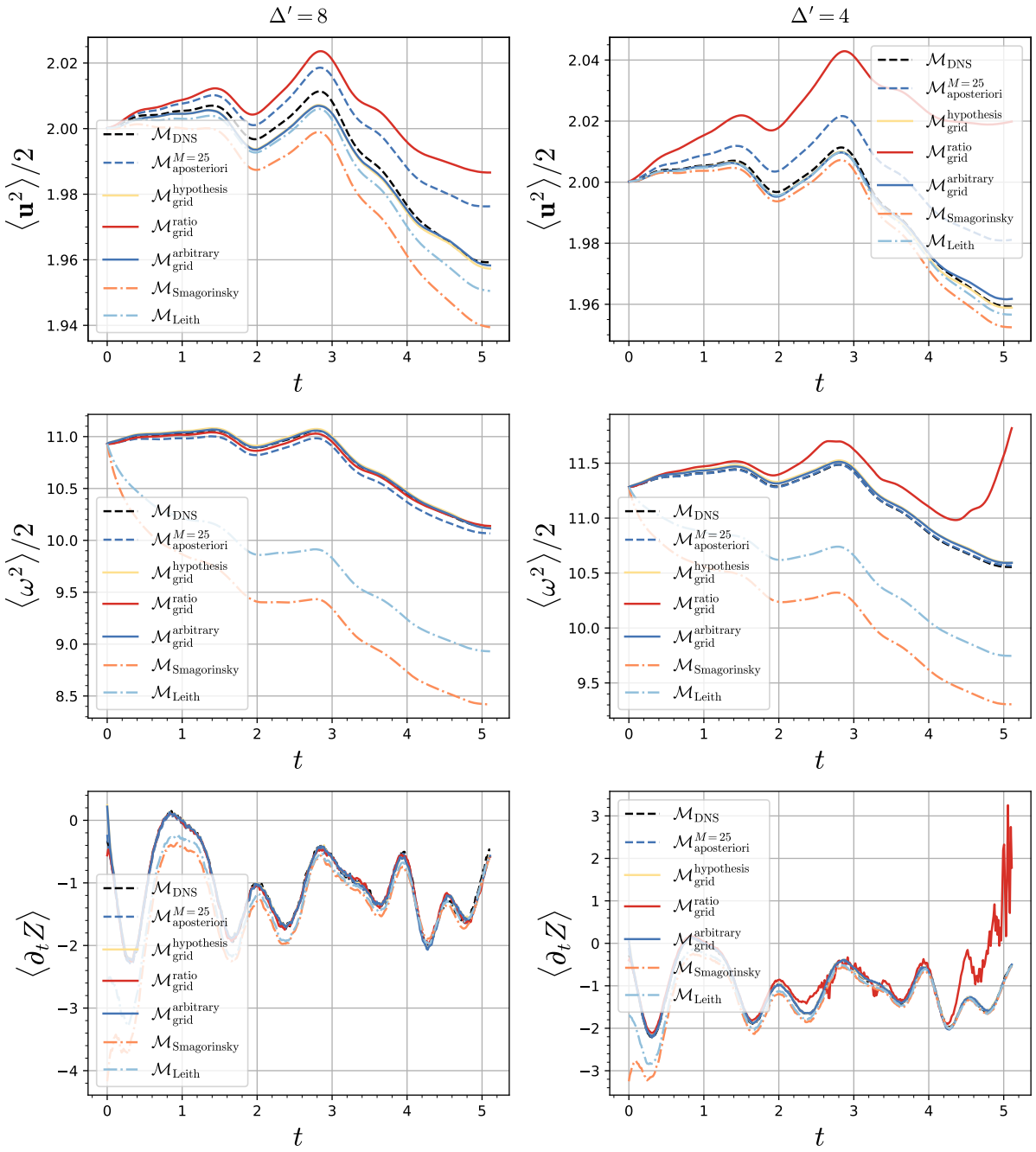

Figure 4.23: Evolution of domain-averaged energy (top), enstrophy (middle) and total enstrophy budget (bottom) on finer grids $\Delta' < \Delta'_{\text{train}}$.

is still not well understood, but it is possible that the dynamics learned by the base model was correcting resolved transfers. Having this information in training data, i.e. including fields from different resolutions might be important to improve the quality of the NN-based models. This experiment is also an empirical indicator that the dynamics of turbulence seems to be universal across grid resolutions. We finally show statistical metrics for the different grid models in Table 4.5, which further confirms that the grid correction improves the performance on coarser and finer grids.

	ℓ_{rms}	ℓ_{rms}	ℓ_{rms} (base)	ℓ_{rms}	ℓ_{rms}
$\langle\bar{\omega}\mathcal{M}\rangle$					
$\mathcal{M}^{M=25}_{\text{aposteriori}}$	2253.927	32.657	0.951	0.861	**0.638**
$\mathcal{M}^{\text{hypothesis}}_{\text{grid}}$	6.067	**0.864**	**0.812**	1.095	2.390
$\mathcal{M}^{\text{ratio}}_{\text{grid}}$	2.461	1.532	0.951	**0.715**	39.156
$\mathcal{M}^{\text{arbitrary}}_{\text{grid}}$	**0.715**	0.916	0.815	1.004	1.732
$\Pi_\omega(k)$					
$\mathcal{M}^{M=25}_{\text{aposteriori}}$	0.878	0.511	0.052	0.088	**0.014**
$\mathcal{M}^{\text{hypothesis}}_{\text{grid}}$	0.106	**0.020**	**0.035**	**0.013**	0.037
$\mathcal{M}^{\text{ratio}}_{\text{grid}}$	0.130	0.024	0.052	0.079	0.281
$\mathcal{M}^{\text{arbitrary}}_{\text{grid}}$	**0.065**	0.029	0.036	0.017	0.030
	$\Delta' = 64$	$\Delta' = 32$	$\Delta'_{\text{train}} = 16$	$\Delta' = 8$	$\Delta' = 4$

Table 4.5: ℓ_{rms} on global modeled enstrophy transfers (first four rows) and enstrophy fluxes (last four rows) of the different NN-based models.

4.4 Summary and implications

In this chapter, we investigated different learning strategies to train subgrid-scale (SGS) models for two-dimensional barotropic quasi-geostrophic turbulent flows. While the state-of-the-art has mostly explored *a priori* learning schemes, our numerical experiments stress the significant improvement brought by the *a posteriori* learning strategy to better reproduce small-scale dynamics on large temporal horizons with great accuracy. For all the flow configurations considered in this chapter, SGS models trained according to an *a posteriori* training loss clearly outperforms both physics-based and machine-learning baselines. The *a posteriori* learning strategy introduced in the chapter opens the possibility to design stable SGS models with more flexibility than state-of-the-art ML-based approaches. Indeed, we have here explored a relatively simple *a posteriori* training loss given by the vorticity MSE, but the *a posteriori* learning scheme offers much greater flexibility for the exploitation and combination of different *a posteriori* metrics during the learning phase. Losses defined from classical performance metrics such as energy transfers and distributions seem particularly appealing. One may also explore application-specific metrics including, e.g., in boundary layers, rotational or compressible flows. As the *a posteriori* learning strategy results in improved stability of the trained SGS models, it may also offer means to explore more complex neural architectures. Here, we considered a relatively simple ConvNet, but state-of-the-art architectures including for instance ResNet, UNet, and transformer networks could also be worth exploring. Another interesting avenue is the joint training of *a posteriori* models in the context of data assimilation such as described in (Bonavita and Laloyaux, 2020) and (Farchi et al., 2021).

Small-scale dynamics correction. An interesting connection can indeed be made between *a posteriori* learning and variational data assimilation techniques. Our *a posteriori* learning algorithm formulates a variational problem that is formally equivalent to the strong constraint 4D-Var scheme (Carrassi et al., 2018). In our case, however, the *control vector* is composed of the parameters of the neural network, and observations are assumed to be perfect. The analogy between *a posteriori* learning and 4D-Var, therefore, brings the question of whether models or more generally corrections to existing models, could be learned directly from sparse and noisy observations (Schneider et al., 2017b). In this sense, *a posteriori* learning is related to the bias correction methods that have been proposed in data assimilation (Dee, 2005), and especially the schemes proposed to infer state-dependent corrections to existing models (D'andrea and Vautard, 2000; Griffith and Nichols, 2000). Interestingly, this field has received renewed attention over recent years with several proposing to approach bias correction with ML. In this context, we stress that our approach is very similar to the scheme introduced in (Farchi et al., 2021), with the noticeable difference that we here learn a correction through the 4D-Var scheme itself, and not from the increment of the assimilation scheme.

Short-term metrics imply long-term performance. By construction, learning from the *a posteriori* strategy should improve the short-term forecasting capabilities of the models. This is true in particular if the training loss is defined as the sum of forecasting errors over a given temporal horizon as considered here. Interestingly, we noted that for SGS models, this short-term forecasting performance translates into better long-term stability and representation of long-term flow patterns where the long-term horizon is several orders of magnitude greater than the temporal horizon considered in the training loss (18000 vs 25 timesteps). While recent studies have explored neural models for the short-term forecasting of realistic geophysical flows, especially for weather forecasting applications (Schultz et al., 2021; Weyn et al., 2021), we believe our study opens new avenues for the exploitation of learning-based components in climate-scale simulations, which remain an open challenge (Rasp et al., 2018). In this respect, to account for the chaotic nature of turbulent flows, *a posteriori* training losses could also benefit from statistical metrics as opposed to synoptic ones as the MSE used in this work.

Solver differentiability. A strong requirement of the proposed framework lies in the differentiability of the considered dynamical solver, which may question its practical applicability. Indeed, most large-scale forward solvers in earth system models (ESM) rely on high-performance languages that do not embed automatic differentiation (AD) capabilities. While it is generally recognized that adjoints models are very useful additional tools for these solvers (Wunsch and Heimbach, 2013), adjoint operators are readily available only for a small fraction of them (Heimbach et al., 2005; Vidard et al., 2015). However, the emergence of a new generation of models written in differentiable programming lan-

guages or libraries such as Python's JAX (Häfner et al., 2018) or Julia's Zygote.jl and Enzyme.jl (Huang and Topping, 2021; Ramadhan et al., 2020; Sridhar et al., 2022) naturally supports our contribution. Besides, differentiable emulators (Hatfield et al., 2021; Kasim et al., 2021; Nonnenmacher and Greenberg, 2021) that learn a differentiable approximation of the non-differentiable forward solver or of its adjoint may also open new avenues for the development of SGS models for state-of-the-art ESMs with *a posteriori* learning strategies. This option will be explored in the next chapter.

Chapter 5

Emulation-based *a posteriori* learning of subgrid models in non-differentiable numerical solvers

In this chapter, we address the technical limitations of the *a posteriori* learning strategy described in chapter 4. While the method seems promising and outperforms physical models and NN-based models trained directly on their SGS term, it requires differentiable numerical solvers. We describe some technical solutions that try to solve this issue but are either too costly to maintain or not efficient enough to be used in practice. Finally, we propose to emulate the non-differentiable solver with a machine learning algorithm. This emulator can then be used to train a SGS model using an *a posteriori* approach. We show that while challenging, emulating large-scale dynamics with sufficient precision is possible for relatively complex systems such as QG turbulence with topography.

The results presented in this chapter will be submitted
Frezat et al. in prep.

Contents

5.1 Differentiable solvers

As discussed in chapter 4, the availability of a differentiable solver is very beneficial in the context of SGS modeling. Optimization based on the gradient of the numerical model has also been successfully applied to parameter tuning (Lyu et al., 2018; Tsai et al., 2021) and their related uncertainty quantification (Ge et al., 2018; Loose and Heimbach, 2021). Recall that for a typical regression task, one would seek to minimize a cost between some prediction $\mathring{\mathbf{z}} = f(\mathbf{y} \,|\, \theta)$ and ground truth of the same input space $\mathbf{z}$, i.e.

$$\arg\min_{\theta} \mathcal{L}(\mathbf{z}, f(\mathbf{y} \,|\, \theta)). \tag{5.1}$$

During the training process, the parameters of f, θ are optimized in order to minimize $\mathcal{L}$. The minimization algorithm requires the computation of the gradient of the loss function w.r.t. the trainable parameters θ,

$$\frac{\partial \mathcal{L}}{\partial \theta}(\mathbf{z}, f(\mathbf{y} \,|\, \theta)) = \frac{\partial f}{\partial \theta}(\mathbf{y} \,|\, \theta) \frac{\partial \mathcal{L}}{\partial f}. \tag{5.2}$$

In practice, f could be a simple operator from which the analytical gradient would be computed beforehand and provided to the optimization process. However, in differentiable solvers, we are interested in the temporal evolution of a system $E(\mathbf{y}) : \mathbb{R}^n \to \mathbb{R}^n$ of a vector-valued quantity of interest $\mathbf{y}(t)$. In a discretized formulation, the operator E is typically defined by a sequence of operations,

$$\mathbf{y}(t + \Delta t) = E_m \circ \cdots \circ E_1(\mathbf{y}(t)). \tag{5.3}$$

Now, if $f \equiv E$, the partial derivative of the system is defined as a Jacobian in the minimization formulation (5.2),

$$\frac{\partial E_i}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial E_{i,1}}{\partial y_1} & \cdots & \frac{\partial E_{i,1}}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial E_{i,n}}{\partial y_1} & \cdots & \frac{\partial E_{i,n}}{\partial y_n} \end{bmatrix} \tag{5.4}$$

for a single step of the temporal evolution operator E. Composing the partial derivatives of the sequences of operators in E can be done by applying the chain rule,

$$\frac{\partial E}{\partial \mathbf{y}} = \frac{\partial (E_m \circ \cdots \circ E_1)}{\partial \mathbf{y}} = \frac{\partial E_m}{\partial E_{m-1}} \cdots \frac{\partial E_2}{\partial E_1} \frac{\partial E_1}{\partial \mathbf{y}} \tag{5.5}$$

The gradient of E can quickly become difficult to maintain by hand. Note that when training NNs, it is required to have a scalar-valued loss function $\mathcal{L}$, which means that its derivative is a gradient instead of a Jacobian. The way partial derivatives are computed is called reverse-mode differentiation (Baydin et al., 2018), which here consists of a series of matrix-vector multiplications starting from

$$\frac{\partial E_1}{\partial \mathbf{y}} \frac{\partial \mathcal{L}}{\partial f} : \mathbb{R}^{m \times n} \to \mathbb{R}^m. \tag{5.6}$$

5.1.1 Technical solutions

The modern solution to computing (5.5) is referred to as automatic differentiation (AD), which gives the ability to compute numerical gradients from compositions of basic operators automatically using differentiability rules. AD is mostly available from rather high-level language libraries, such as Python or Julia. This has been a blocking factor since low-level optimization and manual memory management is quite important when implementing high-performance solvers. However, just-in-time (JIT) (Aycock, 2003) compilation has improved the capacity of these languages to generate efficient binaries. Some popular AD libraries include Python's JAX (Bradbury et al., 2018) and Julia's Zygote.jl (Innes et al., 2019). One side advantage of implementing a solver supporting AD is that it interfaces well with data-driven methods (Irrgang et al., 2021). Although it is now possible to implement efficient numerical solvers in languages that expose support for AD, most of the maintained codebases in geoscience remain in low-level languages, i.e., Fortran and C. Note that there is support for AD in these languages, e.g. (Bischof et al., 1997; Utke et al., 2008) but the solutions either require a large amount of modification in the original code, or generate memory inefficient gradient calculations (Margossian, 2019). When the solver is a large multi-year effort and using AD is not possible, due to various reasons enumerated above, it is still possible to manually derive and implement an adjoint solver, usually from a tangent linear model. This has been a common use for numerical weather prediction, where gradient-based optimization is also required in data assimilation schemes such as 4D-Var (Rabier et al., 1998). Manually deriving an adjoint solver can lead to optimal performances, but also requires a considerable amount of human work, and must be maintained alongside the forward solver. Some tools have been designed to analyze the forward code and generate an adequate adjoint code (Giering and Kaminski, 1998; Hascoet and Pascual, 2013). While these have been successfully applied to large solvers (Heimbach et al., 2002), they can generate code for hybrid (CPU and/or GPU) architectures and do not interact well with modern ML tools. We summarize the advantages and drawbacks of each of the different approaches in Table. 5.1.

5.1.2 *a posteriori* learning without a differentiable solver

Going back to the SGS modeling problem (4.18) and its *a posteriori* minimization formulation (4.25), the temporal evolution of the system is a combination of a classical numerical solver g and a trainable SGS model $\mathcal{M}$,

$$f(\bar{\mathbf{y}} \,|\, \theta) \equiv \bar{\mathbf{y}}(t_0) + \int_{t_0}^{t_1} g(\bar{\mathbf{y}}(t)) + \mathcal{M}(\bar{\mathbf{y}}(t) \,|\, \theta)\, \mathrm{d}t = \bar{\mathbf{y}}(t_1). \tag{5.7}$$

Here, g is a discretization of the considered dynamical equation with a spatial scheme such as finite differences or a pseudo-spectral method, as used in chapter 4. Training

		Accuracy	Efficiency	Maintenance	Migration	Ecosystem
Manual	(Moore et al., 2004)	+	+	−	+	∼
Numerical		−	−	+	+	∼
Codegen	(Aycock, 2003)	+	−	+	+	∼
AD	(Bradbury et al., 2018)	+	+	+	−	+

Table 5.1: Summary of the different existing methods to compute a partial derivative (adjoint). Note that AD often requires rewriting code in a corresponding framework, but most of them live in a well-developed ML ecosystem.

for $\mathcal{M}$ involves the minimization problem (5.2) for the temporal evolution operator f. This in turn requires the computation of a complex chain of Jacobians (5.5) that is too complicated to maintain by hand. If we assume that in a NN framework, $\mathcal{M}$ is a differentiable model, then the limitation is only caused by the numerical solver g, not being differentiable. Expanding (5.2) with (5.7), we have

$$\frac{\partial f}{\partial \theta}(\bar{\mathbf{y}} \,|\, \theta)\frac{\partial \mathcal{L}}{\partial f} = \left(\int_{t_0}^{t_1} \underbrace{\frac{\partial g(\bar{\mathbf{y}}(t))}{\partial \theta}}_{\text{unknown}} + \underbrace{\frac{\partial \mathcal{M}(\bar{\mathbf{y}}(t) \,|\, \theta)}{\partial \theta}}_{\text{known}} \, \mathrm{d}t \right) \frac{\partial \mathcal{L}}{\partial f} \tag{5.8}$$

where g is the only missing part for allowing *a posteriori* learning. The goal of this chapter is to explore NN-based alternatives to approximate the numerical solver g as a differentiable emulator but also to understand how the emulator error for g will impact the performance on $\mathcal{M}$. We are trying to design,

$$\mathcal{E}(\bar{\mathbf{y}}) \approx g(\bar{\mathbf{y}}), \quad \text{s.t.} \underbrace{\frac{\partial \mathcal{E}}{\partial \theta}}_{\text{known}} \tag{5.9}$$

which can also be defined as a trainable NN-based emulator, benefiting again from differentiable capabilities from the NN framework,

$$\mathcal{E}(\bar{\mathbf{y}} \,|\, \Theta) \approx g(\bar{\mathbf{y}}). \tag{5.10}$$

The trainable weights Θ can be learned simultaneously with the weights of the SGS model θ, but it would not be possible to ensure that $\mathcal{M}$ is only accounting for the SGS term,

$$\arg\min_{\Theta,\theta} \ell(\{\mathcal{T}(\mathbf{y}(t))\}_{t\in[t_0,t_1]}, \{\bar{\mathbf{y}}(t_0) + \int_{t_0}^{t} \mathcal{E}(\bar{\mathbf{y}}(t') \,|\, \Theta) + \mathcal{M}(\bar{\mathbf{y}}(t') \,|\, \theta)\, \mathrm{d}t'\}_{t\in[t_0,t_1]}), \tag{5.11}$$

except if the loss function $\mathcal{L}$ correctly penalizes both terms separately,

$$\mathcal{L} \equiv \mathcal{L}_{\mathcal{E}}(\mathcal{E}(\bar{\mathbf{y}} \,|\, \Theta)) + \mathcal{L}_{\mathcal{M}}(\mathcal{M}(\bar{\mathbf{y}} \,|\, \theta)). \tag{5.12}$$

5.2 Differentiable emulators for subgrid modeling

We emphasize the fact that we are looking for a differentiable emulator, but the final goal is to effectively learn a SGS model that accounts for the unresolved scales in a simulation. It has been demonstrated that the *a posteriori* learning strategy outperforms the common *a priori* strategy and is remarkably stable over long-term simulations. To be usable in practice, we would like to be able to train a SGS model using this strategy without having to migrate an entire solver to an AD framework. The goal is thus to design a SGS model,

$$\mathcal{T}(g(\mathbf{y})) - g(\mathcal{T}(\mathbf{y})) = \tau \approx \mathcal{M}(\bar{\mathbf{y}} \,|\, \theta). \tag{5.13}$$

As explained above, *a posteriori* learning requires the temporal integration of the system (5.7), but here g is replaced with an emulator $\mathcal{E}(\bar{\mathbf{y}}) \approx g(\bar{\mathbf{y}})$ such that

$$\bar{\mathbf{y}}(t_0) + \int_{t_0}^{t_1} \mathcal{E}(\bar{\mathbf{y}}(t)) + \mathcal{M}(\bar{\mathbf{y}}(t) \,|\, \theta)\, \mathrm{d}t \approx f(\bar{\mathbf{y}} \,|\, \theta). \tag{5.14}$$

It is clear from this equation that the performance of the SGS model $\mathcal{M}$ is likely to depend on how well the emulator $\mathcal{E}$ approximates the numerical solver g.

5.2.1 Numerical setup and required accuracy

We know that the *a posteriori* learning strategy is beneficial in turbulent systems where backscatter has an important impact on the dynamics of the flow. Going one step further, we may be interested in flows exhibiting forming, merging, and splitting jets as fixed by topographical features. Such a system can be described by the quasi-geostrophic equations with bottom topography η,

$$\begin{cases} \dfrac{\partial \omega}{\partial t} + N_{\omega+\eta} = \nu \nabla^2 \omega - \mu \omega + F_\omega, & \omega \in \Omega \\ \dfrac{\partial \bar{\omega}}{\partial t} + N_{\bar{\omega}+\bar{\eta}} = \nu \nabla^2 \bar{\omega} - \mu \bar{\omega} + F_{\bar{\omega}} + \underbrace{N_{\bar{\omega}+\bar{\eta}} - \bar{N}_{\omega+\eta}}_{\tau_{\omega+\eta}}, & \bar{\omega} \in \bar{\Omega} \end{cases} \tag{5.15}$$

where $F_{\bar{\omega}}$ is the wind-forcing described in (4.38) and η is made of unaligned sinusoidal bumps as described in (Thompson, 2010),

$$\eta = 5(\sin(3y) + \cos(3x)). \tag{5.16}$$

Note that wind and topography forcings are only applied at large scale $k = 4$ in order to separate their effect from the SGS term $\tau_{\omega+\eta}$. The corresponding terms for the solver emulator and SGS model are thus,

$$\mathcal{E}(\bar{\mathbf{y}}) \equiv \nu \nabla^2 \omega - \mu \omega + F_\omega - N_{\omega+\eta}, \tag{5.17}$$

$$\mathcal{M}(\bar{\mathbf{y}}) \equiv \tau_{\omega+\eta}. \tag{5.18}$$

Applying the learning algorithms described in chapter 4 on this configuration results in larger instabilities that occur rapidly for the *a priori* strategy, while the *a posteriori* model remains stable on this temporal horizon (see Fig. 5.1). The flow produced in this configuration has more complex dynamics with many interactions between the horizontal jets. It is clear here that the *a priori* strategy without any special treatment will not be applicable to realistic geophysical flows. Again, most of the weather and climate solvers are developed using non-differentiable languages and thus won't allow the direct application of *a posteriori* learning. The challenge is thus to approach as close as possible the results obtained by the *a posteriori* strategy without using solver differentiability.

Controled emulation error. Before trying to learn differentiable emulators, we want to understand the precision required by the emulator to train a stable and accurate SGS model with the *a posteriori* strategy. We know that convolutions, which will be used in the emulator architectures can be seen as discrete partial derivative approximations on a stencil corresponding to their kernel width. In the numerical solver, spatial derivatives are computed using a spectral method, as described in 2.3 and might be challenging to reproduce with a NN (Li et al., 2020). Interestingly, it is possible to reproduce finite-difference (FD) schemes by modifying the wavenumbers k in a Fourier spectral method (Kravchenko and Moin, 1997). An error related to the discrete approximation of partial derivatives is often called *truncation* error. These errors can be generated from modified wavenumbers corresponding to a particular numerical scheme, e.g., for a FD approximation of the first derivative of a differentiable function $f(\mathbf{y}(\mathbf{x}))$,

$$\widehat{\frac{\delta f}{\delta x}} = i \underbrace{k'(k)}_{\text{truncation error}} \hat{f}. \tag{5.19}$$

We replace the partial derivatives calculations in the pseudo-spectral solver with modified wavenumbers for three different FD schemes. We refer to the new solvers as δ^2, δ^4 and δ^6 for a 2nd-order, 4th-order central FD schemes and a 6th-order Padé scheme, respectively. During *a posteriori* training, the numerical solvers are integrated for $M = 25$ temporal steps. Thus, the emulator is expected to reproduce an acceptable approximation of the pseudo-spectral solver that holds for at least 25 steps. Here, we run reduced versions, i.e. $\bar{N} = 128$ of the numerical solver and the FD emulators of different order without SGS terms, $\tau_{\omega+\eta} = 0$. We then compute the ℓ_{rms} between the reduced reference (pseudo-spectral, δ^∞) and the FD emulators, in addition to the SGS error between filtered direct simulation and pseudo-spectral reference (see Fig. 5.2). The first important observation is that the error on the velocity $\bar{\mathbf{u}}$ is smaller than the error on the vorticity $\bar{\omega}$ by one order of magnitude. It is not surprising since the FD error is mostly located on the largest wavenumbers, corresponding to small-scale features that are more visible in the vorticity fields. After 25 iterations, the error of the 6th-order scheme is slightly larger than the SGS

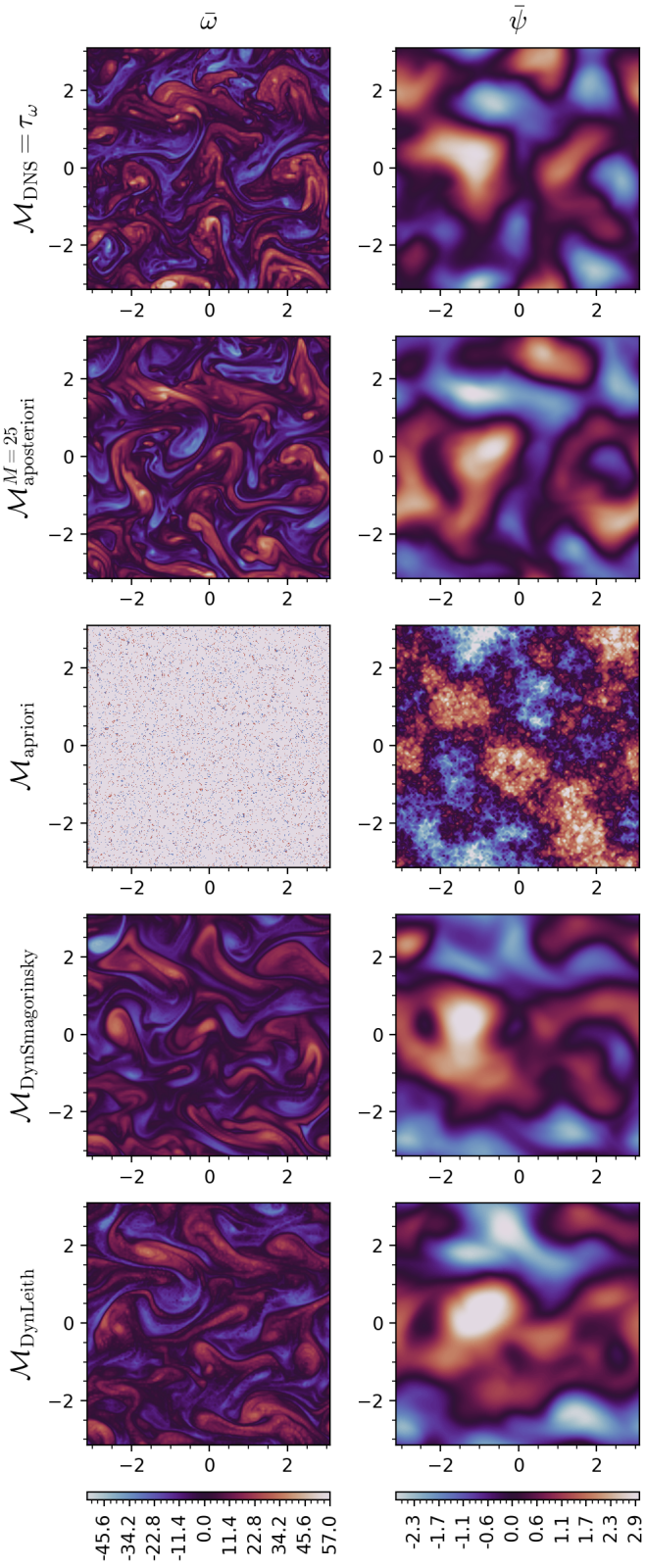

Figure 5.1: Reduced fields for vorticity $\bar{\omega}$ (left) and streamfunction $\bar{\psi}$ (right) at the end of a testing horizon (3000 temporal iterations) from the direct simulation ($\tau_{\omega+\eta}$, top) and the different models for the **TOPOGRAPHY** configuration.

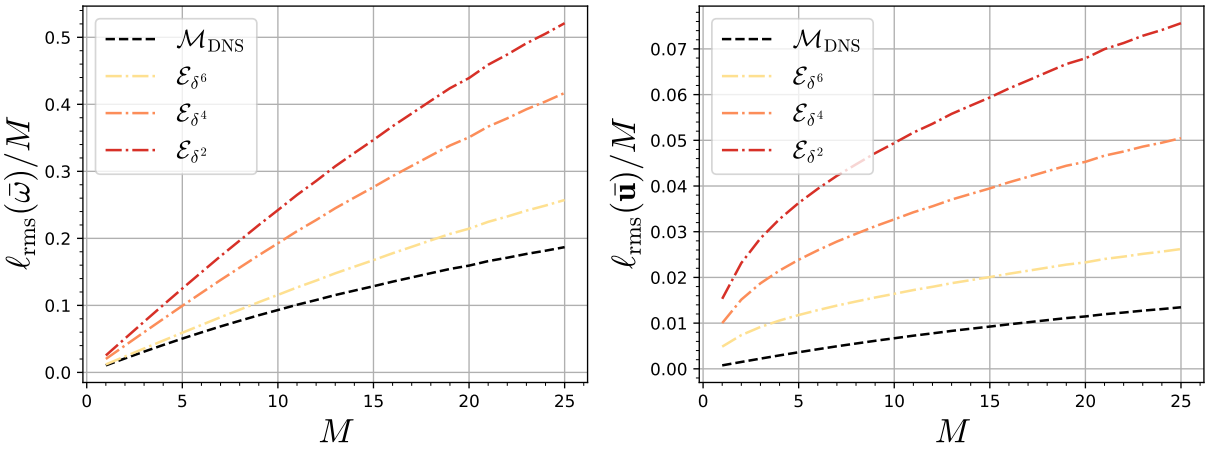

Figure 5.2: ℓ_{rms} normalized by the number of temporal integration steps M for the vorticity ω (left) and velocity $\mathbf{u}$ (right). FD emulators $\mathcal{E}$ are compared to the pseudo-spectral δ^∞ reference numerical solver. The error accounting for missing SGS term between δ^∞ and filtered DNS is shown in dashed black for comparison.

error on the vorticity (≈ 0.26 against ≈ 0.19) while the error on the 2nd-order scheme is much larger (≈ 0.52). We observe a large error difference in the velocity, already twice as large for the 6th-order scheme compared to the SGS error contribution, up to 7 times larger for the 2nd-order scheme. To better understand the spatial error distribution of the FD emulators, we also show in Fig. 5.3 the difference between pseudo-spectral vorticity fields $\bar{\omega}$, enstrophy $Z = \bar{\omega}^2$ and kinetic energy $E = \bar{u}^2 + \bar{v}^2$. The magnitude of the errors is in agreement with the above discussion, but we can see here that the largest errors are aligned with vorticity gradients.

Emulator error impact on the SGS model. Now that we have analyzed the FD errors, we assess the impact of this emulation error on the ability to train a stable SGS model using the *a posteriori* strategy. Assuming that the FD solvers are differentiable, we can train SGS models using the same data and parameters as described in chapter 4. Note that the models are expected to reproduce the unresolved dynamics but will also inevitably try to correct the error of the emulator, i.e. spatial discretization in this case. We can see in Fig. 5.4 that we SGS models can improve the discretization error and almost reach pseudo-spectral performance on the vorticity. However, the larger mismatch happening on the velocity is not improved by the SGS model. It might be interesting to include a measure of the error between velocity fields in the loss function in order to improve the large-scale predictions. We suspect that the trained models are learning a different SGS term, i.e.

$$\mathcal{M}(\bar{\omega}) \approx \tau_{\omega+\eta} + (g(\bar{\omega}) - \mathcal{E}(\bar{\omega})) \tag{5.20}$$

where $g(\bar{\omega}) - \mathcal{E}(\bar{\omega})$ is the bias between the pseudo-spectral reference solver and the FD emulators. Depending on this error term, the SGS may compensate for an error that

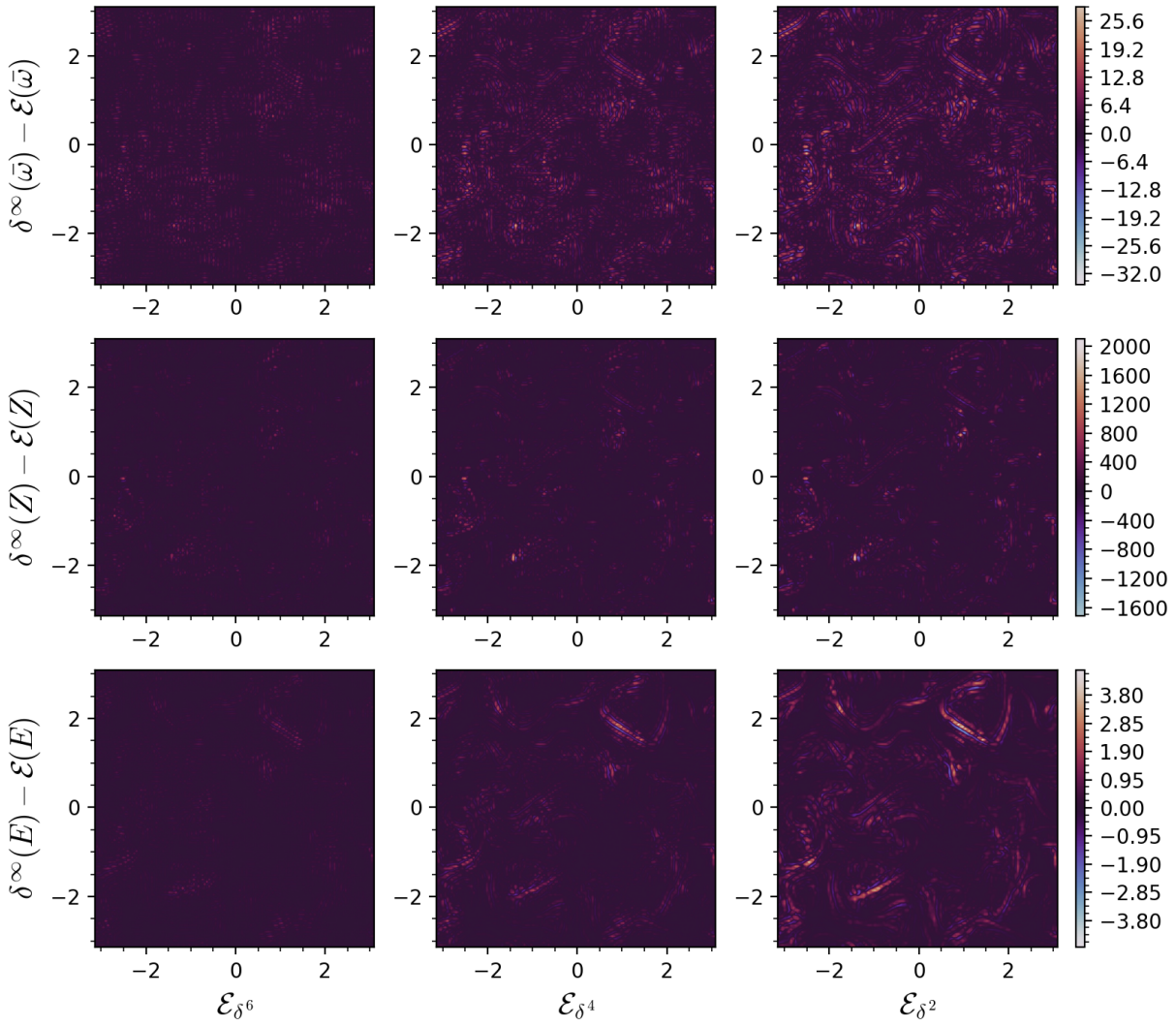

Figure 5.3: Difference between reduced pseudo-spectral reference and finite-difference emulators for vorticity (top), enstrophy (middle), and kinetic energy (bottom) after $M = 25$ temporal steps.

does not exist anymore when it is run with the exact numerical solver. In practice, the evolution of the quadratic invariants in Fig 5.5 draw some contradiction with respect to the ℓ_{rms} plots and show that both kinetic energy and enstrophy increase at a similar rate. We see that the stability of the SGS models is closely related to the order of the FD emulator used for training, with stable behavior for the 6th-order scheme and blowup due to instabilities for the 2nd-order scheme. The spectral statistics shown in Fig. 5.6 are particularly satisfying for the 6th-order scheme. It is important to mention that most of the error is located on the largest scales, which is in agreement with the potential velocity error improvements discussed above. Note that these results already indicate that it is possible to learn a stable SGS model from a deteriorated emulator using the *a posteriori* strategy.

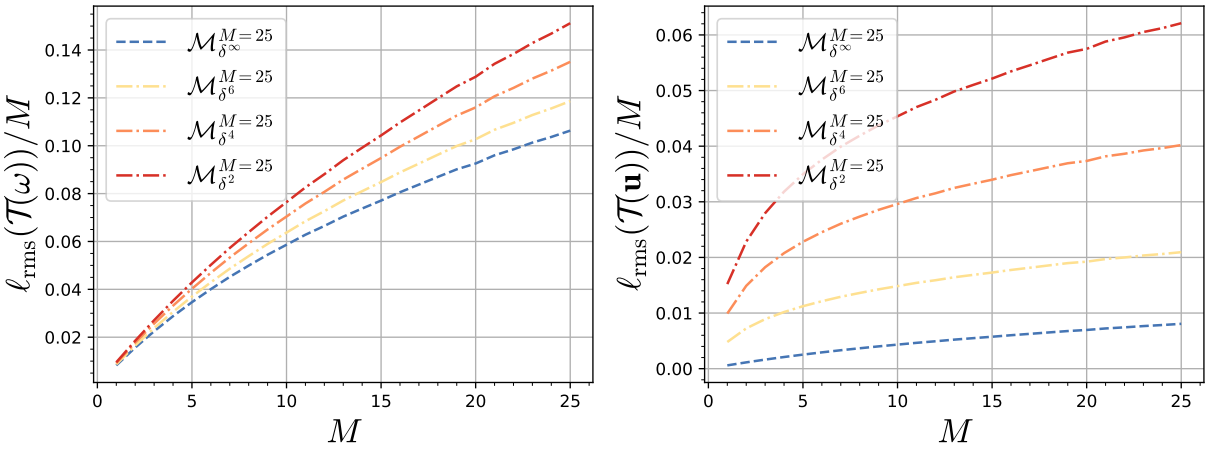

Figure 5.4: ℓ_{rms} normalized by the number of temporal integration steps M for the vorticity ω (left) and velocity $\mathbf{u}$ (right). Simulations are run with pseudo-spectral δ^∞ solver or a finite-difference emulator and their respective SGS model so that they don't include the error bias (5.20).

5.2.2 An algorithm for trainable emulators

In the previous experiment that aims at understanding the emulator error and its impact during the training phase of the SGS model, we used the modified differentiable pseudo-spectral solver. Now, we want to learn a differentiable emulator representation from a NN. Building on the successful application of the *a posteriori* strategy, we propose an algorithm that trains both emulator and SGS model using an end-to-end approach. As explained in (5.11), jointly training the $\mathcal{E}$ and $\mathcal{M}$ might lead to ambiguities except if loss functions are penalizing each term separately. Using a single *a posteriori* loss, it is possible to train $\mathcal{E}$ and $\mathcal{M}$ sequentially by applying the following iterative algorithm,

Iterative step 1 (emulator training) :

$$\arg\min_{\Theta} \ell(\{\Phi_\Theta^{t(i)}(\bar{\mathbf{y}}(t_0))\}_{t\in[t_0,t_1]}, \{\bar{\mathbf{y}}(t) + \mathcal{M}_i(\bar{\mathbf{y}}(t))\}_{t\in[t_0,t_1]}), \tag{5.21}$$

Iterative step 2 (subgrid-scale training) :

$$\arg\min_{\theta} \ell(\{\Phi_\theta^{t(i)}(\bar{\mathbf{y}}(t_0))\}_{t\in[t_0,t_1]}, \{\mathcal{T}(\mathbf{y}(t))\}_{t\in[t_0,t_1]}). \tag{5.22}$$

where the flow operators Φ for emulator and SGS parameters Θ and θ are defined respectively at iteration i,

$$\Phi_\Theta^{t(i)}(\bar{\mathbf{y}}(t_0)) = \bar{\mathbf{y}}(t_0) + \int_{t_0}^{t} \mathcal{E}_{i+1}(\bar{\mathbf{y}}(t') \,|\, \Theta) + \mathcal{M}_i(\bar{\mathbf{y}}(t'))\, \mathrm{d}t', \tag{5.23}$$

$$\Phi_\theta^{t(i)}(\bar{\mathbf{y}}(t_0)) = \bar{\mathbf{y}}(t_0) + \int_{t_0}^{t} \mathcal{E}_{i+1}(\bar{\mathbf{y}}(t')) + \mathcal{M}_{i+1}(\bar{\mathbf{y}}(t') \,|\, \theta)\, \mathrm{d}t'. \tag{5.24}$$

Here, we can see that the right-hand side of (5.21) requires the current iteration of the SGS model $\mathcal{M}_i$, which cancels with the flow (5.23) and avoids learning a corrective bias in the

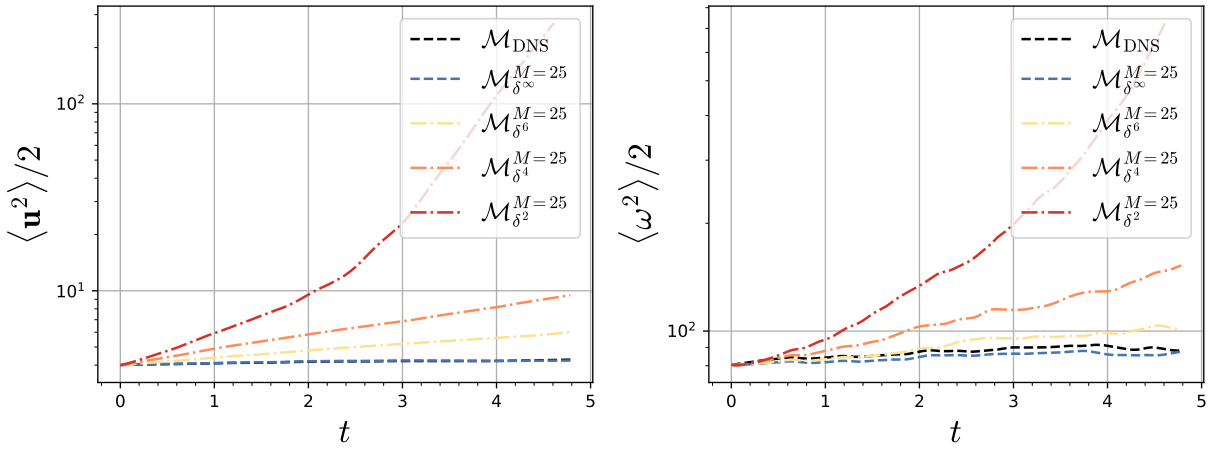

Figure 5.5: Evolution of domain-averaged energy (left) and enstrophy(right) of the finite-difference discretization models for 3000 reduced temporal steps.

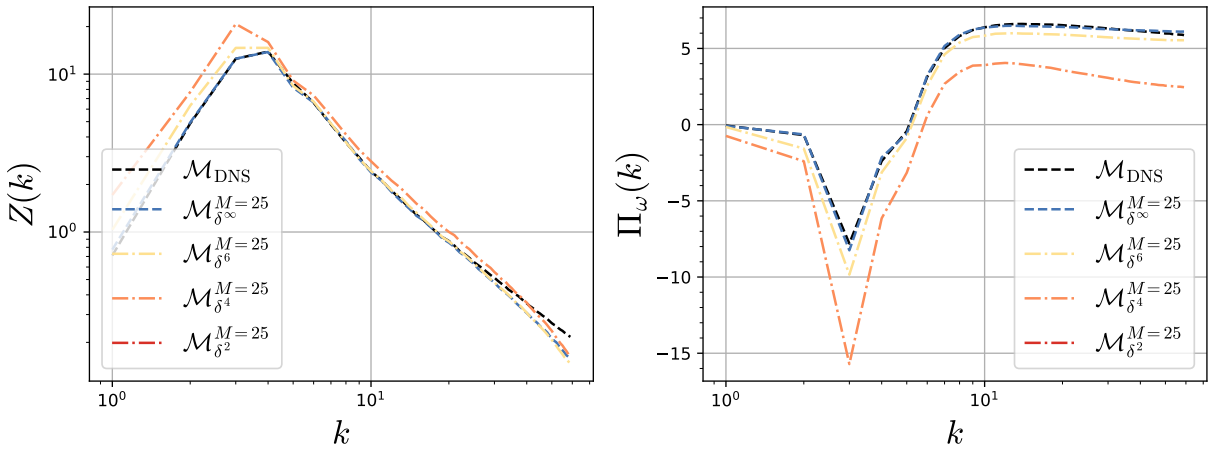

Figure 5.6: Time-averaged enstrophy spectrum (left) and enstrophy flux (right) of the finite-difference discretization models for 3000 reduced temporal steps.

emulator. The iterative algorithm has to be initialized at iteration 1 with a differentiable SGS model $\mathcal{M}_0$. It is possible to start with a model trained using the *a priori* strategy (4.20), or explicitly implement a differentiable physical SGS model, such as those described in 2.1.4. The intuition behind this iterative algorithm is that the dynamics diversity might gradually improve the performance of the emulator $\mathcal{E}_i$, which in turn propagates to the SGS model $\mathcal{M}_i$. Note however that this is extremely expansive from a computational point of view. In our experiments, we simplify the above two-step algorithm by removing

the SGS term in the emulation training,

Direct step 1 (emulator training) :

$$\arg\min_{\Theta} \ell(\{\varphi_{\Theta}^{t(i)}(\bar{\mathbf{y}}(t_0))\}_{t\in[t_0,t_1]}, \{\bar{\mathbf{y}}(t)\}_{t\in[t_0,t_1]}), \tag{5.25}$$

Direct step 2 (subgrid-scale training) :

$$\arg\min_{\theta} \ell(\{\varphi_{\theta}^{t(i)}(\bar{\mathbf{y}}(t_0))\}_{t\in[t_0,t_1]}, \{\mathcal{T}(\mathbf{y}(t))\}_{t\in[t_0,t_1]}). \tag{5.26}$$

where it is now clear that step 1 is dedicated to the emulator, and step 2 to the direct system. One concern is that reduced simulations without SGS terms are likely to accumulate small-scale energy and blowup, which limits the applicability of this algorithm to a small number of temporal iterations. Still, we have seen previously that $M = 25$ is enough to train a stable SGS model, which is in the stability range of a reduced solver run without SGS model. The SGS flow operator φ_Θ is unchanged while changes are reflected in the emulator flow φ_θ, only integrating $\mathcal{E}$ in isolation,

$$\varphi_{\Theta}^{t(i)}(\bar{\mathbf{y}}(t_0)) = \bar{\mathbf{y}}(t_0) + \int_{t_0}^{t} \mathcal{E}(\bar{\mathbf{y}}(t') \,|\, \Theta)\, \mathrm{d}t', \tag{5.27}$$

$$\varphi_{\theta}^{t(i)}(\bar{\mathbf{y}}(t_0)) = \bar{\mathbf{y}}(t_0) + \int_{t_0}^{t} \mathcal{E}(\bar{\mathbf{y}}(t')) + \mathcal{M}(\bar{\mathbf{y}}(t') \,|\, \theta)\, \mathrm{d}t'. \tag{5.28}$$

With this algorithm, we can train the differentiable emulator $\mathcal{E}$ once and then use it to train $\mathcal{M}$. A second concern with this approach is that the optimization algorithm for the emulator will only see data that tend towards numerical instabilities. Step 1 is similar to approaches that have been explored to fully replace numerical schemes for turbulent simulations with purely learned components. Several architectures have been successfully applied such as multi-scale convolutional neural networks (Wang et al., 2020), graph neural networks (Sanchez-Gonzalez et al., 2020), Fourier neural operators (Li et al., 2020) or dilated residual networks (Stachenfeld et al., 2021). For the specific application to the geophysical system (5.15), it is equivalent to substracting the SGS term $\tau_{\omega+\eta}$ from the numerical solver g,

$$\mathcal{E}(\bar{\omega} \,|\, \Theta) \approx g(\bar{\omega}) - \tau_{\omega+\eta} = \nu\nabla^2\bar{\omega} - \mu\bar{\omega} + F_\omega - N_{\bar{\omega}+\bar{\eta}}, \quad \bar{\omega} \in \bar{\Omega}. \tag{5.29}$$

Note that here we naturally chose the vorticity field as input of the emulator since we are emulating the NS equations in vorticity formulation. Consequently, the SGS model can not use streamfunction as input since the emulator does not know how to provide this quantity. It is possible however to additionally train the emulator to predict the couple $(\bar{\omega}, \bar{\psi})$ but this might introduce new sources of error. Instead, a SGS model trained from a differentiable emulator only uses vorticity fields,

$$\mathcal{M}(\bar{\omega} \,|\, \theta) \approx \tau_{\omega+\eta}, \quad \bar{\omega} \in \bar{\Omega}. \tag{5.30}$$

Hyperparameters	CNN	DRN	FNO
Trainable params.	616064	580704	102246977
Evaluation time	1.0	11.81	13.32
Kernel size	5	3	-
Latent size	64	48	32
Depth	7	7	6
Activation	Relu	Relu	Relu
Convolution stack depth	7		
Dilated block depth		7	
Dilations		(1, 2, 4, 8, 4, 2, 1)	
Dilated blocks		4	
Fourier layers			6
Modes			128, 65 (full)

Table 5.2: Model hyperparameters for the different neural architectures used for emulation training.

NN architectures and learning schemes. To represent the large-scale dynamics (5.29), we explore different neural architectures (see Table 5.2). We use a simple fully convolutional architecture, as well as several baselines that have been used in different PDE integration problems. Note that these neural models are "fully learned" without any hard-coded physical components. For the SGS model (5.26), we use the same convolutional architecture described (4.39) in the previous chapter.

Convolutional Neural Network (CNN). We simplify the fully convolutional architecture (CNN) described in (4.39) to 7 convolutional blocks instead of 10.

Dilated Residual Network (DRN). We implement a state-of-the-art model that combines residual networks with dilated convolutions, recently applied to different types of flows (Stachenfeld et al., 2021). This architecture effectively combines the encode-process-decode paradigm (Sanchez-Gonzalez et al., 2018) with dilated convolutions. In our application, the input vorticity is encoded into a 48-feature tensor, processed by 4 residual blocks each containing 7 dilated convolution blocks, and finally decoded back to a 1-dimensional feature tensor.

Fourier Neural Operator (FNO). We also implement a state-of-the-art model that directly learns the integral kernel in Fourier space recently applied to different fluid dynamics PDEs (Li et al., 2020). In our application, we use all the modes of the input vorticity, i.e. without truncation, and apply 6 Fourier layers.

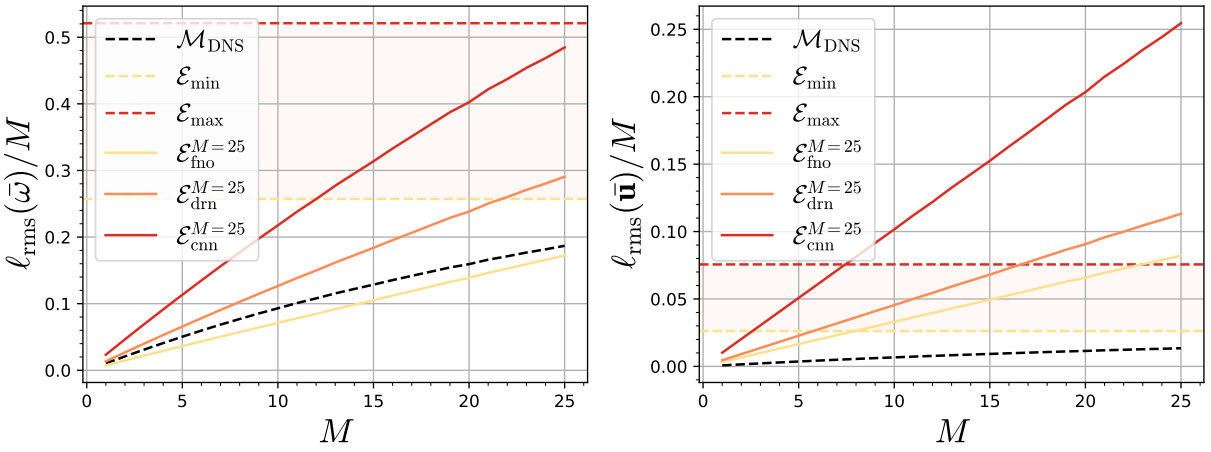

Figure 5.7: ℓ_{rms} normalized by the number of temporal integration steps M for the vorticity ω (left) and velocity $\mathbf{u}$ (right). Reduced fields for trained differentiable emulators $\mathcal{E}$ are compared to the pseudo-spectral δ^∞ reference numerical solver.

5.3 Trained emulator performance

We first evaluate in Fig. 5.7 the performance of the trained emulator in isolation, i.e. without SGS terms against the reference pseudo-spectral solver. We notice that the emulators are able to learn an accurate representation of the reduced vorticity fields $\bar{\omega}$ and perform better than the FD emulators, even for the simple CNN. In particular, the FNO produces an error that is smaller than the error related to the SGS term. However, we also find that the trained emulators are performing particularly poorly on the predictions of large scales where all three models produce a larger velocity error than the 2nd-order FD emulator. While this is not surprising, we see in Fig. 5.8 that the error on the kinetic energy is not located on strong vorticity gradients, which differs from the behavior of the FD emulators. This is a mixed result since the emulators are both outperforming the 6th-order FD emulator on the vorticity error and underperforming the 2nd-order FD emulator on the velocity error. Again, improving on the velocity error is difficult since the emulator only has access to vorticity.

5.3.1 Fully data-driven simulations

Now that we trained SGS models (5.26) from the previously trained emulators (5.25), we first proceed as with the FD emulators and run simulations using both (trained) emulators and trained SGS models. Even if we trained the emulator and SGS model separately, we think that there is a substantial overlap between the two caused by the bias (5.20). In this case, the evolution of the system is fully data-driven and does not contain any physical components,

$$\frac{\partial \bar{\omega}}{\partial t} = \mathcal{E}(\bar{\omega}) + \mathcal{M}(\bar{\omega}). \tag{5.31}$$

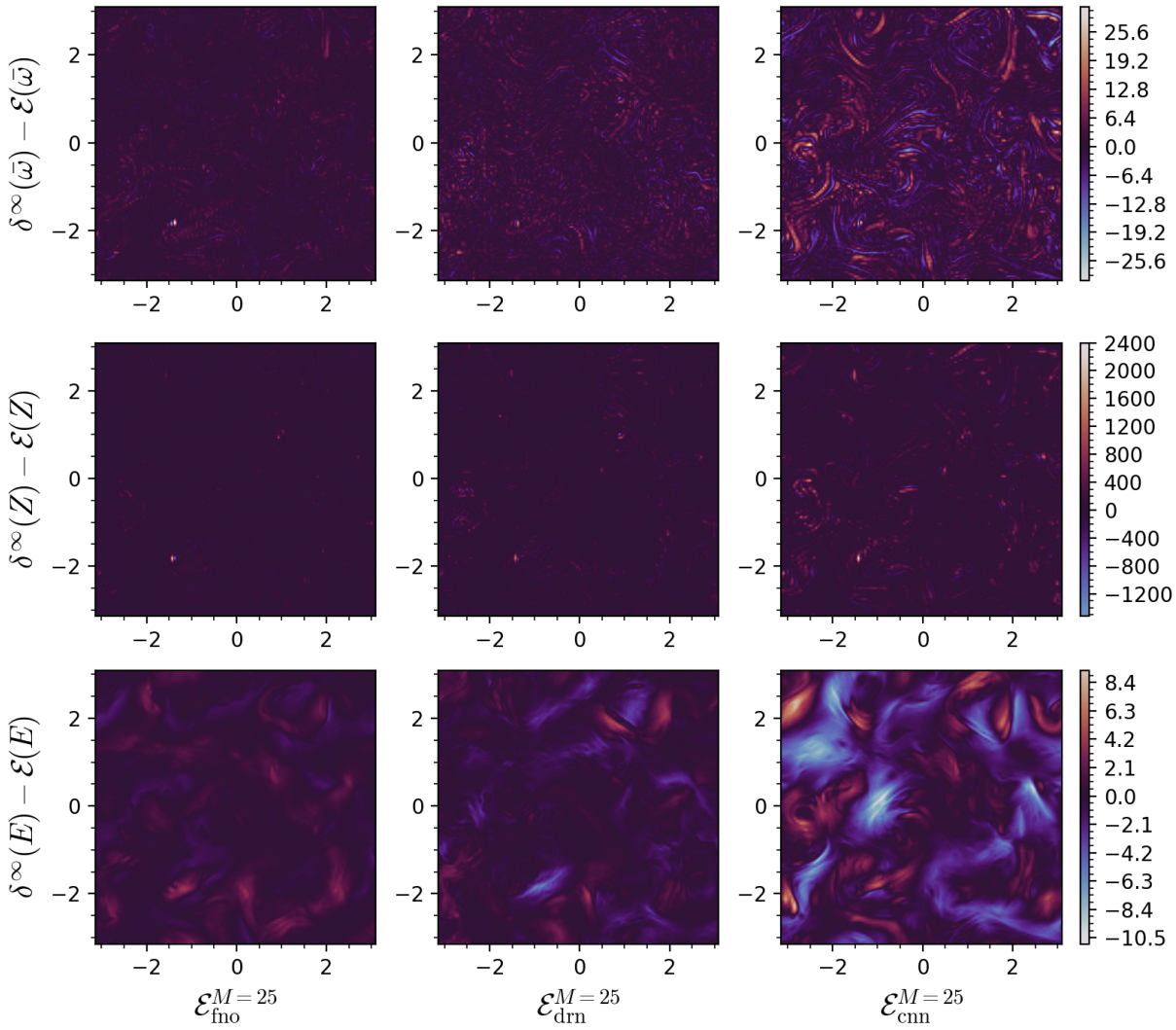

Figure 5.8: Difference between reduced pseudo-spectral reference and trained differentiable emulators for vorticity (top), enstrophy (middle), and kinetic energy (bottom) after $M = 25$ temporal steps.

Unfortunately, Fig. 5.9 shows that the produced error w.r.t the DNS is larger than the unstable 2nd-order FD emulator-SGS couple for all three trained emulator architectures. We believe that the large-scale error produced by the emulator is negatively impacting the training of the SGS model. Longer simulation runs are not shown here because they always quickly (< 500 temporal iterations) led to numerical instabilities and blow up. Increasing the complexity of the emulator in this fully data-driven scenario is also not an interesting solution, because the run-time computational cost of evaluation might outweigh the benefits gained from running simulations in reduced resolution, i.e. we want to avoid,

$$\text{cost}(\mathcal{E}(\bar{\omega})) + \text{cost}(\mathcal{M}(\bar{\omega})) > \text{cost}(f(\omega)) \tag{5.32}$$

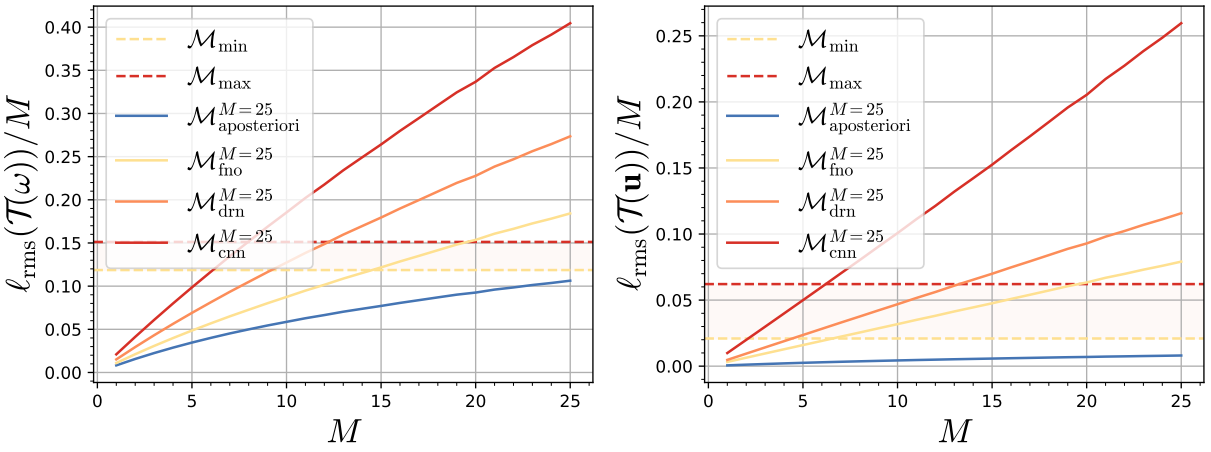

Figure 5.9: ℓ_{rms} normalized by the number of temporal integration steps M for the vorticity ω (left) and velocity $\mathbf{u}$ (right). Simulations are run with pseudo-spectral δ^∞ solver or a trained emulator and their respective SGS model so that they don't include the bias (5.20).

5.3.2 Non-differentiable simulations

One of the motivations of the differentiable emulator approach is to get back to using the non-differentiable solver once the SGS model is trained using the *a posteriori* strategy. As discussed previously, we think that trained SGS models might learn an "emulator bias" (5.20) that try to correct the error made by the emulator. However, when plugging the trained SGS model back into the pseudo-spectral reference, this error does not exist anymore, and the emulator bias now introduces a new error. Recall that when using the *a posteriori* strategy, it is convenient to define the loss function on the trajectory state, as described in (5.26),

$$\ell \equiv \ell_{\text{state}}(\{\bar{\mathbf{y}}(t)\}_{t\in[t_0,t_1]}, \{\mathcal{T}(\mathbf{y}(t))\}_{t\in[t_0,t_1]}) \tag{5.33}$$

where elements of the left-hand side of the loss are clearly dependent on the previous temporal iteration, i.e. extracting one timestep from (5.28) with $t = t_0$ and $t_1 = t_0 + 1$ we have,

$$\bar{\mathbf{y}}(t) = \bar{\mathbf{y}}(t-1) + \mathcal{E}(\bar{\mathbf{y}}(t-1)) + \mathcal{M}(\bar{\mathbf{y}}(t-1)). \tag{5.34}$$

Hence, the loss function will evaluate both the (fixed) error from the emulator $\mathcal{E}$ and the current error produced by the SGS model $\mathcal{M}$. As a conservative solution, we propose to isolate the model in the loss function, removing any occurrence of the emulator. Here, we do not increase the computational cost of the training step and only require changing the loss function during (5.26), leaving (5.25) untouched. This "mixed" loss is thus directly based on the SGS term τ,

$$\ell \equiv \ell_{\text{mixed}}(\{\mathcal{M}(\bar{\mathbf{y}})(t)\}_{t\in[t_0,t_1]}, \{\tau(t)\}_{t\in[t_0,t_1]}) \tag{5.35}$$

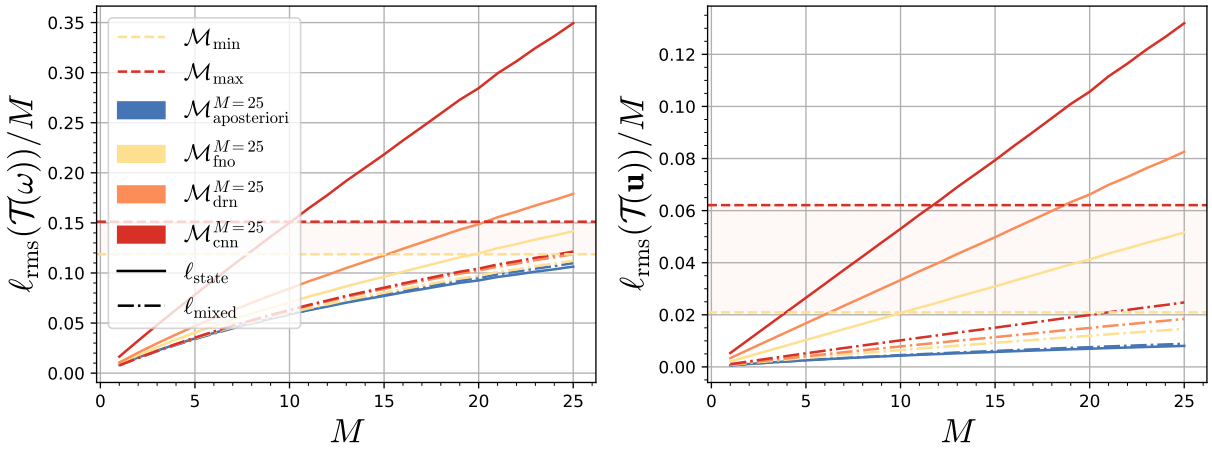

Figure 5.10: ℓ_{rms} normalized by the number of temporal integration steps M for the vorticity ω (left) and velocity $\mathbf{u}$ (right). Simulations are run with pseudo-spectral δ^{∞} solver and SGS models trained from the different emulator architectures either using the state-based (solid) or the mixed (dash-dot) loss function.

An obvious alternative would be to increase the complexity of the emulator architecture to more accurately reproduce the large-scale dynamics and reduce the emulator bias since the emulator is only temporarily used to train the SGS model. In this scenario, the cost we would like to avoid is,

$$\text{cost}(g(\bar{\omega})) + \text{cost}(\mathcal{M}(\bar{\omega})) > \text{cost}(f(\omega)) \tag{5.36}$$

with $\text{cost}(g(\bar{\omega})) \ll \text{cost}(f(\omega))$ in practice. When run with the non-differentiable pseudo-spectral solver, we can see in Fig. 5.10 that SGS models trained with ℓ_{mixed} perform much better than models trained with ℓ_{state}. It is important to note here that ℓ_{mixed} is less sensitive to the quality of the emulator, particularly on the vorticity error. Also surprisingly, ℓ_{state} still performs better when run with the pseudo-spectral solver, indicating that the emulator bias is not as large as the emulator error. However, we see in Fig. 5.11 that the models are unstable in longer simulations except for the DRN trained with ℓ_{mixed}. Still, models trained with ℓ_{state} are generally accumulating energy and enstrophy much faster than those trained with ℓ_{mixed}. Finally, Fig. 5.12 shows relatively satisfying enstrophy spectrum and fluxes, with a majority of the mismatch located in the smallest wavenumbers.

5.4 Discussion

In this chapter, we have explored emulation strategies to leverage the *a posteriori* learning strategy in non-differentiable contexts. On a relatively small temporal horizon ($M = 25$ temporal iterations), errors caused by finite-difference schemes can be small enough to lead

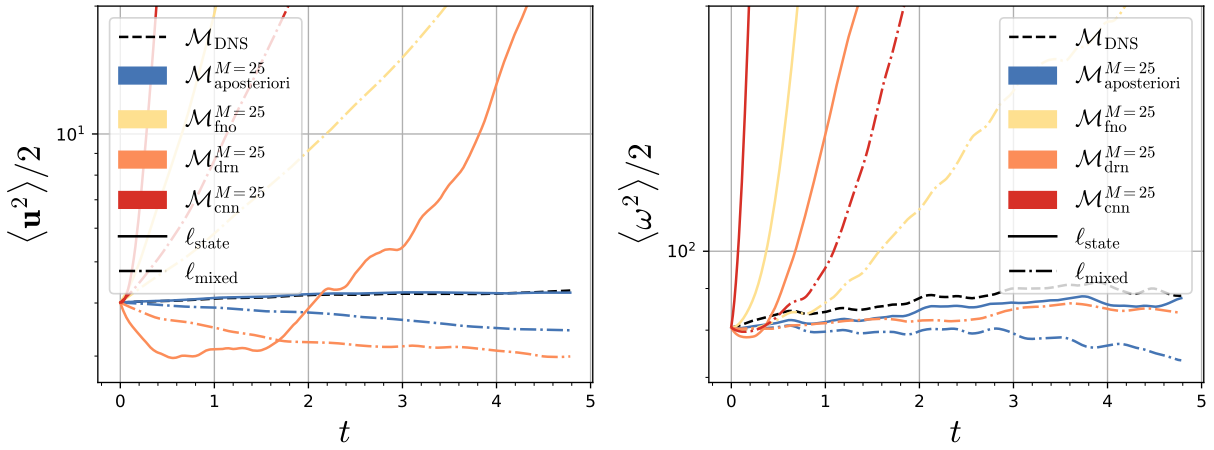

Figure 5.11: Evolution of domain-averaged energy (left) and enstrophy(right) of the SGS models trained from differentiable emulators for 3000 reduced temporal steps.

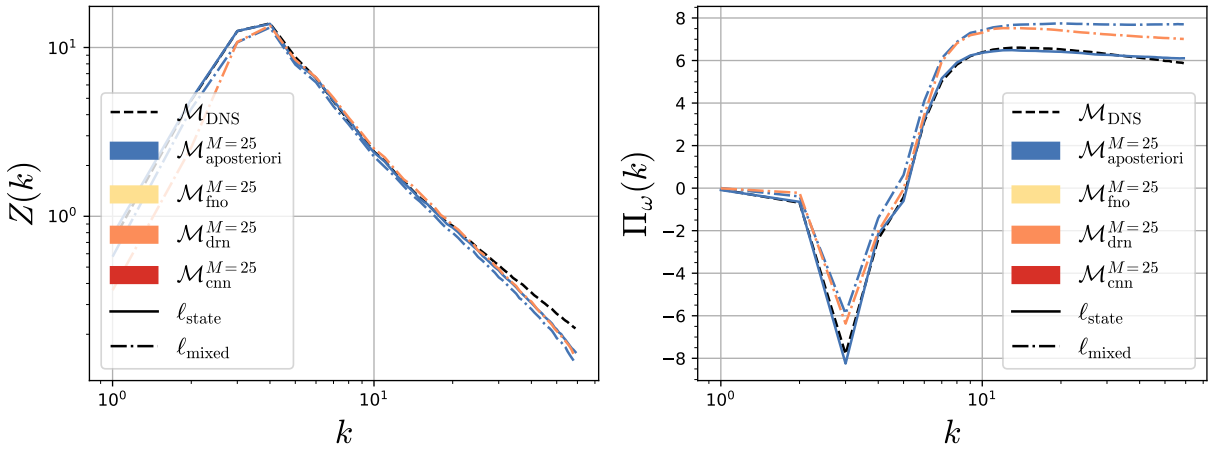

Figure 5.12: Time-averaged enstrophy spectrum (left) and enstrophy flux (right) of the SGS models trained from differentiable emulators for 3000 reduced temporal steps.

to a stable SGS model, except for low discretization order. However, we demonstrated that training a differentiable emulator leading to a stable SGS model is a much more challenging task. Indeed, FD schemes are mostly affected by small-scale error located on the largest wavenumbers while trainable emulators are more robust at small scale due to their loss function based on the vorticity. The trainable emulators are however highly affected by large-scale errors, also due to their lack of penalization on these quantities, e.g., velocities. In practice, this error is responsible for a bias that is partially corrected in the SGS model and negatively affects the large-scale behavior of the simulation. We demonstrated that changing the SGS loss to be constrained by the SGS term instead of the state of the system can lead to stable simulations and limit the impact of the emulator bias. While more expansive than the *a priori* approach, we have shown here that it is possible to train a stable SGS model in a more realistic geophysical configuration without physics-informed losses.

Improving the emulator. The first obvious improvement would be to either increase the complexity of the architecture and/or the dataset used in this context. We have demonstrated that the DRN architecture was able to improve significantly from a simple CNN architecture, which might indicate that larger gains are possible. We also note that dynamics happening at large scale has a much larger turnover time compared to the SGS dynamics and it could be interesting to train the emulator on *a posteriori* trajectories longer than $M = 25$.

Improving the SGS model. In parallel, it might be possible to compute the emulator bias during training and regularize the SGS loss function accordingly. If this error can be completely compensated, it might be possible to learn an equivalent of the *a posteriori* model trained from the pseudo-spectral reference, using relatively simple neural emulator architectures.

Chapter 6

Summary and discussion

During this thesis timespan, the field of scientific machine learning (SciML) has developed substantially (Willard et al., 2020). This progress is a collective effort gathering multiple research communities. While new ideas were starting to gain interest a few years ago, there is now a large literature in the fluid dynamics community (Brunton, 2022; Brunton et al., 2020; Vinuesa and Brunton, 2022). New projects are also transitioning from theoretical to real-world applications in earth (Gettelman et al., 2022) and climate (Rolnick et al., 2022) sciences.

In contrast with early exploratory works that consisted of direct applications of generic algorithms developed by the computer science community, ML models applied to physical problems are now using domain-specific knowledge –or physics-informed (Raissi et al., 2019)– in order to gain interpretability and generalization capabilities. Obviously, large progress has been inspired by theoretical work and collaborations at the intersection of these fields. In particular, generalized approaches to constrain the structure of a NN for a physical problem were proposed (Cranmer et al., 2020; Greydanus et al., 2019). Major improvements were also targeted to embed invariant properties, such as general invariances (Keriven and Peyré, 2019) or conservation laws (Alet et al., 2021). Applications to *subgrid* models proved successful in two-dimensional turbulence (Pawar et al., 2023), wall-bounded turbulence (Kim et al., 2022) or passive scalar advection-diffusion, to which we contributed in (Frezat et al., 2021). In practice, these models are cost-free in the sense that there are no negative impacts in specifying invariances in a NN. It simultaneously improves training time, as the training space is reduced to only cover regions that fulfill the invariances, it helps generalization since turbulent systems are expected to be governed by the same set of invariances and most of the time improve the overall performance. Transfer to larger systems has also started (Beucler et al., 2021b) but remains quite difficult due to complex interactions, e.g. between fluid motions and thermodynamics that may not lead

to a possible invariant description. In this context, having *subgrid* models dedicated to specific processes with a clear definition and invariant properties might be more beneficial than the "global" approach (Gentine et al., 2021) where a ML model is trained to represent many *subgrid* processes.

Following the computer science and robotics communities, physicists started applying end-to-end learning to systems of PDEs (Holl et al., 2020; Um et al., 2020) and variational systems (Fablet et al., 2021b). This technique has gained a lot of interest recently in different areas of applications and is now referred to as *differentiable physics* (Qiao et al., 2020; Schoenholz and Cubuk, 2020). Framing the learning problem in this framework has been shown to improve performance for two-dimensional turbulent flows when applied as a correction (Kochkov et al., 2021). In this context, we have demonstrated numerous improvements in quasi-geostrophic *subgrid* modeling, including long-term stability and universality to grid resolution within turbulence hypotheses (Frezat et al., 2022). A particularly important advantage of this approach is that the learning problem can now be described from quantities of interest, instead of unresolved terms. It is interesting to note the connection between end-to-end learning and data assimilation schemes (Pawar and San, 2021). A unified approach to realistic weather and climate models may thus be possible and beneficial (Brajard et al., 2021). However, the application of such a learning strategy to large-scale realistic solvers still requires solutions to the following challenges. First, the cost of end-to-end learning is much larger than traditional "offline" approaches, in particular when the integrated system is computationally expansive. Then, the solver is required to be differentiable, which can be prohibitive for legacy numerical models containing millions of lines of code in a non-differentiable language. Many paths could then be explored, such as reinforcement learning (Novati et al., 2021) which alleviate differentiability requirements, or deep differentiable emulators (Nonnenmacher and Greenberg, 2021) that approximate some system with a differentiable parametric function.

Training a differentiable emulator to temporarily reproduce the large-scale dynamics of a turbulent system seems promising in order to enable end-to-end learning of a *subgrid* model from non-differentiable solvers. In practice, however, the application to non-linear PDEs is difficult and learning an accurate representation of the entire numerical system composed of spatial and temporal discretization schemes is still an open problem. It might be possible to define a very complex architecture that almost exactly reproduces the numerical solver dynamics, at a considerable cost. If this cost is prohibitive, we can just assume that the error in the emulator will then be propagated to the *subgrid* model, as we have shown in the last chapter of this thesis. This corrective bias, due to the non-linear term of the differential equation is often dominant at the largest scales for a turbulent flow. Unfortunately, these scales are not supposed to have a significant impact in the *subgrid* model, and will often lead to a simulation blowup due to energy accumulation. Isolating

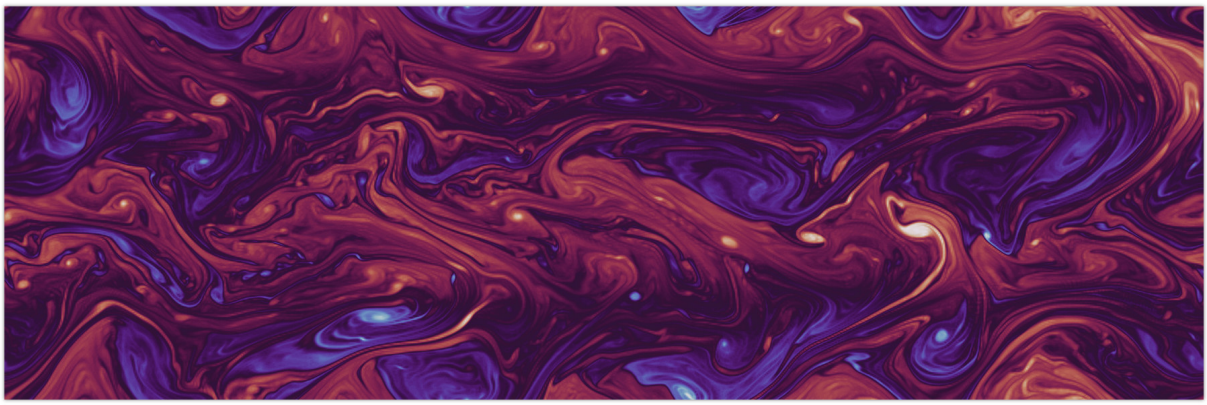

Figure 6.1: Vorticity in a turbulent quasi-geostrophic system with topography in a non-rectangular periodic domain. The simulation was produced by a non-differentiable solver and uses a subgrid model trained end-to-end by a NN-based emulator in a square domain.

the *subgrid* model in the loss function is a first-step solution to this issue and enabled the first stable emulation-based simulations of complex quasi-geostrophic flows (see Fig. 6.1). Reaching the same level of accuracy on the *subgrid* model without a differentiable solver would open new possibilities with realistic, large-scale numerical models. While some have proposed NN-based discretization schemes (Ranade et al., 2021; Zhuang et al., 2021), it might be possible to emulate numerical schemes to learn *subgrid* terms that both account for unresolved scales and spatial discretization errors, as we explored using a modified pseudo-spectral scheme.

The field of physics-based machine learning, together with applications to *subgrid* turbulence modeling has evolved substantially in a short period. Three years ago, the field was at a state where direct applications to idealized configurations using standard, i.e. not physics-specific ML models were just emerging. Today, an entire dedicated field at the intersection of physics and machine learning has emerged. It has already been provided with many theoretical applications that remain to be applied to realistic scenarios, which will certainly happen in the near future (Bauer et al., 2021; Irrgang et al., 2021; Sonnewald et al., 2021). We can be confident in the idea that future ocean, atmosphere, and climate models will use ML components. However, some challenges remain about the coupling –or hybridization– of numerical modeling and artificial intelligence within these systems (Reichstein et al., 2019). Whether modern workflows will use tools allowing a smooth and efficient coupling (Mozaffari et al., 2022; Partee et al., 2022) or reimplement existing numerical solvers in differentiable languages (Häfner et al., 2018; Sridhar et al., 2022) is still an open question.

Appendix A

Spectral decorrelation over learning horizons for subgrid models

When learning a dynamical component, performance drastically increases when samples are representing a diversified distribution of the target space. If data is extracted from numerical simulations, it is possible to either start from many independent initial conditions. However, startup time can be prohibitively expansive, in particular when the system takes time to reach an equilibrium state and when the transitional regime should not be seen in the training data. Using a single simulation, one must separate each sample by a certain number of temporal iterations in order to avoid inter-sample correlation. In

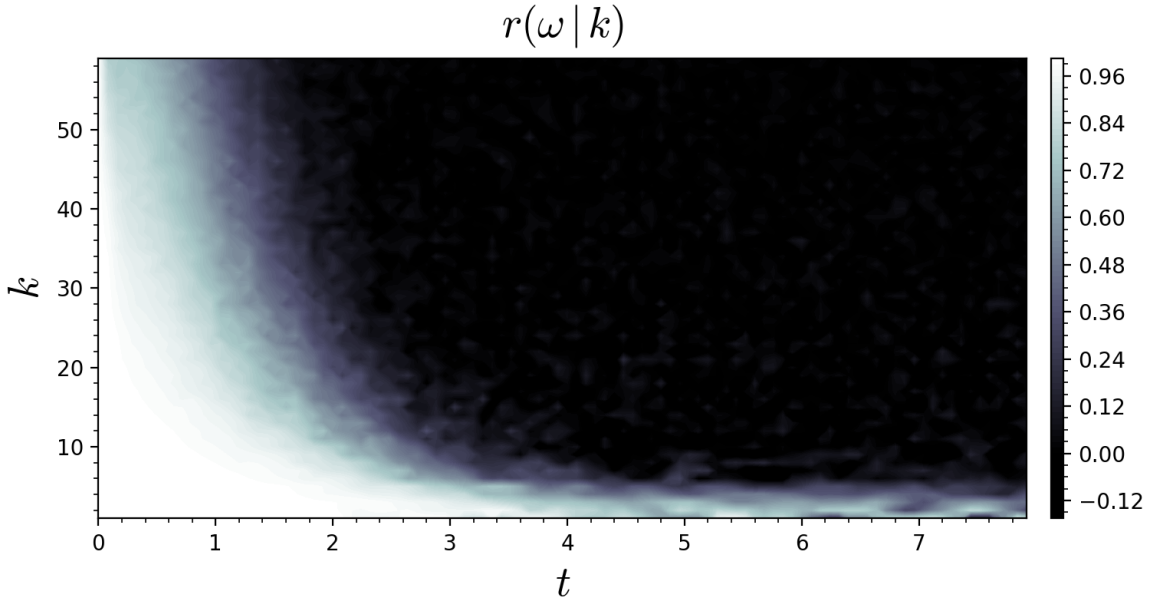

Figure A.1: Spectral decorrelation coefficients over time at some wavenumber k for vorticity fields in a quasi-geostrophic system with topography as described in (5.15). In the dynamical system, the propagation to the smallest wavenumbers follows an inverse exponential e^{-t}.

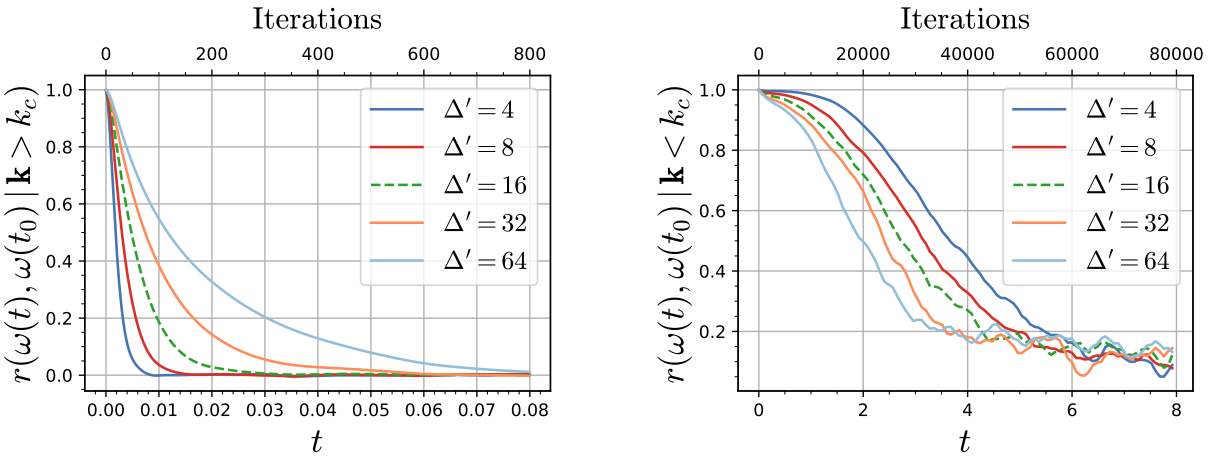

Figure A.2: Correlation coefficients between vorticity fields at a different time for the largest (left) and smallest (right) scales corresponding to modeled and resolved dynamics in a SGS formulation, respectively.

the SGS learning problem, it is possible to split the dynamics into resolved and modeled dynamics containing the largest and smallest scales of the system, respectively. The separation between resolved and modeled is based on the spectral representation of the system, i.e. at some scale Δ' we have,

$$k_c = \frac{k_{\max}}{\Delta'} \tag{A.1}$$

where $k_{\max}$ is the largest wavenumber represented in some domain Ω. Now, the resolved and modeled dynamics are contained in $\mathbf{k} < k_c$ and $\mathbf{k} > k_c$, respectively. We show in Fig. A.1 the correlation coefficients between the initial and temporally advanced fields in a specific dynamical system. In this system, the resolved dynamics is decorrelating rapidly compared to the modeled dynamics (see A.2). This is expected since large coherent structures tend to live longer than small vortices. This type of correlation plot helps in choosing the sampling strategy for *a priori* learning (4.20) and the maximum learning temporal horizon for the *a posteriori* learning strategy (4.25). Depending on the scale (or wavenumber span) of the learned dynamics, it is thus important to adapt either the sampling strategy or the temporal integration horizon. In the SGS-emulation context, we would ideally have (without considering technical limitations),

$$M(\mathbf{k}) = \frac{t}{\Delta t}, \quad \text{s.t.} \quad r(\omega(t), \omega(t_0) \,|\, \mathbf{k}) \approx 0, \tag{A.2}$$

with $M_{\text{emu}} \gg M_{\text{sgs}}$.

Bibliography

Alet, F., Doblar, D., Zhou, A., Tenenbaum, J., Kawaguchi, K., and Finn, C. (2021). Noether networks: meta-learning useful conserved quantities. *Advances in Neural Information Processing Systems*, 34.

Alvelius, K. (1999). Random forcing of three-dimensional homogeneous turbulence. *Physics of Fluids*, 11(7):1880–1889.

Aycock, J. (2003). A brief history of just-in-time. *ACM Computing Surveys (CSUR)*, 35(2):97–113.

Bakarji, J. and Tartakovsky, D. M. (2021). Data-driven discovery of coarse-grained equations. *Journal of Computational Physics*, 434:110219.

Balarac, G., Le Sommer, J., Meunier, X., and Vollant, A. (2013). A dynamic regularized gradient model of the subgrid-scale scalar flux for large eddy simulations. *Physics of Fluids*, 25(7):075107.

Bardina, J. (1983). *Improved turbulence models based on large eddy simulation of homogeneous, incompressible, turbulent flows*. Stanford University.

Bardina, J., Ferziger, J., and Reynolds, W. (1980). Improved subgrid-scale models for large-eddy simulation. In *13th fluid and plasmadynamics conference*, page 1357.

Batchelor, G. K. (1959). Small-scale variation of convected quantities like temperature in turbulent fluid part 1. general discussion and the case of small conductivity. *Journal of Fluid Mechanics*, 5(1):113–133.

Batchelor, G. K. (1969). Computation of the energy spectrum in homogeneous two-dimensional turbulence. *The Physics of Fluids*, 12(12):II–233.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Bauer, P., Dueben, P. D., Hoefler, T., Quintino, T., Schulthess, T. C., and Wedi, N. P. (2021). The digital revolution of earth-system science. *Nature Computational Science*, 1(2):104–113.

Bauer, P., Thorpe, A., and Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43.

Beck, A., Flad, D., and Munz, C.-D. (2019). Deep neural networks for data-driven les closure models. *Journal of Computational Physics*, 398:108910.

Beck, A. and Kurz, M. (2021). A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, 44(1):e202100002.

Belbute-Peres, F. D. A., Economon, T., and Kolter, Z. (2020). Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, pages 2402–2411.

Berselli, L., Iliescu, T., and Layton, W. (2006). *Mathematics of large eddy simulation of turbulent flows*. Springer.

Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P., and Gentine, P. (2021a). Enforcing analytic constraints in neural networks emulating physical systems. *Physical Review Letters*, 126(9):098302.

Beucler, T., Pritchard, M., Yuval, J., Gupta, A., Peng, L., Rasp, S., Ahmed, F., O'Gorman, P. A., Neelin, J. D., Lutsko, N. J., et al. (2021b). Climate-invariant machine learning. *arXiv preprint arXiv:2112.08440*.

Bischof, C. H., Roh, L., and Mauer-Oats, A. J. (1997). Adic: an extensible automatic differentiation tool for ansi-c. *Software: Practice and Experience*, 27(12):1427–1456.

Bode, M., Gauding, M., Lian, Z., Denker, D., Davidovic, M., Kleinheinz, K., Jitsev, J., and Pitsch, H. (2021). Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows. *Proceedings of the Combustion Institute*, 38(2):2617–2625.

Boffetta, G. and Ecke, R. E. (2012). Two-dimensional turbulence. *Annual review of fluid mechanics*, 44(1):427–451.

Bolton, T. and Zanna, L. (2019). Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems*, 11(1):376–399.

Bonavita, M. and Laloyaux, P. (2020). Machine learning for model error inference and correction. *Journal of Advances in Modeling Earth Systems*, 12(12):e2020MS002232.

Bouchet, F. and Venaille, A. (2012). Statistical mechanics of two-dimensional and geophysical flows. *Physics reports*, 515(5):227–295.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Brajard, J., Carrassi, A., Bocquet, M., and Bertino, L. (2021). Combining data assimilation and machine learning to infer unresolved scale parametrization. *Philosophical Transactions of the Royal Society A*, 379(2194):20200086.

Brunton, S. L. (2022). Applying machine learning to study fluid mechanics. *Acta Mechanica Sinica*, pages 1–9.

Brunton, S. L., Noack, B. R., and Koumoutsakos, P. (2020). Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52:477–508.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937.

Canuto, C., Hussaini, M. Y., Quarteroni, A., Thomas Jr, A., et al. (2012). *Spectral methods in fluid dynamics*. Springer Science & Business Media.

Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. (2007). *Spectral methods: fundamentals in single domains*. Springer Science & Business Media.

Carati, D., Ghosal, S., and Moin, P. (1995). On the representation of backscatter in dynamic localization models. *Physics of Fluids*, 7(3):606–616.

Carati, D., Winckelmans, G., and Jeanmart, H. (1999). Exact expansions for filtered-scales modelling with a wide class of les filters. In *Direct and large-eddy simulation III*, pages 213–224. Springer.

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., and Zdeborová, L. (2019). Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002.

Carrassi, A., Bocquet, M., Bertino, L., and Evensen, G. (2018). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535.

Charalampopoulos, A.-T. G. and Sapsis, T. P. (2022). Machine-learning energy-preserving nonlocal closures for turbulent fluid flows and inertial tracers. *Physical Review Fluids*, 7(2):024305.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Chollet, J.-P. and Lesieur, M. (1981). Parameterization of small scales of three-dimensional isotropic turbulence utilizing spectral closures. *Journal of Atmospheric Sciences*, 38(12):2747–2757.

Clark, R. A., Ferziger, J. H., and Reynolds, W. C. (1979). Evaluation of subgrid-scale models using an accurately simulated turbulent flow. *Journal of fluid mechanics*, 91(1):1–16.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*, pages 2990–2999.

Corrsin, S. (1951). On the spectrum of isotropic temperature fluctuations in an isotropic turbulence. *Journal of Applied Physics*, 22(4):469–473.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S.

(2020). Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*.

da Silva, C. B. and Pereira, J. C. (2007). Analysis of the gradient-diffusion hypothesis in large-eddy simulations based on transport equations. *Physics of Fluids*, 19(3):035106.

D'andrea, F. and Vautard, R. (2000). Reducing systematic errors by empirically correcting model errors. *Tellus A*, 52(1):21–41.

Danilov, S., Juricke, S., Kutsenko, A., and Oliver, M. (2019). Toward consistent subgrid momentum closures in ocean models. In *Energy transfers in atmosphere and ocean*, pages 145–192. Springer.

de Avila Belbute-Peres, F., Smith, K., Allen, K., Tenenbaum, J., and Kolter, J. Z. (2018). End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31.

Deardorff, J. W. (1970). A numerical study of three-dimensional turbulent channel flow at large reynolds numbers. *Journal of Fluid Mechanics*, 41(2):453–480.

Dee, D. P. (2005). Bias and data assimilation. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(613):3323–3343.

Dueben, P. D. and Bauer, P. (2018). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11(10):3999–4009.

Duraisamy, K., Iaccarino, G., and Xiao, H. (2019). Turbulence modeling in the age of data. *Annu. Rev. Fluid Mech*, 51:357–77.

Endres, D. M. and Schindelin, J. E. (2003). A new metric for probability distributions. *IEEE Transactions on Information theory*, 49(7):1858–1860.

Eswaran, V. and Pope, S. B. (1988). An examination of forcing in direct numerical simulations of turbulence. *Computers & Fluids*, 16(3):257–278.

Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., and Taylor, K. E. (2016). Overview of the coupled model intercomparison project phase 6 (cmip6) experimental design and organization. *Geoscientific Model Development*, 9(5):1937–1958.

Fablet, R., Chapron, B., Drumetz, L., Mémin, E., Pannekoucke, O., and Rousseau, F. (2021a). Learning variational data assimilation models and solvers. *Journal of Advances in Modeling Earth Systems*, 13(10):e2021MS002572.

Fablet, R., Drumetz, L., and Rousseau, F. (2021b). End-to-end learning of variational models and solvers for the resolution of interpolation problems. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Sig-*

nal Processing (ICASSP), pages 2360–2364. IEEE.

Fabre, Y. and Balarac, G. (2011). Development of a new dynamic procedure for the clark model of the subgrid-scale scalar flux using the concept of optimal estimator. *Physics of Fluids*, 23(11):115103.

Farchi, A., Laloyaux, P., Bonavita, M., and Bocquet, M. (2021). Using machine learning to correct model error in data assimilation and forecast applications. *Quarterly Journal of the Royal Meteorological Society*, 147(739):3067–3084.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. (2020). How to train your neural ode: the world of jacobian and kinetic regularization. In *International conference on machine learning*, pages 3154–3164. PMLR.

Fornberg, B. (1990). High-order finite differences and the pseudospectral method on staggered grids. *SIAM Journal on Numerical Analysis*, 27(4):904–918.

Fox-Kemper, B. and Menemenlis, D. (2008). Can large eddy simulation techniques improve mesoscale rich ocean models? *Washington DC American Geophysical Union Geophysical Monograph Series*, 177:319–337.

Frederiksen, J. S., O'Kane, T. J., and Zidikheri, M. J. (2012). Stochastic subgrid parameterizations for atmospheric and oceanic flows. *Physica Scripta*, 85(6):068202.

Frezat, H., Balarac, G., Le Sommer, J., Fablet, R., and Lguensat, R. (2021). Physical invariance in neural networks for subgrid-scale scalar flux modeling. *Physical Review Fluids*, 6(2):024607.

Frezat, H., Le Sommer, J., Fablet, R., Balarac, G., and Lguensat, R. (2022). A posteriori learning for quasi-geostrophic turbulence parametrization. *Journal of Advances in Modeling Earth Systems*, 14(11):e2022MS003124.

Frisch, U. and Kolmogorov, A. N. (1995). *Turbulence: The legacy of AN Kolmogorov*. Cambridge University Press.

Gamahara, M. and Hattori, Y. (2017). Searching for turbulence models by artificial neural network. *Physical Review Fluids*, 2(5):054604.

Ge, H., Xu, K., and Ghahramani, Z. (2018). Turing: a language for flexible probabilistic inference. In *International conference on artificial intelligence and statistics*, pages 1682–1690. PMLR.

Gentine, P., Eyring, V., and Beucler, T. (2021). Deep learning for the parametrization of subgrid processes in climate models. *Deep Learning for the Earth Sciences: A Comprehensive Approach to Remote Sensing, Climate Science, and Geosciences*, pages 307–314.

Germano, M., Piomelli, U., Moin, P., and Cabot, W. H. (1991). A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765.

Gettelman, A., Geer, A. J., Forbes, R. M., Carmichael, G. R., Feingold, G., Posselt, D. J., Stephens, G. L., van den Heever, S. C., Varble, A. C., and Zuidema, P. (2022). The future of earth system prediction: Advances in model-data fusion. *Science Advances*, 8(14):eabn3488.

Giering, R. and Kaminski, T. (1998). Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS)*, 24(4):437–474.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Graham, J. P. and Ringler, T. (2013). A framework for the evaluation of turbulence closures used in mesoscale ocean large-eddy simulations. *Ocean Modelling*, 65:25–39.

Greydanus, S., Dzamba, M., and Yosinski, J. (2019). Hamiltonian neural networks. *Advances in neural information processing systems*, 32.

Griffith, A. K. and Nichols, N. K. (2000). Adjoint methods in data assimilation for estimating model error. *Flow, turbulence and combustion*, 65(3):469–488.

Guan, Y., Chattopadhyay, A., Subel, A., and Hassanzadeh, P. (2022a). 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.

Guan, Y., Subel, A., Chattopadhyay, A., and Hassanzadeh, P. (2022b). Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate les. *Physica D: Nonlinear Phenomena*, page 133568.

Häfner, D., Jacobsen, R. L., Eden, C., Kristensen, M. R., Jochum, M., Nuterman, R., and Vinter, B. (2018). Veros v0. 1–a fast and versatile ocean simulator in pure python. *Geoscientific Model Development*, 11(8):3299–3312.

Hascoet, L. and Pascual, V. (2013). The tapenade automatic differentiation tool: principles, model, and specification. *ACM Transactions on Mathematical Software (TOMS)*, 39(3):1–43.

Hatfield, S., Chantry, M., Dueben, P., Lopez, P., Geer, A., and Palmer, T. (2021). Building tangent-linear and adjoint models for data assimilation with neural networks. *Journal of Advances in Modeling Earth Systems*, 13(9):e2021MS002521.

Heimbach, P., Hill, C., and Giering, R. (2002). Automatic generation of efficient adjoint code for a parallel navier-stokes solver. In *International Conference on Computational Science*, pages 1019–1028. Springer.

Heimbach, P., Hill, C., and Giering, R. (2005). An efficient exact adjoint of the parallel mit general circulation model, generated via automatic differentiation. *Future Generation Computer Systems*, 21(8):1356–1371.

Holl, P., Koltun, V., and Thuerey, N. (2020). Learning to control pdes with

differentiable physics. *arXiv preprint arXiv:2001.07457*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Houghton, J. T., Ding, Y., Griggs, D. J., Noguer, M., van der Linden, P. J., Dai, X., Maskell, K., and Johnson, C. (2001). *Climate change 2001: the scientific basis: contribution of Working Group I to the third assessment report of the Intergovernmental Panel on Climate Change*. Cambridge university press.

Huang, L. and Topping, D. (2021). Jlbox v1. 1: a julia-based multi-phase atmospheric chemistry box model. *Geoscientific Model Development*, 14(4):2187–2203.

Ince, T., Kiranyaz, S., Eren, L., Askar, M., and Gabbouj, M. (2016). Real-time motor fault detection by 1-d convolutional neural networks. *IEEE Transactions on Industrial Electronics*, 63(11):7067–7075.

Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., and Tebbutt, W. (2019). A differentiable programming system to bridge machine learning and scientific computing. *arXiv preprint arXiv:1907.07587*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 448–456.

Irrgang, C., Boers, N., Sonnewald, M., Barnes, E. A., Kadow, C., Staneva, J., and Saynisch-Wagner, J. (2021). Towards neural earth system modelling by integrating artificial intelligence in earth system science. *Nature Machine Intelligence*, 3(8):667–674.

Jain, P., Kar, P., et al. (2017). Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363.

Jansen, M. F., Held, I. M., Adcroft, A., and Hallberg, R. (2015). Energy budget-based backscatter in an eddy permitting primitive equation model. *Ocean Modelling*, 94:15–26.

Juricke, S., Danilov, S., Koldunov, N., Oliver, M., and Sidorenko, D. (2020). Ocean kinetic energy backscatter parametrization on unstructured grids: Impact on global eddy-permitting simulations. *Journal of Advances in Modeling Earth Systems*, 12(1):e2019MS001855.

Juricke, S., Danilov, S., Kutsenko, A., and Oliver, M. (2019). Ocean kinetic energy backscatter parametrizations on unstructured grids: Impact on mesoscale turbulence in a channel. *Ocean Modelling*, 138:51–67.

Kamnitsas, K., Ledig, C., Newcombe, V. F., Simpson, J. P., Kane, A. D., Menon, D. K., Rueckert, D., and Glocker, B. (2017). Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78.

Kasim, M., Watson-Parris, D., Deaconu, L., Oliver, S., Hatfield, P., Froula, D., Gregori, G., Jarvis, M., Khatiwala, S., Korenaga, J., et al. (2021). Building high accuracy emulators for scientific simulations with deep neural architecture search. *Machine Learning: Science and Technology*, 3(1):015013.

Kassam, A.-K. and Trefethen, L. N. (2005). Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233.

Keriven, N. and Peyré, G. (2019). Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32.

Kharin, V. V. and Zwiers, F. W. (2000). Changes in the extremes in an ensemble of transient climate simulations with a coupled atmosphere–ocean gcm. *Journal of climate*, 13(21):3760–3788.

Kim, J., Kim, H., Kim, J., and Lee, C. (2022). Deep reinforcement learning for large-eddy simulation modeling in wall-bounded turbulence. *Physics of Fluids*, 34(10):105132.

Kim, J. and Lee, C. (2020). Prediction of turbulent heat transfer using convolutional neural networks. *Journal of Fluid Mechanics*, 882.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR 2015)*.

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118.

Kolmogorov, A. N. (1941). The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers. *Cr Acad. Sci. URSS*, 30:301–305.

Kolmogorov, A. N. (1962). A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high reynolds number. *Journal of Fluid Mechanics*, 13(1):82–85.

Kravchenko, A. and Moin, P. (1997). On the effect of numerical errors in large eddy simulations of turbulent flows. *Journal of computational physics*, 131(2):310–322.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NeurIPS 2012)*, pages 1097–1105.

Laloyaux, P., Balmaseda, M., Dee, D., Mogensen, K., and Janssen, P. (2016). A coupled data assimilation system for climate reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 142(694):65–78.

Lapeyre, C. J., Misdariis, A., Cazard, N., Veynante, D., and Poinsot, T. (2019). Training convolutional neural networks to estimate turbulent sub-grid scale reaction

rates. *Combustion and Flame*, 203:255–264.

Leith, C. (1996). Stochastic models of chaotic systems. *Physica D: Nonlinear Phenomena*, 98(2-4):481–491.

Leonard, A. (1975). Energy cascade in large-eddy simulations of turbulent fluid flows. In *Advances in geophysics*, volume 18, pages 237–248. Elsevier.

Lesieur, M. and Metais, O. (1996). New trends in large-eddy simulations of turbulence. *Annual review of fluid mechanics*, 28(1):45–82.

Lesieur, M., Métais, O., Comte, P., et al. (2005). *Large-eddy simulations of turbulence*. Cambridge University Press.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Lilly, D. K. (1992). A proposed modification of the germano subgrid-scale closure method. *Physics of Fluids A: Fluid Dynamics*, 4(3):633–635.

Ling, J., Jones, R., and Templeton, J. (2016a). Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35.

Ling, J., Kurzawski, A., and Templeton, J. (2016b). Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166.

Liu, Y., Lu, L., Fang, L., and Gao, F. (2011). Modification of spalart–allmaras model with consideration of turbulence energy backscatter using velocity helicity. *Physics Letters A*, 375(24):2377–2381.

Long, Z., Lu, Y., Ma, X., and Dong, B. (2018). Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216. PMLR.

Loose, N. and Heimbach, P. (2021). Leveraging uncertainty quantification to design ocean climate observing systems. *Journal of Advances in Modeling Earth Systems*, 13(4):e2020MS002386.

Lund, T. S. and Novikov, E. (1993). Parameterization of subgrid-scale stress by the velocity gradient tensor. *Annual Research Briefs, 1992*.

Lyu, G., Köhl, A., Matei, I., and Stammer, D. (2018). Adjoint-based climate model tuning: Application to the planet simulator. *Journal of Advances in Modeling Earth Systems*, 10(1):207–222.

MacArt, J. F., Sirignano, J., and Freund, J. B. (2021). Embedded training of neural-network subgrid-scale turbulence models. *Physical Review Fluids*, 6(5):050502.

Majda, A. and Wang, X. (2006). *Nonlinear dynamics and statistical theories for basic geophysical flows*. Cambridge University Press.

Margossian, C. C. (2019). A review of automatic differentiation and its efficient implementation. *Wiley interdisciplinary re-*

views: data mining and knowledge discovery, 9(4):e1305.

Markham, I. S. and Rakes, T. R. (1998). The effect of sample size and variability of data on the comparative performance of artificial neural networks and regression. *Computers & operations research*, 25(4):251–263.

Mason, P. J. (1989). Large-eddy simulation of the convective atmospheric boundary layer. *Journal of Atmospheric Sciences*, 46(11):1492–1516.

Maulik, R., San, O., Rasheed, A., and Vedula, P. (2019). Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144.

Mellor, G. L. (1985). Ensemble average, turbulence closure. In *Advances in Geophysics*, volume 28, pages 345–358. Elsevier.

Meneveau, C. (1994). Statistics of turbulence subgrid-scale stresses: Necessary conditions and experimental tests. *Physics of Fluids*, 6(2):815–833.

Meneveau, C. and Katz, J. (2000). Scale-invariance and turbulence models for large-eddy simulation. *Annual Review of Fluid Mechanics*, 32(1):1–32.

Meneveau, C., Lund, T. S., and Cabot, W. H. (1996). A lagrangian dynamic subgrid-scale model of turbulence. *Journal of fluid mechanics*, 319:353–385.

Métais, O. and Lesieur, M. (1992). Spectral large-eddy simulation of isotropic and stably stratified turbulence. *Journal of Fluid Mechanics*, 239:157–194.

Mohan, A. T., Lubbers, N., Livescu, D., and Chertkov, M. (2020). Embedding hard physical constraints in convolutional neural networks for 3d turbulence. Workshop on Integration of Deep Neural Models and Differential Equations (ICLR 2020).

Moin, P., Squires, K., Cabot, W., and Lee, S. (1991). A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Physics of Fluids A: Fluid Dynamics*, 3(11):2746–2757.

Moore, A. M., Arango, H. G., Di Lorenzo, E., Cornuelle, B. D., Miller, A. J., and Neilson, D. J. (2004). A comprehensive ocean prediction and analysis system based on the tangent linear and adjoint of a regional ocean model. *Ocean Modelling*, 7(1-2):227–258.

Mozaffari, A., Langguth, M., Gong, B., Ahring, J., Campos, A. R., Nieters, P., Escobar, O. J. C., Wittenbrink, M., Baumann, P., and Schultz, M. G. (2022). Hpc-oriented canonical workflows for machine learning applications in climate and weather prediction. *Data Intelligence*, 4(2):271–285.

Nonnenmacher, M. and Greenberg, D. S. (2021). Deep emulators for differentiation, forecasting, and parametrization in earth science simulators. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002554.

Novati, G., de Laroussilhe, H. L., and Koumoutsakos, P. (2021). Automating

turbulence modelling by multi-agent reinforcement learning. *Nature Machine Intelligence*, 3(1):87–96.

Oberlack, M. (1997). Invariant modeling in large-eddy simulation of turbulence. *Annual Research Briefs*, pages 3–22.

Obukhov, A. M. (1970). Structure of temperature field in turbulent flow. *Izvestiia Akademii Nauk SSSR, Ser. Geogr. i Geofiz.*, 13(1):58–69.

Ollinaho, P., Bechtold, P., Leutbecher, M., Laine, M., Solonen, A., Haario, H., and Järvinen, H. (2013). Parameter variations in prediction skill optimization at ecmwf. *Nonlinear processes in Geophysics*, 20(6):1001–1010.

Orszag, S. A. and Patterson Jr, G. (1972). Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Physical review letters*, 28(2):76.

Ouala, S., Debreu, L., Pascual, A., Chapron, B., Collard, F., Gaultier, L., and Fablet, R. (2021). Learning runge-kutta integration schemes for ode simulation and identification. *arXiv preprint arXiv:2105.04999*.

Partee, S., Ellis, M., Rigazzi, A., Shao, A. E., Bachman, S., Marques, G., and Robbins, B. (2022). Using machine learning at scale in numerical simulations with smartsim: An application to ocean climate modeling. *Journal of Computational Science*, page 101707.

Pawar, S. and San, O. (2021). Data assimilation empowered neural network parametrizations for subgrid processes in geophysical flows. *Physical Review Fluids*, 6(5):050501.

Pawar, S., San, O., Rasheed, A., and Vedula, P. (2020). A priori analysis on deep learning of subgrid-scale parameterizations for kraichnan turbulence. *Theoretical and Computational Fluid Dynamics*, 34(4):429–455.

Pawar, S., San, O., Rasheed, A., and Vedula, P. (2023). Frame invariant neural network closures for kraichnan turbulence. *Physica A: Statistical Mechanics and its Applications*, 609:128327.

Piomelli, U. (2014). Large eddy simulations in 2030 and beyond. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2022):20130320.

Piomelli, U., Cabot, W. H., Moin, P., and Lee, S. (1991). Subgrid-scale backscatter in turbulent and transitional flows. *Physics of Fluids A: Fluid Dynamics*, 3(7):1766–1771.

Pope, S. B. (2000). *Turbulent Flows*. Cambridge University Press.

Portwood, G. D., Nadiga, B. T., Saenz, J. A., and Livescu, D. (2021). Interpreting neural network models of residual scalar flux. *Journal of Fluid Mechanics*, 907:A23.

Qiao, Y.-L., Liang, J., Koltun, V., and Lin, M. C. (2020). Scalable differentiable physics for learning and control. *arXiv preprint arXiv:2007.02168*.

Raäisaänen, J. (2007). How reliable are climate models? *Tellus A: Dynamic Meteorology and Oceanography*, 59(1):2–29.

Rabier, F., Thépaut, J.-N., and Courtier, P. (1998). Extended assimilation and forecast experiments with a four-dimensional variational assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 124(550):1861–1887.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., and Edelman, A. (2020). Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). 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.

Ramadhan, A., Wagner, G., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza, A., Edelman, A., Ferrari, R., and Marshall, J. (2020). Oceananigans. jl: Fast and friendly geophysical fluid dynamics on gpus. *Journal of Open Source Software*, 5(53).

Ranade, R., Hill, C., and Pathak, J. (2021). Discretizationnet: A machine-learning based solver for navier–stokes equations using finite volume discretization. *Computer Methods in Applied Mechanics and Engineering*, 378:113722.

Rasp, S., Pritchard, M. S., and Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39):9684–9689.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., et al. (2019). Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204.

Richardson, L. F. (1922). *Weather prediction by numerical process*. University Press.

Rogallo, R. S. (1981). *Numerical experiments in homogeneous turbulence*, volume 81315. National Aeronautics and Space Administration.

Rolnick, D., Donti, P. L., Kaack, L. H., Kochanski, K., Lacoste, A., Sankaran, K., Ross, A. S., Milojevic-Dupont, N., Jaques, N., Waldman-Brown, A., et al. (2022). Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2):1–96.

Rosales, C. and Meneveau, C. (2005). Linear forcing in numerical simulations of isotropic turbulence: Physical space implementations and convergence properties. *Physics of Fluids*, 17(9):095106.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Sagaut, P. (2006). *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media.

San, O., Staples, A. E., and Iliescu, T. (2015). A posteriori analysis of low-pass spatial filters for approximate deconvolution large eddy simulations of homogeneous incompressible flows. *International Journal of Computational Fluid Dynamics*, 29(1):40–66.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. (2020). Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. (2018). Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR.

Sarghini, F., De Felice, G., and Santini, S. (2003). Neural networks based subgrid scale modeling in large eddy simulations. *Computers & fluids*, 32(1):97–108.

Schneider, T., Lan, S., Stuart, A., and Teixeira, J. (2017a). Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations. *Geophysical Research Letters*, 44(24):12–396.

Schneider, T., Teixeira, J., Bretherton, C. S., Brient, F., Pressel, K. G., Schär, C., and Siebesma, A. P. (2017b). Climate goals and computing the future of clouds. *Nature Climate Change*, 7(1):3–5.

Schoenholz, S. and Cubuk, E. D. (2020). Jax md: a framework for differentiable physics. *Advances in Neural Information Processing Systems*, 33:11428–11441.

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., Mozaffari, A., and Stadtler, S. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A*, 379(2194):20200097.

Schumacher, J., Scheel, J. D., Krasnov, D., Donzis, D. A., Yakhot, V., and Sreenivasan, K. R. (2014). Small-scale universality in fluid turbulence. *Proceedings of the National Academy of Sciences*, 111(30):10961–10965.

Schumann, U. (1995). Stochastic backscatter of turbulence energy and scalar variance by random subgrid-scale fluxes. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 451(1941):293–318.

Shevchenko, I. and Berloff, P. (2015). Multi-layer quasi-geostrophic ocean dynamics in eddy-resolving regimes. *Ocean Modelling*, 94:1–14.

Sirignano, J. and Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364.

Smagorinsky, J. (1963). General circulation experiments with the primitive equations: I. the basic experiment. *Monthly weather review*, 91(3):99–164.

Sonnewald, M., Lguensat, R., Jones, D. C., Dueben, P., Brajard, J., and Balaji, V. (2021). Bridging observations, theory and numerical simulation of the ocean using machine learning. *Environmental Research Letters*.

Speziale, C. G. (1985). Galilean invariance of subgrid-scale stress models in the large-eddy simulation of turbulence. *Journal of Fluid Mechanics*, 156:55–62.

Sridhar, A., Tissaoui, Y., Marras, S., Shen, Z., Kawczynski, C., Byrne, S., Pamnany, K., Waruszewski, M., Gibson, T. H., Kozdon, J. E., et al. (2022). Large-eddy simulations with climatemachine v0. 2.0: a new open-source code for atmospheric simulations on gpus and cpus. *Geoscientific Model Development*, 15(15):6259–6284.

Stachenfeld, K., Fielding, D. B., Kochkov, D., Cranmer, M., Pfaff, T., Godwin, J., Cui, C., Ho, S., Battaglia, P., and Sanchez-Gonzalez, A. (2021). Learned coarse models for efficient turbulence simulation. *arXiv preprint arXiv:2112.15275*.

Stensrud, D. J. (2009). *Parameterization schemes: keys to understanding numerical weather prediction models*. Cambridge University Press.

Stevens, B. and Bony, S. (2013). What are climate models missing? *Science*, 340(6136):1053–1054.

Stewart, K., Hogg, A. M., Griffies, S., Heerdegen, A., Ward, M., Spence, P., and England, M. H. (2017). Vertical resolution of baroclinic modes in global ocean models. *Ocean Modelling*, 113:50–65.

Stolz, S. and Adams, N. A. (1999). An approximate deconvolution procedure for large-eddy simulation. *Physics of Fluids*, 11(7):1699–1701.

Subel, A., Chattopadhyay, A., Guan, Y., and Hassanzadeh, P. (2021). Data-driven subgrid-scale modeling of forced burgers turbulence using deep learning with generalization to higher reynolds numbers via transfer learning. *Physics of Fluids*, 33(3):031702.

Thompson, A. F. (2010). Jet formation and evolution in baroclinic turbulence with simple topography. *Journal of Physical Oceanography*, 40(2):257–278.

Trias, F., Folch, D., Gorobets, A., and Oliva, A. (2015). Building proper invariants for eddy-viscosity subgrid-scale models. *Physics of Fluids*, 27(6):065103.

Tsai, W.-P., Feng, D., Pan, M., Beck, H., Lawson, K., Yang, Y., Liu, J., and Shen, C. (2021). From calibration to parameter learning: Harnessing the scaling effects of big data in geoscientific modeling. *Nature communications*, 12(1):1–13.

Um, K., Brand, R., Fei, Y. R., Holl, P., and Thuerey, N. (2020). Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122.

Utke, J., Naumann, U., Fagan, M., Tallent, N., Strout, M., Heimbach, P., Hill, C.,

and Wunsch, C. (2008). Openad/f: A modular open-source tool for automatic differentiation of fortran codes. *ACM Transactions on Mathematical Software (TOMS)*, 34(4):1–36.

Vidard, A., Bouttier, P.-A., and Vigilant, F. (2015). Nemotam: tangent and adjoint models for the ocean modelling platform nemo. *Geoscientific Model Development*, 8(4):1245–1257.

Vinuesa, R. and Brunton, S. L. (2022). Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366.

Vollant, A., Balarac, G., and Corre, C. (2017). Subgrid-scale scalar flux modelling based on optimal estimation theory and machine-learning procedures. *Journal of Turbulence*, 18(9):854–878.

Waldrop, M. M. (2016). The chips are down for moore's law. *Nature News*, 530(7589):144.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. (2020). Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1457–1466.

Wang, Y. E., Wei, G.-Y., and Brooks, D. (2019). Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*.

Warhaft, Z. (2000). Passive scalars in turbulent flows. *Annual Review of Fluid Mechanics*, 32(1):203–240.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. (2018). 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2018)*, pages 10381–10392.

Weyn, J. A., Durran, D. R., Caruana, R., and Cresswell-Clay, N. (2021). Subseasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002502.

Wiens, J. A., Stralberg, D., Jongsomjit, D., Howell, C. A., and Snyder, M. A. (2009). Niches, models, and climate change: assessing the assumptions and uncertainties. *Proceedings of the National Academy of Sciences*, 106(supplement_2):19729–19736.

Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V. (2020). Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 1(1):1–34.

Wunsch, C. and Heimbach, P. (2013). Dynamically and kinematically consistent global ocean circulation and ice state estimates. In *International Geophysics*, volume 103, pages 553–579. Elsevier.

Xie, C., Wang, J., Li, K., and Ma, C. (2019). Artificial neural network approach to large-eddy simulation of compressible isotropic turbulence. *Physical Review E*, 99(5):053113.

Xie, C., Wang, J., and Weinan, E. (2020). Modeling subgrid-scale forces by spatial artificial neural networks in large eddy simulation of turbulence. *Physical Review Fluids*, 5(5):054606.

Yan, H., Du, J., Tan, V. Y., and Feng, J. (2019). On robustness of neural ordinary differential equations. *arXiv preprint arXiv:1910.05513*.

Zhou, L., Pan, S., Wang, J., and Vasilakos, A. V. (2017). Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361.

Zhou, Z., He, G., Wang, S., and Jin, G. (2019). Subgrid-scale model for large-eddy simulation of isotropic turbulent flows using an artificial neural network. *Computers & Fluids*, 195:104319.

Zhu, L., Zhang, W., Sun, X., Liu, Y., and Yuan, X. (2021). Turbulence closure for high reynolds number airfoil flows by deep neural networks. *Aerospace Science and Technology*, 110:106452.

Zhuang, J., Kochkov, D., Bar-Sinai, Y., Brenner, M. P., and Hoyer, S. (2021). Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. *Physical Review Fluids*, 6(6):064605.

Learning sub-grid dynamics in idealized turbulent systems

Apprentissage de la dynamique sous-maille dans des systèmes turbulents idéalisés

Abstract

Climate predictions and weather forecasting strongly rely on simulations of the Earth's oceans and atmosphere turbulent dynamics. But the simulation of turbulent processes is so computationally expansive that it is only possible to resolve the largest physical scales. The representation of unresolved scales in these simulations is therefore a key source of uncertainty and its modeling is still an open problem. Recently, machine learning techniques have been receiving growing attention for the design of parametrizations and subgrid-scale models. In this thesis, we explore the impact of explicitly embedding law invariances in neural networks trained to represent the small-scale dynamics of a scalar quantity advected by a turbulent flow. We also propose a new training algorithm inspired by the end-to-end approach applied to turbulence modeling, where the loss can be optimized on so-called « a posteriori » metrics. While the strategy gives promising results, it requires a differentiable numerical solver during the learning phase. We try to address this limitation with an additional step during which we train a differentiable emulator of the resolved dynamics. The error patterns in the emulator are shown to be propagated as a correction bias in the subgrid-scale model, limiting its performance. However, regularizing the model loss enable stable simulations and bring « a posteriori » learning benefits in non-differentiable solvers. Our results show that neural networks can respect physical principles and outperform classical models in long-term stable simulations. Their implementation in realistic solvers is expected to improve climate understanding and turbulence in general.

Keywords : subgrid modeling, machine learning, turbulence