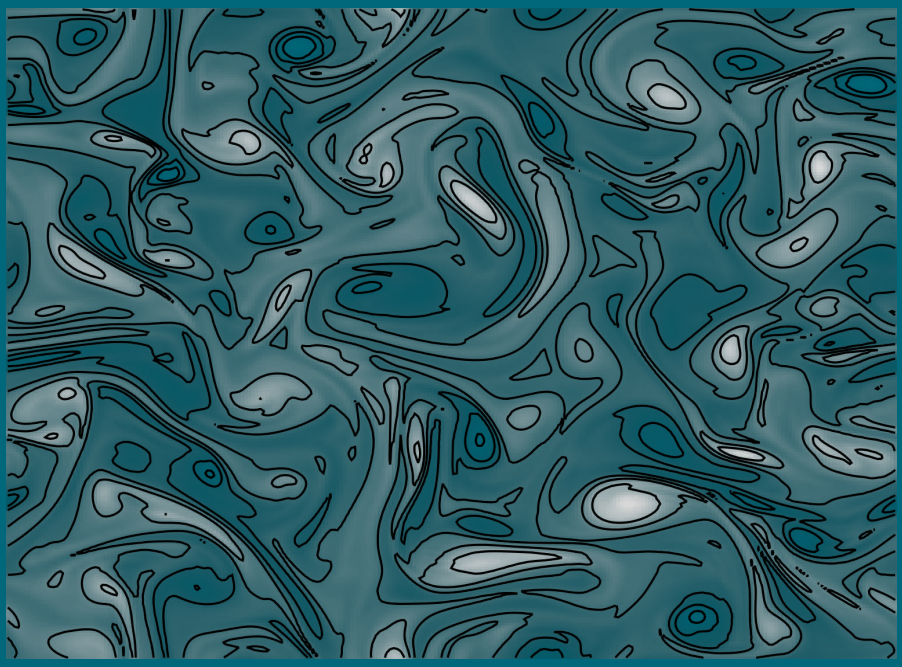
Structure-Preserving Data-Driven Methods for Modeling Turbulent Flows



Toby van Gastelen

Structure-Preserving Data-Driven Methods for Modeling Turbulent Flows

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr. S.K. Lenaerts, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op dinsdag 19 mei 2026 om 13:30 uur

door

Toby van Gastelen

geboren te Purmerend

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr. J.N. Kok
promotor:	dr.ir. B. Sanderse
copromotor(en):	dr.ir. W.N. Edeling (Centrum Wiskunde & Informatica)
leden:	prof.dr.ir. B. Koren
	prof.dr. K.P. Veroy-Grepl
	dr. R.P. Dwight (Technische Universiteit Delft)
	prof.dr. T. Iliescu (Virginia Tech)
	Prof.Dr.-Ing. N. Thürey (Technische Universität München)

Het onderzoek dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

© 2026 Toby van Gastelen

ISBN: 978-90-386-6696-9

Printed by: ProefschriftMaken, De Bilt, the Netherlands







The research presented in this thesis was carried out at the Centrum Wiskunde & Informatica (CWI), the national research institute for mathematics and computer science in the Netherlands. The thesis forms part of the project "Unraveling Neural Networks with Structure-Preserving Computing" (project number OCENW.GROOT.2019.044), funded under the NWO XL research programme by the Dutch Research Council (NWO). Additional financial support for this work was provided by the Eindhoven University of Technology.

Summary

In this thesis, we present novel data-driven approaches for efficient simulation of fluid flows. The ideas are mainly applied to the incompressible Navier-Stokes equations, but also to Burgers' equation, Korteweg-de Vries equation, and the linear advection equation. For the first three equations, we mainly stick to the large eddy simulation (LES) framework. In the LES framework, the small-scale fluctuations of the turbulent flow are discarded using a spatial filter. The filtering leads to an unclosed system of equations, including a closure term that still depends on the discarded small scales. From our inability to compute this term exactly during simulation time arises the need for modeling. During modeling, the closure term is approximated by a closure model, which solely depends on large-scale quantities. This closure model is then tasked with accounting for the effect of these small scales on the large scales. In our work, we often take the 'discretize first, filter next' approach. The advantage of first discretizing the equations is that this not only accounts for the absence of the small-scale fluctuations, but also for the spatial discretization error.

Following recent trends, we leverage the power of machine learning algorithms, and particularly convolutional neural networks (CNNs), to learn such closure models from high-fidelity simulations. After training the machine learning models, this results in a significant reduction of simulation time. However, even when large amounts of data are used to train the machine learning models, stability is not guaranteed. This is because these models do not inherently abide by certain physical structure of the fluid flow equations, such as energy conservation. The result is that physics-agnostic machine learning-based closure models often cause instabilities during the simulation. In this thesis, we aim to resolve these instabilities.

To achieve stable and physics-aware closure models, we take inspiration from classical numerical discretization techniques to build this structure directly into the machine learning models. The result is a specialized formulation for the closure model, based on existing data-driven algorithms. Besides CNNs, we also consider linear models in the evolve-filter-relax (EFR) framework, which is closely related to LES.

In addition to LES, we also consider reduced-order models (ROMs). These are related to LES in the fact that both reduce the dimension of the problem. However, in ROMs

this dimension reduction is not achieved using a physical filter, but using data-driven approaches to obtain a reduced basis. This is typically done by carrying out a singular value decomposition (SVD) on simulation data. However, the resulting reduced basis is notoriously bad at representing turbulent flows. In this thesis, we suggest a new way to obtain this basis, which is more suitable for representing turbulent flows.

The core contributions are presented in the four main chapters of the thesis:

2) Energy-Conserving Neural Network for Turbulence Closure Modeling: In this chapter, we consider Burgers' equation and Korteweg-de Vries equation (in 1D). A spatial averaging filter is introduced and applied to a high resolution discretization. We show that the turbulent fluctuations, removed by the filter, can be compressed and their energy reintroduced into the system. This leads to a new energy conservation law. A structure-preserving neural network architecture is introduced, which satisfies this law. This results in improved stability and accuracy for the resulting LES, as opposed to standard CNNs.

3) Energy-Conserving Neural Network Closure Model for Long-Time Accurate and Stable 2D LES: In this chapter, we consider the incompressible Navier-Stokes equations in 2D. We build upon the work in Chapter 2 by applying the same neural network architecture in 2D, but without the compressed turbulent representation, as their representation for multiple dimensions is still an open problem. The result is a strictly dissipative closure model. We show that the model outperforms standard CNNs, as well as existing eddy-viscosity models, while producing stable simulations. For long-time simulations, the benefits become especially clear when looking at statistical quantities such as energy spectra.

4) A New Data-Driven Energy-Stable Evolve-Filter-Relax Model for Turbulent Flow Simulation: In this chapter, we take another perspective on LES by working in the EFR framework. This framework serves a similar purpose as LES, but has the advantage of being easier to integrate into existing codes. Our contribution consists of proposing a new data-driven filter, efficiently applied in Fourier space, which replaces the differential filter commonly used in the EFR framework. The data-driven filter is optimized efficiently by solving a simple least-squares optimization problem. This circumvents the need for computationally expensive neural network training. Using limited training data, the filter is capable of outperforming existing approaches. In the sparse data regime, applying physical constraints, such as energy and enstrophy conservation, offers further stability and accuracy benefits.

5) Modeling Advection-Dominated Flows with Space-Local Reduced-Order Models: In this final contribution, we step away from LES and consider projection-based ROMs.

In particular, we address the limited capability of the proper orthogonal decomposition (POD) basis to represent advection-dominated flows (such as turbulent flows). As a solution for this, we propose a space-local POD basis. Furthermore, we introduce a space-local POD basis with overlapping subdomains, which draws inspiration from a finite element basis to obtain a smooth reconstruction. We show that both space-local approaches result in a basis which generalizes better for advection-dominated problems. Due to the local support of the space-local bases the resulting ROMs not only generalize better, but are also more computationally efficient than standard POD Galerkin ROMs.

This thesis offers novel insights into the benefits of building physical structure into data-driven methods for fluid simulations. The introduced approaches serve as an important building block to bringing structure-preserving data-driven approaches to simulating real-life flow scenarios.

Samenvatting

In dit proefschrift presenteren we nieuwe data-gedreven benaderingen voor de efficiënte simulatie van stromingen. De ideeën worden voornamelijk toegepast op de incompressibele Navier–Stokes vergelijkingen, maar ook op de Burgers vergelijking, de Korteweg–de Vries vergelijking en de lineaire advectievergelijking. Voor de eerste drie vergelijkingen blijven we grotendeels binnen het large eddy simulation (LES)-raamwerk. Binnen het LES-raamwerk worden de kleinschalige fluctuaties van een turbulente stroming verwijderd door middel van een ruimtelijk filter. Het toepassen van dit filter leidt tot een niet-gesloten stelsel van vergelijkingen, waarin een sluitingsterm voorkomt die nog steeds afhankelijk is van de verworpen kleine schalen. Doordat deze term niet exact te berekenen is tijdens de simulatie ontstaat de noodzaak tot modellering. Tijdens deze modellering wordt de sluitingsterm benaderd door een sluitingsmodel dat uitsluitend afhankelijk is van grootschalige grootheden. Dit sluitingsmodel heeft als taak het effect van de kleine schalen op de grote schalen te modelleren. In ons werk hanteren we vaak de strategie 'eerst discretiseren, daarna filteren'. Het voordeel van deze aanpak is dat niet alleen rekening wordt gehouden met het ontbreken van kleinschalige fluctuaties, maar ook met de fout die voortkomt uit de ruimtelijke discretisatie.

Aansluitend bij recente trends benutten we de kracht van machine learning-algoritmen, en in het bijzonder convolutionele neurale netwerken (CNNs), om dergelijke sluitingsmodellen te leren uit hoge-resolutie simulaties. Na training van de machine learning modellen, resulteert dit in een aanzienlijke reductie van de rekentijd. Echter, zelfs wanneer grote hoeveelheden data worden gebruikt om de machine learning modellen te trainen, is stabiliteit niet gegarandeerd. Dit komt doordat deze modellen niet intrinsiek voldoen aan bepaalde fysische structuren van de stromingsvergelijkingen, zoals energiebehoud. Het gevolg is dat fysisch-agnostische op machine learning gebaseerde sluitingsmodellen vaak instabiliteiten veroorzaken tijdens simulaties. In dit proefschrift streven we ernaar deze instabiliteiten te verhelpen.

Om stabiele en fysisch consistente sluitingsmodellen te verkrijgen, laten we ons inspireren door klassieke numerieke discretisatietechnieken en bouwen we deze structuur direct in de machine learning modellen in. Dit leidt tot een gespecialiseerde formulering van het

sluitingsmodel, gebaseerd op bestaande data-gedreven algoritmen. Naast CNNs beschouwen we ook lineaire modellen binnen het EFR-raamwerk, dat nauw verwant is aan LES.

Naast LES behandelen we ook gereduceerde-orde-modellen (ROMs). Deze zijn verwant aan LES in de zin dat beide methoden de dimensie van het probleem reduceren. Bij ROMs wordt deze dimensiereductie echter niet bereikt door middel van een ruimtelijk filter, maar via data-gedreven technieken om een gereduceerde basis te construeren. Dit gebeurt doorgaans door het uitvoeren van een singuliere-waardenontbinding (SVD) op simulatiegegevens. De resulterende basis staat er echter om bekend slecht geschikt te zijn voor het representeren van turbulente stromingen. In dit proefschrift stellen we een nieuwe manier voor om een dergelijke basis te construeren die meer geschikt is voor het representeren van turbulente stromingsvelden.

De kernbijdragen worden uiteengezet in de vier kernhoofdstukken van dit proefschrift:

2) Energiebehoudend Neuraal Netwerk voor Turbulentie-Sluitingsmodellering: In dit hoofdstuk beschouwen we de Burgers-vergelijking en de Korteweg–de Vries-vergelijking (in 1D). Een ruimtelijke gemiddelde filter wordt geïntroduceerd en toegepast op een hoge-resolutie discretisatie. We laten zien dat de turbulente fluctuaties die door het filter worden verwijderd, kunnen worden gecomprimeerd en dat hun energie opnieuw in het systeem kan worden ingebracht. Dit leidt tot een nieuwe energiebehoudswet. Vervolgens introduceren we een structuur-behoudende neurale netwerkarchitectuur die aan deze wet voldoet. Dit resulteert in verbeterde stabiliteit en nauwkeurigheid van de resulterende LES ten opzichte van standaard CNNs.

3) Energiebehoudend Neuraal-Netwerk-Sluitingsmodel voor Langdurig Nauwkeurige en Stabiele 2D LES: In dit hoofdstuk behandelen we de incompressibele Navier–Stokes-vergelijkingen in twee dimensies. We bouwen voort op het werk in Hoofdstuk 2 door dezelfde neurale netwerkarchitectuur toe te passen in 2D, maar zonder de gecomprimeerde representatie van turbulente fluctuaties, aangezien een dergelijke representatie in meerdere dimensies nog een open probleem vormt. Het resultaat is een strikt dissipatief sluitingsmodel. We tonen aan dat dit model beter presteert dan standaard CNNs en bestaande eddy-viscositeitsmodellen, terwijl stabiele simulaties worden verkregen. Bij langdurige simulaties worden de voordelen vooral duidelijk bij het beschouwen van statistische grootheden zoals energiespectra.

4) Een Nieuw Data-Gedreven Energie-Stabiel Evolve-Filter-Relax Model voor Turbulente Stromingssimulaties: In dit hoofdstuk bekijken we LES vanuit een ander perspectief door te werken binnen het EFR-raamwerk. Dit raamwerk dient een vergelijkbaar doel als LES, maar heeft als voordeel dat het eenvoudiger te integreren is in bestaande codes. Onze bijdrage bestaat uit het voorstellen van een nieuw data-gedreven filter, dat efficiënt

wordt toegepast in de Fourierruimte en het differentiaalfilter vervangt dat gebruikelijk is binnen het EFR-raamwerk. Het data-gedreven filter wordt efficiënt geoptimaliseerd door het oplossen van een eenvoudig kleinste-kwadratenprobleem, waarmee de noodzaak van dure training van neurale netwerken wordt vermeden. Met beperkte trainingsdata is het filter in staat bestaande benaderingen te overtreffen. In het regime van schaarse data bieden fysische restricties, zoals energie- en enstrofiebehoud, aanvullende voordelen op het gebied van stabiliteit en nauwkeurigheid.

5) Modellering van Advectie-Gedomineerde Stromingen met Ruimtelijk-Lokale Gereduceerde-Orde-Modellen: In deze laatste bijdrage stappen we af van LES en beschouwen we projectiegebaseerde ROMs. In het bijzonder richten we ons op de beperkte geschiktheid van de gepaste orthogonale ontbinding (POD) basis voor het representeren van advectie-gedomineerde stromingen (zoals turbulente stromingen). Als oplossing stellen we een ruimtelijk lokale POD basis voor. Daarnaast introduceren we een ruimtelijk lokale POD basis met overlappende subdomeinen, geïnspireerd door een eindige-elementenbasis, om een gladde reconstructie te verkrijgen. We tonen aan dat beide ruimtelijk lokale benaderingen leiden tot een basis die beter generaliseert voor advectie-gedomineerde problemen. Door de lokale ondersteuning van de basis generaliseren de resulterende ROMs niet alleen beter, maar zijn ook rekenkundig efficiënter dan standaard POD-Galerkin ROMs.

Dit proefschrift biedt nieuwe inzichten in de voordelen van het inbouwen van fysische structuur in data-gedreven methoden voor stromingssimulaties. De geïntroduceerde benaderingen vormen een belangrijke bouwsteen voor het toepassen van structuur-behoudende data-gedreven methoden bij de simulatie van realistische stromingsscenario's.

Contents

Summary iii

Samenvatting vii

1 Introduction 5

1.1 Incompressible Navier–Stokes equations . . . 6
1.1.1 Physical structure . . . 7
1.2 Spatial discretization . . . 9
1.2.1 Staggered grid discretization . . . 9
1.3 Large eddy simulation . . . 10
1.3.1 Filtering and the subgrid-scale stress tensor . . . 12
1.3.2 Modeling the SGS stress . . . 12
1.4 Machine learning approaches . . . 14
1.4.1 Neural networks . . . 14
1.4.2 Convolutional neural networks . . . 15
1.5 Research topics . . . 17
1.5.1 Topic I: Machine learning-based closure models for large eddy simulation 17
1.5.2 Topic II: Data-driven extensions of the evolve-filter-relax framework . 18
1.5.3 Topic III: Reduced-order modeling for advection-dominated flows . . . 19
1.6 Structure of the thesis . . . 21
1.6.1 Topic I: Machine learning-based closure models for large eddy simulation (Chapters 2 & 3) . . . 22
1.6.2 Topic II: Data-driven extensions of the evolve-filter-relax framework (Chapter 4) . . . 22
1.6.3 Topic III: Reduced-order modeling for advection-dominated flows (Chapter 5) . . . 23
1.7 List of publications . . . 25

2 Energy-Conserving Neural Network for Turbulence Closure Modeling 27
2.1 Introduction . . . 28
2.2 Governing equations, discrete filtering, and closure problem . . . 31
2.2.1 Spatial discretization . . . 31
2.2.2 Burgers' and Korteweg-de Vries equation, and physical structure . . . 31
2.2.3 Discrete filtering . . . 33
2.2.4 Discrete closure problem . . . 35
2.2.5 Inner products and energy decomposition . . . 36
2.2.6 Momentum conservation . . . 37
2.3 Structure-preserving closure modeling framework . . . 37
2.3.1 The framework . . . 37
2.3.2 SGS variables . . . 39
2.3.3 Construction of the operators . . . 40
2.3.3.1 Diffusive operator . . . 41
2.3.3.2 Advective operator . . . 41
2.3.3.3 Momentum conservation . . . 42
2.3.3.4 Properties & further discussion . . . 43
2.3.4 Finding the optimal parameter values . . . 44
2.4 Results . . . 45
2.4.1 Test cases . . . 45
2.4.1.1 Considered closure models . . . 45
2.4.2 Training the closure models . . . 46
2.4.3 Closure model performance . . . 47
2.4.3.1 Convergence . . . 47
2.4.3.2 Computation time . . . 48
2.4.4 Consistency of the training procedure . . . 49
2.4.5 Structure preservation . . . 50
2.4.6 Burgers' equation & energy spectra . . . 52
2.4.7 Extrapolation in parameter space . . . 53
2.4.8 Extrapolation in space and time . . . 55
2.5 Discussion on the applicability to 2D/3D Navier-Stokes . . . 57
2.6 Conclusion . . . 58

3 Energy-Conserving Neural Network Closure Model for Long-Time Accurate and Stable LES 61
3.1 Introduction . . . 62
3.2 Preliminaries . . . 65
3.2.1 Navier-Stokes equations . . . 65

3.2.2 Physical structure . . . 65
3.2.3 Discretization . . . 66
3.2.4 Structure of the discretization . . . 66
3.2.5 Pressure projection . . . 67
3.3 Closure modeling . . . 68
3.3.1 Filtering . . . 68
3.3.2 System of equations . . . 68
3.3.3 Energy analysis of the closure term . . . 70
3.3.4 Fitting of the model parameters . . . 70
3.4 Methodology . . . 72
3.4.1 Smagorinsky model . . . 72
3.4.2 Neural network closure . . . 73
3.4.3 Skew-symmetric framework . . . 74
3.4.3.1 Skew-symmetric term . . . 75
3.4.3.2 Negative-definite term . . . 76
3.4.4 Overview of the closure models . . . 77
3.5 Results . . . 77
3.5.1 Experimental setup . . . 77
3.5.2 Decaying turbulence . . . 79
3.5.3 Consistency of closure model performance . . . 86
3.5.4 Kolmogorov flow . . . 88
3.5.5 Comparison to the dynamic Smagorinsky model . . . 91
3.5.6 Limitations of Machine Learning Closure Models . . . 92
3.6 Conclusion . . . 93

4 A New Data-Driven Energy-Stable Evolve-Filter-Relax Model for Turbulent Flow Simulation 95
4.1 Introduction . . . 96
4.2 Preliminaries . . . 99
4.2.1 The Navier-Stokes equations . . . 99
4.2.2 Finite volume discretization . . . 100
4.2.3 Evolve-Filter-Relax . . . 101
4.2.3.1 Stability of the differential filter . . . 102
4.3 Methods . . . 104
4.3.1 Data-driven linear filter . . . 104
4.3.2 Conservation of energy using χ . . . 106
4.3.3 Conservation of enstrophy using χ . . . 108
4.4 Numerical results . . . 109

4.4.1 Test case 1: two-dimensional decaying homogeneous turbulence 111
4.4.1.1 Data-driven EFR in the case of full data 111
4.4.1.2 Data-driven EFR in the case of scarce data 116
4.4.2 Test case 2: two-dimensional Kolmogorov flow 119
4.4.2.1 Data-driven EFR in the case of full data 119
4.4.2.2 Data-driven EFR in the case of scarce data 120
4.4.3 Computational time considerations 122
4.5 Conclusion and Outlook 124

5 Modeling Advection-Dominated Flows with Space-Local Reduced-Order Models 127
5.1 Introduction 128
5.2 Full-order model 131
5.2.1 Advection equation 131
5.2.2 Finite difference discretization 131
5.2.3 Energy conservation of the FOM 132
5.3 Local and global POD 132
5.3.1 Global POD 133
5.3.2 Space-local POD 134
5.3.3 Local POD with overlapping subdomains 137
5.4 Space-local, energy-conserving reduced-order model 139
5.4.1 Projection on reduced basis 139
5.4.2 Galerkin projection 140
5.4.3 Energy conservation of the ROM 141
5.5 Results & discussion 141
5.5.1 Test case setup 141
5.5.2 Projection error 143
5.5.3 Resulting basis 144
5.5.4 Accuracy and energy conservation of ROM 144
5.5.5 Convergence with increasing ROM dimension 148
5.5.6 2D advection-diffusion equation 149
5.5.6.1 2D basis 150
5.5.6.2 ROM performance 151
5.5.7 Applicability to 2D Navier-Stokes 153
5.6 Conclusions 155

6 Conclusion **159**
6.1 Conclusions . . . 159
6.1.1 Topic I: Machine learning-based closure models for large eddy simulation (Chapters 2 & 3) . . . 160
6.1.2 Topic II: Data-driven extensions of the evolve-filter-relax framework (Chapter 4) . . . 161
6.1.3 Topic III: Reduced-order modeling for advection-dominated flows (Chapter 5) . . . 162
6.2 Outlook and future work . . . 163
6.2.1 General outlook . . . 164

Bibliography **167**

Curriculum Vitae **185**

List of Presentations **187**

Acknowledgements **189**

7 Appendix **191**
A Appendix Chapter 2 . . . 191
A.1 Filter properties . . . 191
A.2 Comparing coarse and fine-grid dissipation . . . 192
A.3 SGS compression . . . 192
A.3.1 Non-periodic boundary conditions . . . 194
A.3.2 Training procedure . . . 197
B Appendix Chapter 3 . . . 201
B.1 Stucture-preserving finite volume discretization . . . 201
B.2 Pressure projection . . . 202
B.3 Motivation behind skew-symmetric architecture . . . 202
B.4 Training of the neural networks . . . 205
B.5 Vorticity fields . . . 206
C Appendix Chapter 4 . . . 209
C.1 Parameter optimization in state-of-the-art approaches . . . 209
C.2 Additional results on data-driven EFR strategies . . . 212
C.2.1 Decaying homogeneous turbulence test case . . . 212
C.2.2 Kolmogorov test case . . . 214
D Appendix Chapter 5 . . . 216
D.1 2D Bump kernel . . . 216

D.2 Choice of subdomains and modes for 2D test case 216
D.3 Time step size 217

Acronyms

BC boundary condition.

CNN convolutional neural network.

CNN-C convolutional neural network with backscatter clipping.

DIV divergence of stress tensor predicted by a neural network.

DNS direct numerical simulation.

DOF degrees of freedom.

EFR evolve-filter-relax.

FDNS filtered direct numerical simulation.

FFT fast Fourier transform.

FOM full-order model.

G-POD space-global proper orthogonal decomposition.

I-NRMSE integrated normalized root-mean-squared-error.

I/O input/output.

KDE kernel density estimate.

KdV Korteweg de-Vries.

L-POD space-local proper orthogonal decomposition.

LES large eddy simulation.

LO-POD space-local proper orthogonal decomposition with overlapping subdomains.

LSTM long short-term memory.

MSE mean squared error.

NC no closure.

NMRSE normalized root-mean-squared-error.

NN neural network.

NSE incompressible Navier-Stokes equations.

PDE partial differential equation.

PINN physics informed neural network.

POD proper orthogonal decomposition.

RANS Reynolds-averaged Navier-Stokes.

RHS right-hand side.

RK4 Runge-Kutta 4.

ROM reduced-order model.

SGS subgrid-scale.

SKEW skew-symmetric neural network architecture.

SM Smagorinsky model (Chapter 2).

SMAG Smagorinsky model (Chapter 3).

SP structure-preserving neural network architecture.

SVD singular value decomposition.

1

Introduction

The incompressible Navier-Stokes equations model the motion of fluids in different scenarios, such as the flow of liquids in pipes, atmospheric flows for weather predictions, and the air flow around the wings of an aircraft, to name a few [1, 2]. As such, this set of equations is of great interest in many disciplines of science and engineering. Given the equations' nonlinear and complex nature, numerical methods have become essential tools for exploring fluid behavior and obtaining approximate solutions in practical applications [3, 4].

The difficulty in obtaining these approximations arises from the nature of turbulence. Turbulent flows are characterized by a wide range of interacting spatial and temporal scales, leading to highly irregular and chaotic motion [2]. The occurrence of turbulence in a flow is mostly determined by the Reynolds number, a dimensionless quantity that depends on the ratio between the inertial and viscous forces in a flow [5]. When the Reynolds number is low (less inertial/more viscous forces), the flow is typically laminar, whereas a high Reynolds number (more inertial/less viscous forces) typically results in a turbulent flow. In this way, the Reynolds number essentially quantifies the expected flow behavior. Turbulence is especially problematic in numerical simulations, since fine computational grids are required to resolve all turbulent fluctuations in the flow [6]. This makes direct numerical simulation (DNS), simulations that resolve all turbulent motions directly from the Navier–Stokes equations, of high Reynolds number flows computationally expensive and generally infeasible. Alleviating

this problem is an active area of research [6]. Existing approaches typically involve reducing the spatial degrees of freedom of the system. Such approaches include reduced-order models (ROMs) [7, 8], Reynolds-averaged Navier-Stokes (RANS) [9], and large eddy simulation (LES) [6, 10], ordered from a large to smaller reduction in degrees of freedom. In this thesis, we solely concern ourselves with LES and ROMs, which will both be introduced in this chapter.

With recent increases in computing power, data-driven approaches have emerged as viable methods to simulate physical systems [7, 11]. In particular, machine learning approaches have proven to be powerful tools. However, using such methods to enhance ROMs and LES has its drawbacks: the lack of adherence to physical structure, such as conservation laws, can lead to inaccurate predictions and even instabilities during the simulation [12–16].

The primary focus of this thesis is to incorporate physical structure into data-driven approaches to ensure physical consistency, combat stability issues, and, in turn, produce more accurate simulation results. The first part of the research focuses on developing specialized machine learning algorithms within the context of LES. This work is presented in Chapters 2 and 3. Next, we move towards the evolve-filter-relax (EFR) framework, which is closely related to LES but offers the advantage of being easier to integrate into existing codebases [17, 18]. This work is presented in Chapter 4. Here, we mostly focus on linear models. The final research topic of this thesis concerns the generalizability of ROMs for advection-dominated flows, which are notoriously challenging to capture in a ROM setting [19, 20]. This work is presented in Chapter 5.

For the remainder of this chapter, we discuss the basic preliminaries underlying these contributions, introduce a set of research topics, and conclude by formulating a set of research questions for each of the topics that this thesis aims to address.

1.1 Incompressible Navier–Stokes equations

The incompressible Navier–Stokes equations in non-dimensional conservative form are written as the following partial differential equation (PDE) [1, 2]:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}^T) = -\nabla p + \nu \Delta \mathbf{u} + \mathbf{f}, \tag{1.1a}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{1.1b}$$

which describes the behavior of a d-dimensional velocity field $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^d$ evolving in space $\mathbf{x} \in \mathbb{R}^d$ and time t, where ν represents the kinematic viscosity. These equations are obtained by non-dimensionalizing the dimensional incompressible Navier-Stokes equations for a fluid of constant density ρ and dynamic viscosity μ, using a characteristic length scale L and reference velocity U, with $\nu = \mu/\rho$. The resulting dimensionless system is governed by the

Reynolds number

$$\mathrm{Re} = \frac{\rho L U}{\mu} = \frac{L U}{\nu}.$$

Without loss of generality, we choose $L = 1$ and $U = 1$, so that the kinematic viscosity appearing in (1.1a) satisfies

$$\nu = \frac{1}{\mathrm{Re}}.$$

The terms in the PDE describe the different forces acting on the velocity field. From left to right, these contributions are the convective force, the pressure gradient, and viscous diffusion. The final term $\mathbf{f}(\mathbf{x}, t) \in \mathbb{R}^d$ represents external forces, such as gravity.

1.1.1 Physical structure

The incompressible Navier–Stokes equations represent a set of fundamental physical laws, namely: conservation of mass, momentum, and as a consequence, kinetic energy (in the absence of dissipation) [1, 21]. These will be collectively referred to by us as the physical structure of the system. Conservation of mass for an incompressible fluid can easily be shown by computing the change of mass in an arbitrary volume $\mathcal{V}$ due to the flux across the surface $\mathcal{S}$:

$$\oint_{\mathcal{S}} \mathbf{u} \cdot \mathbf{n} \mathrm{d}S = \int_{\mathcal{V}} \nabla \cdot \mathbf{u} \mathrm{d}V = 0, \tag{1.2}$$

where we used the divergence theorem to write the surface integral as a volume integral. $\mathbf{n}$ represents the outward unit normal vector of $\mathcal{S}$ and dS denotes the corresponding scalar surface-area element.

Next, we consider the momentum of the system. Let $\Omega \subset \mathbb{R}^d$ denote the spatial domain occupied by the fluid. The total momentum is defined as

$$\mathbf{P} := \int_{\Omega} \mathbf{u} \, \mathrm{d}\Omega, \tag{1.3}$$

where Ω is assumed to be a periodic domain with boundary $\partial\Omega$, and $\mathrm{d}V$ and $\mathrm{d}\Omega$ are used interchangeably. The change in momentum is computed as follows:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} &= \int_{\Omega} \frac{\partial \mathbf{u}}{\partial t} \, \mathrm{d}\Omega \\ &= -\oint_{\partial\Omega} (\mathbf{u}\mathbf{u}^T) \cdot \mathbf{n} \mathrm{d}S - \oint_{\partial\Omega} p\mathbf{n} \, \mathrm{d}S + \nu \oint_{\partial\Omega} \nabla \mathbf{u} \cdot \mathbf{n} \mathrm{d}S + \int_{\Omega} \mathbf{f} \, \mathrm{d}\Omega \\ &= \int_{\Omega} \mathbf{f} \, \mathrm{d}\Omega, \end{aligned} \tag{1.4}$$

where we substituted (1.1a) and rewrote most terms as boundary integrals, which vanish on periodic domains (which we assume here). This means that the momentum in each direction

only changes due to the body force.

The total (kinetic) energy of the system is defined as

$$E := \frac{1}{2}\int_\Omega \mathbf{u}\cdot\mathbf{u}\,\mathrm{d}\Omega. \tag{1.5}$$

Using the product rule we can write the change in energy as

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \int_\Omega \mathbf{u}\cdot\frac{\partial\mathbf{u}}{\partial t}\mathrm{d}\Omega = \int_\Omega \mathbf{u}\cdot(-\nabla\cdot(\mathbf{u}\mathbf{u}^T) - \nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f})\mathrm{d}\Omega. \tag{1.6}$$

The pressure and viscous contributions can be simplified using integration by parts:

$$\int_\Omega -\mathbf{u}\cdot\nabla p\mathrm{d}\Omega = \int_\Omega p(\nabla\cdot\mathbf{u})\mathrm{d}\Omega = 0, \tag{1.7}$$

$$\int_\Omega \nu\mathbf{u}\cdot\nabla^2\mathbf{u}\mathrm{d}\Omega = -\int_\Omega \nu||\nabla\mathbf{u}||_2^2\mathrm{d}\Omega, \tag{1.8}$$

where we used the fact $\nabla\cdot\mathbf{u} = 0$ and that the boundary contributions cancel on periodic domains.

To find the energy contribution we write the convective term in non-conservative form:

$$\nabla\cdot(\mathbf{u}\mathbf{u}^T) = (\mathbf{u}\cdot\nabla)\mathbf{u} + \mathbf{u}(\nabla\cdot\mathbf{u}) = (\mathbf{u}\cdot\nabla)\mathbf{u}, \tag{1.9}$$

which is simplified using the divergence-freeness of the velocity field. This allows us to rewrite the term as

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}) - \mathbf{u}\times(\nabla\times\mathbf{u}), \tag{1.10}$$

using a vector calculus identity. We then take the inner product with $\mathbf{u}$ to obtain

$$\mathbf{u}\cdot\nabla\cdot(\mathbf{u}\mathbf{u}^T) = \mathbf{u}\cdot\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}) - \mathbf{u}\cdot(\mathbf{u}\times(\nabla\times\mathbf{u})) = \mathbf{u}\cdot\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}), \tag{1.11}$$

where the second term cancels because the cross product between two vectors is orthogonal to both of these vectors. Using the product rule, starting from $\nabla\cdot(\mathbf{u}(\mathbf{u}\cdot\mathbf{u}))$, this is rewritten to

$$\mathbf{u}\cdot\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}) = \frac{1}{2}\nabla\cdot(\mathbf{u}(\mathbf{u}\cdot\mathbf{u})) - \frac{1}{2}(\mathbf{u}\cdot\mathbf{u})(\nabla\cdot\mathbf{u}) = \frac{1}{2}\nabla\cdot(\mathbf{u}(\mathbf{u}\cdot\mathbf{u})), \tag{1.12}$$

which is in divergence form. Once again, we used the fact that $\mathbf{u}$ is divergence-free to simplify this expression. This term integrates to zero:

$$\int_\Omega \mathbf{u}\cdot\nabla\cdot(\mathbf{u}\mathbf{u}^T)\mathrm{d}\Omega = \int_\Omega \frac{1}{2}\nabla\cdot(\mathbf{u}(\mathbf{u}\cdot\mathbf{u}))\mathrm{d}\Omega = \oint_{\partial\Omega}\frac{1}{2}(\mathbf{u}(\mathbf{u}\cdot\mathbf{u}))\cdot\mathbf{n}\mathrm{d}S = 0, \tag{1.13}$$

due to the divergence theorem and the fact that Ω is a periodic domain. The change in

energy is finally written as

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\int_{\Omega} \nu ||\nabla \mathbf{u}||_2^2 \mathrm{d}\Omega + \int_{\Omega} \mathbf{u} \cdot \mathbf{f} \mathrm{d}\Omega, \tag{1.14}$$

which means that the energy is always decreasing (or constant for $\nu = 0$) in the absence of forcing [2, 21].

1.2 Spatial discretization

To obtain numerical solutions of the incompressible Navier–Stokes equations, one first has to discretize the continuous problem. While discretizing it is important to account for the underlying physical structure of the system, which we discussed in the previous section. Besides mass and momentum conservation, energy dissipation and, in the inviscid limit, conservation of kinetic energy, is a particularly desirable property for the discretization to inherit. This is because this property dictates the stability and physical fidelity of simulations [22].

For instance, a naive discretization of the convective term can lead to artificial energy growth, which is unphysical and may destabilize computations. To avoid this, structure-preserving schemes reformulate the convective operator in skew-symmetric form, ensuring that the discrete energy balance mirrors its continuous counterpart. Such discretizations guarantee that, in the absence of viscosity and external forcing, the total kinetic energy is conserved exactly up to machine precision. This feature is particularly important when simulating turbulent flows, where spurious energy errors can pollute the dynamics over long timescales [23, 24].

1.2.1 Staggered grid discretization

While discretizing, the aim is to approximate the infinite-dimensional PDE by a finite-dimensional system of algebraic equations that can be solved on a computer. The discretization involves both space and time, and, as stated earlier, the choice of method can have a significant impact on stability, accuracy, and the preservation of physical structure. In this section, we focus on a spatial discretization, where the computational domain is partitioned into a finite number of control volumes. The velocity and pressure fields are then approximated on this discrete mesh. In this thesis, we make use of the second-order finite difference discretization presented in [24, 25]. This discretization employs a staggered grid, such as depicted in Figure 1.1 for a 2D velocity field $\mathbf{u} = \begin{bmatrix} u & v \end{bmatrix}^T$, and is used because it respects the underlying conservation laws and structural properties of the original equations. As stated earlier, such schemes tend to be more robust and physically faithful, especially for

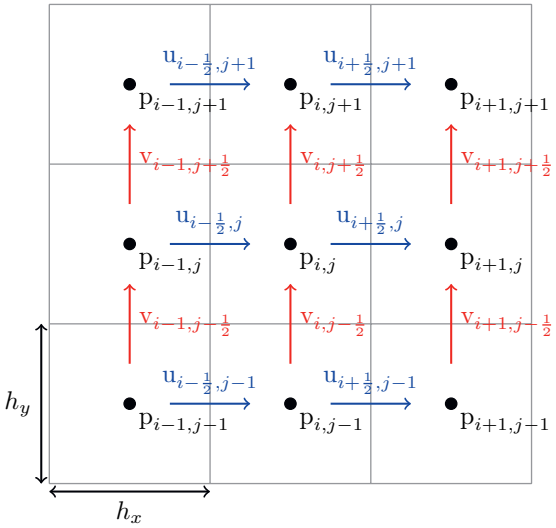


Figure 1.1: Staggered grid discretization of Navier-Stokes equations. The grid-spacings are indicated by h_x and h_y for the x and y direction, respectively. The pressure points are indexed with whole numbers, while for the velocity components we have an offset of $\frac{1}{2}$ in the appropriate direction.

long-time simulations. In particular, this scheme discretely satisfies conservation of mass, momentum, and energy, see (1.1b), (1.4), and (1.14), respectively.

1.3 Large eddy simulation

As stated earlier, DNS of the Navier-Stokes equations is prohibitively expensive at high Reynolds numbers due to the wide range of dynamically active scales that must be resolved [2, 6]. Central to understanding the resolution requirements of turbulent simulations is the Kolmogorov microscale η, which represents the smallest dissipative scales in the flow. The corresponding wavenumber $k_\eta \sim 1/\eta$ marks the onset of the dissipation range in the energy spectrum. The ratio between the largest and smallest scales grows with Reynolds number as $\mathrm{Re}^{3/4}$, making DNS prohibitively expensive at high Re as all scales down to k_η must be resolved.

Large eddy simulation (LES) offers a practical alternative to DNS by explicitly resolving only the large, energy-containing structures of the flow, while modeling the effect of unresolved small scales [2, 26]. This is achieved by applying a spatial filter to the governing equations, which separates the resolved scales from the unresolved scales [27]. The filtered equations retain the same general structure as the original Navier-Stokes system but include an additional subgrid-scale (SGS) stress term that must be modeled. This will be discussed

later in this section.

Because LES relies on resolving only a limited range of dynamically important scales, a spectral description of turbulence provides a natural framework for assessing which portions of the flow are captured explicitly and how energy is transferred to unresolved scales. In this context, the energy spectrum $E(k)$ provides a convenient quantitative measure of how turbulent kinetic energy is distributed across scales. Low wavenumbers correspond to large energy-containing eddies, while high wavenumbers are associated with small dissipative scales. The total turbulent kinetic energy is given by $\int_0^\infty E(k)dk$. Between the energy-containing and dissipation ranges lies the inertial subrange, where energy cascades from large to small scales at a constant rate. The inertial range is characterized by $E(k) \sim k^{-5/3}$ for 3D turbulence [28].

Figure 1.2 shows the energy spectrum $E(k)$ as a function of wavenumber k for different simulation approaches. The DNS resolves the full spectrum down to the Kolmogorov scale k_η,

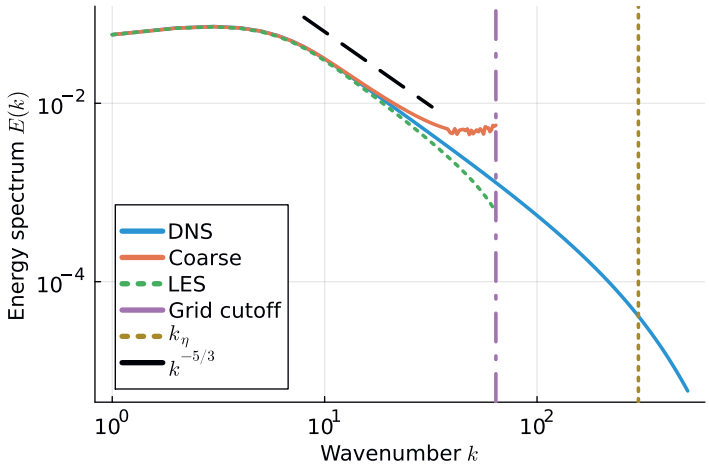


Figure 1.2: Schematic of the energy spectrum $E(k)$ as a function of wavenumber k for different simulation approaches. The DNS resolves all scales up to the Kolmogorov scale k_η, exhibiting the characteristic $k^{-5/3}$ inertial range scaling. A coarse simulation without SGS modeling is truncated at the grid cutoff k_c and suffers from energy pile-up near the cutoff due to insufficient dissipation. In contrast, an LES incorporates a SGS model that dissipates energy near the cutoff, more closely matching the DNS spectrum within the resolved range. The black dashed line indicates the reference $k^{-5/3}$ slope.

capturing both the inertial range with its characteristic $k^{-5/3}$ slope and the dissipation range. A coarse simulation without any SGS model is truncated at the grid cutoff k_c and exhibits an unphysical energy pile-up near the cutoff due to insufficient dissipation. In contrast, LES includes a SGS model that removes energy near the cutoff scale, producing an energy spectrum that closely follows that of DNS over the resolved scales, though it tends to be

overly dissipative at the largest resolved wavenumbers. The reason for this will be discussed later in this section.

1.3.1 Filtering and the subgrid-scale stress tensor

Mathematically, the filtering operation can be expressed as

$$\bar{f}(\mathbf{x},t) = \int_{\Omega} G(\mathbf{x}-\mathbf{r})\, f(\mathbf{r},t)\, d\mathbf{r}, \tag{1.15}$$

where G is the filter kernel and $\bar{f}$ denotes the filtered (resolved) component of the field f [2, 6]. Applying this filter to the incompressible Navier-Stokes equations yields:

$$\frac{\partial \bar{\mathbf{u}}}{\partial t} + \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}^T) = -\nabla \bar{p} + \nu\, \Delta \bar{\mathbf{u}} - \nabla \cdot \boldsymbol{\tau}, \tag{1.16a}$$

$$\nabla \cdot \bar{\mathbf{u}} = 0, \tag{1.16b}$$

where the SGS stress tensor is defined as

$$\boldsymbol{\tau} = \overline{\mathbf{u}\mathbf{u}^T} - \bar{\mathbf{u}}\,\bar{\mathbf{u}}^T. \tag{1.17}$$

Physically, $\boldsymbol{\tau}$ represents the momentum flux contributed by the unresolved scales. In other words, it accounts for the interaction between resolved and unresolved scales, which is unknown when only the filtered velocity $\bar{\mathbf{u}}$ is evolved [6, 27].

1.3.2 Modeling the SGS stress

The SGS stress tensor $\boldsymbol{\tau}$ represents the effects of the motions that are not resolved. LES requires modeling this tensor in terms of the resolved (filtered) velocity field. In general, two broad classes of SGS models can be distinguished: functional models and structural models [6, 29].

Functional models, also known as eddy-viscosity models, aim primarily to reproduce the correct dissipative behavior of the unresolved scales. They introduce an effective turbulent viscosity ν_t that enhances dissipation and ensures numerical stability. A prototypical example is the Smagorinsky model [26], which assumes that the deviatoric part of the SGS stress tensor is proportional to the local resolved strain rate:

$$\boldsymbol{\tau}^{\mathrm{SGS}} - \frac{1}{3}\,\mathrm{tr}(\boldsymbol{\tau})\,\mathbf{I} = -2\nu_t\,\bar{\mathbf{S}}, \qquad \nu_t = (C_s\Delta)^2\,|\bar{\mathbf{S}}|,$$

where the resolved strain-rate tensor is defined as

$$\bar{\mathbf{S}} = \frac{1}{2}\left(\nabla\bar{\mathbf{u}} + (\nabla\bar{\mathbf{u}})^T\right)$$

and its magnitude is given by

$$|\bar{\mathbf{S}}| = \sqrt{2\,\mathrm{tr}(\bar{\mathbf{S}}^T\bar{\mathbf{S}})}.$$

The Smagorinsky model and its many variants guarantee a forward transfer of energy from the resolved to the subgrid scales, maintaining stability and yielding physically realistic energy spectra (see Chapters 2 and 3). However, because the eddy-viscosity assumption enforces purely dissipative behavior, it cannot capture local backscatter (energy transfer from unresolved to resolved scales) or the true tensorial structure of τ_{ij} [30, 31]. Therefore, such SGS models are typically too dissipative. This becomes especially apparent at large wavenumbers (see Figure 1.2).

Dynamic models are more flexible, as such models determine the Smagorinsky coefficient C_s dynamically from the resolved field using a test-filtering procedure [10, 32]. This allows ν_t to vary in space and time, enhancing accuracy in complex or inhomogeneous flows. This model is used as a benchmark in Chapter 3.

In contrast, structural models, such as scale-similarity and gradient models, seek to reconstruct the SGS stress tensor more directly from the resolved velocity field, often by assuming that the small-scale structures resemble the larger, resolved ones [27, 33]. For instance, scale-similarity models approximate $\boldsymbol{\tau}$ by filtering products of the resolved velocity components at multiple scales. These models can accurately reproduce the local structure of $\boldsymbol{\tau}$, and they naturally allow for backscatter. However, because they do not enforce a net dissipative effect, structural models are often prone to numerical instability.

To combine the advantages of both approaches, mixed models introduce a superposition of a dissipative eddy-viscosity term and a structural, scale-similarity term [33, 34]. This hybrid formulation retains the stability and spectral behavior of functional models while improving the physical realism and correlation with the true SGS stresses.

In summary, functional (eddy-viscosity) models are robust, stable, and capable of reproducing the correct energy cascade, but they provide only an approximate representation of the true SGS tensor. Structural models, on the other hand, better reflect the actual dynamics of the subgrid stresses, but suffer from inherent instabilities. Mixed approaches aim to balance these complementary properties, representing the current trend in advanced LES modeling.

Throughout this thesis, functional SGS models will serve as the primary benchmark, selected for their robust numerical stability and their broad use in practical LES applications [2, 6, 32]. From this point onward, SGS models will often be referred to as closure models.

1.4 Machine learning approaches

In the last decade, machine learning has rapidly become a highly active field of research with many applications, such as image recognition, text-to-speech, self-driving cars, large language models etc [35]. Also in the field of scientific computing the application of machine learning algorithms has seen an increase in use, such as for predicting inter-atomic forces in molecular dynamics simulations [36], approximating solutions to PDEs in the form of physics informed neural networks (PINNs) [37], and closure models for LES [14, 16, 38–40]. In this thesis, we concern ourselves with the latter. For this purpose, we shortly introduce the basic building block of many machine learning algorithms, namely the neural network [35].

1.4.1 Neural networks

Neural networks are a class of machine learning models designed to recognize patterns and approximate complex functions [35]. They are loosely inspired by the structure of biological brains, where networks of neurons process and transmit signals. In artificial neural networks, the fundamental building block is the artificial neuron, which takes in numerical inputs, applies a transformation, and produces an output.

At their core, neural networks consist of layers:

- **Input layer**, which receives raw data (e.g., pixels of an image, words in a sentence).
- **Hidden layers**, which transform the input into higher-level features through successive computations.
- **Output layer**, which produces the final prediction (e.g., class label, probability, or numerical value).

Each connection between neurons carries a weight, and each neuron has a bias. During training, the network adjusts these weights and biases to minimize the difference between its predictions and the true outcomes, typically using an algorithm called backpropagation. This algorithm determines the gradient based on the error metric, referred to as the loss function. The optimization of these weights is often done using the Adam optimization algorithm, which is based on gradient descent [41].

For an input vector $\mathbf{x} \in \mathbb{R}$, a single-layer neural network with weight matrix $\mathbf{W}$, bias vector $\mathbf{b}$, and nonlinear activation function $\sigma(\cdot)$ computes:

$$\mathbf{h}(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1.18}$$

For deep neural networks, this computation is applied repeatedly across multiple layers:

$$\mathbf{f}(\mathbf{x}) = \sigma^{(L)}\Big(\mathbf{W}^{(L)}\big(\dots\sigma^{(1)}(\mathbf{W}^{(1)}\mathbf{x}+\mathbf{b}^{(1)})\dots\big)+\mathbf{b}^{(L)}\Big) \tag{1.19}$$

where L is the number of layers. This recursive structure, in combination with the nonlinear activation functions [42], allows neural networks to approximate highly nonlinear functions. In fact, the universal approximation theorem states that, given enough neurons, a neural network can approximate any continuous function on a compact domain [35, 43]. The dimension of the weight matrices $\mathbf{W}$ is determined by the width of the neural network and L corresponds to the depth. A schematic representation of a neural network is depicted in Figure 1.3.

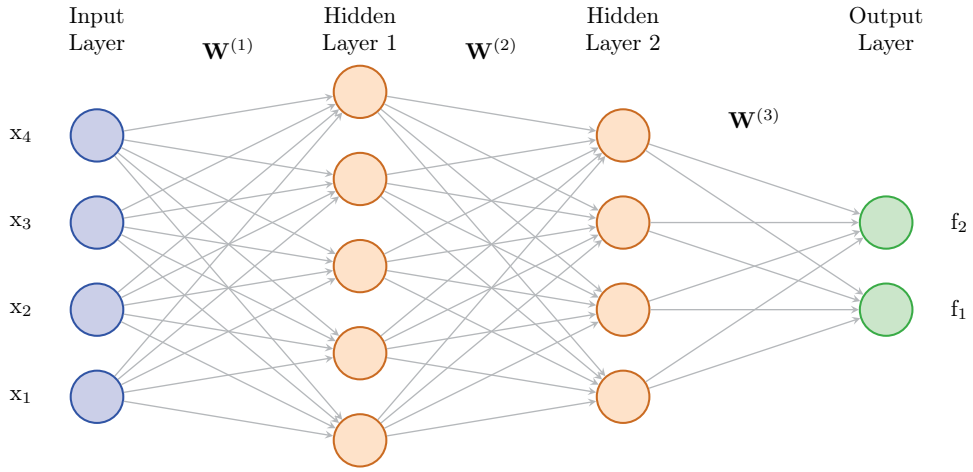


Figure 1.3: Structure of a neural network. The input layer receives data, the hidden layers transform it into higher-level features, and the output layer produces predictions. Connections represent weighted links between neurons.

1.4.2 Convolutional neural networks

While fully connected neural networks are powerful, they become inefficient when dealing with high-dimensional data such as images, where each pixel would be treated as a separate input. Convolutional neural networks (CNNs) address this problem by exploiting the spatial structure of the data [35, 44].

Instead of connecting every input neuron to every hidden neuron, CNNs use convolutional layers, which apply small filters (also called kernels) that slide across the input. Each filter detects specific local patterns, such as edges or textures in an image, and produces a feature map that highlights where those patterns occur. Multiple kernels allow the network to

capture a variety of features.

For an input image $\mathbf{X} \in \mathbb{R}^{m \times n}$ and a kernel $\mathbf{K} \in \mathbb{R}^{p \times q}$, the convolution operation producing feature map S given by:

$$\mathbf{S}_{i,j} = \sum_{u=1}^{p} \sum_{v=1}^{q} \mathbf{K}_{u,v} \, \mathbf{X}_{i+u-1,j+v-1}. \tag{1.20}$$

A schematic representation of this is depicted in Figure 1.4. In practice, nonlinear activation

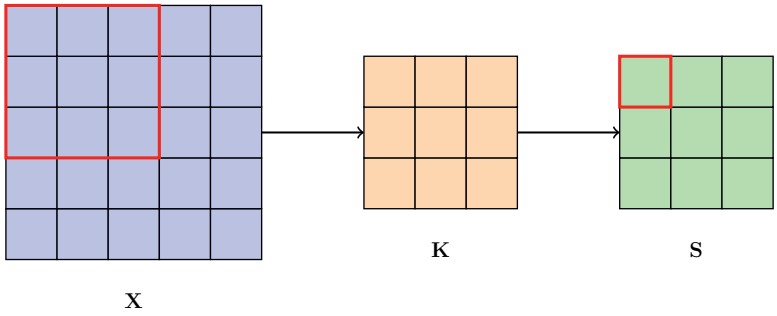


Figure 1.4: Convolution operation: a kernel **K** slides over the input **X**, producing a feature map **S**. The red box shows the one patch currently being convolved.

functions $\sigma(\cdot)$ are applied after the convolution, and multiple layers of convolutions allow CNNs to build hierarchical feature representations: from low-level edges to high-level concepts.

Translation invariance

A key strength of CNNs lies in their ability to achieve translation invariance. Because the same kernel (with shared weights) is applied across all positions of the input, the network can recognize a pattern regardless of its location. For example, a kernel trained to detect an edge will respond similarly whether that edge appears in the top-left or bottom-right of an image.

1.5 Research topics

A wide number of different approaches for combining physics-driven and data-driven methods exist for turbulent flow simulation. Three prominent and promising approaches are (I) data-driven large eddy simulation (LES), (II) evolve-filter-relax (EFR), and (III) reduced-order models (ROMs). In this section, we introduce the topics, and mention the state-of-the-art and research gaps for each of the approaches. In Section 1.6 we will formulate the corresponding research questions that we will aim to answer in this thesis.

1.5.1 Topic I: Machine learning-based closure models for large eddy simulation

As stated earlier, one of the central aims of this thesis is to investigate how machine learning algorithms can enhance LES. While traditional eddy-viscosity closures provide robustness, they are overly dissipative and suppress important physical mechanisms such as backscatter [2, 6, 26]. Machine-learned closure models have shown strong promise in accurately recovering subgrid stresses offline [14, 39, 45–51], but they frequently become unstable when deployed in actual simulations [14, 38, 39, 45, 48, 51]. Attempts at stabilization, such as clipping toward strictly dissipative behavior [14, 38, 51], noise augmentation [12], stochastic regularization [50], or a posteriori training strategies [16, 39, 46, 52–56], improve performance but still do not guarantee stability. Consequently, long-time LES with data-driven closures remains unreliable, particularly when statistical quantities of interest are considered [51].

To illustrate this more clearly, we present a brief example of simulation results from Chapter 3. In that chapter, a structure-preserving neural network architecture is proposed, specifically designed for closure modeling of the incompressible Navier-Stokes equations [13]. The results are shown in Figure 1.5, where the reference DNS was computed on a 2048×2048 grid, while the LES simulations were carried out on a 64×64 grid. The comparison highlights the qualitative differences between several closure approaches (for additional details on the experimental setup, we refer the reader to Chapter 3). From the figure, it is clear that omitting a closure model leads to a noisy and inaccurate simulation. In contrast, as discussed in Section 1.3, the Smagorinsky model introduces excessive dissipation, leading to an over-smoothed solution. A CNN-based closure model produces improved results but begins to exhibit numerical noise in the upper-left region, which eventually destabilizes the simulation. This motivates the work presented in Chapters 2 and 3. The structure-preserving architecture introduced in Chapter 3 produces a simulation that closely resembles the filtered DNS, illustrating the advantage of embedding physical structure directly into the neural network architecture.

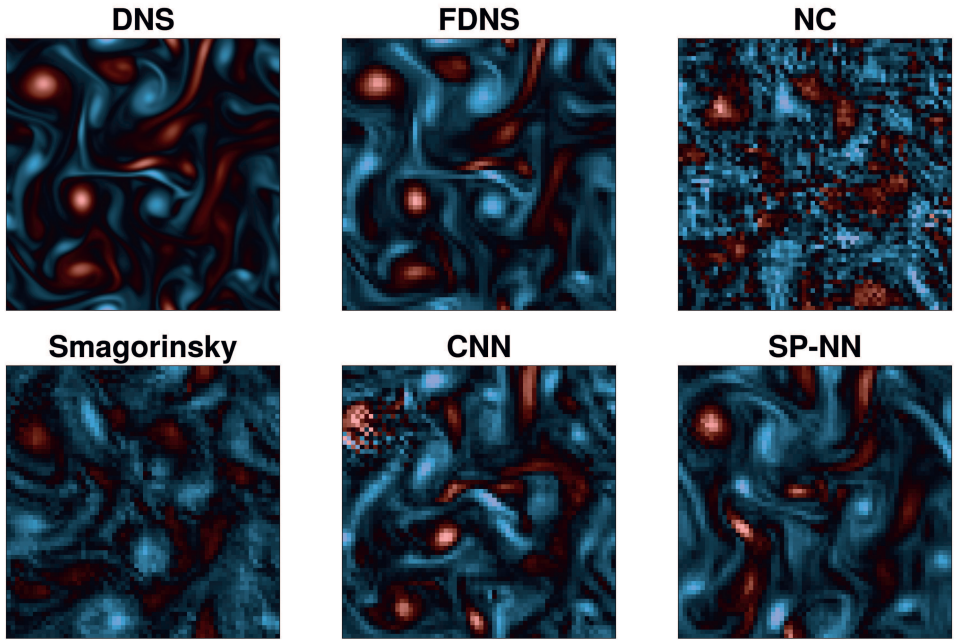


Figure 1.5: Resolved vorticity fields $\nabla \times \bar{\mathbf{u}}$ produced from simulations carried out using different closure models. The plots show results from: the direct numerical simulation (DNS) on a 2048×2048 grid, the corresponding filtered direct numerical simulation (FDNS) on the LES grid (64×64), no closure (NC), a Smagorinsky model, a convolutional neural network (CNN), and a structure-preserving neural network (SP-NN). Detailed simulation conditions and the introduction of the SP-NN can be found in Chapter 3.

1.5.2 Topic II: Data-driven extensions of the evolve-filter-relax framework

Next, we consider an alternative approach to LES, namely the evolve-filter-relax (EFR) framework. This framework is closely related to LES in both its objective and mathematical interpretation [17, 57]. Similarly to LES, EFR controls the effects of unresolved scales in under-resolved flows by introducing additional dissipation while preserving the dominant flow dynamics. This connection has been formalized in several works, where EFR is interpreted as a filtering-based LES methodology [17, 57]. The primary advantage of EFR lies in its modularity: the filtering and relaxation steps can be incorporated into existing solvers with minimal modifications, independent of the underlying spatial discretization. In this way, EFR provides a flexible alternative to classical LES closures.

The EFR framework consists of three steps performed at each time step. First, the governing equations are evolved on a coarse grid. Second, a filtering operation is applied

to damp numerical oscillations or unresolved flow features. In most formulations, this filter is defined implicitly through a differential equation involving a modified diffusion operator [58, 59]. Finally, a relaxation step is applied, which computes a convex combination of the evolved and filtered solutions. In Chapter 4, this procedure is covered in detail.

From an LES perspective, the differential filter used in EFR can be interpreted as introducing SGS dissipation. In particular, applying an elliptic filter has been shown to be equivalent to an eddy-viscosity model within a backward Euler time discretization [17]. Within the EFR framework, such models are often expressed through indicator functions that act as spatially varying weights, activating dissipation primarily in regions of strong gradients or insufficient resolution [60]. This localized action further reinforces the conceptual link between EFR and LES.

Despite its advantages, the classical EFR framework exhibits some limitations. Applying the differential filter requires solving a sparse linear system at every time step. This may become a computational bottleneck in large-scale simulations [61]. Moreover, the method is governed by only two parameters: the filter radius δ and the relaxation parameter χ [17, 57]. While various heuristic and physically motivated strategies for parameter selection exist [62, 63], they do not fully overcome the limited flexibility of the standard EFR formulation [64].

These limitations motivate the developments presented in this thesis, where we aim to enhance the flexibility and efficiency of the EFR framework through data-driven filtering strategies. These strategies retain their modular structure while alleviating their computational and parametric constraints. Both our data-driven strategy and a detailed introduction to EFR are presented in Chapter 4.

1.5.3 Topic III: Reduced-order modeling for advection-dominated flows

Besides coarsening the grid, as done in LES and EFR, another way of reducing the degrees of freedom of the system is by finding a representation in a reduced space. This is often done using simulation snapshots of a DNS. Finding the best linear approximation of the snapshot data is done using a proper orthogonal decomposition (POD). Non-linear machine learning-based approaches to reduce the dimension of the system also exist, e.g. autoencoders [65, 66]. However, in this thesis, we focus on the POD approach. In this approach, a snapshot matrix

$$\mathbf{X} := \begin{bmatrix} \mathbf{u}_h(t_1) & \dots & \mathbf{u}_h(t_s) \end{bmatrix} \quad \in \mathbb{R}^{N \times s} \tag{1.21}$$

is used, which contains s snapshots of the spatially discretized velocity $\mathbf{u}_h(t) \in \mathbb{R}^N$ field at different points in time. A singular value decomposition (SVD) of this matrix is carried out:

$$\mathbf{X} = \mathbf{\Phi}\mathbf{\Sigma}\mathbf{V}^T \tag{1.22}$$

which decomposes $\mathbf{X}$ into the product of orthonormal matrix $\mathbf{\Phi} \in \mathbb{R}^{N\times N}$, diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{N\times s}$, and orthonormal matrix $\mathbf{V} \in \mathbb{R}^{s\times s}$. We are mainly interested in the left-singular vectors contained in $\mathbf{\Phi}$ and the singular values σ_i contained on the diagonal of $\mathbf{\Sigma}$. We focus on the left-singular vectors because they form an orthonormal basis capturing the dominant modes of the data, with each singular value indicating the relative importance of its corresponding mode. By truncating the singular vectors, up to degree r, we obtain a basis of r basis vectors which best describe the snapshot matrix in an energy norm (see Chapter 5 for details). These vectors are typically referred to as POD modes. The decay of the singular values tells us how much of the information in the snapshot matrix is captured by truncating the basis at degree r. The reduced-order model (ROM) is obtained by projecting the discrete system of equations on the reduced basis [67, 68]. This will be discussed in Chapter 5.

In Figure 1.6 the POD modes obtained from DNS data of a simulation of Kolmogorov flow at $\mathrm{Re} = 500$ are shown. For the exact flow conditions, we refer to Chapter 3. In addition, the relative amount of information captured by truncating the POD basis at mode i for $\mathrm{Re} = 500$ and $\mathrm{Re} = 100$ is shown. If a significant amount of the flow's information can be captured in a small number of modes, one can significantly reduce the degrees of freedom of the system by projecting it onto the POD basis [67, 69]. To simulate the flow in terms of the reduced representation, a Galerkin projection of the semi-discrete system of equations onto the basis is performed. This will be discussed in Chapter 5.

However, looking at Figure 1.6, it becomes apparent that at higher Reynolds numbers more modes are needed to obtain a proper reconstruction of the flow field. This is related to the slow Kolmogorov N-width decay (a measure of how well the solution space of a PDE can be represented by a linear combination of N basis functions) of advection-dominated flows [7, 66, 70]. This makes simulating turbulent flows using ROMs computationally expensive, as many POD modes are required to capture these phenomena. Failure to include enough modes causes oscillatory and inaccurate solutions of the ROM [71, 72]. Several strategies exist to mitigate these issues, including time-local ROMs [73, 74], adaptive or Petrov–Galerkin formulations [75, 76], ROM closure modeling [77], nonlinear reduced manifolds [78], and machine learning-based ROMs [66, 79, 80]. While these methods improve robustness or expressiveness, they still face trade-offs between accuracy, stability, computational efficiency, and generalization capability outside the training data. In Chapter 5, we propose a space-local POD basis to address these issues.

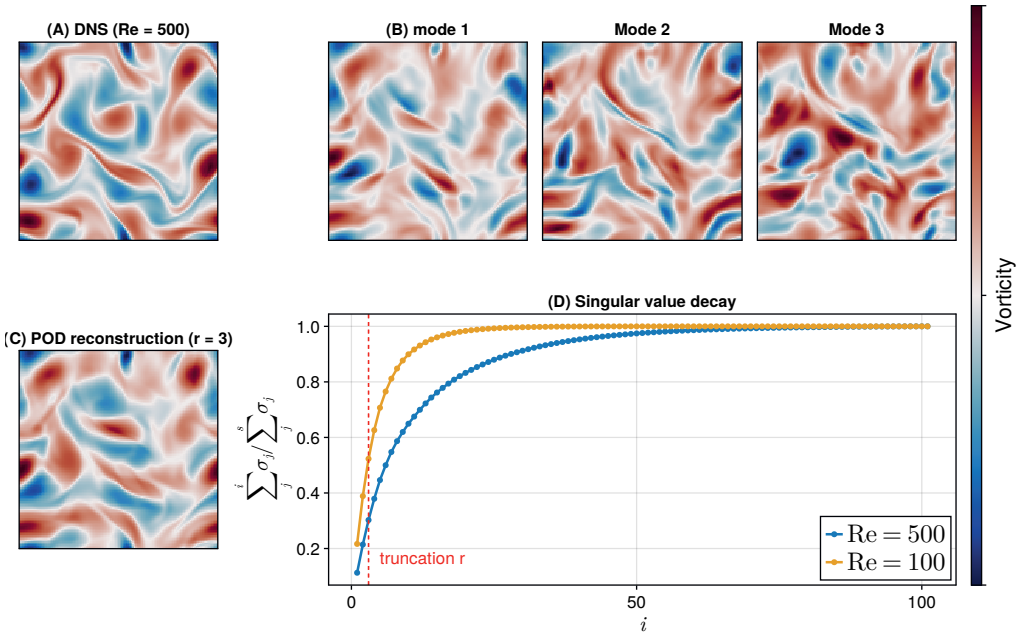


Figure 1.6: (A) Vorticity snapshot of a simulation of Kolmogorov flow at Re = 500, (B) first three POD modes for the vorticity field, (C) POD reconstruction, and (D) relative amount of information captured by truncating the POD basis at mode i for both Re = 500 and Re = 100. For the exact simulation conditions we refer the reader to Chapter 3.

1.6 Structure of the thesis

Based on the research gaps identified in Section 1.5 we formulate the following research question (RQ) which we aim to address in this thesis:

- **RQ0: How can physical structure be embedded into data-driven approaches for modeling fluid flows, and does doing so lead to more stable, accurate, and, generalizable models?**

What follows is an outline of the research topics addressed in the different chapters of the thesis, together with a set of research questions used to answer **RQ0**, along with the approaches used to answer them.

1.6.1 Topic I: Machine learning-based closure models for large eddy simulation (Chapters 2 & 3)

As discussed in Section 1.5.1, accurately modeling the effect of unresolved turbulent scales in LES is challenging. Traditional eddy-viscosity closures are robust but overly dissipative. Machine learning-based closure models show promise in representing subgrid stresses, but cause instabilities as they are unaware of the underlying physical laws.

The aim of this research topic is to understand how stability of machine learning-based closure models for LES can be enforced by construction, while retaining the ability to represent essential physical mechanisms. This aim is addressed through the following research questions:

- **RQ1: How can a stable and physically consistent machine learning-based closure model be formulated and how does its performance compare to existing approaches?** This question is addressed in Chapter 2. Here, we investigate the structure of 1D non-linear equations, namely Burgers' equation and the Korteweg-de Vries equation. We discretize these equations and apply a spatial averaging filter to reduce the degrees of freedom. The information that is removed by the filter is reintroduced into the system by so-called SGS variables. A novel neural network architecture that conserves the total energy of the system is introduced as a closure model, guaranteeing stability of the resulting dynamical system. Our neural network architecture is compared against existing (machine learning) approaches.

- **RQ2: Can machine learning-based closure models with stability guarantees, developed for simple 1D systems, be successfully extended to more complex fluid dynamics problems?** This question is addressed in Chapter 3, where the neural network architecture proposed in Chapter 2 is applied to the 2D incompressible Navier-Stokes equations. However, this time, without the inclusion of the SGS variables, as their extension to multiple spatial dimensions is still an open problem. Our approach is once again compared against existing (machine learning) approaches.

1.6.2 Topic II: Data-driven extensions of the evolve-filter-relax framework (Chapter 4)

As discussed in Section 1.5.2, the classical evolve-filter-relax (EFR) framework provides a stable and modular approach to simulating turbulent flows, but its practical use is limited by computational bottlenecks and restricted flexibility. In particular, the reliance on a differential filter and only two governing parameters limits efficiency and expressiveness in complex flows.

The aim of this research topic is to enhance the flexibility and computational efficiency of the EFR framework through data-driven extensions, while preserving its stability properties. This aim is addressed in Chapter 4 through the following research questions:

- **RQ3: Can introducing a data-driven linear filter into the EFR framework enhance both computational efficiency and accuracy?** To answer this question, the classical differential filter is replaced by a data-driven linear filter learned from DNS data. The filter parameters are determined through least-squares optimization, and the filter is applied efficiently using a fast Fourier transform (FFT). The approach is applied to the 2D incompressible Navier-Stokes equations, and its performance is compared to the standard EFR approach (using the differential filter) and the Smagorinsky LES closure model, regarding both accuracy and efficiency.
- **RQ4: How does one explicitly embed physical structure into the data-driven EFR framework, and how does this affect stability and performance?** This question is addressed by coupling the learned filtering strategy with an energy-conserving discretization and a relaxation mechanism designed to control energy and enstrophy. This retains the stability advantages of the original EFR formulation. Once again, this approach is compared to the standard EFR approach and the Smagorinsky LES closure model, for different amounts of training data.

1.6.3 Topic III: Reduced-order modeling for advection-dominated flows (Chapter 5)

As discussed in Section 1.5.3, classical POD-Galerkin ROMs often struggle with advection-dominated flows due to stability issues and the need for a large reduced bases. This motivates the development of alternative reduced-order modeling strategies that exploit spatial locality to improve stability, accuracy, and efficiency.

The aim of this research topic is to develop reduced-order modeling strategies for advection-dominated flows that are computationally efficient, stable, and accurate, while maintaining good generalization properties. This aim is in Chapter 5 through the following research questions:

- **RQ5: Can space-local reduced-order modeling strategies address the limited generalizability and high computational cost, and stability issues of classical space-global POD-Galerkin approaches while preserving accuracy?** To answer this equation, we propose a space-local POD approach in which localized basis functions are repeated across the domain, yielding sparse ROM operators and significantly reducing computational cost. We further enhance this framework through overlapping subdomains and demonstrate that the resulting ROMs inherit energy conservation

from the full system, guaranteeing stability. This framework is particularly suited to advection-dominated problems with spatially repetitive dynamics, such as traveling-wave phenomena. We examine its efficiency, accuracy, generalization capability, and stability on 1D and 2D advection test cases.

- **RQ6: To what extent are the proposed space-local reduced-order models applicable to multi-dimensional fluid flows?** This question is addressed by investigating the ability of the space-local POD basis to represent vorticity fields stemming from numerical solutions of the 2D incompressible Navier-Stokes equations. This is evaluated on both the training data and unseen vorticity snapshots to assess the generalization of the space-local POD basis. The performance is compared against the standard space-global POD basis.

1.7 List of publications

The chapters in this thesis are based on the following publications:

- **Chapter 2**

  T van Gastelen, W Edeling, B Sanderse. "Energy-conserving neural network for turbulence closure modeling". Journal of Computational Physics 508 (2024): 113003.

  Contributions of the author of this thesis: Conceptualization, Methodology, Software, Writing

- **Chapter 3**

  T van Gastelen, W Edeling, B Sanderse. "Energy-Conserving Neural Network Closure Model for Long-Time Accurate and Stable 2D LES". Computers & Fluids 311 (2026): 107028.

  Contributions of the author of this thesis: Conceptualization, Methodology, Software, Writing

- **Chapter 4**

  A Ivagnes[1], **T van Gastelen**[1], S D Agdestein, B Sanderse, G Stabile, G Rozza. "A New Data-Driven Energy-Stable Evolve-Filter-Relax Model for Turbulent Flow Simulation". Computer Methods in Applied Mechanics and Engineering 250 (2026): 118654.

  Contributions of the author of this thesis: Conceptualization, Methodology, Software, Writing

- **Chapter 5**

  T van Gastelen, W Edeling, B Sanderse. "Modeling Advection-Dominated Flows with Space-Local Reduced-Order Models". Computer & Fluids 305 (2026): 106911.

  Contributions of the author of this thesis: Conceptualization, Methodology, Software, Writing

[1]The first two authors, Anna Ivagnes and Toby van Gastelen, contributed equally to this publication.

2

Energy-Conserving Neural Network for Turbulence Closure Modeling

In turbulence modeling, we are concerned with finding closure models that represent the effect of the subgrid scales on the resolved scales. Recent approaches gravitate towards machine learning techniques to construct such models. However, the stability of machine-learned closure models and their abidance by physical structure (e.g. symmetries, conservation laws) are still open problems. To tackle both issues, we take the 'discretize first, filter next' approach. In this approach, we apply a spatial averaging filter to existing fine-grid discretizations. The main novelty is that we introduce an additional set of equations that dynamically model the energy of the subgrid scales. Having an estimate of the energy of the subgrid scales, we can use the concept of energy conservation to derive stability. The subgrid energy containing variables are determined via a data-driven technique. The closure model is used to model the interaction between the filtered quantities and the subgrid energy. Therefore, the total energy should be conserved. Abiding by this conservation law yields guaranteed stability of the system. In this work, we propose a novel skew-symmetric convolutional neural network architecture that satisfies this law. The result is that stability is guaranteed, independent of the weights and biases of the network. Importantly, as our framework allows for energy exchange

This chapter is based on [15].

between resolved and subgrid scales it can model backscatter. To model dissipative systems (e.g. viscous flows), the framework is extended with a diffusive component. The introduced neural network architecture is constructed such that it also satisfies momentum conservation. We apply the new methodology to both the viscous Burgers' equation and the Korteweg-De Vries equation in 1D. The novel architecture displays superior stability properties when compared to a vanilla convolutional neural network.

2.1 Introduction

The direct numerical simulation (DNS) of turbulent flows is often infeasible due to the high computational requirements. Especially for applications in design and uncertainty quantification, this rapidly becomes computationally infeasible, as typically many simulations are required [81, 82]. To tackle this issue, several different approaches have been proposed, such as reduced-order models (ROMs) [67], Reynolds-averaged Navier-Stokes (RANS) [9], and large eddy simulation (LES) [6]. These approaches differ in how much of the physics is modeled. Here we will focus on the LES approach.

In LES, the large-scale physics is modeled directly by a coarse-grid discretization. The coarse grid is accounted for by applying a filter to the original equations. However, as the filter does not commute with the nonlinear terms in the equations, a commutator error arises. This prevents one from obtaining an accurate solution without knowledge of the subgrid-scale (SGS) content. This commutator error is referred to as the closure term. Modeling this term is the main concern of the LES community. A major difficulty in this process is dealing with energy moving from the small scales to the large scales (backscatter) [83, 84]. This is because the SGS energy is unknown during the time of the simulation. This makes accounting for backscatter without leading to numerical instabilities difficult [85]. Classical physics-based closure models are therefore often represented by a dissipative model, e.g. of eddy-viscosity type [26]. This ensures a net decrease in energy. Another option is that the closure model is clipped such that backscatter is removed [86]. Even though the assumption of a global net decrease in energy is valid [26], explicit modeling of backscatter is still important. This is because locally the effect of backscatter can be of great significance [30, 31]. Closure models that explicitly model the global SGS energy at a given point in time, to allow for backscatter without sacrificing stability, also exist [87]. Recently, machine learning approaches, or more specifically neural networks (NNs), have come forward as viable closure models. They have been shown to outperform the classical approaches for different use cases [14, 39, 49, 51, 52]. However, stability remains an important issue, along with abidance by physical structure such as mass, momentum, and energy conservation [12, 14, 38, 45].

In [38] homogeneous isotropic turbulence for the compressible Navier-Stokes equations was treated. A convolutional neural network (CNN) was trained to reproduce the closure

term from high-resolution flow data. Although *a priori* cross-correlation analysis on the training data showed promising results, stable models could only be achieved by projecting onto an eddy-viscosity basis. In [45], a gated recurrent NN was applied to the same test case. This network displayed even higher cross-correlation with the closure term, but still yielded unstable models. Even after training on data with added artificial noise, the model remained unstable [12]. In [14], incompressible turbulent channel flow was treated. Here, NNs with varying levels of locality were used to construct a closure model. They showed that increasing the view of the NN improves *a priori* performance. However, *a posteriori* analysis showed that this increased input space also led to instabilities. Even after introducing backscatter clipping, these larger models were still outperformed by the highly localized NN models. Two promising approaches to improving the stability of NN closure models are 'trajectory fitting' [39, 52–55] and reinforcement learning [46, 56]. Both of these approaches have in common that instead of fitting the NN to the exact closure term (which is what we will refer to as 'derivative fitting'), one optimizes directly with respect to how well the solution is reproduced. This has been shown to yield more accurate and stable closure models [39, 52, 55]. The difference between these two methods is that trajectory fitting uses exact gradients, such that gradient-based optimizers can be applied to optimize the NN weights [41]. Reinforcement learning does not require these gradients. This makes it suitable for non-differentiable processes such as chess and self-driving cars [88].

These approaches have all been applied to the Navier-Stokes equations. However, in this chapter, we consider a 1D simplification, namely Burgers' equation. Several studies have been carried out which apply machine learning to this equation. In [89] physics informed neural networks (PINNs) were successfully applied to Burgers' equation. In PINNs, the partial differential equation (PDE) is encoded into the loss function of the neural network. The advantage of PINNs is that they allow us to approximate the solution without explicitly discretizing space and time. In the context of closure modeling, several studies have been carried out. For example, in [90] and [91], a neural network was fitted to predict the closure term for Burgers' equation with forcing. In [91], the trained neural network was successfully applied to unseen viscosity values. This was achieved by limited retraining on new data. This technique is known as transfer learning. Furthermore, several studies have been carried out which show the benefits of trajectory fitting [55, 89, 92]. In [89], trajectory fitting is combined with a Fourier neural operator to predict a spatially dependent Smagorinsky coefficient. This combination outperformed the other considered approaches. It also guarantees stability by being strictly dissipative. Furthermore, Fourier neural operators have the advantage that they are grid independent [93].

However, none of the discussed approaches leads to a provably stable NN closure model, while still allowing for backscatter. In addition, they do not guarantee abidance by the underlying energy conservation law. The latter is something that, to our knowledge, does

not yet exist in the case of LES closure models. To resolve these shortcomings, we present *a new NN closure model that satisfies both momentum and kinetic energy conservation and is therefore stable by design.* As stated earlier, the difficulty of this task mainly lies in the fact that: (i) The kinetic energy conservation law includes terms which depend on the SGS content, which is too expensive to simulate directly. (ii) Consequently, the kinetic energy of the large scales is not a conserved quantity (in the limit of vanishing viscosity). To tackle these issues, we take the 'discretize first, filter next' approach [54, 55]. This means that we start from a high-resolution discretization with N degrees of freedom, to which we apply a discrete filter. This filter projects the solution onto a coarse computational grid of dimension I, with $I \ll N$. Given the discrete filter, the exact closure term can be obtained by computing the commutator error. The main advantage of this approach is that the closure term now also accounts for the discretization error. Based on the filter's properties, we then derive an energy conservation law that can be split into two components: one that depends solely on the resolved scales (resolved energy) and another that solely depends on the SGS content (SGS energy) [87]. Like in existing works, the closure model is represented by a CNN [44]. The main novelty comes from the addition of a set of SGS variables. These SGS variables represent the SGS energy, projected onto the coarse grid. The key insight is that the resulting system of equations should still conserve energy in the inviscid limit. We then choose our CNN architecture such that it is consistent with this limit. In this way, we still allow for backscatter without sacrificing stability.

This chapter is structured in the following way: In Section 2.2 we discuss Burgers' and Korteweg-de Vries equations and their energy and momentum conservation properties. We introduce the discrete filter, the resulting closure problem, and derive a new energy conservation law. This law describes the exchange between the resolved and the SGS energy. In Section 2.3, we introduce our novel machine learning framework for modeling the closure term. This approach satisfies the derived energy conservation law using the set of SGS variables to represent the SGS energy. In addition, we show how to satisfy momentum conservation. In Section 2.4, we study the convergence and stability of our closure modeling framework and compare this to a vanilla CNN. We also analyze its structure-preserving properties in terms of momentum and energy conservation and its ability to extrapolate in space and time. In Section 2.5 we present a short discussion on the applicability of our framework to the Navier-Stokes equations. In Section 2.6, we conclude our work.

2.2 Governing equations, discrete filtering, and closure problem

Before constructing a machine learning closure, we formulate a description of the closure problem on the discrete level. For this purpose we introduce the filter and reconstruction operator which we apply to the discrete solution. In this way, we also account for the discretization error. In addition, we discuss the effects of filtering on the physical structure of the system.

2.2.1 Spatial discretization

We consider an initial value problem of the following form:

$$\frac{\partial u}{\partial t} = f(u), \tag{2.1}$$

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \tag{2.2}$$

which describes the evolution of some quantity $u(\mathbf{x}, t)$ in space $\mathbf{x} \in \Omega$ and time t on the spatial domain $\Omega \subseteq \mathbb{R}^d$, given initial state u_0. The dynamics of the system is governed by the right-hand side (RHS) $f(u)$, which typically involves partial derivatives of u with respect to $\mathbf{x}$. After spatial discretization (method of lines), we obtain the vector $\mathbf{u}(t) \in \mathbb{R}^N$. The elements u_i of this vector approximate the value of u at each of the N grid points $\mathbf{x}_i \in \Omega$ for $i = 1, \ldots, N$, such that $\mathrm{u}_i \approx u(\mathbf{x}_i)$. The discrete analogue of the IVP is then

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = f_h(\mathbf{u}), \tag{2.3}$$

$$\mathbf{u}(0) = \mathbf{u}_0, \tag{2.4}$$

where f_h represents a spatial discretization of f. It is assumed that all the physics described by equation (2.1) is captured in the discrete solution $\mathbf{u}$. This means that whenever the physics involves a wide range of spatial scales, a very large number of degrees of freedom N is needed to adequately resolve all the scales. This results in a large amount of computational resources required to solve these equations. This is what we aim to alleviate.

2.2.2 Burgers' and Korteweg-de Vries equation, and physical structure

We are interested in the modeling and simulation of turbulent flows. For this purpose, we first consider Burgers' equation, a 1D simplification of the Navier-Stokes equations. Burgers'

equation describes the evolution of the velocity $u(x,t)$ according to the PDE

$$\frac{\partial u}{\partial t} = -\frac{1}{2}\frac{\partial u^2}{\partial x} + \frac{\partial}{\partial x}\left(\nu \frac{\partial u}{\partial x}\right). \tag{2.5}$$

The first term on the RHS represents nonlinear convection and the second term diffusion, weighted by the positive viscosity field $\nu(x) \geq 0$. This equation expresses similar behavior to 3D turbulence in the fact that smaller scales are created by the nonlinear convective term, which then dissipate through diffusion [94]. We will be interested in two properties of the Burgers' equation, which we collectively call 'structure'.

Firstly, momentum P is conserved on periodic domains:

$$\frac{\mathrm{d}P}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\underbrace{\int_\Omega u \mathrm{d}\Omega}_{=:P} = \int_\Omega -\frac{1}{2}\frac{\partial u^2}{\partial x} + \frac{\partial}{\partial x}\left(\nu\frac{\partial u}{\partial x}\right)\mathrm{d}\Omega = 0. \tag{2.6}$$

Secondly, on periodic domains (kinetic) energy E is conserved in the absence of viscosity:

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\underbrace{\frac{1}{2}\int_\Omega u^2 \mathrm{d}\Omega}_{=:E} = \int_\Omega -\frac{u}{2}\frac{\partial u^2}{\partial x} + u\frac{\partial}{\partial x}\left(\nu\frac{\partial u}{\partial x}\right)\mathrm{d}\Omega = -\int_\Omega \nu\left(\frac{\partial u}{\partial x}\right)^2 \mathrm{d}\Omega \leq 0, \tag{2.7}$$

where we used integration by parts. As we solely deal with viscous flows, i.e. $\nu(x) > 0$, we disregard energy loss due to the presence of shocks [95]. Note that these conservation laws only hold in the absence of forcing.

These properties can be preserved in a discrete setting by employing a structure-preserving scheme [95]. On a uniform grid, the convective term is discretized with the following skew-symmetric scheme:

$$(\mathbf{C}(\mathbf{u})\mathbf{u})_i = -\frac{1}{6h}(\mathrm{u}_{i+1}^2 - \mathrm{u}_{i-1}^2) - \frac{1}{6h}\mathrm{u}_i(\mathrm{u}_{i+1} - \mathrm{u}_{i-1}), \tag{2.8}$$

where h is the grid spacing. The skew-symmetry entails that $\mathbf{u}^T\mathbf{C}(\mathbf{u})\mathbf{u} = 0$. This is used later to derive energy conservation. Furthermore, the diffusion operator is discretized as

$$(-\mathbf{Q}^T \mathrm{diag}(\boldsymbol{\nu})\mathbf{Q}\mathbf{u})_i = \frac{1}{h^2}(\nu_i(\mathrm{u}_{i+1} - \mathrm{u}_i) + \nu_{i-1}(\mathrm{u}_{i-1} - \mathrm{u}_i)), \tag{2.9}$$

where $\mathbf{Q}$ is a simple forward difference approximation of the first derivative and $\nu_i = \nu(x_i)$ [24, 67]. In this chapter, we deal with constant viscosity $\nu(x) = \nu$ such that we obtain the following system of ordinary differential equations (ODEs):

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{C}(\mathbf{u})\mathbf{u} + \nu\mathbf{D}\mathbf{u}. \tag{2.10}$$

Here $\mathbf{D} = -\mathbf{Q}^T\mathbf{Q}$ corresponds to a simple central difference approximation of the second derivative. For the time discretization, we employ an explicit Runge-Kutta 4 (RK4) scheme [96].

This discretization conserves the discrete momentum $P_h = h\mathbf{1}^T\mathbf{u}$ in the periodic case:

$$\frac{\mathrm{d}P_h}{\mathrm{d}t} = h\mathbf{1}^T\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = 0, \tag{2.11}$$

where $\mathbf{1}$ is a column vector with all entries equal to one. Note that this equation discretely represents the integral in (2.6). Furthermore, due to the skew-symmetry of the convection operator, the evolution of the discrete kinetic energy $E_h = \frac{h}{2}\mathbf{u}^T\mathbf{u}$ is given by:

$$\text{Burgers' equation:} \qquad \frac{\mathrm{d}E_h}{\mathrm{d}t} = h\mathbf{u}^T\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = h\nu\mathbf{u}^T\mathbf{D}\mathbf{u} = -h\nu||\mathbf{Q}\mathbf{u}||_2^2 \leq 0. \tag{2.12}$$

This is the discrete equivalent of (2.7). In both the continuous and discrete formulation, we used the product rule to obtain the derivative. The norm $\|.\|_2$ represents the conventional 2-norm. From (2.12) we conclude that this discretization ensures net kinetic energy dissipation, and conservation in the inviscid limit. From this point forward, we will refer to the kinetic energy simply as energy.

In addition to Burgers' equation, we will consider the Korteweg de-Vries (KdV) equation:

$$\frac{\partial u}{\partial t} = -\frac{\varepsilon}{2}\frac{\partial u^2}{\partial x} - \mu\frac{\partial^3 u}{\partial x^3}, \tag{2.13}$$

where ε and μ are scalar parameters. The KdV equation conserves momentum and energy irrespective of the values of ε and μ. We discretize the nonlinear term in the same way as for Burgers' equation, using the skew-symmetric scheme. The third-order spatial derivative is approximated by a skew-symmetric central difference stencil: $(-\mathrm{u}_{i-2} + 2\mathrm{u}_{i-1} - 2\mathrm{u}_{i+1} + \mathrm{u}_{i+2})/(2h^3)$, see [97]. The resulting discretization is not only momentum conserving, but also energy conserving:

$$\text{KdV equation:} \qquad \frac{\mathrm{d}E_h}{\mathrm{d}t} = 0. \tag{2.14}$$

2.2.3 Discrete filtering

To alleviate the high computational expenses for large N we apply a spatial averaging filter to the fine-grid solution $\mathbf{u}$. This results in the coarse-grid approximation $\bar{\mathbf{u}} \in \mathbb{R}^I$. The coarse grid follows from dividing Ω into I non-overlapping cells Ω_i with cell centers $\mathbf{X}_i$. The coarse grid is refined into the fine grid by splitting each Ω_i into J subcells ω_{ij} with cell centers $\mathbf{x}_{ij}$. The subdivision is intuitively pictured in Figure 2.1, for a uniform 1D grid. Furthermore, we define the mass matrices $\boldsymbol{\omega} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{\Omega} \in \mathbb{R}^{I\times I}$ which contain the volumes of the fine

and coarse cells on the main diagonal, respectively.

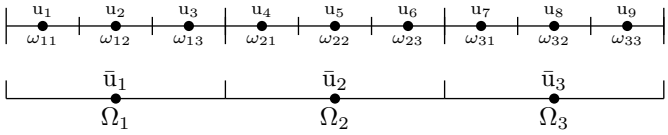


Figure 2.1: Subdivision of the spatial grid. The dots represent cell centers $\mathbf{x}_{ij}$ and $\mathbf{X}_i$ for $N = 9$ and $I = J = 3$.

To reduce the degrees of freedom of the system, we apply a spatial averaging filter to $\mathbf{u}$. This filter simply computes $\bar{\mathbf{u}}$ as a weighted average of $\mathbf{u}$, within each coarse cell. We represent this by the following matrix vector product:

$$\bar{\mathbf{u}} = \mathbf{W}\mathbf{u}, \tag{2.15}$$

where $\mathbf{W} \in \mathbb{R}^{I \times N}$ is the filter. The filter is defined as

$$\mathbf{W} = \boldsymbol{\Omega}^{-1}\mathbf{O}\boldsymbol{\omega} \tag{2.16}$$

with overlap matrix $\mathbf{O} \in \mathbb{R}^{I \times N}$:

$$\mathbf{O} := \begin{bmatrix} 1 & \dots & 1 & & & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & & & 1 & \dots & 1 \end{bmatrix}. \tag{2.17}$$

This matrix contains ones at index (i, j) if ω_{ij} lies in Ω_i. Mathematically, $\bar{\mathbf{u}}$ can be regarded as a representation of $\mathbf{u}$ in a reduced basis. To project back onto the original basis we use a reconstruction operator $\mathbf{R}$:

$$\mathbf{R} := \mathbf{O}^T \tag{2.18}$$

which is a right inverse of $\mathbf{W}$, i.e.

$$\mathbf{W}\mathbf{R} = \mathbf{I}. \tag{2.19}$$

The matrix $\mathbf{R}$ approximates the reconstruction by a piece-wise constant function [22]. This is intuitively pictured in Figure 2.2. Filtering a reconstructed solution $\mathbf{R}\bar{\mathbf{u}}$ leaves $\bar{\mathbf{u}}$ unchanged, i.e.

$$\bar{\mathbf{u}} = \underbrace{(\mathbf{W}\mathbf{R})^p}_{=\mathbf{I}}\mathbf{W}\mathbf{u} \tag{2.20}$$

for $p \in \mathbb{N}_0$. We will refer to this property as the 'projection' property, as it is similar to how repeated application of a projection operator leaves a vector unchanged.

By subtracting $\mathbf{R}\bar{\mathbf{u}}$ from $\mathbf{u}$ we obtain the SGS content $\mathbf{u}' \in \mathbb{R}^N$:

$$\mathbf{u}' := \mathbf{u} - \mathbf{R}\bar{\mathbf{u}}. \tag{2.21}$$

In theory, one could define a more accurate $\mathbf{R}$, e.g. through polynomial reconstruction or data-driven approaches [54], and obtain a smaller $\mathbf{u}'$. However, this particular choice of $\mathbf{R}$ is made such that the energy is invariant under reconstruction, as will be shown in equation (2.27). This is an important property for our methodology to work. Furthermore, we will refer to the SGS content in a single coarse cell Ω_i as $\boldsymbol{\mu}_i \in \mathbb{R}^J$, see Figure 2.2. Applying the filter to $\mathbf{u}'$ yields zero:

$$\mathbf{W}\mathbf{u}' = \mathbf{W}\mathbf{u} - \underbrace{\mathbf{W}\mathbf{R}}_{=\mathbf{I}}\bar{\mathbf{u}} = \bar{\mathbf{u}} - \bar{\mathbf{u}} = \mathbf{0}_\Omega, \tag{2.22}$$

where $\mathbf{0}_\Omega$ is a vector with all entries equal to zero, defined on the coarse grid. This can be seen as the discrete equivalent of a property of a Reynolds operator [6]. To illustrate, we display each of the introduced quantities for a 1D sinusoidal wave in Figure 2.2.

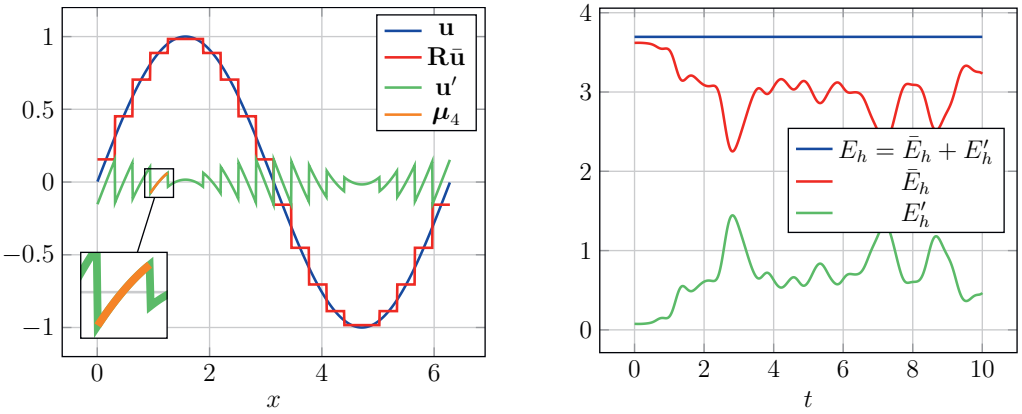


Figure 2.2: (Left) Fine-grid $\mathbf{u}$, reconstructed $\mathbf{R}\bar{\mathbf{u}}$, and SGS content $\mathbf{u}'$ for $u = \sin(x)$. Here $N = 1000$, $I = 20$, and $J = 50$. The SGS content in the fourth coarse cell $\boldsymbol{\mu}_4$ is also indicated. (Right) Energy during a simulation of KdV equation with periodic boundary conditions (BCs) before and after filtering.

2.2.4 Discrete closure problem

Next, we consider the time evolution of $\bar{\mathbf{u}}$. Since we employ a spatial filter which does not depend on time, filtering and time-differentiation commute: $\mathbf{W}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \frac{\mathrm{d}(\mathbf{W}\mathbf{u})}{\mathrm{d}t}$. The closure problem arises because this is not true for the spatial discretization, i.e.

$$\mathbf{W} f_h(\mathbf{u}) \neq f_H(\mathbf{W}\mathbf{u}) \tag{2.23}$$

where f_H represents the same discretization scheme as f_h, but on the coarse grid. The closure problem is that the equations for $\bar{\mathbf{u}}$ are 'unclosed'. This means that knowing the fine-grid solution $\mathbf{u}$ is required to evolve $\bar{\mathbf{u}}$ in time. In this way, we do not achieve any computational speedup.

To resolve this, we write the filtered system in closure model form:

$$\frac{\mathrm{d}\bar{\mathbf{u}}}{\mathrm{d}t} = f_H(\bar{\mathbf{u}}) + \underbrace{(\mathbf{W}f_h(\mathbf{u}) - f_H(\bar{\mathbf{u}}))}_{=:\mathbf{c}(\mathbf{u})}, \tag{2.24}$$

where $\mathbf{c}(\mathbf{u}) \in \mathbb{R}^I$ is the closure term. Note that this equation is still exact. $\mathbf{c}(\mathbf{u})$ is essentially the discrete equivalent of the commutator error in LES [6]. One advantage of having first discretized the problem is that $\mathbf{c}(\mathbf{u})$ also includes the discretization error, with respect to a fine-grid simulation. The aim in closure modeling is generally to approximate $\mathbf{c}(\mathbf{u})$ by a closure model $\tilde{\mathbf{c}}(\bar{\mathbf{u}})$. In Section 2.3 we choose to represent $\tilde{\mathbf{c}}$ by a neural network.

2.2.5 Inner products and energy decomposition

To describe the total energy that is present in the system, we define the following inner products and norms:

$$(\mathbf{a}, \mathbf{b})_\xi := \mathbf{a}^T \boldsymbol{\xi} \mathbf{b} \tag{2.25}$$

$$||\mathbf{a}||_\xi^2 := (\mathbf{a}, \mathbf{a})_\xi \tag{2.26}$$

for $\boldsymbol{\xi} \in \{\boldsymbol{\omega}, \boldsymbol{\Omega}\}$, and vectors $\mathbf{a}$ and $\mathbf{b}$. With this notation we can represent the inner product on both the fine and coarse grid. For $\boldsymbol{\xi} = \mathbf{I}$ we obtain the conventional inner product and 2-norm, denoted as $(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T\mathbf{b}$ and $||\mathbf{a}||_2^2$. Besides the projection property (2.20) an additional characteristic of the filter/reconstruction pair is that the inner product is conserved under reconstruction (see Appendix A.1):

$$(\mathbf{R}\bar{\mathbf{a}}, \mathbf{R}\bar{\mathbf{b}})_\omega = (\bar{\mathbf{a}}, \bar{\mathbf{b}})_\Omega. \tag{2.27}$$

Using this, the total energy E_h in the system can be decomposed as

$$\begin{aligned} E_h :=& \frac{1}{2}||\mathbf{u}||_\omega^2 = \frac{1}{2}||\mathbf{R}\bar{\mathbf{u}} + \mathbf{u}'||_\omega^2 \\ =& \frac{1}{2}||\mathbf{R}\bar{\mathbf{u}}||_\omega^2 + (\mathbf{R}\bar{\mathbf{u}}, \mathbf{u}')_\omega + \frac{1}{2}||\mathbf{u}'||_\omega^2 = \underbrace{\frac{1}{2}||\bar{\mathbf{u}}||_\Omega^2}_{=:\bar{E}_h} + \underbrace{\frac{1}{2}||\mathbf{u}'||_\omega^2}_{=:E_h'} \end{aligned} \tag{2.28}$$

where we replaced $\mathbf{u}$ by the decomposition in (2.21). Furthermore, we used the fact that $\mathbf{R}\bar{\mathbf{u}}$ is orthogonal to $\mathbf{u}'$ to simplify the expression, see Appendix A.1. The final expression shows

that our choice of $\mathbf{W}$ and $\mathbf{R}$ is such that the total energy of the system can be split into two parts. These constitute of the resolved energy $\bar{E}_h$, which exclusively depends on $\bar{\mathbf{u}}$, and the SGS energy E'_h, which exclusively depends on $\mathbf{u}'$. The energy conservation law can also be decomposed into a resolved and SGS part:

$$\frac{\mathrm{d}E_h}{\mathrm{d}t} = \frac{\mathrm{d}\bar{E}_h}{\mathrm{d}t} + \frac{\mathrm{d}E'_h}{\mathrm{d}t} = \left(\bar{\mathbf{u}}, \frac{\mathrm{d}\bar{\mathbf{u}}}{\mathrm{d}t}\right)_\Omega + \left(\mathbf{u}', \frac{\mathrm{d}\mathbf{u}'}{\mathrm{d}t}\right)_\omega = 0, \tag{2.29}$$

where we used the product rule to arrive at this relation. For Burgers' equation with $\nu > 0$, the last equality sign changes to $\leq$. This means that even for dissipative systems, the resolved energy can still increase (so-called 'backscatter'), as long as the total energy is decreasing. For the KdV equation (2.13), which is strictly energy conserving, this decomposition can be seen in Figure 2.2. Here one can clearly see the continuous exchange of energy between $\bar{E}_h$ and E'_h, while the sum of the two remains constant.

2.2.6 Momentum conservation

Next to the energy, we investigate the effect of filtering on the momentum. The total discrete momentum is given by

$$P_h = (\mathbf{1}_\omega, \mathbf{u})_\omega, \tag{2.30}$$

where $\mathbf{1}_\omega$ is a vector with all entries equal to one, defined on the fine grid. From this definition we can show, see Appendix A.1, that the discrete momentum is invariant upon filtering, i.e.

$$(\mathbf{1}_\omega, \mathbf{u})_\omega = (\mathbf{1}_\Omega, \bar{\mathbf{u}})_\Omega. \tag{2.31}$$

This means the closure term does not add momentum into the system, i.e.

$$(\mathbf{1}_\Omega, \mathbf{c}(\mathbf{u}))_\Omega = 0. \tag{2.32}$$

2.3 Structure-preserving closure modeling framework

In this section, the derived discrete energy and momentum balances will be used to construct a novel structure-preserving closure model. We will also discuss how to fit the parameters of the model.

2.3.1 The framework

Many existing closure approaches aim at approximating $\mathbf{c}(\mathbf{u})$ by a closure model $\tilde{\mathbf{c}}(\bar{\mathbf{u}}; \boldsymbol{\Theta})$. Here $\boldsymbol{\Theta}$ are parameters to be determined such that the approximation is accurate. In this work, we propose a novel formulation, in which we extend the system of equations for $\bar{\mathbf{u}}$ with

a set of I auxiliary SGS variables $\mathbf{s} \in \mathbb{R}^I$. These SGS variables locally approximate the SGS energy, but projected onto the coarse grid. This will be detailed later. The extended system of equations has the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\bar{\mathbf{u}}\\ \mathbf{s}\end{bmatrix} \approx \mathcal{G}_\Theta(\bar{\mathbf{u}},\mathbf{s}) := \begin{bmatrix}f_H(\bar{\mathbf{u}})\\ \mathbf{0}_\Omega\end{bmatrix} + \mathbf{\Omega}_2^{-1}(\mathcal{K}-\mathcal{K}^T)\begin{bmatrix}\bar{\mathbf{u}}\\ \mathbf{s}\end{bmatrix} - \mathbf{\Omega}_2^{-1}\mathcal{Q}^T\mathcal{Q}\begin{bmatrix}\bar{\mathbf{u}}\\ \mathbf{s}\end{bmatrix}, \tag{2.33}$$

where $\mathcal{K} = \mathcal{K}(\bar{\mathbf{u}},\mathbf{s},\mathbf{\Theta}) \in \mathbb{R}^{2I\times 2I}$ and $\mathcal{Q} = \mathcal{Q}(\bar{\mathbf{u}},\mathbf{s},\mathbf{\Theta}) \in \mathbb{R}^{2I\times 2I}$ depend on the solution in a parameterized fashion. Next to the introduction of $\mathbf{s}$, the second main novelty in this work is to formulate the closure model in terms of a skew-symmetric and a dissipative term. The skew-symmetric term is introduced to allow for the exchange of energy between $\bar{\mathbf{u}}$ and $\mathbf{s}$. The dissipative term is introduced to provide additional dissipation, as this is required (see Appendix A.2). The operators $\mathcal{K}$ and $\mathcal{Q}$ will be modeled in terms of neural networks (NNs) with trainable parameters (contained in $\mathbf{\Theta}$). So even though the notation in (2.33) suggests linearity of the closure model, the dependence of $\mathcal{K}$ and $\mathcal{Q}$ on $\bar{\mathbf{u}}$ and $\mathbf{s}$ makes the model nonlinear. The construction of the introduced operators will be detailed in Sections 2.3.3. As our energy definition includes $\mathbf{\Omega}$, we include the inverse of the concatenated mass matrix $\mathbf{\Omega}_2^{-1}$ in our system of equations (2.33). This ensures energy conservation/dissipation regardless of the grid topology. This mass matrix is defined as

$$\mathbf{\Omega}_2 = \begin{bmatrix}\mathbf{\Omega} & \\ & \mathbf{\Omega}\end{bmatrix}. \tag{2.34}$$

Given the extended system of equations, the total energy (2.28) is approximated as

$$E_h \approx E_s := \frac{1}{2}||\mathbf{a}||^2_{\Omega_2} = \frac{1}{2}||\bar{\mathbf{u}}||^2_\Omega + \frac{1}{2}||\mathbf{s}||^2_\Omega, \tag{2.35}$$

where the second term approximates the SGS energy. Furthermore, we concatenate $\bar{\mathbf{u}}$ and $\mathbf{s}$ into a single state vector $\mathbf{a} \in \mathbb{R}^{2I}$:

$$\mathbf{a} := \begin{bmatrix}\bar{\mathbf{u}}\\ \mathbf{s}\end{bmatrix}. \tag{2.36}$$

The evolution equation for the approximated total energy is given by

$$\frac{\mathrm{d}E_s}{\mathrm{d}t} = \left(\mathbf{a}, \frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t}\right)_{\Omega_2} = (\bar{\mathbf{u}}, f_H(\bar{\mathbf{u}}))_\Omega - ||\mathcal{Q}\mathbf{a}||^2_2, \tag{2.37}$$

as the skew-symmetric term involving $\mathcal{K}-\mathcal{K}^T$ cancels. This is a property of skew-symmetric matrices [24]. Consequently, this formulation guarantees stability provided that f_H is structure-preserving.

Our key insight is that *by explicitly including an approximation of the SGS energy we*

are able to satisfy the energy conservation balance, equation (2.29). The energy balance serves not only as an important constraint for the closure model (represented by a NN), but also guarantees the stability of our closure model. This is because the energy is a norm of the solution, which is bounded in time. This framework thus allows for the modeling of backscatter without sacrificing stability.

2.3.2 SGS variables

Next, let us consider appropriate expressions for $\mathbf{s}$. The exact SGS energy on the coarse grid is given by:

$$\mathbf{W}(\mathbf{u}')^2, \tag{2.38}$$

where $(.)^2$ is to be interpreted element-wise. This would yield $\mathbf{s} = \pm\sqrt{\mathbf{W}(\mathbf{u}')^2}$, where $\sqrt{(.)}$ is also to be interpreted element-wise. The square root is taken to comply with the energy definition in (2.35). However, this definition for $\mathbf{s}$ resulted in poor performance during testing. We argue this was caused by the strict positivity (or negativity). Inevitable small errors in the model predictions caused some of the elements of $\mathbf{s}$ to switch sign. As the model was trained on positive $\mathbf{s}$, the simulation quickly diverged from the true trajectory. Attempts at resolving this issue were not successful.

Instead, we propose the use of a local linear compression. This formulation naturally allows for both positive and negative values of s_i. This compression is written as (assuming a uniform grid):

$$\mathrm{s}_i = \mathbf{t}^T \boldsymbol{\mu}_i, \qquad i = 1, \dots, I, \tag{2.39}$$

where we recall that $\boldsymbol{\mu}_i \in \mathbb{R}^J$ represents the SGS content in a single coarse cell Ω_i. Furthermore, $\mathbf{t} \in \mathbb{R}^J$ are the compression parameters. We aim to choose $\mathbf{t}$ such that we obtain $\mathbf{s}^2 \approx \mathbf{W}(\mathbf{u}')^2$. The optimal values of $\mathbf{t}$ are obtained using a singular value decomposition of the SGS content. This is outlined in Appendix A.3. From (2.39) we construct an operator $\mathbf{T}_s \in \mathbb{R}^{I \times N}$ which transform the SGS content into $\mathbf{s}$:

$$\mathbf{s} = \mathbf{T}_s \mathbf{u}'. \tag{2.40}$$

Combining the compression with the filter, see (2.15), we define the operator $\mathbf{T} \in \mathbb{R}^{2I \times N}$ which transforms $\mathbf{u}$ into the state vector $\mathbf{a}$:

$$\mathbf{a} = \underbrace{\begin{bmatrix} \mathbf{W} \\ \mathbf{T}_s(\mathbf{I} - \mathbf{R}\mathbf{W}) \end{bmatrix}}_{=:\mathbf{T}} \mathbf{u}. \tag{2.41}$$

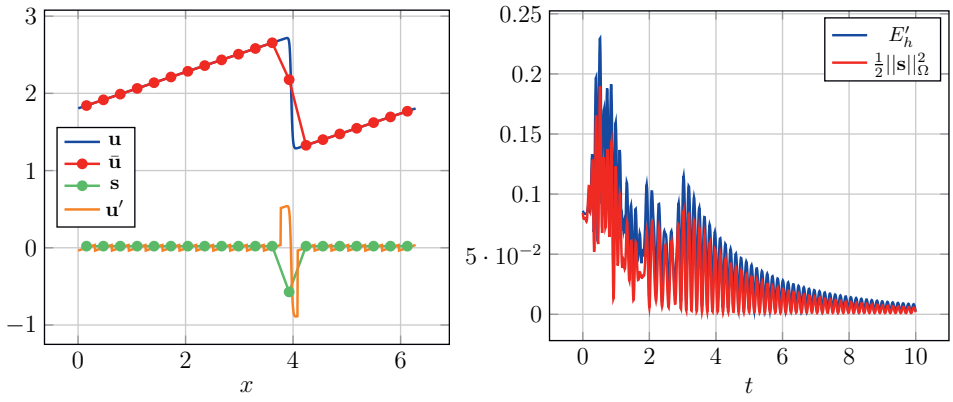


Figure 2.3: (Left) Learned SGS compression applied to Burgers' equation for $N = 1000$, with $I = 20$ and $J = 50$. By filtering and applying the SGS compression the degrees of freedom of this system are effectively reduced from $N = 1000$ to $2I = 40$. (Right) True SGS energy and compressed SGS energy during this simulation of Burgers' equation.

Due to the linearity of the transformation, we simply obtain

$$\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t} = \mathbf{T}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}, \tag{2.42}$$

where $\mathbf{T}$ is the Jacobian of the transformation. In the linear case, this Jacobian does not depend on $\mathbf{u}$, which significantly simplifies computing reference data for $\frac{\mathrm{d}\mathbf{a}}{\mathrm{d}t}$. In addition, it follows that if the true RHS includes a forcing term $\mathbf{F} \in \mathbb{R}^N$ we simply account for this by adding $\mathbf{TF}$ to the RHS of (2.33).

To illustrate how the compression works in practice, we consider a snapshot from a simulation of Burgers' equation ($\nu = 0.01$) with periodic BCs, see Figure 2.3. We observe that $\mathbf{s}$ serves as an energy storage for the SGS content, which is mainly present near shocks. If we look at the SGS energy trajectory, we find that its behavior is captured both qualitatively and quantitatively by the SGS compression. Although we still miss some of the energy, the oscillations due to the traveling shock are nicely captured. From this, we argue that a linear compression suffices. For more complex systems, autoencoders might offer an alternative [65].

2.3.3 Construction of the operators

Similarly to the structure-preserving discretization for Burgers' equation, presented in Section 2.2.2, we want our closure model to locally advect momentum and energy through the domain. In this way we do not violate the conservation laws. It is therefore that we inspire our machine learning closure model on this structure-preserving discretization. In this section we

outline how to construct $\mathcal{K}$ and $\mathcal{Q}$ from the output of a CNN [44].

2.3.3.1 Diffusive operator

Let us start by considering the diffusion operator in (2.9), namely $-\mathbf{Q}^T\text{diag}(\boldsymbol{\nu})\mathbf{Q}$. This operator locally diffuses energy and momentum through space at a rate which is determined by a positive viscosity field. In the periodic case the momentum is conserved, as $\mathbf{1}^T\mathbf{Q}^T = \mathbf{0}$. and the energy is dissipated, as $-\mathbf{u}^T\mathbf{Q}\text{diag}(\boldsymbol{\nu})\mathbf{Q}\mathbf{u} = -||\sqrt{\text{diag}(\boldsymbol{\nu})}\mathbf{Q}\mathbf{u}||_2^2 \leq 0$.

Drawing inspiration from this operator we introduce the following form for the diffusive term in our closure model:

$$\mathcal{Q}(\mathbf{a};\boldsymbol{\Theta}) = \mathbf{q}(\mathbf{a};\boldsymbol{\Theta})\mathcal{B}_1(\boldsymbol{\Theta}) \tag{2.43}$$

such that $\mathcal{Q}^T\mathcal{Q} = \mathcal{B}_1^T\mathbf{q}^2\mathcal{B}_1$ resembles the discussed diffusion operator. Here $\mathbf{q}(\mathbf{a};\boldsymbol{\Theta}) = \text{diag}(\mathbf{q}_1, \mathbf{q}_2) \in \mathbb{R}^{2I\times 2I}$ is constructed from two output channels $\mathbf{q}_1$ and $\mathbf{q}_2$ of a CNN which takes $\bar{\mathbf{u}}, \mathbf{s}$, and $f_H(\bar{\mathbf{u}})$ as inputs. $f_H(\bar{\mathbf{u}})$ was added as input channel because it significantly improved the performance of the neural network. This is supported by [55]. Looking at the introduced form, $\mathbf{q}^2$ can be thought of as a set of learned non-uniform and nonlinear viscosity fields. The square ensures positivity of these fields. Furthermore, the introduced matrix $\mathcal{B}_1(\boldsymbol{\Theta}) \in \mathbb{R}^{2I\times 2I}$ is a linear operator which encodes a set of parameterized convolutions. It will be used to satisfy momentum conservation, similarly to $\mathbf{Q}$. This will be discussed later in Section 2.3.3.3.

2.3.3.2 Advective operator

For our skew-symmetric operator $\mathcal{K} - \mathcal{K}^T$ we take a similar approach and introduce the following form:

$$\mathcal{K}(\mathbf{a};\boldsymbol{\Theta}) = \mathcal{B}_2^T(\boldsymbol{\Theta})\mathbf{k}(\mathbf{a};\boldsymbol{\Theta})\mathcal{B}_3(\boldsymbol{\Theta}) \tag{2.44}$$

such that $\mathcal{K}-\mathcal{K}^T = \mathcal{B}_2^T\mathbf{k}\mathcal{B}_3 - \mathcal{B}_3^T\mathbf{k}\mathcal{B}_2$. Similarly to the diffusive operator we use $\mathcal{B}_2(\boldsymbol{\Theta}), \mathcal{B}_3(\boldsymbol{\Theta}) \in \mathbb{R}^{2I\times 2I}$ to satisfy momentum conservation. As is the case for $\mathbf{q}$, the fields $\mathbf{k}(\mathbf{a};\boldsymbol{\Theta}) = \text{diag}(\mathbf{k}_1, \mathbf{k}_2)$ are constructed from an additional set of two outputs channels of the CNN, i.e.

$$\begin{bmatrix}\mathbf{q}_1 & \mathbf{q}_2 & \mathbf{k}_1 & \mathbf{k}_2\end{bmatrix} = \text{CNN}(\bar{\mathbf{u}}, \mathbf{s}, f_H(\bar{\mathbf{u}}); \boldsymbol{\Theta}). \tag{2.45}$$

This means the CNN has in total four output channels to construct both $\mathbf{q}$ and $\mathbf{k}$. Furthermore, $\mathbf{k}$ can be thought of as a set of learned velocity fields which advect momentum and energy through the domain, in addition to exchanging energy between $\bar{\mathbf{u}}$ and $\mathbf{s}$.

2.3.3.3 Momentum conservation

The entire framework (2.33) is summarized as

$$\mathcal{G}_\Theta(\mathbf{a}) = \begin{bmatrix} f_H(\bar{\mathbf{u}}) \\ \mathbf{0}_\Omega \end{bmatrix} + \mathbf{\Omega}_2^{-1}(\mathcal{B}_2^T \mathbf{k} \mathcal{B}_3 - \mathcal{B}_3^T \mathbf{k} \mathcal{B}_2)\mathbf{a} - \mathbf{\Omega}_2^{-1} \mathcal{B}_1^T \mathbf{q}^2 \mathcal{B}_1 \mathbf{a}. \tag{2.46}$$

From this we find that momentum conservation, see (2.32), places an additional constraint on the operators:

$$\left(\begin{bmatrix} \mathbf{1}_\Omega \\ \mathbf{0}_\Omega \end{bmatrix}, \mathcal{G}_\Theta(\mathbf{a}) \right)_{\Omega_2} = \begin{bmatrix} \mathbf{1}_\Omega \\ \mathbf{0}_\Omega \end{bmatrix}^T (\mathcal{B}_2^T \mathbf{k} \mathcal{B}_3 - \mathcal{B}_3^T \mathbf{k} \mathcal{B}_2)\mathbf{a} - \begin{bmatrix} \mathbf{1}_\Omega \\ \mathbf{0}_\Omega \end{bmatrix}^T \mathcal{B}_1^T \mathbf{q}^2 \mathcal{B}_1 \mathbf{a} = 0, \tag{2.47}$$

assuming that f_H is momentum conserving. This constraint is to be satisfied for periodic BCs, and we choose to compose $\mathcal{B}_i$ of convolutions (or stencils) expressed as linear operators:

$$\mathcal{B}_i = \begin{bmatrix} \mathbf{B}_i^{11} & \mathbf{B}_i^{12} \\ \mathbf{B}_i^{21} & \mathbf{B}_i^{22} \end{bmatrix}. \tag{2.48}$$

The operator $\mathcal{B}_i$ can therefore be thought of as the connection between two layers in a CNN, with each layer containing two channels [44]. In 1D each of the submatrices is characterized by $2B + 1$ parameters, where $B > 0$ is the width of the convolution. In this way, each convolution takes into account B neighboring grid cells from each side. Momentum conservation is ensured by constraining these submatrices in a clever way. From the constraint (2.47) and the definition of $\mathcal{B}_i$, in (2.48), we find that momentum conservation is satisfied if the sum of the convolution weights for $\mathbf{B}_i^{j1}$ is zero $\forall i, j$, as is the case for $\mathbf{Q}$. To see how this works we consider a general convolution operator $\mathbf{B}$, characterized by parameters $b_{-B}, \ldots, b_B \in \mathbb{R}$. Applying this operator to a discrete field $\mathbf{f}$, while constraining the sum of the weights to zero, is achieved as follows:

$$(\mathbf{B}\mathbf{f})_i = \sum_{j=-B}^{B} \bar{b}_j \mathrm{f}_{i+j}, \tag{2.49}$$

$$\bar{b}_j = b_j - \sum_{k=-B}^{B} \frac{b_k}{2B+1}, \tag{2.50}$$

such that $\sum_{j=-B}^{B} \bar{b}_j = 0$ indeed holds. Applying this procedure to $\mathbf{B}_i^{j1}$, $\forall i, j$, ensures (2.47) is satisfied up to machine precision, independent of the parameter values. The remaining convolutions are left unconstrained, i.e. we simply take $\bar{b}_j = b_j$.

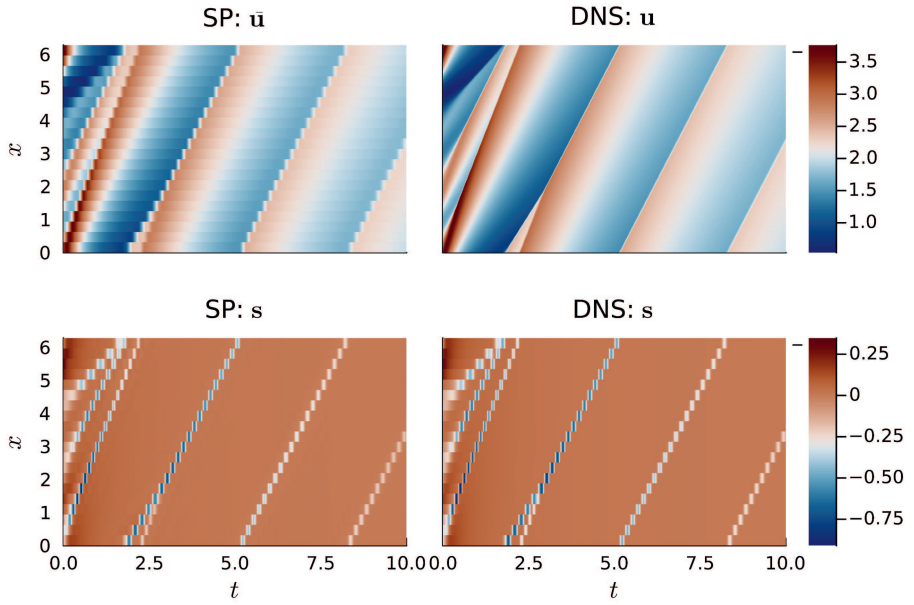


Figure 2.4: A simulation of Burgers' equation with periodic BCs using our trained structure-preserving closure model for $I = 20$ and $J = 50$ (left), along with the DNS solution for $N = 1000$ (right).

2.3.3.4 Properties & further discussion

The key insight is that we are free to choose any set of parameters $\boldsymbol{\Theta}$ without violating the prescribed structure of the system. Furthermore, as $\mathcal{K}$ and $\mathcal{Q}$ are based solely on convolutions in the CNN and the $\mathcal{B}$ matrices, they effectively correspond to nonlinear local stencils. The entire framework is therefore translation equivariant. For periodic BCs we apply circular padding to both the CNN inputs and the state vector [44]. For non-periodic BCs we refer to Appendix A.3.1. The parameters $\boldsymbol{\Theta}$ include the weights of the CNN, as well as the parameters characterizing the $\mathcal{B}$ matrices. As both the CNN and the convolution operations in $\mathcal{B}$ are sparse, our model remains computationally efficient.

In Figure 2.4 the framework is applied to Burgers' equation, where we compare it to the direct numerical simulation (DNS). It is once again interesting to see that $\mathbf{s}$ is largest at the shocks, indicating the presence of significant SGS content. Comparing the magnitude of the different terms in (2.33), see Figure 2.5, we observe that the skew-symmetric term, that is responsible for redistributing the energy, is most important. In fact it is more important than the coarse-grid discretization. In other words, our closure model has learned dynamics that are highly significant to correctly predict the evolution of the filtered system. This means that even though the closure term is large, we can still accurately model it.

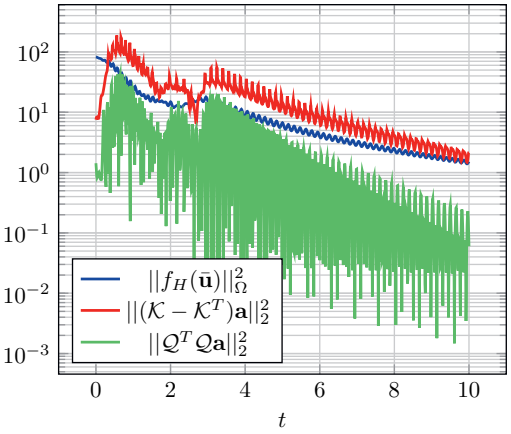


Figure 2.5: Magnitude of each of the different terms present in (2.33) corresponding to the simulation in Figure 2.4 with $I = 20$, $J = 50$, and $N = 1000$.

2.3.4 Finding the optimal parameter values

The optimal parameter values of the network can be obtained numerically by minimizing

$$\mathcal{L}(\mathbf{X}_\mathrm{u};\boldsymbol{\Theta}) := \frac{1}{p}\sum_{\mathbf{u}\in\mathbf{X}_\mathrm{u}} ||\mathcal{G}_\Theta(\mathbf{Tu}) - \mathbf{T}f_h(\mathbf{u})||_2^2 \tag{2.51}$$

with respect to $\boldsymbol{\Theta}$ for the training set $\mathbf{X}_\mathrm{u}$ containing p DNS snapshots. We will refer to this approach as 'derivative fitting', as we minimize the residual between the predicted and the true RHS.

An alternative is to optimize $\boldsymbol{\Theta}$ such that the solution itself is accurately reproduced. To achieve this we minimize

$$\mathcal{L}_n(\mathbf{X}_\mathrm{u};\boldsymbol{\Theta}) := \frac{1}{pn}\sum_{\mathbf{u}\in\mathbf{X}_\mathrm{u}}\sum_{i=1}^{n} ||\bar{\mathcal{S}}^i_{\boldsymbol{\Theta}}(\mathbf{Tu}) - \mathbf{T}\mathcal{S}^{i(\overline{\Delta t}/\Delta t)}(\mathbf{u})||_2^2. \tag{2.52}$$

Here $\bar{\mathcal{S}}^i_{\boldsymbol{\Theta}}(\mathbf{Tu})$ represents the output of the solver after successive application of an explicit time integration scheme for i time steps, with step size $\overline{\Delta t}$, starting from initial condition $\mathbf{Tu}$, using the introduced closure model in (2.33). The DNS counterpart is indicated by $\mathcal{S}^{i(\overline{\Delta t}/\Delta t)}(\mathbf{u})$, with step size Δt, starting from initial condition $\mathbf{u}$. Note the appearance of the ratio $\overline{\Delta t}/\Delta t$. This is because the coarse grid allows for larger time steps [98]. We will refer to this method as 'trajectory fitting'. This approach has been shown to yield more accurate and stable closure models [39, 52–55]. In this chapter we propose a hybrid of the two approaches, as trajectory fitting is more computationally more expensive. This hybrid approach will be detailed later.

2.4 Results

2.4.1 Test cases

To test our closure modeling framework we consider the previously introduced Burgers' equation with $\nu = 0.01$ on the spatial domain $\Omega = [0, 2\pi]$ for two test cases: (i) periodic BCs without forcing and (ii) input/output (I/O) BCs with time-independent forcing. The implementation of BCs is discussed in Appendix A.3.1. We also consider a third test case: (iii) the KdV equation with $\varepsilon = 6$ and $\mu = 1$ on the spatial domain $\Omega = [0, 32]$ for periodic BCs. Parameter values for Burgers' and KdV are taken from [90]. Reference DNSs are carried out on a uniform grid of $N = 1000$ for Burgers' and $N = 600$ for KdV up to time $T = 10$. The data that is generated from these reference simulations is split into a training (70%) and a validation set (30%). The simulation conditions (initial conditions, BCs, and forcing) for training are generated randomly, as described in Appendix A.3.2. The closure models will be tested on unseen simulation conditions, sampled from the same distributions as the training data. In addition to this, the construction of the training and validation set, the training procedure, and the hyperparameter tuning procedure are also described in Appendix A.3.2.

2.4.1.1 Considered closure models

For the analysis, we will compare our structure-preserving neural network architecture (SP) to a vanilla CNN. The vanilla CNN output is multiplied by a forward difference operator and the result is used as a closure model, i.e.

$$\tilde{\mathbf{c}}(\mathbf{u}; \boldsymbol{\Theta}) = \bar{\mathbf{Q}}\text{CNN}(\bar{\mathbf{u}}, f_H(\bar{\mathbf{u}}); \boldsymbol{\Theta}), \tag{2.53}$$

where $\bar{\mathbf{Q}}$ is a forward difference discretization of the first derivative on the coarse grid. This is done to satisfy momentum conservation [55, 89]. In addition, we consider a constant Smagorinsky model (Chapter 2) (SM) for use case (i) and (ii). This model contains a single scalar parameter C_s. This parameter is optimized in the same way as the NN parameters. We chose this model as it was the best performing non-machine learning closure in [89], outperforming the dynamic Smagorinsky model. After discretization, it takes the following form:

$$\tilde{\mathbf{c}}(\mathbf{u}; C_s) = -\bar{\mathbf{Q}}^T \text{diag}(\boldsymbol{\nu}_t(C_s))\bar{\mathbf{Q}}\bar{\mathbf{u}}, \tag{2.54}$$

$$\boldsymbol{\nu}_t(C_s) = (hC_s)^2|\bar{\mathbf{Q}}\bar{\mathbf{u}}|, \tag{2.55}$$

such that $\boldsymbol{\nu}_t$ represents a parameterized and solution dependent viscosity. Here we have taken the filter width equal to the grid-spacing h. Moreover, we also consider SP0 which initializes the simulation with $\mathbf{s}(t=0) = \mathbf{0}_\Omega$ instead of the true $\mathbf{s}$. This emulates the situation in which the true initial condition is unknown. Finally, we consider the no closure (NC) case, which corresponds to a coarse-grid solution of the PDEs. To march the solution forward in time we employ an explicit RK4 scheme [96] with time step size $\overline{\Delta t} = 0.01$ ($4\times$ larger than the DNS) for Burgers' and $\overline{\Delta t} = 5 \times 10^{-3}$ ($50\times$ larger than the DNS) for KdV.

To make a fair comparison, we compare closure models with the same number of degrees of freedom (DOF). For SP we have DOF $= 2I$, as we obtain an additional set of I degrees of freedom corresponding to the addition of the SGS variables. For the remaining closure model,s we simply have DOF $= I$.

2.4.2 Training the closure models

As use cases (i) and (ii) both correspond to Burgers' equation, we train the corresponding closure models on a dataset containing both simulation conditions. In this way, we end up with a single closure model that works for both (i) and (ii). The SP closure models contain a total of 2780 parameters (consisting of two hidden layers with each 20 channels and a kernel size of 5 for the underlying CNN) for Burgers' equation and 5352 (consisting of two hidden layers with each 30 channels and a kernel size of 5) for KdV. The purely CNN-based closure models consist of 3261 parameters (two hidden layers with each 20 channels and a kernel size of 7). These settings are based on the hyperparameter tuning procedure in Appendix A.3.2. For KdV we omit the dissipative component in (2.33), as it is a conservative system. In between hidden layers, we employ the ReLU activation function. We employ a linear activation function at the final layer. For Burgers' we choose $B = 1$ for the construction of the $\mathcal{B}$ matrices, matching the width of the coarse discretization. For KdV we do the same and therefore take $B = 2$.

As stated earlier, the model parameters are optimized by first derivative fitting and then trajectory fitting. During testing, we observed that solely derivative fitting resulted in instabilities and poor performance for the vanilla CNN, especially in the KdV case. In contrast, our SP method is guaranteed to be stable, and simply derivative fitting already resulted in reasonable performance. To maximize the potential of each closure model, we decided to include trajectory fitting in the training procedure. The full procedure is outlined in Appendix A.3.2. We implemented our closure models in the Julia programming language [99] using the Flux.jl package [100, 101]. The code can be found at `https://github.com/tobyvg/ECNCM_1D`.

2.4.3 Closure model performance

We examine the performance of the trained closure models based on how well the filtered DNS solution is reproduced. For our comparison, we will make extensive use of the normalized root-mean-squared-error (NMRSE) metric. This metric is defined as

$$\text{NRMSE } \bar{\mathbf{u}}(t) = \sqrt{\frac{1}{|\Omega|}||\bar{\mathbf{u}}(t) - \bar{\mathbf{u}}^{\text{DNS}}(t)||_{\Omega}^{2}}. \tag{2.56}$$

It is used to compare the approximated solution $\bar{\mathbf{u}}$ at time t, living on the coarse grid, to the filtered DNS result $\bar{\mathbf{u}}^{\text{DNS}}$. We will refer to this metric as the solution error. In addition, we define the integrated normalized root-mean-squared-error (I-NRMSE) as

$$\text{I-NRMSE } \bar{\mathbf{u}} = \frac{1}{T}\sum_{i} \overline{\Delta t}\ \text{NRMSE } \bar{\mathbf{u}}(i\overline{\Delta t}), \qquad 0 \leq i\overline{\Delta t} \leq T, \tag{2.57}$$

such that the sum represents integrating the solution error in time. We will refer to this metric as the integrated solution error.

2.4.3.1 Convergence

As we refine the resolution of the coarse grid, and with this, increase the number of DOF, we expect convergence of both the compression error $\mathcal{L}_s$, see Appendix A.3, and the solution error. We consider DOF $\in \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$. For each DOF value, we train a different set of closure models. If N is not divisible by I, we first project the DNS result onto a grid with a resolution that is divisible by I before applying the filter. In total 36 closure models are trained: two (SP and CNN) for each combination of the 9 considered coarse-grid resolutions and equation (Burgers' and KdV).

The SGS compression error evaluated over the validation set is shown in Figure 2.6. We observe monotonic convergence of the compression error as we refine the grid. We expect the compression error to further converge to zero as we keep on refining. The faster convergence for the KdV equation is likely caused by the lower fine-grid resolution, as compared to Burgers' equation.

Next, we consider the integrated solution error, see Figure 2.7. The presented plots represent an average taken over 20 simulations with unseen simulation conditions. We consider different numbers of DOF, which enables us to evaluate the convergence. For test cases (i) and (ii), we observe almost monotonic convergence of the solution error for NC, SM, and SP/SP0. Furthermore, SP improves upon NC with roughly one order of magnitude, surpassing SM. SP0 only performs slightly worse than SP, but still converges relatively smoothly. On the other hand, the solution error for the CNN behaves quite

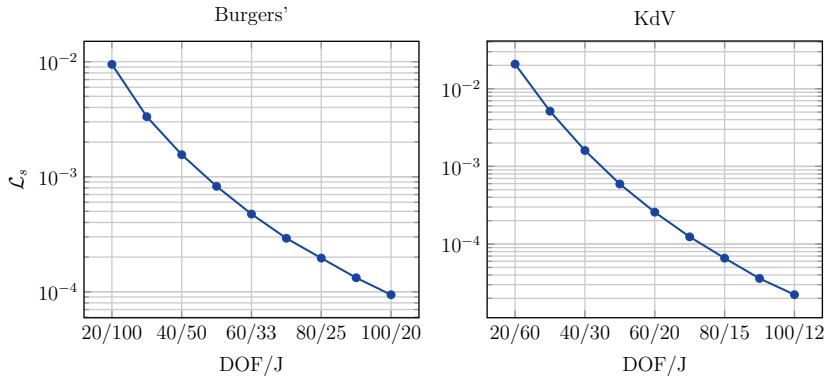


Figure 2.6: Convergence of the SGS compression error $\mathcal{L}_s$ when refining the coarse grid, evaluated on the validation set for Burgers' equation ($N = 1000$) and KdV equation ($N = 600$). Both the effective DOF and the corresponding compression factor J are depicted on the x-axis.

erratically: sometimes more accurate than SP, and sometimes less accurate than NC (case (i), DOF = 90).

For test case (iii) (KdV), we find that for most numbers of DOF the CNN outperforms SP, while not resulting in stable closure models for DOF $\in \{90, 100\}$. Furthermore, for the lower numbers of DOF we observe slightly better performance for SP. From this, we conclude that the compression error (see Figure 2.6) is likely not the limiting factor of the closure model performance. Looking at the difference between SP and SP0, it seems that initializing with the true $\mathbf{s}$ seems to lead to a larger difference, as compared to Burgers' equation. As SGS energy does not dissipate in the KdV equation, but flows back into the resolved scales, the dependence on the true $\mathbf{s}$ is likely higher than for Burgers'.

Overall, the conclusion is that our proposed SP closure model leads to more robust simulations than the CNN, while still improving upon NC with roughly an order of magnitude.

2.4.3.2 Computation time

Looking at the relative computation times in Figure 2.7 we find that both SP and the CNN are at least 2× faster than the DNS, for Burgers' equation. Furthermore, the computation time for the CNN is slightly shorter, but seems to increase at a larger rate when increasing the DOF. On our laptop CPU, the computation time of a single DNS took no longer than 5 seconds. For such small simulations, we expect the computation time to be largely determined by computational overhead. We therefore expect the relative speedup to increase for larger systems. In fact, the training time amounts to roughly 20 minutes for each neural network. This is rather long compared to a DNS, however for more complex/larger systems we expect

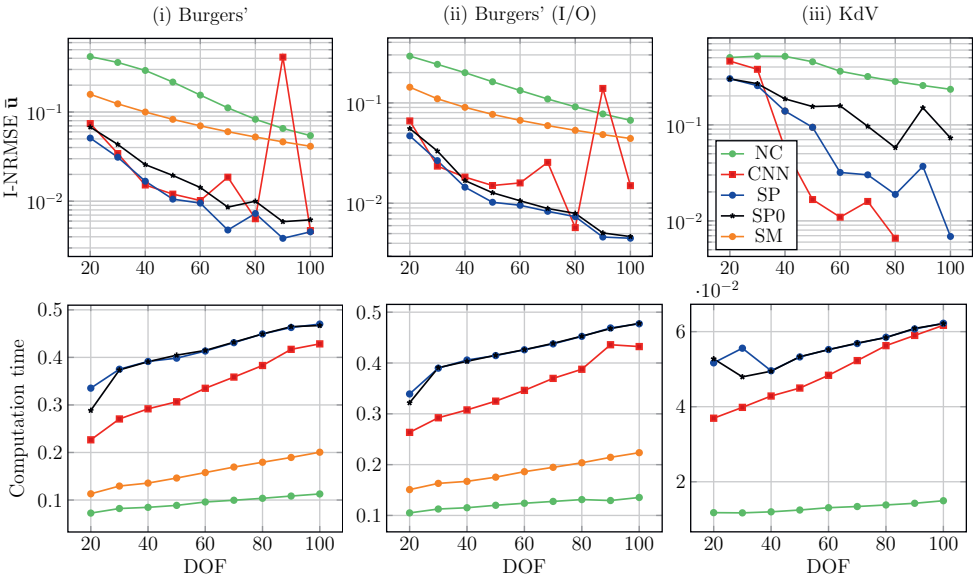


Figure 2.7: (Top) Integrated solution error evaluated at $T = 10$ averaged over 20 simulations for the different use cases (i)-(iii) and an increasing number of DOF. Only stable simulations are considered for the depicted averages. Absence of a scatter point indicates none of the simulations were stable. (Bottom) Similar to the top plots, but depicting the average simulation time as a fraction of the DNS time. **NC** = no closure, **CNN** = convolutional neural network closure, **SP** = structure-preserving closure, **SP0** = structure-preserving closure with $\mathbf{s}(t = 0) = \mathbf{0}_\Omega$, and **SM** = constant Smagorinsky model.

the significance of the training time to decrease, relative to the DNS time. Especially for 2D/3D problems where the curse of dimensionality sets in, significant speedups can be expected [39, 49]. Also, once a model is trained, it can be applied to unseen simulation conditions without retraining.

For the KdV equation, the relative speedup is roughly 20×. This is likely caused by the small time step size required for the DNS ($\Delta t = 10^{-4}$). Larger time step sizes for the DNS consistently resulted in unstable simulations within a few time steps. In addition, we find that computing the true $\mathbf{s}$ for the initial condition does not really affect the computation time.

2.4.4 Consistency of the training procedure

It is important to note that the closure models trained in the previous section possess a degree of randomness. This is caused by the (random) initialization of the network weights and the random selection of the mini-batches. This likely caused the irregular convergence behavior shown in the previous section. In order to evaluate this effect, we train 10 separate

replica models for DOF = 60, which only differ in the random seed. The trained models are evaluated in terms of stability (number of unstable simulations) and integrated solution error. A simulation is considered unstable when it produces NaN values for $\bar{\mathbf{u}}$. In total, 20 simulations per closure model are carried out using the same simulation conditions as in the convergence study. The results are depicted in Figure 2.8. With regard to stability, we

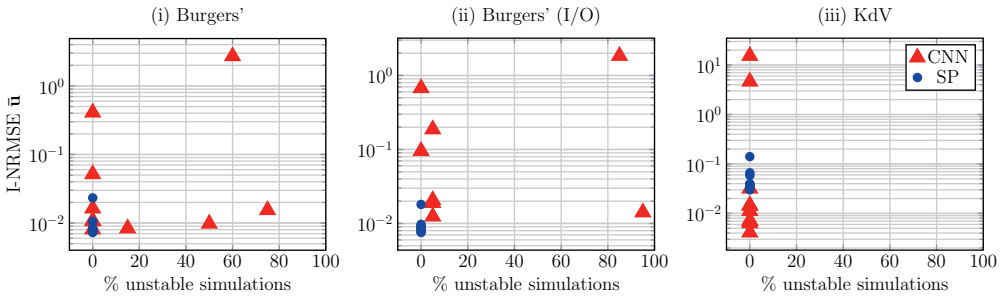


Figure 2.8: Integrated solution error evaluated at $T = 10$ averaged over 20 simulations and % of unstable simulations for each closure model in the trained ensemble of 10 closure models (DOF = 60). Use cases (i)-(iii) are considered. For (ii), two CNN closure models produced 100% unstable simulations and are therefore omitted from the graph. **CNN** = convolutional neural network closure and **SP** = structure-preserving closure.

observe that all trained SP closure models produced exclusively stable simulations. This is in accordance with the earlier derived stability condition (2.37). For the non-periodic test case (ii) we also observe a clear stability advantage, as compared to the CNN.

Regarding this integrated solution error, we observe that the SP closure models all perform very consistently (errors are almost overlapping). The CNNs sometimes outperform SP, but also show very large outliers. This confirms our conclusion of the previous section that our SP closure models are much more robust than the CNNs, which can be 'hit or miss' depending on the randomness in the training procedure. However, we still find that SP is often outperformed by a CNN for test case (iii).

2.4.5 Structure preservation

As we formulated a structure-preserving closure model, it is important to evaluate if the structure is indeed preserved. For this, we consider a single simulation of the KdV equation with periodic BCs, for DOF = 40. This is an interesting test case, as the energy should be exactly conserved. In Figure 2.9, we look at both the change in momentum and energy during the simulation. For momentum conservation, we also include the CNN, as this satisfies momentum conservation through multiplication by a discrete derivative operator, see (2.53). Regarding momentum conservation, we find that both SP and the CNN indeed conserve

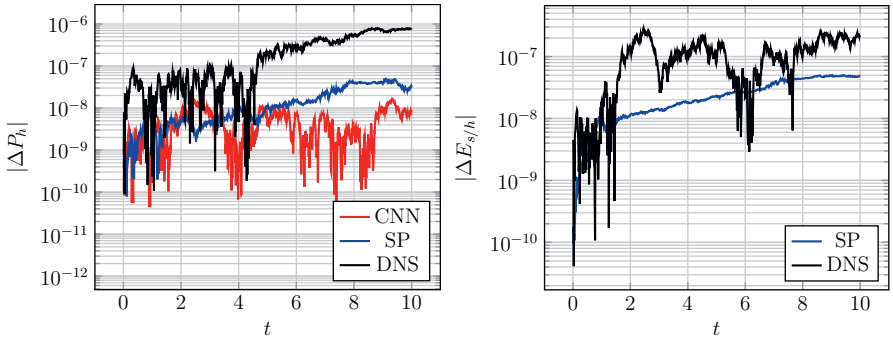


Figure 2.9: Change in momentum $\Delta P_h(t) = P_h(t) - P_h(0)$ (left) and total energy $\Delta E_{h/s}(t) = E_{h/s}(t) - E_{h/s}(0)$ (right) for a simulation of the KdV equation with periodic BCs starting from an unseen initial condition. The presented results correspond to DOF = 40. **DNS** = direct numerical simulation, **CNN** = convolutional neural network closure and **SP** = structure-preserving closure.

momentum up to machine precision (single-precision). For energy conservation, we observe the same for SP. From this, we conclude that SP indeed preserves the relevant structure.

Furthermore, we also consider a single simulation of Burgers' equation with periodic BCs, see Figure 2.10. Here we find that the total energy is always decreasing for SP. However,

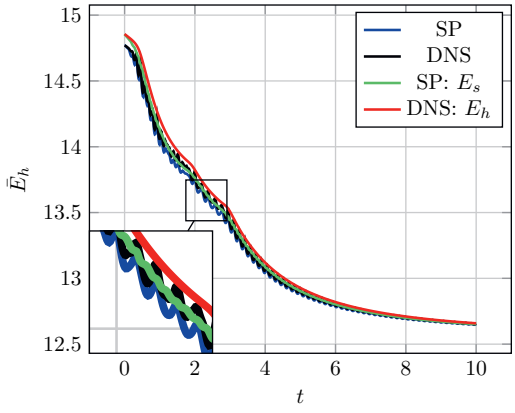


Figure 2.10: Resolved and total energy for a simulation of Burgers' equation with periodic BCs starting from an unseen initial condition. The presented results correspond to DOF = 40. **DNS** = direct numerical simulation and **SP** = structure-preserving closure.

we find the resolved energy to oscillate due to backscatter, which matches the filtered DNS result. From this, we conclude that, although the energies do not match the DNS result exactly, SP indeed allows for backscatter to be modeled correctly.

2.4.6 Burgers' equation & energy spectra

To find out what happens when a CNN becomes unstable, we consider Burgers' equation with periodic BCs, for DOF = 90. For this CNN we observed a rather large error spike in the convergence study (Figure 2.7). To analyze this error spike we consider a single simulation starting from an unseen initial condition. The results are depicted in Figure 2.11. Here we observe a large buildup of numerical noise for the CNN, as the simulation progresses. In addition, we observe that SP nicely suppresses the wiggles produced by NC around the shock [95]. SM also manages to do this, although to a lesser extent.

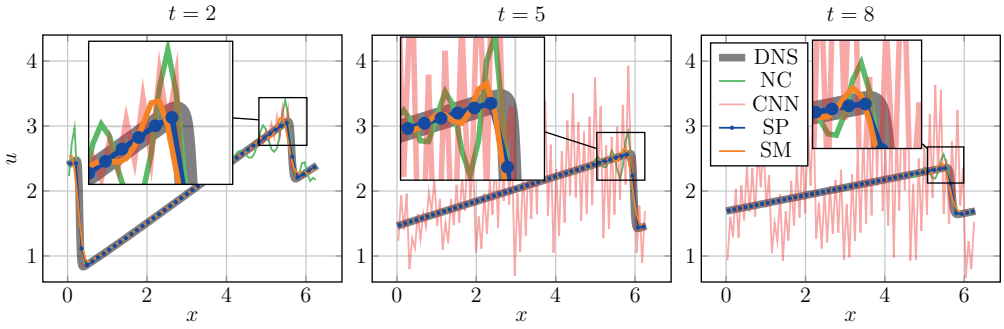


Figure 2.11: Solutions of Burgers' equation with periodic BCs at different points in time, starting from an unseen initial condition. The solutions are produced by the different closure models corresponding to DOF = 90. **DNS** = direct numerical simulation, **NC** = no closure, **CNN** = convolutional neural network closure, **SP** = structure-preserving closure, and **SM** = constant Smagorinsky model.

Next, we look at the energy trajectories and energy spectra corresponding to this simulation, see Figure 2.12. The energy spectra are given as a function of the wavenumber k. These spectra are computed by carrying out a discrete Fourier transform of $\bar{\mathbf{u}}$ and computing the energy for every k [102]. Furthermore, the energy spectra are only depicted up to the wavenumber resolved by the closure models. We find that for the CNN the energy starts to diverge at around $t = 2$. This worsens as the simulation progresses, and a large increase in energy is observed. Looking at the energy spectrum, this corresponds to a buildup of energy in the small scales (large k) and the numerical noise observed in the solution. This can cause the simulation to blow up. Furthermore, we find that SM is too dissipative. This results mostly in a lack of energy in the small scales. In addition, NC is not dissipative enough, resulting in a slight buildup of energy in the small scales. Finally, SP seems to capture the energy balance nicely, looking at both the trajectory and the spectrum. SP, and to a lesser extent NC, show the expected k^{-2} slope in the energy spectrum [103].

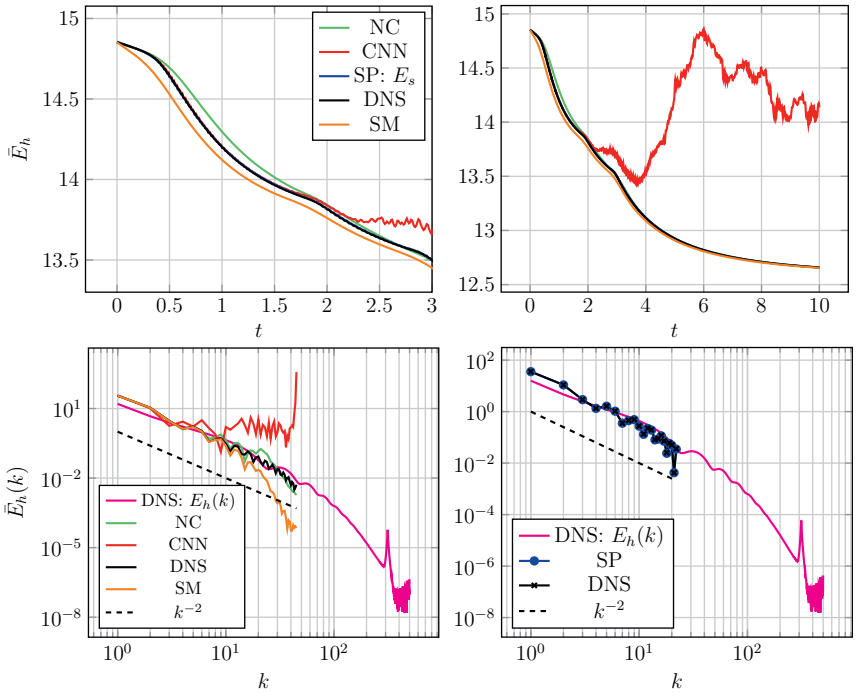


Figure 2.12: (Top) Resolved energies corresponding to the simulation in Figure 2.11 using the different closures for DOF = 90. For SP we depicted the total energy. The SP energy trajectory mostly overlaps with the DNS. (Bottom) The average energy spectra of this simulation, evaluated on the interval $3 \leq t \leq 7$. The left plot corresponds to $I = \text{DOF}$ and the right plot to $I = \text{DOF}/2$. Furthermore, we also show the spectra of the unfiltered DNS. These slightly differ from the filtered DNS in the small wavenumbers, as our filter is not a spectral filter [102]. **DNS** = direct numerical simulation, **NC** = no closure, **CNN** = convolutional neural network closure, **SP** = structure-preserving closure, and **SM** = constant Smagorinsky model.

2.4.7 Extrapolation in parameter space

Next, we are interested in how well such closure models are capable of extrapolating in parameter space. In particular, we consider different viscosity values ν for the Burgers' equation. For this purpose, we provide ν as an additional input to the neural network. For the training data we consider $\nu \in [10^{-2}, 10^{-1.75}, 10^{-1.5}, 10^{-1.25}, 10^{-1}]$. To generate the training data, we randomly generate three different initial conditions and carry out a DNS for each considered ν. To accommodate smaller values of ν used in the extrapolation experiment (up to $\nu = 10^{-3}$), we refine the grid to $N = 5000$. Furthermore, we use $\Delta t = 10^{-5}$ for the time integration and simulate up to $T = 5$. The smaller time step size is used to accommodate the finer grid. Every 1000th time step, we save the solution and use it as training data for

the closure models. For both SP and the CNN we train three closure models which only differ in the random seed used for the training procedure and initial parameter values. The networks underlying the methodologies are kept small, namely, only a single hidden layer with 20 channels. This is done, as smaller networks show higher potential for generalization [104]. The remaining hyperparameters and the training procedure are kept the same, see Appendix A.3.2. For our closure models, we reduce the DOF of the system from 5000 to 50 (a 100-fold reduction). We also use a larger time step size of $\overline{\Delta t} = 0.01$ (a 1000-fold increase). We evaluate the performance of each of the closure models for three simulations starting from different initial conditions. For each initial condition, we consider 9 different values of ν in the range $[10^{-3}, 10^{-1}]$. The integrated solution error, see (2.57), is once again used as the performance metric. The results are depicted in Figure 2.13. For each model, we find

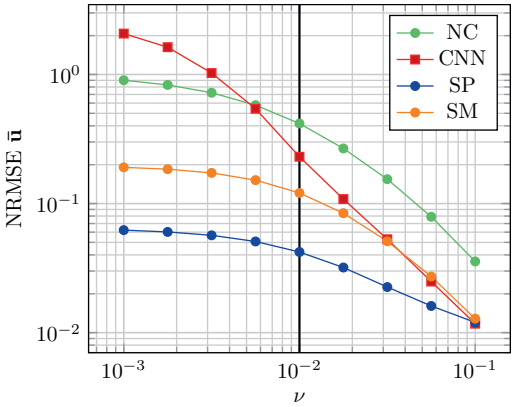


Figure 2.13: Integrated solution error evaluated at $T = 5$ averaged over the three trained closure models and three simulations for each viscosity value. The black vertical separates the training range (right) from the extrapolation range (left). **NC** = no closure, **CNN** = convolutional neural network closure, **SP** = structure-preserving closure, and **SM** = constant Smagorinsky model.

that the error increases for smaller values of ν. This is to be expected as smaller values of ν lead to stronger spatial gradients (shocks) which are harder to deal with in a numerical setting [24, 95]. When comparing the performance of the different closures, we find that our proposed SP scheme consistently outperforms the other closure models. This is the case for both inter- and extrapolation. Regarding the CNN, we observe a drop in performance as the viscosity decreases, even within the training range. It seems that the CNN is not able to adapt to the range of values of ν, as previously the CNN was able to perform quite well for DOF = 50 and a single viscosity value, see Figure 2.7. Furthermore, we find that SM outperforms NC across the considered range. This means that the obtained Smagorinsky constant generalizes well. This is to be expected as it is a simple model containing only a

single parameter [26, 104].

2.4.8 Extrapolation in space and time

As a final test case, we evaluate how well the closure models are capable of extrapolating in space and time. We consider the KdV equation on an extended spatial domain $\Omega = [0, 96]$ and run the simulation until $T = 50$. This corresponds to a $3\times$ and $5\times$ increase, respectively, with respect to the training data. We choose the KdV equation for this experiment as it is non-dissipative. It therefore leads to more interesting long-term behavior. As closure models, we use the ones trained during the convergence study that correspond to the grid-spacing of the grids employed in this test case.

Let us start by looking at snapshots from different points in the simulation, see Figure 2.14. At the edge of the training region $t = 10$, we find that the solutions are still roughly aligned, except for NC which already contains a lot of numerical noise. At $t = 25$, we find that both SP and the CNN start to diverge from the DNS. In addition, we observe a buildup of numerical noise for the CNN. This worsens at $t = 50$, while SP seems to remain free of numerical noise.

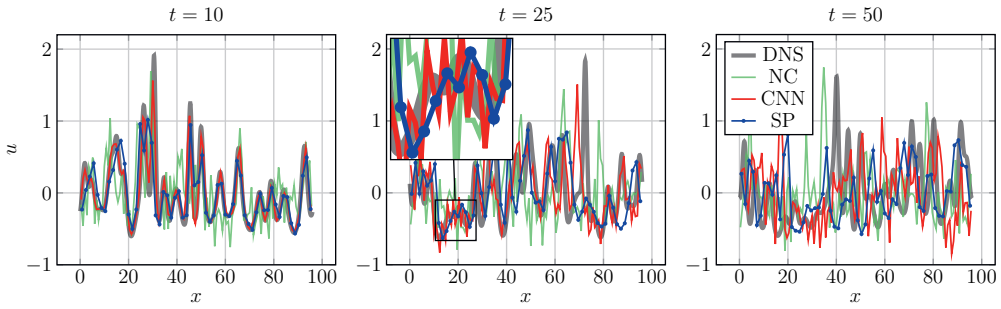


Figure 2.14: Solutions of the KdV equation with periodic BCs at different point in time, starting from an unseen initial condition. The solutions are produced by the different closure models corresponding to DOF = 40. In this case, the spatial and temporal domain are increased $3\times$ and $5\times$ with respect to the training data. **DNS** = direct numerical simulation, **NC** = no closure, **CNN** = convolutional neural network closure, and **SP** = structure-preserving closure.

To make a more thorough analysis, we consider the trajectories of the resolved energy. This is presented in Figure 2.15. We find that for SP the resolved energy (in blue) stays in close proximity to the filtered DNS (in magenta). This is in contrast to the CNN (in red), which starts to diverge from the DNS (in black) around $t = 5$. The resolved energy for the CNN also exceeds the maximum allowed total energy E_h (in orange) at different points in the simulation, which is nonphysical. We conclude that adding the SGS variables and conserving the total energy helps with capturing the delicate energy balance between resolved and SGS

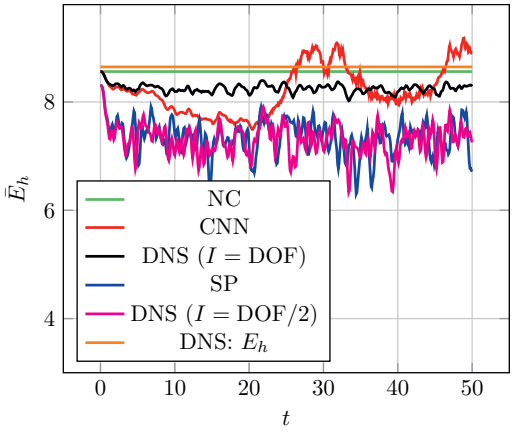


Figure 2.15: Trajectory of the resolved energy $\bar{E}_h$ for the simulation presented in Figure 2.14 for each of the different models corresponding to DOF = 40. The DNS resolved energy is depicted for both I = DOF (to compare with the CNN) and I = DOF/2 (to compare with SP). **DNS** = direct numerical simulation, **NC** = no closure, **CNN** = convolutional neural network closure, and **SP** = structure-preserving closure.

energy that characterizes the DNS. It is also interesting to note that NC (in green) conserves the resolved energy, as the coarse discretization conserves the discrete energy.

To make a more quantitative analysis of this phenomenon, we investigate the trajectory of the solution error and the Gaussian kernel density estimate (KDE) [105] of the resolved energy distributions. The latter analysis is carried out to analyze whether the closure models capture the correct energy balance between the resolved and SGS energy. The results for DOF $\in \{40, 60, 80\}$ are depicted in Figure 2.16. Looking at the solution error trajectories, we find that at the earlier stages of the simulation the CNN outperforms SP. However, SP slowly catches up with the CNN past the training region. With regards to the resolved energy distribution, we find that for each of the considered numbers of DOF SP is capable of reproducing the DNS distribution. On the other had, the CNN closure models struggle to capture this distribution. For DOF = 40 a significant part of the distribution even exceeds the total energy present in the DNS, i.e. there occurs a nonphysical influx of energy.

From this, we conclude that both SP and the CNN closure models are capable of extrapolating beyond the training data. However, only SP is capable of correctly capturing the energy balance between the resolved and unresolved scales. This allows it to more accurately capture the statistics of the DNS result.

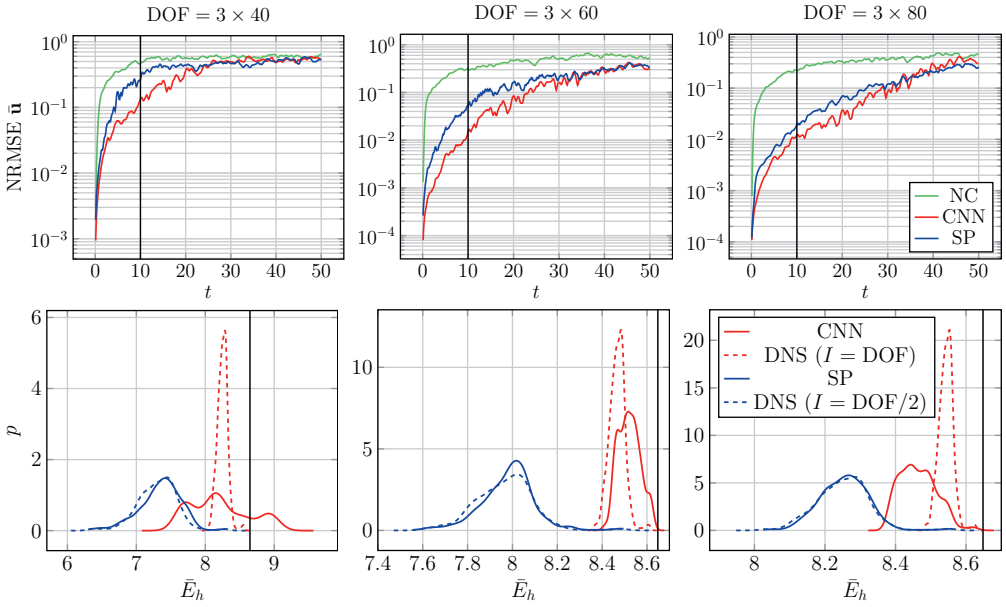


Figure 2.16: Solution error trajectory (top) and KDEs estimating the distribution of $\bar{E}_h$ (bottom) for the trained closure models corresponding to different numbers of DOF. These quantities are computed for a simulation of the KdV equation with the same initial condition on the extended spatial and temporal domain. In the top row, the vertical black line indicates the maximum time present in the training data, while in the bottom row it indicates the total energy of the DNS (which should not be exceeded). The DNS resolved energy is again depicted for both I = DOF (to compare with the CNN) and I = DOF/2 (to compare with SP). **NC** = no closure, **CNN** = convolutional neural network closure, and **SP** = structure-preserving closure.

2.5 Discussion on the applicability to 2D/3D Navier-Stokes

A number of challenges arise when considering to apply the proposed methodology to the 2D/3D Navier-Stokes equations. Firstly, the discrete nature of the presented closure modeling framework makes it dependent on the compression factor J between the coarse and the fine-grid resolution. This is a result of taking the 'discretize first' approach. While this allows the method to be highly specialized to the discretization, it can limit the applicability of the closure model to grid pairs with the associated J. One possibility is to train a single closure model with training data from multiple compression factors. Another possibility is to consider a continuous formulation of the closure problem, along with a continuous expression for the SGS variables. The finite element framework might be a useful starting

point, as it approximates the solution using a continuous, but still highly local, basis [106]. In addition, it easily allows for non-Cartesian grids. The convolutional layers in the closure model could then be replaced by graph convolutions [107], which work for unstructured grids. Another useful alternative is to consider the Fourier neural operator, which promises to be discretization invariant [93]. Furthermore, one could do a sparse regression on the closure model to identify physical terms [108], and train the model for different J to understand the effect of discretization error. This would effectively separate the two sources of error. Note that our framework can also be applied to nonuniform Cartesian grids, as long as the compression factor J is constant, and an energy-conserving discretization is available. The size of the computational domain can even be increased by an arbitrary factor as compared to the training domain, without retraining the closure model. The latter was shown in Section 2.4.8. For even more irregular grids, graph neural networks could be used to increase the flexibility of the framework [107].

A second challenge is the linear compression for $\mathbf{s}$, which might be a limiting factor for 2D/3D flows. As an alternative, non-linear compression methods like autoencoders could be employed [65]. Furthermore, the natural extension of the compressed SGS energy $\mathbf{s}$ to 2D and 3D is probably the sub-grid scale stress tensor. This means that our framework needs to be extended with multiple sub-grid scale equations, e.g. $\mathbf{s}_1$, $\mathbf{s}_2$, ... Finally, we could draw inspiration from the scale-adaptive simulation method, which also uses the SGS energy in its equations [109].

2.6 Conclusion

In this chapter, we proposed a novel way of constructing machine learning-based closure models in a structure-preserving fashion. We started by applying a spatial averaging filter to a fine-grid solution and writing the resulting system in closure model form. We showed that by applying this filter, we effectively remove part of the energy. Next, we introduced a linear compression of the SGSs into a set of SGS variables, defined on the coarse grid. These serve as a means of reintroducing the removed energy back into the system. This allows us to use the concept of kinetic energy conservation in closure modeling. In turn, we introduced an extended system of equations that models the evolution of the filtered solution as well as the evolution of the SGS variables. For this extended system, we proposed a structure-preserving closure modeling framework which allows for energy exchange between the filtered solution and the SGS variables, in addition to dissipation. This framework serves to constrain the underlying CNN such that no additional energy enters the system. In this way we achieve stability by abiding by the underlying energy conservation law. The advantage is that the framework still allows for backscatter through the energy present in the SGS variables. In addition, momentum conservation is also satisfied. Finally, the framework was applied to

both Burgers' and Korteweg-de Vries (KdV) equations.

A convergence study showed that the learned SGS variables are able to accurately match the original SGS energy content, with accuracy consistently improving when refining the coarse-grid resolution.

Given the SGS compression, our proposed SP framework was compared to a vanilla CNN. Overall, our SP method performed roughly on par with the CNN in terms of accuracy, albeit the CNN outperformed SP slightly for the KdV equation. However, the results for the CNN were typically inconsistent, not showing clear convergence of the error upon increasing the grid resolution. In addition, *the CNN suffered from stability issues.* On the other hand, our SP method produced stable results in all cases, while also consistently improving upon the 'no closure model' result. To be more specific, it did so by roughly an order of magnitude in terms of reproducing the reference solution.

This conclusion was further strengthened by training an ensemble of closure models. This was done to investigate the consistency of the closure model performance with respect to the randomness inherent to the training procedure. We observed that the trained vanilla CNNs differed significantly in performance and stability, whereas the different SP models performed very similarly to each other. The SP closures also displayed no stability issues. Our SP framework has therefore shown to be more robust and successfully resolves the stability issues which plague conventional CNNs.

Our numerical experiments confirmed the structure-preserving properties of our method: exact momentum conservation, energy conservation up to a time discretization error, and a strictly decreasing energy in the presence of dissipation. We also showed that our method succeeds in accurately modeling backscatter in both Burgers' and the KdV equation.

Regarding extrapolation in parameter space, our method outperformed both the CNN and the Smagorinsky model on viscosity values in and outside the training range. From this, we conclude our methodology leads to more accurate results than the conventional methods for this type of application. Furthermore, when extrapolating in space and time, the advantage of including the SGS variables and embedding structure-preserving properties became even more apparent: Our method is much better at capturing the delicate energy balance between the resolved and SGS energy. This, in turn, yielded better long-term error behavior.

Based on these results, we conclude that including the SGS variables, as well as adherence to the physical structure, has the important advantages of stability and long-term accuracy. In addition, it also leads to more consistent performance. This work, therefore, serves as an important starting point for building physical constraints into machine learning-based turbulence closure models. More generally, our framework is potentially applicable to a wide range of systems that possess multiscale behavior, while also possessing a secondary conservation law, for example, incompressible pipe flow [110]. Currently, our efforts are mainly

directed towards the incompressible Navier-Stokes equations. For instance, Kolmogorov flow would be a good starting point [111].

3

Energy-Conserving Neural Network Closure Model for Long-Time Accurate and Stable LES

Machine learning-based closure models for large eddy simulation (LES) have shown promise in capturing complex turbulence dynamics but often suffer from instabilities and physical inconsistencies. In this work, we develop a novel skew-symmetric neural architecture for closure modeling that enforces stability while preserving key physical conservation laws. The work in this chapter extends the work presented in Chapter 2 to multiple spatial dimensions. Our approach leverages a discretization that ensures mass, momentum, and energy conservation, along with a face-averaging filter to maintain mass conservation in coarse-grained velocity fields. We compare our model against several conventional data-driven closures (including unconstrained convolutional neural networks), and the physics-based Smagorinsky model. Performance is evaluated on decaying turbulence and Kolmogorov flow for multiple coarse-graining factors. In these test cases, we observe that unconstrained machine learning models suffer from numerical instabilities. In contrast, our skew-symmetric model remains stable across all tests, though at the cost of increased dissipation. Despite this trade-off, we demonstrate that our model still outperforms the Smagorinsky (both standard and dynamic variant) model in unseen scenarios. These findings highlight the potential of

This chapter is based on [13].

structure-preserving machine learning closures for reliable long-time LES.

3.1 Introduction

The incompressible Navier-Stokes equations are a set of partial differential equations (PDEs) that describe conservation of mass and momentum of fluid flows. They are used to model a multitude of flow phenomena, such as for the design of aircraft and ships, weather modeling, and even the formation of galaxies [82, 112]. To simulate these phenomena, we solve the Navier-Stokes equations on a computational grid. This requires discretizing the differential operators present in the PDE. This can be done with various techniques, such as finite difference, finite volume, and finite element methods [24, 106, 113]. In this work we employ a structure-preserving finite difference scheme, introduced in [25]. The advantage of this scheme is that it not only satisfies mass and momentum conservation, but also conserves the global kinetic energy (in absence of viscosity and boundary contributions). We refer to conservation of mass, momentum and kinetic energy collectively as the physical 'structure' of the system, and we refer to such conservative discretization schemes as 'structure-preserving' or 'symmetry-preserving' [23, 114]. Such schemes have the advantage of being unconditionally stable without relying on artificial diffusion, such as upwind schemes [115]. This makes them more suitable for long-time simulations, where correct physical energy behavior is crucial. However, problems arise when considering high Reynolds number flows [6]. For such flows we require very fine computational grids to resolve the smallest eddies in the system. This places a large (often insurmountable) burden on the computational resources.

A common approach to reduce computational requirements is large eddy simulation (LES). In LES, the system is coarse-grained by applying a filter to the velocity field, so that the dimension of the problem is effectively reduced. In this work, we take the 'discretize first, filter next' approach [15, 54, 55]. This means that coarse-graining is done on the discrete level by applying a discrete filter to a fine-grid discretization. In particular, we use a face-averaging filter, which has the advantage that the filtered velocity field still satisfies mass conservation [40]. The latter is necessary to preserve the energy-conserving properties of the convection operator. This provides substantial stability benefits, see [40]. From the coarse-graining procedure, a commutator error arises. In the 'discretize first, filter next' framework, this commutator error also includes the discretization error. Modeling the commutator error is referred to as closure modeling, and the corresponding models are closure models. Closure modeling is a main subject in LES research [6].

The most commonly used closure models are eddy-viscosity (functional) models [26, 47]. Their primary job is to remove (dissipate) energy from the system to account for energy transfer from the resolved scales to the unresolved scales. These models approximate the subgrid stress solely from the resolved scales by assuming proportionality between the subgrid-

scale stress tensor and the rate-of-strain tensor [2, 6]. The classical Smagorinsky model is the best-known example, but it is strictly dissipative [2, 6, 26] and therefore cannot represent backscatter, i.e., energy transfer from unresolved to resolved scales. Backscatter, however, plays an important role in many flows, including geophysical turbulence relevant for weather and climate modeling [84, 116, 117].

To address this limitation, structural models, such as scale-similarity closures, aim to reproduce the actual structure of the subgrid-scale stress tensor rather than parameterize its dissipation. While these models can, in principle, capture backscatter, they lack inherent dissipation. Consequently, they are often numerically unstable, requiring stabilization procedures such as explicit filtering or averaging. This need for both physical fidelity and numerical robustness motivated the development of mixed models, which combine a structural component with a dissipative functional component to achieve the accuracy of the former and the stability of the latter [29]. The dynamic Smagorinsky model [32] represents a related effort to increase flexibility within functional closures by adapting the eddy-viscosity coefficient based on resolved-scale information. Although it can produce limited backscatter, it too can suffer from numerical instabilities [10]. Several physics-based strategies have since been proposed to improve backscatter representation while maintaining stability. These include enforcing a consistent subgrid-scale energy budget [118], introducing filter-based artificial dissipation [119], and imposing explicit dissipation constraints [120, 121]. Nevertheless, state-of-the-art subgrid-scale closures used in global climate models still exclude backscatter [122], primarily due to concerns over robustness. In response, the machine-learned closure models that we will propose aim, in a somewhat similar spirit as mixed models, to combine the best of functional and structural approaches: we keep the structure of the subgrid stress tensor, while at the same time ensuring that the models are energy dissipative (but not as much as existing functional models).

Shifting our attention to such machine learning algorithms, neural networks have shown great potential in accurately modeling the closure term, while accounting for backscatter [14, 39, 45–51]. Offline testing often shows good agreement of the neural network outputs with the true closure term. However, when using the closure term in an actual simulation, problems arise and instabilities occur [14, 38, 39, 45, 48, 51]. One approach to handle this issue is by clipping the neural network such that the output becomes strictly dissipative, for example by projecting onto an eddy-viscosity basis [14, 38, 51]. However, this results in closure models which are generally too dissipative [51], which will be confirmed by our results. Another way to deal with instabilities is to add artificial noise to the training [12]. However, in [12] it was shown that this only delays the instability and does not prevent it entirely. Stochastic approaches have also been suggested, e.g. [50] combines idealized LES with neural stochastic differential equations and show increased stability.

Stability issues are often combatted by introducing some form of *a posteriori* learning

[16, 39, 46, 52–56]. In this way, a closure model is trained based on reproducing the solution trajectory rather than reproducing the closure term itself. This aids in the stability of the LES. In [39], it was shown that by increasing the number of solver steps one unrolls during training, the stability and performance are improved. [55] shows that there is an optimum in the number of unrolled steps, related to the chaotic nature of the system. However, in [51] it is shown that limited training data still causes instabilities for neural network-based closure models.

In [123], a strictly local small neural network approach is suggested for the representation of the subrid-scale stress tensor. Here, a small set of carefully chosen Galilean invariant and non-dimensionalized inputs are used such that the resulting closure model is symmetric, Galilean invariant, rotationally and reflectionally invariant, and unit invariant. The authors show that training on a single snapshot is enough to obtain a closure model which generalizes well to their considered test cases, even without clipping. However, in [47] it is shown that convolutional neural networks (CNNs), combined with Fourier neural operators, with non-invariant inputs, is capable of outperforming such approaches. For an overview on machine learning-based closure modeling see [16].

However, while the aforementioned references observed improved stability, none of the discussed approaches *guarantees* stability, without applying some form of backscatter clipping, or other ad-hoc measures, such as e.g. trial-and-error by changing input features [14] or data augmentation [12]. This makes long-time LES with a data-driven closure model unreliable. Given that we are especially interested in the statistics of turbulent flows, e.g. the average energy spectrum obtained over long time horizons, it is of crucial importance that the combination of discretization and closure model yields stable simulations. Hence, the main aim and novelty of this article is to derive a machine-learned LES closure model, where stability is guaranteed by design, independent of the network weights or amount of training data. To achieve this, we propose to build upon our earlier work presented in [15] (see Chapter 2). In this work, the closure model was represented by an energy-conserving skew-symmetric term and a dissipative symmetric negative-definite term. This was accomplished by using a set of compressed subgrid-scale variables that allow the transfer of energy from unresolved scales to resolved scales. In this work, we extend this approach from one to two spatial dimensions. While it is certainly true that turbulence is fundamentally a 3D phenomenon, we reiterate that the main aim of our article is focused on the stability issue, which has largely been studied in the context of 2D turbulence, see [39, 45, 47–49, 51, 52] among others.

As a first step, we exclude the compressed subgrid variables, introduced in our earlier work, as their precise meaning and definition in multiple dimensions is still an open question. Instead, we show that the skew-symmetric framework without subgrid variables already offers significant stability advantages over existing approaches. Although this means we do not explicitly account for backscatter, the skew-symmetric term is still capable of redistributing

energy throughout the domain. This extends the predictive capability of the closure model beyond an eddy-viscosity basis. The work presented in [124] discusses similar ideas regarding structure-preserving neural networks, although outside the realm of LES.

The outline of this chapter is as follows: In Section 3.2, we start with introducing the incompressible Navier-Stokes equations, the physical structure present in the system, and the structure-preserving discretization. In Section 3.3, we discuss coarse-graining of the discrete system of equations by applying a discrete spatial filter, we derive the exact closure term, and discuss closure modeling approaches. In Section 3.4, we introduce the machine learning-based closure models: using a CNN to model the closure term, using a CNN to model the subgrid-scale stress tensor, and finally our skew-symmetric neural network architecture. An extended motivation for this architecture can be found in Appendix B.3. Next, we discuss the test cases and the results in Section 3.5, regarding closure model performance and stability. We conclude our work in Section 3.6.

3.2 Preliminaries

3.2.1 Navier-Stokes equations

In conservative form, the incompressible Navier-Stokes equations read as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}^T) = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{3.1a}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{3.1b}$$

This PDE describes the evolution of a fluid velocity field $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^d$ in d-dimensional space $\mathbf{x} \in \mathbb{R}^d$ and time t. Here we restrict ourselves to 2D such that $\mathbf{u}(\mathbf{x}, t) = \begin{bmatrix} u(\mathbf{x}, t) & v(\mathbf{x}, t) \end{bmatrix}^T$. The different forces acting on the velocity field are due to convection, the gradient of the pressure $p(\mathbf{x}, t)$, and friction (for a nonzero viscosity $\nu \geq 0$), appearing from left to right in the equation. In addition, there is $\mathbf{f}(\mathbf{x}, t) \in \mathbb{R}^2$ which represents body-forces, such as gravity. In this text we refrain from discussing boundary conditions, as we consider a periodic spatial domain Ω.

3.2.2 Physical structure

This set of equations represent a set of fundamental physical laws, namely: conservation of mass, momentum $\mathbf{P} = \int_\Omega \mathbf{u} \mathrm{d}\Omega$, and (kinetic) energy $E := \frac{1}{2} \int_\Omega \mathbf{u} \cdot \mathbf{u} \mathrm{d}\Omega$ (for $\nu = 0$). These are

collectively referred to as the system's physical structure. In equation form, these laws read

$$\nabla \cdot \mathbf{u} = 0, \tag{3.2}$$

$$\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} = \int_\Omega \frac{\partial \mathbf{u}}{\partial t} \mathrm{d}\Omega = \int_\Omega \mathbf{f}(x,t) \mathrm{d}\Omega, \tag{3.3}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \int_\Omega \mathbf{u} \cdot \frac{\partial \mathbf{u}}{\partial t} \mathrm{d}\Omega = -\int_\Omega \nu ||\nabla \mathbf{u}||_2^2 \mathrm{d}\Omega + \int_\Omega \mathbf{u} \cdot \mathbf{f}(\mathbf{x},t) \mathrm{d}\Omega, \tag{3.4}$$

Derivations of these conservation laws are presented in Section 1.1.1. From these laws, we can see that momentum and energy (for zero dissipation) are conserved in the absence of forcing.

3.2.3 Discretization

To simulate practical use cases, we require discretizing the set of equations (3.1) on a computational grid. Here we employ a structure-preserving second-order accurate finite difference discretization on a staggered grid [24, 25]. This discretization is chosen as it satisfies the energy-conserving properties of the convective term. The employed discretization consists of in total $N_x \times N_y = N$ cells Ω_{ij} with $i = 1, \dots N_x$ and $i = 1, \dots N_y$ for a 2D flow case. The semi-discrete set of equations are written as

$$\mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} + \mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h = -\mathbf{G}_h \mathbf{p}_h + \nu \mathbf{D}\mathbf{u}_h + \mathbf{\Omega}_h \mathbf{f}_h, \tag{3.5a}$$

$$\mathbf{M}_h \mathbf{u}_h = \mathbf{0}, \tag{3.5b}$$

where $\mathbf{u}_h \in \mathbb{R}^{2N}$ contains the approximations of u and v on the cell faces and $\mathbf{p}_h \in \mathbb{R}^N$ the approximation of p in the cell centers. A schematic representation of the employed uniform staggered grid is displayed in Figure 3.1. The grid-spacing in each direction is indicated by h_x and h_y. Furthermore, $\mathbf{\Omega}_h$ contains the cell volumes on the diagonal. The operators in (3.1) are now represented by matrices. $\mathbf{C}_h(\mathbf{u}_h) \in \mathbb{R}^{2N \times 2N}$ represents the convection operator, $\mathbf{G}_h \in \mathbb{R}^{2N \times N}$ the gradient operator, $\mathbf{D}_h \in \mathbb{R}^{2N \times 2N}$ the diffusion operator, $\mathbf{M}_h \in \mathbb{R}^{N \times 2N}$ the divergence operator, and $\mathbf{f}_h \in \mathbb{R}^{2N}$ the forcing at the cell faces.

3.2.4 Structure of the discretization

This discretization preserves the physical structure in a discrete sense. Discretely, the total momentum and energy are approximated as

$$\mathbf{P}_h = \underbrace{\begin{bmatrix} \mathbf{1}_h & \mathbf{0}_h \\ \mathbf{0}_h & \mathbf{1}_h \end{bmatrix}^T}_{=:\mathbb{1}_h} \mathbf{\Omega}_h \mathbf{u}_h, \tag{3.6}$$

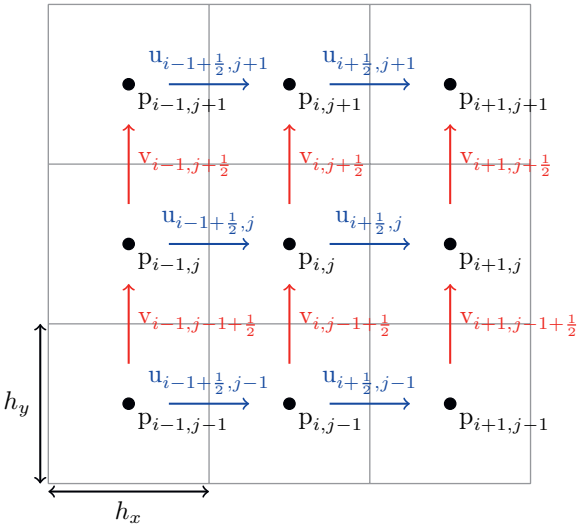


Figure 3.1: Staggered grid discretization of Navier-Stokes equations. The pressure points are indexed with whole numbers, while for the velocity components we have an offset of $\frac{1}{2}$ in the appropriate direction.

$$E_h = \frac{1}{2}\mathbf{u}_h^T \mathbf{\Omega}_h \mathbf{u}_h, \tag{3.7}$$

where $\mathbf{0}_h, \mathbf{1}_h \in \mathbb{R}^N$ are column vectors of zeros and ones, respectively. The change of these quantities for this discretization are given by

$$\frac{\mathrm{d}\mathbf{P}_h}{\mathrm{d}t} = \mathbb{1}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = \mathbb{1}_h \mathbf{\Omega}_h \mathbf{f}_h, \tag{3.8}$$

$$\frac{\mathrm{d}E_h}{\mathrm{d}t} = \mathbf{u}_h^T \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = -\nu ||\mathbf{Q}_h \mathbf{u}_h||_2^2 + \mathbf{u}_h^T \mathbf{\Omega}_h \mathbf{f}_h, \tag{3.9}$$

where we used the fact that the diffusion operator $\mathbf{D}_h$ can be Cholesky decomposed as $-\mathbf{Q}_h^T \mathbf{Q}_h$ [67]. As the discrete energy is solely decreasing, in the absence of forcing, this discretization provides stability. The convective contribution disappears due to the skew-symmetry of the discrete operator, however this requires the discrete solution $\mathbf{u}_h$ to be divergence-free, i.e., $\mathbf{M}_h \mathbf{u}_h = \mathbf{0}_h$. A more elaborate discussion is presented in Appendix B.1.

3.2.5 Pressure projection

The divergence-freeness can also be written as a projection of the right-hand side (RHS) of the PDE discretization (3.5) on a divergence-free basis. We introduce this formulation of the discrete system because it is more convenient for closure modeling [40], as it combines (3.5a)

and (3.5b) into a single equation. The projection looks as follows:

$$\mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = \mathcal{P}_h \mathbf{m}_h(\mathbf{u}_h), \tag{3.10}$$

where $\mathbf{m}_h(\mathbf{u}_h) \in \mathbb{R}^{2N}$ contains the operators on the RHS of (3.5a), i.e.,

$$\mathbf{m}_h(\mathbf{u}_h) = -\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h + \nu \mathbf{D}_h \mathbf{u}_h + \mathbf{\Omega}_h \mathbf{f}_h, \tag{3.11}$$

and $\mathcal{P}_h \in \mathbb{R}^{2N \times 2N}$ projects the RHS on a divergence-free basis. $\mathcal{P}_h$ is defined as

$$\mathcal{P}_h := (\mathbf{I} - \mathbf{G}_h(\mathbf{M}_h \mathbf{\Omega}_h^{-1} \mathbf{G}_h)^{-1} \mathbf{M}_h \mathbf{\Omega}_h^{-1}). \tag{3.12}$$

A derivation of this operator is presented in Appendix B.2.

3.3 Closure modeling

3.3.1 Filtering

As stated earlier, carrying out a direct numerical simulation (DNS) is often infeasible for practical use cases. This is why we aim to solve a coarse-grained set of equations. Coarse-graining is done by applying a filter to the velocity field. In our case, we take the 'discretize first, filter next' approach [54, 55]. This means we start by discretizing our velocity field on an adequately fine grid, such that we resolve the relevant scales of the flow. Next, we apply a linear filter $\mathbf{W}^{2\bar{N} \times 2N}$ to obtain the filtered velocity field:

$$\bar{\mathbf{u}}_H = \mathbf{W}\mathbf{u}_h, \tag{3.13}$$

where the filtered velocity $\bar{\mathbf{u}}_H \in \mathbb{R}^{2\bar{N}}$ lives on a coarse grid consisting of $\bar{N} = \bar{N}_x \times \bar{N}_y$ grid cells. In our case we employ a face-averaging filter, as suggested in [40]. A schematic representation of the filter is shown in Figure 3.2. The advantage of this filter, as opposed to, for example, a volume-averaging filter, is that the filtered velocity satisfies divergence-freeness on the coarse grid. This ensures skew-symmetry of the coarse-grid convection operator, which aids in stability of the coarse-grained system of equations [40].

3.3.2 System of equations

To model the behavior of the filtered velocity field, we take the following ansatz:

$$\mathbf{\Omega}_H \frac{\mathrm{d}\bar{\mathbf{u}}_H}{\mathrm{d}t} \approx \mathcal{P}_H \mathbf{m}_H(\bar{\mathbf{u}}_H) + \tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta), \tag{3.14}$$

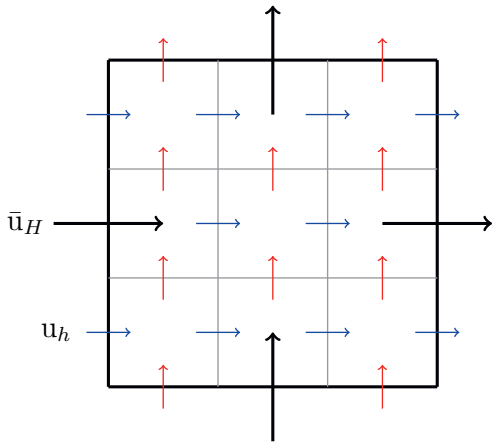


Figure 3.2: Schematic representation of the face-averaging filter. In this example, a single coarse-grained cell contains nine fine-grid cells. Three fine-grid velocity components in $\mathbf{u}_h$ on each of the coarse cell faces are averaged to obtain the filtered velocity field $\bar{\mathbf{u}}_H$.

where the subscript H indicates a coarse-grid equivalent to the operators discussed in Section 3.2.3 and $\tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta)$ is a (neural network-based) closure model with parameters θ. The projection on a divergence-free basis is achieved through $\mathcal{P}_H$, which is justified due to the face-averaging filter. The closure model is required as the coarse discretization does not resolve all the relevant scales of the flow. In addition, the coarse grid results in a discretization error. Both of these are captured in the commutator error with respect to the fine-grid discretization. The true evolution of $\bar{\mathbf{u}}_H$ is given by

$$\boldsymbol{\Omega}_H \frac{\mathrm{d}\bar{\mathbf{u}}_H}{\mathrm{d}t} = \mathcal{P}_H \mathbf{m}_H(\bar{\mathbf{u}}_H) + \underbrace{(\mathbf{W}\mathcal{P}_h \mathbf{m}_h(\mathbf{u}_h) - \mathcal{P}_H \mathbf{m}_H(\bar{\mathbf{u}}_H))}_{=:\mathbf{c}(\mathbf{u}_h)}, \tag{3.15}$$

where the commutator error $\mathbf{c}_h(\mathbf{u}_h)$ includes both sources of error. As the true filtered velocity field is divergence-free on the coarse grid, we include the closure model in the projection to preserve divergence-freeness, as suggested by [40]. This changes ansatz (3.14) to

$$\boldsymbol{\Omega}_H \frac{\mathrm{d}\bar{\mathbf{u}}_H}{\mathrm{d}t} \approx \mathcal{P}_H(\mathbf{m}_H(\bar{\mathbf{u}}_H) + \tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta)). \tag{3.16}$$

The energy contribution of the closure model is computed as

$$\text{energy contribution} = \bar{\mathbf{u}}_H^T \tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta), \tag{3.17}$$

where the resolved energy is given by

$$\bar{E}_H = \frac{1}{2}\bar{\mathbf{u}}_H^T \mathbf{\Omega}_H \bar{\mathbf{u}}_H. \tag{3.18}$$

In addition, the total momentum $\bar{\mathbf{P}}_H = \mathbb{1}_H \mathbf{\Omega}_H \mathbf{u}_H$ is conserved if

$$\mathbb{1}_H \tilde{\mathbf{c}}(\mathbf{u}_H, \theta) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.19}$$

holds for the closure model.

3.3.3 Energy analysis of the closure term

Based on training data, we analyze the energy contribution of the true closure term, see (3.17), for different levels of coarse-graining. In this case, the reference simulation was carried out on a 2048×2048 grid. Exact simulation conditions are discussed in Section 3.5.1. The resulting energy contributions, in addition to the resolved energy trajectories, are presented in Figure 3.3. We observe that for a resolution of 32×32 the closure term produces a significant amount of backscatter, as the energy contribution is mostly positive. Also, in the resolved energy trajectory we observe that the decay is less smooth, as opposed to larger resolutions. This means dissipative models will likely perform poorly for this resolution. For a resolution of 64×64, we find the energy contribution to be mostly dissipative, whereas for 128×128 it is strictly dissipative for this test case. This motivates the use of dissipative closure models, such as the skew-symmetric neural network architecture we will introduce in Section 3.4.3

3.3.4 Fitting of the model parameters

To find the optimal set of parameters θ, one can take different approaches. The most straightforward approach would be to match the true commutator error as best as possible. However, this often results in inaccurate simulations or even instabilities [12, 38, 39, 45, 46, 54, 55]. This is why we resort to optimizing the parameters in order to accurately reproduce the filtered direct numerical simulation (FDNS) solution, also known as 'trajectory fitting' or 'solver in the loop'. The corresponding loss function is:

$$\mathcal{L}_n(\mathbf{X}; \theta) = \sum_{\mathbf{u}_h \in \mathbf{X}} \sum_{i=1}^{n} ||\bar{\mathcal{S}}_\theta^i(\mathbf{W}\mathbf{u}_h) - \mathbf{W}\mathcal{S}^{i(\overline{\Delta t}/\Delta t)}(\mathbf{u}_h)||_2^2, \tag{3.20}$$

where $\mathbf{X}$ is a snapshot matrix consisting of samples of $\mathbf{u}_h$ as columns, representing the training data set. The notation $\bar{\mathcal{S}}_\theta^i(\mathbf{W}\mathbf{u}_h)$ represents the predicted coarsened velocity field

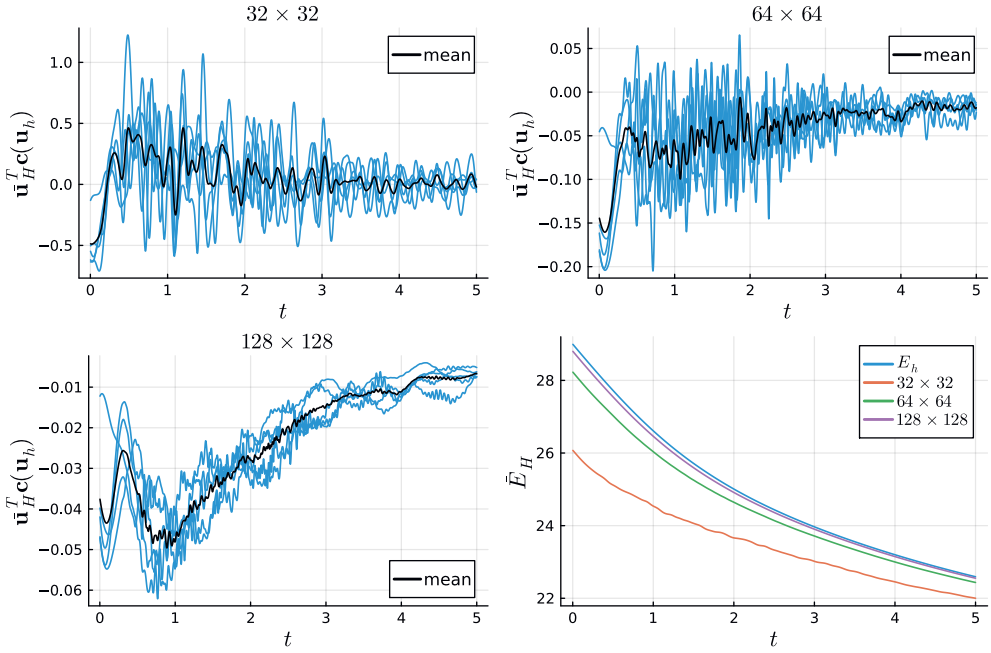


Figure 3.3: Resolved energy contributions for the true closure term, computed for five decaying turbulence simulations which constitute the training data for the machine learning closure models. The trajectories are presented for different levels of coarse-graining. The reference grid has a resolution of 2048 × 2048. See Section 3.5.1 for the exact simulation conditions. (Bottom-right) Resolved energy trajectories for one of the simulations.

after applying an explicit time integration scheme for i steps, each with a step size of $\overline{\Delta t}$, starting from the initial condition $\mathbf{W}\mathbf{u}_h$, incorporating the closure model. The corresponding DNS solution is denoted by $\mathcal{S}^{i(\overline{\Delta t}/\Delta t)}(\mathbf{u}_h)$, using a smaller step size Δt and initialized with $\mathbf{u}_h$. The ratio $\overline{\Delta t}/\Delta t$ arises because the coarse grid permits larger time steps [98]. Note that $\bar{\mathcal{S}}_\theta$ is the solver that works on the coarse grid and incorporates the closure model, while $\mathcal{S}$ is the DNS solver that works on the fine grid. The Adam optimization algorithm [41] will be used to minimize the loss. Further details on the training procedure are discussed in Section 3.5.1.

3.4 Methodology

As explained in Section 3.3, a main challenge in closure modeling is deriving an expression for $\tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta)$. In this section, we propose a skew-symmetric framework that results in a new expression for $\tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta)$, providing stability. In sections 3.4.1 and 3.4.2 we first introduce the Smagorinsky model and CNNs, respectively. These not only serve as something to compare our framework to, but also as necessary preliminaries. In Section 3.4.3 the new framework is derived.

3.4.1 Smagorinsky model

We start off with the Smagorinsky model, which is an eddy-viscosity model. As stated earlier, the main assumption in eddy-viscosity models is that this subgrid-scale stress tensor is proportional to the rate-of-strain tensor $\bar{\mathbf{S}} = \frac{1}{2}\left(\nabla\bar{\mathbf{u}} + (\nabla\bar{\mathbf{u}})^T\right)$,, where $\bar{\mathbf{u}} \in \mathbb{R}^2$ is a continuous representation of the resolved velocity field. Eddy-viscosity type closure models, for 2D turbulence, are of the following form:

$$\tilde{c}(\bar{\mathbf{u}}) = \nabla \cdot (\nu_t \bar{\mathbf{S}}), \tag{3.21}$$

where $\nu_t \geq 0$ is the eddy-viscosity. The Smagorinsky model assumes the following form for ν_t:

$$\nu_t = (C_s\Delta)^2\sqrt{2\mathrm{tr}(\bar{\mathbf{S}}^2)}, \tag{3.22}$$

where Δ is often chosen as the grid-spacing, i.e. $\Delta = \sqrt{h_x h_y}$. The model contains only a single parameter C_s, which can be tuned. In practice, we use a discretization of the Smagorinsky model. This yields a discrete closure model:

$$\tilde{\mathbf{c}}^{\mathrm{SMAG}}(\bar{\mathbf{u}}_H, C_s) = (C_s\Delta)^2\sqrt{2\mathrm{tr}(\bar{\mathbf{S}}_H^2)}\nabla_H \cdot \bar{\mathbf{S}}_H, \tag{3.23}$$

where the subscript H indicates a discretization of the derivative operators. We employ a central difference scheme for these derivatives. The global energy contribution, see (3.17), of the Smagorinsky model is always negative, i.e., it is strictly dissipative. In addition, the momentum conservation constraint (3.19) is satisfied for the Smagorinsky model.

3.4.2 Neural network closure

A straightforward machine learning approach is to use a CNN as a closure model, i.e.

$$\tilde{\mathbf{c}}^{\mathrm{CNN}}(\bar{\mathbf{u}}_H, \theta) = \mathrm{CNN}(\bar{\mathbf{u}}_H, \theta). \tag{3.24}$$

These models are well-suited to Cartesian grids [39, 40, 49], such as the one we employ here. In addition, they are translation equivariant. Here a CNN is used to map the filtered solution $\bar{\mathbf{u}}_H \in \mathbb{R}^{2\bar{N}}$ to $\tilde{\mathbf{c}} \in \mathbb{R}^{2\bar{N}}$ by chaining a series of convolutions with non-linear activation functions σ^n, where n indicates which layer of the CNN is considered. A single layer of such a network is represented as

$$\mathbf{z}^{n+1} = \sigma^n(\mathcal{A}^n \mathbf{z}^n + \mathbf{b}^n), \tag{3.25}$$

where each vector $\mathbf{z}^n$ contains a set of fields, typically referred to as 'channels' in CNN literature. The matrix $\mathcal{A}$ contains $s_{n+1} \times s_n$ submatrices encoding convolutional stencils, where s_n is the number of channels represented in $\mathbf{z}_n$. By choosing $s_{n+1} > s_n$, the data is effectively lifted to a higher-dimensional space. The vector $\mathbf{b}_n$ contains s_{n+1} bias channels, which are constant fields each determined by a single parameter. For example, a single convolution $\mathbf{A}$ (parameterized by weights α_{jk}) of a single channel $\mathbf{z}$, with bias vector $\mathbf{b}$ (parameterized by bias β), is represented as

$$(\mathbf{A}\mathbf{z} + \mathbf{b})_{ij} = \sum_{k,l=-r}^{r} (\alpha_{kl} z_{i+k,j+l}) + \beta \tag{3.26}$$

The double index notation is used to indicate the location on the 2D grid (one index for each spatial dimension), see Figure 3.1. Moreover, r represents the radius of the convolution in both spatial directions. On the edge of the domain, one uses padding of $\mathbf{z}$ to keep the size of the channels constant. In our case, we use circular padding to represent periodic boundary conditions (BCs). For intermediate layers, we use a ReLU activation function σ^n, and for the final layer, we take σ^n to be the identity [42]. Using a CNN as a closure model does allow for modeling backscatter as its energy contribution, see (3.17), can be both negative and positive. However, it violates momentum conservation, see (3.19).

A straightforward way to resolve the latter is to use a CNN to predict a stress tensor

$\boldsymbol{\tau}^{\mathrm{CNN}}$. The closure model is then obtained by taking the divergence of this tensor [14, 55]:

$$\tilde{\mathbf{c}}^{\mathrm{DIV}}(\bar{\mathbf{u}}_H, \theta) = \nabla_H \cdot \boldsymbol{\tau}^{\mathrm{CNN}}(\bar{\mathbf{u}}_H, \theta). \tag{3.27}$$

This ensures that momentum conservation, see (3.19), is satisfied. This formulation will be referred to as DIV. However, neither of these formulations is guaranteed to be stable and can therefore result in poor simulation results.

3.4.3 Skew-symmetric framework

For our novel closure modeling approach we build on earlier work which we presented in [15] (see Chapter 2). In this work a skew-symmetric neural network architecture was introduced and applied to 1D equations. Moreover, a coarse-grid representation of the subgrid-scale energy was included in the coarse-grained system. However, as stated earlier in Section 3.1, it is unclear how to extend this representation to multiple dimensions. On the other hand, applying the proposed neural network architecture, without this subgrid-scale representation, to 2D problem does not require any modifications to the architecture.

In this skew-symmetric framework, the closure model is represented as the sum of a skew-symmetric and negative-definite term:

$$\tilde{\mathbf{c}}^{\mathrm{SKEW}}(\bar{\mathbf{u}}_H, \theta) = (\mathcal{K} - \mathcal{K}^T)\bar{\mathbf{u}}_H - \mathcal{Q}^T\mathcal{Q}\bar{\mathbf{u}}_H, \tag{3.28}$$

where $\mathcal{K}(\bar{\mathbf{u}}_H, \theta), \mathcal{Q}(\bar{\mathbf{u}}_H, \theta) \in \mathbb{R}^{2\bar{N}\times 2\bar{N}}$ are build from CNN outputs. The different terms in this closure model are represented by matrix multiplications of the velocity vector. Model flexibility stems from the fact that these matrices depend nonlinearly on the solution vector via neural network outputs. The closure model is always dissipative:

$$\bar{\mathbf{u}}_H^T\tilde{\mathbf{c}}^{\mathrm{SKEW}}(\bar{\mathbf{u}}_H, \theta) = \bar{\mathbf{u}}_H^T(\mathcal{K} - \mathcal{K}^T)\bar{\mathbf{u}}_H - \bar{\mathbf{u}}_H^T\mathcal{Q}^T\mathcal{Q}\bar{\mathbf{u}}_H = -||\mathcal{Q}\bar{\mathbf{u}}_H||_2^2 \leq 0, \tag{3.29}$$

as the skew-symmetric contribution cancels. This means the closure model is guaranteed to be stable, unlike the previously presented machine learning approaches. Note that stability is only guaranteed up to a time discretization error. Analyzing stability while accounting for the time discretization error would require knowledge of the Jacobian of the system. This makes formal analysis rather cumbersome, due to non-linearity of the neural network [125]. Therefore, we consider this outside the scope of this research.

To ensure stability, ignoring the time discretization, the closure model does not allow for backscatter, but does increase modeling freedom beyond an eddy-viscosity basis through the skew-symmetric term. To see this, we note any real square matrix $\mathbf{A}$ can be decomposed

uniquely into the sum of a symmetric and a skew-symmetric matrix [126, 127]:

$$\mathbf{A} = \mathbf{S} + \mathbf{R}, \qquad \mathbf{S} = \tfrac{1}{2}(\mathbf{A} + \mathbf{A}^T), \quad \mathbf{R} = \tfrac{1}{2}(\mathbf{A} - \mathbf{A}^T). \tag{3.30}$$

By constraining the symmetric part $\mathbf{S}$ to be negative definite, we restrict ourselves to the space of stable linear operators, since for any nonzero $\mathbf{x}$,

$$\mathbf{x}^T\mathbf{S}\mathbf{x} < 0 \quad \Rightarrow \quad \mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{S}\mathbf{x} < 0, \tag{3.31}$$

implying strictly decreasing energy in all directions. Diagonalizing the symmetric operator $\mathbf{S}$ yields real, strictly negative eigenvalues and an orthogonal eigenbasis. Hence, the system's energy decays independently along these orthogonal modes. This negative-definite term can thus be interpreted as a generalized eddy-viscosity model that captures dissipative effects. However, a purely negative-definite formulation cannot represent conservative or rotational behaviors, since those correspond to energy-preserving dynamics for which $\mathbf{x}^T\mathbf{A}\mathbf{x} = 0$. The skew-symmetric component $\mathbf{R}$ addresses this limitation: it represents interactions that conserve energy, such as rotations, couplings, or advective terms in fluid dynamics [128]. By combining both components, we obtain the representation

$$\mathbf{A} = (\mathbf{K} - \mathbf{K}^T) - \mathbf{Q}^T\mathbf{Q}, \tag{3.32}$$

which spans the entire space of stable linear operators while remaining expressive enough to capture both dissipative and conservative effects. This serves as motivation for the proposed decomposition of the neural network architecture. This decomposition is not unique, since $\mathbf{A}$ remains the same if $\mathbf{K}$ is perturbed by an arbitrary symmetric matrix.

During our testing, we observed the effect of the skew-symmetric term to be much larger than that of the negative-definite term, see Figure 3.8. This further motivates its presence. From the computed energy contributions of the true closure term (see Figure 3.3), we believe backscatter is limited for reasonable coarse-graining factors. This means such a constrained closure model should be able to capture the energy behavior of the true closure term. In the upcoming sections, we will explain how the operators in this skew-symmetric architecture are built.

3.4.3.1 Skew-symmetric term

As described in [15] (see Chapter 2), $\mathcal{K}$ is constructed as follows:

$$\mathcal{K}(\bar{\mathbf{u}}_H, \theta) = \mathcal{B}_1^T\mathbf{k}(\bar{\mathbf{u}}_H, \theta)\mathcal{B}_2, \tag{3.33}$$

where $\mathbf{k}(\bar{\mathbf{u}}_H, \theta) = \text{diag}(\mathbf{k}_1, \mathbf{k}_2) \in \mathbb{R}^{2\bar{N}\times 2\bar{N}}$ is a matrix containing the neural network outputs $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{R}^{2\bar{N}}$ on the diagonal. Here we choose to represent each matrix in the decomposition to be square. However, one is free to choose the dimensions of the matrices, as long as the $\mathcal{K}$ is square. Also, the sparsity of the matrices can be varied. In our case, the underlying neural network architecture is a CNN, as we employ a uniform grid. This means a CNN is used to map $\bar{\mathbf{u}}_H$ to output channels $\mathbf{k}_1$ and $\mathbf{k}_2$. The matrices $\mathcal{B}$ are linear convolutional layers mapping from two input channels to two output channels, similarly to the convolutional layers introduced in Section 3.4.2. The $\mathcal{B}$ matrices thus contain four submatrices encoding convolutions. A clear motivation for this specific decomposition of $\mathcal{K}$, besides sparsity and computational efficiency, is outlined in Appendix B.3. The main idea is that the proposed decomposition resembles a simple advection operator for specific choices of $\mathcal{B}$ matrices, where the matrix $\mathbf{k}$ supplies enough degrees of freedom to freely traverse the energy-conserving solution space, by locally controlling the advection in each direction. Another option would be to use more output channels, i.e., $\mathbf{k}_1, \ldots, \mathbf{k}_n$ and construct a larger matrix $\mathbf{k} = \text{diag}(\mathbf{k}_1, \ldots, \mathbf{k}_n)$, where n would be a hyperparameter. This would require the $\mathcal{B}$ matrices to be non-square and contain more submatrices encoding different convolutions. Such an extended architecture could possibly lead to faster convergence of the training procedure and more expressive power. In addition, a smaller $\mathbf{k}$ matrix, i.e. $\mathbf{k} \in \mathbb{R}^{c\times c}$ with $c << \bar{N}$, could also be considered. This would effectively apply the non-linear neural network operations in a reduced latent space, encoded by the $\mathcal{B}$ matrices [65]. We consider both options outside the scope of the current work, but view them as interesting future research directions.

To satisfy momentum conservation, see (3.19), we require the sum of the convolution weights in the submatrices to be zero such that both $\mathcal{B}$ and $\mathcal{B}^T$ are in the nullspace of $\mathbb{1}_H$. To achieve this, we let the weights $\bar{b}_{kl}$ of such a convolution depend on a set of parameters b_{kl}:

$$\bar{b}_{kl} = b_{kl} - \frac{1}{(2r+1)^2} \sum_{k,l=-r}^{r} b_{kl}, \tag{3.34}$$

such that $\sum_{k,l=-r}^{r} \bar{b}_{kl} = 0$ holds. To extend the architecture to unstructured grids, the convolutions in the CNN and in the $\mathcal{B}$ matrices can simply be replaced by graph convolutions.

3.4.3.2 Negative-definite term

As described in [15], $\mathcal{Q}$ is constructed as follows

$$\mathcal{Q}(\bar{\mathbf{u}}_H, \theta) = \mathbf{q}(\bar{\mathbf{u}}_H, \theta)\mathcal{B}_3, \tag{3.35}$$

where $\mathbf{q}(\bar{\mathbf{u}}_H, \theta) = \text{diag}(\mathbf{q}_1, \mathbf{q}_2)$ is also constructed from outputs of the CNN $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^{2\bar{N}}$. This means the CNN has in total four output channels to build the fields $\mathbf{k}_1, \mathbf{k}_2, \mathbf{q}_1, \mathbf{q}_2$.

Table 3.1: Overview of the different closure models and their properties when combined with the coarse discretization. These closure models consist of the Smagorinsky model (Chapter 3) (SMAG), a CNN, the divergence of a CNN (DIV), our skew-symmetric neural network architecture (SKEW), and no closure (NC).

	SMAG	**CNN**	**DIV**	**SKEW** (ours)	**NC**
Mass conservation	✓	✓	✓	✓	✓
Momentum conservation	✓	✗	✓	✓	✓
Dissipative	✓	✗	✗	✓	✓

The parameters of the model include both the CNN weights and the parameters of the $\mathcal{B}$ matrices. As these are all sparse operations, the model remains computationally efficient. Because the model only contains convolutions, it is translation equivariant. Furthermore, it can be safely applied to larger grids because it uses only local information. Similarly to the skew-symmetric term, this decomposition could be adjusted by extending the matrix containing the neural network outputs to multiple channels, i.e. $\mathbf{q} = \text{diag}(\mathbf{q}_1, \ldots, \mathbf{q}_n)$ A more in-depth motivation behind the proposed neural network architecture is presented in Appendix B.3, where we relate the choice of $n = d$, where d is the number of spatial dimensions, to a diffusion operator. In this way, the diagonal elements of $\mathbf{q}$ locally control the diffusivity.

3.4.4 Overview of the closure models

In this section, we introduced a set of four closure models, namely the standard Smagorinsky model (Chapter 3) (SMAG), a CNN, using a CNN to predict a stress tensor and then taking the divergence (DIV), and finally our skew-symmetric neural network architecture (SKEW). In addition, we also compare to no closure (NC), i.e. $\tilde{\mathbf{c}} = \mathbf{0}_H$. An overview of the closure models and their properties is depicted in Table 3.1.

3.5 Results

3.5.1 Experimental setup

To evaluate the closure models,, we consider two test cases: 2D decaying turbulence and Kolmogorov flow. The test cases are inspired by those considered in [47]. For both test cases we consider a periodic domain of $\Omega = [-\pi, \pi] \times [-\pi, \pi]$ and a viscosity of $\nu = \frac{1}{1000}$. Each velocity component of the flow is initialized by a random initial condition only containing

energy in the low wavenumbers (below $\kappa_{\max} = 10$):

$$\mathbf{u}(\mathbf{x}, 0) = \mathrm{Re}\left(\begin{bmatrix}\sum_{\{\boldsymbol{\kappa}\in\mathbb{Z}^2|0<||\boldsymbol{\kappa}||_2<\kappa_{\max}\}} c^u_{\boldsymbol{\kappa}} e^{i\boldsymbol{\kappa}\cdot\mathbf{x}} \\ \sum_{\{\boldsymbol{\kappa}\in\mathbb{Z}^2|0<||\boldsymbol{\kappa}||_2<\kappa_{\max}\}} c^v_{\boldsymbol{\kappa}} e^{i\boldsymbol{\kappa}\cdot\mathbf{x}}\end{bmatrix}\right), \tag{3.36}$$

where the real and complex components of the coefficients $c^u_{\boldsymbol{\kappa}}, c^v_{\boldsymbol{\kappa}} \in \mathbb{C}$ are sampled uniformly from the interval $(-1, 1)$. Finally, the coefficients are scaled such that the normalized kinetic energy at $t = 0$, namely

$$\frac{1}{2|\Omega|}\int_\Omega ||\mathbf{u}(\mathbf{x}, 0)||_2^2 \,\mathrm{d}\Omega,$$

equals 1.2. Increasing this initial energy amplifies the characteristic velocity scales of the flow, thereby increasing the effective Reynolds number $\mathrm{Re}_{\text{eff}} = UL/\nu$, where U is a representative velocity magnitude and L a characteristic length scale of the largest eddies. A higher Re_{eff} leads to a broader separation of scales and stronger non-linear interactions, which make closure modeling more challenging. The chosen value of 1.2 was found, through preliminary numerical tests, to yield sufficiently rich turbulent dynamics while remaining numerically stable and allowing meaningful comparison between closure models. Before the simulation starts, the sampled initial condition is projected onto a divergence-free basis.

The training data set consists of five simulations starting from an initial condition sampled in this manner. For the DNS we use a computational grid of resolution 2048×2048 and a time step size $\Delta t = 2 \times 10^{-4}$, as in [47]. For the time integration of both the DNS and the closure model-based simulations, we use a 4th-order Runge-Kutta integration scheme [96, 129]. For training purposes, we save a snapshot every 10 time steps until $t = 5$. Regarding coarse-graining, we consider three different resolutions, namely 32×32, 64×64, and 128×128, and a time step size of $\overline{\Delta t} = 10\Delta t$, as coarser grids allow for larger time steps [98, 130]. For each coarse-graining factor, the machine learning models are trained to reproduce the true solution, i.e., minimize (3.20), for $n = 5$ time steps. To optimize the closure model parameters, we use the ADAM optimization algorithm [41] with a learning rate of 10^{-3}, decay rates for the first and second momentum estimates at 0.9 and 0.999, respectively, and a mini-batch size of 20. We optimize for in total 500 epochs. These hyperparameter settings resulted in smooth convergence, see Appendix B.4. We consider further hyperparameter studies to be outside the scope of this research, as this research focuses on the neural network architecture. We implemented the closure models in the Julia programming language [99] using the Flux.jl package [100, 101]. For each of the CNN-based closure we use four input channels, namely $\bar{\mathbf{u}}_H$ and $\mathbf{m}_H(\bar{\mathbf{u}}_H)$, each containing two channels. The intermediate layers of the CNNs each contain 32 channels, with a total of four intermediate layers (same amount as in [40]). The convolutions in each layer have a radius of $r = 2$. The number of parameters is dominated by the number of hidden layers and channels, which is the same for each of the neural network-based closure models. In this way, the comparison is carried out as fairly as

possible. In between the hidden layers, we employ a ReLU activation function and a linear activation function at the final layer [42]. The purely CNN-based closure simply has two output channels to model the closure term in each spatial direction. For DIV we have three output channels, two for the diagonal elements of the stress tensor and one for the off-diagonal ones, due to the symmetry of the true stress tensor [2, 6]. For SKEW the underlying CNN has four output channels corresponding to $\mathbf{k}_1, \mathbf{k}_2, \mathbf{q}_1, \mathbf{q}_2$. The convolutions in the $\mathcal{B}$ matrices are chosen to have a convolution radius of $r = 2$. Training of a single neural network takes roughly two hours on an A100 GPU of the Dutch National supercomputer Snellius [131]. Regarding the optimization of the Smagorinsky constant, we choose the constant value that minimizes the L_2-norm of the energy spectra for the training data at $t = 2$ in $\log_{10}$ space. The energy spectra are determined by computing a fast Fourier transform (FFT) of the velocity field. The wavenumbers are then divided into dyadic bins, and the energy belonging to the wavenumbers in the bin is summed to produce the spectrum. This procedure is described in [40, 132]. The considered values for the Smagorinsky constant are 0.0 to 0.30 in intervals of 0.01. The obtained optimal values for C_s at each resolution are 0.23, 0.22, and 0.18 for 32×32, 64×64, and 128×128, respectively. These values are within the range of what is typically used for 2D turbulence, namely around 0.18 [133, 134].

3.5.2 Decaying turbulence

The first test case we consider is decaying turbulence, i.e., there is no forcing. Here we consider a single simulation starting from a randomly generated initial condition, generated according to (3.36). The simulations are carried out up to the final time $t = 10$. Note this is twice as long as present in the training data. Therefore, the second half of the simulation corresponds to extrapolation in time. For each coarse-graining factor, the resolved energy and error trajectories are presented in Figure 3.4. The error is defined as

$$\text{error} := \sqrt{\frac{||\bar{\mathbf{u}}_H - \bar{\mathbf{u}}_H^{\text{model}}||_2^2}{||\bar{\mathbf{u}}_H||_2^2}}, \tag{3.37}$$

where $\bar{\mathbf{u}}_H$ is the FDNS result and $\bar{\mathbf{u}}_H^{\text{model}}$ is the model prediction. We observe that initially the unconstrained machine learning approaches (CNN and DIV) produce the best energy trajectories for resolutions 64×64 and 128×128, whereas SKEW is too dissipative and the Smagorinsky model is even more dissipative. However, as the simulation progresses, the unconstrained machine learning methods often suddenly become unstable. For a resolution of 32×32, these approaches tend to diverge rather quickly, whereas finer resolutions can stay stable over a more extended time period. That said, this seems merely a postponement of the inevitable, as unconstrained machine learning acting on a finer resolution remains highly prone to a sudden and unpredictable catastrophic failure. One can hope for a one-off stable

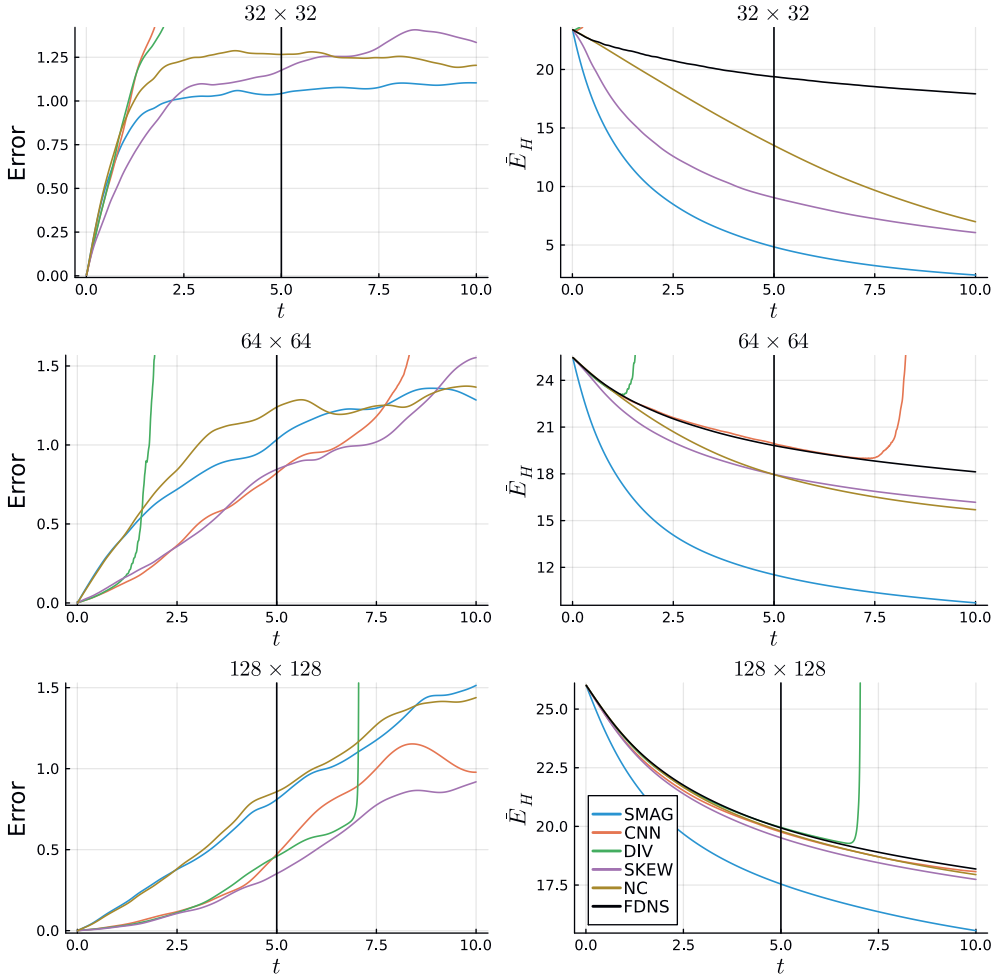


Figure 3.4: (Left) Error for each coarse-graining factor as a function of time for the decaying turbulence test case. (Right) Resolved energy trajectories for each coarse-graining factor. For an overview of the methods, see Table 3.1. The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time.

simulation, as shown by the 128×128 resolution CNN. This particular simulation remained stable during the entire considered time period, producing a good energy trajectory and an improved error with respect to NC and SMAG. However, in the vorticity fields presented in Figure 3.5, we already notice a build-up of numerical noise for the CNN at the end of the simulation, which may well result in instabilities in the future. In fact, in Section 3.5.3 we show that this is indeed the expected outcome. As such, the purely data-driven, physics-blind machine learning methods considered here are unsuitable as a viable closure modeling approach for long-term predictions, incapable of outperforming existing physics-based closure models.

On the other hand, for our constrained approach SKEW, we find it consistently provides improved error trajectories with respect to NC, except at a resolution of 32×32. However, it is too dissipative, as stated earlier. This likely comes from the fact that it is constrained to not create energy. This means energy is solely redistributed through the skew-symmetric term and dissipated through the negative-definite term. Especially at large coarse-graining factors, this becomes more problematic, see resolution of 32×32. This is likely caused by the fact that backscatter is more prevalent, see Figure 3.3, as more information is being discarded to the subgrid scales. At a resolution of 64×64 SKEW is also too dissipative, however, it does produce an improved error trajectory with respect to the other closures, even in the extrapolation region $t \in (5, 10]$. However, near the end of the simulation, the error becomes larger than for NC and SMAG. Later in this section, we consider the energy spectra to compare the velocity fields in a more statistical fashion, see Figure 3.6 and Figure 3.7.

Carrying out the DNS simulation took roughly 45 minutes on an A100 GPU, whereas the closure model-based simulations took between 3 and 4 minutes (with a training time of roughly two hours), for all closure models, see Table 3.2. This amounts to a computational speed-up of more than $10\times$ with respect to the DNS. For these coarse resolutions, the evaluation time seems to be dominated by computational overhead, as we did not observe a big difference in computation time between the different closure models and resolutions. All closure model-based simulations require more computation time than NC, where SKEW consistently takes the most time. This is likely caused by the additional convolutions in the $\mathcal{B}$ matrices. Next, we look at the scalar vorticity field $\omega = (\nabla \times \mathbf{u})_3$, at different points in time, produced by the closure models. For a resolution of 64×64, these are depicted in Figure 3.5. The other resolutions are depicted in Appendix B.5. For the remainder of this text, we will stick to a resolution of 64×64, as we believe this is a nice middle ground between the poor results obtained for 32×32 and almost perfect reproduction for 128×128. This coincides with the coarse-graining factor applied in [47]. Regarding the vorticity fields, we find that initially all three machine learning closures nicely match the FDNS result. However, as the simulation progresses, we observe a build-up of numerical noise for the unconstrained

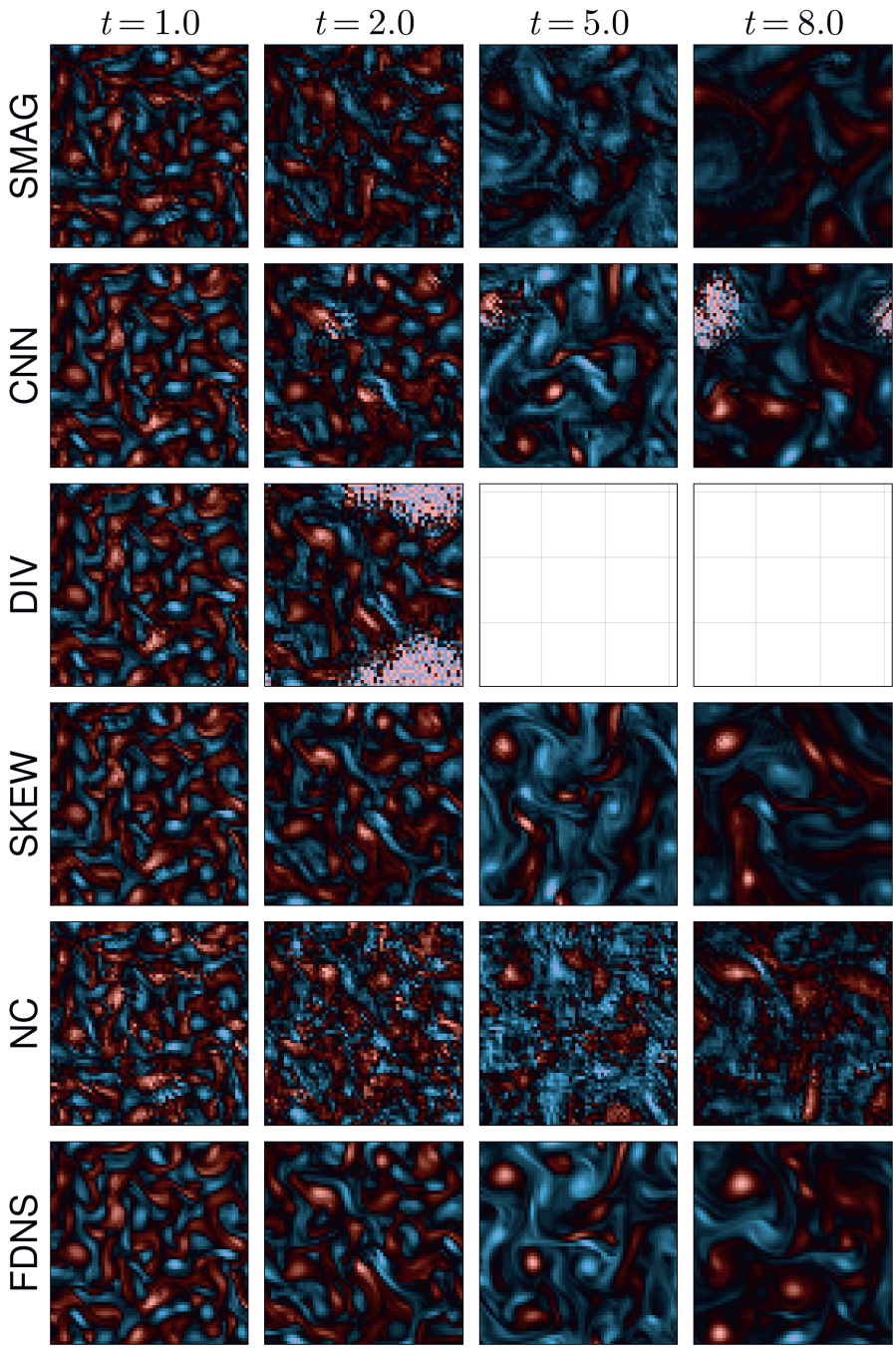


Figure 3.5: Vorticity fields at each point in time for each of the closure models on a 64×64 grid. Simulations correspond to the decaying turbulence test case. Blank boxes indicate an unstable simulation.

Table 3.2: Computation time in seconds for the different closure models for the decaying turbulence test case. For an overview of the methods, see Table 3.1.

$\bar{N}$	**SMAG**	**CNN**	**DIV**	**SKEW**	**NC**	**DNS**
32×32	182.69	186.64	189.99	211.39	170.42	2559.88
64×64	210.46	216.26	203.76	225.79	165.27	2559.88
128×128	195.29	200.06	217.75	243.32	187.82	2559.88

machine learning closures. This eventually leads to unstable simulations. For SKEW this does not happen. Even after diverging from the FDNS, it still produces smooth results which seem to match the FDNS on a qualitative level. The instabilities in the unconstrained machine learning approaches can possibly be alleviated by adding more training data or adding noise to the training [12, 51]. However, we argue that given the same amount of training data, using SKEW has clear stability benefits.

To assess the physical consistency of the produced trajectories, we compare the energy spectra at different points during the simulation.This is done, as a pointwise error, such as presented in Figure 3.4, no longer provides relevant information after the solutions diverge from the FDNS [47]. The spectra are depicted in Figure 3.6. Here we observe that SMAG is too dissipative at the large scales, but nicely reproduces the κ^{-3} decay of the energy spectrum expected from 2D turbulence [135]. Regarding SKEW we find it has a better overall fit with the FDNS, as compared to SMAG, for both the high and low wavenumbers. Regarding the unconstrained machine learning approaches, we observe a clear buildup in energy in the large wavenumbers. From the depicted vorticity fields, we can clearly see the numerical noise responsible for this.

In Figure 3.4, we observed that SKEW reaches a larger error than NC and SMAG at the end of the simulations. To assess whether or not the simulation produced by SKEW is still physically consistent, we consider the energy spectrum at this point. This is depicted in Figure 3.7. Here we observe that even though the error, see (3.37), is larger, SKEW still produces an energy spectrum that more closely matches the FDNS spectrum, as compared to SMAG and NC. It also produces the expected κ^{-3} slope. SMAG also achieves this, but underestimates the energy in all wavenumbers, whereas NC suffers from a build-up of energy in the large wavenumbers. The latter likely corresponds to numerical noise, due to the coarseness of the grid. After a while SKEW diverges from the FDNS. This behavior is expected given the characteristic sensitivity on initial conditions of the two-dimensional Navier–Stokes equations in the turbulent regime [136]. Nevertheless, the SKEW model continues to produce physically consistent statistics and spectra, even when extrapolating in time.

Finally, we consider the contributions of both the skew-symmetric and negative-definite

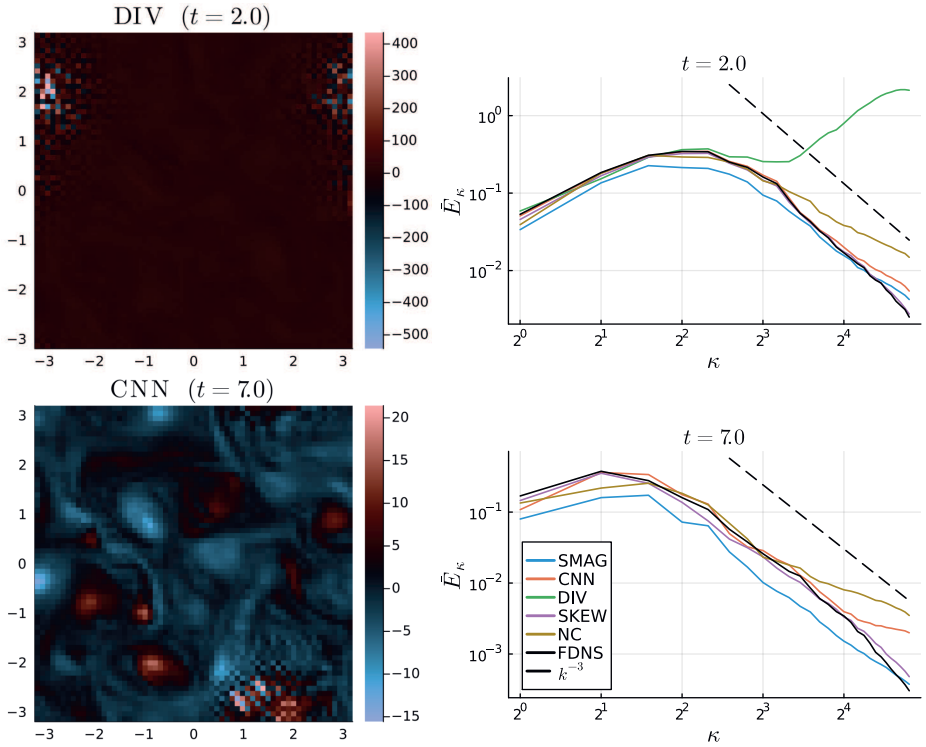


Figure 3.6: (Left) Two snapshots where we see numerical oscillations occurring for DIV and CNN. The oscillations eventually result in instabilities. (Right) Energy spectra at the time of these snapshots. The numerical oscillations cause a clear increase in energy in the high wavenumbers.

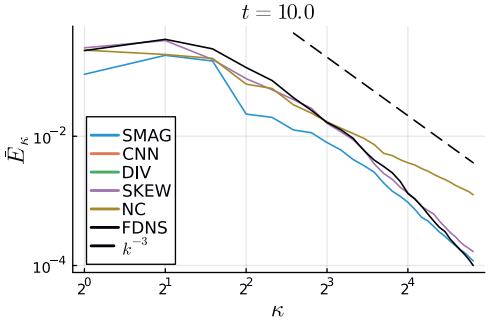


Figure 3.7: Energy spectra at the end of the decaying turbulence simulation at $t = 10$. Both the CNN and DIV have become unstable at this point and are therefore omitted from the figure.

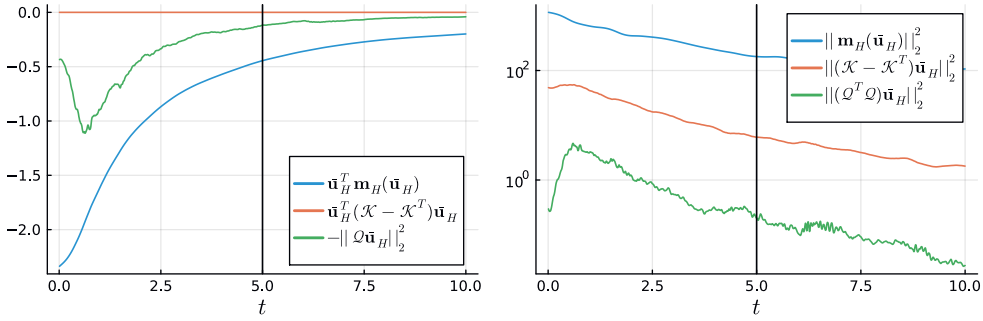


Figure 3.8: (Left) Energy contribution for each of the terms in the SKEW architecture along with the contribution of the coarse discretization. (Right) Magnitude of each of the terms. The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time.

term in the SKEW architecture. We consider both the energy contribution and the magnitude of the terms. This is depicted in Figure 3.8. The first observation is that the energy contribution of the skew-symmetric term is indeed zero, as it should be. Furthermore, we find that the negative-definite term is slightly less dissipative than the coarse discretization. The trajectory also has a different shape; the dissipation coming from the negative-definite term peaks around $t = 1$, whereas the dissipation from the coarse discretization decreases smoothly over time. The shape of the dissipation trajectory of the negative-definite term might arise from the transition of the flow from one physical regime to another. More specifically, the initial condition contains energy only at low wavenumbers. This means the flow undergoes a transient period before the formation of its characteristic slope, see Figure 3.6. Next, we look at the magnitude of the different terms. Here we find that the skew-symmetric term has a larger magnitude than the negative-definite term. This means it has a more significant impact on the simulation. This supports the use of a skew-symmetric term in the closure model.

To conclude this section, we also examine the effect of omitting either the skew-symmetric term or the negative-definite term from the SKEW architecture. To this end, we perform the decaying turbulence simulation with one of the two terms set to zero. The resulting resolved energy trajectories and energy spectra are shown in Figure 3.9. We observe that omitting the negative-definite term leads to markedly less dissipative behavior, as expected. The resulting energy trajectory is nearly identical to that of NC. The corresponding energy spectrum matches the FDNS result well at low wavenumbers; however, at high wavenumbers, a clear build-up of energy appears, resembling the NC spectrum and indicating numerical noise. Conversely, omitting the skew-symmetric term produces more dissipative behavior and yields physically consistent energy levels at high wavenumbers, with the characteristic

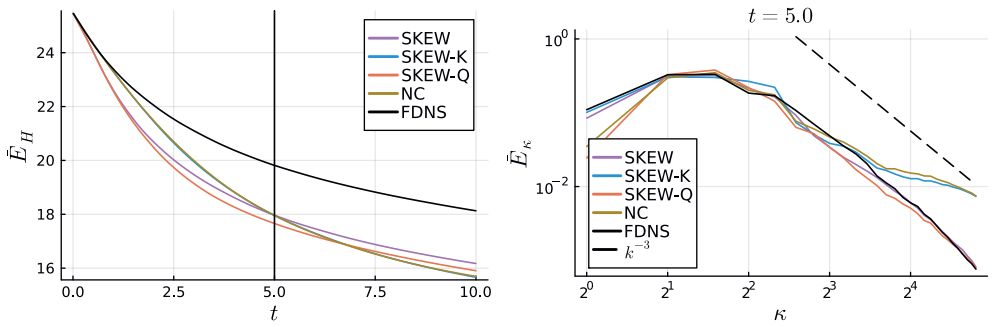


Figure 3.9: (Left) Resolved energy trajectories for the full SKEW closure model, SKEW-K (skew-symmetric term only), and SKEW-Q (dissipative term only). The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time. (Right) Energy spectra halfway through the decaying turbulence simulation at $t = 5$.

κ^{-3} slope accurately recovered. Nonetheless, this variant over-dissipates energy at the low wavenumbers.

From these simulations, we conclude that the skew-symmetric term is essential for accurately capturing the energy at low wavenumbers, whereas the negative-definite term plays a crucial role in suppressing numerical noise and recovering the characteristic κ^{-3} slope. This means both terms make significant and complementary contributions to the accuracy and physical consistency of the simulation.

3.5.3 Consistency of closure model performance

Training neural networks is inherently random, due to the selection of mini-batches, initialization of the weights, etc. This is why we instantiate multiple replicas of each network to fully assess their potential as a closure model. For this purpose, we train an ensemble of five replicas for each neural network architecture. This allows us to evaluate the consistency of the training procedure in terms of producing good closure models. Before evaluating the networks, we ensured no convergence issues occurred during training. The resulting error and energy trajectories, for each neural network, evaluated on the decaying turbulence test case, are depicted in Figure 3.10. Regarding the plain CNN architecture, we observe that two out of five networks result in unstable simulations, and for DIV, three out of five. Hence, it is worth pointing out that, therefore, simply retraining exactly the same machine-learning architecture can be the difference between a stable simulation and a numerical failure, highlighting the fickle nature of these methods. For SKEW, no ensemble members became unstable.

Regarding the error trajectories, we observe similar performances for all the networks that remained stable, whereas the energy trajectories were best reproduced by some of the unconstrained machine learning closures. However, to evaluate the build-up of numerical noise

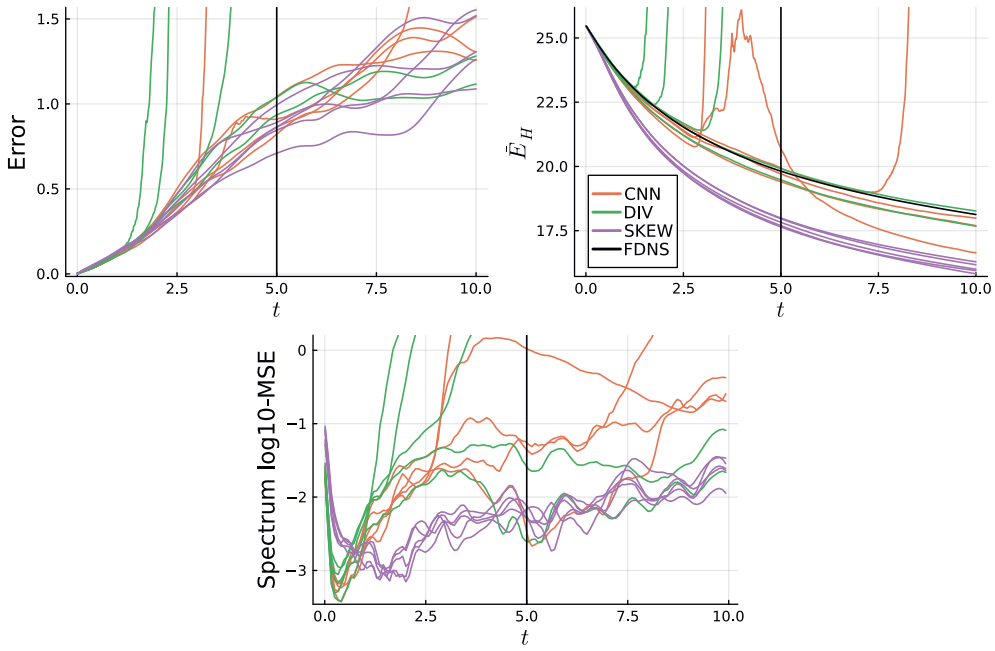


Figure 3.10: (Top-left) Error over time for each closure model in the ensemble of five. (Top-right) Resolved energy trajectories for each closure model in the ensemble of five. (Bottom) Error of the energy spectrum over time calculated by computing the spectrum in $\log_{10}$ space, then computing the mean squared error (MSE) with respect to the FDNS spectrum, and finally reporting the $\log_{10}$ of this value. These trajectories are also depicted for each closure model in the ensemble of five. The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time.

and physical consistency, we computed the error in energy spectrum during the simulation. This is also depicted in Figure 3.10. Here we find that the SKEW architecture consistently outperforms the other architectures. From this, we conclude that our SKEW neural network architecture is not only guaranteed to be stable, but also consistently produces physical results, without a build-up of numerical noise.

3.5.4 Kolmogorov flow

Next, we aim to evaluate the long-term performance of the closure models and their extrapolation capabilities. To do this, we require a different test case, as decaying turbulence from the previous test case eventually decays to zero. We therefore consider Kolmogorov flow, with the same viscosity $\nu = \frac{1}{1000}$ and periodic domain $\Omega = [-\pi, \pi] \times [-\pi, \pi]$. Kolmogorov flow is characterized by the following forcing:

$$\mathbf{f}(\mathbf{u}, \mathbf{x}, t) = \begin{bmatrix} \sin(4y) \\ 0 \end{bmatrix} - 0.1\mathbf{u}, \tag{3.38}$$

and is often used to evaluate machine learning closure models [40, 47, 49]. The machine learning models are not retrained for this test case. This means the models will have to extrapolate from the decaying turbulence training data to a test case that includes forcing. To initialize the simulation, we first carry out a DNS on a 2048×2048 grid until $t = 25$, starting from initial condition (3.36). This serves as a warm-up of the system, such that it reaches a statistical equilibrium. The final velocity field then serves as an initial condition to evaluate the closure models. We then simulate for 500 model time units starting from this initial condition. This is a significant extrapolation, as the training data was for the interval $t \in [0, 5]$ and for a test case without forcing. In addition to the previously introduced closure models, we apply backscatter clipping to the trained CNN to obtain a stable closure model by removing backscatter. This is done by projecting the CNN output on an eddy-viscosity model. From this, we obtain an eddy-viscosity value $\nu_t^{\text{clipping}}(\mathbf{x})$ such that the CNN output is matched as close as possible in the L_2-norm. Negative values for $\nu_t^{\text{clipping}}(\mathbf{x})$ are then set to zero to provide stability. The clipping procedure is described in [38]. We will refer to this closure model with the acronym CNN-C. Vorticity snapshots from the simulations are depicted in Figure 3.11. Snapshots for CNN and DIV are not depicted, as these simulations become unstable quickly after their initialization, see Figure 3.12. For NC we observe a lot of numerical noise, whereas in SMAG (to a lesser extent), SKEW, and CNN-C this is smoothed out. Regarding the snapshot at $t = 5$, we find the filtered DNS results are most closely matched by SKEW.

To make a more thorough comparison, we consider the resolved energy trajectories, along with an average energy spectrum for the simulation, see Figure 3.12. The first thing we

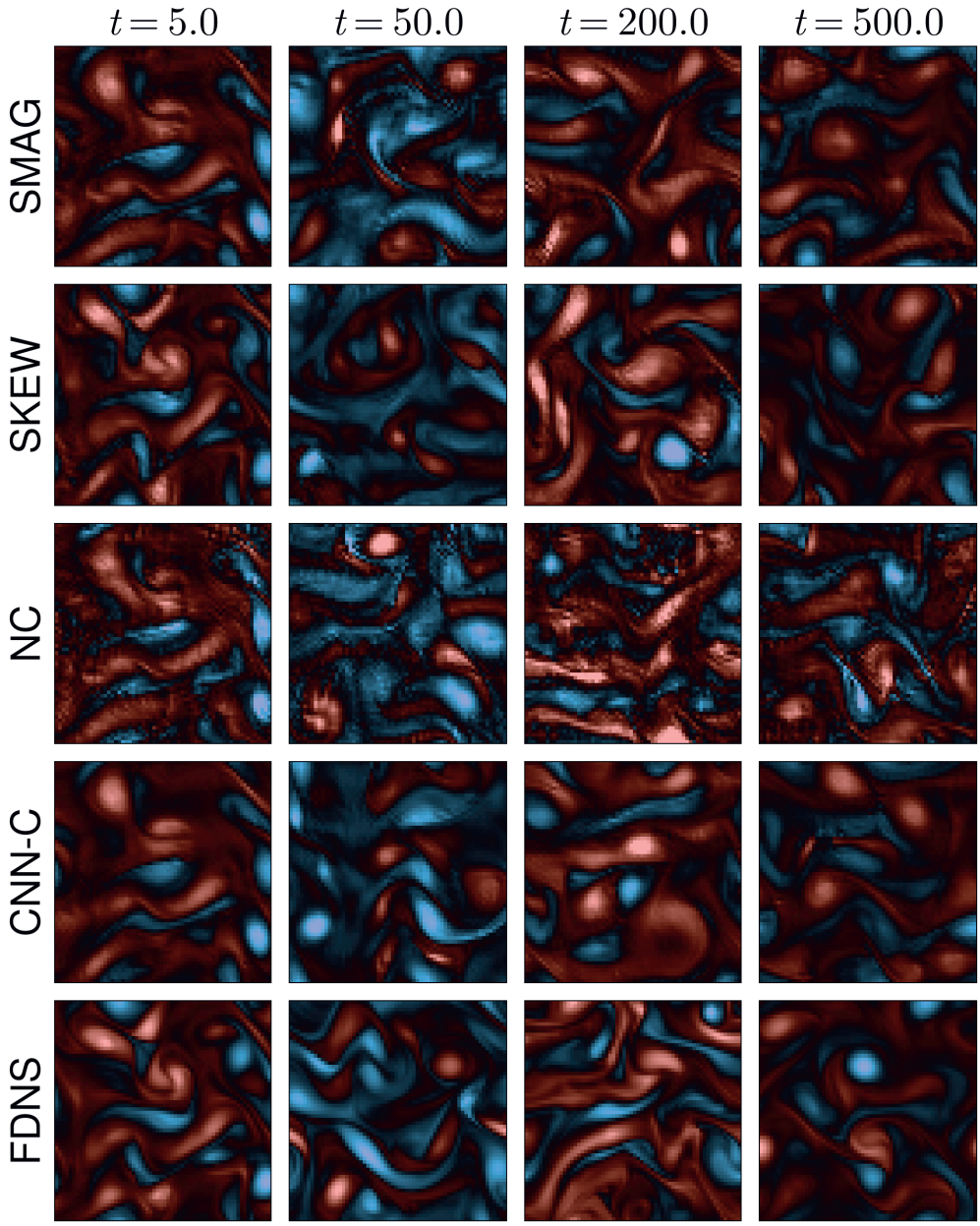


Figure 3.11: Vorticity fields at each point in time for each of the closure models on a 64×64 grid. Simulations correspond to the Kolmogorov flow test case.

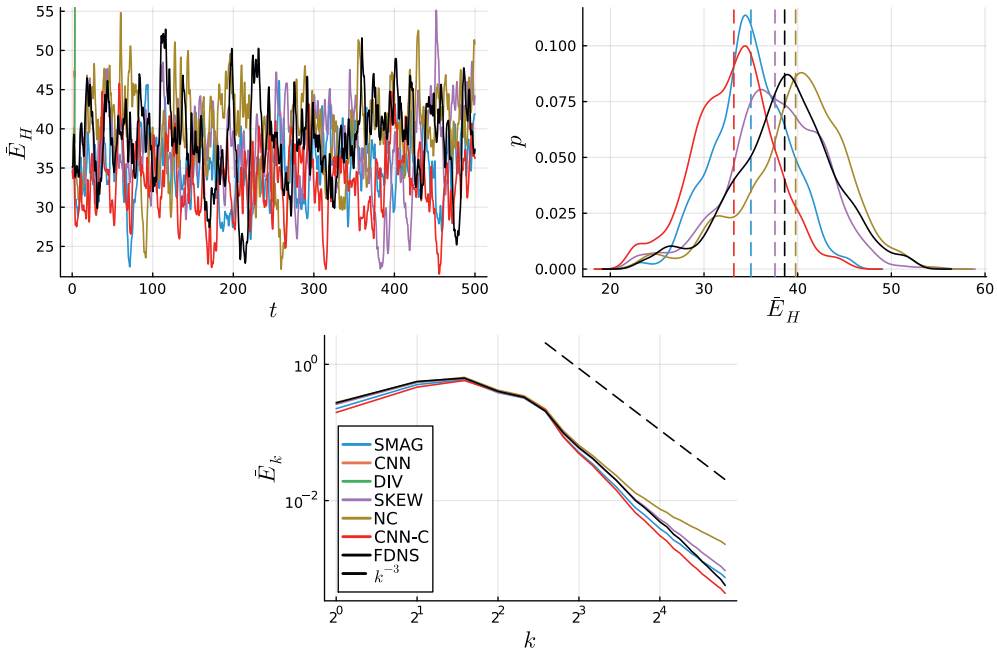


Figure 3.12: (Top-left) Resolved energy trajectories for each of the closure models in the Kolmogorov flow test case. (Top-left) Corresponding distributions of the resolved energy for the entire simulation. Dashed vertical lines correspond to the mean. (Bottom) Energy spectra were obtained by first computing the energy spectrum for each snapshot and then computing the average value for each wavenumber. Both the CNN and DIV closure models resulted in unstable simulations, so their spectrum is omitted from the figures.

observe from the resolved energy trajectories is that the unconstrained machine learning approaches both become unstable at the start of the simulation. The remaining closure models remain stable, as they are strictly dissipative. To make a statistical comparison of the resulting flow fields, we consider the distribution of the resolved energy for the entire simulation and the average energy spectra. Here we find that SMAG and CNN-C are too dissipative, as the distributions are shifted to the left with respect to FDNS. For CNN-C, this phenomenon is reported in [51]. Both NC and SKEW seem to give a good prediction of both the mean and shape of the distribution.

Looking at the energy spectra, we find that NC indeed produces numerical noise, looking at the energy in the large wavenumbers. In addition, we find that SKEW performs the best in the low and intermediate wavenumbers, while SMAG performs the best in the high wavenumbers. Overall, we find that for this extrapolation test case SKEW performs, at worst, as well as SMAG. From this, we conclude that SKEW is both stable and accurate, and is capable of extrapolating to different test cases, without being retrained.

3.5.5 Comparison to the dynamic Smagorinsky model

To provide a comparison between our SKEW architecture and a more robust physics-based closure model, we consider the dynamic Smagorinsky model. The formulation and underlying theory of this model are described in [32]. The main idea is that the Smagorinsky constant in (3.22) is determined dynamically and depends on space, time, and the resolved velocity field, i.e. $C_s = C_s(\mathbf{x}, t, \bar{\mathbf{u}})$.

The dynamic procedure builds on the assumption that the subgrid-scale stresses are self-similar across filter scales and that the same eddy-viscosity ansatz holds at both the grid filter and a larger test filter. By enforcing consistency between these two levels, the model computes C_s from the resolved flow itself rather than prescribing it a priori. This allows the model to reduce dissipation in laminar or well-resolved regions while increasing it where subgrid activity is strong. Its main strength is therefore that it provides dissipation only where it is needed, making it significantly less dissipative than the standard Smagorinsky model.

The resulting energy trajectory and energy spectrum for the dynamic Smagorinsky model in the decaying turbulence test case are shown in Figure 3.13, together with the results from our SKEW architecture.

For the energy trajectory, we find that the dynamic Smagorinsky model is indeed less dissipative than the standard Smagorinsky model (Chapter 3) (SMAG). However, it remains slightly more dissipative than our SKEW architecture, which matches the FDNS result the closest. Regarding the energy spectrum, the dynamic Smagorinsky model provides a substantial improvement over SMAG and NC. Nonetheless, it still underpredicts energy

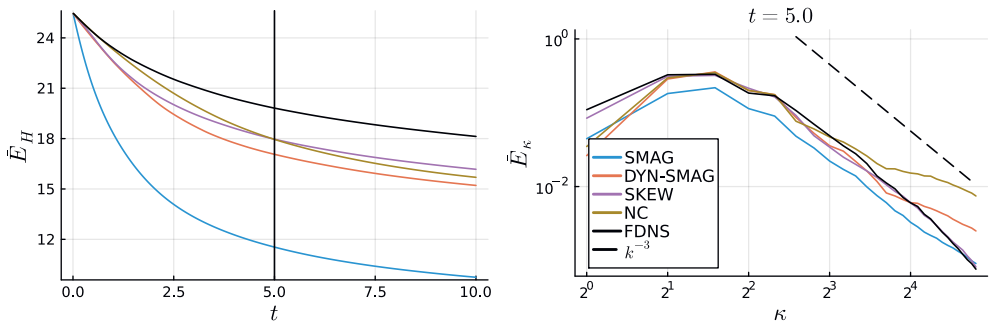


Figure 3.13: (Left) Resolved energy trajectories for the decaying turbulence test case, where DYN-SMAG corresponds to the dynamic Smagorinsky model. The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time. (Right) Energy spectra halfway through the decaying turbulence simulation at $t = 5$.

at low wavenumbers and overpredicts energy at high wavenumbers, whereas our SKEW architecture does not exhibit these deficiencies.

3.5.6 Limitations of Machine Learning Closure Models

Note that a neural network is not strictly necessary to enforce a skew-symmetric framework. Traditional calibration or data assimilation approaches could be used, which typically assume a fixed functional form and tune a limited set of coefficients to DNS data. In contrast, a neural network provides a flexible functional representation capable of capturing complex, non-linear dependencies of the closure term on local flow features, dependencies that are difficult to express or calibrate explicitly using fixed parametric forms. While the skew-symmetric structure enforces energy conservation, the entries of the skew-symmetric matrix (as well as those of the dissipative term) are learned functions of the flow state, allowing the model to adapt dynamically to local flow conditions rather than relying on globally calibrated coefficients.

Finally, it should be noted that fundamental challenges exist when employing machine learning within the context of LES (or any physics-based simulation) [137]. Neural networks easily have millions (or more) tunable parameters with no physical meaning, are data-hungry, and act as black boxes, making them more complex than their physics-based counterparts. They are also prone to instabilities over long integration times, an issue that we have explicitly addressed in the present manuscript. Our proposed framework, therefore, represents a step towards embedding physical constraints (in this case, guaranteed energy conservation) directly into the architecture of a data-driven model. This combination of physical structure and data-driven flexibility embodies a key advantage of physics-informed machine learning: it ensures physical consistency while retaining the expressive power of modern machine-learning

methods. That said, many of the aforementioned issues remain and require further study.

3.6 Conclusion

In this work, we started off by exploring the conservation laws inherent in the incompressible Navier-Stokes equations, specifically, mass, momentum, and energy conservation. We employed a discretization that preserves these laws in a discrete sense. To coarse-grain the simulation, we used a face-averaging filter, ensuring that the resulting coarse-grained velocity field continues to satisfy mass conservation. We then examined different approaches to modeling the commutator error introduced by coarse-graining. We first considered the Smagorinsky model, which is strictly dissipative, followed by more advanced machine learning approaches based on convolutional neural networks, capable of modeling backscatter. However, these unconstrained models lack stability guarantees. To address this limitation, we introduced our skew-symmetric neural architecture [15]. This architecture enforces stability while increasing model freedom from the negative-definite eddy-viscosity basis by introducing a skew-symmetric term. A change from our previous work [15] (see Chapter 2) is that the subgrid-scale energy is no longer explicitly modeled. In addition, we tackled much more challenging 2D turbulence applications, as opposed to simple 1D test cases considered in [15]. Based on offline analysis on the training data, we hypothesized that dissipative closure models, such as the skew-symmetric architecture we introduce, are likely to perform well for the considered test cases.

We evaluated our closure models across three coarse-graining factors, from a 2048×2048 grid down to resolutions of 128×128, 64×64, and 32×32. The closure models were tested in a decaying turbulence simulation with an initial condition different from those in the training data and over an extended simulation time. We found that none of the models performed well at the largest coarse-graining factor (from 2048×2048 down to 32×32). Initially, the unconstrained machine learning models provided promising results, even outperforming our skew-symmetric model in kinetic energy predictions. However, numerical errors accumulated, leading to instability. Trajectory fitting, i.e., training the closure models to reproduce the filtered DNS solution, alone is therefore not enough to guarantee stable closure models. In contrast, our skew-symmetric model remained stable throughout, albeit at the cost of a larger dissipation rate. Despite the increased dissipation, we argue that stability is a worthwhile trade-off. In addition, our skew-symmetric architecture outperformed the Smagorinsky model in this test case and reproduced the expected k^{-3} in the energy spectrum for 2D turbulence. Furthermore, our architecture also outperformed the dynamic Smagorinsky model, a widely used and well-established physics-based closure model, in this test case.

To account for the inherent randomness in training neural networks, we trained five instances of each model with different weight initializations and mini-batch selections. All

instances of the skew-symmetric model remained stable and accurate, while conventional machine learning models exhibited significant instabilities. This highlights the improved consistency of our approach in training robust closure models.

We further assessed the models on the Kolmogorov flow test case, which was not represented in the training data. The unconstrained machine learning models again suffered from instabilities, whereas our skew-symmetric closure model successfully captured the correct energy spectrum when averaged over time. In this regard, it performed comparably to the Smagorinsky model, which proved to be well-suited to this test. We also applied backscatter clipping to the trained CNN. This resulted in a stable simulation; however, it did not outperform the skew-symmetric architecture or the Smagorinsky model.

Overall, our results demonstrate that the skew-symmetric architecture we introduced here significantly enhances the stability of machine-learning-based closure models, albeit at the cost of increased dissipation. We believe this work represents a step toward enabling long-time simulations with machine learning closures.

For future work, several directions merit exploration. The treatment of boundary conditions, particularly the padding used for convolutional neural networks, requires careful attention. Extending our approach to unstructured grids is another promising avenue, where graph neural networks could provide a useful framework [107]. Finally, introducing an additional energy source to counteract the absence of backscatter from the skew-symmetric architecture could be beneficial, though careful clipping mechanisms would be needed to prevent instabilities. Explicitly modeling the subgrid-scale energy, as in [15], is another logical extension.

4

A New Data-Driven Energy-Stable Evolve-Filter-Relax Model for Turbulent Flow Simulation

We present a novel approach to define the filter and relax steps in the evolve-filter-relax framework for simulating turbulent flows. The main advantages of this framework, as compared to the large eddy simulation (LES) framework considered in Chapter 2 and Chapter 3, are its ease of implementation and computational efficiency. However, as it only contains two parameters (one for the filter step and one for the relax step) its flexibility is rather limited. In this work, we propose a data-driven approach in which the optimal filter is found based on direct numerical simulation data in the frequency domain. The optimization step is computationally efficient and only involves one-dimensional least-squares problems for each wavenumber. Across both decaying turbulence and Kolmogorov flow, our learned filter decisively outperforms the standard differential filter and the Smagorinsky model, yielding significantly improved accuracy in energy spectra and in the temporal evolution of both energy and enstrophy. In addition, the relax parameter is determined by requiring energy and/or enstrophy conservation, which enforces stability of the method and reduces the appearance

This chapter is based on [138]. The first two authors, Anna Ivagnes and Toby van Gastelen, contributed equally to this publication.

of numerical wiggles, especially when the filter is built in scarce data regimes. Applying the learned filter is also more computationally efficient compared to traditional differential filters, as it circumvents solving a linear system.

4.1 Introduction

It is well known that modeling turbulent flows poses significant challenges for traditional numerical discretization methods due to the emergence of turbulent eddies [139]. In incompressible flows, this complexity is primarily governed by the Reynolds number, which dictates the scale of the smallest eddies present in the flow [2]. Accurately capturing these features requires a computational grid fine enough to resolve all relevant scales. As the Reynolds number increases, so does the need for finer resolution, leading to a substantial rise in computational cost. This often renders direct numerical simulation (DNS) impractical for high Reynolds number flows [140]. To address this, various modeling strategies have been developed. These include Reynolds-averaged Navier-Stokes (RANS) methods [141], which model all turbulent fluctuations through additional transport equations for the averaged quantities, and LES [6], which resolves the large, energy-containing eddies while modeling only the smaller, subgrid-scale motions. The two approaches thus differ not only in the degree of resolution but also in their underlying treatment of turbulence and computational requirements. Beyond turbulence modeling, numerical stabilization techniques [142, 143], such as the streamline upwind/Petrov–Galerkin (SUPG) method, are often employed to mitigate numerical oscillations in convection-dominated or under-resolved flows. This improves stability without explicitly modeling turbulence.

One of the most popular numerical stabilization techniques is the evolve-filter-relax (EFR) framework [17, 57]. It is a widely used approach for stabilizing numerical simulations by suppressing spurious oscillations. It has proven its worth in a multitude of applications such as nuclear engineering design, simulation of ocean circulation, and cardiovascular modeling [17, 18, 144–148]. It consists of three steps: (1) evolving the solution using a coarse spatial discretization, (2) applying a differential filter to the evolved state to dampen high-frequency numerical noise, and (3) computing a convex combination of the evolved and filtered states to mitigate excessive diffusion introduced by the filter. This process is repeated at each time step to maintain stability and accuracy throughout the simulation. Key advantages of EFR are its computational efficiency and ease of implementation. It only requires access to a discrete diffusion operator to construct the filter, making it compatible with a variety of numerical methods, including finite volumes, finite elements, and reduced-order models (ROMs) [18, 58, 59, 149–153]. In ROM applications, the filter is typically projected onto the reduced basis. In this work, however, we focus on the application of the EFR framework in LES context [17, 18], where the differential filter is constructed using a coarsened or modified

diffusion operator. This allows the method to effectively emulate subgrid-scale dissipation while remaining flexible and easy to integrate into existing simulation codes.

Applying the differential filter involves solving a linear system, where the filtered field is obtained by applying the inverse of a differential operator to the original field. In [17], it is shown that solving an elliptic filter equation is equivalent to applying an eddy-viscosity model in the LES context using a backward Euler scheme. In the context of EFR, different eddy-viscosity models are referred to as indicator functions. These functions act as spatially-varying weights that determine where and to what extent the filter is applied [60]. Furthermore, they are designed to identify regions within the domain with strong gradients or under-resolved features, such as near sharp interfaces or turbulent structures. This allows the differential filter to act locally rather than globally. This adaptive approach improves the accuracy and efficiency of the EFR method, making it suitable for complex flow simulations. Moreover, although it is sparse, the filter step still requires solving a linear system to apply the differential filter, which is computationally expensive.

The EFR framework only contains two parameters, namely the filter radius δ and the relaxation parameter χ [17, 57]. Choosing the values for these parameters requires careful consideration. If the filter radius is too large, the solution fields become too diffused, while a too small value might fail in adequately suppressing the noise. Different ways of choosing these parameters have been suggested: e.g. setting the filter radius either equal to the mesh size [18, 63], or equal to the Kolmogorov lengthscale [62], and setting the relaxation parameter proportional to the size of the time step [62, 63]. The sensitivity of regularized and Leray-type models to the choice of the filter radius has also been analyzed in detail [154], while the impact of filtering parameters in patient-specific LES simulations has been investigated through global sensitivity analysis [155].

However, these approaches for choosing the parameters do not fully alleviate the limited flexibility of the model, as highlighted by recent results in [64]. In that work, it was shown that the optimal values of the parameters are highly variable during the course of a simulation. By introducing a data-driven approach to determine the optimal parameters at each time step, the results more closely resembled the DNS results. However, the extrapolation capabilities of such an approach are limited, as DNS data for each time step is required to determine the optimal parameter settings. Making use of different indicator functions has been shown to improve the performance of the EFR framework. Different indicator functions are available, which effectively make the differential filter non-linear [62]. An overview of such functions can be found here [17] specialized to different use cases.

While traditional EFR approaches rely on physically or mathematically motivated indicator functions to control dissipation and filter strength, recent years have seen growing interest in data-driven alternatives. These data-driven approaches are especially researched in the LES community. The LES and EFR frameworks are related in the fact that both

aim to suppress spurious oscillations during simulation time and do this in a similar manner [17]. In particular, machine learning approaches have come forward as a way to improve the accuracy of LESs [14, 40, 45, 48, 49, 51, 89, 156, 157]. An example of this is the work in [39], where a convolutional neural network was added to the right hand side of the system to improve LES. It was found that the accuracy of the resulting models relies heavily on the way the neural network parameters are optimized. Using a differentiable solver, gradient-based optimization techniques can be applied to optimize these parameters with respect to how well the coarsened DNS solution is reproduced. This is often required, as training neural networks to solely reproduce the right-hand side, although requiring significantly less computational effort, often displays inaccuracies and instabilities [39, 52–55]. The clear downside of the machine learning approach is the required computational effort to generate the training data and train the neural networks, and the required access to a differentiable solver. The latter makes the approach incompatible with many of the existing codebases and requires specialized solvers such as [158]. Furthermore, the neural networks are less interpretable than the differentiable filter used in the EFR framework.

In this chapter, we aim to combine the increased flexibility of data-driven approaches with the modularity and computational efficiency of the EFR framework. We therefore propose the following idea: *learn the optimal data-driven linear filter* which is trained on DNS data. This allows us to construct a filter that is tailored to the problem at hand. We combine the learned filter with an energy-conserving finite volume discretization such that we can analyze the stability of the method. Next to being more flexible than the differentiable filter, this approach has some other key advantages: Making use of a fast Fourier transform (FFT) the filter can be applied in a computationally efficient manner and no longer involves solving a linear system. Furthermore, computing the optimal filter coefficients involves only the solution of one-dimensional least-squares problems for each coefficient. This makes finding the optimal coefficients much more efficient than training a neural network [41]. Furthermore, we introduce a clever way to use the relax parameter χ to preserve both energy and enstrophy. This is done with the goal of ensuring both accuracy and stability when little DNS data is available. A flowchart of our developed methodology is shown in Figure 4.1.

The chapter is structured as follows: In Section 4.2 we start off by introducing the incompressible Navier-Stokes equations, discuss the used spatial discretization, and introduce the EFR framework. These are regarded as *preliminaries*. Furthermore, in Section 4.3 we introduce our data-driven methodology and the way we impose the energy and enstrophy constraints. After this, we present our results in Section 4.4, where we compare to existing approaches. We conclude our work in Section 4.5.

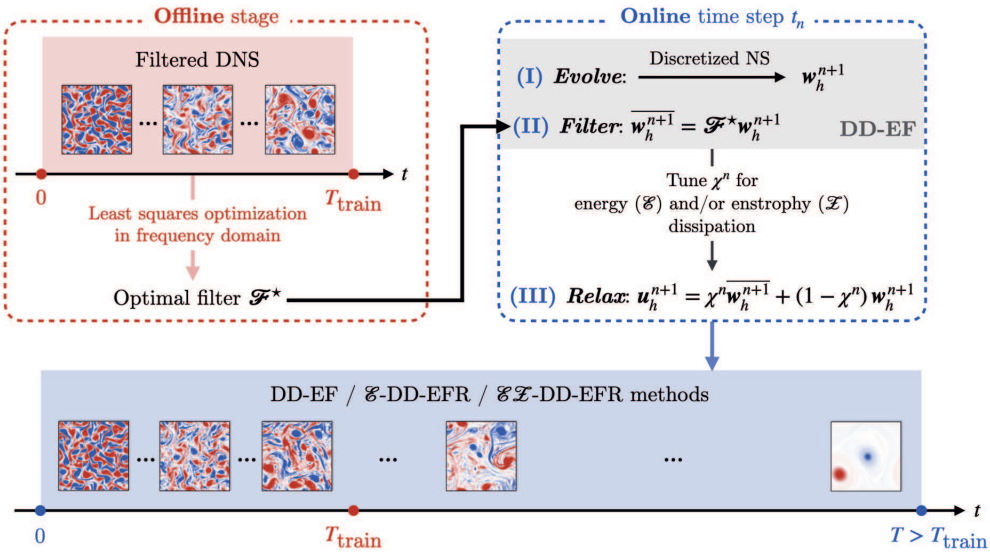


Figure 4.1: Workflow of the DD-EFR framework presented in the article. An optimal filter $\mathcal{F}^\star$ is learned *offline* from filtered DNS data. The relax parameter is tuned *online* at each time step to control energy and/or enstrophy dissipation.

4.2 Preliminaries

In this section, we introduce the Navier-Stokes equations and describe the discretization in space and time. Moreover, we introduce the evolve-filter-relax (EFR) framework to account for coarse grids.

4.2.1 The Navier-Stokes equations

We model the dynamics of an incompressible fluid using the incompressible Navier-Stokes equations (NSE). Let $\Omega \subset \mathbb{R}^d$ be a fixed spatial domain, where $d = 2$ or 3. The fluid velocity is denoted by $\mathbf{u} \doteq \mathbf{u}(\boldsymbol{x}, t) \in \mathbb{U}$, and the pressure by $p \doteq p(\boldsymbol{x}, t) \in \mathbb{Q}$. The governing equations for the fluid motion are:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \Delta \mathbf{u} + \nabla p = f & \text{in } \Omega \times (t_0, T), \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega \times (t_0, T), \end{cases} \tag{4.1}$$

with the initial condition $\mathbf{u} = \mathbf{u}_0$ in $\Omega \times t_0$. Here, f denotes external forcing, ν is the kinematic viscosity, and $\mathbb{U}$ and $\mathbb{Q}$ are appropriate Hilbert spaces for velocity and pressure, respectively. The simulation runs over the time interval (t_0, T).

In this study, we adopt periodic boundary conditions and therefore omit an explicit discussion of boundary terms. The incompressibility condition, $\nabla \cdot \mathbf{u} = 0$, enforces mass conservation. The nature of the flow is characterized by the dimensionless Reynolds number:

$$Re \doteq \frac{UL}{\nu}, \tag{4.2}$$

where U and L are the characteristic velocity and the length scale of the system, respectively. A high Reynolds number indicates that inertial effects outweigh viscous effects, leading to a convection-dominated regime.

4.2.2 Finite volume discretization

In the numerical tests, we employ a finite difference discretization on a staggered grid presented in [24, 25], and an explicit fourth-order Runge-Kutta time integration scheme. Using this discretization, the velocity field is approximated on the cell-faces and is discretely represented by the vector $\mathbf{u}_h \in \mathbb{R}^{N_u}$, whereas the pressure field is approximated in the cell-centers as $\mathbf{p}_h \in \mathbb{R}^{N_p}$. N_u and N_p are the number of discrete points for the velocity field and the pressure, respectively. Using the finite difference discretization (4.1) is written in matrix representation as

$$\frac{d\mathbf{u}_h}{dt} = -\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h + \nu \mathbf{D}_h \mathbf{u}_h + \mathbf{f}_h - \mathbf{G}_h \mathbf{p}_h, \tag{4.3}$$

where $\mathbf{C}_h(\mathbf{u}_h) \in \mathbb{R}^{N_u \times N_u}$ represents the nonlinear convection operator, $\mathbf{D}_h \in \mathbb{R}^{N_u \times N_u}$ the linear diffusion operator, and $\mathbf{f}_h \in \mathbb{R}^{N_u}$ the external forces. Moreover, $\mathbf{G}_h \in \mathbb{R}^{N_u \times N_p}$ is the pressure gradient operator. In matrix representation, the divergence-freeness condition is written as

$$\mathbf{M}_h \mathbf{u}_h = \mathbf{0}_h, \tag{4.4}$$

where $\mathbf{M}_h \in \mathbb{R}^{N_p \times N_u}$ represents the divergence operator and $\mathbf{0}_h \in \mathbb{R}^{N_p}$ represents a vector of zeros. The staggered grid discretization is chosen as it preserves the kinetic energy in the inviscid limit. To see this, we start by defining the (kinetic) energy as

$$\mathcal{E} := \frac{1}{2|\Omega|} \int_\Omega \mathbf{u} \cdot \mathbf{u} \, \mathrm{d}\Omega. \tag{4.5}$$

Discretely, we represent this as

$$\mathcal{E}_h := \frac{h^2}{2|\Omega|} \|\mathbf{u}_h\|_2^2, \tag{4.6}$$

where

$$\|\mathbf{u}_h\|_2^2 := \mathbf{u}_h^T \mathbf{u}_h, \tag{4.7}$$

assuming a uniform 2D grid with grid spacing h in each direction, as used in this work. The appearance of h^2 is such that the unit of the discrete energy is the same as its continuous counterpart, i.e. (4.6) discretely approximates the integral in (4.5). The discretization in (4.3) produces the following energy behavior:

$$\frac{\mathrm{d}\mathcal{E}_h}{\mathrm{d}t} = \frac{h^2}{2|\Omega|}\mathbf{u}_h^T\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = \frac{h^2}{2|\Omega|}(-\nu\|\mathbf{Q}_h\mathbf{u}_h\|_2^2+\mathbf{u}_h^T\boldsymbol{f}_h). \tag{4.8}$$

The energy behavior in (4.8) is derived by using the product rule, starting from (4.6), and using the fact that $\mathbf{D}_h$ can be Cholesky decomposed as $\mathbf{D}_h := -\mathbf{Q}_h\mathbf{Q}_h$, while the pressure gradient and convective term contributions cancel, see [24, 25]. From (4.8) it is clear that this discretization provides stability, as in the absence of forcing, the energy can never increase. Note that this energy behavior is true for periodic boundary conditions. In the case of boundary conditions of different type, boundary contributions will appear in the energy equation.

4.2.3 Evolve-Filter-Relax

In under-resolved or marginally-resolved simulations, central spatial discretization schemes, such as the one adopted in this work, tend to produce spurious oscillations in convection-dominated regimes. To mitigate these artifacts, we employ the EFR algorithm. Let the time step be denoted by Δt, with $t_n = t_0 + n\Delta t$ for $n = 0, \ldots, N$, and total simulation time $T = t_0 + N\Delta t$. We write y^n to denote the numerical approximation of a generic quantity y at time t_n.

The EFR procedure consists of the following three steps:

$$\textbf{(I)} \quad \textit{Evolve:} \quad \begin{cases} \dfrac{\mathbf{w}_h^{n+1}-\mathbf{u}_h^n}{\Delta t} = -\mathbf{C}_h(\mathbf{u}_h^n)\mathbf{u}_h^n + \nu\mathbf{D}_h\mathbf{u}_h^n + \mathbf{f}_h - \mathbf{G}_h\boldsymbol{p}_h^n, \\ \mathbf{M}_h\mathbf{w}_h^{n+1} = \mathbf{0}_h, \end{cases}$$

$$\textbf{(II)} \quad \textit{Filter:} \quad (\mathbf{I} - 2\delta^2\mathbf{D}_h)\overline{\mathbf{w}}_h^{n+1} = \mathbf{w}_h^{n+1}$$

$$\textbf{(III)} \quad \textit{Relax:} \quad \mathbf{u}_h^{n+1} = (1-\chi)\mathbf{w}_h^{n+1} + \chi\,\overline{\mathbf{w}}_h^{n+1}$$

Here, $\mathbf{w}_h^{n+1}$ is the velocity field obtained from the evolve step, $\overline{\mathbf{w}}_h^{n+1}$ is its filtered counterpart, and $\mathbf{u}_h^{n+1}$ is the final relaxed velocity field. The parameters δ and χ denote the *filter radius* and *relaxation parameter*, respectively, with $\chi \in [0, 1]$.

In step **(I)**, the velocity field is advanced in time using a standard temporal discretization of the NSE. For simplicity, a forward Euler method is shown here, though in practice higher-order schemes are commonly used to enhance accuracy and stability [24, 96].

Step **(II)** applies a *differential filter* to smooth the velocity field. The filter is defined via an elliptic operator involving the discrete Laplacian $\mathbf{D}_h$ and an explicit spatial scale δ. This step removes small-scale (high-frequency) components and has been shown to be mathematically robust [60]. The filtered field $\overline{\mathbf{w}}_h^{n+1}$ is obtained by solving a linear system.

Step **(III)** combines the evolved and filtered velocities using a convex combination controlled by the relaxation parameter χ. The extremes of this procedure are:

(i) $\chi = 1$: the fully filtered velocity is used, i.e., $\mathbf{u}_h^{n+1} = \overline{\mathbf{w}}_h^{n+1}$. We will refer to this approach as EF;

(ii) $\chi = 0$: no filtering is applied, and $\mathbf{u}_h^{n+1} = \mathbf{w}_h^{n+1}$, and we refer to this as noEFR.

Thus, the parameter χ modulates the influence of filtering, allowing one to tune the balance between stability and accuracy [59, 159, 160]. This is particularly useful when filtering leads to excessive numerical diffusion, as discussed in [62].

The filter radius δ is typically chosen in relation to either the minimum grid size $h_{\min}$ or the *Kolmogorov length scale* $\eta = L\,Re^{-3/4}$, and usually $\eta \simeq h_{\min}$ for well-resolved DNSs [62, 63].

4.2.3.1 Stability of the differential filter

In Section 4.2.2 we discussed the stability of the spatial discretization. In this section, we do the same for the EFR algorithm. In order to derive stability of the differential filter, we consider the relaxed velocity field at time step $n+1$ as

$$\mathbf{u}_h^{n+1} = (1-\chi)\mathbf{w}_h^{n+1} + \chi\boldsymbol{\mathcal{F}}\mathbf{w}_h^{n+1}, \tag{4.9}$$

where $\boldsymbol{\mathcal{F}} = (\mathbf{I} - 2\delta^2\mathbf{D}_h)^{-1}$ represents the filter such that $\overline{\mathbf{w}_h}^{n+1} = \boldsymbol{\mathcal{F}}\mathbf{w}_h^{n+1}$. In order to prove stability, we aim to show the following:

$$\|\mathbf{u}_h^{n+1}\|_2^2 \leq \|\mathbf{w}_h^{n+1}\|_2^2, \tag{4.10}$$

i.e. the relax step dissipates energy from the system. Since the relax parameter satisfies $0 \leq \chi \leq 1$, expression (4.9) is a convex combination of vectors, and the following holds [161]:

$$\min(\|\mathbf{w}_h^{n+1}\|_2^2, \|\boldsymbol{\mathcal{F}}\mathbf{w}_h^{n+1}\|_2^2) \leq \|\mathbf{u}_h^{n+1}\|_2^2 \leq \max(\|\mathbf{w}_h^{n+1}\|_2^2, \|\boldsymbol{\mathcal{F}}\mathbf{w}_h^{n+1}\|_2^2). \tag{4.11}$$

From this, we find that

$$\|\boldsymbol{\mathcal{F}}\mathbf{w}_h\|_2^2 \leq \|\mathbf{w}_h\|_2^2 \tag{4.12}$$

is the required condition for the filter to be dissipative, where we omitted the superscripts n and $n+1$ for ease of notation.

To show this for the differential filter, we express the diffusion operator $\mathbf{D}_h$ in terms of its eigenvalues $\mathbf{\Lambda}_{ii} = \lambda_i$ and orthonormal eigenvectors P and use this to rewrite the filter:

$$\begin{aligned}\boldsymbol{\mathcal{F}} &= (\mathbf{I} - 2\delta^2\mathbf{D}_h)^{-1} = (\mathbf{P}\mathbf{I}\mathbf{P}^\dagger - 2\delta^2\mathbf{P}\mathbf{\Lambda}\mathbf{P}^\dagger)^{-1} = (\mathbf{P}(\mathbf{I} - 2\delta^2\mathbf{\Lambda})\mathbf{P}^\dagger)^{-1} \\ &= (\mathbf{P}^\dagger)^{-1}(\mathbf{I} - 2\delta^2\mathbf{\Lambda})^{-1}\mathbf{P}^{-1} = \mathbf{P}(\mathbf{I} - 2\delta^2\mathbf{\Lambda})^{-1}\mathbf{P}^\dagger,\end{aligned} \tag{4.13}$$

where the eigenvectors form an orthogonal basis due to the symmetry of $\mathbf{D}_h$, i.e. $\mathbf{P}^\dagger\mathbf{P} = \mathbf{P}\mathbf{P}^\dagger = \mathbf{I}$, with the superscript $\dagger$ indicating the Hermitian conjugate. Note that for periodic boundary conditions and uniform grids, the eigenvectors in $\mathbf{P}$ simply correspond to the Fourier modes [162]. Next, we express the filtering procedure in eigenvector space:

$$\boldsymbol{\mathcal{F}}\mathbf{w}_h = \mathbf{P}(\mathbf{I} - 2\delta^2\mathbf{\Lambda})^{-1}\mathbf{P}^\dagger\mathbf{w}_h. \tag{4.14}$$

The diagonal entries of this matrix, which correspond to the eigenvalues of $\boldsymbol{\mathcal{F}}$, are given by

$$\hat{\mathcal{F}}_{ii} = \hat{f}_i := \frac{1}{1 - 2\delta^2\lambda_i} \in \mathbb{C}, \quad i = 1, \ldots, N_u. \tag{4.15}$$

Next, we use (4.14) to rewrite the stability inequality (4.12):

$$\|\mathbf{P}\hat{\boldsymbol{\mathcal{F}}}\mathbf{P}^\dagger\mathbf{w}_h\|_2^2 = (\mathbf{w}_h)^\dagger\mathbf{P}\hat{\boldsymbol{\mathcal{F}}}^\dagger\mathbf{P}^\dagger\mathbf{P}\hat{\boldsymbol{\mathcal{F}}}\mathbf{P}^\dagger\mathbf{w}_h = (\mathbf{w}_h)^\dagger\mathbf{P}\hat{\boldsymbol{\mathcal{F}}}^\dagger\hat{\boldsymbol{\mathcal{F}}}\mathbf{P}^\dagger\mathbf{w}_h = \sum_{i=1}|\hat{f}_i|^2|\hat{\mathrm{w}}_{h,i}|^2 \leq \sum_{i=1}|\hat{\mathrm{w}}_{h,i}|^2, \tag{4.16}$$

where

$$\mathbf{P}^\dagger\mathbf{w}_h = \hat{\mathbf{w}}_h \in \mathbb{C}^{N_u} \tag{4.17}$$

and

$$|\hat{f}_i|^2 := \hat{f}_i^\dagger\hat{f}_i. \tag{4.18}$$

In order for this inequality to be satisfied we require $|\hat{f}_i|^2 \leq 1\,\forall i = 1, \ldots, N_u$. Looking at (4.15), this is satisfied as the eigenvalues for the diffusion operator are both real and negative [67]. This condition is therefore sufficient to show diffusivity/stability of the EFR formulation, and independent of the value of δ. An alternative viewpoint is that the filter $\boldsymbol{\mathcal{F}} = (\mathbf{I} - 2\delta^2\mathbf{D}_h)^{-1}$ is the discrete analogue of the resolvent of the Laplace operator $J_\alpha = (I - \alpha\Delta)^{-1}$ [163], which satisfies $\|J_\alpha\|_{L^2\to L^2} \leq 1$ for $\alpha \geq 0$. Since $\mathbf{D}_h$ is a symmetric negative semidefinite discretization of Δ, the same nonexpansive property holds in the discrete setting, i.e. $\|\boldsymbol{\mathcal{F}}\|_2 \leq 1$.

4.3 Methods

In this section, we aim to increase the flexibility of the EFR framework with a data-driven approach and suggest a structure-preserving extension using the relaxation parameter χ. A graphical version of the methodological pipeline can be found in Figure 4.1.

4.3.1 Data-driven linear filter

In the previous section, we expressed the filter in the EFR framework as the matrix vector product $\mathbf{P}^\dagger \hat{\boldsymbol{\mathcal{F}}} \mathbf{P}$, where $\hat{\boldsymbol{\mathcal{F}}}$ is a diagonal matrix. In this section, we present the following idea, which is one of the key novelties of our work: We can derive the diagonal elements $\hat{f}_i$ using a data-driven approach. We will indicate with $\hat{\boldsymbol{\mathcal{F}}}^\star$ the *optimal* filter, fitted to some reference data, and with $\hat{f}_i^\star, i = 1, \ldots, N_u$ its diagonal elements. It is important to emphasize that the optimal filter needs to be computed only *once*, serving as the *offline* stage in the typical surrogate modeling fashion, as depicted in the red dashed box of Figure 4.1.

For periodic boundary conditions and uniform grids, the eigenvectors in $\mathbf{P}$ simply correspond to the Fourier modes. Using the notation $\hat{\cdot}$ for the Fourier coefficients, we can use a FFT to efficiently obtain $\hat{\mathbf{w}}_h$ [164], i.e.

$$\hat{\mathbf{w}}_h = \mathbf{P}^\dagger \mathbf{w}_h = \mathrm{FFT}(\mathbf{w}_h), \tag{4.19}$$

$$\mathbf{w}_h = \mathbf{P} \hat{\mathbf{w}}_h = \mathrm{FFT}^{-1}(\hat{\mathbf{w}}_h), \tag{4.20}$$

For non-periodic boundary conditions, this is more complicated as we require the eigenvectors to satisfy the boundary conditions. This can possibly be done through the use of lifting functions, like in [69]. However, in this chapter, we restrict ourselves to the periodic case for simplicity.

To derive the values for the complex filter coefficients $\hat{f}_i^\star$ we use reference data $\mathbf{U}^{\mathrm{true}}$ and $\mathbf{W}^{\mathrm{true}}$, collected within the time window $[0, T_{\mathrm{train}}]$.

In particular, we consider a parametrized flow case with parameter $\boldsymbol{\mu}$, given in our numerical tests by different initial conditions. For the derivation of the optimal filter, we consider:

- $\boldsymbol{\mu}^{(j)}, j = 1, \ldots, I_{\mathrm{train}}$ different initial conditions;
- $n = 1, \ldots, N_{\mathrm{train}}$ time instances for each configuration, where N_{train} corresponds to the training time T_{train}.

Matrix $\mathbf{U}^{\mathrm{true}}$ contains in its columns the snapshots of the *true* velocity field $\mathbf{u}^n_{\mathrm{true}}(\boldsymbol{\mu}^{(j)})$, $n = 2, \ldots, N_{\mathrm{train}}$; $j = 1, \ldots, I_{\mathrm{train}}$. Then, $\mathbf{U}^{\mathrm{true}} \in \mathbb{R}^{N_u \times M}$, with $M = N_{\mathrm{train}}\,(I_{\mathrm{train}} - 1)$. The true velocity fields are obtained from fine-grid DNS data that is coarse-grained using a

face-averaging filter, as described in [40]. Matrix $\mathbf{U}^{\text{true}}$ may be written as:

$$\mathbf{U}^{\text{true}} = \begin{bmatrix} | & & | & & | & & | \\ \mathbf{u}^2_{\text{true}}(\boldsymbol{\mu}^{(1)}) & \dots & \mathbf{u}^{N_{\text{train}}}_{\text{true}}(\boldsymbol{\mu}^{(1)}) & \dots & \mathbf{u}^2_{\text{true}}(\boldsymbol{\mu}^{(I_{\text{train}})}) & \dots & \mathbf{u}^{N_{\text{train}}}_{\text{true}}(\boldsymbol{\mu}^{(I_{\text{train}})}) \\ | & & | & & | & & | \end{bmatrix}. \tag{4.21}$$

On the other hand, matrix $\mathbf{W}^{\text{true}}$ is obtained by evolving each true velocity field $\mathbf{u}^n_{\text{true}}$, $n = 1, \dots, N_{\text{train}} - 1$ for a single time step with the coarse grid solver, namely:

$$\begin{aligned} \mathbf{W}^{\text{true}} &= \begin{bmatrix} | & & | \\ \mathbf{w}^2_{\text{true}}(\boldsymbol{\mu}^{(1)}) & \dots & \mathbf{w}^{N_{\text{train}}}_{\text{true}}(\boldsymbol{\mu}^{(I_{\text{train}})}) \\ | & & | \end{bmatrix} \\ &= \text{evolve}\left(\begin{bmatrix} | & & | \\ \mathbf{u}^1_{\text{true}}(\boldsymbol{\mu}^{(1)}) & \dots & \mathbf{u}^{N_{\text{train}-1}}_{\text{true}}(\boldsymbol{\mu}^{(I_{\text{train}})}) \\ | & & | \end{bmatrix} \right). \end{aligned} \tag{4.22}$$

Note that for the initial state $\mathbf{u}^1_{\text{true}} = \mathbf{w}^1_{\text{true}}$, which is why it is omitted from the snapshot matrices. In the numerical results, we will consider two different *data regimes* to find the optimal filter: the *scarce* data regime ($I_{\text{train}} = 1$), and the *full* data regime ($I_{\text{train}} = 10$).

In this subsection, we focus on the filter step (equivalent to choosing $\chi = 1$ in the relax step). The goal is to find filter coefficients such that the filtered coarse-grid solution is close to the reference solution $\mathbf{U}^{\text{true}}$:

$$\mathbf{U}^{\text{true}} \approx \boldsymbol{\mathcal{F}}\mathbf{W}^{\text{true}}. \tag{4.23}$$

A suitable error to satisfy this approximation is

$$\mathcal{L}_{\hat{f}}(\mathbf{W}^{\text{true}}, \mathbf{U}^{\text{true}}) := \|\boldsymbol{\mathcal{F}}\mathbf{W}^{\text{true}} - \mathbf{U}^{\text{true}}\|^2_F, \tag{4.24}$$

where the subscript F indicates the Frobenius norm. As $\mathbf{P}^\dagger$ is orthogonal, we can place it inside the norm

$$\mathcal{L}_{\hat{f}}(\mathbf{W}^{\text{true}}, \mathbf{U}^{\text{true}}) = \|\mathbf{P}^\dagger\boldsymbol{\mathcal{F}}\mathbf{W}^{\text{true}} - \mathbf{P}^\dagger\mathbf{U}^{\text{true}}\|^2_F = \|\hat{\boldsymbol{\mathcal{F}}}\hat{\mathbf{W}}^{\text{true}} - \hat{\mathbf{U}}^{\text{true}}\|^2_F, \tag{4.25}$$

where $\hat{\cdot}$ indicates the Fourier coefficients and we use the fact that $\boldsymbol{\mathcal{F}} = \mathbf{P}\hat{\boldsymbol{\mathcal{F}}}\mathbf{P}^\dagger$. As $\hat{\boldsymbol{\mathcal{F}}}$ is

diagonal Equation (4.25) can be rewritten as

$$\begin{aligned}\mathcal{L}_{\hat{f}}(\mathbf{W}^{\text{true}}, \mathbf{U}^{\text{true}}) &= \|\hat{\boldsymbol{\mathcal{F}}}\hat{\mathbf{W}}^{\text{true}} - \hat{\mathbf{U}}^{\text{true}}\|_F^2 = \sum_i \|\hat{f}_i \hat{\mathbf{W}}_i^{\text{true}} - \hat{\mathbf{U}}_i^{\text{true}}\|_2^2 \\ &= \sum_i (|\hat{f}_i|^2 (\hat{\mathbf{W}}_i^{\text{true}})^\dagger \hat{\mathbf{W}}_i^{\text{true}} - \hat{f}_i^\dagger (\hat{\mathbf{W}}_i^{\text{true}})^\dagger \hat{\mathbf{U}}_i^{\text{true}} - (\hat{\mathbf{U}}_i^{\text{true}})^\dagger \hat{f}_i \hat{\mathbf{W}}_i^{\text{true}} \\ &\quad + (\hat{\mathbf{U}}_i^{\text{true}})^\dagger \hat{\mathbf{U}}_i^{\text{true}},\end{aligned} \tag{4.26}$$

where the subscript i represents the i-th row of the matrices. The optimal values for $\hat{f}_i$ can be found by computing the gradient of $\mathcal{L}_{\hat{f}}$ with respect to $\hat{f}_i^\dagger$ and setting it to zero:

$$\frac{\partial \mathcal{L}_{\hat{f}}(\mathbf{W}^{\text{true}}, \mathbf{U}^{\text{true}})}{\partial \hat{f}_i^\dagger} = 2\hat{f}_i (\hat{\mathbf{W}}_i^{\text{true}})^\dagger \hat{\mathbf{W}}_i^{\text{true}} - 2(\hat{\mathbf{W}}_i^{\text{true}})^\dagger \hat{\mathbf{U}}_i^{\text{true}} = 0. \tag{4.27}$$

Note we take the gradient with respect to $\hat{f}_i^\dagger$ and not $\hat{f}_i$, to obtain an equation which solely depends on $\hat{f}_i$ and not $\hat{f}_i^\dagger$. Solving (4.27) yields

$$\boxed{\hat{f}_i^* = \frac{(\hat{\mathbf{W}}_i^{\text{true}})^\dagger \hat{\mathbf{U}}_i^{\text{true}}}{(\hat{\mathbf{W}}_i^{\text{true}})^\dagger \hat{\mathbf{W}}_i^{\text{true}}}} \tag{4.28}$$

for the optimal values $\hat{f}_i^*$, $i = 1, \ldots, N_u$. As $\mathbf{U}^{\text{true}}$ is real-valued the resulting $\boldsymbol{\mathcal{F}}^\star$ is also real. Simulations are carried out using this filter and setting $\chi = 1$, such that we effectively skip the relax step. This will be referred to as DD-EF. The filtered velocity field will be denoted as

$$\overline{\mathbf{w}}_h := \boldsymbol{\mathcal{F}}^\star \mathbf{w}_h = \mathbf{P}^\dagger \hat{\boldsymbol{\mathcal{F}}}^\star \mathbf{P} \mathbf{w}_h, \tag{4.29}$$

which is computed efficiently in the frequency domain using a FFT, as $\hat{\boldsymbol{\mathcal{F}}}^\star$ is diagonal.

We finally remark that this data-driven filter can be built offline from a limited amount of data, and then employed in online simulations in extrapolation regimes, both in time and with respect to varying parameters such as initial conditions or forcing terms. This separation between offline training and online deployment enables fast and flexible evaluations across multiple scenarios, without the need for recomputing the filter.

4.3.2 Conservation of energy using χ

As discussed in 4.2.3.1, one of the desirable properties of the existing differential filter is that it guarantees stability of the system by dissipating kinetic energy. This is guaranteed by the fact that $|\hat{f}_i| \leq 1, \ \forall i$ for this filter.

However, this condition is in general not met by the filter in (4.28). Requiring the same bound on the DD-EF approach seems natural, but can lead to overdiffusive behavior, as it enforces energy dissipation for each frequency. In fact, $|\hat{f}_i^\star|^2 \leq 1$ is a *sufficient* but not

a *necessary* condition to achieve stability. Instead, we therefore impose a *global* constraint on the energy, which ensures stability while still allowing for some frequencies to have an increasing energy.

This constraint can be enforced by a smart choice of the relaxation parameter χ at each time step t^n, as follows:

$$\mathcal{E}_h^n \propto \|\mathbf{u}_h^n\|_2^2 = \|(1-\chi)\mathbf{w}_h^n + \chi\overline{\mathbf{w}}_h^n\|_2^2 \leq \|\mathbf{w}_h^n\|_2^2, \tag{4.30}$$

where $\mathcal{E}_h^n$ is the discrete global energy at t_n. In these expressions, $\mathbf{w}_h^n$, $\overline{\mathbf{w}}_h^n$, and $\mathbf{u}_h^n$ correspond to the *online* solution approximations at time step t^n, and not to the snapshot matrices $\mathbf{W}_{\text{true}}$ and $\mathbf{U}_{\text{true}}$ used in the offline computation of $\hat{\boldsymbol{\mathcal{F}}}^\star$ in Section 4.3.1.

One effective approach to enforce the global dissipativity constraint is to compute the filter *offline*, as in equation (4.28), and adaptively tune $\chi^n = \chi(t^n)$ at each *online* time step, as represented in the blue dashed box in Figure 4.1. From the constraint (4.30), we derive the following inequality:

$$(\chi^n)^2\tilde{a}^n + 2\chi^n\tilde{b}^n \leq 0, \tag{4.31}$$

where

$$\tilde{a}^n = \|\mathbf{w}_h^n - \overline{\mathbf{w}}_h^n\|_2^2 \quad \text{and} \quad \tilde{b}^n = (\mathbf{w}_h^n)^T\overline{\mathbf{w}}_h^n - \|\mathbf{w}_h^n\|_2^2.$$

Given that $0 \leq \chi^n \leq 1$, to remain consistent with the relax step in the EFR approach (see Section 4.2.3), inequality (4.31) yields the bound:

$$\boxed{\begin{cases} \chi^n \leq -\frac{2\tilde{b}^n}{\tilde{a}^n} & \text{if } \tilde{b}^n \leq 0, \\ \chi^n = 0 & \text{otherwise.} \end{cases}} \tag{4.32}$$

Accordingly, during the online phase, we select the optimal relaxation parameter (we refer to it as $\chi_{\mathcal{E}\text{-opt}}$) at each time step t^n as follows:

- If constraint (4.30) is satisfied for $\chi = 1$:

  $$\chi_{\mathcal{E}\text{-opt}} = 1,$$

  i.e., we only perform a filtering step. Relaxation is not needed since the filtered solution satisfies the global energy inequality.

- If constraint (4.30) is *not* satisfied:
  - $\chi_{\mathcal{E}\text{-opt}} = -\frac{2\tilde{b}^n}{\tilde{a}^n}$, if $\tilde{b}^n \leq 0$ (the least dissipative value that ensures the constraint);
  - $\chi_{\mathcal{E}\text{-opt}} = 0$, if $\tilde{b}^n > 0$. This means that we simply use the coarse grid solution without applying any filtering or relaxation. While this yields an energy-stable

result, it may exhibit spurious oscillations due to the coarse resolution, making it a less desirable option in practice.

This approach will be referred to as $\mathcal{E}$-DD-EFR. Note that these computations are all carried out in physical space, as the FFT is only required to efficiently apply the data-driven filter. Moreover, computing the inner products in this procedure in the frequency domain does not offer any computational speedup.

4.3.3 Conservation of enstrophy using χ

The derivation in Section 4.3.2 does not inherently ensure a monotonic decrease in enstrophy, a property that should hold for the viscous 2D NSE [165]. Preserving this property is important, as an increase in enstrophy often indicates the emergence of unphysical oscillations the numerical solution. This occurs because enstrophy is sensitive to high-frequency modes, as it involves the squared vorticity norm:

$$\mathcal{Z} := \frac{1}{2|\Omega|}\int_{\Omega}\|\nabla\times\mathbf{u}\|_2^2\,\mathrm{d}\Omega. \tag{4.33}$$

In section 4.4, and specifically in Figure 4.8, we demonstrate that enforcing this physical constraint improves simulation accuracy, particularly in data-scarce training regimes.

To satisfy this physical constraint, we impose the following inequality at each online time step t_n:

$$\mathcal{Z}_h^n := \frac{h^2}{2|\Omega|}\|\mathbf{B}_h\mathbf{u}_h^n\|_2^2 \le \frac{h^2}{2|\Omega|}\|\mathbf{B}_h\mathbf{w}_h^n\|_2^2, \tag{4.34}$$

where $\mathbf{B}_h \in \mathbb{R}^{N_p\times N_u}$ is a discrete curl operator, such that $\mathcal{Z}_h^n$ approximates the global enstrophy. The above expression may be rewritten in a form similar to the one found for the energy-preserving inequality, namely:

$$(\chi^n)^2\,\tilde{a}_{\mathcal{Z}}^n + 2\chi^n\tilde{b}_{\mathcal{Z}}^n \le 0, \tag{4.35}$$

where

$$\tilde{a}_{\mathcal{Z}}^n = \|\mathbf{B}_h\mathbf{w}_h^n - \mathbf{B}_h\overline{\mathbf{w}}_h^n\|_2^2 \quad\text{and}\quad \tilde{b}_{\mathcal{Z}}^n = (\mathbf{B}_h\mathbf{w}_h^n)^T\mathbf{B}_h\overline{\mathbf{w}}_h^n - \|\mathbf{B}_h\mathbf{w}_h^n\|_2^2.$$

Exactly as for the energy inequality, the optimal χ satisfying inequality (4.35) is $\chi_{\mathcal{Z}\text{-opt}} = -2\frac{\tilde{b}_{\mathcal{Z}}^n}{\tilde{a}_{\mathcal{Z}}^n}$. Also in this case, we have:

$$\boxed{\begin{cases}\chi^n \le -\frac{2\tilde{b}_{\mathcal{Z}}^n}{\tilde{a}_{\mathcal{Z}}^n} & \text{if } \tilde{b}_{\mathcal{Z}}^n \le 0,\\ \chi^n = 0 & \text{otherwise.}\end{cases}} \tag{4.36}$$

As for $\tilde{b}^n$, $\tilde{b}^n_{\mathcal{Z}}$ may also be negative. In that case, one would simply retain $\chi_{\mathcal{Z}-\text{opt}} = 0$. This approach can easily be combined with the energy-conserving approach by simply setting χ to $\chi_{\mathcal{E}\mathcal{Z}\text{-opt}} = \min\{\chi_{\mathcal{E}\text{-opt}}, \chi_{\mathcal{Z}\text{-opt}}\}$ to minimize dissipation. While conceptually straightforward, its implementation is slightly more involved due to the need to compute discrete gradients within the staggered grid framework. We refer to this approach as $\mathcal{E}\mathcal{Z}$-DD-EFR.

4.4 Numerical results

The methodology presented in Section 4.3 is tested here on two test cases: 2D decaying homogeneous turbulence (4.4.1), and the 2D Kolmogorov flow (4.4.2). Both cases are performed in a square domain $\Omega = [0, 1]^2$ at Reynolds number $Re = 4 \times 10^4$, with periodic boundary conditions. Time integration is performed using a fourth-order explicit Runge–Kutta method [166].

The DNS is carried out on a refined grid, whereas both the proposed and state-of-the-art methods are implemented and compared on a coarse grid. The goal is to find a filtering strategy that allows us to obtain accurate results despite the reduced resolution. More details on the different methods compared in the article and on the corresponding mesh sizes are available in Table 4.1. We will refer to the DNS simulation on the coarse grid as *noEFR*.

The performance of the proposed methodology is compared with state-of-the-art approaches with respect to the filtered DNS on the coarse grid, where the filter is a simple face-average [40]. The accuracy is measured as the discrepancy with respect to the filtered DNS in the global kinetic energy $\mathcal{E}^n_h$, in the global enstrophy $\mathcal{Z}^n_h$, for $n = 1, \dots, N$, and in the energy spectrum. More in detail, we employ a range of initial conditions to evaluate the robustness of the methods, and we analyze the results in terms of averages and 95 % confidence intervals across multiple initializations.

Table 4.2 shows train and test settings adopted for the different methodologies, specifying the training time window T_{train}, the simulation time T, the number of training and test initializations (I_{train} and I_{test}, respectively), and the corresponding results' sections. We also specify that T_{train} has different meanings for DD-EF(R) methods and for state-of-the-art approaches (Standard EFR, Standard EF, and Smagorinsky). On the one hand, in the DD-EF(R) methods, $[0, T_{\text{train}}]$ is the simulation time window used to collect the snapshots' matrices $\mathbf{U}^{\text{true}}$ and $\mathbf{W}^{\text{true}}$ employed in the optimization problem (Section 4.3.1). On the other hand, for state-of-the-art approaches, it is the simulation time window considered to perform hyperparameter tuning. For more details about the state-of-the-art approaches, we refer the reader to Appendix C.1.

Simulation type	**Description**	**Resolution**	**Mesh size**
DNS	Fully-resolved simulation of all turbulence scales	512^2	1.95×10^{-3}
Filtered DNS	DNS with small scales removed via explicit filter	128^2	7.81×10^{-3}
noEFR	DNS on coarse grid (i.e., no filter, no regularization)		
Standard EF	Standard approach for *Evolve–Filter* using the differential filter in Equation (4.13) (with optimized δ)		
Standard EFR	Standard approach for *Evolve–Filter–Relax* using the differential filter in Equation (4.13) (with prescribed $\delta = h$ and optimized χ)		
Smagorinsky	Smagorinsky model (with optimized parameter θ)		
DD-EF	*Evolve–Filter* using the data-driven filter introduced in Section 4.3.1		
$\mathcal{E}$-DD-EFR	Energy-conserving *Evolve–Filter–Relax* with: • the data-driven filter introduced in Section 4.3.1; • the relax parameter adjusted to ensure energy dissipation, as in Section 4.3.2.		
$\mathcal{EZ}$-DD-EFR	Energy and enstrophy-conserving *Evolve–Filter–Relax* with: • the data-driven filter introduced in Section 4.3.1; • the relax parameter adjusted to ensure energy dissipation as in Section 4.3.2 and enstrophy dissipation as in Section 4.3.3.		

Table 4.1: Brief description of the different types of simulation, including the details on the grids employed.

Method	Train		Test		Section(s)
	T_{train}	I_{train}	T	I_{test}	
State-of-the-art methods	1 s	10	10 s	5	Appendix C.1
DD-EF, $\mathcal{E}$-DD-EFR	1 or 3 s	10 (full data)			4.4.1.1, 4.4.2.1
DD-EF, $\mathcal{E}$-DD-EFR, $\mathcal{EZ}$-DD-EFR		1 (scarce data)			4.4.1.2, 4.4.2.2

Table 4.2: Train and test setup for DD-EF(R) and for state-of-the-art approaches, and the corresponding results' sections where the methods are applied.

4.4.1 Test case 1: two-dimensional decaying homogeneous turbulence

The first test case is decaying homogeneous isotropic turbulence, already treated in [40, 49, 156]. The initial DNS velocity is sampled from a random velocity field with prescribed energy spectrum, as in [167, 168]. More in detail, as in [40], the velocity field is initialized with the following procedure:

1. The velocity is sampled in spectral space to match a prescribed energy $\hat{\mathcal{E}}_\kappa$ at each wave number κ. Each component of the spectral velocity is indeed initialized such that $\|\hat{\mathbf{u}}_\kappa\| = |a_\kappa|$, where $a_\kappa = \sqrt{2\hat{\mathcal{E}}_\kappa}\mathrm{e}^{2\pi i \tau_\kappa}$ and τ_κ is a random phase shift.

2. The velocity is then projected to make it divergence-free: $\hat{\mathbf{u}}_\kappa = \frac{a_\kappa \hat{\boldsymbol{P}}_\kappa \boldsymbol{e}_\kappa}{\hat{\boldsymbol{P}}_\kappa \boldsymbol{e}_\kappa}$, where $\hat{\boldsymbol{P}}_\kappa$ is a projector: $\hat{\boldsymbol{P}}_\kappa = I - \frac{\kappa \kappa^T}{\kappa^T \kappa}$ and $\boldsymbol{e}_\kappa$ is a random unit vector. In 2D $\boldsymbol{e}_\kappa = (\cos(\theta_\kappa), \sin(\theta_\kappa))$ with $\theta_\kappa \sim \mathcal{U}[0, 2\pi]$.

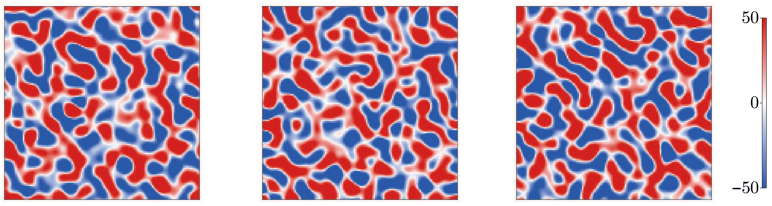
3. The velocity field is finally obtained by a transformation into physical space ($\mathbf{u}_h^1 = \mathrm{FFT}^{-1}(\hat{\mathbf{u}})$), and a further projection step such that $\mathbf{M}_h \mathbf{u}_h^1 = \mathbf{0}$.

Following the notation in subsection 4.3.1, the initial velocity field can be written as $\mathbf{u}_h^1(\boldsymbol{\mu}^{(j)})$, where the superscript j represents the random seed used to generate θ_κ and τ_κ for each wavenumber κ. Three examples of random initializations are represented in Figure 4.2.

4.4.1.1 Data-driven EFR in the case of full data

This part of the manuscript is dedicated to the results of DD-EF and $\mathcal{E}$-DD-EFR approaches (presented in Sections 4.3.1 and 4.3.2, respectively).

The first *offline* step involves constructing the filter matrix in the frequency domain (Equation (4.28)), considering a full data regime ($I_{\text{train}} = 10$ random initializations). Additionally, we remark that the snapshot data is subsampled by retaining one snapshot every 10 time steps to reduce the computational burden.



(a) $\mathbf{B}_h\mathbf{u}_h^1(\boldsymbol{\mu}^{(1)})$ (b) $\mathbf{B}_h\mathbf{u}_h^1(\boldsymbol{\mu}^{(5)})$ (c) $\mathbf{B}_h\mathbf{u}_h^1(\boldsymbol{\mu}^{(10)})$

Figure 4.2: Vorticity fields for initial time step, at different random initializations.

We analyze the effect of the filter on the velocity field in Figure 4.3 by considering the 2D average of the filter elements $\hat{\boldsymbol{\mathcal{F}}}^\star$ for both x and y components. We first associate an average filter value to all modes κ with wavenumber magnitude $\kappa = \|\boldsymbol{k}\| = \sqrt{k_x^2 + k_y^2}$. Such an average value is computed as the mean magnitude of the filter coefficients $\hat{f}_i^\star$ across all modes:

$$\langle|\hat{\boldsymbol{\mathcal{F}}}^\star(\kappa)|\rangle = \frac{1}{|\mathcal{B}_\kappa|}\sum_{i=1}^{|\mathcal{B}_\kappa|}\hat{f}_i^\star\mathcal{M}_i(\kappa), \tag{4.37}$$

where $\mathcal{B}_k$ collects the modes satisfying $\kappa - 0.5 < \|\boldsymbol{k}\| < \kappa + 0.5$, while $\mathcal{M}_i$ is a mask defined as:

$$\mathcal{M}_i(\kappa) = \begin{cases} 1 \text{ if } \kappa - 0.5 < \|\boldsymbol{k}_i\| < \kappa + 0.5, \\ 0 \text{ otherwise,} \end{cases}$$

and $|\mathcal{B}_\kappa| = \sum_i \mathcal{M}_i(\kappa)$ is the number of Fourier modes in $\mathcal{B}_\kappa$.

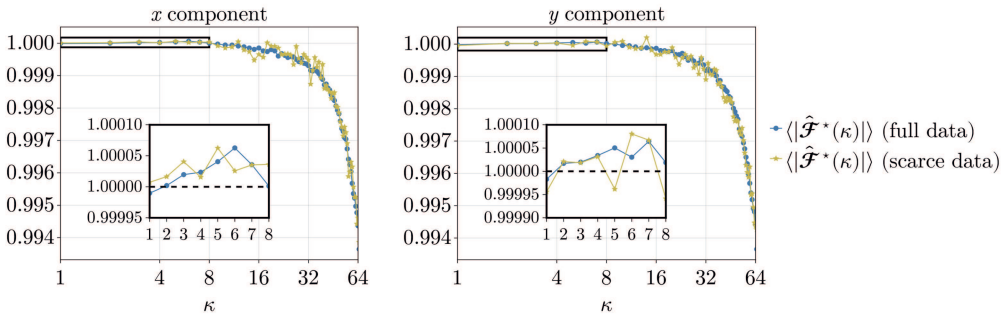


Figure 4.3: 2D average of components x and y of matrix $\hat{\boldsymbol{\mathcal{F}}}^\star$, for the cases $T_{\text{train}} = 3$ s and $I_{\text{train}} = 10$ (full data), and $T_{\text{train}} = 1$ s and $I_{\text{train}} = 1$ (scarce data).

Values greater than one indicate that the filter amplifies the corresponding frequency components, while values smaller than one correspond to attenuation. A magnitude close to one implies that the filter leaves those frequencies mostly unchanged. The plot shows that

the filter slightly amplifies the low frequencies, which are the highest-energy ones, while it attenuates the medium-to-high frequencies. The data-driven filter is highly influenced by the amount of data considered in the least squares problem. Indeed, the plot highlights that the filter built in the *data scarcity* regime presents more oscillations and the optimization may converge to values bigger than one, also at medium frequencies ($\kappa = 9, 10, 14$ for the y component). The energy injection at medium scales may increase the velocity magnitude significantly, and the simulation might blow up. This instability may be overcome by introducing the energy and/or enstrophy dissipation constraints in the relax step, as presented in Section 4.3.2 and 4.3.3. We will analyze the combined effect of such constraints in Section 4.4.1.2.

After building the filter, we perform the *online* EFR simulations. In particular, Figure 4.4 shows the performance of the method in terms of the total kinetic energy and enstrophy trend in time, and of the energy spectrum at fixed time $t = 1$ s. In Figures 4.5 and 4.6 we can see the corresponding vorticity fields at fixed times $t = 0.3$ and $t = 1$ s.

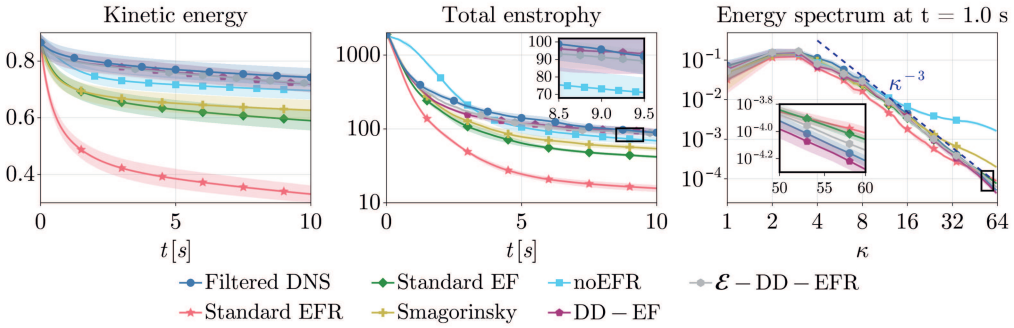


Figure 4.4: Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed methodologies (DD-EF and $\mathcal{E}$-DD-EFR) and for state-of-the-art approaches. In particular, we show the case $T_{\text{train}} = 3$ s. For each method, the figure shows the average values (solid lines) and the 95% confidence interval among 5 test configurations, in the decaying turbulence test case.

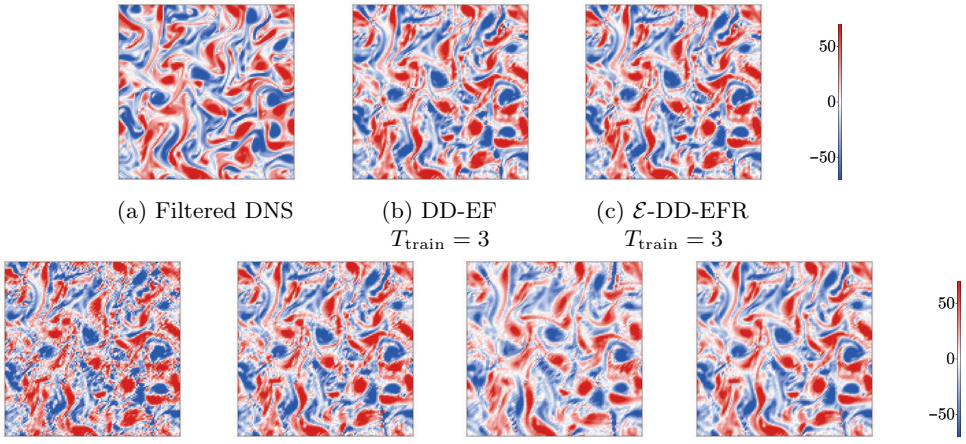
From the plots, we can draw the following conclusions:

- DD-EF and $\mathcal{E}$-DD-EFR outperform all existing methods in terms of kinetic energy, global enstrophy, and spectrum. The DD-EF and $\mathcal{E}$-DD-EFR vorticity fields at initial times are close to the filtered DNS in local features (Figure 4.5b and 4.5c), while being *qualitatively similar* to the reference at later times (Figure 4.6b and 4.6c).

- The DD-EF and $\mathcal{E}$-DD-EFR methods exhibit comparable performance when trained in the full data regime. Using multiple random initializations and a long training time window helps stabilize the filter's effect. In fact, this approach inherently yields

decaying energy and enstrophy, without the need for explicit constraints.

- The most accurate methods in the spectrum are the DD-EF(R) approaches and standard EF (at fixed $\delta = \delta_{\text{opt}}$).
- All state-of-the-art methods appear overdiffusive (as seen in the analysis in Figure 7.10), while the data-driven filter successfully retrieves the dissipated energy. From a qualitative point of view, the vorticity fields look all close to the reference at initial times (Figure 4.5f, 4.5g, and 4.5h), but overdiffusive at later time $t = 1$ s (Figure 4.6f, 4.6g, and 4.6h).
- The noEFR simulation exhibits large enstrophy values in the time window $[0, 3]$ s. This is also reflected in the presence of noise and wiggles in the vorticity field (Figures 4.5e and 4.6e). As time evolves and turbulence is dissipated, the noEFR enstrophy decreases and closely follows the filtered DNS behaviour. However, the noEFR energy is the closest one to the filtered DNS in the global kinetic energy with respect to traditional filters. Therefore, relying solely on global kinetic energy as a metric can be misleading, as it may conceal deficiencies in accurately capturing the flow's fine-scale structures and local dynamics, which are better reflected in enstrophy and vorticity-related quantities.

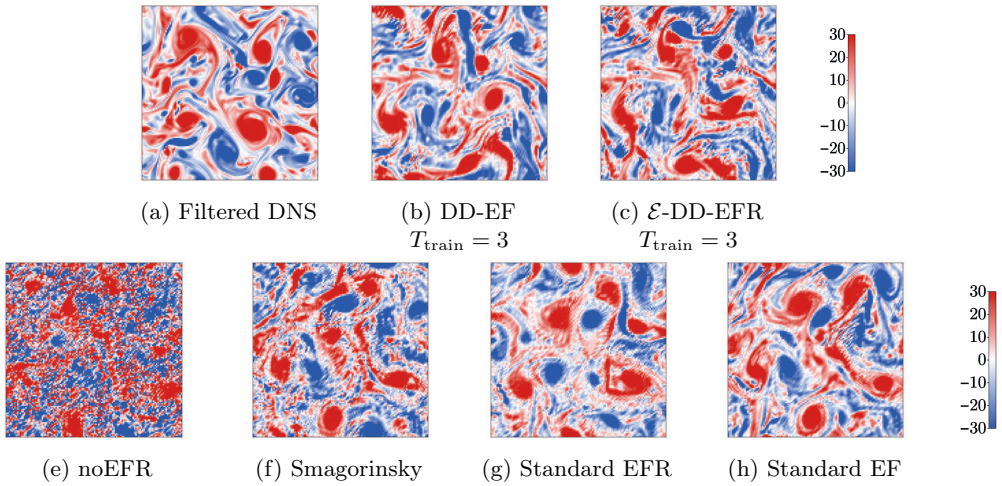


(a) Filtered DNS (b) DD-EF $T_{\text{train}} = 3$ (c) $\mathcal{E}$-DD-EFR $T_{\text{train}} = 3$

(e) noEFR (f) Smagorinsky (g) Standard EFR (h) Standard EF

Figure 4.5: Vorticity fields at $t = 0.3$ for different methodologies in a test configuration, in the decaying turbulence test case.

It is also useful to analyze the online time evolution of the optimal relaxation parameter $\chi_{\mathcal{E}\text{-opt}}$. Figure 4.7 shows the distribution and time history of $\chi_{\mathcal{E}\text{-opt}}$ in the case of $T_{\text{train}} = 3$ s, for a specific test initialization. $\chi_{\mathcal{E}\text{-opt}}$ is mostly oscillating between its admissible lower and upper bounds, namely 0 and 1. As pointed out in Section 4.3.2, the enforcement of the



(a) Filtered DNS (b) DD-EF $T_{\text{train}} = 3$ (c) $\mathcal{E}$-DD-EFR $T_{\text{train}} = 3$

(e) noEFR (f) Smagorinsky (g) Standard EFR (h) Standard EF

Figure 4.6: Vorticity fields at $t = 1$ for different methodologies in a test configuration, in the decaying turbulence test case.

energy constraint implies that, whenever $\tilde{b}_n > 0$, we impose $\chi_{\mathcal{E}\text{-opt}} = 0$, which is the only admissible value satisfying the constraint and corresponds to noEFR. Additionally, we can see that the most frequent value ($\sim 85\%$) is $\chi_{\mathcal{E}\text{-opt}} = 1$, which coincides to DD-EF. This confirms the similar performances of the methods observed in Figure 4.4.

When analyzing the qualitative features at later times (Figure 4.6), we observe more coherent vortex structures in DD-EF and $\mathcal{E}$-DD-EFR with respect to state-of-the-art approaches. This is confirmed by the spectral analysis at $t = 1\,s$ in Figure 4.4. DD-EF and $\mathcal{E}$-DD-EFR models show a better match with the filtered DNS in terms of energy at intermediate-to-high wavenumbers, indicating reduced numerical dissipation and improved preservation of vortex structures. Smagorinsky retains excessive energy at the large wavenumbers, while standard EFR exhibits a steeper decay in the spectrum at intermediate wavenumbers.

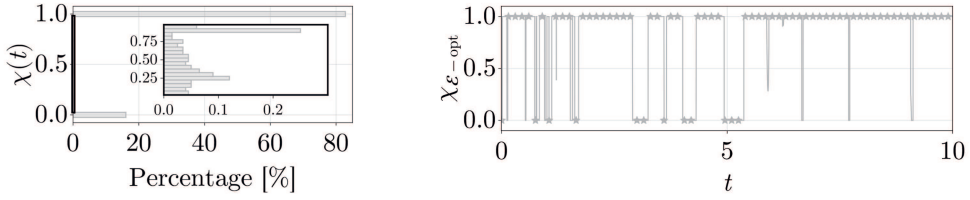


Figure 4.7: Distribution and time history of the optimal relax parameter $\chi_{\mathcal{E}\text{-opt}}$ for a test initialization. The plots refer to the case $T_{\text{train}} = 3$ s, $I_{\text{train}} = 10$ in the decaying turbulence test case.

Additional results about the data-driven approaches in the full data regime in case of a

smaller training time window ($T_{\text{train}} = 1$ s), are available in Appendix C.2.1.

4.4.1.2 Data-driven EFR in the case of scarce data

We now analyze the performance of the methods for scarce data, namely when the filter is computed from a single random initialization ($I_{\text{train}} = 1$).

Figure 4.8 presents the statistics for kinetic energy, enstrophy, and energy spectrum at fixed time, and we focus on the case $T_{\text{train}} = 1$ s, which is the most challenging one.

Figure 4.8 highlights the following aspects:

- DD-EF is unstable both in terms of energy and enstrophy, which increase and blow up after $t = 5$ s. It confirms the fact that the scarce-data-filter can be unstable, as it can also inject energy at medium wave numbers, as previously shown in Figure 4.3.
- $\mathcal{E}$-DD-EFR, which satisfies the energy dissipation constraint, presents a stable decreasing energy behaviour, but it is inaccurate in the global enstrophy after $t = 6$ s.
- $\mathcal{EZ}$-DD-EFR is stable both in the energy and in the enstrophy. It provides an accurate approximation of the filtered DNS in the enstrophy. However, both $\mathcal{E}$-DD-EFR and $\mathcal{EZ}$-DD-EFR lead to overdissipative results in the energy at large times. This is a consequence of using too little training data for the filter construction.

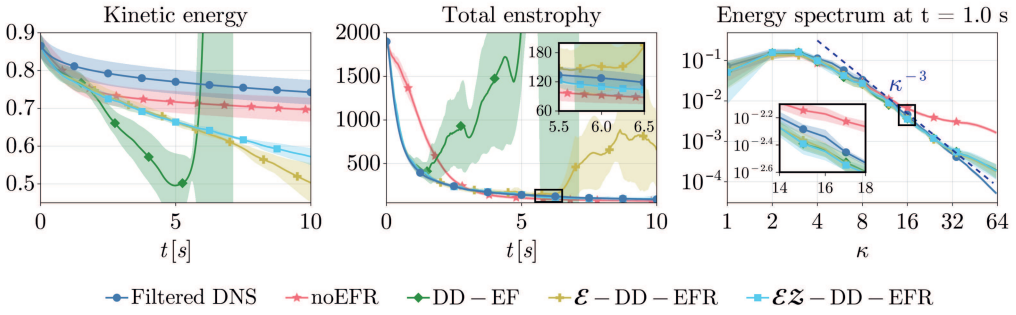


Figure 4.8: Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed DD-EF(R) methodologies and for state-of-the-art approaches. In particular, we show the case $T_{\text{train}} = 1$ s and $I_{\text{train}} = 1$. For each method, the Figure shows the average values (solid lines) and the 95% confidence interval among 5 test configurations, in the decaying turbulence test case.

Figure 4.9 illustrates the time evolution of $\chi_{\mathcal{E}\text{-opt}}$ and $\chi_{\mathcal{EZ}\text{-opt}}$ for a test initialization. $\chi_{\mathcal{E}\text{-opt}}$ exhibits stronger temporal oscillations with respect to the previous full data case. Indeed, the use of a reduced amount of data leads to increasing instability and error of the

filtering approach in terms of kinetic energy and enstrophy, as previously noticed from the analysis of the filter in Figure 4.3.

Moreover, since $\chi_{\mathcal{EZ}\text{-opt}}$ fulfills both energy and enstrophy dissipation constraints, it has more temporal variability. $\chi_{\mathcal{EZ}\text{-opt}}$ also assumes more values within the interval $(0, 1)$ especially at $t > 5$ s, while $\chi_{\mathcal{E}\text{-opt}}$ mostly converges to either 0 (noEFR) or 1 (DD-EF).

We can conclude that the scarce-data filter may lead to unstable results due to the small amount of data, but the imposition of the energy and/or enstrophy constraints (Equations (4.32) and (4.36), respectively) stabilizes the simulation. It is worth remarking that such constraints are only imposed *online* during the simulation by modifying the relax step, and without affecting the filter itself. This highlights the *flexibility* of the approach, allowing stability to be enforced without altering the underlying filter design.

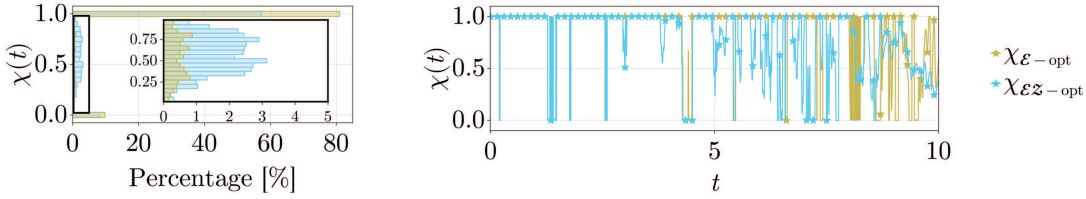


Figure 4.9: Distribution and time history of the optimal relax parameter $\chi_{\mathcal{E}\text{-opt}}$ and $\chi_{\mathcal{EZ}\text{-opt}}$ for a test initialization. The plots refer to the case $T_{\text{train}} = 1$ s, $I_{\text{train}} = 1$ in the decaying turbulence test case.

Additional results on data-driven approaches in the scarce data regime in the case of a larger training time window ($T_{\text{train}} = 3$ s), are available in Appendix C.2.1.

Given the observed differences in performance between filters trained with full and scarce data, the questions we will try to address are: *(i)* "*How do the scarce-data and full-data filters compare in terms of overall accuracy across different physical metrics?*", and *(ii)* "*To what extent can constraints bridge the gap in terms of overall error?*". To answer these questions, we analyze the following error metrics.

- Relative errors in global energy and enstrophy averaged in time, computed as:

$$\begin{aligned} \text{err}_{\mathcal{E}}(\boldsymbol{\mu}^{(i)}) &= \frac{1}{N_{t=3}} \sum_{n=1}^{N_{t=3}} \left| \frac{\mathcal{E}_h^n(\boldsymbol{\mu}^{(i)}) - \mathcal{E}_{\text{ref}}^n(\boldsymbol{\mu}^{(i)})}{\mathcal{E}_{\text{ref}}^n(\boldsymbol{\mu}^{(i)})} \right|, \\ \text{err}_{\mathcal{Z}}(\boldsymbol{\mu}^{(i)}) &= \frac{1}{N_{t=3}} \sum_{n=1}^{N_{t=3}} \left| \frac{\mathcal{Z}_h^n(\boldsymbol{\mu}^{(i)}) - \mathcal{Z}_{\text{ref}}^n(\boldsymbol{\mu}^{(i)})}{\mathcal{Z}_{\text{ref}}^n(\boldsymbol{\mu}^{(i)})} \right|, \end{aligned} \tag{4.38}$$

  where $\boldsymbol{\mu}^{(i)}$ represents the i-th test random initialization, $\mathcal{E}_{\text{ref}}^n$ and $\mathcal{Z}_{\text{ref}}^n$ are the energy and enstrophy of the filtered DNS solution at time step t_n.

- Absolute logarithmic errors in the energy spectrum averaged in the wave numbers and in time, computed as:

$$\mathrm{err}_{\mathrm{spectrum}} = \frac{1}{N_{t=3}} \sum_{n=1}^{N_{t=3}} \frac{1}{K} \sum_{\kappa=1}^{K} \log_{10} \left(\frac{\mathcal{E}_h^n(\kappa)(\boldsymbol{\mu}^{(i)})}{\mathcal{E}_{\mathrm{ref}}^n(\kappa)(\boldsymbol{\mu}^{(i)})} \right). \tag{4.39}$$

The value $N_{t=3}$ in expressions (4.38) and (4.39) is the number of time steps corresponding to $t = 3$ s. We choose the time window $[0, 3]$ for error computation because in the first test case, the energy dissipates significantly at later times, and the flow loses its turbulent characteristics.

The plots in Figure 4.10 display the above-mentioned errors in their average value and confidence interval among $I_{\mathrm{test}} = 5$ initializations.

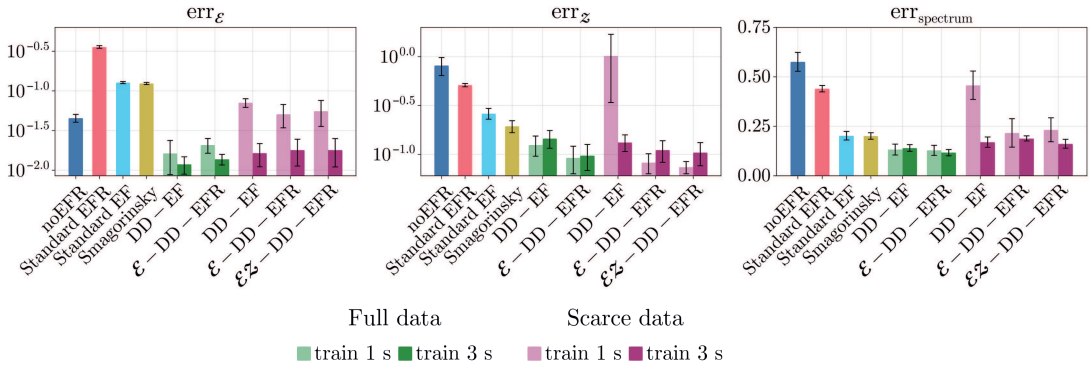


Figure 4.10: Relative error of the total kinetic energy, total enstrophy, and absolute error in logarithmic scale of the spectrum, of the different methodologies with respect to the filtered DNS. The errors are averaged in the time interval $[0, 3]$, and then evaluated as average values and confidence intervals (error bars) among $I_{\mathrm{test}} = 5$ test configurations. This figure refers to the decaying turbulence test case.

From Figure 4.10, we can draw the following conclusions:

- DD-EF(R) approaches with $T_{\mathrm{train}} = 3$ s are the best performing methods in all the metrics considered.
- Among the data-driven methods, DD-EF and $\mathcal{E}$-DD-EFR with $T_{\mathrm{train}} = 3, I_{\mathrm{train}} = 10$ (full data) reach the highest precision with respect to the filtered DNS in terms of global energy and spectrum.
- As previously noticed from Figure 4.8, $\mathcal{E}$-DD-EFR and $\mathcal{E}\mathcal{Z}$-DD-EFR when $T_{\mathrm{train}} = 1, I_{\mathrm{train}} = 1$ (scarce data) significantly improve the DD-EF results, especially in the enstrophy. On the other hand, only a slight improvement is observed when

$T_{\text{train}} = 3, I_{\text{train}} = 1$, likely because the filter, built from a larger dataset, exhibits greater stability.

- In both full and scarce data, we obtain the largest accuracy in energy and spectrum when $T_{\text{train}} = 3$ s, as it aligns with the increased amount of data. However, we have more accurate results in enstrophy when $T_{\text{train}} = 1$ s. A possible explanation is that the system exhibits more chaotic behavior in the initial phase, while dissipation dominates as time progresses. Therefore, a filter built using data from a shorter time window may be more accurate in capturing gradients compared to those constructed over longer time intervals.

Hence, we can conclude that the answers to the previous questions are:
(i) In general, full-data filters are more accurate in terms of kinetic energy and energy spectrum.
(ii) In the scarce-data regime, the constraints significantly improve the global enstrophy accuracy, reaching even better results than full-data regimes.

4.4.2 Test case 2: two-dimensional Kolmogorov flow

The second test case has a similar setting to the first one, but in the presence of a forcing term. To prevent energy decay during long simulations, we add a Kolmogorov-type body force to inject energy into the system at specific wave numbers, as in [39, 40, 49, 111]. This force is only characterized by a horizontal component

$$f_x(y) = c_f \sin(2\pi k_f y),$$

where $k_f = 4$ is the wave number at which the energy is injected. The forcing amplitude c_f is selected to ensure that the system attains a statistically stationary energy level sufficiently high to sustain turbulence without leading to unbounded energy growth. In our case $c_f = 0.65$.

To test extrapolation of our approach across test cases, in the DD-EF(R) we keep the filter matrix $\hat{\boldsymbol{\mathcal{F}}}^{\star}$ computed in the first test case, effectively performing *extrapolation in this test scenario* as well.

For the state-of-the-art methods, we also consider the optimal parameters computed in the first test case.

4.4.2.1 Data-driven EFR in the case of full data

As done for the first test case in Section 4.4.1.1, we dedicate this Section to discuss the DD-EF and $\mathcal{E}$-DD-EFR approaches in the full data regime ($I_{\text{train}} = 10$). For the sake of brevity, we only consider the case $T_{\text{train}} = 3$ s.

In particular, we analyze the time evolution of the global energy and enstrophy, and the spectrum at fixed time $t = 1$ s, comparing such metrics with the filtered DNS reference (Figure 4.11).

The figure shows that:

- The DD-EF and $\mathcal{E}$-DD-EFR provide the best-performing solutions in all metrics considered. While both have similar accuracy in energy and enstrophy, $\mathcal{E}$-DD-EFR is closer to the reference in the spectrum, especially at large wave numbers.
- As expected, the reference kinetic energy in the Kolmogorov case remains stable in time, while state-of-the-art methods provide overdiffusive solutions, as in the first test case.
- The reference global enstrophy has a similar trend as before, mostly decreasing in time. As for the previous test case, the noEFR solution is characterized by larger gradients and enstrophy at the initial time, which is dissipated at $t > 3$ s.

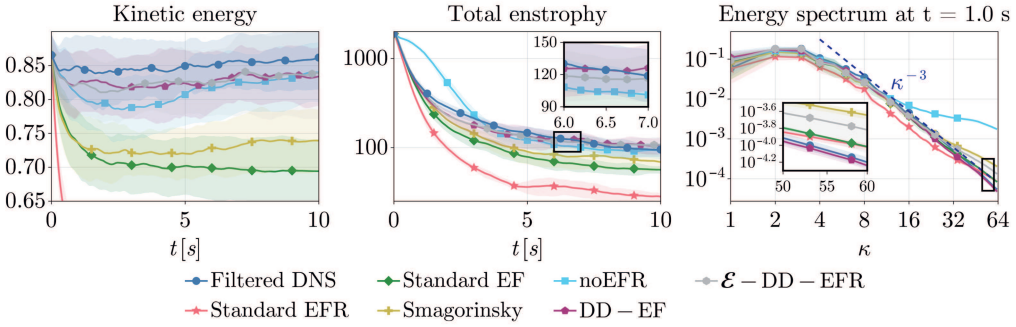


Figure 4.11: Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed DD-EF(R) methodologies and for state-of-the-art approaches. In particular, we show the case $T_{\text{train}} = 3$ s, $I_{\text{train}} = 10$. For each method, the figure shows the average values (solid lines) and the 95% confidence interval among 5 test configurations, in the Kolmogorov test case.

The time evolution of $\chi_{\mathcal{E}\text{-opt}}$ in the $\mathcal{E}$-DD-EFR approach is similar to the one obtained in the previous test case (Figure 4.7). The same holds for the qualitative behavior of the solutions, which the reader can find in Appendix C.2.2.

4.4.2.2 Data-driven EFR in the case of scarce data

In this part, we show the results when training the filter with scarce data (one random initialization), and we focus on $T_{\text{train}} = 3$ s.

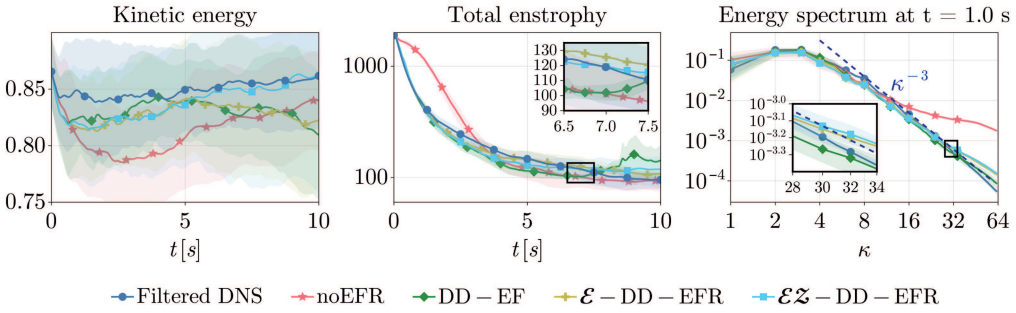


Figure 4.12: Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed DD-EF(R) methodologies and for state-of-the-art approaches. In particular, we show the case $T_{\text{train}} = 3$ s, $I_{\text{train}} = 1$. For each method, the figure shows the average values (solid lines) and the 95% confidence interval among 5 test configurations, in the Kolmogorov test case.

Different from what was observed in Section 4.4.1.2, where we considered the case $T_{\text{train}} = 1$ s, Figure 4.12 shows more accurate and less dissipative DD-EF(R) solutions. This is consistent with the increased amount of time instances used to compute the filter. More in detail, DD-EF and $\mathcal{E}$-DD-EFR have similar energy and enstrophy time evolution. However, DD-EF exhibits instabilities and inaccuracies in the global enstrophy, especially at $t > 6$ s. The addition of the energy and enstrophy constraints stabilizes the enstrophy at large time and better matches the energy spectrum at medium wave numbers. The further addition of the enstrophy constraint improves the accuracy in the kinetic energy, especially after $t = 5$ s.

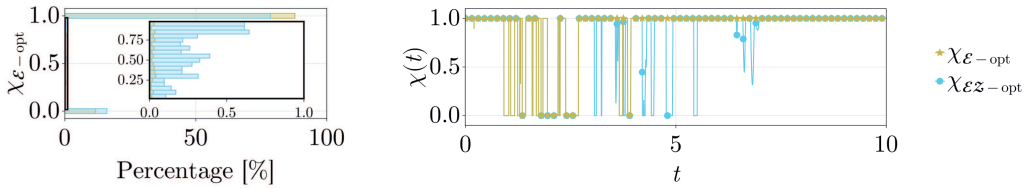


Figure 4.13: Distribution and time history of the optimal relax parameter $\chi_{\mathcal{E}\text{-opt}}$ and $\chi_{\mathcal{E}\mathcal{Z}\text{-opt}}$ for a test initialization. The plots refer to the case $T_{\text{train}} = 3$ s, $I_{\text{train}} = 1$ in the Kolmogorov test case.

The time evolution of the optimal $\chi_{\mathcal{E}\text{-opt}}$ and $\chi_{\mathcal{E}\mathcal{Z}\text{-opt}}$ is displayed in Figure 4.13. As in Figure 4.9, in both cases the optimal value is mostly $\chi = 1$, but $\chi_{\mathcal{E}\mathcal{Z}\text{-opt}}$ has more oscillations in time, especially at large simulation time.

From the comparison between Figure 4.9 ($T_{\text{train}} = 1$ s) and 4.13 ($T_{\text{train}} = 3$ s), we can notice that a larger training window leads to increased stability, reduced oscillations in the optimal χ, and fewer intermediate values.

Based on the observed results, we can conclude that the novel filtering approach, originally computed on a different configuration, retains its effectiveness in the current test case, both in full and scarce data regimes. This validates the method's capability to *extrapolate* and *adapt* to qualitatively similar flow conditions.

As done for the first test case, we now compare the full and scarce data regimes in terms of global errors (expressions (4.38) and (4.39)). Figure 4.14 depicts the global error analysis for the Kolmogorov test case, which is similar to the one shown in 4.10. We specify that the total number of time steps considered in this case for error computation is N instead of $N_{t=3}$, and we only show the errors related to the best-performing DD-EF(R) strategies (namely, those with $T_{\text{train}} = 3$ s).

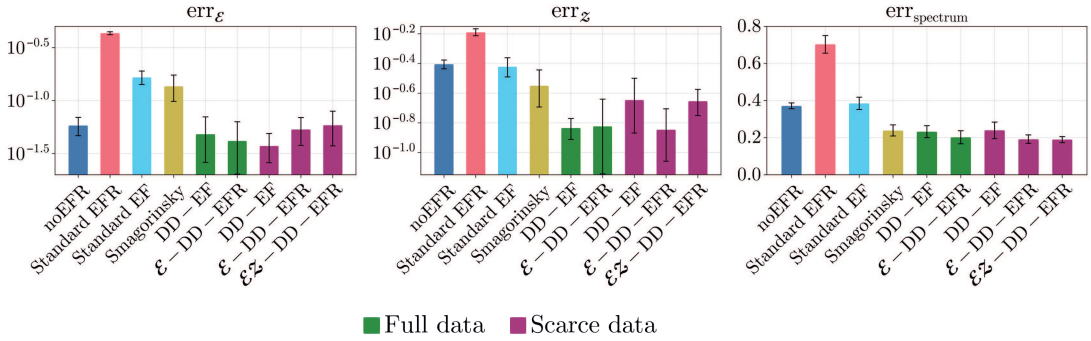


Figure 4.14: Relative error of the total kinetic energy, total enstrophy, and absolute error in logarithmic scale of the spectrum, of the different methodologies with respect to the filtered DNS. The errors are averaged in the time interval $[0, 10]$, and then evaluated as average values and confidence intervals (error bars) among 5 test configurations. This figure refers to the Kolmogorov test case.

The main difference with respect to Figure 4.10 lies in the noEFR accuracy, which improves when considering N instead of $N_{t=3}$. As it no longer exhibits spurious oscillations and noisy behavior at later times, it is qualitatively closer to the reference solution and has a smaller error.

4.4.3 Computational time considerations

In this Section, we discuss the computational cost required to construct the proposed methods and compare their efficiency with that of state-of-the-art approaches .

The offline and online wall times are reported in Table 4.3. We call *offline* wall time the time needed to perform the parameters' optimization in the case of state-of-the-art approaches

The simulations have been performed on an Apple M2 chip (8-core), 16GB unified memory.

(standard EFR/EF and Smagorinsky), and the time used to compute the filter matrix $\hat{\boldsymbol{\mathcal{F}}}^{\star}$ in DD-EF(R) methods. We refer to Appendix C.1 for more details on the parameters' tuning.

The offline wall time in DD-EF(R) includes:

- the time needed to evolve the filtered DNS solutions of one time step, namely to obtain the snapshot matrix W^{true};
- the time needed to solve the least squares minimization problem and find the filter components as in Equation (4.28).

We refer to *online* wall time as the time needed to perform the simulation at $T = 3$ s, with fixed $\Delta t = 5 \times 10^{-4}$ s, to have a fair comparison among the methods.

As from Table 4.3, the offline time needed to perform parameter tuning in state-of-the-art methods may be large, depending on the optimization strategy and the amount of data considered in the optimization. Here, we used an approximate gradient descent, although one may also rely on literature-based parameter choices to avoid this step.

On the other hand, the cost to compute the data-driven filter matrix is relatively small, as it only involves the resolution of a least-squares problem. We remark that the filter is the same in DD-EF, $\mathcal{E}$-DD-EFR and $\mathcal{EZ}$-DD-EFR approaches and the offline time only depends on the given amount of data (T_{train} and I_{train}) used to compute it, as specified in Table 4.3.

Method	**Offline wall time** [s]	**Online wall time** [s]
Filtered DNS	-	224.9
noEFR	-	13.5
Standard EFR/EF	$\mathcal{O}(10^5)$	213.4
Smagorinsky	$\mathcal{O}(10^2)$	20.6
DD-EF	3.8 ($T_{\text{train}} = 1$, $I_{\text{train}} = 1$)	18.3
$\mathcal{E}$-DD-EFR	42.7 ($T_{\text{train}} = 1$, $I_{\text{train}} = 10$)	25.7
$\mathcal{EZ}$-DD-EFR	12.1 ($T_{\text{train}} = 3$, $I_{\text{train}} = 1$) 145.4 ($T_{\text{train}} = 3$, $I_{\text{train}} = 10$)	33.9

Table 4.3: Offline and online wall times for the strategies considered. The wall time is measured for online simulations with $T = 3$ s, at fixed time step $\Delta t = 5 \times 10^{-4}$ s.

The online cost is significantly large for standard EFR/EF, since it involves the solution of the differential filter equation. As one would expect, noEFR is the fastest approach, since it does not include any filtering operation or closure terms, followed by DD-EF and Smagorinsky. The addition of the energy and enstrophy constraints in data-driven approaches slightly increases the computational time, but still remains in the same order of magnitude of DD-EF. It is important to note that the energy and enstrophy dissipation constraints depend solely on the data from the online simulation, and not on the reference filtered DNS. As

a result, the optimization of $\chi(t)$ in $\mathcal{E}$-DD-EFR and $\mathcal{EZ}$-DD-EFR introduces no significant computational overhead, since it only involves computing the system's energy and enstrophy.

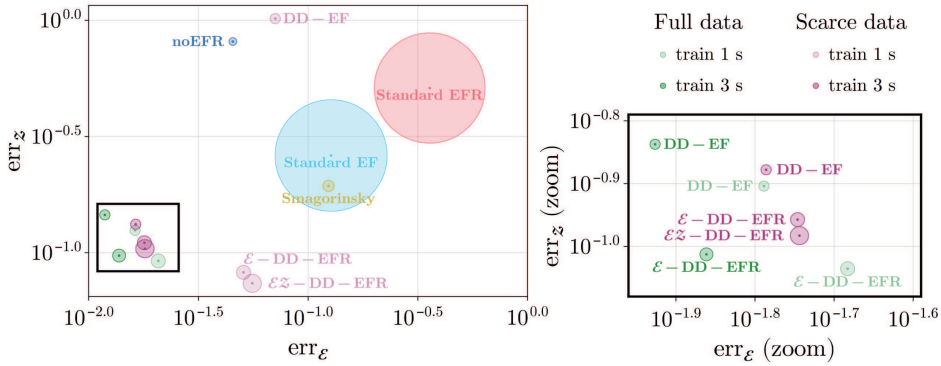


Figure 4.15: Average energy-enstrophy accuracy of the simulations, where the radius of circles is proportional to the average wall time needed to perform online simulations. The errors are measured in the decaying turbulence test case.

We include Figure 4.15 as a graphic comparison between the methods. The plot shows the energy and enstrophy errors on the x and y axis, respectively, as presented in (4.38). The average error points are surrounded by circles whose size is proportional to the online simulations' wall time. The most accurate methods are located in the bottom-left part of the plots, while the cheapest simulations have the smallest circles. Hence, an ideal approach is located in the bottom left part (high accuracy) and is characterized by a small circle (low wall time).

We can see that all DD-EF(R) methods, except for the case $T_{\text{train}} = 1$ s and $I_{\text{train}} = 1$ s (pink circles in the plot), exhibit good accuracy in both energy and enstrophy and are ~ 10 times faster with respect to standard EFR and EF, as already formalized in Table 4.3.

The DD-EF(R) simulations in the case of scarce data and $T_{\text{train}} = 1$ (pink circles) are less accurate with respect to the other data-driven approaches. However, the results highlight a large improvement in the enstrophy accuracy in the presence of energy and/or enstrophy constraints ($\mathcal{E}$-DD-EFR and $\mathcal{EZ}$-DD-EFR), while keeping a similar CPU time.

Based on the observed results, we can conclude that the novel DD-EF(R) methods outperform all the existing methods, not only in the accuracy but also in the computational effort.

4.5 Conclusion and Outlook

In this work, we proposed a novel data-driven extension of the EFR framework for coarse-grid simulation of turbulent flows.

First, we propose a new learned filter operator, computed through efficient offline least-squares optimization from filtered DNS data, and integrated into a structure-preserving staggered grid finite-volume discretization.

Second, we propose a novel way to choose the relaxation parameter $\chi(t)$, by enforcing global dissipation of energy and/or enstrophy during online simulations. The resulting $\mathcal{E}$-DD-EFR and $\mathcal{EZ}$-DD-EFR approaches ensure numerical stability and preserve key physical properties of the flow. Specifically, $\mathcal{E}$-DD-EFR enforces global energy dissipation at each time step, while $\mathcal{EZ}$-DD-EFR extends this approach by additionally enforcing enstrophy dissipation. These constraints act directly on the online simulation data and help stabilize the learned filtering strategy, making the methods robust, especially when the training data is scarce.

The proposed methodology was tested on two benchmark cases: decaying homogeneous isotropic turbulence and the Kolmogorov flow. In both scenarios, we employed the same filter operator, which was computed solely from the first test case. Despite this, the approach yielded excellent results also in the second case, clearly demonstrating the capability of the learned filter to successfully extrapolate across different flow configurations.

The proposed data-driven method led to significant improvements over traditional approaches, such as Smagorinsky and standard EFR strategies, particularly in terms of accuracy in energy spectra, global energy, and enstrophy.

Moreover, the methodology maintains low computational cost: the learned filter avoids the need for solving linear systems (as in the traditional differential filter framework), and the adaptive computation of the relaxation parameter relies solely on online simulation data. As such, the method proves both accurate and efficient, especially when compared with classical differential filtering techniques.

Future work will focus on extending the approach to non-periodic boundary conditions, unstructured grids, and three-dimensional flows, as well as integrating alternative learning strategies and uncertainty quantification tools to further enhance robustness and generalization.

Regarding three-dimensional flows, the presented methodology can be directly extended, as the formulation itself is dimension-independent. The main challenge lies in the increased computational cost associated with adding another spatial dimension.

For unstructured or non-uniform grids, the use of FFTs must be revised, as it relies heavily on the Cartesian nature of the grid. In the current implementation, the filter is learned and applied in Fourier space. For unstructured discretizations, a natural alternative is to express the filter in the eigenbasis of the discrete diffusion operator, whose eigenfunctions generalize the Fourier modes. However, projecting onto the full eigenbasis at each time step can be computationally expensive. To alleviate this cost, a reduced spectral basis can be employed, for example, obtained via singular value decomposition or by selecting a subset of dominant

eigenmodes. This approach significantly reduces the number of operations required to apply the filter, thereby mitigating the added computational burden.

Finally, to handle non-periodic boundary conditions, one could take inspiration from the approach proposed in [69], where lifting functions are used to satisfy the boundary constraints.

Acknowledgements

This project has been conducted as part of the CWI (Centrum Wiskunde & Informatica, Amsterdam) Internship Program 2025 and is partially funded by CWI. In addition, we acknowledge the support of the Dutch Research Council (NWO) under project number OCENW.GROOT.2019.044, the Eindhoven University of Technology, and INdAM-GNCS: Istituto Nazionale di Alta Matematica — Gruppo Nazionale di Calcolo Scientifico. We also acknowledge financial support from the European Union under the National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.1, Call for tender No. 1409 published on 14.9.2022 by the Italian Ministry of University and Research (MUR), funded by NextGenerationEU – Project Title ROMEU – CUP P2022FEZS3, Grant Assignment Decree No. 1379 adopted on 01.09.2023 by MUR, as well as support from the European Union (ERC, DANTE, GA-101115741).

The views and opinions expressed in this chapter are those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency; neither the European Union nor the granting authorities can be held responsible for them.

5

Modeling Advection-Dominated Flows with Space-Local Reduced-Order Models

Reduced-order models (ROMs) are often used to accelerate the simulation of large physical systems. However, traditional ROM techniques, such as proper orthogonal decomposition (POD)-based methods, often struggle with advection-dominated flows due to the slow decay of singular values. This results in high computational costs and potential instabilities.

This chapter proposes a novel approach using space-local POD to address the challenges arising from the slow singular value decay. Instead of global basis functions, our method employs local basis functions that are applied across the domain, analogous to the finite element method, but with a data-driven basis. By dividing the domain into subdomains and applying the space-local POD, we obtain a sparse representation that generalizes better outside the training regime. This allows the use of a larger number of basis functions compared to standard POD, without prohibitive computational costs. To ensure smoothness across subdomain boundaries, we introduce overlapping subdomains inspired by the partition of unity method.

Our approach is validated through simulations of the 1D and 2D advection equation. We demonstrate that using our space-local approach, we obtain a ROM that generalizes better to

This chapter is based on [169].

flow conditions not included in the training data. In addition, we show that the constructed ROM inherits the energy conservation and non-linear stability properties from the full-order model. Finally, we find that using a space-local ROM allows for larger time steps.

5.1 Introduction

Simulating large physical systems is an ongoing challenge in the field of computational sciences. This especially becomes challenging when dealing with multiscale systems. Such systems exhibit interesting behavior at various spatial and temporal scales. A prominent example and our main incentive for this work are turbulent flows described by the incompressible Navier-Stokes equations. Systems described by these equations feature the formation of turbulent eddies of a range of different sizes. The significant difference in size between the largest and the smallest eddies gives rise to the multiscale nature of the problem. The problem with simulating such systems is that they require high-resolution computational meshes to obtain accurate simulations and small time steps. This places a significant burden on the available computational resources [40, 82].

To make simulation of turbulent flows feasible, one typically resorts to Reynolds averaged Navier-Stokes [9], large eddy simulation [6], and reduced-order models (ROMs) [7, 170]. Here we focus on the latter approach. In reduced-order modelling, data is used to speed up simulations. The data can be obtained from simulations or experiments. The collected data is then used to identify the most critical features of the flow. This is typically done by a proper orthogonal decomposition (POD) of the collected flow data. The resulting features are then used to construct a reduced basis. By projecting the fluid flow equations on this basis one obtains the widely used POD-Galerkin ROM [68, 170, 171]. However, two common issues with POD-Galerkin ROMs are their stability and their accuracy for convection-dominated systems. In [67], it is shown that the stability issue can be resolved by making sure that the energy-conserving property of the Navier-Stokes equations is still satisfied under Galerkin projection. For the incompressible Navier-Stokes equations, the only prerequisite for an energy-conserving ROM is that the discretization is structure-preserving, i.e. it is such that it inherits the energy conservation property from the continuous equations.

However, the stability property derived in [67] does not guarantee accuracy; it is possible to have energy-stable simulations that are highly inaccurate. This issue can already be observed on academic test cases such as the linear advection equation. For example, in [7], the authors discuss the problem of a single traveling wave through a periodic domain. Even though it is clear that a one-dimensional representation of the system exists, it is not recovered by the POD algorithm. The result is a slow decay in singular values of the snapshot matrix. This means that many POD modes are required to accurately describe the flow. The slow decay in singular values is often related to the Kolmogorov N-width. This is a measure

of how well the solution space of a partial differential equation (PDE) can be represented by a linear combination of N basis functions [66, 70]. Advection-dominated flows are notorious for displaying a slow Kolmogorov N-width decay, requiring a large N for accurate simulation. The resulting ROM is then expensive to evaluate [19, 20]. When not including a sufficient number of basis functions, inaccurate results are obtained, e.g., they contain oscillations [71, 72].

Different approaches to deal with this problem have been suggested. One way of dealing with this is by taking a local in time approach [73]. In this approach, the ROM switches basis for different time intervals. This means that the ROM can employ a smaller basis, since each basis is specialized to deal with a specific time interval. Switching between bases during simulation can also be done in a structure-preserving (energy-conserving) manner [74]. A related approach is to update the basis during the ROM simulation to make it more robust to changes in simulation conditions [75]. In [76], a Petrov-Galerkin approach is suggested, which leads to more stable ROMs when using a small POD basis than the standard Galerkin approach. In [77], it is suggested to add a closure model to the ROM to account for a small POD basis. The closure model is then tasked with modeling the interaction between the part of the flow that is covered by the POD basis and the part that is removed. Another suggestion is to construct ROMs on non-linear reduced subspaces instead of linear ones. In [78], a more accurate representation of the solution space is obtained using rational quadratic manifolds. Here, the ROM construction was also performed in a structure-preserving (entropy-stable) manner, yielding stability of the ROM.

Machine learning approaches have also gained traction for the construction of ROMs. In [79], a long short-term memory (LSTM) neural network is used instead of Galerkin projection, as the computational complexity of LSTM is more favorable than Galerkin projection-based ROMs. They also allow one to take larger time steps [80]. In [80], this approach is combined with an autoencoder to reduce the dimensionality of the system. It is demonstrated that a significantly greater reduction in degrees of freedom (DOF) can be achieved using this approach compared to the standard POD-Galerkin approach. In [66], an advection-aware autoencoder is suggested to reduce the dimensionality of this system. This is achieved by introducing an additional decoder which is trained to reconstruct a “shifted" version of the encoded snapshots. This shift can, for example, be a snapshot taken at a later time. This forces the latent space of the autoencoder to become aware of the dominant advective features of the flow.

In this work, we focus on an alternative approach to address the issue of ROM accuracy in advection-dominated flows. Namely, we propose to use the idea of a space-local POD to tackle the slow singular value decay of advection-dominated flows. The idea is as follows: instead of obtaining a set of global basis functions that span the entire domain, we obtain a set of space-local basis functions. These basis functions are repeated across the entire

domain, similarly to how a finite element basis covers the domain [172]. Furthermore, they are only nonzero on their designated subdomain. The advantage of this is that the resulting Galerkin projected operators are sparse. This makes them much cheaper to evaluate when simulating the ROM. For example, in [173] a speedup of up to 1.5 orders of magnitude is reported with respect to a standard POD-Galerkin ROM. Our approach differs from the one presented in [173]. In our approach, the solution data in each subdomain is treated with the same local POD basis, rather than obtaining a different local POD basis for each subdomain. This approach ensures that a feature observed in part of the domain can also be represented in a different part of the domain using the same basis. In this way, less data is required to obtain a POD basis that generalizes well. Note that this approach is designed specifically for problems where the dynamics are similar throughout the domain, such as a traveling wave. For problems where the dynamics are more variable throughout the domain, the approach suggested by [173] is likely still superior, as it builds a specialized basis for each subdomain. For diffusion-dominated problems the standard space-global POD approach is likely still superior due to the rapid Kolmogorov N-width decay [174]. Note that while our approach does not directly solve the slow Kolmogorov N-width decay (we still use linear approximations), the sparsity of the basis allows us to use a much larger number of basis functions. Sparsity and generalizability are therefore key features of our approach. Furthermore, the authors note this approach is similar to the methodology described in [175] where different types of subdomains were specified, each with a shared POD basis. These subdomains were then used as building blocks to extrapolate the POD basis obtained from small spatial domains to larger domains. The work in this chapter differs in the fact that it focuses on temporal extrapolation for time-dependent PDEs, as opposed to steady-state problems. In addition, we introduce another novel idea in this work: the use of overlapping subdomains (to avoid discontinuities at the subdomain boundaries). Lastly, we show that our space-local ROMs satisfy energy conservation if the full-order model (FOM) does. In this way, stability of the ROM is guaranteed.

This chapter is structured in the following way. In Section 5.2 we introduce the FOM, namely a central difference discretization of the 1D advection equation. The central difference discretization ensures the scheme is energy conserving. In Section 5.3 we introduce the POD approaches. We begin with the standard POD approach and then introduce two space-local approaches, one with and one without overlapping subdomains. In Section 5.4 we use Galerkin projection to project the FOM onto the POD basis and show that the resulting ROMs satisfy energy conservation. Finally, in Section 5.5, we evaluate the different POD-based approaches in numerical experiments using a set of different metrics. In addition, we assess the performance of the ROMs on a case where we extrapolate beyond the data used to construct the POD basis. Furthermore, we investigate the energy-conserving properties of the ROMs. To conclude this section, we apply the introduced methodologies to a 2D

advection equation test case and evaluate the computational efficiency, among other metrics. Finally, in Section 5.6, we present our main findings and suggest future research topics.

5.2 Full-order model

5.2.1 Advection equation

In this work, we focus on developing a reduced-order model for the linear advection equation in both one-dimensional (1D) and two-dimensional (2D) settings. This equation is chosen as it exhibits similar difficulties as the incompressible Navier-Stokes equations when applying model reduction. For the sake of simplicity, we discuss the properties of the system for 1D, but the ideas easily carry over to 2D. To start, we consider a scalar solution $u(x,t)$ to the linear advection equation

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}, \tag{5.1}$$

with initial condition $u(x,0) = u_0(x)$ and constant c. In this work, we stick to periodic boundary conditions (BCs). An important property of this equation is that the total energy of the system

$$E := \frac{1}{2}(u,u), \tag{5.2}$$

is conserved, where

$$(a(x), b(x)) := \int_\Omega a(x)b(x)\mathrm{d}\Omega \tag{5.3}$$

on the spatial domain Ω. This can easily be shown using the product rule of differentiation:

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}(u,u) = \left(u, \frac{\partial u}{\partial t}\right) = -c\left(u, \frac{\partial u}{\partial x}\right) = c\left(\frac{\partial u}{\partial x}, u\right) = 0. \tag{5.4}$$

In the final step, we carried out integration by parts and used the fact that the boundary term cancels on periodic domains. The energy-conserving property of this equation will be mimicked by the discretization and by the ROMs developed in this work. This leads to unconditionally stable methods, as in [67].

5.2.2 Finite difference discretization

Although equation (5.1) can be easily solved exactly, for more complex equations, this is not the case. In general, we approximate the solution by representing $u(x,t)$ on a grid. In this case, we employ a uniform grid with grid spacing $h = \frac{|\Omega|}{N}$ such that $\mathrm{u}_i(t) \approx u(x_i,t)$. The approximated solution is contained within the state vector $\mathbf{u}(t) \in \mathbb{R}^N$. To approximate the

spatial derivative, we use a central difference approximation:

$$\frac{\partial u}{\partial x}|_{x_i} \approx \frac{\mathrm{u}_{i+1} - \mathrm{u}_{i-1}}{2h}. \tag{5.5}$$

This leads to the following semi-discrete system of equations

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -c\mathbf{D}\mathbf{u}, \tag{5.6}$$

where the linear operator $\mathbf{D} \in \mathbb{R}^{N\times N}$ is skew-symmetric and encodes the stencil in (5.5). For the time integration, we use a classic Runge-Kutta 4 (RK4) scheme [96]. This time integration scheme introduces a small energy conservation error, which is negligible in our test cases. Alternatively, one can use the implicit midpoint method for exact energy conservation in time [176]. Equation (5.6) will be regarded as the full-order model (FOM). We note that other discretization techniques, such as the finite element method, can also be employed to derive a FOM; our framework in Section 5.3 is also applicable in this context.

5.2.3 Energy conservation of the FOM

It is well known that this FOM mimics the energy-conservation property (5.4) in a discrete setting. This can be shown by defining the discretized energy as

$$E_h := \frac{h}{2}\mathbf{u}^T\mathbf{u}, \tag{5.7}$$

where the inclusion of h is such that this definition discretely represents the inner product in (5.2).

Using the product rule, we obtain the following evolution equation for the energy

$$\frac{\mathrm{d}E_h}{\mathrm{d}t} = \frac{h}{2}\frac{\mathrm{d}\mathbf{u}^T\mathbf{u}}{\mathrm{d}t} = h\mathbf{u}^T\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -ch\mathbf{u}^T\mathbf{D}\mathbf{u} = ch\mathbf{u}^T\mathbf{D}^T\mathbf{u} = 0. \tag{5.8}$$

In the final step, we used the fact that $\mathbf{D}$ is skew-symmetric, i.e., $\mathbf{D} = -\mathbf{D}^T$, to see that the energy is conserved using this stencil. This yields both stability and consistency with the continuous equation.

5.3 Local and global POD

To reduce the computational cost of solving the FOM, we construct a ROM. For this purpose, we use simulation data to build a data-driven basis. This basis is obtained through a POD of the simulation data [7, 170]. The most common approach is to employ a global POD basis, defined over the entire simulation domain, similar to a Fourier basis. An alternative is a

local basis, as proposed in [173, 175]. In this section, we discuss both approaches. We will present our version of the space-local POD framework for finite difference discretizations, see (5.5). However, the ideas can also be applied to different discretization techniques, as shown in [175]. Our space-local approach has some key differences from the one presented in [173], which will be highlighted.

5.3.1 Global POD

In the global approach we first express the discrete solution $u_h(x,t)$ in terms of an orthogonal basis $\{\psi_i\}$:

$$u(x,t) \approx u_h(x,t) = \sum_{i=1}^{N} \mathrm{c}_i(t)\psi_i(x). \tag{5.9}$$

Here ψ_i can represent, for example, a Fourier basis or a localized box function in the case of the presented finite difference discretization. In the finite difference case we have $\mathbf{c}(t) = \mathbf{u}(t)$ and

$$\psi_i(x) = \begin{cases} 1 & \text{if } x_i - \frac{h}{2} \leq x < x_i + \frac{h}{2}, \\ 0 & \text{elsewhere.} \end{cases} \tag{5.10}$$

For finite element discretizations, one could use, e.g., Löwdin orthogonalization to express the solution in terms of an orthogonal basis [177].

The snapshot matrix $\mathbf{X}$ is constructed from the coefficient vector at different points in time

$$\mathbf{X} = \begin{bmatrix} \mathbf{c}(t_1) & \mathbf{c}(t_2) & \dots & \mathbf{c}(t_s) \end{bmatrix} \quad \in \mathbb{R}^{N \times s}, \tag{5.11}$$

where s is the number of snapshots. We decompose this snapshot matrix using a singular value decomposition (SVD) [178]

$$\mathbf{X} = \tilde{\mathbf{\Phi}} \mathbf{\Sigma} \mathbf{V}^T. \tag{5.12}$$

We use the first r left-singular vectors in $\tilde{\mathbf{\Phi}} \in \mathbb{R}^{N \times N}$ to obtain a reduced set of orthonormal basis vectors $\mathbf{\Phi} \in \mathbb{R}^{N \times r}$. This basis minimizes the projection error in the Frobenius norm:

$$\mathbf{\Phi} = \underset{\mathbf{M} \in \mathbb{R}^{N \times r}}{\arg\min} ||\mathbf{X} - \mathbf{M}\mathbf{M}^T\mathbf{X}||_F^2, \tag{5.13}$$

under the orthonormality constraint $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}$ [7, 68, 170, 171]. We can project the coefficient vector onto this basis as follows:

$$\mathbf{a}(t) = \mathbf{\Phi}^T \mathbf{c}(t) \quad \in \mathbb{R}^r, \tag{5.14}$$

such that projecting back onto the FOM space yields the approximation

$$\mathbf{c}(t) \approx \mathbf{c}_r(t) := \mathbf{\Phi a}(t). \tag{5.15}$$

By substituting this into (5.9) we obtain the approximated solution $u_r(x,t)$:

$$u_h(x,t) \approx u_r(x,t) = \sum_{i=1}^{N} \mathrm{c}_{r,i}(t)\psi_i(x). \tag{5.16}$$

We can also write this approximation in terms of the POD basis expansion:

$$\begin{aligned} u_r(x,t) &= \sum_{i=1}^{N} \mathrm{c}_{r,i}(t)\psi_i(x) = \sum_{i=1}^{N} (\mathbf{\Phi a}(t))_i \psi_i(x) = \sum_{i=1}^{N} (\sum_{j=1}^{r} \Phi_{ij}\mathrm{a}_j(t))\psi_i(x) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{r} \Phi_{ij}\mathrm{a}_j(t)\psi_i(x) = \sum_{j=1}^{r} \mathrm{a}_j(t) \sum_{i=1}^{N} \Phi_{ij}\psi_i(x) = \sum_{j=1}^{r} \mathrm{a}_j(t)\phi_j(x), \end{aligned} \tag{5.17}$$

where the reduced POD basis $\{\phi_j\}$ is obtained by applying the following transformation:

$$\phi_j(x) = \sum_{i=1}^{N} \Phi_{ij}\psi_i(x). \tag{5.18}$$

As the obtained basis functions span the entire domain, we refer to this approach as space-global proper orthogonal decomposition (G-POD).

5.3.2 Space-local POD

In space-local proper orthogonal decomposition (L-POD) we take a different approach from G-POD. We start by assuming a uniform finite difference discretization for the discrete FOM solution $u_h(x,t)$. Once the solution is represented on this grid, we subdivide the domain Ω into I non-overlapping subdomains $\Omega_i = [\alpha_i, \beta_i)$, $i = 1 \dots I$, such that $\alpha_i < \beta_i = \alpha_{i+1} < \beta_{i+1}$. Later in this section, these subdomains will be used to construct the L-POD basis. Furthermore, we assume each of these subdomains contains exactly J grid points. This means the total number of grid points N has to equal the product $N = I \cdot J$. This is essential for our methodology to work, as for the L-POD procedure, each subdomain is treated equally in the snapshot matrix, which is only possible if they contain the same number of grid points. This helps us obtain a basis that generalizes better outside the training data for advection-dominated problems, where the dynamics are similar throughout the domain. This

If $u_h(x,t)$ would stem from a simulation on an unstructured grid, one could potentially first project the discrete solution on a uniform grid, after which the presented methodology could still be applied. We consider this outside the scope of this research.

approach is fundamentally different from what is presented in [173], where each subdomain has its own POD basis. The latter is more suitable for problems where the dynamics differ throughout the domain, such that the local bases can be specialized to these dynamics. If N/I is not an integer, one could possibly resolve this by projecting the FOM solution on a compatible grid of M grid points for which M/I is an integer.

As stated earlier, we use a common finite difference basis $\{\chi_{ij}\}$ of size J within each subdomain Ω_i to describe the solution:

$$u_h(x,t) = \sum_{i=1}^{I} \sum_{j=1}^{J} \mathrm{U}_{ij}(t)\chi_{ij}(x), \tag{5.19}$$

where χ_{ij} is only nonzero within Ω_i and $\mathbf{U}(t) \in \mathbb{R}^{I\times J}$ contains the coefficient values. Note that in this case $\mathbf{U}(t)$ is simply a reshaped version of the solution vector $\mathbf{u}(t)$. The finite difference basis functions are defined as

$$\chi_{ij}(x) = \begin{cases} 1 & \text{if } \alpha_i + (j-\frac{1}{2})h \leq x < \alpha_i + (j+\frac{1}{2})h, \\ 0 & \text{elsewhere,} \end{cases} \tag{5.20}$$

similarly to (5.10). An example for a Gaussian solution profile for $I = 3$ and $J = 8$ is given in Figure 5.1.

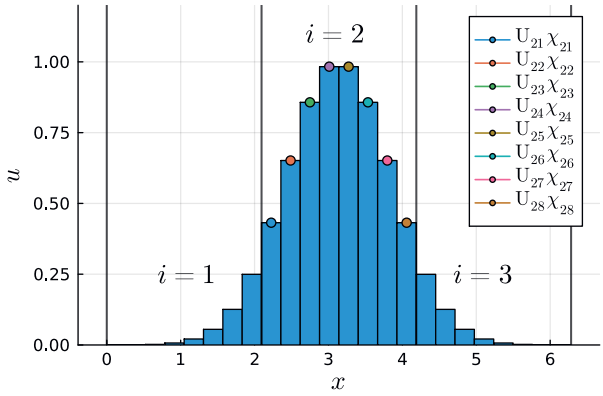


Figure 5.1: A Gaussian wave discretized by a finite difference scheme represented by a local basis of box functions for $I = 3$ subdomains and $J = 8$ points per subdomain. The edges of the subdomains are indicated by the vertical grey lines.

After obtaining $\mathbf{U}(t)$ at different points in time, the snapshot matrix is constructed as follows

$$\mathbf{X}_\ell = \begin{bmatrix} \mathbf{U}^T(t_1) & \mathbf{U}^T(t_2) & \dots & \mathbf{U}^T(t_s) \end{bmatrix} \quad \in \mathbb{R}^{J\times Is}. \tag{5.21}$$

In this way, each subdomain is treated equally in the snapshot matrix. This is what differentiates our work from [173] and similar to what is done [175]. A schematic representation of this is shown in Figure 5.2, where the G-POD snapshot matrix $\mathbf{X}$ is reshaped into the L-POD snapshot matrix $\mathbf{X}_\ell$. Using a single common snapshot matrix allows us to obtain a

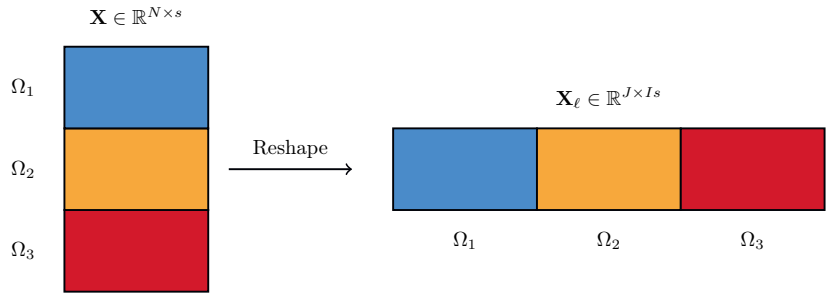


Figure 5.2: Schematic representation of the G-POD snapshot matrix $\mathbf{X}$ being reshaped into the L-POD snapshot matrix $\mathbf{X}_\ell$ for $I = 3$.

space-local POD basis that generalizes well outside the training data. The way we divide the domain into subdomains is decided by what results in an L-POD basis that generalizes best on a validation data set, see Section 5.5.2.

Note that this matrix has fewer rows but more columns than the G-POD snapshot matrix, see (5.11). This makes the SVD cheaper to compute for a large number of snapshots s [178]. As in the global case, we use a SVD of the snapshot matrix to obtain a truncated basis $\hat{\mathbf{\Gamma}} \in \mathbb{R}^{J \times q}$ with $q < J$ from the left-singular vector. In terms of the local basis $\{\chi_{ij}\}$ the POD basis is written as

$$\gamma_{ij}(x) = \sum_{k=1}^{J} \chi_{ik}(x)\hat{\Gamma}_{kj}, \tag{5.22}$$

analogous to (5.18). In this basis, the approximated solution u_ℓ is obtained as

$$u_h(x,t) \approx u_\ell(x,t) = \sum_{i=1}^{I}\sum_{j=1}^{q} \mathrm{A}_{ij}(t)\gamma_{ij}(x), \tag{5.23}$$

similarly to the global case, see (5.17). The coefficients A_{ij} are obtained as

$$\mathbf{a}_\ell = \mathbf{\Gamma}^T \mathbf{u}, \tag{5.24}$$

where

$$\mathbf{\Gamma} = \begin{bmatrix} \hat{\mathbf{\Gamma}} & & \\ & \ddots & \\ & & \hat{\mathbf{\Gamma}} \end{bmatrix} \in \mathbb{R}^{N \times r}. \tag{5.25}$$

This matrix, containing non-overlapping blocks, is constructed such that multiplying by its transpose projects $\mathbf{u}$ onto the L-POD basis. For the mapping from $\mathbf{a}_\ell \in \mathbb{R}^{Iq}$ to $\mathbf{A} \in \mathbb{R}^{I \times q}$ we follow the same convention as for $\mathbf{u}$ and $\mathbf{U}$. The effective number of basis functions for L-POD is $Iq = r$ with $q < J$. This basis is orthonormal, i.e. $\mathbf{\Gamma}^T \mathbf{\Gamma} = \mathbf{I}$. The sparsity of $\mathbf{\Gamma}$, as opposed to $\mathbf{\Phi}$, is what yields a ROM which is cheaper to evaluate than a G-POD-based ROM [173].

An example of an L-POD approximation u_ℓ to a Gaussian wave is shown in Figure 5.3. For the L-POD we used $I = 10$ subdomains with $q = 6$ basis functions per subdomain. The training data used to obtain the basis is discussed in Section 5.5, as shown in Figure 5.5. Although the L-POD approximation is accurate in most of the domain, some oscillations appear near the boundaries. In addition, the approximation is not guaranteed to be smooth on the edges of the subdomains, see Figure 5.3. Discontinuities appear when transitioning from one subdomain to the next. In Section 5.5, we will show that these discontinuities tend to grow increasingly severe when using the L-POD basis to construct a ROM, see Figure 5.8. To remedy this issue, we introduce space-local POD with overlapping subdomains in the next section.

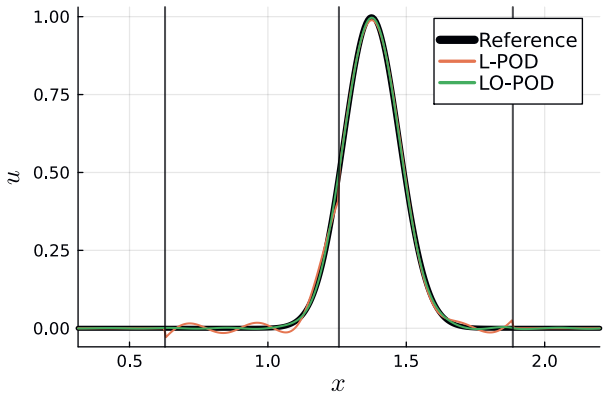


Figure 5.3: L-POD and LO-POD representation of a Gaussian wave. The data used to obtain the local POD basis is explained in Section 5.5. The depicted snapshot is part of the snapshot matrix used to obtain the POD basis. The edges of the subdomains are indicated by the vertical grey lines. Only part of the domain $\Omega = [0, 2\pi)$ is shown.

5.3.3 Local POD with overlapping subdomains

In finite element discretizations, similar discontinuities are resolved by imposing continuity of the approximation, while using a local basis. In this work, we aim to achieve the same by introducing a novel space-local POD formulation with *overlapping* subdomains. We will refer

to this approach as LO-POD. To start off, we subdivide the domain Ω into I overlapping subdomains $\Omega_i = [\alpha_i, \beta_i)$. This means $\beta_{i-2} = \alpha_i < \beta_{i-1} = \alpha_{i+1} < \beta_i = \alpha_{i+2}$. Once again, each subdomain contains J grid points. Note that due to the overlap, each grid point is now located in two subdomains. This means $\frac{I \cdot J}{2} = N$, as opposed to $I \cdot J = N$ for L-POD.

To obtain a smooth approximation u_ℓ, we require the LO-POD basis to smoothly decay to zero on the edge of the subdomain. This is enforced through a post-processing step of the local snapshot matrix given by (5.21). For this purpose, we introduce a kernel $k_i(x)$ to divide the solution between the subdomains. This places the following constraint on the kernels:

$$\sum_{i=1}^{I} k_i(x) = 1, \tag{5.26}$$

such that this set of functions forms a partition of unity. Here, we propose the following kernel

$$k_i(x) = \begin{cases} \sin^2(\frac{1}{2}\frac{x-\alpha_i}{\alpha_i-\alpha_{i+1}}\pi) & \text{if } \alpha_i \leq x < \alpha_{i+1}, \\ \sin^2(\frac{1}{2}\frac{x-\alpha_{i+1}}{\alpha_{i+2}-\alpha_{i+1}}\pi + \frac{1}{2}\pi) & \text{if } \alpha_{i+1} \leq x < \alpha_{i+2}, \\ 0 & \text{elsewhere,} \end{cases} \tag{5.27}$$

which is chosen as it smoothly decays to zero at the subdomain boundaries. This approach is inspired by the partition of unity method [179]. Here, the right half of each subdomain is overlapped by the subdomain to its right, and the left half by the subdomain to its left. A visualization of the kernels is displayed in Figure 5.4. Different kernels and different amounts

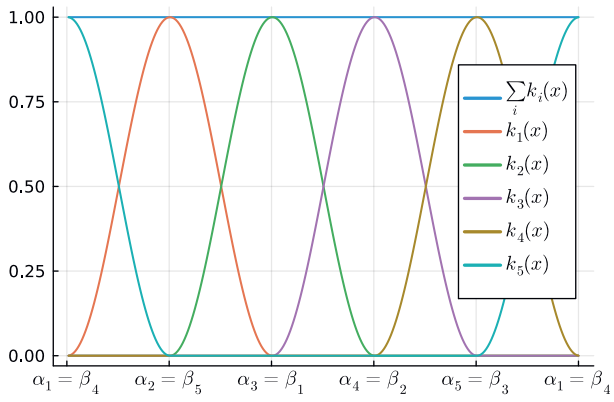


Figure 5.4: Kernels k_i for a subdivision of the periodic domain into five overlapping subdomains of equal size.

of overlap between subdomains can also be considered. However, we consider this outside the scope of the work presented in this chapter.

Using the introduced kernel, the coefficients $\mathbf{U}$ for the local expansion in (5.19) are obtained as

$$\mathrm{U}_{ij}(t) = \frac{(\chi_{ij}(x), k_i(x)u_h(x,t))}{(\chi_{ij}(x), \chi_{ij}(x))}. \tag{5.28}$$

These integrals can simply be approximated using the midpoint rule for integration [180]. The remaining procedure stays the same as for L-POD, i.e., we build the snapshot matrix in (5.21) and carry out a SVD to obtain $\hat{\mathbf{\Gamma}}$. However, what changed is that the blocks in $\mathbf{\Gamma}$, see (5.25), are now overlapping. This means the basis is no longer orthonormal, i.e. $\mathbf{\Gamma}^T\mathbf{\Gamma} \neq \mathbf{I}$. Due to the non-orthonormality, the coefficients $\mathbf{a}_\ell$ for the expansion in (5.23) are obtained by solving the following linear system

$$(\mathbf{\Gamma}^T\mathbf{\Gamma})\mathbf{a}_\ell = \mathbf{\Gamma}^T\mathbf{u}, \tag{5.29}$$

as opposed to (5.24). Obtaining the coefficients from the FOM solution is therefore more expensive. In addition, evaluating the resulting ROM requires solving a linear system, as will be discussed in Section 5.4. This makes the LO-POD ROM more expensive to evaluate than its non-overlapping counterpart L-POD, see Sections 5.4.2 and 5.5.5. However, looking at Figure 5.3, we find that using the LO-POD results in a much smoother approximation of the Gaussian wave. Note that the parameters are kept the same, i.e., $I = 10$ and $q = 6$. In Section 5.5.2, the associated error will be quantified more precisely.

5.4 Space-local, energy-conserving reduced-order model

5.4.1 Projection on reduced basis

Consider again $\mathbf{c}$, the state vector of the full-order model, and $\mathbf{\Phi} \in \mathbb{R}^{N\times r}$ a reduced basis where $r < N$, obtained from either G-POD, L-POD, or LO-POD. For the space-local approaches, this operator was referred to as $\mathbf{\Gamma}$, see Section 5.3.2. The corresponding 'Gram' matrix $\mathbf{S} \in \mathbb{R}^{r\times r}$ of this basis is computed as

$$\mathbf{S} = \mathbf{\Phi}^T\mathbf{\Phi}. \tag{5.30}$$

We can project $\mathbf{c}$ onto the subspace spanned by the POD basis as

$$\mathbf{c}_r = \underbrace{\mathbf{\Phi}\mathbf{S}^{-1}\mathbf{\Phi}^T}_{=:\mathbf{P}}\mathbf{c}. \tag{5.31}$$

Note that for both G-POD and L-POD, computing the inverse of $\mathbf{S}$ is trivial, as it is simply the identity. This is because the basis is orthonormal. However, for LO-POD the basis is non-orthogonal, which makes computing its inverse less trivial. As stated earlier, this is a

downside to LO-POD and increases its computational cost. It is quite straightforward to see that $\mathbf{P}$ is idempotent as $\mathbf{P}^2 = \mathbf{P}$. The POD coefficient vector is obtained as

$$\mathbf{a} = \mathbf{S}^{-1}\mathbf{\Phi}^T\mathbf{c}. \tag{5.32}$$

5.4.2 Galerkin projection

Having obtained a reduced basis, we construct a ROM for the advection equation as follows [7, 68, 170, 171]: based on (5.6) we define the residual $\mathbf{r} \in \mathbb{R}^N$ as

$$\mathbf{r}(\mathbf{u}_r) := \frac{\mathrm{d}\mathbf{u}_r}{\mathrm{d}t} + c\mathbf{D}\mathbf{u}_r \neq \mathbf{0}, \tag{5.33}$$

where we replaced $\mathbf{u}$ by the POD approximation $\mathbf{u}_r = \mathbf{\Phi}\mathbf{a}$. As $\mathbf{u}$ comes from the finite difference discretization of the advection equation, we have $\mathbf{c}(t) = \mathbf{u}(t)$. Next, we carry out the Galerkin projection of the residual on the POD basis, according to (5.32), and set this to zero:

$$\mathbf{S}^{-1}\mathbf{\Phi}^T\mathbf{r}(\mathbf{u}_r) = \mathbf{0}. \tag{5.34}$$

This ensures the residual is orthogonal to the basis. This results in the following ROM:

$$\frac{\mathrm{d}\mathbf{a}^{\mathrm{ROM}}}{\mathrm{d}t} := -c\mathbf{S}^{-1}\mathbf{A}\mathbf{a}^{\mathrm{ROM}}, \tag{5.35}$$

where the ROM operator $\mathbf{A} \in \mathbb{R}^{r\times r}$ is defined as

$$\mathbf{A} := \mathbf{\Phi}^T\mathbf{D}\mathbf{\Phi}. \tag{5.36}$$

Note that we introduced $\mathbf{a}^{\mathrm{ROM}}$ here as the POD state vector predicted by ROM. This is typically not equal to the true $\mathbf{a}$, see (5.32), past $t = 0$. Going from the FOM to the ROM, we reduced the DOF in the system from N to r. In the G-POD case $\mathbf{A}$ is typically dense, whereas in the L-POD/LO-POD it is sparse. The sparsity decreases the cost of evaluating the ROM [173]. The amount of nonzero entries in the G-POD ROM operator scales with $\mathcal{O}(r^2)$. For L-POD it scales with $\mathcal{O}(((n+1)q)^2 I)$, where n is the number of neighbors per subdomain. This accounts for the interaction between the subdomains. For LO-POD, the neighbors of the neighbors have to be included in the interactions due to the overlap. This results in a scaling of $\mathcal{O}(((\tilde{n}+1)q)^2 I)$, where $\tilde{n}$ is the number of subdomains that can be reached by crossing at most one subdomain starting from one of the subdomains. For 1D problems $n = 2$ and $\tilde{n} = 4$. For a large number of subdomains I and a small number of POD basis functions per subdomain q, with the total number of POD modes being $I \cdot q = r$, this scales more favorably than G-POD. This allows us to include a larger number of basis functions in the POD basis without sacrificing computational efficiency. Hyper-reduction

techniques, including energy-conserving ones, can also be employed to further sparsen the ROM operators [74, 181, 182].

5.4.3 Energy conservation of the ROM

To ensure stability of the ROMs, we aim to mimic the energy conservation property of the FOM, see (5.8). To investigate if the constructed ROMs satisfy this property we define the ROM energy E_r^{ROM} as

$$E_r^{\mathrm{ROM}} = \frac{h}{2}(\mathbf{u}_r^{\mathrm{ROM}})^T(\mathbf{u}_r^{\mathrm{ROM}}), \tag{5.37}$$

where $\mathbf{u}_r^{\mathrm{ROM}} := \mathbf{\Phi}\mathbf{a}^{\mathrm{ROM}}$. Employing (5.35) we obtain the ROM evolution of $\mathbf{u}_r^{\mathrm{ROM}}$ as

$$\frac{\mathrm{d}\mathbf{u}_r^{\mathrm{ROM}}}{\mathrm{d}t} = \mathbf{\Phi}\frac{\mathrm{d}\mathbf{a}^{\mathrm{ROM}}}{\mathrm{d}t} = -c\mathbf{\Phi}\mathbf{S}^{-1}\mathbf{A}\mathbf{a}^{\mathrm{ROM}}. \tag{5.38}$$

The evolution of the ROM energy follows as

$$\begin{aligned}\frac{\mathrm{d}E_r^{\mathrm{ROM}}}{\mathrm{d}t} &= h(\mathbf{u}_r^{\mathrm{ROM}})^T\frac{\mathrm{d}\mathbf{u}_r^{\mathrm{ROM}}}{\mathrm{d}t} = -ch(\mathbf{u}_r^{\mathrm{ROM}})^T\mathbf{\Phi}\mathbf{S}^{-1}\mathbf{A}\mathbf{a}^{\mathrm{ROM}} \\ &= -ch(\mathbf{a}^{\mathrm{ROM}})^T\underbrace{\mathbf{\Phi}^T\mathbf{\Phi}}_{=\mathbf{S}}\mathbf{S}^{-1}\mathbf{\Phi}^T\mathbf{D}\mathbf{\Phi}\mathbf{a}^{\mathrm{ROM}} \\ &= -ch(\mathbf{a}^{\mathrm{ROM}})^T\mathbf{\Phi}^T\mathbf{D}\mathbf{\Phi}\mathbf{a}^{\mathrm{ROM}} = -ch(\mathbf{u}_r^{\mathrm{ROM}})^T\mathbf{D}\mathbf{u}_r^{\mathrm{ROM}} = 0,\end{aligned} \tag{5.39}$$

employing the product rule of differentiation. From the final expression, we conclude that a ROM based on Galerkin projection inherits energy conservation and stability from the FOM. This is true for both orthogonal and non-orthogonal projections. This means that not only the G-POD ROM inherits this property, as shown in [67], but also the newly introduced space-local approaches L-POD and LO-POD.

5.5 Results & discussion

5.5.1 Test case setup

To construct a ROM, we first require data from the FOM. As stated earlier, we use the finite difference discretization of the advection equation, detailed in Section 5.2.2, for this purpose. The system is simulated on a periodic domain $\Omega = [0, 2\pi)$ discretized with $N = 1000$ grid points for $t = [0, 5]$ and constant $c = 1$. A time step size of $\Delta t = 0.01$ is used for the time integration, using a RK4 scheme. This is the largest time step size that still yielded stable simulations. Snapshot data for the ROM construction is collected in the interval $t = [0, 1]$. We refer to this as the training data. Data generated in the interval $t = (1, 2]$ will be referred to as validation data. This data is used to evaluate the generalization of the POD basis. The

remaining part of the simulation data, $t = (2, 5]$, will be used to evaluate the extrapolation capabilities of the ROMs. As an initial condition, we use a Gaussian wave centered around $x = \frac{1}{4}\pi$, namely

$$u(x, 0) = \exp(-50(x - \frac{1}{4}\pi)^2). \tag{5.40}$$

A visualization of this simulation is displayed in Figure 5.5.

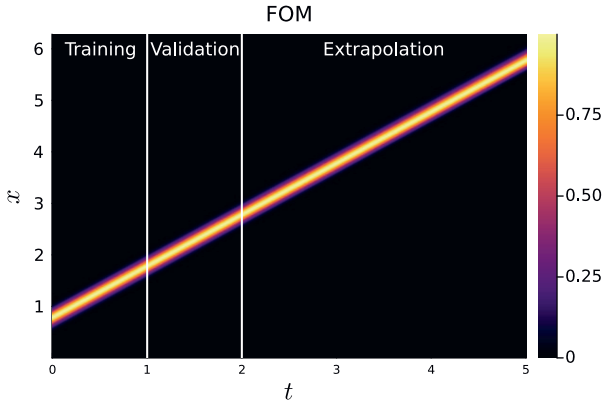


Figure 5.5: Reference simulation of a Gaussian wave being advected throughout the domain. White bars separate the training data from the validation data and the validation data from the extrapolation data, respectively.

For the ROMs we consider the three bases discussed in this work: the global POD basis G-POD, the space-local POD basis L-POD, and the space-local POD basis with overlapping subdomains LO-POD. For the space-local approaches, the domain is subdivided into I subdomains. For the local basis $\{\chi_{ij}\}$, we use J box functions contained within each subdomain, such that $IJ = N$. In this way, the local basis aligns with the FOM finite difference basis, see (5.10). The integrals in the LO-POD post-processing step, equation (5.28), are approximated using the midpoint rule for integration [180]. For the ROM time integration, we use the same RK4 scheme as for the FOM.

To evaluate the ROMs we evaluate the difference between the ROM solution $\mathbf{u}_r^{\mathrm{ROM}} := \mathbf{\Phi a}^{\mathrm{ROM}}$ and the FOM solution $\mathbf{u}^{\mathrm{FOM}}$:

$$\underbrace{\frac{\mathbf{u}_r^{\mathrm{ROM}}(t) - \mathbf{u}^{\mathrm{FOM}}(t)}{||\mathbf{u}^{\mathrm{FOM}}(t)||_2}}_{:=\text{solution error}} = \underbrace{\frac{\mathbf{u}_r^{\mathrm{ROM}}(t) - \mathbf{P}\mathbf{u}^{\mathrm{FOM}}(t)}{||\mathbf{u}^{\mathrm{FOM}}(t)||_2}}_{:=\text{ROM error}} + \underbrace{\frac{\mathbf{P}\mathbf{u}^{\mathrm{FOM}}(t) - \mathbf{u}^{\mathrm{FOM}}(t)}{||\mathbf{u}^{\mathrm{FOM}}(t)||_2}}_{:=\text{projection error}}, \tag{5.41}$$

where $\mathbf{P}$ projects the solution on the POD basis, see (5.31). This difference will be referred to as the solution error. In (5.41), this error is decomposed as the sum of the ROM error (the error made by the ROM during the time integration) and the projection error (the

error made by the reduced basis approximation). Our implementation, in Julia [99], of the introduced methodologies and experiments is freely available on Github, see `https://github.com/tobyvg/local_POD_overlap.jl`.

5.5.2 Projection error

For the construction of the L-POD and LO-POD ROMs, we have to determine the number of subdomains I and the number of modes per subdomain q, with $Iq = r$, which results in a basis that generalizes best outside the training data. To do this, we evaluate the projection error, see (5.41), on both the training and validation data. The results are depicted in Figure 5.6 for different I, averaged over both the training (solid line) and validation data (dashed line). A basis that generalizes well should perform well on both the training and validation

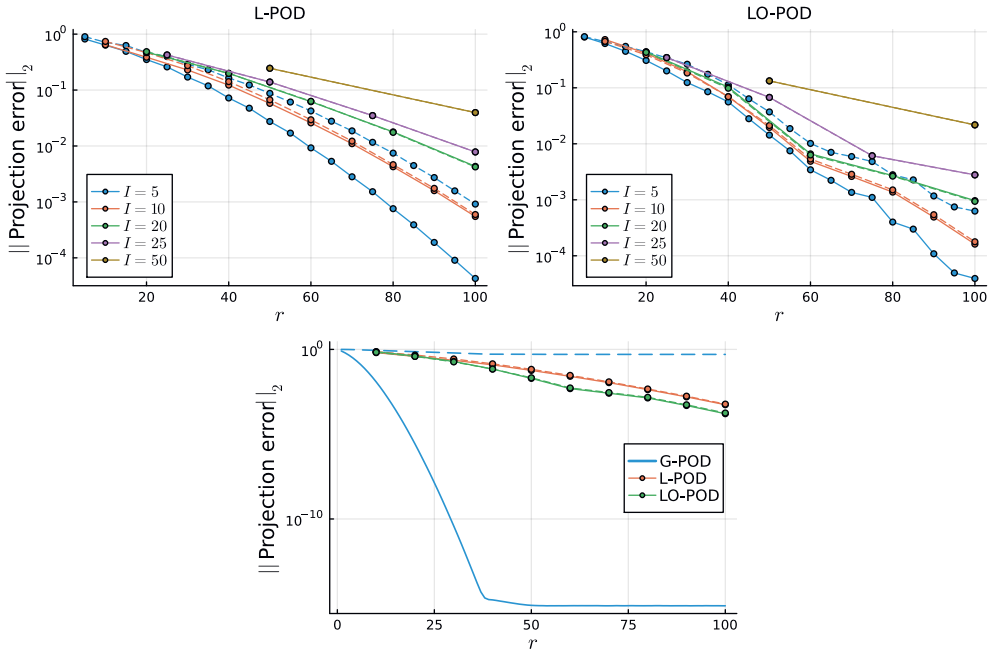


Figure 5.6: (Top) Projection error evaluated over the training (solid line) and validation (dashed line) data for different I for the space-local approaches. (Bottom) Projection error for $I = 10$ for L-POD and LO-POD evaluated over the training (solid line) and validation (dashed line) data. The projection error for G-POD is also depicted.

data. We observe that for $I = 5$, the training error is lowest, but the validation error is rather large, indicating that the basis does not generalize well. For higher values of I, the training and validation errors are much closer, but tend to increase with increasing I. In this case, $I = 10$ is considered optimal, as the training and validation errors are small and of

similar size. This is true for both L-POD and LO-POD. For the remainder of this text, we will therefore stick to $I = 10$ for the construction of the local ROMs. In general, the choice of I is likely problem-dependent, and an a priori study similar to Figure 5.6 needs to be performed to determine an optimal value.

Next, we compare the performance of the space-local approaches against each other, as well as to G-POD. This is also displayed in Figure 5.6 (bottom plot). We observe that the projection error converges faster for LO-POD, as compared to L-POD. This can be explained by the fact that the LO-POD basis functions are constructed from twice as many finite difference points as for L-POD, due to the overlapping subdomains. This means the fit is carried out over twice as many DOF, which yields a lower projection error. The difference between convergence on the training and validation data is slight for both space-local approaches. On the other hand, for G-POD, the convergence of the projection error on the training data is very fast, see (5.13). However, on the validation data, the error hardly converges. This is because the G-POD is only suited for representing the wave on the left side of the domain, as detailed in the next section. We conclude that the space-local approaches yield a basis that generalizes better than the global approach for this particular problem.

5.5.3 Resulting basis

The resulting local basis functions for $q = 6$ are displayed in Figure 5.7 along with the first six G-POD modes. We find that the G-POD modes are only nonzero where the traveling wave is represented in the training data. This explains why the error on the validation data hardly converged in Figure 5.6. For the space-local approaches, where a common basis is “copied/pasted" across the entire domain, this is not an issue. Regarding L-POD, we observe that the obtained basis functions do not smoothly decay to zero at the edge of the subdomain, but instead end abruptly. Finally, we observe that for LO-POD the post-processing procedure in (5.28) indeed results in a basis that smoothly decays on the edge of the subdomains.

5.5.4 Accuracy and energy conservation of ROM

Having obtained a basis for each of the POD approaches we construct a set of ROMs. Based on the results in Section 5.5.2, we select $r = 60$, $I = 10$, and $q = 6$ for the space-local approaches. To keep the size of the basis the same, we choose $r = 60$ for G-POD. The ROM results are obtained by evaluating (5.35) from the initial condition in (5.40). The resulting simulations up to $t = 5$ are presented in Figure 5.8. For G-POD we find that after the training data the performance degrades significantly. This exposes the limitations of G-POD, as it is unable to adapt to wave traveling outside the training range. The space-local approaches perform better in this regard. Both L-POD and LO-POD are capable of extrapolating the traveling of the wave past the training region. However, L-POD seems to suffer from

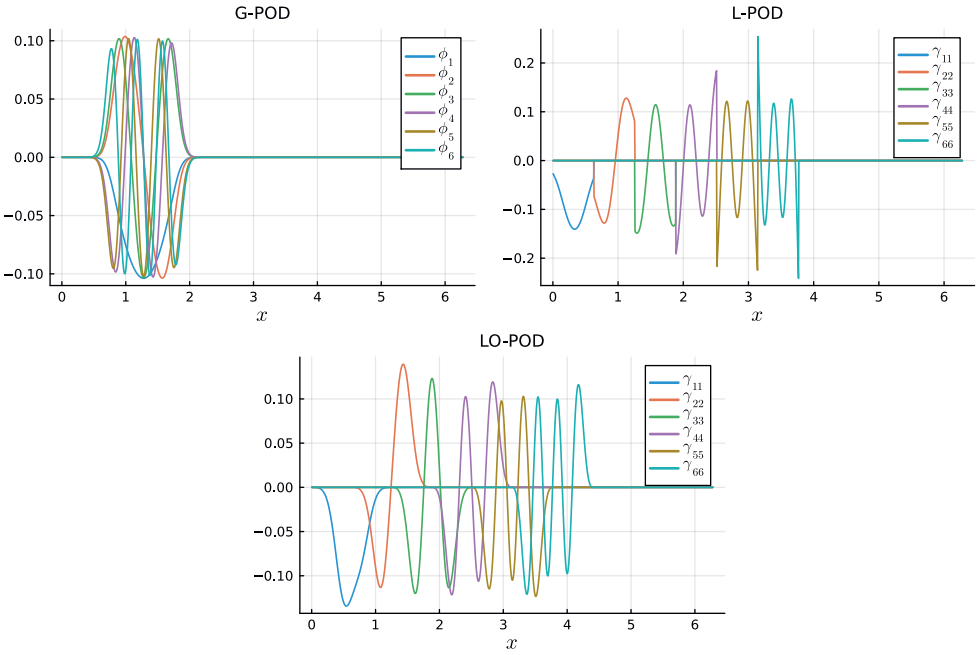


Figure 5.7: The first six G-POD modes and the first six space-local POD modes for L-POD and LO-POD. For visualization purposes, the space-local POD modes are displayed on adjacent subdomains.

discontinuities in $\mathbf{u}_r^{\mathrm{ROM}}$, as the edges of the subdomains become increasingly visible as the simulation progresses. LO-POD does not suffer from this issue and smoothly extrapolates the solution past the training region. This means the smooth reconstruction coming from LO-POD indeed improves the quality of the simulation for the same number of DOF.

Next, we evaluate the ROMs on a set of performance metrics. The first metric is the solution error, see (5.41). The other metrics (defined on the figure axes) focus on how well the energy is conserved during the simulation, namely the total change in energy and the instantaneous change in energy. The results are shown in Figure 5.9. The results are depicted for both the FOM time step size Δt (solid line) and a five times larger time step $5 \cdot \Delta t$ (dashed line). This larger time step is based on an eigenvalue analysis of the ROM operator, also presented in Figure 5.9. For this analysis, we determined the eigenvalues of the operator $\mathbf{D}$ projected on the ROM basis. This operator is given by $\mathbf{PDP}$. It can be shown that replacing $\mathbf{D}$ in the FOM, see (5.6), by this projected operator is equivalent to integrating the ROM, see (5.35). The eigenvalues λ_i are ordered according to the magnitude of their absolute values. We observe that the largest eigenvalue for the space-local approaches is roughly five times smaller than that of G-POD and the FOM. This is likely due to the fact that the G-POD

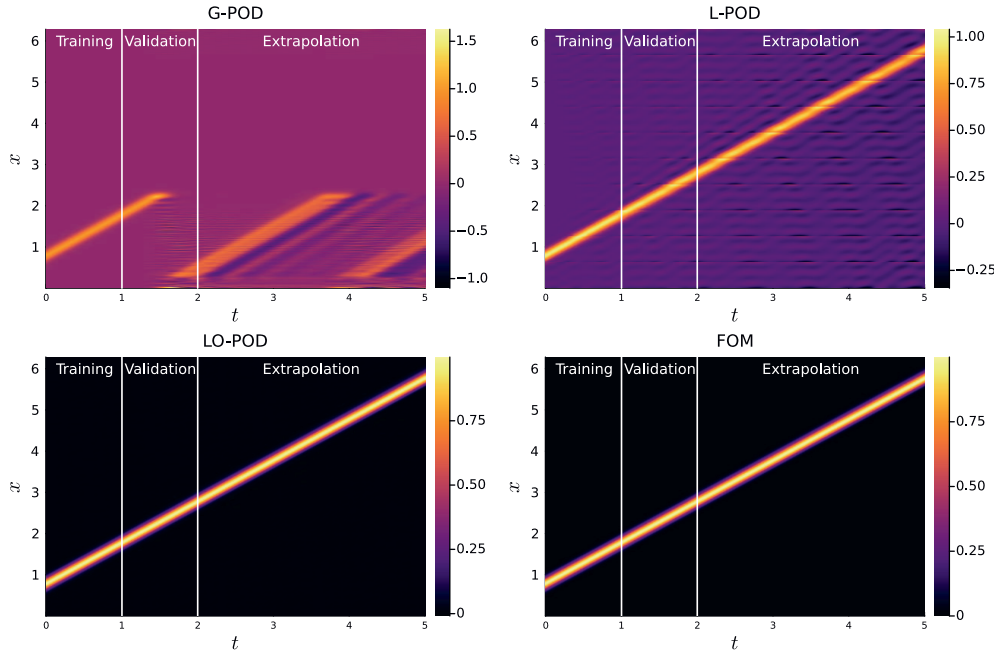


Figure 5.8: Trajectories of $\mathbf{u}_r$ for the different ROMs, along with the FOM trajectory. From left to right, the white lines indicate the end of the training data and validation data, respectively.

basis functions contain higher frequencies than the space-local basis functions, see Figure 5.7. Using a smaller G-POD basis could alleviate this. Based on this analysis, we also evaluate the performance of the ROMs for a five times larger time step [183].

Looking at the solution error, we observe that G-POD performs best in the training region (the error is so small that it is outside the plotting range). However, after leaving the training region ($t > 1$), the performance degrades rapidly. On the other hand, for the space-local approaches, we do not observe this jump of error outside the training region, but rather a steady increase. Importantly, outside the training region, the space-local approaches are much more accurate than G-POD. In particular, LO-POD improves by more than one order of magnitude upon both G-POD and L-POD outside the training region.

When increasing the time step size by fivefold, the simulation quickly becomes unstable for G-POD. For L-POD, both time steps yield stable simulations, giving results that are very close to each other. For LO-POD and a smaller time step size, the error seems to converge to an equilibrium. However, for a larger time step, it increases steadily. This means there is likely still a benefit to taking a smaller time step. An interesting continuation of this research would be to find a way to systematically determine the “sweet spot" between

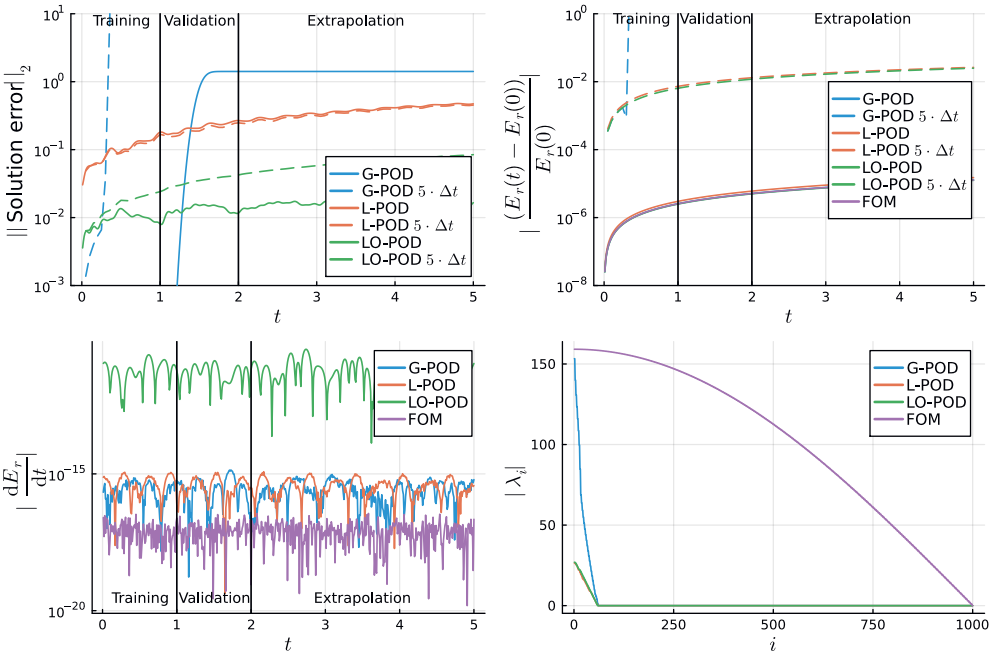


Figure 5.9: (Top-left) Solution error for each of ROMs during the simulation, presented for both $\Delta t = 0.01$ and $\Delta t = 0.05$. From left to right, the black lines indicate the end of the training data and validation data, respectively. (Top-right) Relative change in energy with respect to the start of the simulation. (Bottom-left) Instantaneous change in energy, computed by evaluating (5.39). (Bottom-right) Eigenvalues of the projected FOM operator.

increasing the time step and maintaining accuracy of the ROM. This could for example, be done by considering several different time step sizes and both implicit and explicit integration methods, or adaptive time-stepping based on an error estimator [184].

In terms of energy conservation, we observe that all ROMs conserve the energy as predicted by the theoretical analysis, except for a time discretization error incurred by the use of RK4. This error increases when the time step size is increased. Only for G-POD with an increased time step size, the simulation becomes unstable. For LO-POD, the numerical error in the instantaneous change in energy, see (5.39), is the largest. This is likely due to the linear system that needs to be solved to evaluate the ROM for a non-orthogonal basis, see (5.35). However, the time discretization error is the primary source of error, as the change in energy during the simulation is roughly the same for L-POD and LO-POD.

5.5.5 Convergence with increasing ROM dimension

Next, we look at the convergence of the solution error as we increase the size of ROM basis r. In addition, we compare the ROMs to a finite difference discretization with the same number of DOF. Note that for the ROMs DOF $= r$. For the finite difference simulations, we first project the solution on the FOM grid using linear interpolation. We then compute the difference, such as in (5.41), and compute the L_2-norm to quantify the error. For the local ROMs we stick to $I = 10$ subdomains, while increasing the number of basis functions per subdomain q to increase r. We consider a maximum of $r = 100$ for the ROMs.

The results are depicted in Figure 5.10. Regarding the local ROMs, we observe rapid

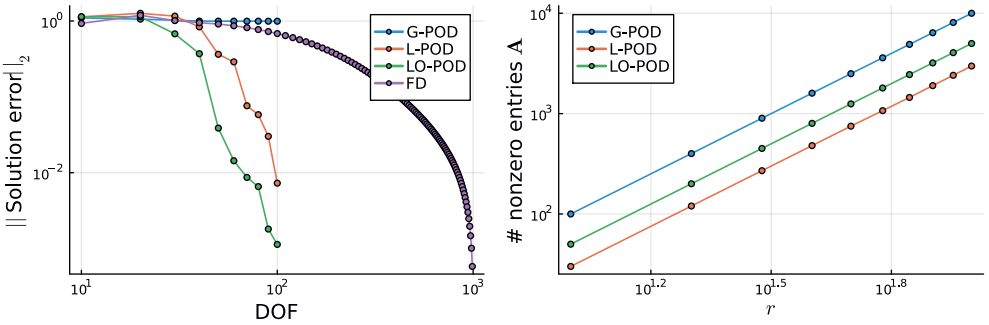


Figure 5.10: (Top-left) Solution error averaged over a simulation up to $t = 5$. Results are presented for the different ROMs, as well as a finite difference discretization (FD) with the same number of DOF. (Bottom-right) Number of nonzero entries in the ROM operator $\mathbf{A}$.

convergence of the solution error as we increase the number of DOF, with LO-POD consistently outperforming L-POD. This is in line with the projection error convergence discussed in Section 5.5.2. Regarding G-POD, we observe no convergence of the solution error. Regarding the finite difference discretization, we find it converges at a much slower rate than the space-local ROMs.

To obtain a conclusive answer about the scaling of the ROMs, we considered the number of nonzero entries in the ROM operator $\mathbf{A}$. This is also depicted in Figure 5.10. Here we find that the number of nonzero entries scales according to a power law, with L-POD scaling the most favorably. This is in line with the discussion presented in Section 5.4.2. Hyper-reduction can be carried out to further reduce the computational cost of the ROMs [74, 181, 182]. Finally, one could also increase the time step size, as discussed in Section 5.5.4, to further decrease the computational cost of the space-local ROMs.

5.5.6 2D advection-diffusion equation

To further evaluate the viability of the space-local ROMs, we consider the 2D advection-diffusion equation:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = \underbrace{-\nabla\cdot(\mathbf{V}(\mathbf{x})u(\mathbf{x},t))}_{\text{advection}} + \underbrace{\nu\nabla^2 u(\mathbf{x},t)}_{\text{diffusion}}, \tag{5.42}$$

where some quantity $u(\mathbf{x},t)\in\mathbb{R}$ is advected through space $\mathbf{x}=\begin{bmatrix}x & y\end{bmatrix}^T\in\mathbb{R}^2$ by a stationary velocity field $\mathbf{V}\in\mathbb{R}^2$ which is divergence free, i.e. $\nabla\cdot\mathbf{V}=0$. The fact that the velocity field is now space-dependent, as opposed to the 1D case where it was uniform, adds to the difficulty of the test case. As opposed to the 1D case this system also contains diffusion. This results in a passive spread of u throughout the domain. The diffusion rate is determined by the scalar viscosity $\nu\geq 0$. In our case, we employ a finite volume discretization with a staggered grid for the velocity field [24]. This is done to preserve the skew-symmetry of the operator. The diffusive term is discretized using a simple second-order scheme.

For the FOM we consider the following setting: A periodic domain $\Omega=[-\pi,\pi]\times[-\pi,\pi]$, a velocity field given by $\mathbf{V}=\begin{bmatrix}\cos(x-y) & \cos(x-y)\end{bmatrix}^T$ discretized on a 256×256 uniform grid, and a viscosity of $\nu=10^{-3}$. The initial condition is given by the sum of three Gaussians, each centered on one of the streamlines of the flow where the velocity is highest:

$$\begin{aligned} u(\mathbf{x},0) = &-\exp(-2x^2-2y^2)+\exp(-2\left(x-\frac{\pi}{2}\right)^2-2\left(y+\frac{\pi}{2}\right)^2)\\ &+\exp(-2\left(x+\frac{\pi}{2}\right)^2-2\left(y-\frac{\pi}{2}\right)^2). \end{aligned} \tag{5.43}$$

The FOM is integrated in time using a RK4 scheme with $\Delta t=0.025$. Every 16 time steps, we save a snapshot of $u(x,t)$ for the construction of the ROM basis. Note that the performance of the ROMs converge to the FOM, when doing the Galerkin projection. To resolve this closure, models can be included in the ROM [185]. The snapshots collected in the interval $t=[0,12]$ are used as training data, the interval $t=(16,20]$ is used as validation data, and the interval $t=(20,40]$ is used to test the extrapolation capabilities of the ROMs. The ROMs are also integrated in time explicitly using a RK4 scheme. However, for the LO-POD ROM we resort to the implicit Crank-Nicolson method, see [172], to limit the computational cost. This scheme enables us to achieve second-order accuracy in time, while solving only a single linear system at each time step, rather than at each evaluation of (5.35) in the RK4 scheme. To solve the resulting system efficiently, we employ an LU-decomposition [186]. This is required, as the LO-POD basis is non-orthogonal, i.e. $\mathbf{S}\neq\mathbf{I}$.

Regarding the subdomain decomposition, we use the fact that the FOM is discretized on a uniform grid. In this way, we can simply subdivide the domain uniformly into subdomains. For L-POD, this results in a straightforward decomposition. For LO-POD the decomposition

is chosen such that each subdomain overlaps with its four neighboring subdomains which share a vertex in the middle of the considered domain. This is depicted in Figure 5.11.

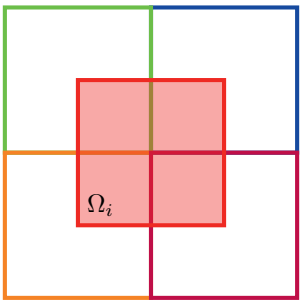


Figure 5.11: Subdomain Ω_i and its overlapping subdomains for LO-POD in 2D.

The number of subdomains I for the space-local approaches and the number of modes q is chosen according to the performance on the validation set. This is presented in Appendix D.2. Based on this, we settle on $I = 8 \times 8$ and $q = 20$ for L-POD and $q = 15$ for LO-POD. We settle on these values for q as these are the lowest values which surpass a projection error threshold of 10^{-2} on the validation set for this value of I. The value of I is chosen as it generalizes well from the training data set to the validation data set. For G-POD, we choose $r = q = 31$, as the snapshot matrix contains 31 snapshots for G-POD such that the SVD only produces 31 basis functions. For the size of the time step, we choose the largest value that does not degrade the ROM performance. This is presented in Appendix D.3. The time step sizes used are presented in Table 5.1.

5.5.6.1 2D basis

The resulting basis computed with the SVD for each of the POD approaches is depicted in Figure 5.12. From the basis functions, we can see that the flow runs diagonally across the domain. This is expected from the prescribed velocity field for which $\mathrm{V}_1 = \mathrm{V}_2$. The G-POD modes contain information on the most dominant features of the flow. From these modes, we can see that some interesting dynamics occur in the region where the flow changes direction. The second and the third modes also seem to be quite similar, but slightly shifted. This is common for advection-dominated flows and results in a slow singular value decay [7]. For the space-local approaches, we find that the basis functions contain less information about the flow, but seem more similar to a generic basis. This likely allows them to generalize better on the validation set. We provide more insight on this in Appendix D.2. For LO-POD we find that the basis functions smoothly decay to zero at the edge of the subdomain, as enforced by the kernel introduced in Appendix D.1. This allows LO-POD to produce smooth approximations. The basis resulting from L-POD does not guarantee this.

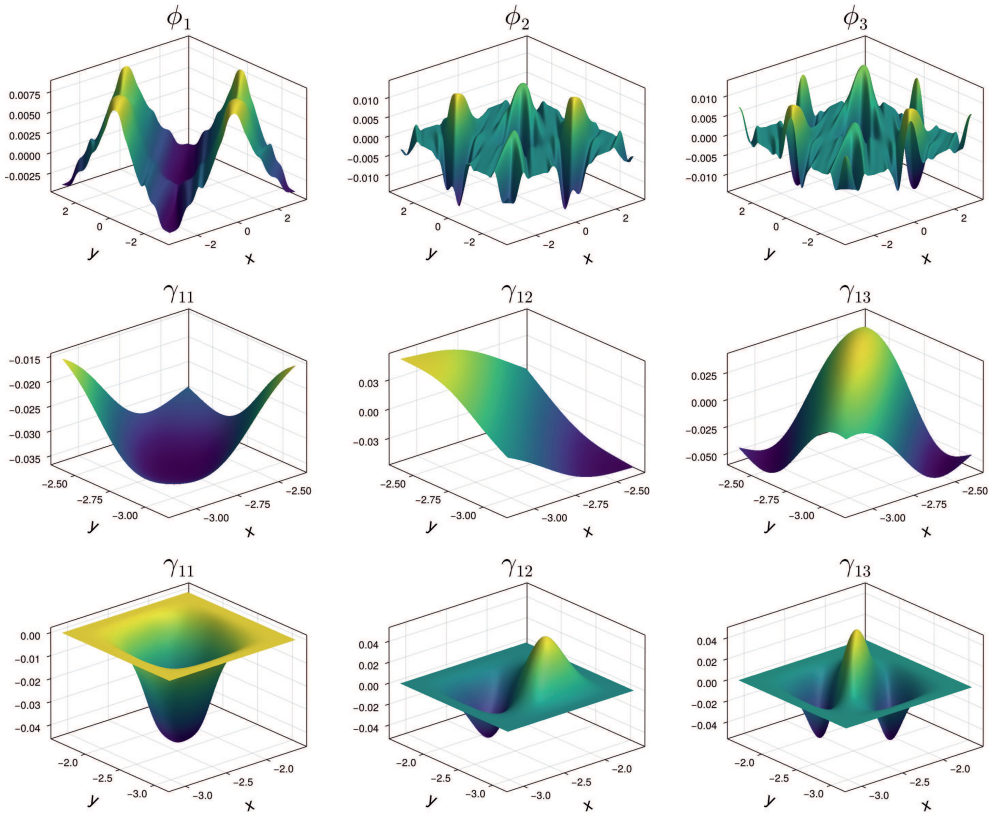


Figure 5.12: The first three G-POD (top) modes and the first three space-local POD modes for L-POD (middle) and LO-POD (bottom) for the 2D advection test case.

5.5.6.2 ROM performance

In this section, we discuss the performance of different ROMs and compare to both the FOM and a finite volume discretization, which is twice as coarse as the FOM in each direction (128×128 as compared to 256×256 for the FOM). This will be referred to as coarse FOM. The results are summarized in Table 5.1 for a full simulation up to $t = 40$ (including the training, validation, and extrapolation region).

For the G-POD ROM we find a short computation time, due to the limited number of modes, but also a large solution and gradient error. The latter is defined as

$$\text{gradient error} := \frac{\mathbf{G}\mathbf{u}_r^{\text{ROM}}(t) - \mathbf{G}\mathbf{u}^{\text{FOM}}(t)}{||\mathbf{G}\mathbf{u}^{\text{FOM}}(t)||_2}, \tag{5.44}$$

where $\mathbf{G} \in \mathbb{R}^{2N \times N}$ is a forward difference approximation of the gradient operator, such

Table 5.1: Computation times (**comp time**) in seconds on a standard laptop CPU and average solution errors (**sol err**) computed for the entire simulation $t \in [0, 40]$ for each of the ROMs for the 2D advection test case. A coarse FOM with a resolution of 128×128 is also included for comparison. For the space-local approaches, the number of DOF is reported as the multiplication $I \times q = r$. Reported computation times are an average of 5 replica simulations. The average error for the gradient of the solution (**grad err**) is also depicted.

	DOF $(I \times q)$	**comp time**	$\|\|$**sol err**$\|\|_2$	$\|\|$**grad err**$\|\|_2$	Δt
G-POD	**31**	$\mathbf{2.5 \times 10^{-3}}$	0.267	0.636	**0.2**
L-POD	$8^2 \times 20 = 1280$	0.207	**0.0353**	0.213	**0.2**
LO-POD	$8^2 \times 15 = 960$	1.02	0.0354	**0.164**	0.05
coarse FOM	$128^2 = 16384$	4.57	0.226	0.514	0.1
FOM	$256^2 = 65536$	25.9	-	-	0.025

that the elements of $\mathbf{G}\mathbf{u}_r^{\mathrm{ROM}}$ approximate ∇u in different parts of the domain. These large errors are likely the result of the G-POD basis's limited generalization capabilities. For the space-local approaches, we observe that the computation time is roughly two orders of magnitude larger than for G-POD. As we have access to more space-local POD modes than global ones, we choose to include enough to accurately represent the solution, see Appendix D.2 for details. This increases the computation time with respect to G-POD, see Table 5.1 for the exact number of DOF for each ROM. However, using the space-local approaches, we still obtain a speedup between one and two orders of magnitude with respect to the FOM. The L-POD ROM is roughly 5 times faster than the LO-POD ROM. This is mostly because the explicit RK4 scheme (fourth order accurate) used for L-POD allows for larger time steps without loss of accuracy, as compared to the Crank-Nicolson scheme (only second-order accurate) used for LO-POD, see Appendix D.3 for an analysis on the time step size. As mentioned earlier, the Crank-Nicolson scheme is used for LO-POD as it limits the number of linear systems needed to be solved per time step to one. L-POD and LO-POD both produce a similar solution error. However, looking at the gradient error LO-POD obtains a lower error. This is likely because LO-POD does not suffer from discontinuities at the subdomain boundaries, while L-POD does, see Figure 5.3. Both space-local approaches outperform the coarse grid finite volume discretization, both in computation time and error metrics.

Next, we consider the solution at the end of the simulation ($t = 40$), see Figure 5.13. Here we find that G-POD does not capture the solution well and contains unphysical artifacts, whereas the space-local approaches do reproduce the solution well. The coarse-grid finite volume discretization also captures the solution quite well, but is slightly smoothed and shifted (due to numerical diffusion and dispersion).

Finally, we consider the solution error and energy trajectory for each of these models, see Figure 5.14. We find that the G-POD ROM performs well within the training region, but it

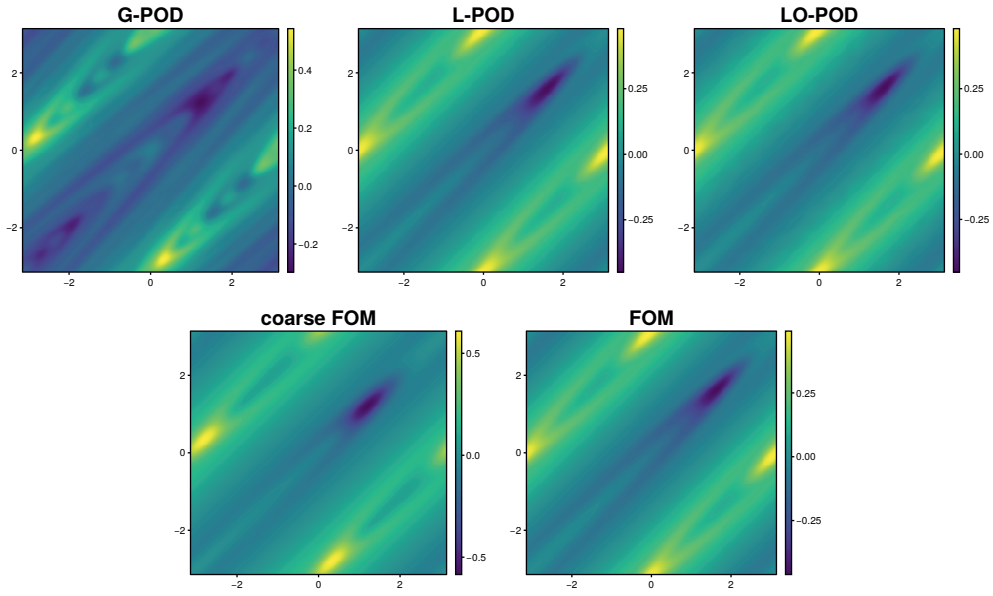


Figure 5.13: Solution at the end of the simulation at $t = 40$ for the 2D advection test case produced by the different ROMs as well as a coarse FOM.

does not extrapolate well. For the space-local approaches, this is not an issue as both perform comparably, both inside and outside the training region, and for both the solution error and the energy. These approaches also outperform the coarse FOM, which slowly accumulates error during the simulation.

5.5.7 Applicability to 2D Navier-Stokes

As a final test case, we evaluate the applicability to the 2D incompressible Navier-Stokes equations by assessing how well the POD bases generalize to representing the solution. The incompressible Navier-Stokes equations read:

$$\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla)\mathbf{V} = -\frac{1}{\rho}\nabla p + \nu \Delta \mathbf{V} + \mathbf{f}, \tag{5.45a}$$

$$\nabla \cdot \mathbf{V} = 0, \tag{5.45b}$$

for a 2D velocity field $\mathbf{V}(\mathbf{x}, t) \in \mathbb{R}^2$. For the density we choose $\rho = 1$ and for the kinematic viscosity $\nu = \frac{1}{1000}$. For the forcing we take $\mathbf{f}(\mathbf{V}, \mathbf{x}, t) = \begin{bmatrix} \sin(4y) & 0 \end{bmatrix}^T - 0.1\mathbf{V}$, as to simulate Kolmogorov flow. This setup is often used for evaluating data-driven turbulence modeling strategies [40, 47, 49]. The PDE is solved numerically using the second-order accurate scheme

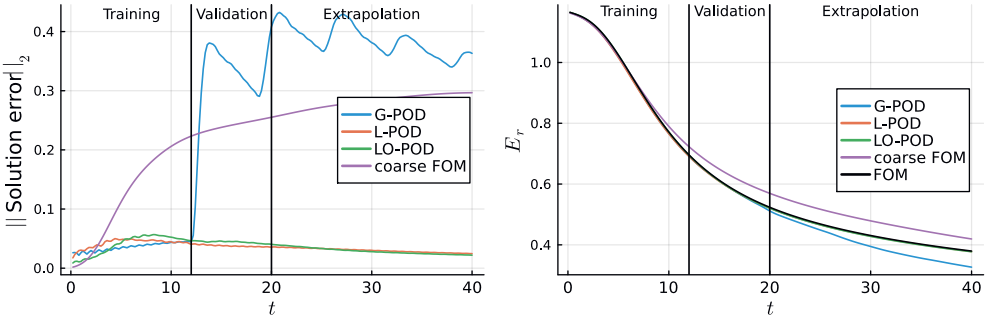


Figure 5.14: (Left) Solution error for each of the ROMs during the simulation of the 2D advection test case. (Right) Energy trajectories during the simulation. Energy trajectories from L-POD and LO-POD overlap with the FOM trajectory. From left to right, the black lines indicate the end of the training data and validation data, respectively.

presented in [25] on a periodic domain $\Omega = [-\pi, \pi] \times [-\pi, \pi]$. The computational grid consists of 2048×2048 grid cells. The simulation is initialized with energy content in the large wave-numbers, and a warm-up period of 25 time units is employed before collecting the training data. After the warm-up period, training data is collected on the interval $t = [0, 10]$ and validation data on the interval $t = [100, 110]$, to remove any temporal correlation. Both the validation and training data consist of 50 snapshots of the vorticity $\boldsymbol{\omega} = \nabla \times \mathbf{V} \in \mathbb{R}^3$. For 2D, only the third component of this vector is nonzero, which we shall refer to simply as ω. We choose to represent the vorticity as for periodic domains in 2D, all the information of the flow is contained within this scalar field [187].

To begin our analysis, we examine the projection error for both the training and validation datasets using an 8×8 decomposition of subdomains, which is identical to the 2D advection-diffusion case, see Figure 5.15. Similar to the previous test cases, we find that the projection

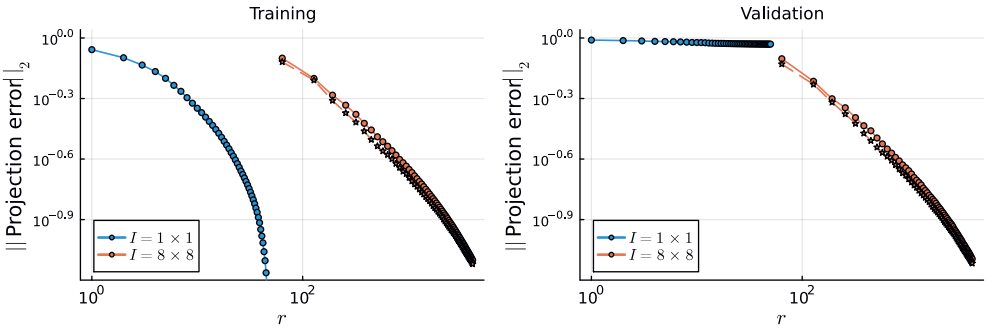


Figure 5.15: Projection error evaluated over the training (left) and validation (right) data set for the 2D Navier-Stokes test case. $I = 1 \times 1$ corresponds to G-POD. Solid dots correspond to L-POD, stars to LO-POD.

error of the G-POD converges fast on the training set. However, upon examining the validation set, we find that the space-local approaches generalize significantly better.

In Figure 5.16, we display the reconstruction of the first snapshot of the validation set produced by the different bases. We depict the reconstructions for both $q = 10$ and

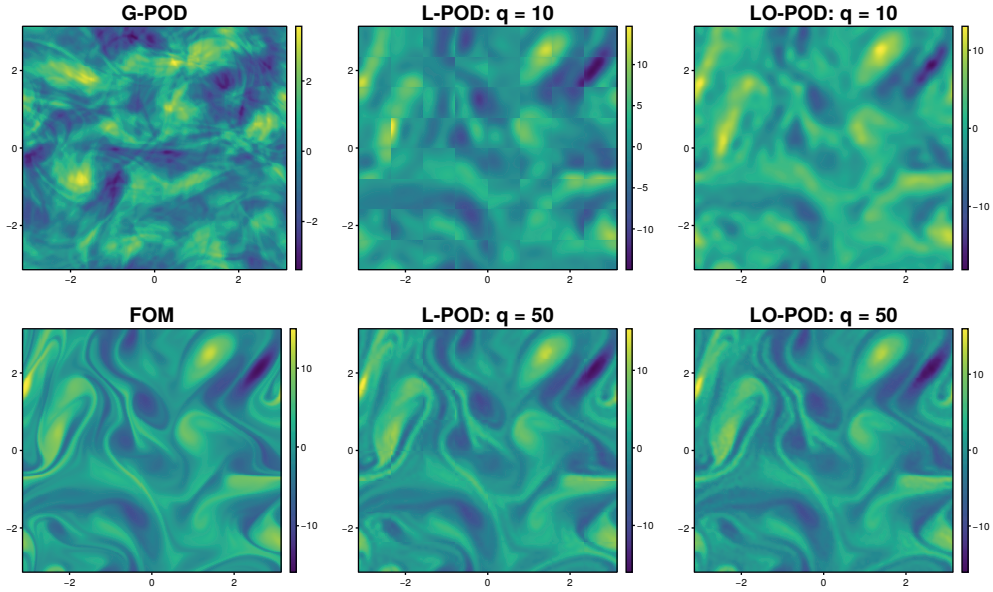


Figure 5.16: First vorticity snapshot of the validation data projected on the different POD bases with varying number of modes per subdomain q for the space-local approaches.

$q = 50$ modes per subdomain for the space-local approaches, and all the available POD modes, i.e. $r = 50$, for G-POD. We find that the space-local approaches capture the flow characteristics quite well, with $8 \times 8 \times 50 = 3200$ DOF, whereas G-POD does not capture any of the characteristics. Regarding L-POD, we find that using a lower number of modes per subdomain q worsens the discontinuities at the subdomain boundaries, whereas LO-POD suffers from noisier reconstructions for low q.

5.6 Conclusions

In this work, we presented a novel way of constructing ROMs for PDEs describing advection-dominated problems. The key properties of our proposed ROM are: *sparsity* and *generalizability* through a space-local ROM with overlapping subdomains; *stability* by embedding energy conservation in the ROM. The space-local ROMs are achieved by building a POD basis within each subdomain, as compared to a single common basis for the standard space-global

approach. In [173], it was shown that such a basis results in a sparse and computationally efficient ROMs. As suggested in [175], we modified this approach to generate a common local basis for the entire domain, similar to a finite element basis. The work in this chapter differs from [175] in that we applied this to time-dependent PDEs, as opposed to steady-state problems. In addition, we introduced overlapping subdomains to ensure a smooth representation of the solution within the space-local POD basis. By generating a common basis, we achieve generalizability in time, as in this way, a feature observed in one subdomain can now be represented in any of the subdomains. In addition, we introduced overlapping subdomains to prevent discontinuities at the subdomain boundaries. To test our methodologies, we made use of the linear advection equation in both 1D and 2D. One of the properties of this system is that the energy is conserved. Our space-local ROMs satisfy this property exactly, even when the basis is non-orthogonal (as is the case for the overlapping approach).

We observed that the resulting space-local ROMs generalize much better for advection-dominated problems than the standard space-global approach. We also demonstrated that such a space-local approach yields sparser and, consequently, more computationally efficient ROMs. In addition, we showed that the space-local ROMs also satisfy energy conservation and allow for larger time steps. The latter further decreases the computational cost of the ROMs. Regarding computational time, we observed an improvement of one to two orders of magnitude with respect to the FOM in the 2D test case. By introducing overlapping subdomains, we demonstrated that we obtain smoother approximations of the solution and require fewer basis functions to represent it compared to the non-overlapping approach. However, this comes at the cost of solving a linear system at each time step. Depending on the application, either overlapping or non-overlapping subdomains might be most suitable. If smoothness of the solution is required, overlapping subdomains might be more suitable, while if computational efficiency is of higher priority, non-overlapping subdomains are likely preferred.

To further evaluate the generalizability of the space-local POD approaches, we applied it to the 2D incompressible Navier–Stokes equations in a Kolmogorov flow setup. In this case we solely considered the ability of POD basis to represent the solution, without building the ROMs. We consider constructing the actual ROM for this non-linear 2D test case outside the scope of this research, as it requires dealing with the divergence-freeness condition, and possibly introducing hyperreduction methods for the non-linear term [181, 182]. Regarding the representation of the solution space-local methods (L-POD and LO-POD) clearly outperformed G-POD, which showed strong overfitting to the training data. In contrast, the local variants maintained low projection errors and accurately reproduced flow features on unseen, temporally decorrelated data. These results indicate that the space-local framework generalizes well for advection-dominated systems and holds strong potential for reduced-order modeling of turbulent flows.

For future research, we consider constructing a space-local ROM for an actual turbulence test case described by the Navier-Stokes equations. To achieve energy conservation, one can make use of the observation presented in [67], where it was shown that carrying out a Galerkin projection on an energy-conserving FOM (with quadratic non-linearity) resulted in an energy-conserving ROM. However, one important requirement is that the POD basis needs to be divergence-free, meaning that the space-local POD basis also needs to be divergence-free. Currently, this is still being actively researched in our group. To resolve this, we could take inspiration from [175] and construct a POD basis, not only for the velocity, but also for the pressure [69]. Another possible research direction would be to circumvent or speed up the solution of the linear system required in LO-POD. For this purpose, one could possibly take inspiration from the finite element community [172]. Finally, the application of the space-local POD methods to steady-state problems could be investigated. The work in [175] offers an interesting starting point for this, as they investigated the generalization of the space-local POD basis when increasing the size of the domain for steady-state problems. An interesting research avenue would be to compare the space-local ROMs to space-global ones, when the spatial domain is kept the same and one of the model parameters is varied.

6

Conclusion

For flows at high Reynolds numbers, the onset of turbulent fluctuations imposes increasing demands on the resolution of the computational grid. To address this challenge, this thesis introduces a set of novel data-driven methods designed to accelerate fluid flow simulations. The core idea is to embed physical structure into machine learning models, with the aim of enhancing both the accuracy and stability of the simulations. In addition, such approaches aim to improve the generalizability of the resulting models. The presented methodologies were tested on 1D problems and then extended to 2D flows, with test cases being mostly limited to flows with periodic boundary conditions on Cartesian grids.

6.1 Conclusions

In Chapter 1, we formulated a set of research topics, where each topic contributes to the overarching theme of this thesis from a different perspective. When introducing these topics, we formulated a set of research questions. These will be answered here.

6.1.1 Topic I: Machine learning-based closure models for large eddy simulation (Chapters 2 & 3)

In Chapters 2 and 3, we aimed to develop machine learning–based large eddy simulation (LES) closure models with structure-preserving properties, with a primary focus on energy conservation as a means of ensuring stability. This approach was motivated by the instability issues commonly observed in physics-agnostic machine learning closures, as well as by the limited flexibility and excessive dissipation associated with classical functional LES models.

To address these challenges, we introduced a novel closure modeling paradigm that explicitly represents the subgrid-scale (SGS) energy and incorporates a neural network architecture with built-in energy-conserving properties. Its performance was evaluated on different test cases, ranging from 1D to 2D applications. The following research questions were answered in Chapters 2 and 3:

- **RQ1: How can a stable and physically consistent machine learning-based closure model be formulated and how does its performance compare to existing approaches?**

  A stable and physically consistent machine learning-based closure model can be formulated by explicitly embedding conservation laws into the learning framework. In Chapter 2, this is achieved by introducing compressed SGS variables that reintroduce unresolved kinetic energy, while substantially reducing the number of degrees of freedom compared to the direct numerical simulation (DNS). A novel neural network architecture is proposed that enforces conservation of the total energy, thereby guaranteeing stability of the resulting dynamical system. This prevents the introduction of unphysical energy into the system. This, while still allowing for backscatter to be modeled. This is done by constraining the neural network architecture to be of skew-symmetric form, which allows for an exchange of energy between the resolved scales and the SGS variables, emulating backscatter. Numerical experiments on Burgers' and Korteweg-de Vries equations demonstrate stable and robust performance of the proposed architecture, consistently improving upon simulations without closure. Compared to standard convolutional neural network (CNN)-based closures, the proposed method achieved comparable or improved accuracy, significantly enhanced stability, and greater consistency across training realizations and extrapolation scenarios. We also observed a substantial improvement in accuracy, compared to the Smagorinsky model.

- **RQ2: Can machine learning-based closure models with stability guarantees, developed for simplified systems, be successfully extended to more complex fluid dynamics problems?**

In Chapter 3, we extended the neural network architecture developed in Chapter 2 to the 2D incompressible Navier–Stokes equations and compared its performance against existing machine-learning and physics-based closure modeling approaches. To enable the extension to 2D, we removed the SGS variables from the framework, since their representation in multiple spatial dimensions remains an open problem. As a consequence, the resulting closure model no longer explicitly represents backscatter.

When evaluating the numerical experiments, we observed that unconstrained machine learning models, although initially accurate, accumulated numerical errors and ultimately became unstable during long-time integrations. In contrast, our architecture remained stable throughout the entire simulation and did not exhibit such numerical artifacts. When compared with established physics-based LES models, including the standard and dynamic Smagorinsky models, our neural network–based closure produced more accurate flow fields and energy spectra. This shows that, even without the inclusion of the SGS variables, the energy-conserving neural network architecture introduced in Chapter 2 serves as a stable and accurate closure model.

6.1.2 Topic II: Data-driven extensions of the evolve-filter-relax framework (Chapter 4)

In Chapter 4, we focused on the evolve-filter-relax (EFR) framework for coarse-grained fluid simulations. The main goal was to alleviate the limited flexibility and improve the computational efficiency of the differential filter using data-driven approaches. To do this, we proposed the use of a linear data-driven filter in Fourier space. In this way, the filter is applied efficiently using fast Fourier transforms (FFTs). This formulation keeps the optimization problem simple while also facilitating stability analysis and interpretability of model weights. The approach was evaluated on its performance for the 2D incompressible Navier-Stokes equations. The following research questions were answered in Chapter 4:

- **RQ3: Can introducing a data-driven linear filter into the EFR framework enhance both computational efficiency and accuracy?**

  The learned filter yielded substantial improvements with regard to the accuracy of energy spectra, global energy, and enstrophy when compared to standard EFR, using a differential filter, and the Smagorinsky model in LES. Moreover, computational efficiency was improved by removing the need to solve a linear system, as required by traditional differential filtering approaches.

- **RQ4: How does one explicitly embed physical structure into the data-driven EFR framework, and how does this affect stability and performance?**

Embedding physical structure in the form of global energy and enstrophy dissipation constraints can be done through adaptive computation of the relaxation parameter, which enforces physically consistent dissipation. This stabilizes the learned filtering strategy during online simulations. This structure-preserving design showed improved stability and generalization, particularly in regimes with limited training data.

6.1.3 Topic III: Reduced-order modeling for advection-dominated flows (Chapter 5)

The aim of Chapter 5 was to address two key limitations of standard space-global proper orthogonal decomposition (POD) basis reduced-order models (ROMs) for advection-dominated flows: their high computational cost and poor generalizability. Both issues stem from the slow Kolmogorov N-width decay characteristic of such flows. To overcome these challenges, we exploit the self-similar nature of turbulence and propose a space-local POD basis. Unlike conventional POD modes, the proposed basis functions have local support, yielding sparse Galerkin projection-based ROMs. The following research questions were answered in Chapter 5:

- **RQ5: Can space-local reduced-order modeling strategies address the limited generalizability and high computational cost, and stability issues of classical space-global POD-Galerkin approaches while preserving accuracy?**

  The effectiveness of our space-local POD approach was demonstrated on both the 1D and 2D advection(-diffusion) equation. We found that using a space-local POD basis improves generalizability by reducing overfitting to the training data. This allowed the space-local ROMs to extrapolate in time, while the space-global ROMs were not able to do so. In addition, this approach yields significantly sparser and more computationally efficient ROMs than the space-global approach, for the same number of POD modes. This allows the use of a larger number of POD modes, without sacrificing computational efficiency. Furthermore, stability was derived by building the ROMs from a structure-preserving, and therefore stable, full-order model (FOM). Furthermore, it was found that space-local ROMs allow for larger time steps, while maintaining accuracy. This further improves their computational efficiency. Finally, the space-local approach yields a substantial speedup relative to the FOM.

- **RQ6: To what extent are the proposed space-local reduced-order models applicable to multi-dimensional fluid flows?**

  The applicability of the proposed space-local POD framework to multi-dimensional fluid flows was assessed using solutions of the 2D incompressible Navier-Stokes equations. The analysis focused on the ability of the resulting POD bases to represent flow states

outside the training data, thereby directly evaluating their generalization capabilities. The results show that the space-local POD bases generalize significantly better to evolving flow scenarios than their space-global counterpart. Although the construction of fully nonlinear space-local ROMs for turbulent flows remains an open research challenge, these findings demonstrate strong potential for extending the proposed methodology to complex multi-dimensional fluid dynamics problems.

After answering these research questions, we return to answering the central question of this thesis posed in Chapter 1:

- **RQ0: How can physical structure be embedded into data-driven approaches for modeling fluid flows, and does doing so lead to more stable, accurate, and generalizable models?**

Based on the answers to **RQ1-RQ6**, we find that embedding physical structure through (I) energy conservation in machine learning-based closure models for LES, (II) controlled dissipation for data-driven filters within the EFR framework, and (III) leveraging the local, self-similar nature of turbulence for POD-based ROMs yields data-driven fluid flow models that are consistently more stable, accurate, and generalizable than existing approaches.

6.2 Outlook and future work

Finally, we discuss the outlook and potential directions for future research for each topic, and conclude by placing the findings within the broader context of the field.

Topic I: Machine learning-based closure models for large eddy simulation The integration of machine learning closures into LES remains an active and rapidly developing field, with many open challenges. An immediate extension of the compressed SGS representation introduced in Chapter 2 is its generalization from one to multiple spatial dimensions [15]. Such an extension would allow the energy conservation law derived in that chapter to be applied more broadly. Promising opportunities also lie in combining the proposed structure-preserving methodology with advanced neural network architectures, such as Fourier neural operators [93], which have demonstrated strong predictive capabilities and reduced dependence on the training mesh. Graph neural networks [107] further provide a pathway toward extending the framework to unstructured grids. Additionally, incorporating spatial symmetries [188] may enhance data efficiency and improve extrapolation performance.

Topic II: Data-driven extensions of the evolve-filter-relax framework This thesis has shown that the use of FFTs enables the efficient learning of data-driven filters in

Fourier space, which is particularly effective for periodic domains. For non-periodic domains with boundary conditions, however, the choice of filter requires reconsideration. Moreover, the current diagonal matrix representation of the filter could be enriched by introducing additional parameters, potentially enhancing expressiveness. Further improvements may also be achieved by incorporating physical constraints more effectively. Finally, the methodologies developed here could also be applied within a ROM framework [189].

Topic III: Reduced-order modeling for advection-dominated flows A natural continuation of the work on space-local POD-Galerkin ROMs is to apply the methodology to the incompressible Navier–Stokes equations. As demonstrated in [67], such ROMs can preserve the energy-conserving properties of the quadratic advection operator. However, ensuring that the space-local basis remains divergence-free is crucial, as this condition is necessary for energy conservation. Another promising direction is the extension of this approach to unstructured domains, where techniques from the finite element community [106] may provide valuable insights. An additional challenge lies in addressing discontinuities between subdomain boundaries in the space-local formulation. Current strategies rely on overlapping subdomains, which mitigate discontinuities but reduce computational efficiency. Developing more efficient alternatives remains an open problem.

6.2.1 General outlook

The data-driven methodologies presented in this thesis provide a foundation for accelerating fluid flow simulations. However, their practical adoption, particularly for industrial-scale problems involving complex geometries and boundary conditions [190, 191], requires addressing several fundamental challenges:

One aspect that has not received much attention in this thesis is parametric extrapolation. While Chapter 2 examined the effect of varying viscosity, the ability to generalize across a wide range of physical parameters, such as Reynolds number, inflow conditions, or geometric variations, is essential for practical application of the method. Transfer learning offers a pathway to adapt pre-trained models to new parameter regimes with minimal additional cost [51]. Another solution may lie in building parametric dependencies directly into the model architecture, for instance, by using the physical parameters as inputs for a neural network. This approach could be integrated with the structure-preserving neural network architecture discussed in Chapters 2 and 3.

Closely related is the issue of grid dependence. Models trained on one discretization are often incompatible with different grids. Although Chapter 2 demonstrated that CNNs can accommodate different grid sizes, they remain tightly coupled to a specific grid spacing. Emerging architectures offer a path toward true grid independence: Fourier neural operators

[93] learn mappings in frequency space, making them naturally resolution-invariant.

Underpinning both of these challenges is the question of data efficiency [2, 6]. Generating high-fidelity DNS data for training is inherently expensive. This makes it important to extract the most information from the fewest samples. In Chapter 4, we showed that embedding physical structure directly into the approach can significantly improve data efficiency. Chapter 5 further demonstrated that a space-local approach to ROM construction can generalize to flow regimes where space-global methods fail, by exploiting the self-similar nature of turbulence. In this way, we effectively learn more from the same data. In that regard, building spatial symmetries into the method can potentially also lead to better data efficiency [188, 192, 193]. In addition, generative models may offer a promising avenue to address this limitation by synthesizing physically consistent flow fields for unseen parameter regimes or geometric configurations. This can serve as augmented training data for other data-driven approaches [194, 195]. Future work should not only pursue novel architectures that reach higher levels of accuracy, but also evaluate them based on data efficiency. Treating data efficiency as a primary design criterion rather than an afterthought.

Ultimately, the path toward industrially relevant data-driven simulation lies at the intersection of these challenges. Progress will require moving beyond proof-of-concept demonstrations on canonical problems toward methods that are grid-agnostic, generalizable, and data-efficient. By leveraging structure-preserving principles, this thesis provides a good starting point for addressing the latter two criteria.

After further development, the methodologies introduced in this thesis could be applied to a wide range of applications where repeated or real-time fluid simulations are required. Examples of such application areas include the automotive, maritime, and aerospace sectors, where design optimization tasks require repeated simulations [81, 82]. These simulations are needed to accurately assess the interaction of a structure with its environment. Fast surrogate models, such as those presented in this thesis, could substantially reduce computational costs and shorten design cycles [170]. Furthermore, one could consider environmental applications, such as urban wind flow modeling, spatial pollution dispersion, and flood prediction, where rapid simulations can greatly assist decision-making. From such applications, one can naturally make the link to digital twins, for example, of large cities, for which the digital twin model is continuously updated with sensor data [196]. Another application would be in the energy sector, where the developed methodologies could aid in wind farm design as well as real-time control [24]. A final application would be in climate studies, which require long-time simulations. The presented methodologies could be particularly useful here, as their stability properties support such long-time simulations [102].

Bibliography

[1] George K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 2000.

[2] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[3] Joel H. Ferziger and Milovan Perić. *Computational Methods for Fluid Dynamics*. Springer, Berlin, third edition, 2002.

[4] Parviz Moin and Krishnan Mahesh. DIRECT NUMERICAL SIMULATION: A Tool in Turbulence Research. *Annual Review of Fluid Mechanics*, 30(Volume 30, 1998):539–578, 1998.

[5] Hendrik Tennekes and John L. Lumley. *A First Course in Turbulence*. MIT Press, 1972.

[6] P. Sagaut and C. Meneveau. *Large Eddy Simulation for Incompressible Flows: An Introduction*. Scientific Computation. Springer, 2006.

[7] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, USA, 1st edition, 2019.

[8] Kunihiko Taira, Maziar S. Hemati, Steven L. Brunton, Yiyang Sun, Karthik Duraisamy, Shervin Bagheri, Scott T. M. Dawson, and Chi-An Yeh. Modal Analysis of Fluid Flows: Applications and Outlook. *AIAA Journal*, 58(3):998–1022, 2020.

[9] Giancarlo Alfonsi. Reynolds-Averaged Navier-Stokes Equations for Turbulence Modeling. *Applied Mechanics Reviews - APPL MECH REV*, 62, 07 2009.

[10] Douglas K Lilly. On the numerical simulation of buoyant convection. *Tellus*, 14(2):148–172, 1962.

[11] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence Modeling in the Age of Data. *Annual Review of Fluid Mechanics*, 51(1):357–377, January 2019.

[12] Marius Kurz and Andrea Beck. Investigating Model-Data Inconsistency in Data-Informed Turbulence Closure Terms. In *14th WCCM-ECCOMAS Congress 2020*, 02 2021.

[13] Toby van Gastelen, Wouter Edeling, and Benjamin Sanderse. Energy-Conserving Neural Network Closure Model for Long-Time Accurate and Stable 2D LES, 2025.

[14] Jonghwan Park and Haecheon Choi. Toward neural-network-based large eddy simulation: application to turbulent channel flow. *Journal of Fluid Mechanics*, 914:A16, 2021.

[15] T. van Gastelen, W. Edeling, and B. Sanderse. Energy-conserving neural network for turbulence closure modeling. *Journal of Computational Physics*, 508:113003, 2024.

[16] Benjamin Sanderse, Panos Stinis, Romit Maulik, and Shady E. Ahmed. Scientific machine learning for closure models in multiscale problems: a review, 2024.

[17] Annalisa Quaini, Omer San, Alessandro Veneziani, and Traian Iliescu. Bridging Large Eddy Simulation and Reduced Order Modeling of Convection-Dominated Flows through Spatial Filtering: Review and Perspectives, 2024.

[18] Michele Girfoglio, Annalisa Quaini, and Gianluigi Rozza. A Finite Volume approximation of the Navier-Stokes equations with nonlinear filtering stabilization. *Computers & Fluids*, 187:27–45, 2019.

[19] Tommaso Taddei, Simona Perotto, and ALFIO Quarteroni. Reduced basis techniques for nonlinear conservation laws. *ESAIM: Mathematical Modelling and Numerical Analysis*, 49(3):787–814, 2015.

[20] Constantin Greif and Karsten Urban. Decay of the Kolmogorov N-width for wave problems. *Applied Mathematics Letters*, 96:216–222, 2019.

[21] L. D. Landau and E. M. Lifshitz. *Fluid Mechanics*. Pergamon Press, 2nd edition, 1987.

[22] F.X. Trias, O. Lehmkuhl, A. Oliva, C.D. Pérez-Segarra, and R.W.C.P. Verstappen. Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids. *Journal of Computational Physics*, 258:246–267, 2014.

[23] Roel Verstappen and Arthur Veldman. Symmetry-preserving discretization of turbulent flow. *Journal of Computational Physics*, 187:343–368, 05 2003.

[24] B. Sanderse. *Energy-conserving discretization methods for the incompressible Navier-Stokes equations : application to the simulation of wind-turbine wakes*. Phd thesis, Technische Universiteit Eindhoven, 2013.

[25] Francis H Harlow, J Eddie Welch, et al. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, 8(12):2182, 1965.

[26] Joseph Smagorinsky. General circulation experiments with the primitive equations: I. The basic experiment. *Monthly weather review*, 91(3):99–164, 1963.

[27] A. Leonard. Energy Cascade in Large-Eddy Simulations of Turbulent Fluid Flows. In F.N. Frenkiel and R.E. Munn, editors, *Turbulent Diffusion in Environmental Pollution*, volume 18 of *Advances in Geophysics*, pages 237–248. Elsevier, 1975.

[28] Uriel Frisch. *Turbulence: The Legacy of A. N. Kolmogorov*. Cambridge University Press, 1995.

[29] Charles Meneveau and Joseph Katz. Scale-Invariance and Turbulence Models for Large-Eddy Simulation. *Annual Review of Fluid Mechanics*, 32(Volume 32, 2000):1–32, 2000.

[30] Ugo Piomelli, William H. Cabot, Parviz Moin, and Sangsan Lee. Subgrid-scale backscatter in turbulent and transitional flows. *Physics of Fluids A*, 3(7):1766 – 1771, 1991. Cited by: 315.

[31] Ugo Piomelli, Thomas A. Zang, Charles G. Speziale, and M. Yousuff Hussaini. On the large-eddy simulation of transitional wall-bounded flows. *Physics of Fluids A*, 2(2):257 – 265, 1990. Cited by: 96; All Open Access, Green Open Access.

[32] Massimo Germano, Ugo Piomelli, Parviz Moin, and William H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, 07 1991.

[33] Jorge Bardina, Joel H. Ferziger, and William C. Reynolds. Improved subgrid models for large eddy simulation. *AIAA Paper*, 80:1357, 1980.

[34] Ugo Piomelli and Junhui Liu. Large-eddy simulation of rotating channel flows using a localized dynamic model. *Physics of Fluids*, 7(4):839–848, 04 1995.

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[36] Oliver T. Unke and Markus Meuwly. PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *Journal of Chemical Theory and Computation*, 15(6):3678–3693, May 2019.

[37] Kevin Linka, Amelie Schäfer, Xuhui Meng, Zongren Zou, George Em Karniadakis, and Ellen Kuhl. Bayesian Physics Informed Neural Networks for real-world nonlinear dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 402:115346, December 2022.

[38] Andrea Beck, David Flad, and Claus-Dieter Munz. Deep neural networks for data-driven LES closure models. *Journal of Computational Physics*, 398:108910, 2019.

[39] Björn List, Li-Wei Chen, and Nils Thuerey. Learned Turbulence Modelling with Differentiable Fluid Solvers, 2022.

[40] Syver Døving Agdestein and Benjamin Sanderse. Discretize first, filter next: Learning divergence-consistent closure models for large-eddy simulation. *Journal of Computational Physics*, 522:113577, February 2025.

[41] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014.

[42] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. volume 15, 01 2010.

[43] Midhun T Augustine. A survey on universal approximation theorems, 2024.

[44] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, 2015.

[45] Marius Kurz and Andrea Beck. A machine learning framework for LES closure terms. *ETNA - Electronic Transactions on Numerical Analysis*, 56:117–137, 01 2022.

[46] Marius Kurz, Philipp Offenhäuser, and Andrea Beck. Deep Reinforcement Learning for Turbulence Modeling in Large Eddy Simulations, 2022.

[47] Varun Shankar, Dibyajyoti Chakraborty, Venkatasubramanian Viswanathan, and Romit Maulik. Differentiable Turbulence: Closure as a partial differential equation constrained optimization, 2024.

[48] R. Maulik, O. San, A. Rasheed, and P. Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, November 2018.

[49] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), May 2021.

[50] Anudhyan Boral, Zhong Yi Wan, Leonardo Zepeda-Núñez, James Lottes, Qing Wang, Yi-Fan Chen, John Anderson, and Fei Sha. Neural Ideal Large Eddy Simulation: Modeling Turbulence with Neural Stochastic Differential Equations. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 69270–69283. Curran Associates, Inc., 2023.

[51] Yifei Guan, Ashesh Chattopadhyay, Adam Subel, and Pedram Hassanzadeh. Stable a posteriori LES of 2D turbulence using convolutional neural networks: Backscattering analysis and generalization to higher Re via transfer learning. *Journal of Computational Physics*, 458:111090, 2022.

[52] Hugo Frezat, Julien Le Sommer, Ronan Fablet, Guillaume Balarac, and Redouane Lguensat. A posteriori learning of quasi-geostrophic turbulence parametrization: an experiment on integration steps, 2021.

[53] Jonathan F. MacArt, Justin Sirignano, and Jonathan B. Freund. Embedded training of neural-network subgrid-scale turbulence models. *Phys. Rev. Fluids*, 6:050502, May 2021.

[54] Syver Døving Agdestein and Benjamin Sanderse. Learning filtered discretization operators: non-intrusive versus intrusive approaches, 2022.

[55] Hugo Melchers, Daan Crommelin, Barry Koren, Vlado Menkovski, and Benjamin Sanderse. Comparison of neural closure models for discretised PDEs. *ArXiv preprint arXiv:2210.14675*, 2022.

[56] H. Jane Bae and Petros Koumoutsakos. Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nature Communications*, 13(1):1443, Mar 2022.

[57] Vincent J. Ervin, William J. Layton, and Monika Neda. Numerical Analysis of Filter-Based Stabilization for Evolution Equations. *SIAM Journal on Numerical Analysis*, 50(5):2307–2335, 2012.

[58] Germano, Massimo. Differential filters for the large eddy numerical simulation of turbulent flows. *The Physics of fluids*, 29(6):1755–1757, 1986.

[59] Paul Fischer and Julia Mullen. Filter-based stabilization of spectral element methods. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 332(3):265–270, 2001.

[60] Luigi C Berselli, Traian Iliescu, William J Layton, et al. *Mathematics of large eddy simulation of turbulent flows*, volume 2. Springer, 2006.

[61] Michael A. Golberg. The method of fundamental solutions for Poisson's equation. *Engineering Analysis with Boundary Elements*, 16(3):205–213, 1995.

[62] L Bertagna, A Quaini, and A Veneziani. Deconvolution-based nonlinear filtering for incompressible flows at moderately large Reynolds numbers. *International Journal for Numerical Methods in Fluids*, 81(8):463–488, 2016.

[63] Strazzullo, Maria and Girfoglio, Michele and Ballarin, Francesco and Iliescu, Traian and Rozza, Gianluigi. Consistency of the full and reduced order models for Evolve-Filter-Relax regularization of convection-dominated, marginally-resolved flows. *International Journal for Numerical Methods in Engineering*, 123(14):3148–3178, 2022.

[64] Anna Ivagnes, Maria Strazzullo, Michele Girfoglio, Traian Iliescu, and Gianluigi Rozza. Data-Driven Optimization for the Evolve-Filter-Relax Regularization of Convection-Dominated Flows. *International Journal for Numerical Methods in Engineering*, 126(9):e70042, 2025.

[65] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, pages 353–374, 2023.

[66] Sourav Dutta, Peter Rivera-Casillas, Brent Styles, and Matthew W. Farthing. Reduced Order Modeling Using Advection-Aware Autoencoders. *Mathematical and Computational Applications*, 27(3), 2022.

[67] Benjamin Sanderse. Non-linearly stable reduced-order models for incompressible flow with energy-conserving finite volume methods. *Journal of Computational Physics*, 421:109736, 07 2020.

[68] Maria-Luisa Rapun and José Vega. Reduced order models based on local POD plus Galerkin projection. *Journal of Computational Physics*, 229, 04 2010.

[69] Henrik Rosenberger and Benjamin Sanderse. No pressure? Energy-consistent ROMs for the incompressible Navier-Stokes equations with time-dependent boundary conditions. *Journal of Computational Physics*, 491:112405, 2023.

[70] Florian Arbes, Constantin Greif, and Karsten Urban. The Kolmogorov N-width for linear transport: Exact representation and the influence of the data, 2024.

[71] Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[72] Nirmal J Nair and Maciej Balajewicz. Transported snapshot model order reduction approach for parametric, steady-state fluid flows containing parameter-dependent shocks. *International Journal for Numerical Methods in Engineering*, 117(12):1234–1262, 2019.

[73] Kyle Washabaugh, David Amsallem, Matthew Zahr, and Charbel Farhat. Nonlinear Model Reduction for CFD Problems Using Local Reduced Order Bases. 06 2012.

[74] R.B. Klein and B. Sanderse. Energy-conserving hyper-reduction and temporal localization for reduced order models of the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 499:112697, 2024.

[75] Benjamin Peherstorfer and Karen Willcox. Online Adaptive Model Reduction for Nonlinear Systems via Low-Rank Updates. *SIAM Journal on Scientific Computing*, 37(4):A2123–A2150, 2015.

[76] Sebastian Grimberg, Charbel Farhat, and Noah Youkilis. On the stability of projection-based model order reduction for convection-dominated laminar and turbulent flows. *Journal of Computational Physics*, 419:109681, 2020.

[77] Zhu Wang, Imran Akhtar, Jeff Borggaard, and Traian Iliescu. Proper orthogonal decomposition closure models for turbulent flows: A numerical comparison. *Computer Methods in Applied Mechanics and Engineering*, 237-240:10–26, 2012.

[78] Robin Klein, Benjamin Sanderse, Pedro Costa, Rene Pecnik, and Ruud Henkes. Entropy-Stable Model Reduction of One-Dimensional Hyperbolic Systems using Rational Quadratic Manifolds, 2024.

[79] Arvind T. Mohan and Datta V. Gaitonde. A Deep Learning based Approach to Reduced Order Modeling for Turbulent Flow Control using LSTM Neural Networks, 2018.

[80] Nikolaj T. Mücke, Sander M. Bohté, and Cornelis W. Oosterlee. Reduced order modeling for parameterized time-dependent PDEs using spatially and memory aware deep learning. *Journal of Computational Science*, 53:101408, 2021.

[81] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*, volume 12. Siam, 2013.

[82] Daisuke Sasaki, Shigeru Obayashi, and Kazuhiro Nakahashi. Navier-Stokes optimization of supersonic wings with four objectives using evolutionary algorithm. *Journal of Aircraft*, 39(4):621–629, 2002.

[83] J. J. O'Neill, X.-M. Cai, and R. Kinnersley. A generalised stochastic backscatter model: large-eddy simulation of the neutral surface layer. *Quarterly Journal of the Royal Meteorological Society*, 141(692):2617–2629, 2015.

[84] Daniele Carati, Sandip Ghosal, and Parviz Moin. On the representation of backscatter in dynamic localization models. *Physics of Fluids*, 7(3):606 – 616, 1995. Cited by: 115.

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

[86] Aviral Prakash, Kenneth E. Jansen, and John A. Evans. Optimal Clipping of Structural Subgrid Stress Closures for Large Eddy Simulation, 2022.

[87] Sandip Ghosal, Thomas S. Lund, Parviz Moin, and Knut Akselvoll. A dynamic localization model for large-eddy simulation of turbulent flows. *Journal of Fluid Mechanics*, 286:229–255, 1995.

[88] Yuxi Li. Deep Reinforcement Learning: An Overview. *CoRR*, abs/1701.07274, 2017.

[89] Varun Shankar, Vedant Puri, Ramesh Balakrishnan, Romit Maulik, and Venkatasubramanian Viswanathan. Differentiable physics-enabled closure modeling for Burgers' turbulence. *Machine Learning: Science and Technology*, 4(1):015017, feb 2023.

[90] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

[91] Adam Subel, Ashesh Chattopadhyay, Yifei Guan, and Pedram Hassanzadeh. 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, 03 2021.

[92] Shinhoo Kang and Emil M. Constantinescu. Learning Subgrid-scale Models with Neural Ordinary Differential Equations, 2023.

[93] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *CoRR*, abs/2010.08895, 2020.

[94] M. D. Love. Subgrid modelling studies with Burgers' equation. *Journal of Fluid Mechanics*, 100(1):87–110, 1980.

[95] Antony Jameson. The Construction of Discretely Conservative Finite Volume Schemes that Also Globally Conserve Energy or Entropy. *J. Sci. Comput.*, 34:152–187, 02 2008.

[96] J. Butcher. Runge-Kutta methods. *Scholarpedia*, 2(9):3147, 2007. revision #91735.

[97] Jin-Liang Yan and Liang-Hong Zheng. A Class of Momentum-Preserving Finite Difference Schemes for the Korteweg-de Vries Equation. *Computational Mathematics and Mathematical Physics*, 59(10):1582–1596, Oct 2019.

[98] Carlos A De Moura and Carlos S Kubrusly. The courant–friedrichs–lewy (cfl) condition. *AMC*, 10(12), 2013.

[99] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[100] Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah. Fashionable Modelling with Flux. *CoRR*, abs/1811.01457, 2018.

[101] Mike Innes. Flux: Elegant Machine Learning with Julia. *Journal of Open Source Software*, 2018.

[102] Wouter Edeling and Daan Crommelin. Reducing data-driven dynamical subgrid scale models by physical constraints. *Computers & Fluids*, 201:104470, 2020.

[103] Sara Moradi, Bogdan Teaca, and Johan Anderson. Role of phase synchronisation in turbulence. *AIP Advances*, 7(11):115213, 11 2017.

[104] Varun Shankar, Vedant Puri, Ramesh Balakrishnan, Romit Maulik, and Venkatasubramanian Viswanathan. Differentiable physics-enabled closure modeling for Burgers' turbulence, 2022.

[105] Weglarczyk, Stanislaw. Kernel density estimation and its application. *ITM Web Conf.*, 23:00037, 2018.

[106] Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms*, volume 5. Springer Science & Business Media, 2012.

[107] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In *international conference on machine learning*, pages 2402–2411. PMLR, 2020.

[108] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Sparse Identification of Nonlinear Dynamics with Control (SINDYc)**SLB acknowledges support from the U.S. Air Force Center of Excellence on Nature Inspired Flight Technologies and Ideas (FA9550-14-1-0398). JLP thanks Bill and Melinda Gates for their active support of the Institute of Disease Modeling and their sponsorship through the Global Good Fund. JNK acknowledges support from the U.S. Air Force Office of Scientific Research (FA9550-09-0174). *IFAC-PapersOnLine*, 49(18):710–715, 2016. 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016.

[109] FR Menter and Y Egorov. The scale-adaptive simulation method for unsteady turbulent flow predictions. Part 1: theory and model description. *Flow, turbulence and combustion*, 85(1):113–138, 2010.

[110] JFH Buist, Benjamin Sanderse, Svetlana Dubinkina, RAWM Henkes, and CW Oosterlee. Energy-conserving formulation of the two-fluid model for incompressible two-phase flow in channels and pipes. *arXiv preprint arXiv:2104.07728*, 2021.

[111] Gary J. Chandler and Rich R. Kerswell. Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow. *Journal of Fluid Mechanics*, 722:554–595, 2013.

[112] Roustam Zalaletdinov. Averaging out Inhomogeneous Newtonian Cosmologies: II. Newtonian Cosmology and the Navier-Stokes-Poisson Equations, 2002.

[113] John C. Strikwerda. Finite Difference Methods for the Stokes and Navier–Stokes Equations. *SIAM Journal on Scientific and Statistical Computing*, 5(1):56–68, 1984.

[114] Gennaro Coppola and Arthur E. P. Veldman. Global and local conservation of mass, momentum and kinetic energy in the simulation of compressible flow. *Journal of Computational Physics*, 475:111879, February 2023.

[115] Shaoshuai Chu, Alexander Kurganov, and Ruixiao Xin. New Low-Dissipation Central-Upwind Schemes. Part II, 2024.

[116] J. Andrzej Domaradzki, Ralph W. Metcalfe, Robert S. Rogallo, and James J. Riley. Analysis of subgrid-scale eddy viscosity with use of results from direct numerical simulations. *Phys. Rev. Lett.*, 58:547–550, Feb 1987.

[117] Ian Grooms, Yoonsang Lee, and Andrew J. Majda. Numerical Schemes for Stochastic Backscatter in the Inverse Cascade of Quasigeostrophic Turbulence. *Multiscale Modeling & Simulation*, 13(3):1001–1021, 2015.

[118] Malte Jansen, Isaac Held, Alistair Adcroft, and Robert Hallberg. Energy budget-based backscatter in an eddy permitting primitive equation model. *Ocean Modelling*, 94, 07 2015.

[119] Ayaboe Edoh and Ann R. Karagozian. Stabilized Scale-Similarity Modeling for Explicitly-Filtered Large-Eddy Simulations. In *55th AIAA Aerospace Sciences Meeting*, Grapevine, Texas, January 2017. American Institute of Aeronautics and Astronautics.

[120] Prahladh S. Iyer and Mujeeb R. Malik. Efficient dynamic mixed subgrid-scale model. *Physical Review Fluids*, 9(9):L092601, September 2024.

[121] Yipeng Shi, Zuoli Xiao, and Shiyi Chen. Constrained subgrid-scale stress model for large eddy simulation. *Physics of Fluids*, 20(1):011701, January 2008.

[122] Helene T. Hewitt, Malcolm Roberts, Pierre Mathiot, Arne Biastoch, Ed Blockley, Eric P. Chassignet, Baylor Fox-Kemper, Pat Hyder, David P. Marshall, Ekaterina Popova, Anne-Marie Treguier, Laure Zanna, Andrew Yool, Yongqiang Yu, Rebecca Beadling, Mike Bell, Till Kuhlbrodt, Thomas Arsouze, Alessio Bellucci, Fred Castruccio, Bolan Gan, Dian Putrasahan, Christopher D. Roberts, Luke Van Roekel, and Qiuying Zhang. Resolving and Parameterising the Ocean Mesoscale in Earth System Models. *Current Climate Change Reports*, 6(4):137–152, Dec 2020.

[123] Aviral Prakash, Kenneth E. Jansen, and John A. Evans. Invariant data-driven subgrid stress modeling in the strain-rate eigenframe for large eddy simulation. *Computer Methods in Applied Mechanics and Engineering*, 399:115457, 2022.

[124] Quercus Hernandez, Alberto Badias, Francisco Chinesta, and Elias Cueto. Thermodynamics-informed Graph Neural Networks. *IEEE Transactions on Artificial Intelligence*, pages 1–1, 2022.

[125] John Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd, 08 2016.

[126] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2 edition, 2012.

[127] Heinz H. Bauschke, Xianfu Wang, and Liangjin Yao. On borwein-wiersma decompositions of monotone linear relations, 2009.

[128] Giovanni Gallavotti. *Foundations of Fluid Dynamics*. Springer, Berlin, Heidelberg, 2002.

[129] B. Sanderse and B. Koren. Accuracy analysis of explicit Runge–Kutta methods applied to the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 231(8):3041–3063, 2012.

[130] Thierry Poinsot and Sébastien M. Candel. The influence of differencing and CFL number on implicit time-dependent non-linear calculations. *Journal of Computational Physics*, 62(2):282–296, 1986.

[131] SURF. Snellius: the National Supercomputer. Accessed: 2025-03-21.

[132] Thomas B Gatski, M Yousuff Hussaini, and John L Lumley. *Simulation and Modeling of Turbulent Flows*. Oxford University Press, 09 1996.

[133] V. M. Canuto and Y. Cheng. Determination of the Smagorinsky–Lilly constant CS. *Physics of Fluids*, 9(5):1368–1378, 05 1997.

[134] Y. Bartosiewicz and M. Duponcheel. 6.1.2 - Large-eddy simulation: Application to liquid metal fluid flow and heat transfer. In Ferry Roelofs, editor, *Thermal Hydraulics Aspects of Liquid Metal Cooled Nuclear Reactors*, pages 245–271. Woodhead Publishing, 2019.

[135] Robert H. Kraichnan. Inertial Ranges in Two-Dimensional Turbulence. *The Physics of Fluids*, 10(7):1417–1423, 07 1967.

[136] Z. C. Feng and Y. C. Li. Short-term unpredictability of high reynolds number turbulence — rough dependence on initial data. *arXiv preprint*, 2017. Preprint available at `https://arxiv.org/abs/1702.02993`.

[137] Peter V Coveney, Edward R Dougherty, and Roger R Highfield. Big data need big theory too. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2080):20160153, 2016.

[138] Anna Ivagnes, Toby van Gastelen, Syver Døving Agdestein, Benjamin Sanderse, Giovanni Stabile, and Gianluigi Rozza. A new data-driven energy-stable evolve-filter-relax model for turbulent flow simulation. *Computer Methods in Applied Mechanics and Engineering*, 450:118654, 2026.

[139] P.J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, 1998.

[140] Parviz Moin and Krishnan Mahesh. Direct Numerical Simulation: A Tool in Turbulence Research. *Annu. Rev. Fluid Mech.*, 30:539–578, 01 1998.

[141] D.C. Wilcox. *Turbulence Modeling for CFD*. Number v. 1 in Turbulence Modeling for CFD. DCW Industries, 2006.

[142] Alexander N. Brooks and Thomas J.R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1):199–259, 1982.

[143] S. Scott Collis and Matthias Heinkenschloss. Analysis of the SUPG Method for the Solution of Optimal Control Problems, 2024.

[144] Girfoglio, Michele and Quaini, Annalisa and Rozza, Gianluigi. A novel Large Eddy Simulation model for the Quasi-Geostrophic equations in a Finite Volume setting. *Journal of Computational and Applied Mathematics*, 418:114656, 2023.

[145] Besabe, Lander and Girfoglio, Michele and Quaini, Annalisa and Rozza, Gianluigi. Linear and nonlinear filtering for a two-layer quasi-geostrophic ocean model. *Applied Mathematics and Computation*, 488:129121, 2025.

[146] Xu, Huijuan and Baroli, Davide and Di Massimo, Francesca and Quaini, Annalisa and Veneziani, Alessandro. Backflow stabilization by deconvolution-based large eddy simulation modeling. *Journal of Computational Physics*, 404:109103, 2020.

[147] Tsai, P. H. *Parametric model order reduction development for Navier-Stokes equations from 2D chaotic to 3D turbulent flow problems*. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, Stati Uniti, 2023.

[148] Layton, W. J. and Rebholz, L. G. *Approximate Deconvolution Models of Turbulence: Analysis, Phenomenology and Numerical Analysis*, volume 2042. Springer Berlin Heidelberg, 2012.

[149] Peter Benner, Serkan Gugercin, and Karen Willcox. A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems. *SIAM Review*, 57(4):483–531, 2015.

[150] Canuto, Claudio and Hussaini, M Yousuff and Quarteroni, Alfio and Zang, Thomas A. Some Algorithms for Unsteady Navier—Stokes Equations. *Spectral Methods in Fluid Dynamics*, pages 201–239, 1988.

[151] Girfoglio, Michele and Quaini, Annalisa and Rozza, Gianluigi. Fluid-structure interaction simulations with a LES filtering approach in. *Communications in Applied and Industrial Mathematics*, 12(1):13–28, 2021.

[152] Xie, X. and Bao, F. and Webster, C. G. Evolve filter stabilization reduced-order model for stochastic Burgers equation. *Fluids*, 3(4):84, 2018.

[153] Tsai, P. H. and Fischer, P. and Iliescu, T. A time-relaxation reduced order model for the turbulent channel flow. *Journal of Computational Physics*, 521:113563, 2025.

[154] Luca Bertagna, Annalisa Quaini, Leo G Rebholz, and Alessandro Veneziani. On the sensitivity to the filtering radius in Leray models of incompressible flow. In *Contributions to Partial Differential Equations and Applications*, pages 111–130. Springer, 2018.

[155] Huijuan Xu, Davide Baroli, and Alessandro Veneziani. Global sensitivity analysis for patient-specific aortic simulations: the role of geometry, boundary condition and large eddy simulation modeling parameters. *Journal of Biomechanical Engineering*, 143(2):021012, 2021.

[156] Marius Kurz, Philipp Offenhäuser, and Andrea Beck. Deep reinforcement learning for turbulence modeling in large eddy simulations. *International journal of heat and fluid flow*, 99:109094, 2023.

[157] Anudhyan Boral, Zhong Yi Wan, Leonardo Zepeda-Núñez, James Lottes, Qing Wang, Yi-Fan Chen, John Anderson, and Fei Sha. Neural Ideal Large Eddy Simulation: Modeling Turbulence with Neural Stochastic Differential Equations. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 69270–69283. Curran Associates, Inc., 2023.

[158] Syver Døving Agdestein, Simone Ciarella, Benjamin Sanderse, and Rik Hoekstra. IncompressibleNavierStokes.jl. `https://github.com/agdestein/IncompressibleNavierStokes.jl`.

[159] Vincent J Ervin, William J Layton, and Monika Neda. Numerical analysis of filter-based stabilization for evolution equations. *SIAM Journal on Numerical Analysis*, 50(5):2307–2335, 2012.

[160] Julie S Mullen and Paul F Fischer. Filtering techniques for complex geometry fluid flows. *Communications in numerical methods in engineering*, 15(1):9–18, 1999.

[161] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley, MA, fourth edition, 2009.

[162] Brian J. Olson, Steven W. Shaw, Chengzhi Shi, Christophe Pierre, and Robert G. Parker. Circulant Matrices and Their Application to Vibration Analysis. *Applied Mechanics Reviews*, 66(4):040803, 06 2014.

[163] Haim Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer, New York, 2011.

[164] P. Duhamel and M. Vetterli. Fast fourier transforms: A tutorial review and a state of the art. *Signal Processing*, 19(4):259–299, 1990.

[165] A. Palha and M. Gerritsma. A mass, energy, enstrophy and vorticity conserving (MEEVC) mimetic spectral element discretization for the 2D incompressible Navier–Stokes equations. *Journal of Computational Physics*, 328:200–220, January 2017.

[166] Benjamin Sanderse and Barry Koren. Accuracy analysis of explicit Runge–Kutta methods applied to the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 231(8):3041–3063, 2012.

[167] Paolo Orlandi. *Fluid flow phenomena: a numerical toolkit*, volume 55. Springer Science & Business Media, 2000.

[168] Omer San and Anne E Staples. High-order methods for decaying two-dimensional homogeneous isotropic turbulence. *Computers & Fluids*, 63:105–127, 2012.

[169] T. Van Gastelen, W. Edeling, and B. Sanderse. Modeling advection-dominated flows with space-local reduced-order models. *Computers & Fluids*, 305:106911, 2026.

[170] Peter Benner, Serkan Gugercin, and Karen Willcox. A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems. *SIAM Review*, 57(4):483–531, 2015.

[171] John Burkardt, Max Gunzburger, and Hyung-Chun Lee. POD and CVT-based reduced-order modeling of Navier–Stokes flows. *Computer Methods in Applied Mechanics and Engineering*, 196:337–355, 12 2006.

[172] J. N. Reddy. *Introduction to the Finite Element Method*. McGraw-Hill Education, New York, 4th edition edition, 2019.

[173] Spenser Anderson, Tina White, and Charbel Farhat. Space-local reduced-order bases for accelerating reduced-order models through sparsity. *International Journal for Numerical Methods in Engineering*, 124, 11 2022.

[174] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: An introduction*. 01 2015.

[175] Seung Whan Chung, Youngsoo Choi, Pratanu Roy, Thomas Moore, Thomas Roy, Tiras Y. Lin, Du T. Nguyen, Christopher Hahn, Eric B. Duoss, and Sarah E. Baker.

Train small, model big: Scalable physics simulators via reduced order modeling and domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 427:117041, 2024.

[176] B. Sanderse. Energy-conserving Runge–Kutta methods for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 233:100–131, 2013.

[177] John G Aiken, John A Erdos, and Jerome A Goldstein. On löwdin orthogonalization. *International Journal of Quantum Chemistry*, 18(4):1101–1108, 1980.

[178] Xiaocan Li, Shuo Wang, and Yinghao Cai. Tutorial: Complexity analysis of Singular Value Decomposition and its variants, 2019.

[179] J.M. Melenk and I. Babuška. The partition of unity finite element method: Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139(1):289–314, 1996.

[180] Sever S Dragomir, Pietro Cerone, and Anthony Sofo. Some remarks on the midpoint rule in numerical integration. *RGMIA research report collection*, 1(2), 1998.

[181] Răzvan Ştefănescu and Ionel Michael Navon. POD/DEIM nonlinear model order reduction of an ADI implicit shallow water equations model. *Journal of Computational Physics*, 237:95–114, 2013.

[182] Van Bo Nguyen, Si Bui Quang Tran, Saif A. Khan, Jiawei Rong, and Jing Lou. POD-DEIM model order reduction technique for model predictive control in continuous chemical processing. *Computers & Chemical Engineering*, 133:106638, 2020.

[183] Michael Baldauf. Stability analysis for linear discretisations of the advection equation with Runge–Kutta time integration. *Journal of Computational Physics*, 227(13):6638–6659, 2008.

[184] Claes Johnson. Error Estimates and Automatic Time Step Control for Nonlinear Parabolic Problems, I. *SIAM Journal on Numerical Analysis*, 24(1):1–14, 1987.

[185] William Snyder, Changhong Mou, Honghu Liu, Omer San, Raffaella De Vita, and Traian Iliescu. Reduced Order Model Closures: A Brief Tutorial, 2022.

[186] R.C. Mittal and A. Al-Kurdi. LU-decomposition and numerical structure for solving large sparse nonsymmetric linear systems. *Computers & Mathematics with Applications*, 43(1):131–155, 2002.

[187] Julien Lequeurre and Alexandre Munnier. Vorticity and stream function formulations for the 2D Navier-Stokes equations in a bounded domain, 2018.

[188] Giulio Ortali, Alessandro Gabbana, Imre Atmodimedjo, and Alessandro Corbetta. Enhancing Lattice Kinetic Schemes for Fluid Dynamics with Lattice-Equivariant Neural Networks. *AIAA Journal*, 63(2):716–731, February 2025.

[189] Maria Strazzullo, Francesco Ballarin, Traian Iliescu, and Claudio Canuto. New feedback control and adaptive evolve-filter-relax regularization for the Navier-Stokes equations in the convection-dominated regime. *arXiv preprint arXiv:2307.00675*, 2023.

[190] Francesco De Vanna. Industrial CFD and Fluid Modeling in Engineering—2nd Edition. *Fluids*, 10:164, 06 2025.

[191] Benedikt Alkin, Tobias Kronlachner, Samuele Papa, Stefan Pirker, Thomas Lichtenegger, and Johannes Brandstetter. NeuralDEM for real time simulations of industrial particular flows. *Communications Physics*, 8, 11 2025.

[192] Liyao Gao, Yifan Du, Hongshan Li, and Guang Lin. RotEqNet: Rotation-equivariant network for fluid systems with symmetric high-order tensors. *Journal of Computational Physics*, 461:111205, 2022.

[193] Marius Kurz, Andrea Beck, and Benjamin Sanderse. Harnessing Equivariance: Modeling Turbulence with Graph Neural Networks, 2025.

[194] Claire Little, Mark Elliot, Richard Allmendinger, and Sahel Shariati Samani. Generative Adversarial Networks for Synthetic Data Generation: A Comparative Study, 2021.

[195] Giovanni Calzolari and Wei Liu. Accelerating Large Eddy Simulations of Urban Airflow with Generative Adversarial Networks. *Building and Environment*, 286:113622, 2025.

[196] Dedy Ariansyah, Mahmud Isnan, Reza Rahutomo, and Bens Pardamean. Digital Twin (DT) Smart City for Air Quality Management. *Procedia Computer Science*, 227:524–533, 2023. 8th International Conference on Computer Science and Computational Intelligence (ICCSCI 2023).

[197] Andrea Dadone and Bernard Grossman. Ghost-cell method for analysis of inviscid three-dimensional flows on Cartesian-grids. *Computers & Fluids*, 36(10):1513–1528, 2007. Special Issue Dedicated to Professor Michele Napolitano on the Occasion of his 60th Birthday.

[198] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[199] D Lilly. The representation of small-scale turbulence in numerical simulation experiments. *NCAR Report*, 1966.

[200] Yifei Guan, Adam Subel, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate LES. *Physica D: Nonlinear Phenomena*, 443:133568, 2023.

[201] Changhong Mou, Elia Merzari, Omer San, and Traian Iliescu. An energy-based lengthscale for reduced order models of turbulent flows. *Nuclear Engineering and Design*, 412:112454, 2023.

[202] John Lee. *Introduction to smooth manifolds. 2nd revised ed*, volume 218. Springer, New York, 01 2012.

Curriculum Vitae

Toby van Gastelen

17-02-1995 Born in Purmerend, The Netherlands

Education

2021-2026 PhD in Applied Mathematics
Centrum Wiskunde & Informatica and Eindhoven University of Technology, The Netherlands
Thesis: *Structure-preserving data-driven methods for modeling turbulent flows*
Promotor: Benjamin Sanderse

2018-2021 Master of Science in Computational Science
University of Amsterdam & Vrije Universiteit Amsterdam, Amsterdam

2014-2018 Bachelor of Science in Pharmaceutical Sciences
Vrije Universiteit Amsterdam, Amsterdam

List of Presentations

Oral presentations

- **ECCOMAS CONGRESS**, Oslo, Norway, 2022.
- **SIAM Conference on Computational Science and Engineering (CSE)**, Amsterdam, Netherlands, 2023
- **IACM Computational Fluids Conference**, Cannes, France, 2023
- **ECMI**, Wrocław, Poland, 2023
- **ERCOFTAC-14**, Barcelona, Spain, 2023
- **ECCOMAS CONGRESS**, Lisbon, Portugal, 2024
- **ROM+ML4LES+**, Atlanta, USA, 2024
- **ENUMATH**, Heidelberg, Germany, 2025

Poster presentations

- **Dutch-Flemish Scientific Computing Society**, Woudschoten, The Netherlands, 2022

Acknowledgements

First of all, I would like to thank my two supervisors, Wouter and Benjamin. You both have offered tremendous guidance since the start.

Benjamin, since my background was quite different from most people in the department, there was quite a steep learning curve for me. I can imagine this also posed a challenge for you. However, from the very beginning, you taught me a great deal about fluid modeling, writing, and mathematics in general. I feel really sorry for you for having to go through my earlier work, but to my surprise, you always saw value in it. Until the end of the process, you were actively involved and supportive, and I could always turn to you for advice. Especially during the writing process, your feedback was always detailed and essential to improving my work.

Wouter, my journey at CWI started under your guidance and supervision. Thank you for taking the time to take me under your wing by offering me an internship position and bringing me into the group. You were my first introduction to the field of climate modeling, and I learned a lot from you. During my PhD, you also took the time to read my work and bring a new perspective when Benjamin and I got caught up in the details.

Next, I would like to thank the rest of the scientific computing group at CWI, not only for the fruitful discussions but also for the good times during lunch, walks, sports activities, and conference trips. Syver, I would like to thank you for the amount of IT support you offered me during this project. There did not seem to be any IT- or math-related question you couldn't provide an answer to. Rik, thank you for all the fruitful discussions we had during the time you were my office mate and the runs we did together. I strive to have as much endurance as you. Henrik, thank you for the nice discussions and for helping me conceptualize my work, especially while working on my first paper. Also, thank you for the nice hikes during the conferences. Pardeep, thanks for always bringing in new perspectives, the nice temple visits for good food, and for all the great jokes, of course. Nikolaj, thank you for helping me out with structuring my thesis and always cooking up something tasty. Robin, thanks for giving me the idea for my second paper and the nice Latin parties. We still need to work on our moves though.

Besides people from the scientific computing group, I would like to thank Nada for always

being there to help with administrative work and for organizing workshops and group outings. I would also like to thank Minnie and Bikkie for the fun chats near the coffee machine. Moreover, I would like to express my gratitude to the reviewers of my publications, whose insightful comments greatly improved the quality of my articles.

Furthermore, I would like to thank my friend group, the Purmerender Pythons, for the fun trips and payday celebrations over the years. I would also like to thank the many people from my athletics club, Phanos, who always offered a welcome distraction from my PhD work, in particular my sprint team, Phast, under the supervision of head coach Daniel van Leeuwen and assistant trainer Timo Stam.

Lastly, I would like to thank my parents for always helping and supporting me throughout this journey.

7

Appendix

A Appendix Chapter 2

A.1 Filter properties

Here, we derive important properties of the spatial averaging filter. We first show that equation (2.27) holds:

$$(\mathbf{R}\bar{\mathbf{a}}, \mathbf{R}\bar{\mathbf{b}})_\omega = \bar{\mathbf{a}}^T\mathbf{R}^T\boldsymbol{\omega}\mathbf{R}\bar{\mathbf{b}} = \bar{\mathbf{a}}^T\boldsymbol{\Omega}\mathbf{W}\boldsymbol{\omega}^{-1}\boldsymbol{\omega}\mathbf{R}\bar{\mathbf{b}} = \bar{\mathbf{a}}^T\boldsymbol{\Omega}\underbrace{\mathbf{W}\mathbf{R}}_{=\mathbf{I}}\bar{\mathbf{b}} = (\bar{\mathbf{a}}, \bar{\mathbf{b}})_\Omega, \tag{7.1}$$

where we used the fact that

$$\boldsymbol{\Omega}\mathbf{W}\boldsymbol{\omega}^{-1} = \mathbf{R}^T. \tag{7.2}$$

Next, we proof that $\mathbf{R}\bar{\mathbf{u}}$ is orthogonal to $\mathbf{u}'$:

$$\begin{aligned}(\mathbf{R}\bar{\mathbf{u}}, \mathbf{u}')_\omega &= (\mathbf{R}\bar{\mathbf{u}}, \mathbf{u} - \mathbf{R}\bar{\mathbf{u}})_\omega = (\mathbf{R}\bar{\mathbf{u}}, (\mathbf{I} - \mathbf{R}\mathbf{W})\mathbf{u})_\omega = \bar{\mathbf{u}}^T\mathbf{R}^T\boldsymbol{\omega}(\mathbf{I} - \mathbf{R}\mathbf{W})\mathbf{u} \\ &= \bar{\mathbf{u}}^T\boldsymbol{\Omega}\mathbf{W}(\mathbf{I} - \mathbf{R}\mathbf{W})\mathbf{u} = (\bar{\mathbf{u}}, (\mathbf{W} - \underbrace{\mathbf{W}\mathbf{R}}_{=\mathbf{I}}\mathbf{W})\mathbf{u})_\Omega = (\bar{\mathbf{u}}, (\mathbf{W} - \mathbf{W})\mathbf{u})_\Omega = 0.\end{aligned} \tag{7.3}$$

Finally, we show that equation (2.31) holds:

$$(\mathbf{1}_\omega, \mathbf{u})_\omega = \mathbf{1}_\omega^T \boldsymbol{\omega} \mathbf{u} = \mathbf{1}_\Omega^T \mathbf{R}^T \boldsymbol{\omega} \mathbf{u} = \mathbf{1}_\Omega^T \boldsymbol{\Omega} \mathbf{W} \boldsymbol{\omega}^{-1} \boldsymbol{\omega} \mathbf{u} = \mathbf{1}_\Omega^T \boldsymbol{\Omega} \mathbf{W} \mathbf{u} = (\mathbf{1}_\Omega, \mathbf{W}\mathbf{u})_\Omega = (\mathbf{1}_\Omega, \bar{\mathbf{u}})_\Omega, \tag{7.4}$$

where we used the fact that $\mathbf{1}_\omega = \mathbf{R}\mathbf{1}_\Omega$.

A.2 Comparing coarse and fine-grid dissipation

Here we compare the rate of dissipation induced by the 1D diffusion operator discretized on the coarse grid, $\bar{\mathbf{D}} \in \mathbb{R}^{I \times I}$ (for grid-spacing H), with the dissipation induced by the same operator but discretized on the fine grid, $\mathbf{D} \in \mathbb{R}^{N \times N}$ (for grid-spacing $h = \frac{H}{J}$). The difference in dissipated energy between these two quantities Δ_D is given by:

$$\begin{aligned}\Delta_D &= (\mathbf{u}, \mathbf{D}\mathbf{u})_\omega - (\bar{\mathbf{u}}, \bar{\mathbf{D}}\bar{\mathbf{u}})_\Omega = \mathbf{u}^T \boldsymbol{\omega} \mathbf{D} \mathbf{u} - (\mathbf{W}\mathbf{u})^T \boldsymbol{\Omega} \bar{\mathbf{D}} \mathbf{W} \mathbf{u} = \mathbf{u}^T \frac{H}{J} \mathbf{D} \mathbf{u} - \mathbf{u}^T H \mathbf{W}^T \bar{\mathbf{D}} \mathbf{W} \mathbf{u} \\ &= \frac{J}{H} \mathbf{u}^T \left(\frac{H^2}{J^2} \mathbf{D} - \frac{H^2}{J} \mathbf{W}^T \bar{\mathbf{D}} \mathbf{W}\right) \mathbf{u} = \frac{J}{H} \mathbf{u}^T \underbrace{\left(h^2 \mathbf{D} - \frac{1}{J} \mathbf{W}^T (H^2 \bar{\mathbf{D}}) \mathbf{W}\right)}_{=: \mathbf{D}_\Delta} \mathbf{u},\end{aligned} \tag{7.5}$$

for periodic boundary conditions (BCs). Here we have written $\mathbf{D}^\Delta$ such that it is independent of the grid-spacing, but only depends on the ratio $J = \frac{H}{h}$. Note that $\mathbf{D}^\Delta$ is a symmetric matrix and as such its eigenvalues $\lambda_N^\Delta \leq \ldots \leq \lambda_1^\Delta$ are real and its eigenvectors can be chosen to form an orthogonal basis. Furthermore, $\lambda_1^\Delta = 0$ for periodic boundary conditions. We can bound Δ_D by noting that

$$\Delta_D \leq \max_i \frac{J}{H} \lambda_i^\Delta ||\mathbf{u}||_2^2. \tag{7.6}$$

If the eigenvalues are all strictly nonpositive, it follows that the difference in dissipation Δ_D is always less than or equal to zero. In other words, $\mathbf{D}$ extracts more (or equal) energy from the reference system as $\bar{\mathbf{D}}$ does from the filtered system. To prove that the eigenvalues of $\mathbf{D}_\Delta$ are indeed nonpositive turns out to be a difficult problem, which we circumvent with a numerical 'proof'. In Figure 7.1 we display the largest non-zero eigenvalue λ_2^Δ for different values of I and J, indicating that $\lambda_i^\Delta \leq 0$ for realistic values of I and J.

A.3 SGS compression

In this section, we outline how we obtain the SGS compression parameter values $\mathbf{t} \in \mathbb{R}^J$ such that $\mathbf{s} \approx \mathbf{W}(\mathbf{u}')^2$. This can be achieved by using a singular value decomposition (SVD). The

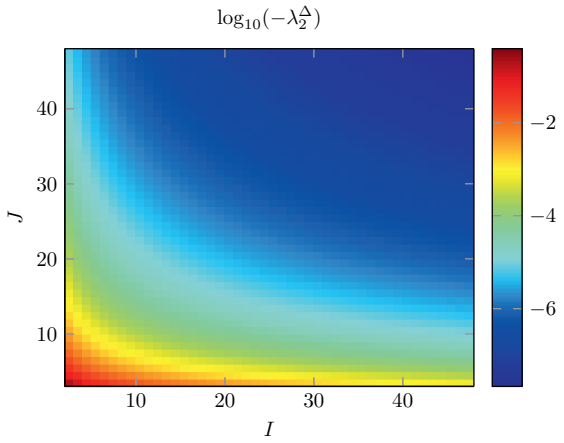


Figure 7.1: Largest non-zero eigenvalue λ_2^Δ of $\mathbf{D}_\Delta$ for different values of I and J.

SVD minimizes the following loss function for $\hat{\mathbf{t}} \in \mathbb{R}^J$ (here we assume a uniform grid):

$$\begin{aligned}\mathcal{L}_s(\mathbf{X}_\mu;\boldsymbol{\Theta}) &= \frac{1}{pI}\sum_{\boldsymbol{\mu}\in\mathbf{X}_\mu} |J^{-1}\boldsymbol{\mu}^T\boldsymbol{\mu} - \boldsymbol{\mu}^T\mathbf{t}\mathbf{t}^T\boldsymbol{\mu}| = \frac{1}{pIJ}\sum_{\boldsymbol{\mu}\in\mathbf{X}_\mu} |\boldsymbol{\mu}^T\boldsymbol{\mu} - \boldsymbol{\mu}^T\hat{\mathbf{t}}\hat{\mathbf{t}}^T\boldsymbol{\mu}| \\ &= \frac{1}{pIJ}\sum_{\boldsymbol{\mu}\in\mathbf{X}_\mu} |\boldsymbol{\mu}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\hat{\mathbf{t}}\hat{\mathbf{t}}^T\hat{\mathbf{t}}\hat{\mathbf{t}}^T\boldsymbol{\mu} - 2\boldsymbol{\mu}^T\hat{\mathbf{t}}\hat{\mathbf{t}}^T\boldsymbol{\mu}| \\ &= \frac{1}{pIJ}\sum_{\boldsymbol{\mu}\in\mathbf{X}_\mu} (\hat{\mathbf{t}}\hat{\mathbf{t}}^T\boldsymbol{\mu} - \boldsymbol{\mu})^T(\hat{\mathbf{t}}\hat{\mathbf{t}}^T\boldsymbol{\mu} - \boldsymbol{\mu}) \\ &\text{subject to} \quad \hat{\mathbf{t}}^T\hat{\mathbf{t}} = 1,\end{aligned} \tag{7.7}$$

where $\boldsymbol{\mu}$ refers to a column vector of snapshot matrix $\mathbf{X}_\mu \in \mathbb{R}^{J\times Ip}$ and $\mathbf{t} = J^{-1/2}\hat{\mathbf{t}}$. The last expression for $\mathcal{L}_s$ is a projection error. Such an error is typically minimized in a reduced-order modeling setting to obtain a proper basis, see [67]. For this one typically uses an SVD. Conveniently, minimizing this error is equivalent to minimizing our required energy error (first expression). In this expression the prefactor J^{-1} accounts for $\mathbf{W}$ in the SGS energy (2.38). The snapshot matrix $\mathbf{X}_\mu$ is constructed from the training data set $\mathbf{X}_\mathrm{u} \in \mathbb{R}^{N\times p}$, which contains p DNS snapshots as the columns. From this matrix we compute the SGS content: $\mathbf{X}_{\mathrm{u}'} = (\mathbf{I} - \mathbf{R}\mathbf{W})\mathbf{X}_\mathrm{u}$. Finally, this matrix is reshaped into $\mathbf{X}_\mu$ such that each column corresponds to the SGS content in a coarse cell, i.e. $\mathbf{u}' \to [\boldsymbol{\mu}_i, \ldots, \boldsymbol{\mu}_I]$ for each column in $\mathbf{X}_{\mathrm{u}'}$. The SVD of $\mathbf{X}_\mu$ is given by

$$\mathbf{X}_\mu = \mathbf{U}_\mu\boldsymbol{\Sigma}_\mu\mathbf{V}_\mu^T, \tag{7.8}$$

where $\mathbf{U}_\mu \in \mathbb{R}^{J\times J}$ and $\mathbf{V}_\mu \in \mathbb{R}^{Ip\times Ip}$ are unitary matrices containing the left and right-singular vectors, respectively, and $\mathbf{\Sigma}_\mu \in \mathbb{R}^{J\times Ip}$ contains the singular values on the diagonal. The values for $\hat{\mathbf{t}}$ that minimize (7.7) correspond to the first column of $\mathbf{U}_\mu$.

A.3.1 Non-periodic boundary conditions

To extend our method to different types of BCs we resort to what the machine learning community refers to as padding [44] and the scientific computing community refers to as the ghost-cell method [197]. We will treat both inflow and outflow BCs, on uniform 1D grids, as this is relevant for Burgers' equation.

Implementation for the fine grid

The ghost-cell method enhances the discretization with ghost cells beyond the domain boundary $\partial\Omega$ (with domain $\Omega = [a, b]$), as displayed in Figure 7.2. Here we present the implementation for the fine grid.

The inflow (Dirichlet) BC is given by $u(x = a, t) = \alpha(t)$. Based on this, we compute ghost value u_0 as

$$u(x = a, t) = \alpha(t) = \frac{\mathrm{u}_1 + \mathrm{u}_0}{2} \quad \rightarrow \quad \mathrm{u}_0 = 2\alpha(t) - \mathrm{u}_1. \tag{7.9}$$

This corresponds to the first ghost cell outside the left boundary, see Figure 7.2. For the outflow BC we use a symmetric BC at the right boundary, given by $\frac{\partial u}{\partial x}|_{x=b,t} = 0$. This is implemented by taking $\mathrm{u}_{N+1} = \mathrm{u}_N$, where u_{N+1} corresponds to the first ghost cell outside the right boundary, see Figure 7.2.

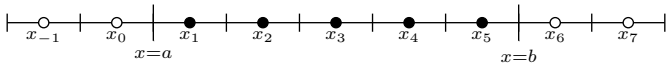


Figure 7.2: 1D grid enhanced with ghost cells beyond the domain boundaries, indicated by the hollow circles.

Implementation for the filtered system

Our structure-preserving closure modeling framework is effectively a nonlinear stencil, due to the presence of a convolutional neural network. It therefore takes information from k neighboring grid cells on each side. This means we require k ghost cells on either side of the domain boundary $\partial\Omega$. To find appropriate choices for the ghost values $\bar{\mathrm{u}}_i$ and s_i $(i = -k + 1, \ldots, 0, I + 1, \ldots, I + k)$ we consider the fine-grid solution $\mathbf{u}$ and appropriately extend this past the domain boundary, see Figure 7.3.

Inflow BC

For the left inflow BC we extend (7.9) to

$$\mathrm{u}_{-i+1} = 2\alpha(t) - \mathrm{u}_i, \qquad i = 1, 2, \dots \tag{7.10}$$

We can rewrite this as a function of $\bar{\mathrm{u}}_i$ and SGS content $\boldsymbol{\mu}_i$:

$$(\bar{\mathrm{u}}_{-i+1} + \mu_{-i+1,J-j}) = 2\alpha(t) - (\bar{\mathrm{u}}_i + \mu_{i,1+j}), \qquad 1 \le i \le k, \quad 0 \le j \le J-1. \tag{7.11}$$

This can be split into a filtered part:

$$\bar{\mathrm{u}}_{-i+1} = 2\alpha(t) - \bar{\mathrm{u}}_i, \qquad 1 \le i \le k, \tag{7.12}$$

which yields the ghost values for $\bar{\mathbf{u}}$ past the left boundary, and a SGS part:

$$\mu_{-i+1,J-j} = -\mu_{i,1+j}, \qquad 1 \le i \le k, \quad 0 \le j \le J-1. \tag{7.13}$$

The latter can be simplified as

$$\boldsymbol{\mu}_{-i+1} = -\mathbf{P}\boldsymbol{\mu}_i, \qquad 1 \le i \le k, \tag{7.14}$$

where $\mathbf{P} \in \mathbb{R}^{J \times J}$ is the permutation matrix that represents the reflection across the boundary.

Outflow BC

For the symmetric outflow BC we extend the fine-grid solution past the domain as

$$\mathrm{u}_{N+i} = \mathrm{u}_{N-i+1}, \qquad i = 1, 2, \dots \tag{7.15}$$

In terms of $\bar{\mathrm{u}}_i$ and $\boldsymbol{\mu}_i$ this becomes

$$\bar{\mathrm{u}}_{I+i} + \mu_{I+i,1+j} = \bar{\mathrm{u}}_{I-i+1} + \mu_{I-i+1,J-j}, \qquad 1 \le i \le k, \quad 0 \le j \le J-1, \tag{7.16}$$

which can again be split into an equation for the ghost values for $\bar{\mathbf{u}}$:

$$\bar{\mathrm{u}}_{I+i} = \bar{\mathrm{u}}_{I-i+1}, \qquad 1 \le i \le k, \tag{7.17}$$

and a SGS part

$$\boldsymbol{\mu}_{I+i} = \mathbf{P}\boldsymbol{\mu}_{I-i+1}, \qquad 1 \le i \le k. \tag{7.18}$$

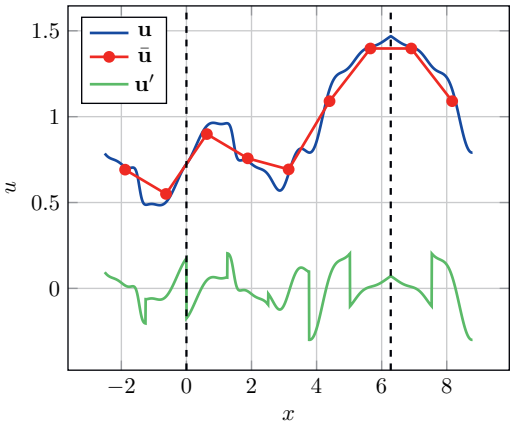


Figure 7.3: An example solution $\mathbf{u}$ with $N = 1000$ filtered onto a coarse grid with $I = 5$ extended past $\partial\Omega$, according to (7.10) ($\alpha = \frac{7}{10}$) for the left boundary and (7.15) for the right boundary. $\partial\Omega$ is indicated by the dashed vertical lines. $\bar{\mathbf{u}}$ is extended past $\partial\Omega$ according to (7.12) and (7.17) and $\mathbf{u}'$ is extended according to (7.14) and (7.18).

BCs for the SGS variables

After extending the solution using the ghost-cell method, we aim to find the value for $\mathbf{s}$ beyond the boundary. In order to find these ghost values, we make use of the fact that we can interpret the operation $\mathbf{t}^T\mathbf{t}$ as the back and forth projection from a reduced basis to the physical SGS basis and back. This means that

$$\boldsymbol{\mu}_i \approx \sqrt{J}\mathbf{t}\mathrm{s}_i$$

and

$$\mathrm{s}_i = J\mathbf{t}^T\mathbf{t}\mathrm{s}_i.$$

The idea is to project s_i back onto physical SGS space, apply the $\mathbf{P}$ operator, and then project back to find the appropriate ghost values. Using this idea we obtain the following relations for the left and right boundary, respectively:

$$\boldsymbol{\mu}_{-i+1} = -\mathbf{P}\boldsymbol{\mu}_i \quad \rightarrow \quad \mathrm{s}_{-i+1} = -J\mathbf{t}^T\mathbf{P}\mathbf{t}\mathrm{s}_i, \qquad 1 \leq i \leq k, \tag{7.19}$$

$$\boldsymbol{\mu}_{I+i} = \mathbf{P}\boldsymbol{\mu}_{I-i+1} \quad \rightarrow \quad \mathrm{s}_{I+i} = J\mathbf{t}^T\mathbf{P}\mathbf{t}\mathrm{s}_{I-i+1}, \qquad 1 \leq i \leq k. \tag{7.20}$$

Note that $\mathbf{t}^T\mathbf{P}\mathbf{t}$ is a scalar which only needs to be computed once.

A simulation for this input/output (I/O) BCs implementation is shown in Figure 7.4. Here

we simulate Burgers' equation with I/O BCs, and some additional forcing, see Appendix A.3.2. We depict the solution produced by our structure-preserving closure modeling framework and compare it to the DNS.

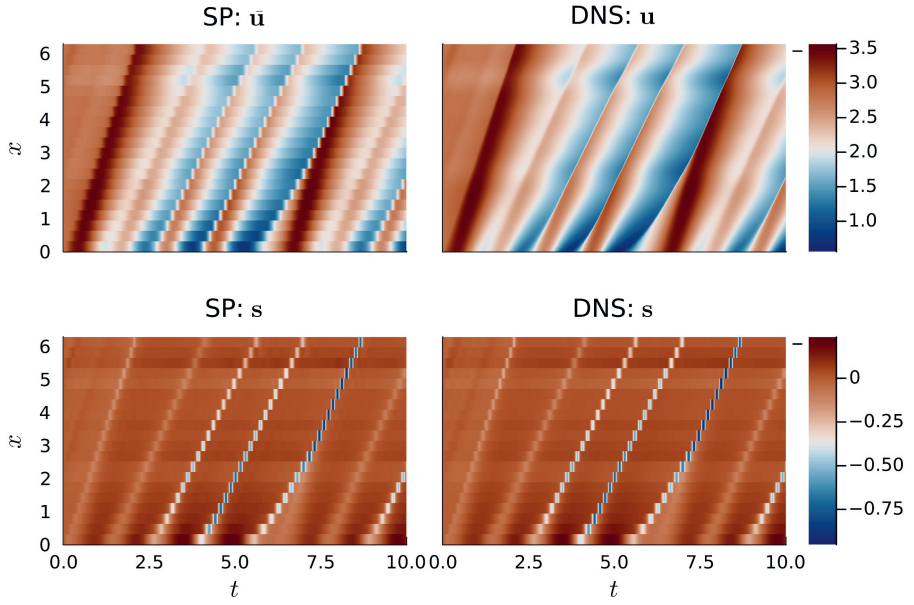


Figure 7.4: A simulation of Burgers' equation with I/O BCs and forcing using our trained structure-preserving closure model for DOF = 40 (left), along with the DNS solution for $N = 1000$ (right).

A.3.2 Training procedure

In order to train our machine learning-based closure models we first require DNSs to generate reference data. This reference data serves as a target for our machine learning models to reproduce. In this section we describe how we randomly generate simulation conditions (initial conditions, BCs, and forcing) for closure model training and testing. In addition, we describe the training procedure and the chosen hyperparameter values, obtained from the hyperparameter tuning procedure.

Generating training data

To generate initial conditions, forcing, and unsteady Dirichlet BCs we make use of the

following parameterized Fourier decomposition (with parameters $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}$):

$$\xi(y; \alpha_1, \alpha_2, \alpha_3) = \alpha_1 + \frac{\alpha_2}{\sqrt{M}} \sum_{i=2}^{M} C_{i1} \sin\left(i \frac{2\pi}{\alpha_3} y\right) + C_{i2} \cos\left(i \frac{2\pi}{\alpha_3} y\right), \tag{7.21}$$

where M is uniformly sampled from $\{2, 3, \ldots, 8\}$ and $C_{ij} \sim p$ from

$$p(y) = \begin{cases} 1, & \text{for } \frac{1}{2} \leq |y| \leq 1, \\ 0, & \text{elsewhere.} \end{cases} \tag{7.22}$$

In the case of Burgers' equation, we carry out 100 reference simulations on a uniform grid with $N = 1000$ on the domain $\Omega = [0, 2\pi]$ for $\nu = 0.01$. To march the solution forward in time, we employ an Runge-Kutta 4 (RK4) scheme with a time step size of $\Delta t = 2.5 \times 10^{-3}$ and simulate up to $T = 10$ [96]. 50 simulations are carried out using periodic BCs and 50 with I/O BCs. For the periodic case, the initial condition is given by $u(x, t = 0) = \xi(x; 2, 1, |\Omega|)$. For the I/O case, the inflow condition is given by $u(0, t) = \xi(t, 2, 1, 2\pi)$ and the outflow condition by a symmetric BC on the right side of the domain. The implementation of the BCs is described in Appendix A.3.1. The initial condition is given by a constant-valued function, equal to the inflow condition at $t = 0$. In addition, we also add a steady forcing term $F(x) = \xi(x; 0, \frac{1}{2}, |\Omega|)$ to the right-hand side (RHS) of (2.5) for the I/O case.

With regards to the Korteweg de-Vries (KdV) equation, we employ a uniform grid with $N = 600$ on the domain $\Omega = [0, 32]$ for $\varepsilon = 6$ and $\mu = 1$. The solution is marched forward in time using an RK4 scheme with a time step size of $\Delta t = 10^{-4}$, up to $T = 10$. In this case we only consider periodic BCs, with the initial condition given by $u(x, 0) = \xi(x; 0, \frac{3}{5}, |\Omega|)$, and perform 100 reference simulations.

For both Burgers' and KdV, reference data is saved at each time interval of 5×10^{-3}. We randomly sample 10% of the data from these datasets to generate the two datasets used for training (one for each equation). Both of these are split into a training (70%) and a validation set (30%). For testing purposes, the unseen simulation conditions are generated in a similar manner, but with different randomly sampled M and C_{ij}.

Hyperparameters and tuning

The chosen hyperparameters for our structure-preserving neural network architecture (SP) closure and the vanilla CNN are displayed in Table 7.1. The weights and biases are initiated using the Glorot normal initialization algorithm [198]. They are optimized using the Adam optimization algorithm with parameters α (learning-rate), β_1 (decay rate for the first momentum estimates), β_2 (decay rate for the second momentum estimates), ϵ (small constant to combat numerical instability) [41]. Hyperparameters are selected based on how

well the trained closure models reproduce the RHS for the solution snapshots present in the validation set, corresponding to DOF = 60. For this purpose, models are trained without trajectory fitting. For the hyperparameter optimization, we opt to vary the number of hidden layers, for which we consider $\{0, 1, 2\}$, and the number of channels per hidden layer, for which we consider $\{10, 20, 30\}$. The performance of each of the trained closure models is shown in figure 7.5. The best performing combination of hyperparameters (displayed in Table 7.1), for

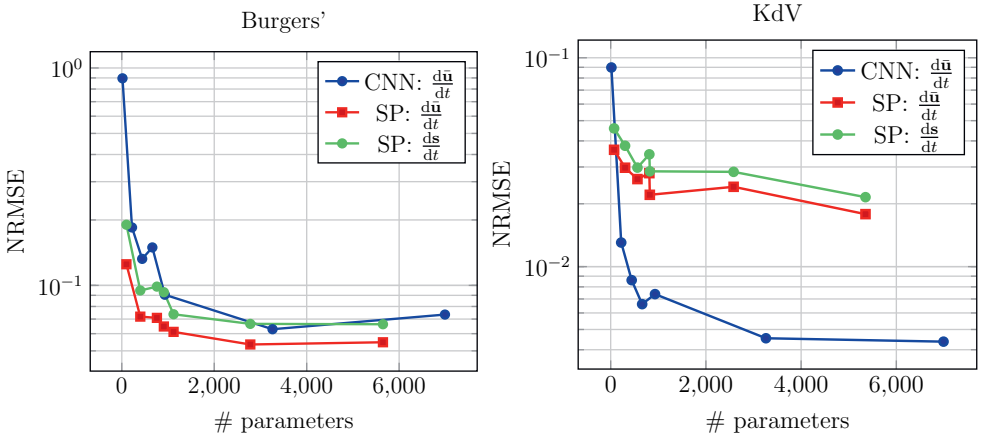


Figure 7.5: NRMSE for reproducing the RHS for each of the considered hyperparameter configurations for Burgers' (left) and KdV (right) averaged over the validation set for DOF = 60.

each equation, is selected to train the final closure models. This time, trajectory fitting is included.

Table 7.1: Hyperparameters for the trained closure models

Hyperparameter	**CNN**	**SP**
α	10^{-3}	10^{-3}
β_1	0.9	0.9
β_2	0.999	0.999
ϵ	10^{-8}	10^{-8}
Mini-batch size	20	20
# iterations derivative fitting	100	100
# iterations trajectory fitting	20	20
Trajectory fitting $\overline{\Delta t}$ (Burgers')	0.01	0.01
Trajectory fitting $\overline{\Delta t}$ (KdV)	5×10^{-3}	5×10^{-3}
Trajectory fitting # time steps (Burgers')	5	5
Trajectory fitting # time steps (KdV)	20	20
Nonlinear activation function (underlying CNN)	ReLU	ReLU
Final activation function (underlying CNN)	linear	linear
Kernel size	7	5
Stride	1	1
# hidden layers	2	2
# channels per hidden layer (Burgers')	20	20
# channels per hidden layer (KdV)	20	30
Total # parameters (Burgers')	3261	2780
Total # parameters (KdV)	3261	5352
B (Burgers')	-	1
B (KdV)	-	2

B Appendix Chapter 3

B.1 Stucture-preserving finite volume discretization

For the employed finite difference discretization, presented in [25], the physical structure of the Navier-Stokes equations is preserved in a discrete sense. Discretely, the total momentum and energy are approximated as

$$\mathbf{P}_h = \mathbb{1}_h \mathbf{\Omega}_h \mathbf{u}_h, \tag{7.23}$$

$$E_h = \frac{1}{2} \mathbf{u}_h^T \mathbf{\Omega}_h \mathbf{u}_h. \tag{7.24}$$

The change in momentum for this discretization is given by

$$\begin{aligned} \frac{\mathrm{d}\mathbf{P}_h}{\mathrm{d}t} &= \mathbb{1}_h \mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} \\ &= \mathbb{1}_h (-\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h - \mathbf{G}_h \mathbf{p}_h + \nu \mathbf{D}_h \mathbf{u}_h + \mathbf{\Omega}_h \mathbf{f}_h) \\ &= \mathbb{1}_h \mathbf{\Omega}_h \mathbf{f}_h, \end{aligned} \tag{7.25}$$

as the discrete operators are carefully constructed such that the column vectors sum up to zero. This means momentum conservation is satisfied by this discretization. Using the product rule, we obtain the change in energy as

$$\begin{aligned} \frac{\mathrm{d}E_h}{\mathrm{d}t} &= \mathbf{u}_h^T \mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} \\ &= \mathbf{u}_h^T (-\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h - \mathbf{G}_h \mathbf{p}_h + \nu \mathbf{D}_h \mathbf{u}_h + \mathbf{\Omega}_h \mathbf{f}_h) \\ &= -\mathbf{u}_h^T \mathbf{Q}_h^T \mathbf{Q}_h \mathbf{u}_h + \mathbf{u}_h^T \mathbf{\Omega}_h \mathbf{f}_h = -||\mathbf{Q}_h \mathbf{u}_h||_2^2 + \mathbf{u}_h^T \mathbf{\Omega}_h \mathbf{f}_h, \end{aligned} \tag{7.26}$$

where we used the fact that the diffusion operator $\mathbf{D}_h$ can be Cholesky decomposed as $-\mathbf{Q}_h^T \mathbf{Q}_h$ [24, 67]. The convective contribution disappears due to the skew-symmetry of the discrete operator:

$$\mathbf{C}_h(\mathbf{u}_h) = -\mathbf{C}_h^T(\mathbf{u}_h) \quad \rightarrow \quad \mathbf{u}_h^T \mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h = -\mathbf{u}_h^T \mathbf{C}_h^T(\mathbf{u}_h)\mathbf{u}_h = 0, \tag{7.27}$$

where we used the symmetry of the inner product. Note that the skew-symmetry of the convection operator is only true for a divergence-free $\mathbf{u}_h$ [40]. The pressure term disappears because the discretization satisfies $\mathbf{G}_h = -\mathbf{M}_h^T$ [67]. Writing the energy contribution we obtain:

$$-\mathbf{u}_h^T \mathbf{G}_h \mathbf{p}_h = \mathbf{u}_h^T \mathbf{M}_h^T \mathbf{p}_h = \mathbf{p}_h^T \mathbf{M}_h \mathbf{u}_h = 0, \tag{7.28}$$

where we used the divergence-free constraint on the velocity field.

B.2 Pressure projection

The divergence-freeness can also be written as a projection of the partial differential equation (PDE) discretization (3.5) on a divergence-free basis [40]. To see this, we compute the divergence of the non-pressure related terms in (3.5) as

$$\mathbf{M}_h\mathbf{\Omega}_h^{-1}(-\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h + \nu\mathbf{D}_h\mathbf{u}_h + \mathbf{\Omega}_h\mathbf{f}_h) = \mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h), \tag{7.29}$$

where we grouped the different terms into $\mathbf{m}_h(\mathbf{u}_h)$. The purpose of the gradient of the pressure is to remove this divergence from the RHS of (3.5). To obtain the pressure that achieves this, we solve the following linear system:

$$\begin{aligned}
&\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h) - \mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h\mathbf{p}_h = \mathbf{0} \quad \rightarrow \\
&\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h) = \mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h\mathbf{p}_h \quad \rightarrow \\
&\mathbf{p}_h = (\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h)^{-1}\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h).
\end{aligned} \tag{7.30}$$

By filling this in to (3.5) we obtain

$$\begin{aligned}
\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} &= \mathbf{m}_h(\mathbf{u}_h) - \mathbf{G}_h(\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h)^{-1}\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h) \quad \rightarrow \\
\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} &= \underbrace{(\mathbf{I} - \mathbf{G}_h(\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h)^{-1}\mathbf{M}_h\mathbf{\Omega}_h^{-1})}_{:=\mathcal{P}_h}\mathbf{m}_h(\mathbf{u}_h) \quad \rightarrow \\
\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} &= \mathcal{P}_h\mathbf{m}_h(\mathbf{u}_h),
\end{aligned} \tag{7.31}$$

where $\mathcal{P}_h \in \mathbb{R}^{2N\times 2N}$ projects $\mathbf{m}_h(\mathbf{u}_h)$ onto a divergence-free basis. This transforms the discretized PDE into a single equation. Note that in practice, we typically solve for the pressure rather than performing the projection in this manner. However, this formulation is more convenient for closure modeling.

B.3 Motivation behind skew-symmetric architecture

To motivate the proposed neural network architecture we consider a simple 1D system for which the equation is unknown. The neural network is tasked with predicting the evolution of this system. The system consists of some quantity $w(x,t)$ which evolves on a periodic domain $x \in \Omega$. We know it satisfies the following conservation laws:

$$\int_\Omega \frac{\mathrm{d}w}{\mathrm{d}t}\mathrm{d}x = 0, \tag{7.32}$$

$$\int_\Omega w\frac{\mathrm{d}w}{\mathrm{d}t}\mathrm{d}x = 0, \tag{7.33}$$

which resemble momentum and energy conservation in the Navier-Stokes equations. $w(x,t)$ is discretized on a finite volume grid, such that $w(x_i,t) \approx \mathrm{w}_i(t)$, where x_i is the center of finite volume cell Ω_i. The grid contains N grid cells such that the discrete solution is described by the state vector $\mathbf{w}(t) \in \mathbb{R}^N$. We consider the case in which the evolution equation for $w(x,t)$ is unknown; however, we do have data for $w(x,t)$ at different points in space and time. The challenge is finding the RHS for $\frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t}$, with (7.32) and (7.33) being satisfied discretely, i.e.

$$\mathbf{1}_h^T \mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t} = 0, \tag{7.34}$$

$$\mathbf{w}_h^T \mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t} = 0, \tag{7.35}$$

are satisfied. As an ansatz for the RHS we use our proposed skew-symmetric neural network architecture

$$\mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t} \approx \underbrace{(\mathbf{\Delta}_c \mathrm{diag}(\mathbf{k}) \mathbf{\Delta}_f - \mathbf{\Delta}_f^T \mathrm{diag}(\mathbf{k}) \mathbf{\Delta}_c^T)}_{=:\mathcal{Y}} \mathbf{w}_h, \tag{7.36}$$

where $\mathbf{\Delta}_c, \mathbf{\Delta}_f \in \mathbb{R}^{N \times N}$ are central and forward difference stencils, respectively, such that $(\mathbf{\Delta}_c \mathbf{w}_h)_i = \mathrm{w}_{h,i+1} - \mathrm{w}_{h,i-1}$ and $(\mathbf{\Delta}_f \mathbf{w}_h)_i = \mathrm{w}_{h,i+1} - \mathrm{w}_{h,i}$. Note that these are specific choices for $\mathcal{B}_1$ and $\mathcal{B}_2$. During testing, we found that predefining the convolutions in $\mathcal{B}_1$ and $\mathcal{B}_2$ resulted in poor performance of the closure model. However, we choose to do so here to ease the analysis. Note that (7.34) is satisfied due to $\mathbf{\Delta}_c$ and $\mathbf{\Delta}_f$ being in the nullspace of $\mathbf{1}_h$ and (7.35) is satisfied due to the skew-symmetry of $\mathcal{Y}$. We are free to choose $\mathbf{k} \in \mathbb{R}^N$ without violating the conservation laws. In our case, it is represented by the output of a CNN, such

that $\mathbf{k} = \mathbf{k}(\mathbf{w}_h, \theta) \in \mathbb{R}^N$. Writing out the matrix-vector product in (7.36) gives us

$$
\begin{aligned}
\mathcal{Y}\mathbf{w}_h &= \begin{bmatrix}
 & & \ddots & & & & \\
 & & \ddots & \mathrm{k}_{i-1} & & & \\
\ddots & \ddots & \ddots & -\mathrm{k}_{i-1} - \mathrm{k}_i & \ddots & & \\
 & -\mathrm{k}_{i-1} & \mathrm{k}_i + \mathrm{k}_{i-1} & 0 & -\mathrm{k}_i - \mathrm{k}_{i+1} & \mathrm{k}_{i+1} & \\
 & & \ddots & \mathrm{k}_{i+1} + \mathrm{k}_i & \ddots & \ddots & \ddots \\
 & & & -\mathrm{k}_{i+1} & \ddots & & \\
 & & & & \ddots & &
\end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ \mathrm{w}_{h,i} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \\
&= \begin{bmatrix}
 & \ddots & & & & \\
\ddots & \ddots & \mathrm{w}_{h,i+1} - \mathrm{w}_{h,i} & & & \\
 & \mathrm{w}_{h,i-1} - \mathrm{w}_{h,i-2} & \mathrm{w}_{h,i-1} - \mathrm{w}_{h,i+1} & \mathrm{w}_{h,i+2} - \mathrm{w}_{h,i+1} & & \\
 & & \mathrm{w}_{h,i} - \mathrm{w}_{h,i-1} & \ddots & \ddots & \\
 & & & \ddots & &
\end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ \mathrm{k}_i \\ \vdots \\ \vdots \end{bmatrix},
\end{aligned}
$$

where it is clear that for both matrices the columns lie in the nullspace of $\mathbf{1}_h$. This gives us two perspectives: the first is that of a highly parameterized skew-symmetric matrix multiplied by the solution vector $\mathbf{w}_h$. The second is that of the parameterized vector $\mathbf{k}(\mathbf{w}_h, \theta)$ being multiplied by a matrix for which each column lies in the nullspace of $\mathbf{w}_h$. Therefore, by training $\mathbf{k}(\mathbf{w}_h, \theta)$ on reference data we effectively learn a highly parameterized, and non-linear, momentum and energy conserving discretization. This formulation allows the neural network to freely explore the energy-conserving vector space. By choosing $k_i = 1/(2h)$ and $k_{i-1} = k_{i+1} = 0$, where h is the grid spacing, we simply obtain a central difference approximation of the first derivative, which corresponds to linear advection. The skew-symmetric term can therefore be viewed as highly parameterized advection.

For dissipative systems, i.e. (7.33) is negative, we require an additional term in our parameterized discretization, as introduced in Section 3.4.3.2. This extends (7.36) to

$$
\mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t} \approx \mathcal{Y}\mathbf{w}_h \underbrace{-\mathbf{\Delta}_f^T \mathrm{diag}(\mathbf{q})^2 \mathbf{\Delta}_f}_{=:\mathcal{Z}} \mathbf{w}_h, \tag{7.37}
$$

where $\mathbf{q} = \mathbf{q}(\mathbf{w}_h, \theta) \in \mathbb{R}^N$ is also an output of the CNN. In this case we choose $\mathcal{B}_3 = \mathbf{\Delta}_f$. This specific choice is once again made to ease the analysis, whereas in the actual neural

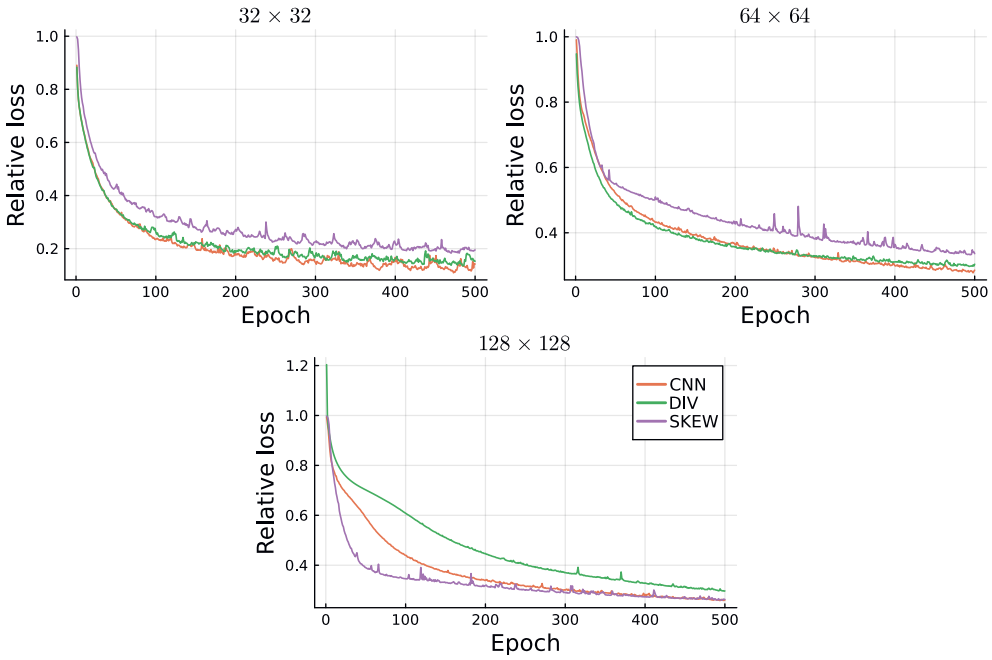


Figure 7.6: Relative loss, see (3.20), with respect to NC for each of the different closure models and coarse-graining factors.

network $\mathcal{B}_3$ is a parameterized operator. Writing out the corresponding stencil gives:

$$(\mathcal{Z}\mathbf{w}_h)_i = \mathrm{q}_i^2(\mathbf{w}_h, \theta)(w_{h,i+1} - w_{h,i}) + \mathrm{q}_{i-1}^2(\mathbf{w}_h, \theta)(w_{h,i-1} - w_{h,i}).$$

By choosing $\mathrm{q}_i = 1/h$, we obtain a second-order approximation of the second derivative. Our dissipative term can therefore be thought of as highly parameterized and non-linear diffusion.

B.4 Training of the neural networks

In Figure 7.6 we depict the loss of the different closure model architectures with respect to no closure (NC). The choice of hyperparameters is discussed in Section 3.5.1.

B.5 Vorticity fields

In Figure 7.7 and Figure 7.8, we display the vorticity fields of the decaying turbulence simulation, using the trained closure models.

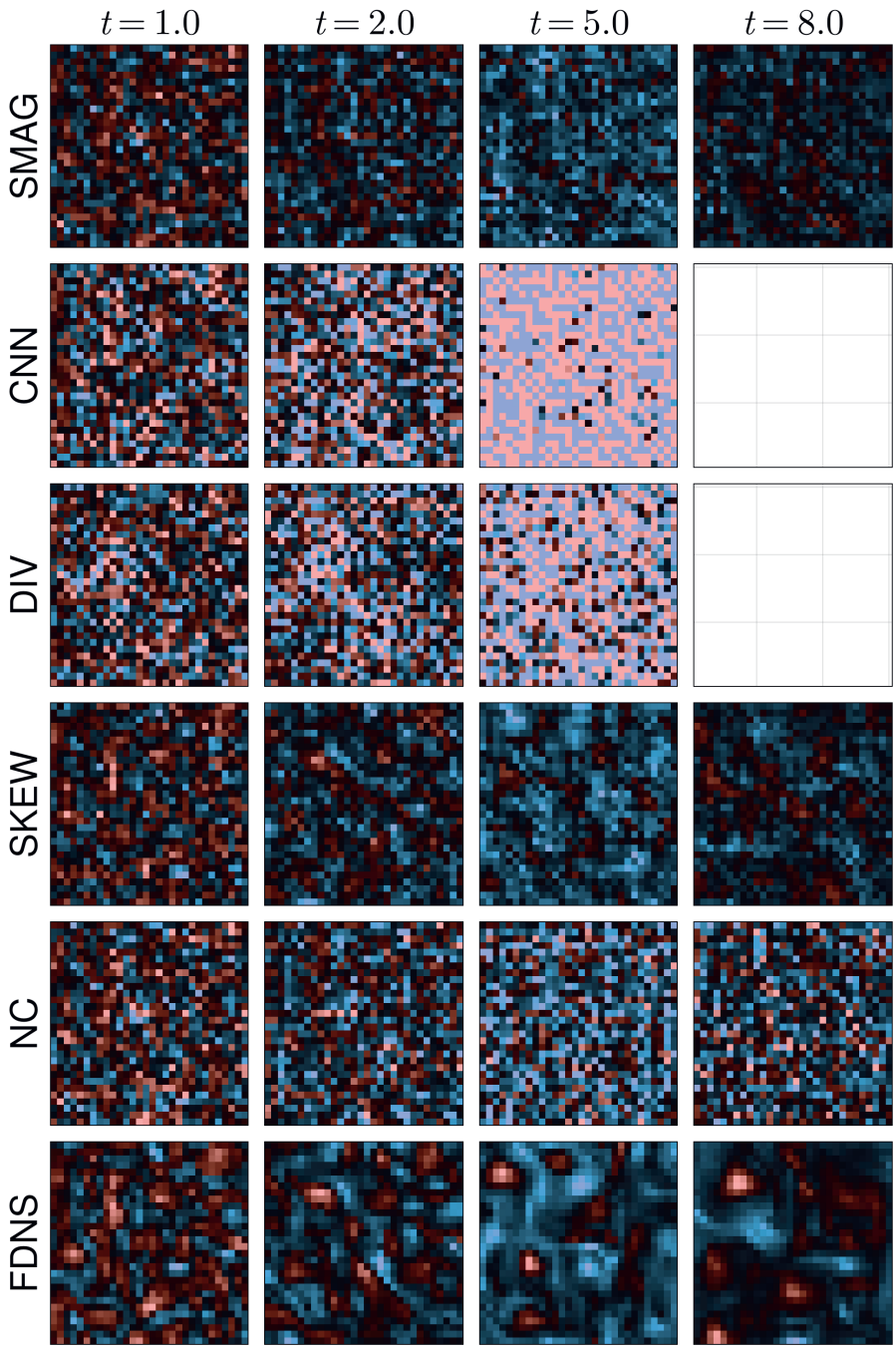


Figure 7.7: Vorticity fields at each point in time for each of the closure models on a 32×32 grid. Simulations correspond to the decaying turbulence test case. Blank boxes indicate an unstable simulation.

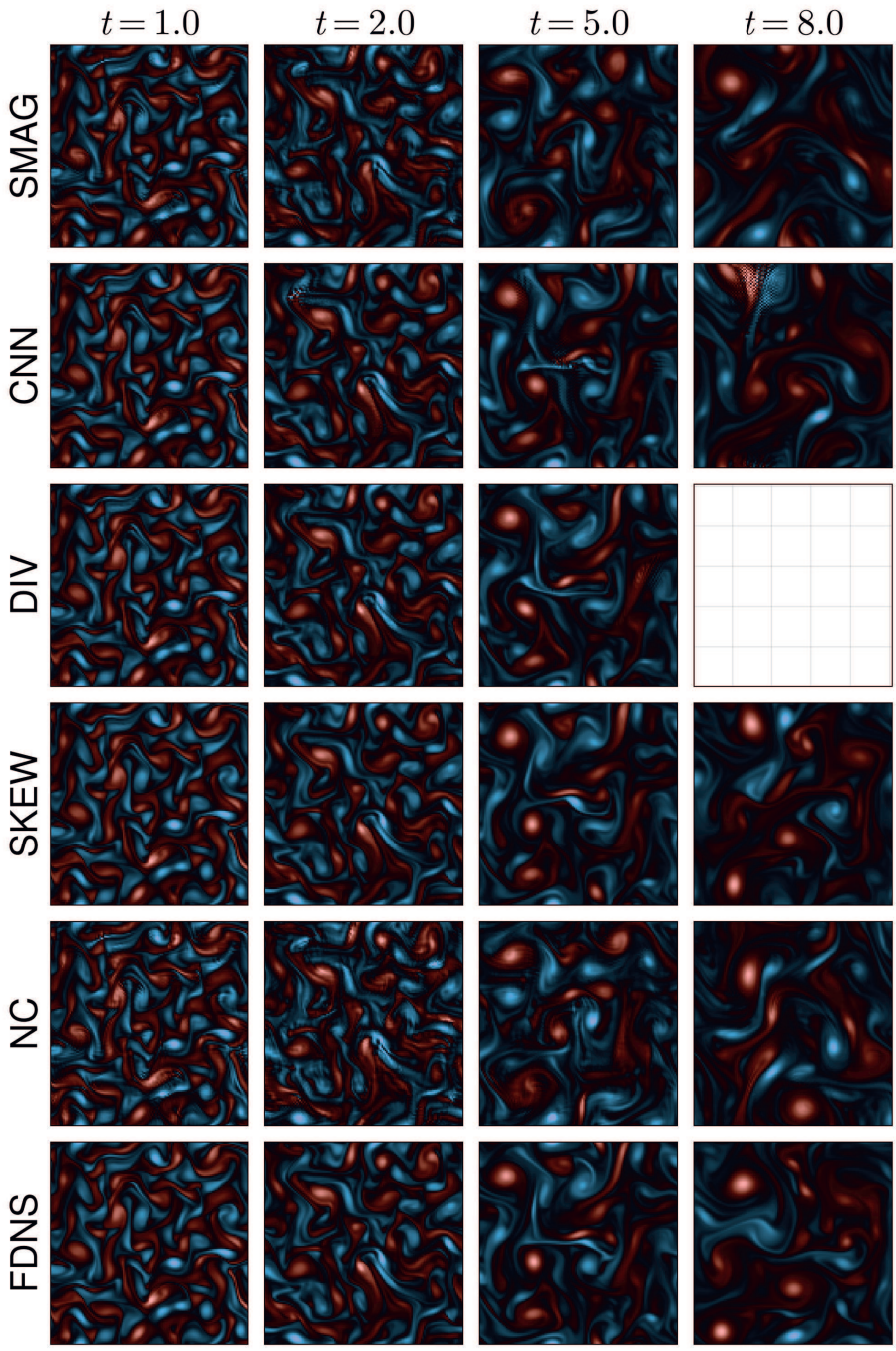


Figure 7.8: Vorticity fields at each point in time for each of the closure models on a 128×128 grid. Simulations correspond to the decaying turbulence test case. Blank boxes indicate an unstable simulation.

C Appendix Chapter 4

C.1 Parameter optimization in state-of-the-art approaches

This section is dedicated to describing additional details about the selection of optimal parameters in well-known state-of-the-art approaches to deal with underresolved turbulence simulations. In particular, we consider:

- **Smagorinsky** model [10, 26, 199], which is one of the earliest and most widely used subgrid-scale models in the LES of turbulent flows. It belongs to the class of eddy viscosity models and introduces an eddy viscosity given by:

$$\nu_t = \theta^2 \overline{\Delta}^2 \sqrt{2\,\mathrm{tr}(\overline{S}\,\overline{S})},$$

  where $\overline{\Delta}$ is the filter width, typically related to the grid size, $\overline{S} = \frac{1}{2}(\nabla\overline{\mathbf{u}} + \nabla\overline{\mathbf{u}}^T)$ is the strain rate tensor, and $\theta \in [0, 1]$ is the *Smagorinsky coefficient*. Such a parameter is typically fitted to reference data [89, 200], in our case the filtered DNS.

- **Standard EFR** model, introduced in Section 4.2.3, which relies on two additional steps after the *evolve* one (corresponding to discretized Navier–Stokes):

  - A differential filter step, depending on the filter radius δ. Common choices for δ are either the minimum mesh size $h_{\min}$ or the Kolmogorov length scale η [201].
  - A relax step depending on a parameter χ. A standard choice is $\chi \sim \Delta t$ [159], where Δt is the time step of the simulation, but other choices have been explored in [18, 62, 189].

  In this case, we select $\delta = h = 7.81 \times 10^{-3}$ and only optimize χ.

- **Standard EF** model, which corresponds to EFR with $\chi = 1$. In this case, we optimize the filter radius δ.

Each of the above-mentioned models depends on one or more parameters that require careful calibration. Here, we use a gradient-based optimization to determine the optimal values. In particular, we employ a *finite-difference stochastic gradient descent*, where the loss function is derived from $I_{\text{train}} = 10$ ensemble-averaged simulations, and the optimization seeks to minimize the enstrophy mismatch by adjusting the parameter of interest.

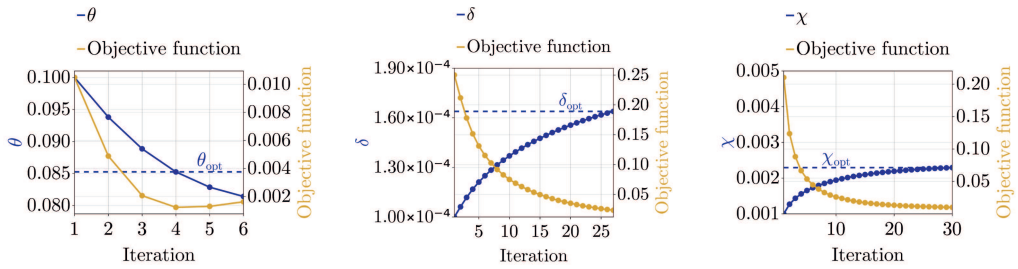
The gradient descent algorithm used for the gradient descent is proposed in 1. In the proposed algorithm, we consider a generic parameter α that needs to be optimized, and we call $\mathcal{M}(\alpha)$ the numerical simulation depending on it. For example, in the EF model, $\mathcal{M}(\alpha)$ is the EF simulation and α coincides with δ.

The loss function is the relative mean squared error (RMSE) in the global enstrophy. More in detail, the enstrophy is evaluated for $I_{\text{train}} = 10$ different train simulations which differ only in the random initialization $\boldsymbol{\mu}^{(j)}$ $(j = 1, \dots, I_{\text{train}})$, and then averaged.

We selected the enstrophy error as objective function because other metrics, like the energy error, do not provide significant insights into the performance of the method. The global enstrophy, instead, is a meaningful metric in fluid dynamics because it quantifies the intensity of vorticity in a flow field, providing direct insight into small-scale structures and dissipative mechanisms.

Figure 7.10 confirms that noEFR is indeed the least overdiffusive method. In fact, if the kinetic energy error is taken as the objective function, the state-of-the-art approaches effectively reduce to noEFR — that is, setting $\delta = 0$, $\chi = 0$ in the standard EF/EFR formulations, and $\theta = 0$ in the Smagorinsky model. This observation supports the fairness of the comparison between the enstrophy-based state-of-the-art approaches and the proposed data-driven methods.

Figure 7.9 shows how the parameters and the objective function are updated during the iterations of the gradient descent. We emphasize that each state-of-the-art method involves parameters with different orders of magnitude. For a fair comparison, we manually tune the tolerance η, the step size β, and the admissible range $[\alpha_{\min}, \alpha_{\max}]$ based on the typical scale of the corresponding quantity. The seemingly slower convergence of standard EF in Figure 7.9b is due to the fact that $\delta \sim \mathcal{O}(1 \times 10^{-4})$, whereas in standard EFR we have $\chi \sim \mathcal{O}(1 \times 10^{-3})$, and in the Smagorinsky model $\theta \sim \mathcal{O}(1 \times 10^{-1})$. As a result, the tolerance used in standard EF is slightly higher in relative terms, which can influence the perceived convergence rate.



(a) Smagorinsky model (b) Standard EF model (c) Standard EFR model

Figure 7.9: Parameter tuning of state-of-the-art approaches using approximate gradient descent method.

Table 7.2 shows an overview of the optimal values obtained for the parameters, and the corresponding admissible range considered in the optimization.

Figure 7.10 represents the kinetic energy, global enstrophy trends in time, and the energy

Algorithm 1 Gradient descent for optimizing a parameter α in a method of type $\mathcal{M}(\alpha)$.

Require: Initial parameter α_0; step size β; perturbation ε; max iterations $N_{\max}$; tolerance η; bounds $[\alpha_{\min}, \alpha_{\max}]$; $\boldsymbol{\mu}^{(i)}$, $i = 1, \ldots, I_{\text{train}}$ random initializations; $n = 1, \ldots, N_{\text{train}}$ training time steps.

Set $\alpha \leftarrow \alpha_0$
for $k = 1$ to $N_{\max}$ **do**
  for $i = 1$ to I_{train} **do**
    $\mathcal{Z}_h^n(\boldsymbol{\mu}^{(i)}) \leftarrow \mathcal{M}(\alpha, \boldsymbol{\mu}^{(i)})$ ▷ Run simulation for each initialization
    $\text{RMSE}^{(i)}(\alpha) = \frac{1}{N_{\text{train}}} \sum_{n=1}^{N_{\text{train}}} \frac{\left(\mathcal{Z}_h^n(\boldsymbol{\mu}^{(i)}) - \mathcal{Z}_{\text{ref}}^n(\boldsymbol{\mu}^{(i)})\right)^2}{(\mathcal{Z}_{\text{ref}}^n(\boldsymbol{\mu}^{(i)}))^2}$ ▷ Compute RMSE in global enstrophy
  end for
  $\mathcal{L}(\alpha) \leftarrow \dfrac{1}{I_{\text{train}}} \sum_{i=1}^{I_{\text{train}}} \text{RMSE}^{(i)}(\alpha)$
  $\alpha_{\text{perturbed}} \leftarrow \text{clamp}(\alpha + \varepsilon, \alpha_{\min}, \alpha_{\max})$ ▷ Restrict parameter to the bounds
  for $i = 1$ to I_{train} **do**
    $\mathcal{Z}_{h,\ \text{perturbed}}^n(\boldsymbol{\mu}^{(i)}) \leftarrow \mathcal{M}(\alpha_{\text{perturbed}}, \boldsymbol{\mu}^{(i)})$
    $\text{RMSE}^{(i)}(\alpha_{\text{perturbed}}) = \frac{1}{N_{\text{train}}} \sum_{n=1}^{N_{\text{train}}} \frac{\left(\mathcal{Z}_{h,\ \text{perturbed}}^n(\boldsymbol{\mu}^{(i)}) - \mathcal{Z}_{\text{ref}}^n(\boldsymbol{\mu}^{(i)})\right)^2}{(\mathcal{Z}_{\text{ref}}^n(\boldsymbol{\mu}^{(i)}))^2}$
  end for
  $\mathcal{L}(\alpha_{\text{perturbed}}) \leftarrow \dfrac{1}{I_{\text{train}}} \sum_{i=1}^{I_{\text{train}}} \text{RMSE}^{(i)}(\alpha_{\text{perturbed}})$
  $g \leftarrow \dfrac{\mathcal{L}(\alpha_{\text{perturbed}}) - \mathcal{L}(\alpha)}{\varepsilon}$ ▷ Gradient estimation
  $\alpha_{\text{new}} \leftarrow \text{clamp}(\alpha - \beta g, \alpha_{\min}, \alpha_{\max})$ ▷ Gradient descent update
  if $|\alpha_{\text{new}} - \alpha| < \eta$ **then**
    Converged; **break**
  end if
  $\alpha \leftarrow \alpha_{\text{new}}$
end for
return α

Model	Parameter	Parameter range	Optimal value
Smagorinsky	θ	$[0, 1]$	0.085
Standard EF	δ	$[0.1\,\eta, 10\,\eta]$	$1.64 \times 10^{-4} \simeq 0.5\eta$
Standard EFR	χ $(\delta = h)$	$[0.1\,\Delta t, 10\,\Delta t]$	$2.3 \times 10^{-3} \simeq 4.5\,\Delta t$

Table 7.2: Optimized parameters in traditional approaches methodologies.

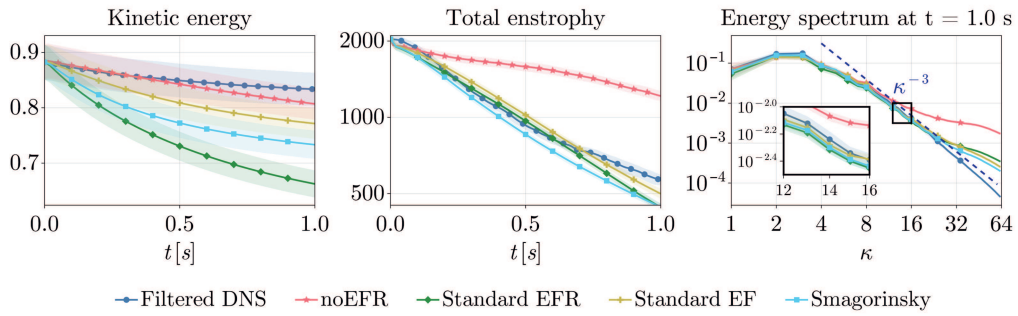


Figure 7.10: Time evolution for kinetic energy and global enstrophy, spectrum at $t = 1$. The plot represents the average and 95% confidence intervals over $I_{\text{train}} = 10$ initializations, in the decaying turbulence test case.

spectrum at $t = 1.$, corresponding to the final time used for the optimization. The results indicate good agreement in the global enstrophy, as this quantity is directly targeted during the optimization process. However, the energy time history reveals that all methods exhibit excessive dissipation, with noEFR showing the closest match to the filtered DNS.

This motivates the development of our data-driven filtering approach in the following part.

C.2 Additional results on data-driven EFR strategies

This part of the supplementary material is dedicated to additional results related to the data-driven approaches presented in the paper.

C.2.1 Decaying homogeneous turbulence test case

This section outlines extra results on the data-driven techniques using $T_{\text{train}} = 1$ s in the full data regime (Figure 7.11) and $T_{\text{train}} = 3$ s in the scarce data regime (Figure 7.12).

In the first case (Figure 7.11), we can notice good accuracies of DD-EF before $t = 5$ s, and an overdissipative behaviour for larger times. This is due to the fact the model takes as input less data and has less extrapolation capability.

In the second case (Figure 7.12), even in the scarce regime, the DD-EF(R) methods are approximating well the solution also at large times, due to the larger training time. Moreover, DD-EF is characterized by some instabilities in the kinetic energy, which are alleviated by the addition of the energy constraint.

The solution fields at $t = 1$ s for DD-EF, $\mathcal{E}$-DD-EFR, and $\mathcal{EZ}$-DD-EFR in scarce data regimes are represented in Figure 7.13. The vorticity solutions are qualitatively very similar, and exhibit some spurious oscillations in the case $T_{\text{train}} = 1$.

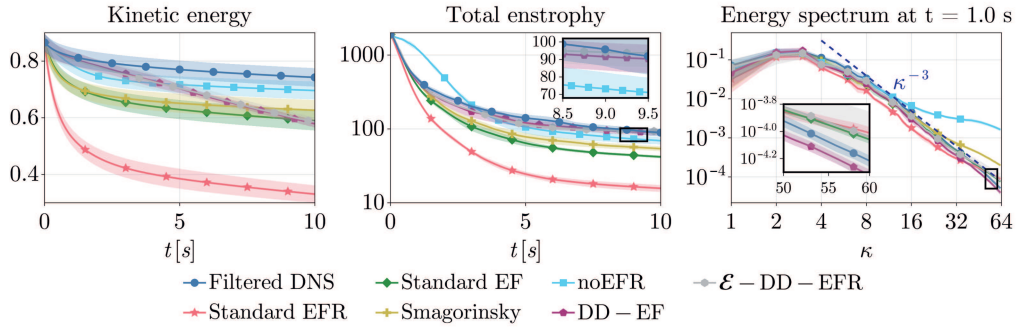


Figure 7.11: Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed DD-EF(R) methodologies and traditional approaches. In particular, we show the case $T_{\text{train}} = 1$ s. For each method, the figure shows the average values (solid lines) and the 95% confidence interval among 5 test configurations, in the decaying turbulence test case.

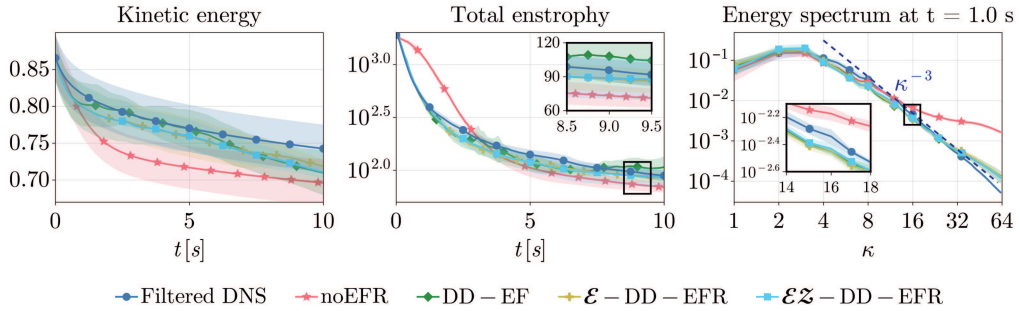
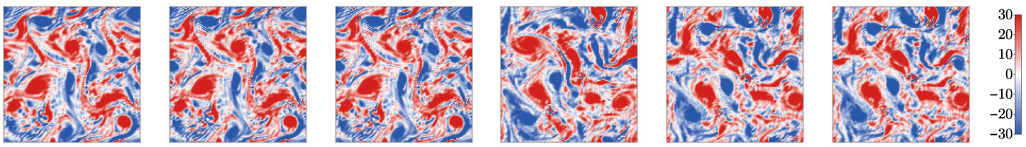


Figure 7.12: Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for DD-EF(R) methodologies and for state-of-the-art approaches. In particular, we show the case $T_{\text{train}} = 3$ s. For each method, the figure shows the average values (solid lines) and the 95% confidence interval among 5 test configurations, in the decaying turbulence test case.



(a) DD-EF $T_{\text{train}} = 3$ s | (b) $\mathcal{E}$-DD-EFR $T_{\text{train}} = 3$ s | (c) $\mathcal{EZ}$-DD-EFR $T_{\text{train}} = 3$ s | (d) DD-EF $T_{\text{train}} = 1$ s | (e) $\mathcal{E}$-DD-EFR $T_{\text{train}} = 1$ s | (f) $\mathcal{EZ}$-DD-EFR $T_{\text{train}} = 1$ s

Figure 7.13: Vorticity fields at $t = 1$ s, for different methodologies in a test configuration, in the decaying turbulence regime.

C.2.2 Kolmogorov test case

This section is dedicated to extra results in the Kolmogorov test case, specifically in the case $T_{\text{train}} = 1$, both for full and scarce data (Figures 7.14 and 7.15, respectively).

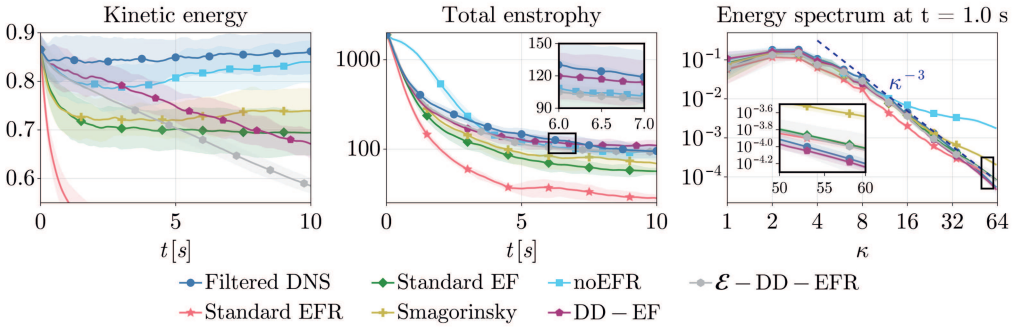


Figure 7.14: Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed methodologies and for state-of-the-art approaches. In particular, we show the case $T_{\text{train}} = 1$ s, $I_{\text{train}} = 10$. For each method, the figure shows the average values (solid lines) and the 95% confidence interval among 5 test configurations, in the Kolmogorov test case.

As noticed in the decaying turbulence case, when T_{train} is smaller, the DD-EF result either in stable but overdiffusive solutions (full data) or in unstable solutions (scarce data). The energy and enstrophy constraints alleviate such oscillations and instabilities.

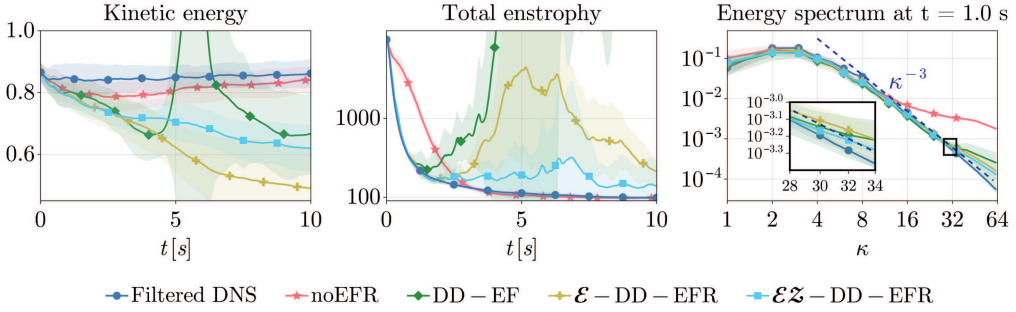
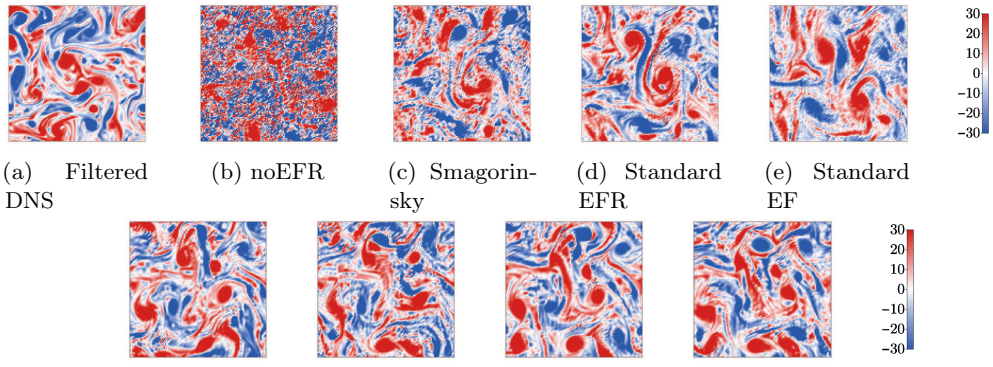


Figure 7.15: Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed methodologies and for state-of-the-art approaches. In particular, we show the case $T_{\text{train}} = 1$ s, $I_{\text{train}} = 1$ (scarce data). For each method, the figure shows the average values (solid lines) and the 95% confidence interval among 5 test configurations, in the Kolmogorov test case.

The solutions at fixed time instance $t = 1$ s are represented in Figure 7.16.



(a) Filtered DNS (b) noEFR (c) Smagorinsky (d) Standard EFR (e) Standard EF

(g) DD-EF $T_{\text{train}} = 3$ (h) $\mathcal{E}$-DD-EFR $T_{\text{train}} = 3$ (i) DD-EF $T_{\text{train}} = 1$ (j) $\mathcal{E}$-DD-EFR $T_{\text{train}} = 1$

Figure 7.16: Vorticity fields at $t = 1$ s, for different methodologies in a test configuration.

D Appendix Chapter 5

D.1 2D Bump kernel

Here we discuss the kernel used to construct the LO-POD basis for the 2D advection test case. This kernel is required to obtain basis functions that smoothly decay to zero at the edge of the subdomain. To achieve this, we use a standard bump function:

$$\tilde{\psi}(x) = \begin{cases} \exp(\frac{-1}{1-|x|}) & \text{if } |x| < 1, \\ 0 & \text{elsewhere,} \end{cases} \tag{7.38}$$

to construct the kernel [202]. This function, along with its derivatives, smoothly decays to zero as $|x|$ approaches 1. Next, we apply normalization

$$\psi(x) = \begin{cases} \frac{\tilde{\psi}(x)}{\tilde{\psi}(x)+\tilde{\psi}(1-|x|)} & \text{if } |x| < 1, \\ 0 & \text{elsewhere,} \end{cases} \tag{7.39}$$

to ensure the kernels form a partition of unity, see (5.26). The 2D kernel is built using a product of two normalized bump functions

$$k(\mathbf{x}) = \psi(x)\psi(y), \tag{7.40}$$

where $\mathbf{x} = (x, y)^T \in \mathbb{R}^2$ are the spatial coordinates. For the presentation of this kernel, we exploit the fact that we use a uniform grid. This allows us to conveniently subdivide the domain into square subdomains Ω_i of the same size. Let $\boldsymbol{\alpha}^i, \boldsymbol{\beta}^i \in \mathbb{R}^2$ denote the coordinates of the bottom-left and top-right vertex. Each subdomain overlaps with its four neighboring subdomains, which share a vertex in the middle of the considered domain. The kernel associated with subdomain Ω_i is

$$k_i(\mathbf{x}) = \psi(\hat{x}_i)\psi(\hat{y}_i), \tag{7.41}$$

where $\hat{x}_i = 2\frac{x-\alpha_1^i}{\beta_1^i-\alpha_1^i} - 1$ and $\hat{y}_i = 2\frac{x-\alpha_2^i}{\beta_2^i-\alpha_2^i} - 1$ are the normalized coordinates. The kernel is depicted in Figure 7.17.

D.2 Choice of subdomains and modes for 2D test case

In this section, we present the projection error on the training and validation sets for the 2D advection test case. This is presented in Figure 7.18. Based on these results, we select the number of modes q which first surpasses the projection error of 10^{-2} for $I = 8 \times 8$. This

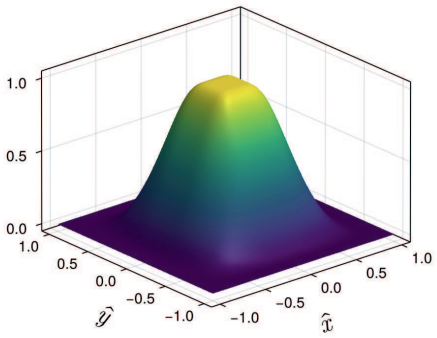


Figure 7.17: Bump kernel used to construct the LO-POD basis for the 2D advection test case.

value of I is chosen as it generalizes well, i.e., the performance on the validation set is similar to the training set and the resulting ROM is sparser than for $I = 4 \times 4$. For L-POD this corresponds to $q = 20$ and for LO-POD to $q = 15$.

D.3 Time step size

In Figure 7.19, we present the error of ROMs integrated using different time step sizes. The maximum time step sizes are selected so that performance degradation or the occurrence of instabilities is avoided. These are presented in Table 5.1.

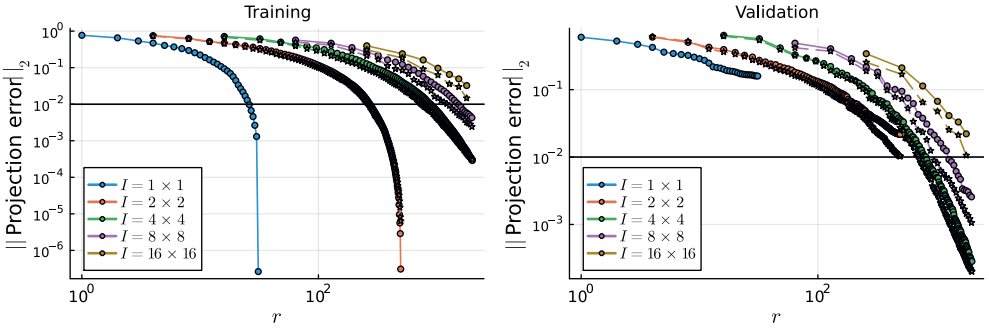


Figure 7.18: Projection error evaluated over the training (left) and validation (right) data set for the 2D advection test case. $I = 1 \times 1$ corresponds to space-global proper orthogonal decomposition (G-POD). Solid dots correspond to space-local proper orthogonal decomposition (L-POD) and stars to LO-POD. The horizontal black line corresponds to a projection error of 10^{-2}.

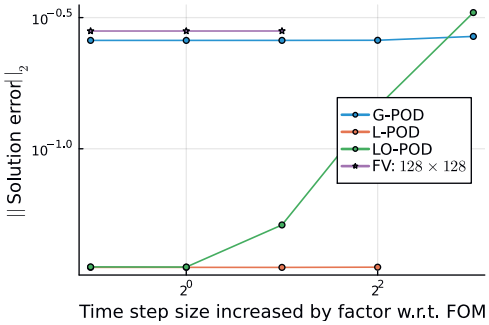


Figure 7.19: Solution error averaged over the simulation for the 2D advection test case, the ROMs for each time step size. Results for a finite volume (FV) discretization on a coarser grid than the FOM resolution (256×256) are also depicted. Absence of markers indicates an unstable simulation. The LO-POD ROM is integrated using a Crank-Nicolson scheme, as opposed to RK4 for the others, and is therefore unconditionally stable.