

SISSA

Scuola
Internazionale
Superiore di
Studi Avanzati

Mathematics Area - PhD course in
Mathematical Analysis, Modelling, and Applications

From Filtering to Data-Driven Closures: Stabilization Strategies for Convection-dominated Flow Simulations

Candidate:
Anna Ivagnes

Advisor:
Prof. Gianluigi Rozza
Co-advisors:
Prof. Giovanni Stabile

Academic Year 2025-26



A Francesco

ACKNOWLEDGEMENTS

I would like to express my gratitude to my PhD advisor, Prof. Gianluigi Rozza, for the invaluable guidance, continuous support, and encouragement throughout my doctoral journey.

I am sincerely grateful to my co-advisor, Prof. Giovanni Stabile, for his thoughtful advice, constructive scientific discussions, and constant availability, despite the physical distance.

I would like to thank both my advisor and co-advisor for the trust they placed in me throughout my doctoral journey. I am particularly grateful for the freedom they granted me to explore the research directions that genuinely inspired me and to establish scientific collaborations beyond the mathLab group, always encouraging my curiosity and independence rather than imposing predefined paths. Their guidance, support, and confidence in my choices allowed me to grow as a researcher and to develop critical thinking, autonomy, and scientific independence.

I would also like to thank all the colleagues and collaborators who have contributed to this journey. I want to thank Prof. Benjamin Sanderse and everyone at CWI -especially Toby, Syver, Henrik, Robin, and Rik, to name just a few- for welcoming me so warmly during my three-month internship in the summer of 2025. Beyond the stimulating scientific environment, they made me feel part of the group from day one, and I will always look back on that experience with great appreciation.

I am deeply grateful to Prof. Traian Iliescu for the fruitful scientific collaborations we have shared and for his insightful advice, which has always been remarkably thoughtful and valuable.

A special thank you goes to Dr. Maria Strazzullo, who believed in me from the very beginning and has always encouraged me, both scientifically and personally. Her support, advice, and enthusiasm made my one-month stay in Virginia in 2023 far more enriching and enjoyable than I could have imagined.

I would also like to thank Prof. Paola Cinnella and Prof. Michele Girfoglio for the opportunity to collaborate with them, for the stimulating discussions, and the enjoyable scientific work we carried out together.

Among my colleagues at SISSA mathLab, I would especially like to thank Dr. Niccolò Tonicello, for the scientific discussions and collaborations, and for his constant availability to exchange ideas.

I would also like to thank Gabriele, Piero, and Maria Garcia. I had the privilege of

supervising their work, and I have learned just as much from them as I hope they have learned from me.

Finally, I would like to thank all my colleagues and friends at SISSA mathLab for making these years both scientifically stimulating and personally enjoyable. Thank you for the discussions, the ideas, the coffee breaks, and for creating such a welcoming and inspiring environment.

At this point, I would like to switch to my mother tongue for the remaining acknowledgements to my friends and family.

Ringrazio le persone che ho conosciuto a Trieste -tra cui Pier, Caterina, Niccolo', Emanuela, Giulio, Gaia (e tanti altri) -per la loro amicizia e per tutti i momenti felici trascorsi insieme.

Grazie agli *araldi* -Maria, Federico, Davide, Ivan, Luca e Mattia- per le conferenze condivise, le risate e le serate trascorse insieme, che hanno reso questo percorso ancora più speciale. Un ringraziamento speciale va a Maria, consigliera scientifica e non, per il mese trascorso insieme a Blacksburg, per la splendida amicizia nata in quell'occasione, e per tutti i momenti condivisi che porterò sempre con me.

Grazie alle amiche storiche -Celeste, Gabriella, Chiara e Ilaria- per essere presenti nonostante la distanza.

Grazie alla mia *opossum* Sara, per esserci sempre, fin dal primo giorno in Via de Jacobis. La tua amicizia è preziosa e insostituibile.

Grazie Nicola, per i momenti condivisi nella quotidianità e per i viaggi incredibili fatti insieme. Grazie per essere stato al mio fianco in questi quattro anni, nonostante i miei alti e bassi dovuti al dottorato, e per avermi fatto ritrovare la fiducia in me stessa nei momenti in cui mi mancava. Grazie per tutti gli input scientifici e il supporto morale che mi hai dato quando il mio cervello era annebbiato dall'ansia.

Grazie a mia sorella Sara per l'affetto, il sostegno, e per essere stata un punto di riferimento in ogni momento della mia vita.

Grazie a mamma e papà per l'amore che mi dimostrate sempre e per aver creduto in me, indipendentemente dalle mie scelte.

Grazie allo zio Totò e alla zia Anna, la vostra forza e il vostro esempio sono per me un riferimento prezioso.

Grazie al mio angelo custode e fan dalle scuole elementari. Se mi leggi, ovunque tu sia, mi manchi.

Trieste, August 2026.

ABSTRACT

This PhD thesis investigates advanced stabilization strategies for the simulation of convection-dominated flows. The primary objective is to enhance the numerical stability and accuracy of classical methods through the integration of filtering and correction strategies.

The first part of the thesis focuses on the stabilization of **marginally-resolved simulations**, where insufficient spatial resolution often leads to numerical instabilities, with the use of filtering strategies, in the so-called Evolve-Filter-Relax (EFR) framework. A comprehensive review of existing filtering strategies is presented, followed by the development of novel methodologies. These include an optimized EFR algorithm with parameter tuning over time, a reinforcement learning-based adaptive filtering strategy, and a novel data-driven energy-conserving EFR formulation. Together, these contributions demonstrate how adaptive and learnable filtering techniques can significantly improve stability and robustness.

Motivated by the need to reduce the computational cost of high-fidelity simulations, the second part focuses on **reduced order methods (ROMs)**, particularly in the Proper Orthogonal Decomposition (POD)-based framework. After reviewing the main stabilization techniques available in the literature, the thesis introduces novel data-driven filtering and correction methodologies to improve the performance of POD-based ROMs. These include replacing classical ROM filters with learned stabilization operators and introducing data-driven closure terms to recover the effects of the truncated POD modes. Furthermore, the proposed closure modeling strategies are extended to non-intrusive ROMs through machine-learning-based correction models operating in a purely data-driven setting. Overall, these methodologies significantly improve both the stability and the predictive accuracy of classical POD-based ROMs.

While the second part remains inherently limited by the linear POD projection, the third part of the thesis explores the transition toward **nonlinear reduced-order modeling**. After reviewing the main nonlinear ROM methodologies available in the literature, the thesis investigates convolutional neural network architectures for the efficient compression of full-order operators, laying the foundations for more expressive and physics-based nonlinear ROMs.

Overall, this work highlights the potential of combining stabilization strategies with machine learning and data-driven techniques to address key challenges in flow simulations, both on physical grids and in ROMs.

CONTENTS

Abstract	v
Contents	vi
List of Figures	x
List of Tables	xx
List of Algorithms	xxiii
Acronyms	xxiv
Introduction	1
I Data-driven and adaptive filtering in marginally-resolved simulations	7
1 Filtering approaches: literature review	8
1.1 The Navier–Stokes Equations and turbulence	8
1.2 Filtering strategies for turbulent flow simulations	10
1.2.1 The Smagorinsky model as an implicit filter	13
1.3 The EFR approach	14
1.3.1 The effect of different filters: a numerical example	17
1.4 Motivation for the proposed approaches	19
2 Opt-EFR: parameters’ optimization for EFR	23
2.1 The flow past a cylinder in FEniCS: standard EFR results	24
2.2 The data-driven optimized EFR algorithms	25
2.2.1 The objective functions	26
2.2.2 The optimization algorithm	29
2.3 Numerical results of Opt-EFR	30
2.3.1 Numerical results of χ -Opt-EFR	30
2.3.2 Numerical results of δ -Opt-EF	33
2.3.3 Numerical results of $\delta\chi$ -Opt-EFR	36
2.3.4 Comparison and discussion of the Opt-EFR results	36
2.4 Conclusions and motivations to RL	40
3 RL-EF: adaptive filtering via Reinforcement learning	43
3.1 The two test cases in FEniCSx: standard EF results	44

3.2	Reinforcement Learning for the Evolve-Filter approach	47
3.2.1	Neural networks	47
3.2.2	An introduction to RL	48
3.2.3	The RL-EF approach	50
3.3	Numerical results of RL-EF	55
3.3.1	Flow past a cylinder test case	55
3.3.2	Decaying homogeneous isotropic turbulence test case	62
3.3.3	Computational details	68
3.4	Conclusions and motivations to DD-EF(R) approaches	69
4	DD-EFR: a data-driven alternative to differential filters	72
4.1	Preliminaries: the energy-conserving solver and the staggered grid frame- work	73
4.2	DD-EFR methodology	75
4.2.1	Data-driven linear filter	75
4.2.2	Energy and enstrophy-conserving approaches	80
4.3	Numerical results of DD-EFR	82
4.3.1	Decaying homogeneous isotropic turbulence test case	82
4.3.2	Kolmogorov test case	88
4.3.3	Efficiency of the methods	91
4.4	Conclusions and outlooks	93
4.4.1	Conclusions to Part I and motivations to Part II	94
II	Stabilization approaches for reduced order methods	96
5	Stabilized POD-based ROMs: literature review	97
5.1	POD-Galerkin ROMs (G-ROMs)	100
5.1.1	Treatment of non-homogeneous Dirichlet boundary conditions	103
5.2	Pressure stabilization techniques in ROMs	104
5.2.1	Supremizer enrichment	104
5.2.2	Pressure Poisson Equation in ROMs	105
5.2.3	Is pressure stabilization enough?	106
5.3	Filtering approaches for ROMs	108
5.3.1	Leray-ROM (L-ROM)	111
5.4	Closure approaches for ROMs	111
5.4.1	Eddy Viscosity-based ROM (EV-ROM)	112
5.4.2	Data-driven closure modeling in ROMs	115
5.5	Motivation for the proposed approaches	118
6	StabOp-L-ROM: data-driven stabilization for Leray-ROMs	120
6.1	Preliminaries: FOM, G-ROM and L-ROM	121

6.1.1	FOM	121
6.1.2	G-ROM	121
6.1.3	L-ROM	122
6.2	StabOp: a data-driven stabilization operator	122
6.3	Numerical results	126
6.3.1	Flow past a cylinder test case	127
6.3.2	Lid-driven cavity test case	130
6.3.3	Flow past a hemisphere test case	133
6.3.4	Minimal channel flow test case	137
6.3.5	Computational cost	140
6.4	Is StabOp a spatial filter?	141
6.5	Conclusions and Motivation to the next chapter	144
7	DD-EV-ROM: data-driven closures for POD-Galerkin ROMs	146
7.1	DD-EV-ROM: the algorithm	147
7.2	Turbulence and correction maps: neural operators	149
7.3	Numerical results	153
7.3.1	Flow past a cylinder test case	154
7.3.2	Channel-driven cavity flow test case	156
7.3.3	Backward-facing step test case	160
7.3.4	Computational time and Discussion	164
7.4	Conclusions and Motivation to the next chapter	167
8	Corrected-ROMs: an extension of closures to nonintrusive ROMs	170
8.1	Preliminaries: nonintrusive ROMs	172
8.2	Corrected-ROMs	173
8.2.1	Quad-LS (state of the art)	174
8.2.2	QuadNet	175
8.2.3	QuadNet- μ	176
8.3	Numerical results	177
8.3.1	Setup of Corrected-ROMs	179
8.3.2	Full data	181
8.3.3	Partial data	182
8.3.4	Scarce data	184
8.3.5	Computational cost	186
8.4	Conclusions	187
8.4.1	Conclusions to Part II and motivations to Part III	188
	III Towards nonlinear reduced order models	190
9	Nonlinear ROMs and convolutional architectures: literature review	191

9.1	Nonlinear dimensionality reduction via AutoEncoders	192
9.2	Convolutional architecture	194
9.2.1	Continuous Convolutional architecture	196
9.3	Motivation for the proposed approach	198
10	Convolutional Operator Compression strategies for ROMs	200
10.1	Methodology	200
10.1.1	The efficient use of continuous convolution for sparse matrices	204
10.2	Numerical results	205
10.2.1	Poisson test case	206
10.2.2	Advection-diffusion test case	210
10.2.3	Burgers test case	213
10.2.4	Computational details	221
10.3	Conclusions and Future Perspectives	225
	Conclusions	227
	List of thesis publications	233
	Bibliography	234

LIST OF FIGURES

1	Illustrative sketch of the pipeline followed in the thesis, and related methodologies.	3
1.1	Energy spectrum qualitative behaviour in the case of two-dimensional and three-dimensional turbulence.	10
1.2	Plot of the transfer functions of some explicit filters.	12
1.3	Initial vorticity in the HIT case.	18
1.4	<i>Decaying HIT test case.</i> Vorticity fields at $t = 0.5$, $t = 1$ and $t = 2$ for the filtered DNS (reference) simulation, noEFR, EF simulations with different filter types (Gaussian, DF), and simulation with Smagorinsky closure.	19
1.5	<i>Decaying HIT test case.</i> The effect of explicit and implicit filtering strategies in the EF approach on the energy spectrum, at different time instances. . . .	19
2.1	<i>Flow past a cylinder test case.</i> Domain and grids considered for the DNS and for the marginally-resolved simulation in FEniCS.	24
2.2	<i>Flow past a cylinder test case.</i> Velocity magnitude and pressure fields for the DNS, the noEFR, the standard EFR ($\delta = \eta$, $\chi = 5\Delta t$), and the standard EF ($\delta = \eta$, $\chi = 1$) simulations, at the final time $T = 4$	26
2.3	Sketch of the standard EFR and of the Opt-EFR procedures.	27
2.4	<i>Flow past a cylinder test case.</i> Optimal values of χ in time, and distribution of χ across all values in the admissible range $[0, 1]$, for the χ -Opt-EFR algorithms.	31
2.5	<i>Flow past a cylinder test case.</i> Velocity magnitude and pressure fields for the χ -Opt-EFR simulations, at the final time $T = 4$. The solutions for χ -Opt-EFR _{global} ^u are represented at $t = 1.6$, as the simulation blows up at later time.	32
2.6	<i>Flow past a cylinder test case.</i> Time evolution of the velocity and pressure relative L^2 errors with respect to the reference solutions. The errors are displayed for all the χ -Opt-EFR algorithms considered, for noEFR, and standard EFR approaches.	32
2.7	<i>Flow past a cylinder test case.</i> Time evolution of the velocity, velocity gradients and pressure L^2 norms, for all the χ -Opt-EFR algorithms considered, for noEFR, standard EFR and for the reference simulation.	33
2.8	<i>Flow past a cylinder test case.</i> Optimal values of δ in time, and distribution of δ across all values in the admissible range $[1 \times 10^{-5}, 1 \times 10^{-3}]$, for the δ -Opt-EF algorithms.	34

2.9	<i>Flow past a cylinder test case.</i> Velocity magnitude and pressure fields for the δ -Opt-EF simulations, at the final time $T = 4$	35
2.10	<i>Flow past a cylinder test case.</i> Time evolution of the velocity, velocity gradients and pressure L^2 norms, for all the δ -Opt-EF algorithms considered, for noEFR, standard EF and for the reference simulation.	35
2.11	<i>Flow past a cylinder test case.</i> Optimal values of δ and χ in time, and distribution of δ and χ across all values in the admissible range, for the $\delta\chi$ -Opt-EFR algorithms.	36
2.12	<i>Flow past a cylinder test case.</i> Time evolution of the velocity, velocity gradients and pressure L^2 norms, for all the $\delta\chi$ -Opt-EFR algorithms considered, for noEFR, standard EFR, standard EF, and for the reference simulation. . . .	37
2.13	<i>Flow past a cylinder test case.</i> Velocity magnitude and pressure fields for the $\delta\chi$ -Opt-EFR simulations at the final time $T = 4$	37
2.14	Time evolution of the velocity, velocity gradients and pressure L^2 norms, for the best performing algorithms of χ -Opt-EFR, δ -Opt-EF, and $\delta\chi$ -Opt-EFR. The results are compared with noEFR, standard EFR, standard EF, and the reference simulation.	37
2.15	<i>Flow past a cylinder test case.</i> Statistics (median and confidence intervals) for noEFR, standard EFR/EF, and methods of type $(\cdot)_{\text{global}}^{u,\nabla u,p}$. The results display the relative errors in the H^1 velocity norm, and in the L^2 pressure norm, with respect to the reference. The plots show the time-averaged error (orange line) and the confidence interval (boxes). The statistics are computed within the time window $[0, 4]$	38
2.16	<i>Flow past a cylinder test case.</i> Gain in the accuracy, considering the H^1 velocity and L^2 pressure relative errors. The results show the time-average gain of the best Opt-EFR methods with respect to the standard EFR and EF simulations.	39
2.17	<i>Flow past a cylinder test case.</i> Representation of how the optimization step affects the efficiency and accuracy of the velocity and pressure solution in $\delta\chi$ -Opt-EFR $_{\text{global}}^{u,\nabla u}$ and $\delta\chi$ -Opt-EFR $_{\text{global}}^{u,\nabla u,p}$ algorithms. The efficiency is represented in terms of the wall clock time. The accuracy is measured in terms of the time-average discrepancy in the velocity (2.17a), pressure (2.17b) and velocity gradients' (2.17c) norm. The numbers identify the number of time steps between two optimizations (N_{opt}).	41
3.1	<i>Flow past a cylinder test case.</i> grids considered for the DNS and for the marginally-resolved simulation in FEniCSx.	44
3.2	<i>Flow past a cylinder test case.</i> Time evolution of the velocity solution for the filtered DNS, standard EF, and noEF simulations.	46
3.3	<i>Decaying HIT test case.</i> Time evolution of the velocity solution for the filtered DNS, standard EF, and noEF simulations.	46

3.4	Basic illustration of a deep fully-connected neural network architecture (or multi-layer perceptron), with 2 inputs, 1 output, and 2 hidden layers of 5 neurons each.	48
3.5	Action space used for DQN in RL-EF.	50
3.6	Illustrative sketch of the RL-EF procedure.	51
3.7	<i>Flow past a cylinder test case.</i> Cumulative reward and loss value during training steps, for the RL-EF methods.	56
3.8	<i>Flow past a cylinder test case.</i> Action values over testing time, and corresponding frequency, for the RL-EF methods.	57
3.9	<i>Flow past a cylinder test case.</i> Time evolution of the velocity and velocity gradients' norms, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods.	58
3.10	<i>Flow past a cylinder test case.</i> Time evolution of drag and lift coefficients, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods. Drag and lift forces are computed on the cylinder boundary.	59
3.11	<i>Flow past a cylinder test case.</i> Energy spectrum at three time instances, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods. The spectrum is computed at the cylinder wake, after "artificially" making the domain periodic.	60
3.12	<i>Flow past a cylinder test case.</i> Time evolution of the velocity solution for the RL-EF simulations, compared with the filtered DNS.	60
3.13	<i>Flow past a cylinder test case.</i> Error of the kinetic energy averaged in time ($\text{err}_{\mathcal{E}}$), and log-space error of the energy spectrum averaged with respect to the wavenumbers and in time ($\text{err}_{\text{spectrum}}$). The energy spectrum error is computed over two different wavenumbers ranges, namely $K = 32$ and $K = 128$	61
3.14	<i>Decaying HIT test case.</i> Cumulative reward and loss value during training steps, for the RL-EF methods.	63
3.15	<i>Decaying HIT test case.</i> Action values over testing time, and corresponding frequency, for the RL-EF methods.	64
3.16	<i>Decaying HIT test case.</i> Time evolution of the velocity and velocity gradients' norms, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods.	65
3.17	<i>Decaying HIT test case.</i> Energy spectrum at three time instances, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods. The spectrum is computed considering the entire squared domain.	65
3.18	<i>Decaying HIT test case.</i> Time evolution of the velocity solution for the RL-EF simulations, compared with the filtered DNS.	66
3.19	<i>Decaying HIT test case.</i> Error of the kinetic energy averaged in time ($\text{err}_{\mathcal{E}}$), and log-space error of the energy spectrum averaged with respect to the wavenumbers and in time ($\text{err}_{\text{spectrum}}$). The energy spectrum error is computed over two different wavenumbers ranges, namely $K = 8$ and $K = 32$	67

4.1	Sketch of the fields evaluation and of the face-average filter in the staggered grid framework.	74
4.2	Workflow of the DD-EF and DD-EFR frameworks. An optimal filter is learned offline from filtered DNS data, and (optionally) the relaxation parameter is tuned to ensure energy and/or enstrophy dissipation.	76
4.3	<i>Decaying HIT test case.</i> The initial vorticity fields at three different random initializations (with seeds 1, 5, and 10). The filtered DNS solutions are considered to train the data-driven filter.	77
4.4	2D average of x and y components of the filter matrix $\hat{\mathcal{F}}^*$ 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). . . .	79
4.5	<i>Decaying HIT test case.</i> 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), noEFR, the state-of-the-art approaches, and the reference filtered DNS. The data-driven filter is computed when $T_{\text{train}} = 3$ s and $I_{\text{train}} = 10$ (full-data regime). For each method, the plots show the average values (solid lines) and the 95% confidence interval among $I_{\text{test}} = 5$ test configurations.	84
4.6	<i>Decaying HIT test case.</i> Vorticity fields at $t = 0.3$ for different methodologies in one of the I_{test} configurations.	84
4.7	<i>Decaying HIT test case.</i> Vorticity fields at $t = 1$ for different methodologies in one of the I_{test} configurations.	85
4.8	<i>Decaying HIT test case.</i> Distribution across values and time evolution of the relaxation parameter in the energy-conserving DD-EFR simulation, when the data-driven filter is computed with $T_{\text{train}} = 3$ s, $I_{\text{train}} = 10$ (full-data).	85
4.9	<i>Decaying HIT test case.</i> 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, $\mathcal{E}$ -DD-EFR, $\mathcal{E}\mathcal{Z}$ -DD-EFR), noEFR, and the reference filtered DNS. The data-driven filter is computed when $T_{\text{train}} = 1$ s and $I_{\text{train}} = 1$ (scarce-data regime). For each method, the plots show the average values (solid lines) and the 95% confidence interval among $I_{\text{test}} = 5$ test configurations.	86
4.10	<i>Decaying HIT test case.</i> Distribution across values and time evolution of the relaxation parameter in the energy-conserving DD-EFR simulation, when the data-driven filter is computed with $T_{\text{train}} = 1$ s, $I_{\text{train}} = 1$ (scarce-data). . . .	87
4.11	<i>Decaying HIT test case.</i> 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 $I_{\text{test}} = 5$ test configurations.	87

4.12	<i>Kolmogorov test case.</i> 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), noEFR, the state-of-the-art approaches, and the reference filtered DNS. The data-driven filter is computed (from the decaying HIT reference data) when $T_{\text{train}} = 3$ s and $I_{\text{train}} = 10$ (full-data regime). For each method, the plots show the average values (solid lines) and the 95% confidence interval among $I_{\text{test}} = 5$ test configurations.	89
4.13	<i>Kolmogorov test case.</i> 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, $\mathcal{E}$ -DD-EFR, $\mathcal{E}\mathcal{Z}$ -DD-EFR), noEFR, and the reference filtered DNS. The data-driven filter is computed (from the decaying HIT reference data) when $T_{\text{train}} = 3$ s and $I_{\text{train}} = 1$ (scarce-data regime). For each method, the plots show the average values (solid lines) and the 95% confidence interval among $I_{\text{test}} = 5$ test configurations.	90
4.14	<i>Kolmogorov test case.</i> Distribution across values and time evolution of the relaxation parameter in the energy-conserving DD-EFR simulation, when the data-driven filter is computed with $T_{\text{train}} = 3$ s, $I_{\text{train}} = 1$ (scarce-data). . . .	90
4.15	<i>Kolmogorov test case.</i> 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 $I_{\text{test}} = 5$ test configurations.	91
4.16	<i>Decaying HIT test case.</i> Average energy-enstrophy accuracy of the simulations, where the radius of the circles is proportional to the average wall time needed to perform online simulations.	93
5.1	Schematic illustration of the general ROM pipeline.	98
5.2	<i>Flow past a cylinder test case.</i> SUP-ROM, PPE-ROM, and FOM velocity magnitude and pressure fields. The FOM and ROM settings are described in [143]. The fields are at final ROM time and at fixed $Re = 50\,000$	107
5.3	<i>Lid-driven cavity test case.</i> Example of the effect of the DF with three different δ values at three latent dimensions. The DF is applied for a single time step to a G-ROM solution, as in [336]. The graphical fields (left) and the reduced coefficients value (right) are displayed before (G-ROM) and after applying the filter (DF).	110
6.1	<i>Flow past a cylinder test case.</i> Kinetic energy variation in time for the FOM reference, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	129

6.2	<i>Flow past a cylinder test case.</i> Mean squared error of kinetic energy with respect to FOM, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	129
6.3	<i>Flow past a cylinder test case.</i> Energy spectrum at latest time for the reference FOM, standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	130
6.4	<i>Flow past a cylinder test case.</i> Stability and accuracy measures for standard G-ROM, L-ROM, and StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	130
6.5	<i>Lid-driven cavity test case.</i> Kinetic energy variation in time for the FOM reference, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	132
6.6	<i>Lid-driven cavity test case.</i> Mean squared error of kinetic energy with respect to FOM, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	132
6.7	<i>Lid-driven cavity test case.</i> Energy spectrum at latest time for the reference FOM, standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	133
6.8	<i>Lid-driven cavity test case.</i> Stability and accuracy measures for standard G-ROM, L-ROM, and StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	133
6.9	<i>Flow past a hemisphere test case.</i> Kinetic energy variation in time for the FOM reference, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	135
6.10	<i>Flow past a hemisphere test case.</i> Mean squared error of kinetic energy with respect to FOM, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	135
6.11	<i>Flow past a hemisphere test case.</i> Stability and accuracy measures for standard G-ROM, L-ROM, and StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	136
6.12	<i>Flow past a hemisphere test case.</i> Energy spectrum at latest time for the reference FOM, standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	136
6.13	<i>Minimal channel flow test case.</i> Kinetic energy variation in time for the FOM reference, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	138
6.14	<i>Minimal channel flow test case.</i> Mean squared error of kinetic energy with respect to FOM, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	139

6.15	<i>Minimal channel flow test case.</i> Energy spectrum at latest time for the reference FOM, standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	139
6.16	<i>Minimal channel flow test case.</i> Stability and accuracy measures for standard G-ROM, L-ROM, and StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).	139
6.17	CPU time of performing the offline training (left) and online simulation (right) of G-ROM, L-ROM, and all StabOp-L-ROMs.	140
6.18	<i>Lid-driven cavity test case.</i> Velocity fields obtained from the standard G-ROM, and applying the DF, projection, and the StabOp operators to one G-ROM solution.	142
6.19	<i>Lid-driven cavity test case.</i> Reduced coefficients' values at different r_u , for the standard G-ROM, and after applying the DF and the StabOp operators to it.	142
6.20	<i>Flow past a hemisphere test case.</i> Velocity fields obtained from the standard G-ROM, and applying the DF, projection, and the StabOp operators to one G-ROM solution.	143
6.21	<i>Flow past a hemisphere test case.</i> Reduced coefficients' values at different r_u , for the standard G-ROM, and after applying the DF and the StabOp operators to it.	143
6.22	<i>Minimal channel flow test case.</i> Velocity fields obtained from the standard G-ROM, and applying the DF, projection, and the StabOp operators to one G-ROM solution.	143
6.23	<i>Minimal channel flow test case.</i> Reduced coefficients' values at different r_u , for the standard G-ROM, and after applying the DF and the StabOp operators to it.	144
7.1	Sketch of the pipeline followed to build the DD-EV-ROM model.	147
7.2	Architecture of the classical DeepONet [210].	150
7.3	Architecture of the turbulence map $\mathcal{G}$, adopted in DD-EV-ROM.	150
7.4	Architecture of the correction map $\mathcal{M}$, adopted in DD-EV-ROM.	151
7.5	<i>Flow past a cylinder test case.</i> Domain and mesh considered.	154
7.6	<i>Flow past a cylinder test case.</i> Train and test parameters for the ROMs, and cumulative eigenvalues obtained from the POD.	155
7.7	<i>Flow past a cylinder test case.</i> Gain in relative error at different latent dimensions and for all the fields considered.	157
7.8	<i>Flow past a cylinder test case.</i> Eddy viscosity approximation obtained from EV-ROM, DD-EV-ROM, DD-EV-ROM*, and projection at final online time instance ($t = 8$) and $\nu = 1.15 \times 10^{-4} \text{ m}^2 \text{ s}^{-1}$, compared with the FOM ground truth. For the ROM the modal regime is $(r_u, r_p, r_{v_i}) = (10, 24, 30)$	158
7.9	<i>Channel-driven cavity flow test case.</i> Domain and mesh considered.	158

7.10	<i>Channel-driven cavity flow test case.</i> Train and test parameters for the ROMs, and cumulative eigenvalues obtained from the POD.	159
7.11	<i>Channel-driven cavity flow test case.</i> Gain in relative error at different latent dimensions and for all the fields considered.	160
7.12	<i>Channel-driven cavity flow test case.</i> Absolute error in the pressure approximation obtained from EV-ROM, DD-EV-ROM, DD-EV-ROM*, and projection at final online time instance ($t = 15$), at $\nu = 7 \times 10^{-6} \text{ m}^2 \text{ s}^{-1}$. For the ROM the modal regime is $(r_u, r_p, r_{v_t}) = (2, 5, 1)$	161
7.13	<i>Backward-facing step test case.</i> Parametrized domain and mesh considered. .	161
7.14	<i>Backward-facing step test case.</i> Train and test parameters for the ROMs, and cumulative eigenvalues obtained from the POD.	162
7.15	<i>Backward-facing step test case.</i> Gain in relative error at different latent dimensions and for all the fields considered.	164
7.16	<i>Backward-facing step test case.</i> Eddy viscosity approximation obtained from EV-ROM, DD-EV-ROM, DD-EV-ROM*, and projection compared with the FOM ground truth, at the test parameters $(\alpha, h_1, h_2) = (6.3^\circ, 0.99 \text{ m}, 1.58 \text{ m})$. For the ROM the modal regime is $(r_u, r_p, r_{v_t}) = (3, 4, 12)$	165
8.1	Graphical abstract to build a corrected ROM, which enhances the standard POD-based nonintrusive ROM with a data-driven machine-learning-based quadratic correction.	171
8.2	Schematic representation of the architecture adopted for QuadNet.	176
8.3	Schematic representation of the architecture adopted for QuadNet- μ	177
8.4	Domain and mesh considered in both the test cases.	178
8.5	POD singular values decay in the full data regime, for both the test cases considered. The values are normalized to the first (largest) singular value. . .	181
8.6	<i>Backward-facing step test case.</i> Velocity FOM solution, and errors of POD-RBF, Quad-LS and QuadNet- μ predictions with respect to the FOM, at an unseen parameter.	181
8.7	<i>Lid-driven cavity test case.</i> Velocity FOM solution, and errors of POD-RBF, Quad-LS and QuadNet- μ predictions with respect to the FOM, at an unseen parameter.	182
8.8	<i>Backward-facing step test case.</i> Comparison between samples of points drawn from different distribution. The purple points are sampled from a Boltzmann-like distribution, while the black crosses are sampled from a uniform distribution. A correction field is plotted in the background for reference.	184
8.9	Boxplots comparing the accuracy of the models considered in both the test cases. The label <i>Partial data</i> ($x\%$) refers to the results obtained using QuadNet- μ with the $x\%$ training snapshots.	184

8.10	Comparison of QuadNet- μ errors with POD-RBF and Quad-LS. The left half of the circles shows the relative difference in error between QuadNet- μ and POD-RBF, while the right half shows the difference between QuadNet- μ and Quad-LS.	185
9.1	An illustrative scheme of the most common nonlinear model order reduction methodologies.	192
9.2	Examples of direct and transposed discrete convolution operations.	195
9.3	Sketch of the continuous convolutional operation [73]: the unstructured points are mapped into the filter domain $\mathbb{X} \times \mathbb{Y} = [0, 1]^2$, then a MLP kernel $\mathcal{K}$ is used to approximate the filter values, and finally a convolution between the mapped input values and the filter values is computed.	197
10.1	Schematic illustration of the proposed CE-AE approach, where a convolutional neural network is used to compress the full-order discrete operators. . . .	201
10.2	Sketch on the operator compression strategy using the continuous convolutional architecture, coupled with a standard convolutional network.	205
10.3	<i>Poisson test case.</i> Train and test parameters (left), and the POD singular values decay (right).	207
10.4	<i>Poisson test case.</i> Full-order matrix $\mathbf{A}$	207
10.5	<i>Poisson test case.</i> Loss trend across epochs for a standard AE, and for the novel models CE-AE and S-CE-AE, at different latent dimensions, namely $r = 2, 3$, and 4.	208
10.6	<i>Poisson test case.</i> Reduced matrices $\mathbf{A}_r$, normalized in range $[0, 1]$. The matrices are obtained by compressing the full-order discrete matrix $\mathbf{A}$ with the convolutional encoder CE and with its symmetric version S-CE, at different latent dimensions $r = 2, 3$, and 4.	209
10.7	<i>Poisson test case.</i> Relative L^2 test errors of different approximations with respect to the FOM reference, at the latent dimensions $r = 2, 3$, and 4.	210
10.8	<i>Poisson test case.</i> Approximated solutions for POD-RBF, AE-RBF, CE-AE-RBF, and S-CE-AE-RBF, compared with the FOM reference, at latent dimension $r = 2$, and considering three test parameters.	211
10.9	<i>Advection-diffusion test case.</i> Full-order matrices $\mathbf{A}_1$ and $\mathbf{A}_2$	212
10.10	<i>Advection-diffusion test case.</i> Train, validation (for NNs), and test parameters (left), and the POD singular values decay (right).	213
10.11	<i>Advection-diffusion test case.</i> Loss trend across epochs for a standard AE, and for the novel model CE-AE, at different latent dimensions, namely $r = 2, 3$, and 4.	214

10.12	<i>Advection-diffusion test case.</i> Reduced matrices $\mathbf{A}_{r1}$ and $\mathbf{A}_{r2}$, normalized in range $[0, 1]$. The matrices are obtained by compressing the full-order discrete matrix $\mathbf{A}_1$ and $\mathbf{A}_2$ with the convolutional encoder CE, at different latent dimensions $r = 2, 3$, and 4.	215
10.13	<i>Advection-diffusion test case.</i> Relative L^2 test errors of different approximations with respect to the FOM reference, at the latent dimensions $r = 2, 3$, and 4.	215
10.14	<i>Advection-dominated test case.</i> Approximated solutions for POD-RBF, AE-RBF, and CE-AE, compared with the FOM reference, at latent dimension $r = 2$, and considering three test parameters.	215
10.15	<i>Burgers test case.</i> Domain and computational grid adopted, with boundaries' notation.	216
10.16	<i>Burgers test case.</i> The POD singular values decay.	217
10.17	<i>Burgers test case.</i> Loss trend across epochs for a standard AE, and for the novel model CCE-AE, at different latent dimensions, namely $r = 2$ and 3.	220
10.18	<i>Burgers test case.</i> Time variation of the relative L^2 errors with respect to the FOM solution, obtained using standard methodologies (POD, POD-RBF, POD-G, AE, AE-RBF), and the novel CCE-AE-RBF approach.	220
10.19	<i>Burgers test case.</i> Solution fields for the x (top) and y (bottom) components, found using POD-RBF, AE-RBF, CCE-AE-RBF, compared with the FOM. The fields are shown at different time steps in a predictive regime, at the fixed latent dimension $r = 3$	222

LIST OF TABLES

1.1	Transfer functions of some explicit filters.	12
1.2	Properties of the grids considered in the HIT case.	18
2.1	<i>Flow past a cylinder test case.</i> Degrees of freedom (dofs) for the velocity and pressure fields, and the wall time needed to perform 4 seconds of simulation on the marginally-resolved and well-resolved grids.	25
2.2	Acronyms and main features of the algorithms Opt-EFR. Gray cells indicate that the algorithm does not present that feature.	28
3.1	Degrees of freedom (dofs) for the velocity and pressure fields and wall times needed to perform 4 seconds of simulation, for both the test cases taken into account, and for both the marginally-resolved and well-resolved grids. . .	45
3.2	Acronyms and main features of the proposed RL-EF approaches.	54
3.3	Computational training times for RL-EF approaches.	68
3.4	Hyperparameters used for RL training in RL-EF approaches.	69
4.1	Data regimes considered to compute the filter matrix for DD-EF simulations. Data refers to the decaying HIT simulation at $Re = 40\,000$	78
4.2	Optimized parameters in state-of-the-art approaches methodologies, used for comparison with DD-EF and DD-EFR.	83
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.	92
5.1	Comparison of resolution regimes in high-fidelity simulations (FOM) and POD-based ROMs.	107
6.1	<i>Flow past a cylinder test case.</i> Hyperparameters for the StabOp-L-ROMs (linear, quadratic, and fully nonlinear). Columns labeled NN correspond to hyperparameters specific to the neural-network model, namely the activation function.	128
6.2	<i>Lid-driven cavity test case.</i> Hyperparameters for the StabOp-L-ROMs (linear, quadratic, and fully nonlinear). Columns labeled NN correspond to hyperparameters specific to the neural-network model, namely the activation function.	131

6.3	<i>Flow past a hemisphere test case.</i> Hyperparameters for the StabOp-L-ROMs (linear, quadratic, and fully nonlinear). Columns labeled NN correspond to hyperparameters specific to the neural-network model, namely the activation function.	134
6.4	<i>Minimal channel flow test case.</i> Hyperparameters for the StabOp-L-ROMs (linear, quadratic, and fully nonlinear). Columns labeled linear, quad, and NN correspond to hyperparameters specific to the linear, quadratic or neural-network model, respectively.	137
7.1	<i>Flow past a cylinder test case.</i> Regimes considered for ROMs for the velocity, pressure and eddy viscosity fields.	155
7.2	<i>Channel-driven cavity flow test case.</i> Regimes considered for ROMs for the velocity, pressure and eddy viscosity fields.	159
7.3	<i>Backward-facing step test case.</i> Regimes considered for ROMs for the velocity, pressure and eddy viscosity fields.	163
7.4	Wall times needed to perform the offline and online stages of the ROM pipeline, in the three cases considered. One-simulation FOM and ROM times are intended for 1 s of simulation in the unsteady test cases, and for the steady simulation in the backstep test case. For neural networks' training and ROMs, we report minimum and maximum wall times indicated as $t_{\min} - t_{\max}$. Indeed, in such cases the wall time depends on the reduced dimensions considered, namely on the combination (r_u, r_p, r_{v_t})	165
7.5	Hyperparameters of DeepONet $\mathcal{G}$ and MIONet $\mathcal{M}$ networks, considered for the turbulence and closure mappings, respectively.	167
8.1	Setup of all the ROMs considered, in different regimes (full, partial, and scarce data). The last column refers to the percentage of spatial coordinates considered for the training of QuadNet- μ	180
8.2	Summary of the network parameters for the QuadNet model.	180
8.3	Summary of the network parameters for the QuadNet- μ model.	180
8.4	<i>Backward-facing step test case.</i> Comparison of models performance in terms of average, median and standard deviation of the velocity relative error.	182
8.5	<i>Lid-driven cavity test case.</i> Comparison of models performance in terms of average, median and standard deviation of the velocity relative error.	183
8.6	Comparison between DD-EV-ROM and Corrected-ROM.	187
10.1	<i>Poisson test case.</i> Architectures of CE architectures, in terms of kernel, stride, padding, and activation function for each layer.	223
10.2	<i>Advection-diffusion test case.</i> Architectures of CE architectures, in terms of kernel, stride, padding, and activation function for each layer.	223
10.3	<i>Burgers test case.</i> Architectures of CCE architectures, in terms of kernel, stride, padding, and activation function for each layer.	224

10.4	Training times, number of parameters, and accuracy in terms of average test error, of the AE, CE-AE, S-CE-AE (for the second test case), and CCE-AE models. The test errors are computed by interpolating the latent variables with an RBF approach.	224
11	Summary of the main methodologies developed in this thesis, including their main advantages and limitations, and the scenarios in which each approach is most suitable.	230

LIST OF ALGORITHMS

1	Pseudocode for algorithms Opt-EFR optimizing both parameters. Row 9 (underlined) is considered only for algorithms of type $\text{Opt-EFR}_{\text{global}}^{u,\nabla u,p}$. The optimization part is coloured in orange.	29
2	Training of $\text{StabOp } \mathcal{S}(\mathbf{a}; \boldsymbol{\theta})$	125
3	Finding optimal filter radius δ for L-ROM.	126
4	Pseudo-code for the online stage of DD-EV-ROM in the steady case. The part highlighted in orange corresponds to the test case with geometrical parametrization.	148
5	Pseudo-code for the online stage of algorithms DD-EV-ROM in the unsteady case with only physical parametrization.	149
6	Pseudo-code for the offline training of the CCE-AE-RBF approach. . .	218
7	Pseudo-code for the online prediction of the CCE-AE-RBF approach. .	219

ACRONYMS

χ -Opt-EFR	EFR with optimized relax parameter
δ -Opt-EF	EF with optimized filter width
$\delta\chi$ -Opt-EFR	EFR with optimized filter and relax parameters
$\mathcal{E}\mathcal{Z}$ -DD-EFR	Energy- and Enstrophy-preserving Data-Driven Evolve-Filter-Relax
$\mathcal{E}$ -DD-EFR	Energy-preserving Data-Driven Evolve-Filter-Relax
AE	AutoEncoder
CAE	Convolutional AutoEncoder
CCAe	Continuous Convolutional AutoEncoder
CCE-AE	Continuous Convolutional Encoder-AutoEncoder
CCNN	Continuous Convolutional Neural Network
CE	Convolutional Encoder
CE-AE	Convolutional Encoder-AutoEncoder
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DD-EF	Data-Driven Evolve-Filter
DD-EV-ROM	Data-Driven Eddy-Viscosity Reduced Order Method
DD-EV-ROM*	Data-Driven Eddy-Viscosity Reduced Order Method with coupled training
DD-RL-EF	Data-Driven Reinforcement Learning-based Evolve-Filter
DD-RL-EF-rand	Data-Driven Reinforcement Learning-based Evolve-Filter with random initialization
DeepONet	Deep Operator Network
DF	differential filter
DF-RL-EF	Data-Free Reinforcement Learning-based Evolve-Filter
DMD	Dynamic Mode Decomposition
DNS	Direct Numerical Simulation
DQN	Deep Q-Network
EF	Evolve-Filter
EFR	Evolve-Filter-Relax

EIM	Empirical Interpolation Method
EV-ROM	Eddy Viscosity-Reduced Order Method
FEM	Finite Element Method
FFT	Fast Fourier Transform
FOM	Full Order Model
FVM	Finite Volume Method
G-ROM	Galerkin-Reduced Order Method
GPR	Gaussian Process Regression
HIT	Homogeneous Isotropic Turbulence
L-ROM	Leray-Reduced Order Method
LES	Large Eddy Simulation
MIONet	Multi-Input Operator Network
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NN	Neural Network
NSE	Navier–Stokes Equations
Opt-EFR	Optimized Evolve-Filter-Relax
PDEs	Partial Differential Equations
PINN	Physics-Informed Neural Network
POD	Proper Orthogonal Decomposition
PPE	Pressure Poisson Equation
QoI	Quantity of Interest
Quad-LS	Quadratic Correction via Least-Squares
QuadNet	Quadratic Correction via DeepONet
QuadNet-μ	Quadratic Correction via MIONet
RANS	Reynolds–Averaged Navier–Stokes
RBF	Radial Basis Function
RL	Reinforcement Learning
RL-EF	Reinforcement Learning-based Evolve-Filter
ROM	Reduced Order Method

S-CE	Symmetric positive definite Convolutional Encoder
S-CE-AE	Symmetric positive definite Convolutional Encoder-AutoEncoder
SEM	Spectral Element Method
SGS	Sub-Grid Scales
SLSQP	Sequential Least Squares Programming
SP-DF-RL-EF	Structure-Preserving Data-Free Reinforcement Learning-based Evolve-Filter
StabOp-L-ROM	Stabilization Operator strategies for Leray-Reduced Order Method
URANS	Unsteady Reynolds-Averaged Navier-Stokes
VMS	Variational Multiscale

INTRODUCTION

Many real-world phenomena can be mathematically described by means of **Partial Differential Equations (PDEs)**. In this thesis, we focus on the mathematical modeling of fluid flows, whose dynamics are governed by the **Navier–Stokes Equations (NSE)**. These equations, first introduced in the nineteenth century [233, 317], describe the evolution of the velocity and pressure fields in a fluid and constitute the fundamental model of fluid dynamics.

Despite nearly two centuries of research devoted to their analysis, the **NSE** remain only partially understood. In particular, the question of global existence and smoothness of solutions in three dimensions is still open and constitutes one of the Millennium Prize Problems. In the field of **Computational Fluid Dynamics (CFD)**, and, in particular, in the study of incompressible fluids, the main difficulties associated with the **NSE** arise from the nonlinear convective term, which leads to highly complex dynamical behaviour [52]. In several relevant applications, including the aerodynamic flows around aircraft wings, atmospheric and oceanic circulation, and industrial fluid processes, these equations give rise to the phenomenon of **turbulence**. The dimensionless Reynolds number Re , introduced by Osborne Reynolds in his investigation of water flow in pipes [272], quantifies the ratio of inertial to viscous forces and serves as a key indicator of the transition from laminar to turbulent regimes.

Turbulence is a phenomenon characterized by irregular and chaotic motion across a wide range of interacting three-dimensional spatial scales. Accurately resolving all these scales requires extremely fine spatial and temporal discretizations. **Direct Numerical Simulation (DNS)** aims to resolve the full range of turbulent scales without introducing modeling assumptions, but its computational cost rapidly becomes prohibitive as the Reynolds number increases. This leads to a complexity estimate of $O(Re^3)$, making **DNS** impractical for realistic applications. To overcome these limitations, several modeling strategies have been developed, including **Large Eddy Simulation (LES)** and **Reynolds–Averaged Navier–Stokes (RANS)** equations [256, 352, 91, 218, 284].

On the one hand, the **RANS** equations, and their unsteady counterpart **Unsteady Reynolds–Averaged Navier–Stokes (URANS)**, provide a description of the flow in terms of Reynolds-averaged quantities, solving for the mean velocity and pressure fields while modeling the effects of turbulent fluctuations through turbulence closure models. This approach drastically reduces the computational cost, making it suitable for engineering applications, but at the expense of losing detailed information about instantaneous turbulent structures.

On the other hand, **LES** reduces the complexity by separating different spatial scales. This separation is obtained by applying a spatial filtering operation to the **NSE**, which decomposes the flow field into resolved large energy-containing scales and unresolved small scales, named **Sub-Grid Scales (SGS)**. **LES** acts as a middle ground between **DNS** and **RANS**, offering high accuracy but still requiring high cost in many practical applications.

Many different modeling assumptions have been introduced in the past decades to model turbulence both for **LES** and **RANS** approaches. One of these theories is the introduction of a fictitious turbulent viscosity named *eddy viscosity*, first introduced in [40], which mimics the dissipative effect of unresolved turbulent fluctuations. A classic example is the Smagorinsky model [312], which is an implicit filtering strategy originally developed in the **LES** context.

Besides **LES** and **RANS**, an alternative approach is to reduce the computational cost of **DNS** by exploiting *marginally-resolved* grids, i.e., grids that capture the largest energetic structures but may produce numerical instabilities. This phenomenon may appear in standard discretization methods, like **Finite Element Method (FEM)**, **Finite Volume Method (FVM)**, or **Spectral Element Method (SEM)**. In this case, the problem becomes of a different nature: it is not the modeling of unresolved physical scales, as in **LES**, but the control of numerical instabilities arising from under-resolution. These instabilities often manifest as spurious oscillations or unphysical vortical structures, which can compromise the accuracy and stability of the solution.¹ Spatial filtering strategies can alleviate or even suppress these numerical issues, selectively damping the small-scale energy content while preserving the physically relevant flow features. However, standard spatial filters, like the differential elliptic filter [107, 108], may introduce excessive numerical diffusion, reducing the accuracy of the resolved large-scale structures, and their effectiveness often depends on the choice of the filter type and on the filter width [190]. Moreover, in many practical simulations, the optimal filtering parameters are problem-dependent and not straightforward to determine, which can limit the robustness of the approach.

These challenges define the first step of the methodological framework developed in this thesis, whose pipeline is divided into three parts and represented in Figure 1.

In particular, the limitations of the state-of-the-art spatial filters motivate the development of **adaptive and data-driven filtering strategies**, which constitute Part I of the thesis. Part I is divided as follows:

- ▶ Chapter 1 reviews the literature on filtering strategies, and introduces strengths and weaknesses of the most used spatial filtering approaches, and in particular of the **Evolve-Filter-Relax (EFR)** approach.
- ▶ Chapter 2 describes how the **EFR** parameters can be optimized in a data-driven way [147]. This is a joint work in collaboration with M. Strazzullo, M. Girfoglio, T.

¹Although energy-stable or dissipative schemes can prevent the onset of numerical blow-up, marginally-resolved simulations may still suffer from inaccurate solutions and spurious oscillations.

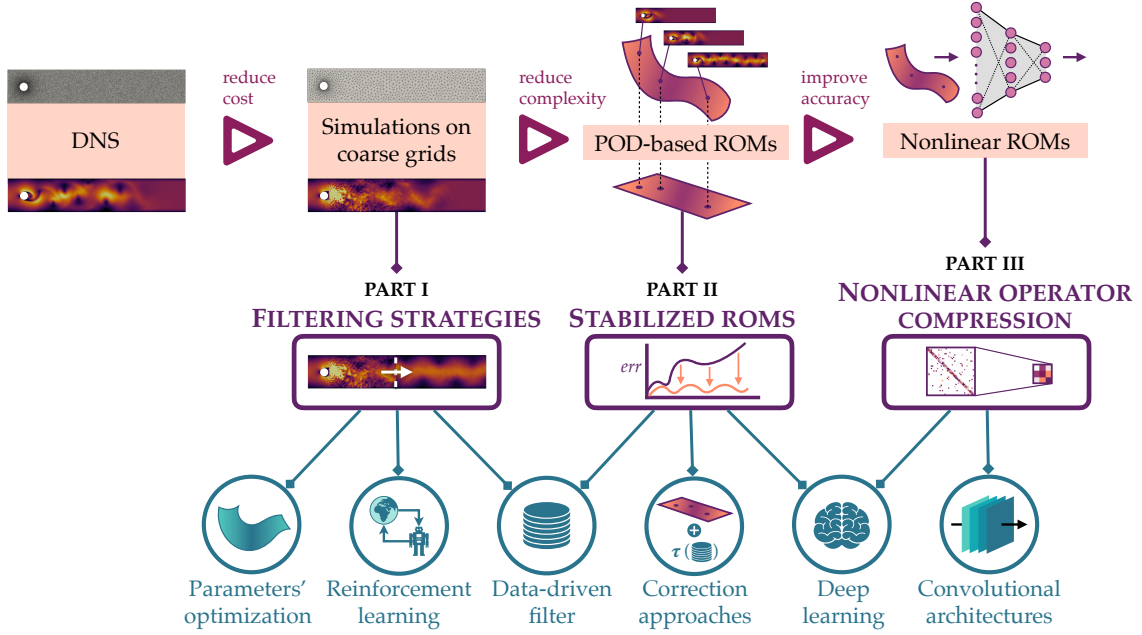


Figure 1: Illustrative sketch of the pipeline followed in the thesis, and related methodologies.

Iliescu, and G. Rozza. Part of this study was held during my one-month secondment at Virginia Tech in October-November 2023.

- Chapter 3 introduces an adaptive way to optimize the filter radius via reinforcement learning, with and without making use of reference data [148]. This work is in collaboration with M. Strazzullo and G. Rozza, and was partially developed during my two-week visiting period at Politecnico di Torino in October 2025.
- Chapter 4 presents a different and totally data-driven filtering operation within the EFR framework [150]. This project is in collaboration with T. V. Gastelen, S. D. Agdestein, B. Sanderse, G. Stabile and G. Rozza. This study has been conducted during my three-month internship at CWI in Amsterdam.

While marginally-resolved simulations reduce the computational complexity compared to DNS, many practical applications, such as design optimization, control, or parameter studies, may require real-time or near-real-time predictions. In these scenarios, a **Reduced Order Method (ROM)** provides an additional level of simplification by projecting the high-dimensional system, typically called **Full Order Model (FOM)**, onto a low-dimensional subspace. This type of model captures the dominant flow features while drastically lowering the computational cost. ROMs are typically constructed using data-driven techniques, such as **Proper Orthogonal Decomposition (POD)**, which extract the most energetic modes from some pre-collected high-fidelity solutions [212, 311]. Once the reduced basis is identified, the governing equations can be projected onto this low-dimensional space, leading to a system of ordinary differential equations that can be solved orders of

magnitude faster than the original **FOM**. This is the logic behind *equation-based* or *intrusive ROMs*.

For convection-dominated flows or flows with complex nonlinear interactions, classical projection-based **ROMs**, like **POD-Galerkin ROMs**, may suffer from numerical instabilities and accuracy loss, especially when considering a marginally-resolved regime. As at the **FOM** level, simulations on marginally-resolved grids may lead to instabilities, also at the **ROM** level, the selected modes may not fully capture small-scale dynamics. These issues can be addressed, just as in the **FOM**, using closure or filtering approaches, like eddy-viscosity modeling [234] or Leray-type regularizations [198]. These types of techniques might be employed in **ROMs** with different scopes. First, the introduction of turbulence models, like eddy viscosity models, ensures consistency with the **FOM** counterpart and acts as a closure also at the reduced level, improving the accuracy of the approximation. Second, a well-known approach in the **ROM** literature is to reintegrate the contribution of the discarded basis by adding some extra data-driven correction terms into the reduced system, in a **Variational Multiscale (VMS)** fashion. In this case, the correction aims to *correct* the existing approximation using data, and does not constitute a closure in its classical sense. Finally, if we consider the class of **ROM** spatial filters, the goal is to smooth out the oscillations and stabilize the simulation, exactly as for the high-fidelity counterpart.

Part II of the thesis describes three ways to improve these techniques in **POD-based ROMs** by means of **data-driven and machine-learning-based stabilization**. More in detail:

- ▶ Chapter 5 presents the basic notions and literature review on **POD-Galerkin ROMs** and associated closure and filtering approaches.
- ▶ Chapter 6 introduces a data-driven stabilization operator for Leray-**ROMs** [336], developed in collaboration with P. H. Tsai, A. Quaini, T. Iliescu, and G. Rozza.
- ▶ Chapter 7 presents a correction-based approach for parametrized **ROMs** via neural networks [146, 145]. These works are done in cooperation with G. Stabile and G. Rozza.
- ▶ Chapter 8 extends the concept of “**POD** correction” to nonintrusive and totally data-driven **ROMs** [68]. This work was held in collaboration with G. Codega, N. Demo and G. Rozza.

The use of linear techniques like **POD**, poses a limit on the **ROM** accuracy, particularly when the flow dynamics exhibit strong nonlinearities or are dominated by transport phenomena. A wide class of research focuses on integrating nonlinear techniques into linear **ROMs** (like machine learning-based correction terms, as in Part II). However, these approaches are always limited by the linear projection step. To overcome this limitation, recent research has explored fully nonlinear dimensionality reduction, where the solution manifold is represented through a nonlinear mapping. In this context, the **AutoEncoder**

(AE) [132, 193], thanks to its inherent architecture, has emerged as a powerful tool to learn a low-dimensional latent representation. Once such a nonlinear representation is available, the reduced modeling problem shifts from approximating the dynamics in the original space to modeling the evolution of the latent variables. Consequently, a significant portion of current research is devoted to identifying suitable operators or surrogate models that evolve the latent state in time or with respect to the parameters of interest. Part III of the thesis represents a step towards a **fully nonlinear and equation-based reduction** approach. In particular, we propose a nonlinear compression technique via **Convolutional Neural Network (CNN)**, for the operators of the high-fidelity model. Our goal is to derive a reduced dynamical system while retaining an explicit connection with the governing equations. More in detail, Part III is divided as follows:

- Chapter 9 presents a literature review on nonlinear approaches and CNNs for ROMs.
- Chapter 10 introduces a novel approach for nonlinear operator compression in PDEs [144], developed in collaboration with G. Stabile and G. Rozza.

The thesis concludes with a comprehensive discussion of the results, the main conclusions, and directions for future research.

This thesis represents a methodological research journey, focused on the development and assessment of numerical and reduced-order strategies to improve the stability of fluid flow simulations. While the work is framed around general methodological developments, all approaches are tested and validated on a range of two- and three-dimensional benchmark problems, spanning from simple convection-dominated flows to more complex geometries. The overarching goal is to leverage both data and physical knowledge of the system to design filtering and closure mechanisms that enhance predictive capabilities.

Software tools The numerical results presented in this thesis have been obtained thanks to different open-source packages: (i) OpenFOAM [240, 153], FEniCS [9, 206, 205, 207], FEniCS-X [21, 10, 303, 304], Nek5000 [93, 92], and IncompressibleNavierStokes.jl [2, 3] for high-fidelity simulations; (ii) ITHACA-FV [313, 314, 138] for intrusive ROMs built from OpenFOAM, NekROM [161] for intrusive ROMs built from NekRS [93, 92], EZyRB [81, 80] for nonintrusive ROMs, PINA [72] for scientific machine learning tools, Smithers [84] for transversal code tools and databases. I warmly acknowledge all the developers and maintainers of the above-mentioned libraries.

Notes on content

- The thesis does not include the work done in collaboration with P. Zappi, N. Tonello, P. Cinnella, and G. Rozza. The paper [149] and its extension [362] combine the potential of different RANS turbulence models with the efficiency of nonintrusive

ROMs via space-dependent aggregation techniques. Since unrelated to the main topics of the thesis, the results have been omitted.

- ▶ Since the three parts of the thesis treat different topics, a more detailed literature review can be found in the first chapter of each part, namely Chapters 1, 5 and 9.
- ▶ The thesis includes orange-coloured boxes like the following:

Proof: ...

for mathematical proofs, and water green-coloured boxes like the following:

Remark: ...

for remarks, regarding, for example, implementation details or notes on notation. These boxes are optional and may be skipped by the reader without losing any essential information.

PART I

DATA-DRIVEN AND ADAPTIVE FILTERING IN MARGINALLY-RESOLVED SIMULATIONS

FILTERING APPROACHES: LITERATURE REVIEW

This chapter reviews the concept of filtering in turbulent flow simulations and presents a far-from-exhaustive list of existing filtering strategies.

1.1 The Navier–Stokes Equations and turbulence

To simulate an incompressible fluid flow in a d -dimensional physical domain $\Omega \subset \mathbb{R}^d$ (with $d = 2$ or 3) and in a time window $t \in [t_0, T]$, we rely on the incompressible **NSE**, which are composed of the momentum conservation equation and the mass conservation (or continuity equation). If we denote with $\mathbf{u} = \mathbf{u}(x, t)$ and $p = p(x, t)$ the velocity and pressure fields, respectively, the incompressible **NSE** read as follows [233, 317].

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \Delta \mathbf{u} - \nabla p + \mathbf{f} & \text{in } \Omega \times (t_0, T], \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega \times (t_0, T]. \end{cases} \quad (1.1)$$

In the above expression¹, ν is the kinematic viscosity, and $\mathbf{f}$ represents external forces, like gravitational or electromagnetic effects. It is important to highlight that p refers to the fluid pressure field divided by the fluid density ρ . Moreover, $\mathbf{u} \in \mathbb{U}$ and $p \in \mathbb{Q}$, where $\mathbb{U}$ and $\mathbb{Q}$ are suitable Hilbert function spaces. The Equations (1.1) are subject to the initial conditions $\mathbf{u} = \mathbf{u}_0$, $p = p_0$ in $\Omega \times \{t_0\}$, and to specific boundary conditions. We can write:

$$\begin{cases} \mathbf{u} = \mathbf{u}_D & \text{on } \Gamma_D \times (t_0, T], \\ -p\mathbf{n} + \nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} = \mathbf{0} & \text{on } \Gamma_N \times (t_0, T], \end{cases} \quad (1.2)$$

where $\mathbf{n}$ is the normal unitary vector, Γ_D and Γ_N are the Dirichlet and Neumann boundaries, with $\Gamma_D \cup \Gamma_N = \partial\Omega$, and $\Gamma_D \cap \Gamma_N = \emptyset$.

The complexity of the **NSE** is largely characterized by the Reynolds number $Re = \frac{UL}{\nu}$, where L and U are the characteristic length and velocity scales, representing the largest flow structures. When $Re \gg 1$, the nonlinear term $(\mathbf{u} \cdot \nabla) \mathbf{u}$ dominates over the viscous term $\nu \Delta \mathbf{u}$. For large Re , namely in turbulent flows, the solutions of the **NSE** exhibit the so-called **energy cascade** [273]. We start by defining the kinetic energy as:

$$\mathcal{E}(t) = \frac{1}{2|\Omega|} \int_{\Omega} \mathbf{u}(x, t) \cdot \mathbf{u}(x, t) dx. \quad (1.3)$$

¹We specify that in system (1.1), “ ∇ ” is a vector field of components $\left[\frac{\partial}{\partial x} \hat{\mathbf{i}}, \frac{\partial}{\partial y} \hat{\mathbf{j}} \right]$ in 2D ($d = 2$) and $\left[\frac{\partial}{\partial x} \hat{\mathbf{i}}, \frac{\partial}{\partial y} \hat{\mathbf{j}}, \frac{\partial}{\partial z} \hat{\mathbf{k}} \right]$ in 3D ($d = 3$), where $\hat{\mathbf{i}}, \hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ are the unitary vectors in the x, y , and z directions, respectively. Alternatively the term $(\mathbf{u} \cdot \nabla) \mathbf{u}$ can be written as $\nabla \cdot (\mathbf{u} \otimes \mathbf{u})$, as will be recalled in Section 5.1.

If we analyze the time variation of the energy for the **NSE** in a periodic domain and in the absence of external forces, we obtain:

$$\frac{d\mathcal{E}}{dt} \propto \int_{\Omega} \mathbf{u} \cdot \frac{\partial \mathbf{u}}{\partial t} d\mathbf{x} = - \int_{\Omega} \mathbf{u} \cdot (\mathbf{u} \cdot \nabla) \mathbf{u} d\mathbf{x} - \int_{\Omega} \mathbf{u} \cdot \nabla p d\mathbf{x} + \nu \int_{\Omega} \mathbf{u} \cdot \Delta \mathbf{u} d\mathbf{x}.$$

The nonlinear term can be written as $\frac{1}{2} \int_{\Omega} \nabla \cdot (|\mathbf{u}|^2 \mathbf{u}) d\mathbf{x}$, which is zero for a periodic domain. The pressure term is decomposed using integration by parts and vanishes under the divergence-free constraint. The only remaining term is the viscous term, leading to:

$$\frac{d\mathcal{E}}{dt} \propto -\nu \int_{\Omega} |\nabla \mathbf{u}|^2 d\mathbf{x} \leq 0. \quad (1.4)$$

The above result means that **the energy is always dissipated** in the absence of external forces, and under periodic boundary conditions. In particular:

- ▶ The energy is transferred by the source forcing term to the large scales of the flow.
- ▶ There exists a medium-scale range where the nonlinear term drives the energy into smaller eddies, and the energy is conserved, as the viscous term is negligible. According to the K41 theory, developed by Kolmogorov in [176, 174], in the medium range (in which energy is not dissipated), named **inertial range**, the energy $\mathcal{E}$ follows the Kolmogorov law:

$$\mathcal{E}(\kappa) = C_K \epsilon^{2/3} \kappa^a, \quad (1.5)$$

where $C_K \in [1.4, 1.7]$ is the Kolmogorov constant, ϵ is the flow dissipation rate, κ is the wavenumber, and $a = -5/3$ in three dimensional turbulence.

- ▶ At the smallest scales, energy decays exponentially, as dissipation is not negligible.

At this point, the natural question is:

What are the maximum and minimum wavenumbers (or length scales) of the inertial range, i.e., for which the Kolmogorov law is applied?

The maximum scale coincides with the large-scale characteristic length L , while the minimum scale is the so-called Kolmogorov length scale (or micro-scale) η . From statistical energy equilibrium, it can be proved that the Kolmogorov length scale is:

$$\eta = Re^{-3/4} L.$$

This result also determines the highest wavenumber in the inertial range ($1/\eta$).

To investigate how the turbulent kinetic energy is distributed among the different flow scales, the energy spectrum is commonly analyzed. The Fourier transform provides a representation of the velocity field in the wavenumber space, where large wavenumbers correspond to small spatial structures and low wavenumbers to large-scale motions. The energy spectrum quantifies the contribution of each wavenumber to the total turbulent kinetic energy. The energy spectrum for three-dimensional turbulence is qualitatively represented in Figure 1.1a.

On the other hand, when dealing with two-dimensional turbulent flows, the dynamics and interaction among different scales are substantially different. In this case, it is essential to introduce a second metric, the **enstrophy**:

$$\mathcal{E}(t) = \frac{1}{2|\Omega|} \int_{\Omega} |\omega(x, t)|^2 dx, \quad (1.6)$$

where $\omega(x, t) = \nabla \times \mathbf{u}(x, t)$ is the **vorticity**. The enstrophy quantifies the strength of rotational motion in a fluid, and has a profound effect on the inertial energy transfer in 2D isotropic turbulence [177].

In two-dimensional turbulence, the dynamics are constrained by the simultaneous conservation of energy and enstrophy [237]. As first described by Kraichnan in [177], this leads to a dual cascade. On the one hand, energy, which is injected at an input wavenumber κ_f , tends to transfer to larger scales (*inverse energy cascade*) following the $\sim \kappa^{-5/3}$ law, as in 3D turbulence. On the other hand, enstrophy cascades toward smaller scales (*direct enstrophy cascade*), producing the characteristic energy spectra κ^{-3} in the inertial range. For more details on 2D turbulence, we refer the reader to [26, 175, 76, 250, 37]. The energy spectrum for 2D turbulence is graphically summarized in Figure 1.1b.

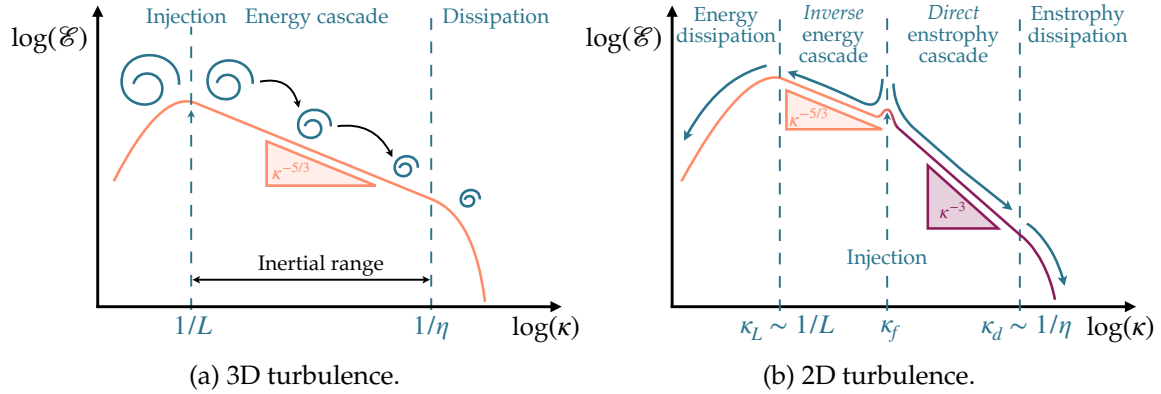


Figure 1.1: Energy spectrum qualitative behaviour in the case of two-dimensional and three-dimensional turbulence.

Since turbulence involves a wide range of interacting spatial scales, resolving all of them is often computationally unfeasible, requiring $O(Re^{9/4})$ mesh points. Filtering techniques are therefore introduced in Section 1.2 to separate the resolved large scales (the filtered velocity) from the smaller unresolved ones, which are either removed or modelled via closure strategies. In this fashion, the velocity field (and every field considered) is interpreted as the sum of a filtered part $\bar{\mathbf{u}}$ and a sub-filtered (unresolved) part $\mathbf{u}'$.

1.2 Filtering strategies for turbulent flow simulations

A filter is an operator $\mathcal{F}_\delta$ which, applied to the velocity, outputs the filtered velocity:

$$\bar{\mathbf{u}} = \mathcal{F}_\delta(\mathbf{u}).$$

In particular, local spatial averaging is a filter, and in this case, the filtering operation can be defined as a **convolution** (or cross-correlation) operation, as follows:

$$\bar{\mathbf{u}} = \mathcal{F}_\delta * \mathbf{u}, \quad (1.7)$$

where $\mathcal{F}_\delta$ denotes the kernel of the convolution [190]. Filter kernels typically depend on the parameter $\delta > 0$, commonly referred to as **filter radius** or **filter width**. This parameter determines the characteristic scale of the filtering operation and controls the amount of smoothing applied to the velocity field. The choice of the filter radius δ is crucial for the success of the method, and typically $\delta \sim O(\eta)$ or $O(h_{\min})$, where $h_{\min}$ is the minimum mesh size [107]. If we consider a smooth function $\mathcal{F}(\mathbf{x})$ with $0 \leq \mathcal{F} \leq 1$, $\mathcal{F}(0) = 1$ and $\int_{\mathbb{R}^d} \mathcal{F} d\mathbf{x} = 1$, the filter kernel is defined as

$$\mathcal{F}_\delta(\mathbf{x}) = \delta^{-d} \mathcal{F}(\mathbf{x}/\delta).$$

We list here two types of local-averaging filters.

- The **top hat filter** (or **box filter**) is fundamentally an unweighted spatial average over the neighbourhood of a given point. In this case:

$$\mathcal{F}_\delta(\mathbf{x}) = \begin{cases} \frac{1}{|\mathcal{B}_\delta|} & \text{for } \mathbf{x} \in \mathcal{B}_\delta, \\ 0 & \text{for } \mathbf{x} \notin \mathcal{B}_\delta, \end{cases} \quad (1.8)$$

where $\mathcal{B}_\delta(\mathbf{x})$ is a ball (or disk in two dimensions) of radius δ centered at $\mathbf{x}$.

- The **Gaussian filter** is a weighted average, decaying exponentially with the distance from a given point. It follows that:

$$\mathcal{F}_\delta(\mathbf{x}) = \delta^{-d} \left(\frac{6}{\pi} \right)^{d/2} \exp(-6|\mathbf{x}/\delta|^2).$$

Another important filter type is the **differential filter (DF)**, proposed by Germano in [107, 108], which reads as:

$$-2\delta^2 \Delta \bar{\mathbf{u}} + \bar{\mathbf{u}} = \mathbf{u}, \quad (1.9)$$

subject to the test case-specific boundary conditions. In this case, the filter is the differential operator² $\mathcal{F}_\delta = (-2\delta^2 \Delta + 1)^{-1}$.

The filtering operation can also be characterized through the transfer function $\widehat{\mathcal{F}}(\kappa)$, which describes how the filter acts across different wavenumbers, i.e., in Fourier space. Using its transfer function, it can be proved that the **DF** always dissipates energy, i.e.:

$$\|\bar{\mathbf{u}}\|_2^2 = \|(-2\delta^2 \Delta + 1)^{-1} \mathbf{u}\|_2^2 \leq \|\mathbf{u}\|_2^2,$$

where the notation $\|\cdot\|_2$ denotes the L^2 norm.

²Some references like [107, 190] report the filter as $\mathcal{F}_\delta = (-\delta^2 \Delta + 1)^{-1}$. We consider the notation $(-2\delta^2 \Delta + 1)^{-1}$, to be consistent with the **EFR** formulation introduced later and used in Chapters 2, 3, and 4.

Proof: the DF always dissipates energy and enstrophy.

We can rewrite the filter operation in the Fourier space as:

$$\widehat{u}_\kappa = \frac{1}{1 + 2\kappa^2\delta^2} \widehat{u}_\kappa,$$

where $\widehat{u}_\kappa$ indicates the Fourier transform of $\mathbf{u}$ at the wavenumber κ . It follows that:

$$\|\overline{\mathbf{u}}\|_2^2 = \sum_\kappa \left| \frac{1}{1 + 2\kappa^2\delta^2} \right|^2 |\widehat{u}_\kappa|^2 \leq \sum_\kappa |\widehat{u}_\kappa|^2 = \|\mathbf{u}\|_2^2 \Rightarrow \boxed{\|\overline{\mathbf{u}}\|_2^2 \leq \|\mathbf{u}\|_2^2}.$$

In Fourier space, the vorticity can be written as $\widehat{\omega}_\kappa \sim i\kappa \widehat{u}_\kappa$, leading to:

$$\|\overline{\omega}\|_2^2 = \sum_\kappa \kappa^2 |\widehat{u}_\kappa|^2 = \sum_\kappa \left| \frac{\kappa}{1 + 2\kappa^2\delta^2} \right|^2 |\widehat{u}_\kappa|^2 \leq \sum_\kappa |\widehat{u}_\kappa|^2 = \|\omega\|_2^2 \Rightarrow \boxed{\|\overline{\omega}\|_2^2 \leq \|\omega\|_2^2}.$$

The transfer functions of the above-mentioned explicit filters can be found in Table 1.1, while a graphical comparison is shown in Figure 1.2. In both the plot and the table, we adopt the notation $\widehat{\mathcal{F}}(\kappa\delta)$ to explicitly highlight the dependence of the transfer function on $\kappa\delta$, without separately considering the effect of the filter width δ . A common feature of these filters is that the lowest wavenumbers, associated with the largest flow scales, are largely unaffected, since the transfer function remains close to 1. Conversely, the filtering mainly acts on the highest wavenumbers, attenuating the energy of the smallest scales.

Table 1.1: Transfer functions of some explicit filters.

Filter	$\widehat{\mathcal{F}}(\kappa\delta)$
Top-hat (1D)	$\frac{\sin(\kappa\delta)}{\kappa\delta}$
Gaussian	$\exp\left(-\frac{\kappa^2\delta^2}{24}\right)$
Differential	$\frac{1}{1+2\kappa^2\delta^2}$

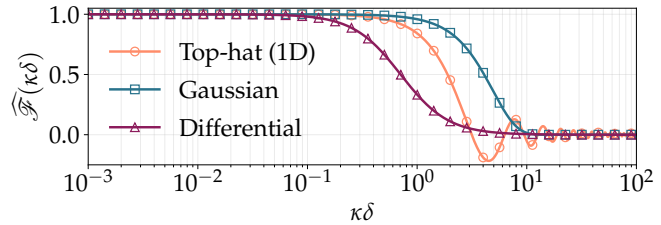


Figure 1.2: Plot of the transfer functions of some explicit filters.

A modification to the DF which preserves the incompressibility constraint is a generalized Stokes problem [86, 188], namely:

$$\begin{cases} -2\delta^2 \Delta \overline{\mathbf{u}} + \overline{\mathbf{u}} + \nabla \pi^{\text{aux}} = \mathbf{u}, \\ \nabla \cdot \overline{\mathbf{u}} = 0, \end{cases} \quad (1.10)$$

where π^{aux} is an auxiliary variable introduced to guarantee the divergence-free constraint for $\overline{\mathbf{u}}$.

Other formulations of the DF, like [189], consider a nonlinear version where an indicator function $a(\mathbf{u})$ is introduced. The filter is written as:

$$-2\delta^2 \nabla \cdot (a(\mathbf{u}) \nabla \overline{\mathbf{u}}) + \overline{\mathbf{u}} = \mathbf{u}, \quad (1.11)$$

where $a(\mathbf{u}) = 1$ leads to the standard **DF** in Equation (1.9). The idea of [189] was to introduce an indicator function that, given the unfiltered velocity field, is able to recognize how much filtering is needed. In particular:

- $a(\mathbf{u}) \in [0, 1]$ for any velocity $\mathbf{u}$,
- $a(\mathbf{u}) \simeq 0$ for laminar regions or persistent flows structures,
- $a(\mathbf{u}) \simeq 1$ for turbulent regions or flow structures which decay rapidly.

This led to the development of several indicator function formulations. A well-known indicator function is:

$$a_S(\mathbf{u}) = \frac{|\nabla \mathbf{u}|}{\|\nabla \mathbf{u}\|_\infty}, \quad (1.12)$$

for which one can recover the Smagorinsky viscosity term [39], briefly recalled in Section 1.2.1.

The above-mentioned filters can be used to construct a **Leray model** [198, 197], which consists of considering the filtered velocity in the nonlinear term of the **NSE**. The classical Leray model is given by:

$$\begin{cases} \frac{\partial \bar{\mathbf{u}}}{\partial t} + (\bar{\mathbf{u}} \cdot \nabla) \bar{\mathbf{u}} = \nu \Delta \bar{\mathbf{u}} - \nabla p + \bar{\mathbf{f}} & \text{in } \Omega \times (t_0, T], \\ \nabla \cdot \bar{\mathbf{u}} = 0 & \text{in } \Omega \times (t_0, T]. \end{cases} \quad (1.13)$$

The filtered velocity $\bar{\mathbf{u}}$ in System (1.13) can be computed selecting, for instance, one of the above-mentioned filters $\mathcal{F}_\delta$, like the **DF**.

1.2.1 The Smagorinsky model as an implicit filter

In the context of **LES**, the spatial filter operation is applied to the entire system of **NSE**, as follows:

$$\begin{cases} \frac{\partial \bar{\mathbf{u}}}{\partial t} + (\bar{\mathbf{u}} \cdot \nabla) \bar{\mathbf{u}} = \nu \Delta \bar{\mathbf{u}} - \nabla \bar{p} - \nabla \cdot \bar{\mathbf{T}}_{\text{SGS}} + \bar{\mathbf{f}} & \text{in } \Omega \times (t_0, T], \\ \nabla \cdot \bar{\mathbf{u}} = 0 & \text{in } \Omega \times (t_0, T]. \end{cases} \quad (1.14)$$

In the above system, the term $\bar{\mathbf{T}}_{\text{SGS}}$ is known as the **SGS** stress tensor, and models the interaction between the resolved (filtered) and unresolved scales. The components of this tensor can be written as $(\bar{\mathbf{T}}_{\text{SGS}})_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j$ [196].

The past decades of research on turbulence were dedicated to properly choose a closure ansatz that models the term $\nabla \cdot \bar{\mathbf{T}}_{\text{SGS}}$. In this context, one popular and widely used approach is the **Smagorinsky model** [312]. This model is based on the concept of eddy viscosity, first introduced in [40]. The eddy viscosity is an artificial and unphysical viscosity, specifically designed to capture turbulence effects. In the Boussinesq hypothesis, the tensor $\bar{\mathbf{T}}_{\text{SGS}}$ is directly related to the turbulent or eddy viscosity ν_t and to the velocity gradients, as follows:

$$(\bar{\mathbf{T}}_{\text{SGS}})_{ij} \sim -2\nu_t \bar{\mathbf{S}}_{ij}, \text{ with } \bar{\mathbf{S}}_{ij} = \frac{1}{2} \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right).$$

In Smagorinsky's theory:

$$\nu_t = (C_S \delta)^2 |\bar{\mathbf{S}}|, \quad (1.15)$$

where C_S is the Smagorinsky constant, which typically assumes values in $[0.1, 0.2]$.

The Smagorinsky model is one of the most used in turbulence modeling, and can be derived from the K41 theory. Now, the question is:

Can the Smagorinsky model be seen as a filtering approach?

The Smagorinsky model is not a filter in its classical mathematical sense. It provides a closure *ansatz* to the **SGS** stress tensor, representing the unresolved scales. However, it acts like an implicit filter, as the variation of the filtered energy $\bar{\mathcal{E}}$ is always decreasing, namely:

$$\frac{d\bar{\mathcal{E}}}{dt} \propto -\nu \int_{\Omega} |\nabla \bar{\mathbf{u}}|^2 dx - (C_S \delta)^2 \int_{\Omega} |\nabla \bar{\mathbf{u}}|^3 dx \leq 0. \quad (1.16)$$

This result shows that the Smagorinsky model can only remove (or at most preserve) the energy of the resolved scales, effectively acting as an **implicit filter**. Moreover, as previously mentioned, it is equivalent to a nonlinear filter with the indicator function of Equation (1.12).

Proof: the Smagorinsky model is a nonlinear filter.

In Smagorinsky's hypothesis:

$$\nabla \cdot \bar{\mathbf{T}}_{\text{SGS}ij} \sim -\nabla \cdot (\nu_t \bar{\mathbf{S}}_{ij}) \simeq -\nabla \cdot (\nu_t \nabla \bar{\mathbf{u}}) \simeq -\nabla \cdot ((C_S \delta)^2 |\nabla \bar{\mathbf{u}}| \nabla \bar{\mathbf{u}}).$$

This leads to:

$$\mathbf{u} - \bar{\mathbf{u}} \simeq -\nabla \cdot ((C_S \delta)^2 |\nabla \bar{\mathbf{u}}| \nabla \bar{\mathbf{u}}). \quad (1.17)$$

The nonlinear differential filter can be written as:

$$\mathbf{u} - \bar{\mathbf{u}} = -2\delta^2 \nabla \cdot (a(\mathbf{u}) \nabla \bar{\mathbf{u}}), \text{ with } a(\mathbf{u}) \propto |\nabla \mathbf{u}|. \quad (1.18)$$

The filtering action does not have a large effect on the velocity gradients, and we can approximate the indicator function as $a(\mathbf{u}) \propto |\nabla \mathbf{u}| \simeq |\nabla \bar{\mathbf{u}}|$, retrieving the same expression as in Equation (1.17).

While being largely used in **LES** applications, the Smagorinsky model often leads to overdiffusive behaviour, and it is not sufficiently expressive [284].

1.3 The EFR approach

The filtering strategies presented in Section 1.2 are widely used in the context of **LES** in a **modeling** fashion: the filter serves to properly select some scales (or wavenumbers), which are resolved, while the smaller scales either remain unresolved or are modelled via tailored closure strategies. However, in this thesis, we adopt a different perspective. Rather than interpreting the filter as a modeling tool, we employ it as a **stabilization**

mechanism for the numerical discretization of the governing equations. When the **NSE** are solved on coarse meshes, the numerical approximation may not accurately capture the velocity field. In particular, we will mainly work in *marginally-resolved* regimes, where the grid size is typically larger than the Kolmogorov length scale: $h_{\min} > O(\eta)$. In this scenario, the smallest dissipative scales remain unresolved, resulting in poor approximations. These inaccuracies can propagate and amplify over the time evolution of the solution, eventually leading to spurious and unphysical oscillations. Such issues are primarily **numerical** in nature and arise from the inability of the discrete approximation to properly represent the smallest scales. In this framework, filtering is introduced as a stabilization technique aimed at damping the high-wavenumber components that cannot be properly represented on the computational mesh.

This idea forms the basis of the **Evolve-Filter-Relax (EFR)** approach, which was presented in [86, 189]. This approach consists of repeating three steps at each time step of the numerical simulation: (I) **evolve** the velocity and pressure solutions via the discretized **NSE**; (II) **filter** the velocity solution with a suitable pre-selected filter approach; (III) **relax** the solution by taking a convex combination of the filtered and evolved velocity.

We consider a suitable discretization of the **NSE**³, like **FEM** or **FVM**, and we call $\mathbf{u}_n \in \mathbb{V}_{\mathbf{u}}$ and $p_n \in \mathbb{V}_p$ the discrete velocity and pressure solutions at time step t_n . $\mathbb{V}_{\mathbf{u}}$ and $\mathbb{V}_p$ are suitable discrete spaces, and we consider a time discretization $t_0, t_1, \dots, t_{N_T}$, where $t_{N_T} = T$, and N_T is the total number of time steps. If we assume a fixed time step Δt , the general **EFR** formulation at the $(n+1)$ -th time step reads as follows.

$$\begin{aligned}
 \text{(I) Evolve : } & \begin{cases} \frac{\mathbf{w}_{n+1} - \mathbf{u}_n}{\Delta t} + (\mathbf{w}_{n+1} \cdot \nabla) \mathbf{w}_{n+1} = \nu \Delta \mathbf{w}_{n+1} - \nabla p_{n+1} + \mathbf{f}_{n+1} & \text{in } \Omega, \\ \nabla \cdot \mathbf{w}_{n+1} = 0 & \text{in } \Omega, \\ \mathbf{w}_{n+1} = \mathbf{u}_D & \text{on } \Gamma_D, \\ -p_{n+1} \cdot \mathbf{n} + \frac{\partial \mathbf{w}_{n+1}}{\partial \mathbf{n}} = 0 & \text{on } \Gamma_N. \end{cases} \\
 \text{(II) Filter : } & \begin{cases} \bar{\mathbf{w}}_{n+1} = \mathcal{F}(\mathbf{w}_{n+1}) & \text{in } \Omega, \\ \bar{\mathbf{w}}_{n+1} = \mathbf{u}_D & \text{on } \Gamma_D, \\ \frac{\partial \bar{\mathbf{w}}_{n+1}}{\partial \mathbf{n}} = 0 & \text{on } \Gamma_N. \end{cases} \\
 \text{(III) Relax : } & \mathbf{u}_{n+1} = \chi \bar{\mathbf{w}}_{n+1} + (1 - \chi) \mathbf{w}_{n+1} \text{ in } \Omega.
 \end{aligned} \tag{1.19}$$

Remark: on the **EFR** notation.

- The notation of (1.19) introduces two intermediate discrete velocities: $\mathbf{w}_n$ (the evolved velocity at time t_n), and $\bar{\mathbf{w}}_n$ (the filtered velocity at time t_n).
- The time integration scheme in the evolve step of (1.19) is a simple implicit Euler

³From now onwards, we will consider the notation “ $\mathbf{s}$ ” ($\text{\texttt{\textbf{s}}}$) for continuous functions and “ $\mathbf{s}$ ” ($\text{\texttt{\textbf{s}}}$) for discrete vectors. For the operators, we use the notation “ $\mathcal{S}$ ” ($\text{\texttt{\textit{S}}}$) for continuous operators, and “ $\mathcal{S}$ ” ($\text{\texttt{\textit{S}}}$) for discrete operators. The corresponding discrete matrix representations are written as “ $\mathbf{S}$ ” ($\text{\texttt{\textbf{S}}}$).

formula, but any kind of time scheme can be considered.

- ▶ $\chi \in [0, 1]$ is the so-called **relaxation parameter**. $\chi = 0$ corresponds to no filtering, i.e., $\mathbf{u}_{n+1} = \mathbf{w}_{n+1}$, while $\chi = 1$ corresponds to only retaining the filtered velocity, namely $\mathbf{u}_{n+1} = \bar{\mathbf{w}}_{n+1}$. We will denote with **Evolve-Filter (EF)** the case $\chi = 1$, and with **noEFR** or **noEF** the case $\chi = 0$.
- ▶ $\mathcal{F}$ denotes a general filtering technique in the discrete setting. Even if many references consider the **DF** as filter in the classical **EFR** algorithm, we decided to keep the notation as general as possible, to allow for generalization and comparison of different models.

One of the main advantages of the **EFR** algorithm is its **modular character**: the filtering and the relaxation steps are applied as post-processing stages, and do not affect the underlying Navier–Stokes solver. This makes the approach compatible with basically all the most-used spatial discretization approaches, including finite element, finite volume, and spectral methods. Additionally, if we consider an explicit time integration scheme, the **EFR** algorithm can be seen as a closure approach, instead of separately applying the evolve, filter, and relax step. If we denote with NS the evolve step which maps the velocity $\mathbf{u}_n$ into the next evolved velocity $\mathbf{w}_{n+1}$, we can write:

$$\mathbf{u}_{n+1} = \chi \mathcal{F}(\text{NS}(\mathbf{u}_n)) + (1 - \chi) \text{NS}(\mathbf{u}_n) = \text{NS}(\mathbf{u}_n) + \chi(\mathcal{F} - I) \text{NS}(\mathbf{u}_n), \quad (1.20)$$

where the term $\chi(\mathcal{F} - I) \text{NS}(\mathbf{u}_n)$ represents the added term by the filter and relaxation steps.

Proof: when considering an explicit Euler time scheme, the **EFR** approach is a closure model.

To make the interpretation of (1.20) more explicit, we approximate the evolve operator at first order in time, as done in (1.19). Assuming a fixed time step Δt , and a time discretization of the form $\text{NS}(\mathbf{u}_n, p_n) \approx \mathbf{u}_n + \Delta t \mathcal{N}(\mathbf{u}_n, p_n)$, with $\mathcal{N}(\mathbf{u}_n, p_n) = -(\mathbf{u}_n \cdot \nabla) \mathbf{u}_n + \nu \Delta \mathbf{u}_n - \nabla p_n + \mathbf{f}_n$, we obtain:

$$\mathbf{u}_{n+1} \approx \mathbf{u}_n + \Delta t \mathcal{N}(\mathbf{u}_n, p_n) + \chi(\mathcal{F} - I)(\mathbf{u}_n + \Delta t \mathcal{N}(\mathbf{u}_n, p_n)).$$

Retaining only first-order contributions yields

$$\boxed{\frac{\mathbf{u}_{n+1} - \mathbf{u}_n}{\Delta t} \approx \mathcal{N}(\mathbf{u}_n, p_n) + \frac{\chi}{\Delta t}(\mathcal{F} - I)\mathbf{u}_n},$$

which can be interpreted as a closure model acting on the unresolved scales. In the case of a differential filter, using the Neumann series approximation, one can write:

$$\mathcal{F} = (I - 2\delta^2 \Delta)^{-1} \approx (I + 2\delta^2 \Delta) \rightarrow (\mathcal{F} - I)\mathbf{u}_n \approx 2\delta^2 \Delta \mathbf{u}_n.$$

Therefore, the closure term becomes

$$\frac{\chi}{\Delta t}(\mathcal{F} - I)\mathbf{u}_n \approx \frac{2\chi\delta^2}{\Delta t}\Delta\mathbf{u}_n,$$

which corresponds to an additional dissipative term of Laplacian type, analogous to an eddy-viscosity contribution.

1.3.1 The effect of different filters: a numerical example

This section is dedicated to analyzing the effect of different filters on the two-dimensional decaying **Homogeneous Isotropic Turbulence (HIT)** test case at $Re = 40\,000$ in a square domain $\Omega = [0, 1]^2$, with periodic boundary conditions. This test case is a benchmark in **CFD** [4, 173, 182], and is characterized by decaying kinetic energy in time, as no external forces are injected into the system.

All numerical simulations are performed using the library `IncompressibleNavierStokes.jl` [2, 3], namely considering a **FVM** on a staggered grid, as presented in [291, 123], and using a fourth-order Runge Kutta time integration scheme. We introduce two grids, whose properties are displayed in Table 1.2: a well-resolved one for the **DNS**, and a coarser one for the no**EFR** and **EF** simulations. It is also important to highlight that the reference is not the **DNS** itself, but the *filtered DNS*, obtained by applying a face-average (or volume-average in 3D) operation [173] to the original solutions on the well-resolved grid⁴. The vorticity at the initial time is represented in Figure 1.3.

Remark: velocity initialization in the decaying **HIT** test case.

The initial **DNS** velocity is sampled from a random velocity field with prescribed energy spectrum [241, 290, 4]. More in detail, the velocity field is initialized with the following procedure:

1. The velocity is sampled in spectral space to match a prescribed energy at each wave number κ (called $\widehat{\mathcal{E}}_\kappa$). Each component of the spectral velocity is indeed initialized such that $\|\widehat{\mathbf{u}}_\kappa\| = |b_\kappa|$, where $b_\kappa = \sqrt{2\widehat{\mathcal{E}}_\kappa}e^{2\pi i\tau_\kappa}$ and τ_κ is a random phase shift.
2. The velocity is then projected to make it divergence-free: $\widehat{\mathbf{u}}_\kappa = \frac{b_\kappa \widehat{\mathbf{P}}_\kappa \mathbf{e}_\kappa}{\widehat{\mathbf{P}}_\kappa \mathbf{e}_\kappa}$, where $\widehat{\mathbf{P}}_\kappa$ is a projector: $\widehat{\mathbf{P}}_\kappa = I - \frac{\kappa \kappa^T}{\kappa^T \kappa}$ and $\mathbf{e}_\kappa$ is a random unit vector. In 2D, $\mathbf{e}_\kappa = (\cos(\theta_\kappa), \sin(\theta_\kappa))$ with $\theta_\kappa \sim \mathcal{U}[0, 2\pi]$, where $\mathcal{U}$ indicates a uniform distribution.
3. The velocity field is finally obtained by a transformation into physical space ($\mathbf{u}_0 = \text{FFT}^{-1}(\widehat{\mathbf{u}})$), and a further projection step such that $\mathbf{H}\mathbf{u}_0 = 0$, where $\mathbf{H}$ is

⁴Since this chapter serves as motivation for the filtering methodologies herein introduced, we postpone the detailed description of the staggered grid framework to Section 4.1, where this test case is also considered.

the discrete divergence operator.

We also remark that we consider as characteristic velocities and length scale $U = 1$ and $L = 1$, while being the viscosity $\nu = 1/Re$.

The goal of this section is to compare classical filtering strategies (Smagorinsky, **DF**, and Gaussian) within the **EF** framework, namely replacing the filter $\mathcal{F}$ in (1.19) with different filters. We consider classical literature values for the filters' parameters, namely: $C_S = 0.15$ and $\delta = h$ for the Smagorinsky model, $\delta = h$ for the Gaussian filter, and $\delta = \eta$ for the **DF**. The resulting graphical vorticity fields and the energy spectrum at different times are represented in Figure 1.5.

Table 1.2: Properties of the grids considered in the HIT case.

Grid	Resolution	Simulation type
Marginally-resolved	128^2	Filtered DNS , no EFR , EF
Well-resolved	512^2	DNS

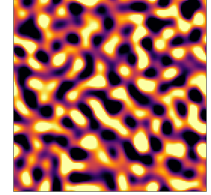


Figure 1.3: Initial vorticity in the HIT case.

First, we can notice that the reference (filtered **DNS**) solution is characterized by dissipation, as the vortices become larger from $t = 0.5$ to $t = 2$ s.

Second, the filtered **DNS** recovers the theoretical 2D enstrophy decay, as represented in Figure 1.1b. If we decrease the resolution of the grid from 512^2 to 128^2 without applying any filter (no**EFR**), we can see that the solution exhibits spurious oscillations (Figure 1.4) and needs regularization. This is also confirmed by the spectrum (Figure 1.5): no**EFR** exhibits larger energy at the small scales (large wavenumbers).

The introduction of a filtering step on the coarse grid (**EF**) leads to a clear qualitative improvement in the solution, particularly in terms of smoothness. Among the considered approaches, both the Gaussian and the **DF** filters exhibit an overly dissipative behavior, whereas the Smagorinsky model yields comparatively more accurate and physically consistent results. This trend is further confirmed by the energy spectra at times $t = 0.5$ and $t = 2$, where the Smagorinsky model remains the closest to the reference solution. Nevertheless, all the considered methods systematically underestimate the energy across the entire range of wavenumbers, with the discrepancy becoming more pronounced as time evolves, indicating an increasing level of overdissipation.

The overdissipative behaviour observed for this test case is well documented in the literature and highlights a fundamental limitation of classical filtering approaches, namely their strong dependence on the choice of hyperparameters (e.g., C_S and δ in the present setting). The next section is devoted to reviewing existing strategies aimed at mitigating these limitations.

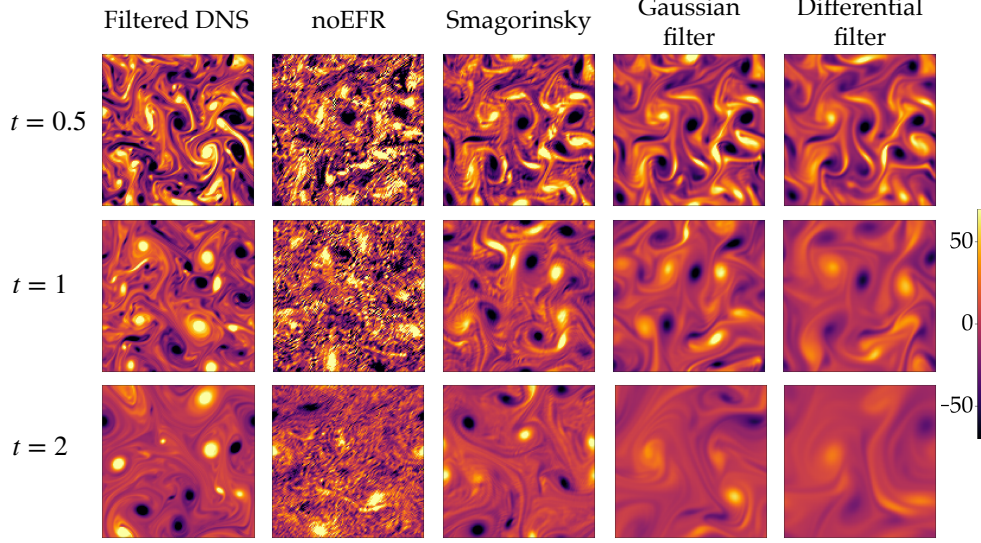


Figure 1.4: *Decaying HIT* test case. Vorticity fields at $t = 0.5$, $t = 1$ and $t = 2$ for the filtered DNS (reference) simulation, noEFR, EF simulations with different filter types (Gaussian, DF), and simulation with Smagorinsky closure.

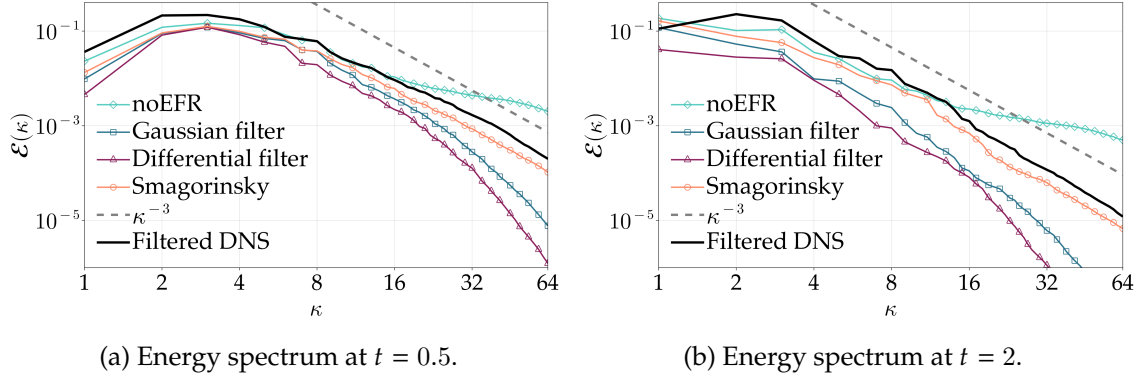


Figure 1.5: *Decaying HIT* test case. The effect of explicit and implicit filtering strategies in the EF approach on the energy spectrum, at different time instances.

1.4 Motivation for the proposed approaches

The limitations of classical filtering strategies have motivated a substantial body of research aimed at developing more accurate and adaptive approaches. In the following, we briefly review the main strategies proposed in the literature to mitigate these issues, and outline the motivations for the methodology introduced in this thesis.

A first line of research is based on the **Variational Multi-Scale (VMS)** framework [134, 135, 136]. The main idea is to decompose the solution into resolved and unresolved scales, typically associated with the numerical discretization. The effect of the unresolved scales on the resolved ones is then modeled, rather than neglected, leading to a closure that acts primarily on the small scales. In this sense, VMS can be interpreted as a projection-based

filtering approach, where the resolved scales correspond to the numerical approximation space and the unresolved scales to its complement. Further developments include residual-based VMS formulations, where the subgrid scales are expressed in terms of the residual of the resolved equations [27]. Compared to classical eddy-viscosity models, VMS-based approaches offer improved stability properties and a clearer connection with the underlying mathematical structure of the governing equations.

Furthermore, another related approach is **approximate deconvolution (AD)** [318, 319, 85]. The main idea is to reconstruct an approximation of the unfiltered solution starting from the filtered one, by approximately inverting the filtering operator. The inverse operator is called approximate deconvolution and is indicated with $\mathcal{D}$. This allows to recover part of the unresolved scales and use them to define a closure term.

One enhancement of the DF is the usage of the **indicator function**, as introduced in Equation (1.11). The indicator functions can be distinguished into two categories: *physics-based* and *mathematics-based* [259]. Physics-based indicator functions [189] are based on physical quantities that are known to vanish for coherent flow structures. Some examples are: indicator functions based on the Q criterion [82], on the Vreman eddy viscosity expression [345, 42], or on the so-called **helicity** [42], defined as the scalar product between velocity and vorticity: $\mathcal{H} = \mathbf{u} \cdot \boldsymbol{\omega}$. This quantity of interest quantifies the presence of helical or swirling flow structures and is often associated with coherent vortical motions. In the context of indicator functions, the helicity can be exploited to detect regions where the flow exhibits organized rotational structures. In such regions, excessive numerical dissipation should be avoided and $a(\mathbf{u}) \simeq 0$. Conversely, in regions where the helicity is small, the flow is less structured and more prone to spurious oscillations, so that stronger filtering may be applied, namely $a(\mathbf{u}) \simeq 1$. Although having a straightforward implementation, physics-based indicator functions do not allow for rigorous mathematical convergence. This issue is instead overcome with mathematics-based indicator functions. In this class, approximate deconvolution-based indicator functions [41] estimate unresolved scales by applying an approximate inverse of the filter ($\mathcal{D}$) to the computed solution. The mismatch between the true field $\mathbf{u}$ and its reconstruction $\mathcal{D}(\mathcal{F}(\mathbf{u}))$ provides a local measure of under-resolution, which is used to adapt the filtering intensity: stronger where needed, weaker where the solution is already well resolved. This approach, when using the Van Cittert approximate deconvolution, has a strong mathematical theory [318, 319, 85].

In the context of the EFR algorithm, some approaches, called **relaxation-consistent** [35], recompute the pressure after the filter step, obtaining an intermediate pressure field, which is then relaxed like the velocity field. This ensures consistency in the velocity-pressure coupling. Consistency and stability motivations might also drive the introduction of a pressure stabilization in the EFR framework [114].

Another important line of research is related to the **dynamic tuning** of the Smagorinsky constant C_s . Dynamic procedures, originally introduced by Germano et al. [109] and reformulated by Lilly [200], allow the model coefficient to be determined adaptively from

the resolved scales, instead of being prescribed *a priori*. These approaches were later extended to improve robustness and locality, for instance through localized dynamic formulations [111] and Lagrangian averaging strategies [224], which avoid the need for homogeneous directions. More recently, attention has shifted towards fully local dynamic models, as in [278], where the authors proposed a dynamic gradient Smagorinsky model, which replaces the rate-of-strain tensor with the velocity gradient, yielding a more robust and stable formulation. The same idea of dynamic tuning has also been extended to filtering-based and deconvolution models. In this context, Tejada [328] proposed a dynamic procedure in which the model parameters are computed adaptively from the resolved scales using a Germano-type identity, avoiding the use of fixed coefficients.

The idea of dynamically adapting model parameters has recently been revisited from a data-driven perspective. In particular, **Reinforcement Learning (RL)** provides a framework to determine model parameters based on the observed flow dynamics, by optimizing suitable performance criteria such as accuracy or stability over time. **RL** can be indeed seen as a generalization of the dynamic parameters selection, and has been applied in the filtering context, for example, for the tuning of the Smagorinsky constant [235, 183, 182, 181].

RL is part of a broader class of **data-driven approaches for flow stabilization**, which have attracted growing interest in recent years as alternatives to traditional physically or mathematically-motivated indicator functions. The **LES** and **EFR** frameworks are particularly relevant in this regard, as both aim to suppress spurious oscillations during the simulation through conceptually similar mechanisms [259], and the **LES** community has been at the forefront of integrating machine learning into this process [180, 305, 219, 173, 248, 38, 118, 4]. The common thread across these contributions is the use of neural networks, most commonly convolutional architectures, to provide a data-driven closure for the subgrid-scale stress tensor, replacing or enriching classical models such as Smagorinsky. Depending on the training strategy, these approaches range from *a priori* methods, where the network is trained to reproduce the filtered **DNS** closure term directly, to fully differentiable end-to-end formulations, where the network parameters are optimized with respect to the accuracy of the resulting **LES** solution. A representative example of the latter is the work of [203], where a convolutional neural network is incorporated into the right-hand side of the governing system to enhance **LES** accuracy.

While improving standard filtering approaches for **LES** and **EFR** frameworks has been the focus of a wide range of recent research works, several improvements can still be made. We now focus on the **EFR** formulation and try to address the following questions:

- *Is it always beneficial to apply filtering? What are the optimal **EFR** parameters?*

While several studies have investigated spatially varying filter intensities (e.g., through indicator functions), we study the variation of the **EFR** parameters in time. To the

best of our knowledge, the time variation of such parameters has not been considered in previous studies. In many practical scenarios, the flow may undergo significant transitions in time, for instance from laminar to turbulent regimes, raising the question of whether filtering and/or relaxation should be applied at every time step and whether constant parameters throughout the simulation are an appropriate choice.

This question is addressed by Chapter 2, where the filter radius δ and the relaxation parameter χ are time-dependent, and optimized at each time step to match the reference data, using gradient-based optimization techniques.

- *Can RL be leveraged to achieve adaptive filtering, without relying on reference solutions?*

While data-driven optimization provides a powerful tool for parameter tuning and for understanding the problem of interest, it typically requires reference data (e.g., DNS), which may be prohibitively expensive or unavailable in practical applications. This naturally leads to the question of whether one can learn optimal filtering strategies directly from the simulation itself.

This challenge motivates Chapter 3, where RL is employed to adaptively tune the filter width. In this setting, we introduced a **reference-free** type of algorithm, based solely on quantities available from the marginally-resolved simulation itself, thus removing the need for high-fidelity reference data.

- *Can the filtering operator be enriched to improve flexibility and accuracy?*

Standard DFs are typically characterized by a single parameter (δ), which may limit their expressiveness. This raises the question of whether more general and data-driven filtering operators, with additional degrees of freedom, can provide enhanced accuracy and adaptability.

This question is addressed by Chapter 4, where the DF is replaced with a novel type of filter, computed from pre-computed reference data.

- *Is it possible to design EFR approaches that preserve relevant physical quantities, such as energy and enstrophy?*

As discussed in Section 1.2, classical DFs naturally preserve energy and enstrophy. However, this property is generally lost when the filtering operator is replaced by a data-driven counterpart.

To recover this physically relevant feature, we introduce an adaptive strategy in which the relaxation parameter χ is dynamically tuned to enforce the conservation of selected quantities. In particular, this approach enables the preservation of energy and enstrophy, leading to two EFR extensions that are presented in Chapter 4.

OPT-EFR: PARAMETERS' OPTIMIZATION FOR EFR

This chapter has the goal of improving the **EFR** approach presented in Section 1.3. In particular, for the filter step, we select the **DF** of Equation (1.9). Therefore, the **EFR** approach depends on two parameters, i.e., δ and χ . Standard choices are associated with the physical parameters and/or with the setup of the simulation [34, 324]. While the filter width δ is typically related to physical quantities ($\delta \sim \eta$ or $\delta \sim h_{\min}$), the choice of the relaxation parameter is typically based on more agnostic arguments and/or on experience. One of the common choices is $\chi \sim O(\Delta t)$, e.g., $\chi = C \Delta t$, $C \sim O(1)$ [86], but it might completely change with respect to the test case considered. Hence, the tuning of these two parameters is the main bottleneck of the **EFR** formulation. If we assign **too large** δ and/or χ values, the simulation might lead to **over-diffusive** solutions that does not manifest any turbulent behavior. If the values of δ and/or χ are **too small**, the filter action may be not enough, leading to **spurious oscillations**, as happens in the no**EFR** simulation.

Moreover, the flow behaviour may not be accurately captured if we consider parameters constant in time, especially in turbulent flows. For example, we may want higher δ and/or χ values in the laminar-to-turbulent transition or in fully-turbulent regimes. This chapter, which presents the work of [147], provides a first investigation on how to tune those parameters. The key novelties are:

- ▶ the introduction of **time-dependent parameters** $\delta(t)$ and $\chi(t)$;
- ▶ a **data-driven adaptive optimization** of the parameters in time, based on a reference well-resolved simulation.

Such methodology is tested on a well-known benchmark in **CFD**, the two-dimensional flow past a cylinder at $Re = 1000$, simulated using the **FEM**.

From now on, we call **Standard EFR** the simulation with $\delta = \eta$ and $\chi = 5 \Delta t$, and **Standard EF** the simulation with $\delta = \eta$ and no relaxation, namely $\chi = 1$.

This chapter is subdivided as follows:

- ▶ Section 2.1, which presents the test case and shows the results of standard **EFR** (and **EF**) approaches;
- ▶ Section 2.2, which presents the **Optimized Evolve-Filter-Relax (Opt-EFR)** algorithm;
- ▶ Section 2.3, highlighting the results achieved with the novel methodology on the flow past a cylinder test case.

- Section 2.4, which draws the conclusions for this work and motivates the RL extension of Chapter 3.

2.1 The flow past a cylinder in FEniCS: standard EFR results

The two-dimensional flow past a cylinder at fixed $Re = 1000$ is a widely used benchmark in fluid dynamics. Our work focuses on the FEM simulation of this test case, using the open-source software FEniCS [9, 206, 205, 207]. The domain Ω is represented in Figure 2.1a, and the cylinder has center in $(0.2, 0.2)$ and radius 0.05. A discretized version of the NSE in System (1.1) is solved here, applying no-slip boundary conditions on $\partial\Omega_{wall} = \partial\Omega_B \cup \partial\Omega_T$, a standard free-flow condition on $\partial\Omega_N$, and the following inlet boundary condition:

$$\mathbf{u}_{in} = \left(\frac{6}{0.41^2} y(0.41 - y); 0 \right) \text{ on } \partial\Omega_{in},$$

where $\mathbf{x} = (x; y)$. The viscosity is set to $\nu = 1 \times 10^{-4}$. We consider a backward differentiation formula of order 1 (BDF1) as a time integration scheme, and standard $\mathbb{P}^2 - \mathbb{P}^1$ triangular elements for the velocity-pressure coupling. The simulation is performed within the time window $(0, 4]$, i.e., $t_0 = 0$ and $T = 4$, while the time step is fixed and $\Delta t = 4 \times 10^{-4}$ in all the simulations.

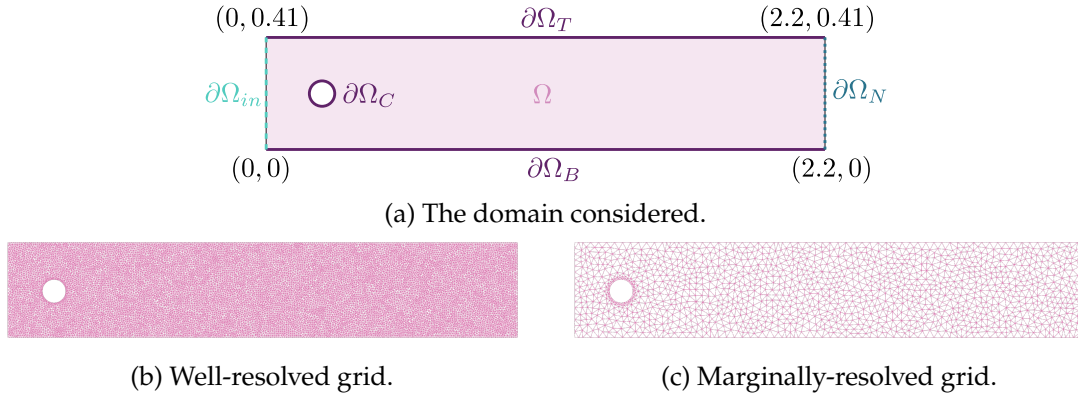


Figure 2.1: *Flow past a cylinder test case.* Domain and grids considered for the DNS and for the marginally-resolved simulation in FEniCS.

Figures 2.1b and 2.1c display the well-resolved and marginally-resolved grids considered, respectively. Our reference solution is the filtered DNS, which is obtained by performing the FEM simulation on the well-resolved grid, and then projecting it onto the coarse grid¹. All the other simulations (noEFR, EFR, EF) are instead performed on the marginally-resolved grid. From now on, we will denote with $\mathbf{u}^{\text{ref}}$ and p^{ref} the reference velocity and pressure solutions, namely the DNS solutions projected on the marginally-resolved grid.

¹We compute the L^2 projection of the DNS velocity field onto a divergence-free space defined on the coarse grid by solving a mixed Stokes-like problem with an auxiliary pressure variable.

Table 2.1 presents the features of the two grids considered, namely the number of degrees of freedom for the velocity and pressure fields, and the wall time needed to perform 4 seconds of simulation². The velocity and pressure solutions of the DNS, the noEFR, the standard EFR and standard EF, namely the EFR/EF approaches using the literature values, are represented in Figure 2.2. We adopt the following notation for the vector spaces: $\mathbf{u} \in \mathbb{V}_{\mathbf{u}}$, $p \in \mathbb{V}_p$, $\mathbf{u}^{\text{ref}} \in \mathbb{V}_{\mathbf{u}}^{\text{ref}}$, and $p^{\text{ref}} \in \mathbb{V}_p^{\text{ref}}$.

Table 2.1: *Flow past a cylinder test case*. Degrees of freedom (dofs) for the velocity and pressure fields, and the wall time needed to perform 4 seconds of simulation on the marginally-resolved and well-resolved grids.

Grid	Velocity dofs	Pressure dofs	Wall time
Marginally-resolved	12430	1623	26 820 s ~ 7 hours 27 minutes
Well-resolved	100676	12760	1 382 400 s ~ 16 days

As can be seen from Table 2.1 and from Figures 2.2a and 2.2b, a well-resolved simulation provides a physically reliable solution³, but might cost a lot in terms of computational time and memory, e.g. days, even for a two-dimensional test case at moderate Re . On the other hand, the simulation on the coarser grid is more feasible in terms of time, but it presents numerical oscillations in the entire domain and needs stabilization (Figures 2.2c and 2.2d). This motivates the use of filtering (and, eventually, relaxation) on the marginally-resolved grid, which leads to the following results:

- ▶ The standard EFR solutions (Figures 2.2e and 2.2f) exhibit spurious oscillations, just as in the noEFR approach. This suggests that the filtering and/or the relaxation parameters are too small, and not adequate for the test case.
- ▶ The standard EF solutions (Figures 2.2g and 2.2h) are overdifusive, and the vortex shedding is less developed compared to the reference solution.

This motivates the use of time-dependent parameters $\delta(t)$ and $\chi(t)$, and the development of methodologies to optimize the value of such parameters in time.

2.2 The data-driven optimized EFR algorithms

In this section, we introduce three specific Opt-EFR strategies aimed at optimizing one or both parameters. If we denote with $\mu_n = \mu(t_n)$ the tuple of parameter(s) optimized at each time step, the three approaches are:

²Wall times refer to the simulations performed on 2 cores on the SISSA HPC cluster Ulysses (200 TFLOPS, 2TB RAM, 7000 cores). The times refer to the noEFR (marginally-resolved grid) and DNS simulations (well-resolved grid).

³Note that also the simulation on the well-resolved grid presents spurious numerical oscillations at the outlet, and it cannot be considered a *fully-resolved* simulation in its classical sense. We call it *reference* as we consider it as the high-fidelity solution in the optimization training.

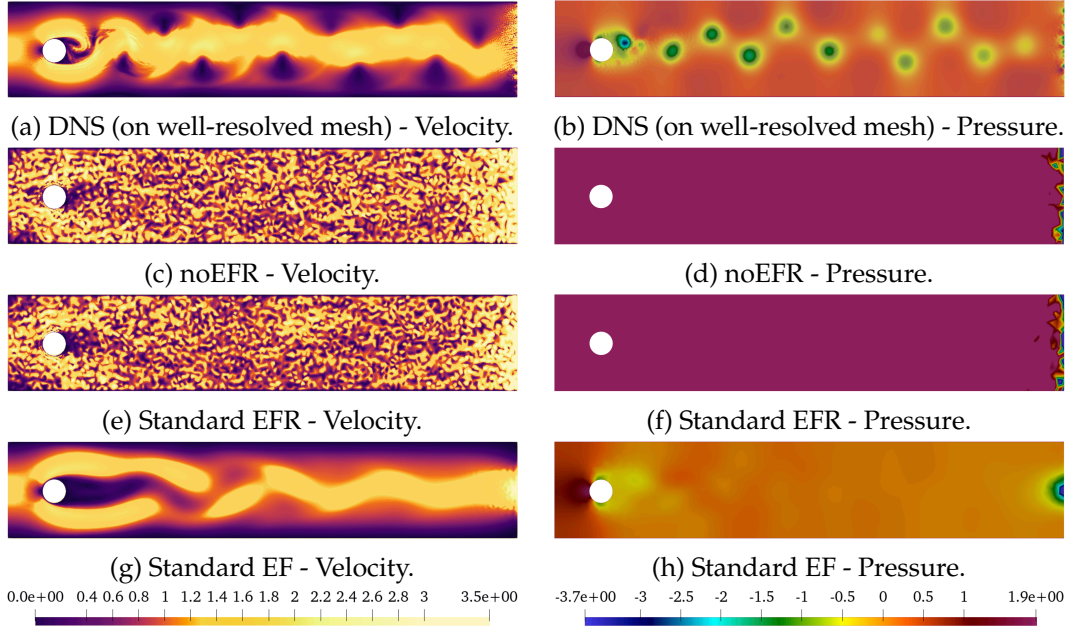


Figure 2.2: *Flow past a cylinder test case.* Velocity magnitude and pressure fields for the DNS, the noEFR, the standard EFR ($\delta = \eta$, $\chi = 5\Delta t$), and the standard EF ($\delta = \eta$, $\chi = 1$) simulations, at the final time $T = 4$.

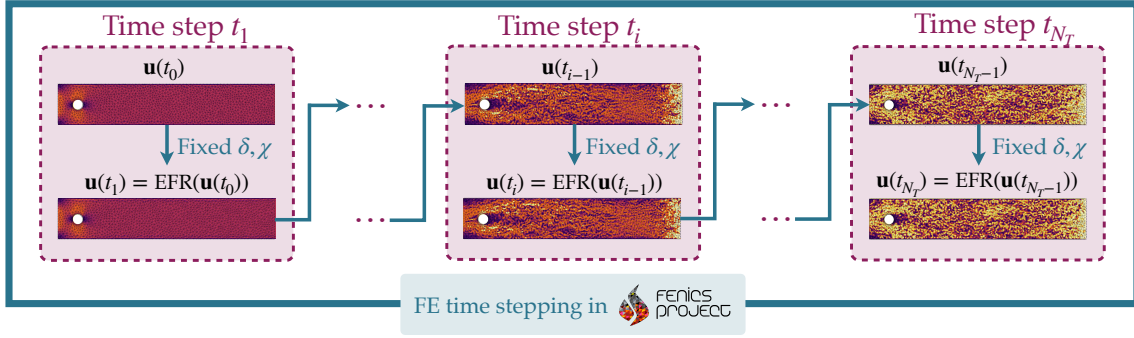
- **EFR with optimized relax parameter (χ -Opt-EFR)**, where $\mu_n = \chi_n$, at fixed filter width $\delta = \eta$;
- **EF with optimized filter width (δ -Opt-EF)**, where $\mu_n = \delta_n$, at no relaxation, namely $\chi = 1$;
- **EFR with optimized filter and relax parameters ($\delta\chi$ -Opt-EFR)**, where $\mu_n = (\delta_n, \chi_n)$.

All these optimization procedures are based on a data-driven algorithm trained at some prescribed time steps, taking as reference the pre-computed filtered DNS solution. A sketch of the standard EFR and of the Opt-EFR methodologies is represented in Figure 2.3.

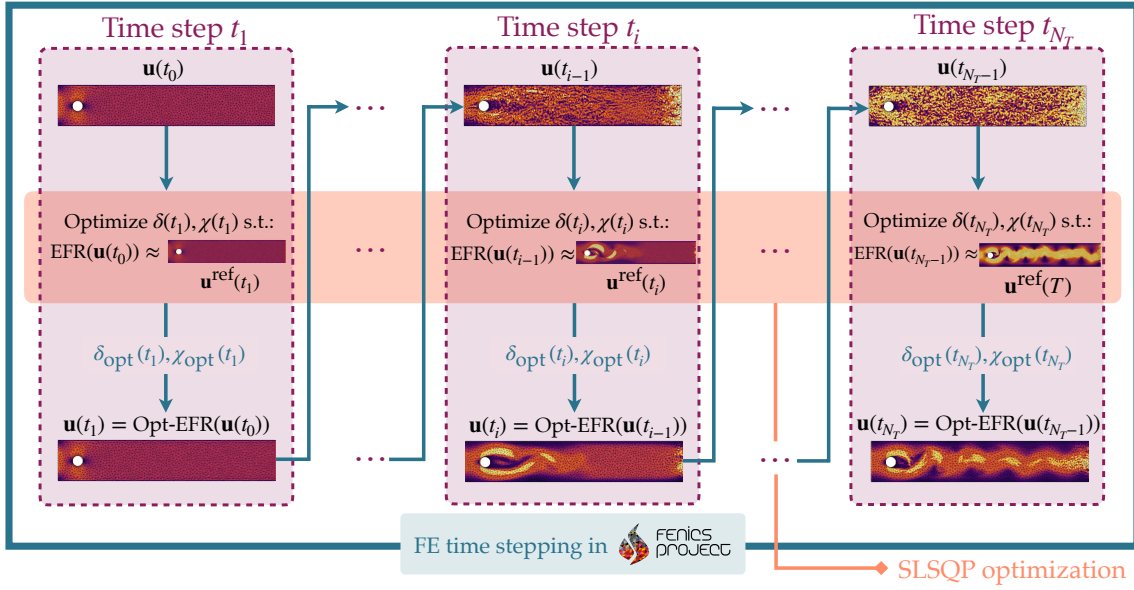
All the Opt-EFR methods will be tested on the marginally-resolved grid, and compared with the standard EFR/EF solutions in terms of accuracy and stability. It is important to remark the **modularity** and **generality** of the Opt-EFR methodologies: we will highlight their potentials in the flow past a cylinder test case, but the proposed algorithms can be applied to any experimental setting.

2.2.1 The objective functions

The crucial part of the proposed Opt-EFR algorithms is the selection of an appropriate objective function to optimize. We will consider either **local** or **global** contributions. More in detail, the **local** contribution refers to a pointwise evaluation of the discrepancy between the prediction and the reference, e.g., the standard **Mean Squared Error (MSE)**. The



(a) Standard EFR scheme.



(b) Opt-EFR scheme.

Figure 2.3: Sketch of the standard EFR and of the Opt-EFR procedures.

global approach refers to the evaluation and minimization of the discrepancy of global flow properties, e.g., the energy. We consider the following local contributions:

- ▶ The local $\mathbf{u}$ discrepancy: $\mathcal{L}_{\text{local}}^{\mathbf{u}}(\mu_n) = \text{MSE}(\mathbf{u}_n(\mu_n), \mathbf{u}_n^{\text{ref}})$,
- ▶ The local $\nabla \mathbf{u}$ discrepancy: $\mathcal{L}_{\text{local}}^{\nabla \mathbf{u}}(\mu_n) = \text{MSE}(\nabla \mathbf{u}_n(\mu_n), \nabla \mathbf{u}_n^{\text{ref}})$.

The local objective function reads as follows:

$$\mathcal{L}_{\text{local}}(\mu_n) = w_{\mathbf{u}} \mathcal{L}_{\text{local}}^{\mathbf{u}}(\mu_n) + w_{\nabla \mathbf{u}} \mathcal{L}_{\text{local}}^{\nabla \mathbf{u}}(\mu_n), \quad (2.1)$$

where $w_{\mathbf{u}}$ and $w_{\nabla \mathbf{u}}$ are scalar quantities. On the other hand, the global contributions are:

- ▶ The global $\mathbf{u}$ discrepancy: $\mathcal{L}_{\text{global}}^{\mathbf{u}}(\mu_n) = \left| \frac{\|\mathbf{u}_n(\mu_n)\|_2^2 - \|\mathbf{u}_n^{\text{ref}}\|_2^2}{\|\mathbf{u}_n^{\text{ref}}\|_2^2} \right|$,
- ▶ The global $\nabla \mathbf{u}$ discrepancy: $\mathcal{L}_{\text{global}}^{\nabla \mathbf{u}}(\mu_n) = \left| \frac{\|\nabla \mathbf{u}_n(\mu_n)\|_2^2 - \|\nabla \mathbf{u}_n^{\text{ref}}\|_2^2}{\|\nabla \mathbf{u}_n^{\text{ref}}\|_2^2} \right|$,

► The global p discrepancy: $\mathcal{L}_{\text{global}}^p(\mu_n) = \left| \frac{\|p_{n+1}(\mu_n)\|_2^2 - \|p_{n+1}^{\text{ref}}\|_2^2}{\|p_{n+1}^{\text{ref}}\|_2^2} \right|$.

The above-listed quantities can be combined together in the following global objective function:

$$\mathcal{L}_{\text{global}}(\mu_n) = w_u \mathcal{L}_{\text{global}}^u(\mu_n) + w_{\nabla u} \mathcal{L}_{\text{global}}^{\nabla u}(\mu_n) + w_p \mathcal{L}_{\text{global}}^p(\mu_n), \quad (2.2)$$

where w_p is a scalar quantity. Table 2.2 provides an overview of the acronyms used, depending on the loss functionals considered. In particular, the “✓” symbol indicates that the scalar weight associated with a specific contribution is equal to 1, while the gray cell indicates that the weight is equal to 0⁴.

Table 2.2: Acronyms and main features of the algorithms Opt-EFR. Gray cells indicate that the algorithm does not present that feature.

Acronym	Objective function type		Contributions in objective function		
	local	global	u discrepancy	∇u discrepancy	p discrepancy
Opt-EFR _{local} ^u	✓		✓		
Opt-EFR _{local} ^{u, ∇u}	✓		✓	✓	
Opt-EFR _{global} ^u		✓	✓		
Opt-EFR _{global} ^{u, ∇u}		✓	✓	✓	
Opt-EFR _{global} ^{u, ∇u, p}		✓	✓	✓	✓

A first question the reader may have at this point is:

Why do we introduce the global functionals?

The answer is that the local quantities might not be appropriate metrics, as we will show in Section 2.3. Even if the MSE is small, the Opt-EFR and the DNS solutions may indeed differ a lot in terms of qualitative and quantitative behaviour.

The second question is:

Why do we add the velocity gradients in the functionals, e.g. $\mathcal{L}_{\text{local}}^{\nabla u}$ and $\mathcal{L}_{\text{global}}^{\nabla u}$?

The answer is that the gradients describe how the flow deforms and rotates locally, making them more sensitive than the velocity alone and, therefore, more effective for capturing small-scale physics and improving agreement with the reference solution.

A third and final question might be:

Why do we consider the pressure discrepancy $\mathcal{L}_{\text{global}}^p$? Why is it evaluated at t_{n+1} instead of t_n ?

We investigate the effect of adding the pressure discrepancy because, once a good accuracy was obtained for the velocity field, we wanted to further improve the pressure field. We consider p_{n+1} in the pressure discrepancy because the filter and relax steps at time t_n only affect the velocity field u_n . The pressure field p_n is instead computed in the first evolve step *before* applying the filter. To evaluate the effect of the optimized δ_n and χ_n on the pressure, one should evolve the solution one more time and compute the pressure

⁴We assign scalar weights exactly equal to 1, when they are not 0, namely, we assign the same importance to all the contributions in the loss. Other choices can be made, but finding the optimal scalar weights was beyond the goal of our work.

p_{n+1} . Since a further evolve step should be performed at each evaluation, the approach $\text{Opt-EFR}_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}, \mathbf{p}}$ is computationally more expensive than the others. Moreover, we only consider the pressure discrepancy in the global objective function because the numerical results obtained for the local approach were not accurate and not worthy further investigation.

2.2.2 The optimization algorithm

The algorithms of Table 2.2 perform at each time t_n Step (I) of (1.19), and then look for the optimal δ_n and/or χ_n to be employed for Steps (II) and (III). The procedure is sketched in Figure 2.3b, and the general pseudocode can be found in Algorithm 1. The algorithm can be applied for all the strategies in Table 2.2 by replacing the objective function $\mathcal{L}(\mu_{n+1})$ with the specific functional considered. The optimization method is the **Sequential Least Squares Programming (SLSQP)** algorithm [253, 354], which is a quasi-Newton iterative method for constrained nonlinear optimization. The constraint is in this case the admissible range for parameters δ and/or χ , which we call A_μ . More in detail, we consider $\chi_n \in [0, 1]$ for all n , and $\delta_n \in [1 \times 10^{-5}, 1 \times 10^{-3}] \approx [0.02\eta, 0.2\eta]$. The window $[0.02\eta, 0.2\eta]$ has been selected after a preliminary sensitivity analysis, as it provides more physically meaningful results. Indeed, the unconstrained optimization may converge to values that are too far from the Kolmogorov length scale η , and in general to the literature choices. Hence, $A_\mu = [0, 1]$ in $\chi\text{-Opt-EFR}$, $A_\mu = [1 \times 10^{-5}, 1 \times 10^{-3}]$ in $\delta\text{-Opt-EF}$, and $A_\mu = [1 \times 10^{-5}, 1 \times 10^{-3}] \times [0, 1]$ in $\delta\chi\text{-Opt-EFR}$.

Algorithm 1 Pseudocode for algorithms Opt-EFR optimizing both parameters. Row 9 (underlined) is considered only for algorithms of type $\text{Opt-EFR}_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}, \mathbf{p}}$. The optimization part is coloured in orange.

```

1:  $\mathbf{u}_n^{\text{ref}}, n = 0, \dots, N_T - 1; \mathbf{u}_0, \Delta t_{\text{opt}}, \mu_0, A_\mu$  range of parameter(s). ▷ Inputs
2: for  $n \in [0, \dots, N_T - 1]$  do
3:   (I)  $\rightarrow \mathbf{w}_{n+1}$  ▷ Evolve step
4:    $\mathbf{u}_{n+1}^{\text{ref}} \leftarrow \mathbf{u}_{n+1}^{\text{ref}} | V_h^{\text{ref}}$  ▷ Project the DNS field into the coarse mesh
5:    $\mu_{n+1} \leftarrow \mu_n$  ▷ Initialize parameter(s)
6:   if  $\text{mod}(n, \Delta t_{\text{opt}}) = 0$  then ▷ Perform optimization every  $\Delta t_{\text{opt}}$ 
7:     iter = 0
8:     repeat
9:       (II)+(III)  $\rightarrow \mathbf{u}_{n+1}$  ▷ Filter and Relax steps
10:      (I)  $\rightarrow p_{n+2}$  ▷ (Second) Evolve step
11:      Optimize  $\mathcal{L}(\mu_{n+1})$ , subject to  $\mu_{n+1} \in A_\mu$ 
12:      iter  $\leftarrow$  iter + 1
13:      Output current solution:  $\mu_{n+1}$ 
14:    until Convergence or maximum iterations reached
15:  end if
16:  Solve (II)+(III) with optimal  $\mu_{n+1} \rightarrow$  Update  $\mathbf{u}_{n+1}$  ▷ Filter and Relax
17: end for

```

In the pseudocode, we introduce Δt_{opt} , that is the **optimization step**, namely the time

window between two consecutive optimizations. In our case, we decide not to optimize at each time step, namely $\Delta t_{\text{opt}} > \Delta t$, since the optimization process would have been too expensive. We consider $\Delta t_{\text{opt}} = N_{\text{opt}} \Delta t$, with $N_{\text{opt}} \in \{10, 20, \dots, 100\}$. The pipeline consists of training the optimization *on the fly* during the marginally-resolved simulation, every Δt_{opt} . The parameters are kept fixed to their optimal values in the time window between two subsequent optimizations. The numerical results in the following sections show a first analysis at fixed $\Delta t_{\text{opt}} = 10$. A deep analysis of the influence of the optimization step on the accuracy and efficiency of the **Opt-EFR** simulations is provided in Section 2.3.4.1. The parameters are initialized with their standard literature values, namely $\delta_0 = \eta$ and $\chi_0 = 5 \Delta t$, and the optimization process is implemented using the Python-based library Scipy [343], integrated within the FEniCS solver.

2.3 Numerical results of Opt-EFR

This Section highlights the numerical results of the three different **Opt-EFR** strategies in the dedicated Subsections 2.3.1, 2.3.2 and 2.3.3.

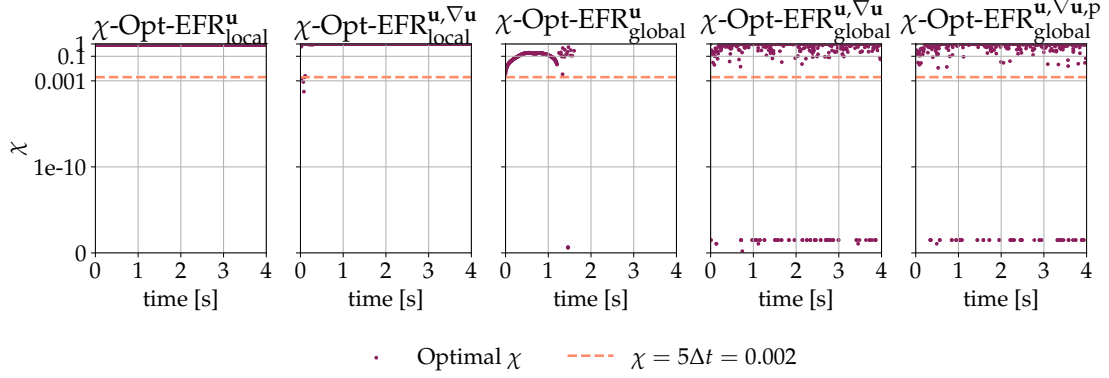
2.3.1 Numerical results of χ -Opt-EFR

We start by considering the χ -**Opt-EFR** methodology, i.e., the optimization of $\chi(t)$ at fixed $\delta = \eta$. In this first analysis, we consider all the proposed objective functions, namely all the algorithms of Table 2.2. Figure 2.4 presents the $\chi(t)$ trends (2.4a) for different local and global loss functions, and the distribution of values across all the values in the admissible range (2.4b). The label $\frac{n_{\text{points}}}{n_{\text{total}}}$ represents the ratio between the number of χ values selected within the discretized intervals and the total number of evaluations, i.e., the total number of optimization steps. This analysis provides statistical insight into the most and least frequently selected values of χ . The corresponding velocity and pressure fields at $t = T = 4$ for the χ -**Opt-EFR** algorithms are represented in Figure 2.5.

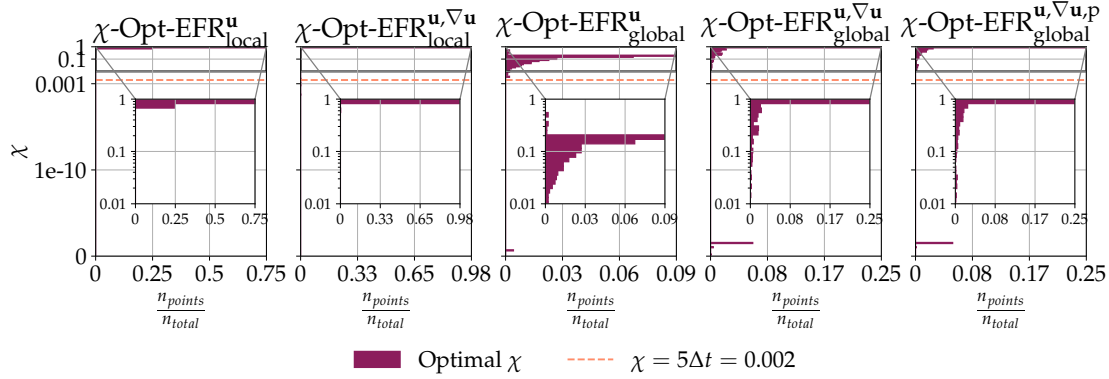
When minimizing the local **MSE** discrepancy, the optimization converges to ≈ 1 for the majority of the time, both when the gradients discrepancy is included (χ -**Opt-EFR**_{local}^u) or not (χ -**Opt-EFR**_{local}^{u, ∇u}) into the objective function. This means that the algorithm retains only the filtered velocity (**EF**), and, hence, we expect an overdiffusive behaviour, reproducing the standard **EF** solution. The velocity fields in Figures 2.5a and 2.5c confirm the overly diffusive dynamics.

When we only consider the energy discrepancy in the objective function (χ -**Opt-EFR**_{global}^u), the optimal χ values are smaller than in the local approach, but the simulation blows up at $t = 1.6$, as can be seen from the spurious oscillations in the velocity field of Figure 2.5e. On the other hand, adding the velocity gradients' (χ -**Opt-EFR**_{global}^{u, ∇u}) and the pressure (χ -**Opt-EFR**_{global}^{u, ∇u, p}) contributions to the loss is beneficial for the optimization: the corresponding velocity and pressure fields in Figures 2.5g, 2.5h, 2.5i, and 2.5j show a good agreement with the reference solutions (Figures 2.2a and 2.2b), and capture the vortex

shedding and the flow dynamics. However, the last two plots of Figure 2.4a show an **oscillating** and **chaotic** behaviour of $\chi(t)$, and Figure 2.4b shows that the number of $\chi(t)$ optimal elements close to 1 is around the 25% of the total selected values, in both algorithms χ -Opt-EFR $_{\text{global}}^{\text{u},\nabla\text{u}}$ and χ -Opt-EFR $_{\text{global}}^{\text{u},\nabla\text{u},\text{p}}$. This means the optimization converges statistically more often to values close to 1, i.e., skipping the relaxation step.



(a) Optimal χ values in χ -Opt-EFR.



(b) Observation counts of χ optimal values in χ -Opt-EFR.

Figure 2.4: Flow past a cylinder test case. Optimal values of χ in time, and distribution of χ across all values in the admissible range $[0, 1]$, for the χ -Opt-EFR algorithms.

We now start analysing the relative L^2 velocity and pressure errors, namely:

$$E_{\text{u}}(t_n) = \frac{\|\mathbf{u}_n - \mathbf{u}_n^{\text{ref}}\|_2}{\|\mathbf{u}_n^{\text{ref}}\|_2}, \quad E_{\text{p}}(t_n) = \frac{\|p_n - p_n^{\text{ref}}\|_2}{\|p_n^{\text{ref}}\|_2}, \quad (2.3)$$

which are displayed in Figure 2.6. The χ -Opt-EFR approaches provide similar L^2 errors, except for the χ -Opt-EFR $_{\text{global}}^{\text{u}}$, whose error drastically increases after $t = 1$, leading to the simulation blow up. The simulation stops because the optimization algorithm for parameter χ is highly unstable and does not converge to a global optimum. The local approaches (having an overdifusive solution) and the global approaches χ -Opt-EFR $_{\text{global}}^{\text{u},\nabla\text{u}}$ and χ -Opt-EFR $_{\text{global}}^{\text{u},\nabla\text{u},\text{p}}$ (which provide a good agreement with the reference) have substantially comparable error. However, the graphical comparison in Figure 2.2 highlights that the global approaches including the gradient contribution provide significantly better results than the local approaches. Hence, we can conclude that the L^2 relative error is not

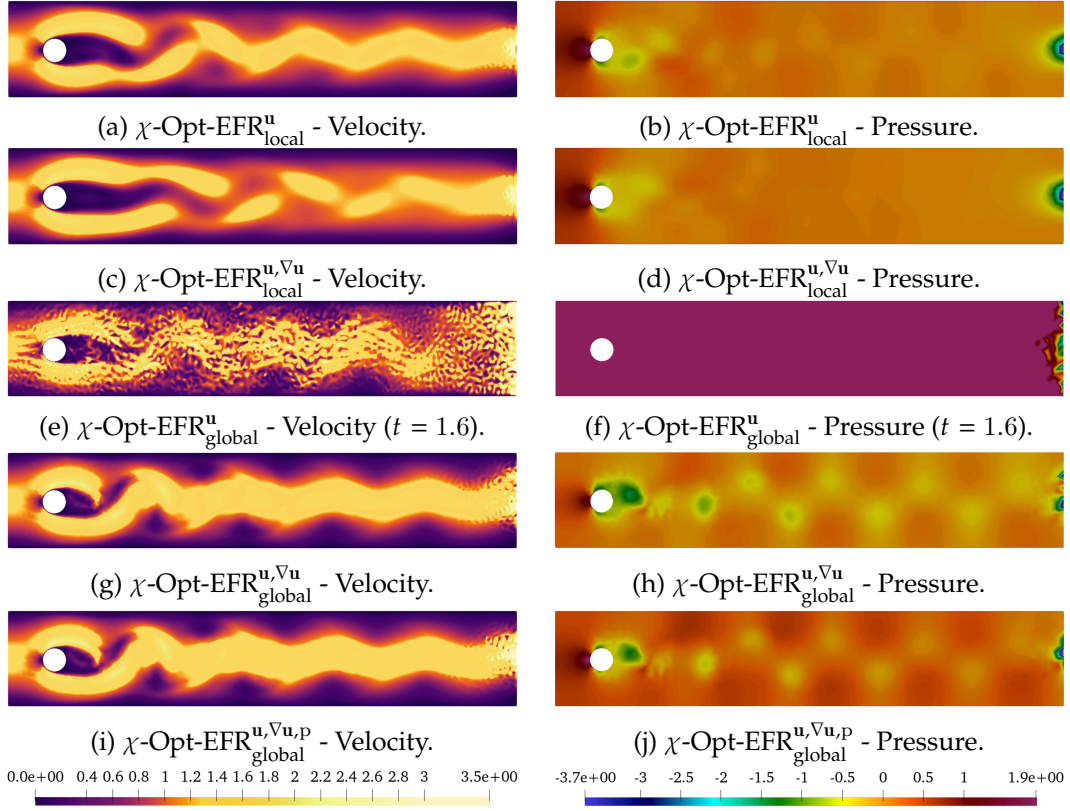


Figure 2.5: Flow past a cylinder test case. Velocity magnitude and pressure fields for the χ -Opt-EFR simulations, at the final time $T = 4$. The solutions for χ -Opt-EFR^u_{global} are represented at $t = 1.6$, as the simulation blows up at later time.

a significant metric to compare our approaches. For this reason, in the next sections we will not compare the solutions in terms of relative errors, but in terms of global metrics, namely $\|\mathbf{u}\|_2$, $\|\nabla \mathbf{u}\|_2 = \|\mathbf{u}\|_{H^1}$, and $\|p\|_2$, which are displayed in Figure 2.7 for the χ -Opt-EFR approaches.

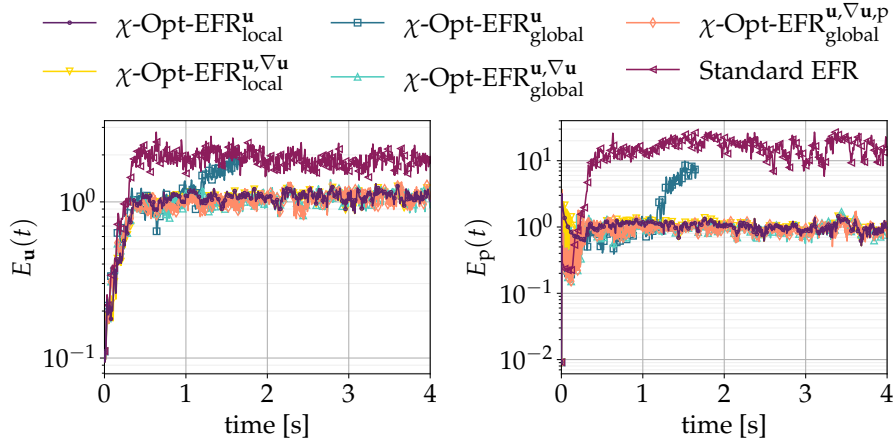


Figure 2.6: Flow past a cylinder test case. Time evolution of the velocity and pressure relative L^2 errors with respect to the reference solutions. The errors are displayed for all the χ -Opt-EFR algorithms considered, for noEFR, and standard EFR approaches.

From Figure 2.7, we can finally quantitatively see the dissipativity of χ -Opt-EFR $_{\text{local}}^{\mathbf{u}}$ and χ -Opt-EFR $_{\text{local}}^{\mathbf{u}, \nabla \mathbf{u}}$, which underestimate all the norms considered. The χ -Opt-EFR $_{\text{global}}^{\mathbf{u}}$, while exploding in $\|\nabla \mathbf{u}\|_2$ and $\|p\|_2$, is the most accurate in the velocity norm until $t = 1.6$. This happens because the discrepancy in the velocity norm is exactly the minimized objective function of the optimization process. The other two global approaches (χ -Opt-EFR $_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}}$ and χ -Opt-EFR $_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}, p}$) exhibit similar results in all the norms considered and are the most accurate with respect to the reference. The velocity gradients' norm for such approaches is oscillating more than the reference ones: this may be related to the data-driven nature of the optimization, which is only performed every 10 time steps. Moreover, we would expect the χ -Opt-EFR $_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}, p}$ to capture the pressure field better than χ -Opt-EFR $_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}}$. However, the χ -Opt-EFR $_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}}$ and χ -Opt-EFR $_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}, p}$ are comparable, and the addition of the pressure contribution only slightly increases the pressure accuracy (Figures 2.5h and 2.5j). Finally, as we expect, the standard EFR and the noEFR norms are larger than the reference values, as both simulations exhibit spurious oscillations.

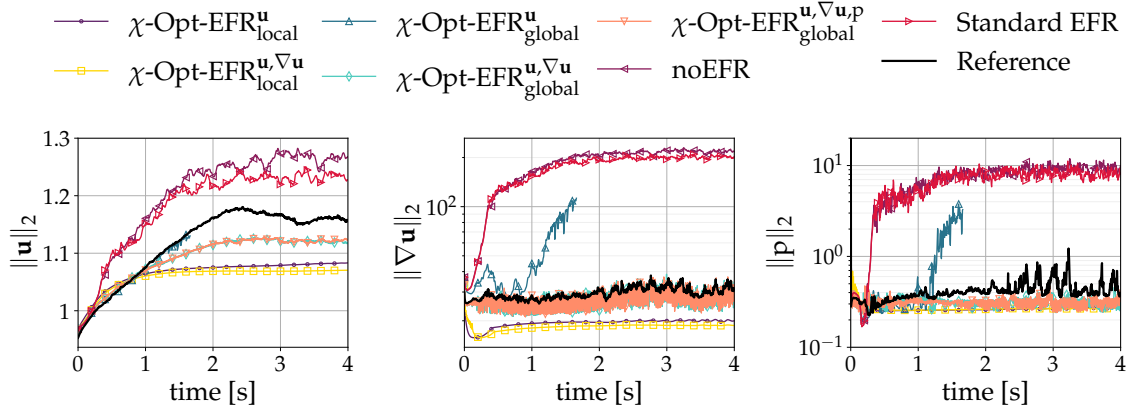


Figure 2.7: *Flow past a cylinder test case.* Time evolution of the velocity, velocity gradients and pressure L^2 norms, for all the χ -Opt-EFR algorithms considered, for noEFR, standard EFR and for the reference simulation.

In conclusion, for the χ -Opt-EFR approach we can state that the local approaches χ -Opt-EFR $_{\text{local}}^{\mathbf{u}}$ and χ -Opt-EFR $_{\text{local}}^{\mathbf{u}, \nabla \mathbf{u}}$ leads to poor and overdissipative solutions. For this reason, we will not consider the local optimization approach in the δ -Opt-EF and $\delta\chi$ -Opt-EFR algorithms.

2.3.2 Numerical results of δ -Opt-EF

This part is dedicated to the δ -Opt-EF procedure, namely the optimization of the $\delta(t)$ value, at fixed $\chi = 1$. As already mentioned, we only consider the global optimization approaches in this subsection, namely δ -Opt-EF $_{\text{global}}^{\mathbf{u}}$, δ -Opt-EF $_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}}$, and δ -Opt-EF $_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}, p}$.

Figure 2.8 shows the optimal δ values in time, and the observation counts at different discretized intervals in the range $[1 \times 10^{-5}, 1 \times 10^{-3}]$, relative to the global number of evaluations.

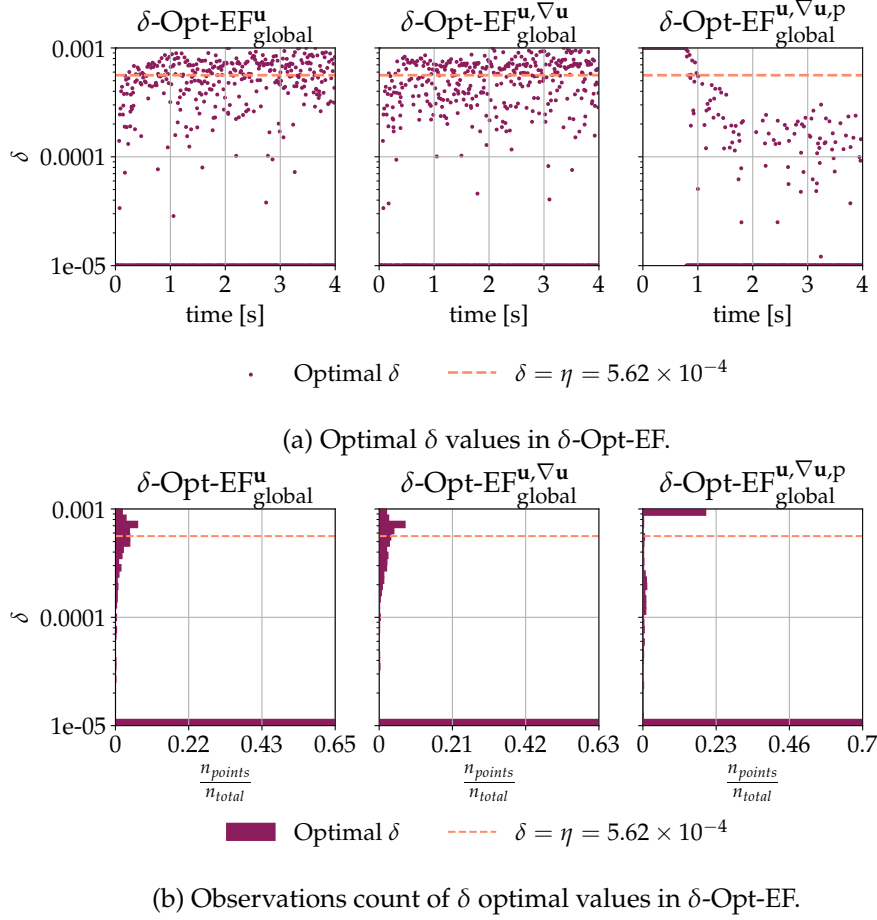


Figure 2.8: Flow past a cylinder test case. Optimal values of δ in time, and distribution of δ across all values in the admissible range $[1 \times 10^{-5}, 1 \times 10^{-3}]$, for the δ -Opt-EF algorithms.

The results here are similar to the results obtained for the χ -Opt-EFR $_{\text{global}}$ approaches in Subsection 2.3.1, namely $\delta(t)$ is oscillating in time. In addition, in the δ -Opt-EF $_{\text{global}}^{u,\nabla u,p}$ algorithm, the optimal parameter converges to its largest value 1×10^{-3} in the initial time window, and then it oscillates but most of the time converges to its smallest value 1×10^{-5} . Figure 2.8b shows that the most frequently occurring value is 1×10^{-5} .

The time evolution of the norms in Figures 2.10, and the graphical results at $t = 4$ in Figure 2.9 lead to the following conclusions:

- The optimized algorithms δ -Opt-EF $_{\text{global}}$ give a better representation of the reference fields, outperforming the over-diffusive approximation of standard EF with fixed δ ;
- The L^2 norms show that the most accurate result is obtained with the δ -Opt-EF $_{\text{global}}^u$ algorithm, which directly includes the measured metric inside the objective function. However, this algorithm provides inaccurate results, as seen both from the other norms' behaviour, and from the graphical visualizations (Figures 2.9a and 2.9b), which show noisy and poor approximations;
- The δ -Opt-EF $_{\text{global}}^{u,\nabla u}$ and the δ -Opt-EF $_{\text{global}}^{u,\nabla u,p}$ approaches are more accurate than the

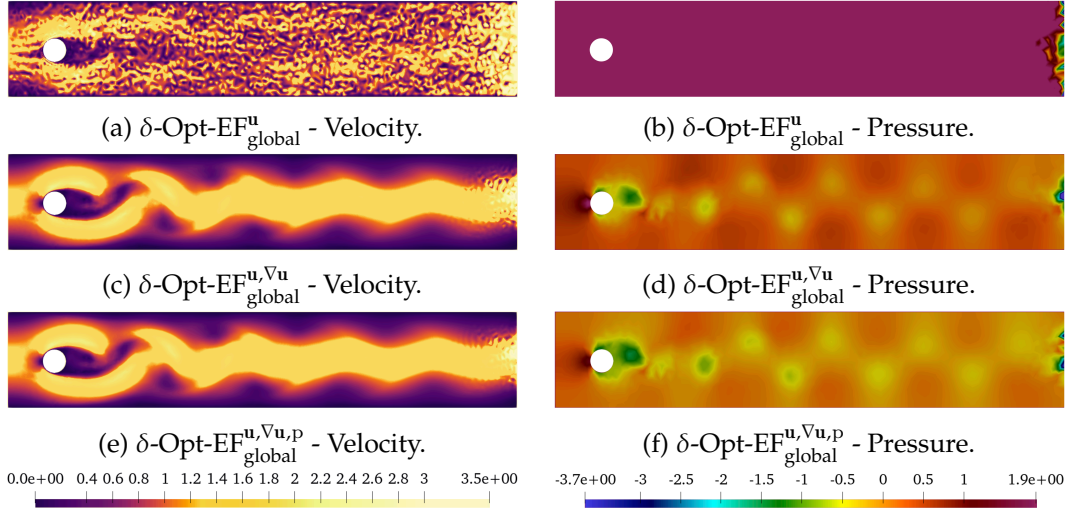


Figure 2.9: *Flow past a cylinder test case.* Velocity magnitude and pressure fields for the δ -Opt-EF simulations, at the final time $T = 4$.

δ -Opt-EF $^u_{\text{global}}$ in the velocity gradients and pressure norms. The δ oscillating behaviour between this minimum value and higher values is, hence, necessary to alleviate the spurious oscillations, which appear in the δ -Opt-EF $^u_{\text{global}}$ solution. The qualitative solutions (Figures 2.9c, 2.9d, 2.9e, and 2.9f) confirm a good agreement with the reference DNS;

- As noticed for χ -Opt-EFR, the addition of the pressure contribution δ -Opt-EF $^{u,\nabla u,p}_{\text{global}}$ does not substantially improve the pressure accuracy and negatively impacts the computational cost of the method, as we will analyze in detail in Section 2.3.4.

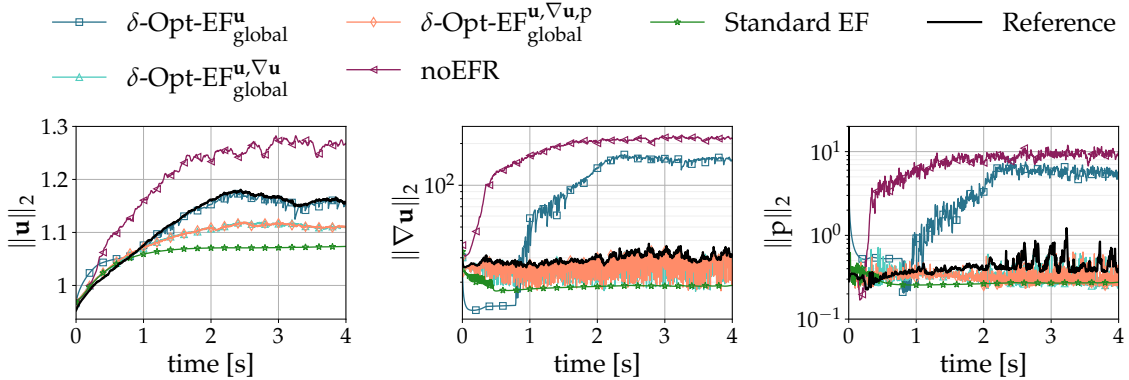


Figure 2.10: *Flow past a cylinder test case.* Time evolution of the velocity, velocity gradients and pressure L^2 norms, for all the δ -Opt-EF algorithms considered, for noEFR, standard EF and for the reference simulation.

2.3.3 Numerical results of $\delta\chi$ -Opt-EFR

This part shows the results obtained with a double optimization of $\delta(t)$ and $\chi(t)$ ($\delta\chi$ -Opt-EFR algorithms). Since both algorithms χ -Opt-EFR $_{\text{global}}^u$ and δ -Opt-EFR $_{\text{global}}^u$ proved to be inaccurate in subsections 2.3.1 and 2.3.3, we only consider here $\delta\chi$ -Opt-EFR $_{\text{global}}^{u,\nabla u}$ and $\delta\chi$ -Opt-EFR $_{\text{global}}^{u,\nabla u,p}$.

We repeat in Figure 2.11 the study on the time evolution of the optimal parameters, and the statistical distribution of the selected values. The optimal $\delta(t)$ reproduces a similar trend and statistics in both the algorithms: it oscillates in the admissible range, and the most common value is 1×10^{-5} , being the optimal one $\approx 65 - 73\%$ of the time. However, the relaxation parameter $\chi(t)$ assumes statistically larger values in δ -Opt-EFR $_{\text{global}}^{u,\nabla u,p}$, being almost always in the range $[1 \times 10^{-1}, 1]$. When the pressure contribution is not included, the relaxation parameter is more oscillating, and also converging to smaller values ($\approx 1 \times 10^{-4}$) at some time instances.

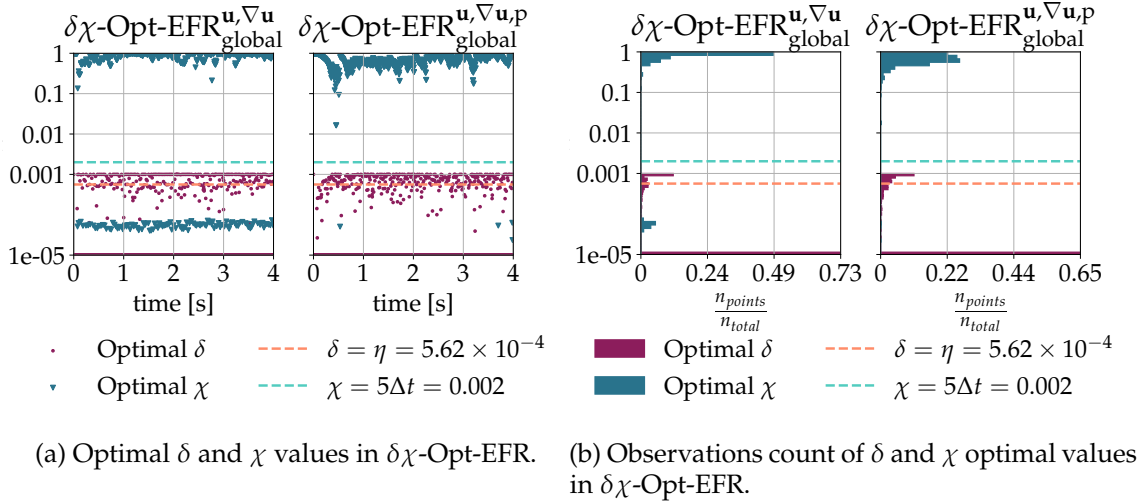


Figure 2.11: *Flow past a cylinder test case*. Optimal values of δ and χ in time, and distribution of δ and χ across all values in the admissible range, for the $\delta\chi$ -Opt-EFR algorithms.

The norms behaviour in Figure 2.12 and the graphical fields in Figure 2.13 confirm that both algorithms accurately reproduce the flow dynamics, and slightly differ in the pressure reconstruction. As one would expect, the pressure norm appears more stable and less oscillating when adding the pressure contribution in the loss functional ($\delta\chi$ -Opt-EFR $_{\text{global}}^{u,\nabla u,p}$).

2.3.4 Comparison and discussion of the Opt-EFR results

This subsection compares the results of the best-performing algorithm for each proposed method, namely the $(\cdot)_{\text{global}}^{u,\nabla u,p}$ class of methods. Hence, we answer the question:

Is it better to optimize the filter width, the relaxation parameter, or both?

Figure 2.14 shows a comparison among the different optimization types in the velocity and pressure norms. The results are overall comparable, and χ -Opt-EFR is slightly more

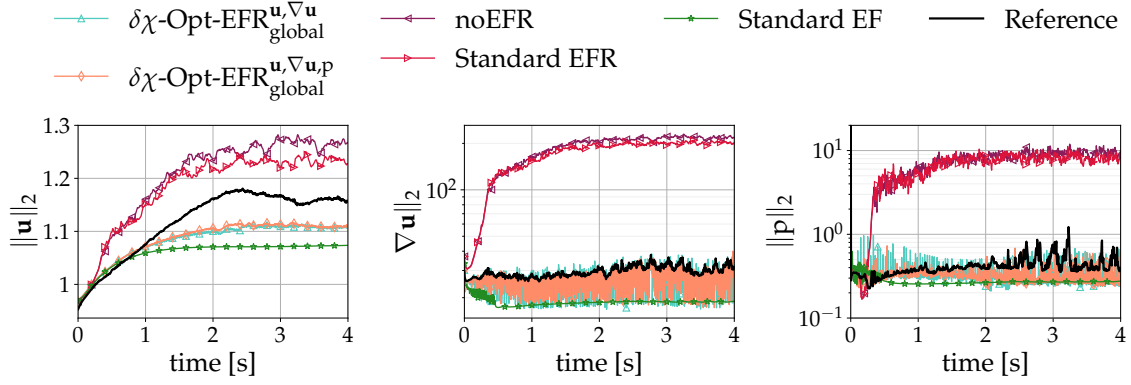


Figure 2.12: *Flow past a cylinder test case.* Time evolution of the velocity, velocity gradients and pressure L^2 norms, for all the $\delta\chi$ -Opt-EFR algorithms considered, for noEFR, standard EFR, standard EF, and for the reference simulation.

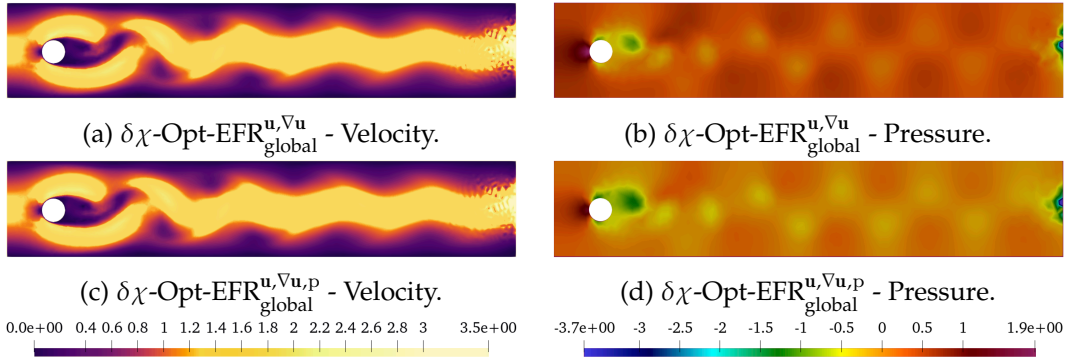


Figure 2.13: *Flow past a cylinder test case.* Velocity magnitude and pressure fields for the $\delta\chi$ -Opt-EFR simulations at the final time $T = 4$.

accurate in the velocity norm. χ -Opt-EFR also provides less oscillating velocity gradients and pressure norms.

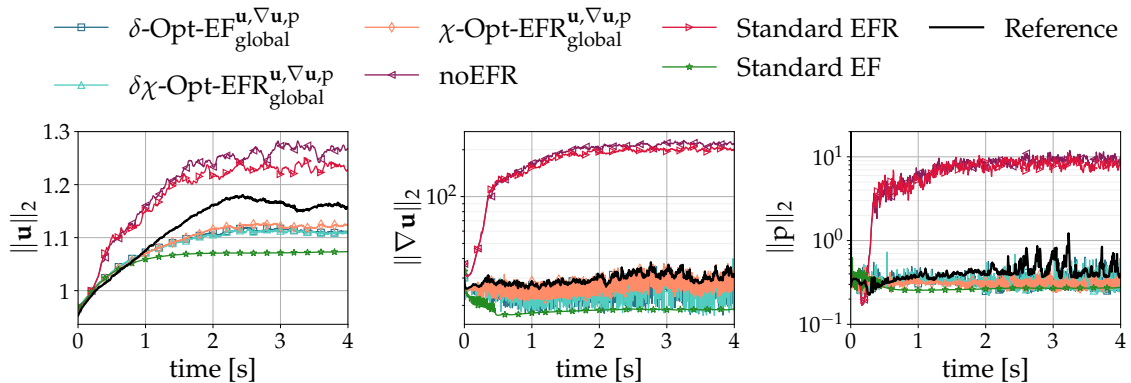


Figure 2.14: Time evolution of the velocity, velocity gradients and pressure L^2 norms, for the best performing algorithms of χ -Opt-EFR, δ -Opt-EF, and $\delta\chi$ -Opt-EFR. The results are compared with noEFR, standard EFR, standard EF, and the reference simulation.

As an overview, we include two additional Figures:

- Figure 2.15 represents the time-average (orange lines) and confidence interval (boxes) of the relative errors in the velocity H^1 norm and pressure L^2 norm. It confirms a significant improvement in the **Opt-EFR** strategies with respect to standard **EFR** and **EF**.
- Figure 2.16 displays the increase in the accuracy of **Opt-EFR** results with respect to the baselines. The “gain” metric is measured as the improvement in the time-averaged relative error of the **Opt-EFR** solutions with respect to standard **EFR** and standard **EF** results. **Opt-EFR** strategies significantly enhance the accuracy of standard methods. In particular, the **Opt-EFR** solution is 99 % more accurate than standard **EFR**, while it is 83 % (velocity) and 27-30 % (pressure) more accurate than standard **EF**.

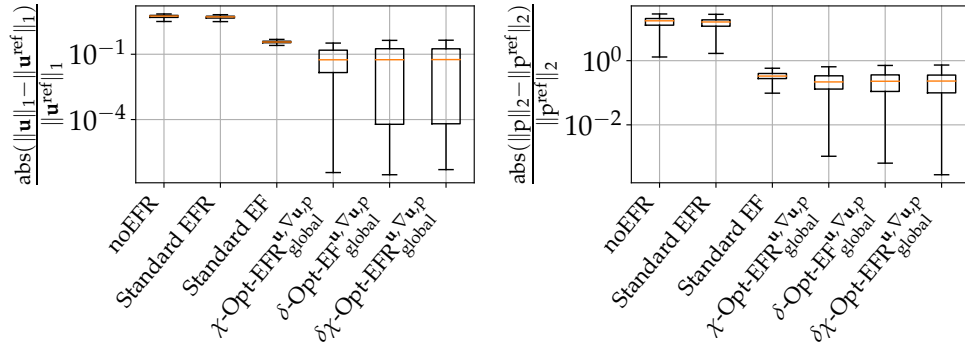


Figure 2.15: *Flow past a cylinder test case*. Statistics (median and confidence intervals) for noEFR, standard EFR/EF, and methods of type $(\cdot)_{\text{global}}^{\mathbf{u}, \nabla \mathbf{u}, p}$. The results display the relative errors in the H^1 velocity norm, and in the L^2 pressure norm, with respect to the reference. The plots show the time-averaged error (orange line) and the confidence interval (boxes). The statistics are computed within the time window $[0, 4]$.

Remark: on the incompressibility constraint.

A final curiosity we address here is the following:

*Is the incompressibility constraint satisfied, even if we do not impose it explicitly in the **DF**?*

The answer is that, in general, the incompressibility constraint is not satisfied, unless periodic boundary conditions are applied [35, 188, 339]. Typically, if the relaxation parameter is small ($\chi \approx \Delta t$), the **EFR** simulation is characterized by low values of the discretized velocity divergence, namely $\nabla \cdot \mathbf{u} \approx 0$, because the evolved velocity is divergence-free and its contribution dominates in the convex combination. However, the constraint should be imposed in **EF** simulations or when $\chi \approx 1$. A technique to impose divergence freeness is the so-called **grad-div stabilization** [77, 156, 187], which consists in penalizing the violation of incompressibility by adding

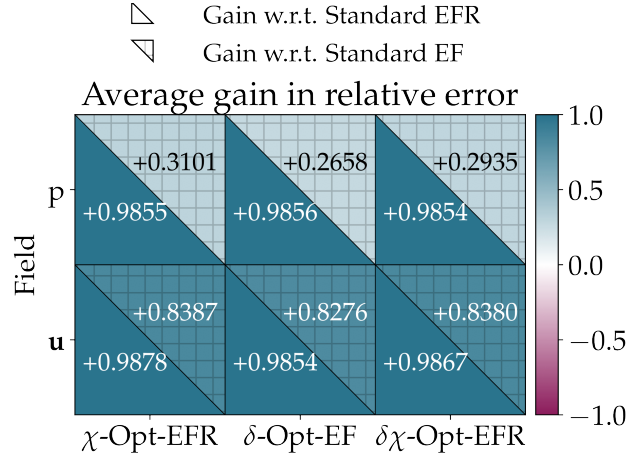


Figure 2.16: *Flow past a cylinder test case*. Gain in the accuracy, considering the H^1 velocity and L^2 pressure relative errors. The results show the time-average gain of the best Opt-EFR methods with respect to the standard EFR and EF simulations.

in the **DF** a term of the form:

$$\gamma_{\text{grad-div}} \nabla(\nabla \cdot \bar{\mathbf{w}}_n),$$

where $\gamma_{\text{grad-div}}$ is a parameter to be calibrated depending on the test case. In the supplementary material of [147], we perform the **δ-Opt-EF** algorithm in the presence of the grad-div stabilization, using $\gamma_{\text{grad-div}} = 100$, as in [126]. Such strategy leads to similar solutions to those obtained without stabilizing the **DF**.

2.3.4.1 On the optimization step

Here, we address the question:

*What is the influence of the optimization step Δt on the accuracy and efficiency of the **Opt-EFR** simulation?*

The previous analysis was performed with a fixed optimization step, $\Delta t_{\text{opt}} = 10$. However, the optimization step plays a crucial role in achieving a good balance between accuracy and efficiency in the proposed approach. We then compare the **δχ-Opt-EFR** strategies in terms of **efficiency** and **accuracy**, at different optimization steps, namely $\Delta t_{\text{opt}} = N_{\text{opt}} \Delta t$, with N_{opt} positive integer in $\{10, 20, \dots, 90, 100\}$. The **accuracy** is measured via the time-average of the metrics $\mathcal{L}_{\text{global}}^u$, $\mathcal{L}_{\text{global}}^{\nabla u}$ and $\mathcal{L}_{\text{global}}^p$, defined in Section 2.2.1. The notation $\overline{(\cdot)}^n$ indicates the time-average. The **efficiency** is assessed in terms of wall-clock time, expressed both as a percentage relative to the **DNS** time (bottom x -axis) and as absolute time (top x -axis). The results of the **δχ-Opt-EFR** simulations with different values of N_{opt} are presented in Pareto plots (see Figure 2.17), where each marker is annotated with the corresponding value of N_{opt} .

Since the x -axis represents computational time and the y -axis represents the error, the

best-performing simulations lie in the lower-left region of the plots, corresponding to low error and low computational cost. Conversely, simulations located in the upper-left region (high error, low cost) and in the lower-right region (low error, high cost) are considered suboptimal.

Figure 2.17 leads to the following observations:

- ▶ Standard EFR and noEFR simulations yield higher errors than Opt-EFR, as expected. In particular, for pressure and velocity gradients, the $\delta\chi$ -Opt-EFR improves accuracy by up to **two** orders of magnitude. Moreover, standard EFR and noEFR simulations may also require more computational time than some $\delta\chi$ -Opt-EFR configurations. This is likely due to their **instability and noise**, which increase the number of iterations required by the FEM Newton solver to reach convergence.
- ▶ In terms of relative wall-clock time, all Opt-EFR simulations exhibit a **significant speedup** with respect to the DNS. However, in absolute terms, their computational time lies in the range [4, 9] hours. This indicates that varying the optimization time window can lead to differences in computational cost on the order of several hours.
- ▶ As expected, simulations with smaller optimization steps (e.g., $10\Delta t$ or $20\Delta t$) are more accurate but also more computationally expensive than those with larger steps. While the accuracy across different optimization steps remains comparable in terms of L^2 norms, notable differences arise in the H^1 velocity seminorm. This suggests that velocity gradients may drift significantly from the global optimum before a new optimization is performed, making subsequent corrections less effective.
- ▶ The $\delta\chi$ -Opt-EFR $_{\text{global}}^{\text{u},\nabla\text{u},p}$ strategy provides a slight improvement in pressure accuracy. However, it is more computationally expensive than $\delta\chi$ -Opt-EFR $_{\text{global}}^{\text{u},\nabla\text{u}}$ without delivering a substantial gain in overall accuracy.

The same analysis varying Δt_{opt} can be done for the other Opt-EFR strategies. For more details, we refer the reader to Appendix B of [147].

2.4 Conclusions and motivations to RL

In this work, we advanced the EFR regularization approach by introducing a **data-driven strategy for the automated tuning of the filter and relaxation parameters**, δ and χ . While existing studies provide heuristic guidelines, they typically rely on constant or problem-dependent parameter choices. In contrast, the proposed Opt-EFR framework dynamically adapts these parameters in time based on some pre-collected reference data, improving robustness and accuracy in marginally-resolved regimes.

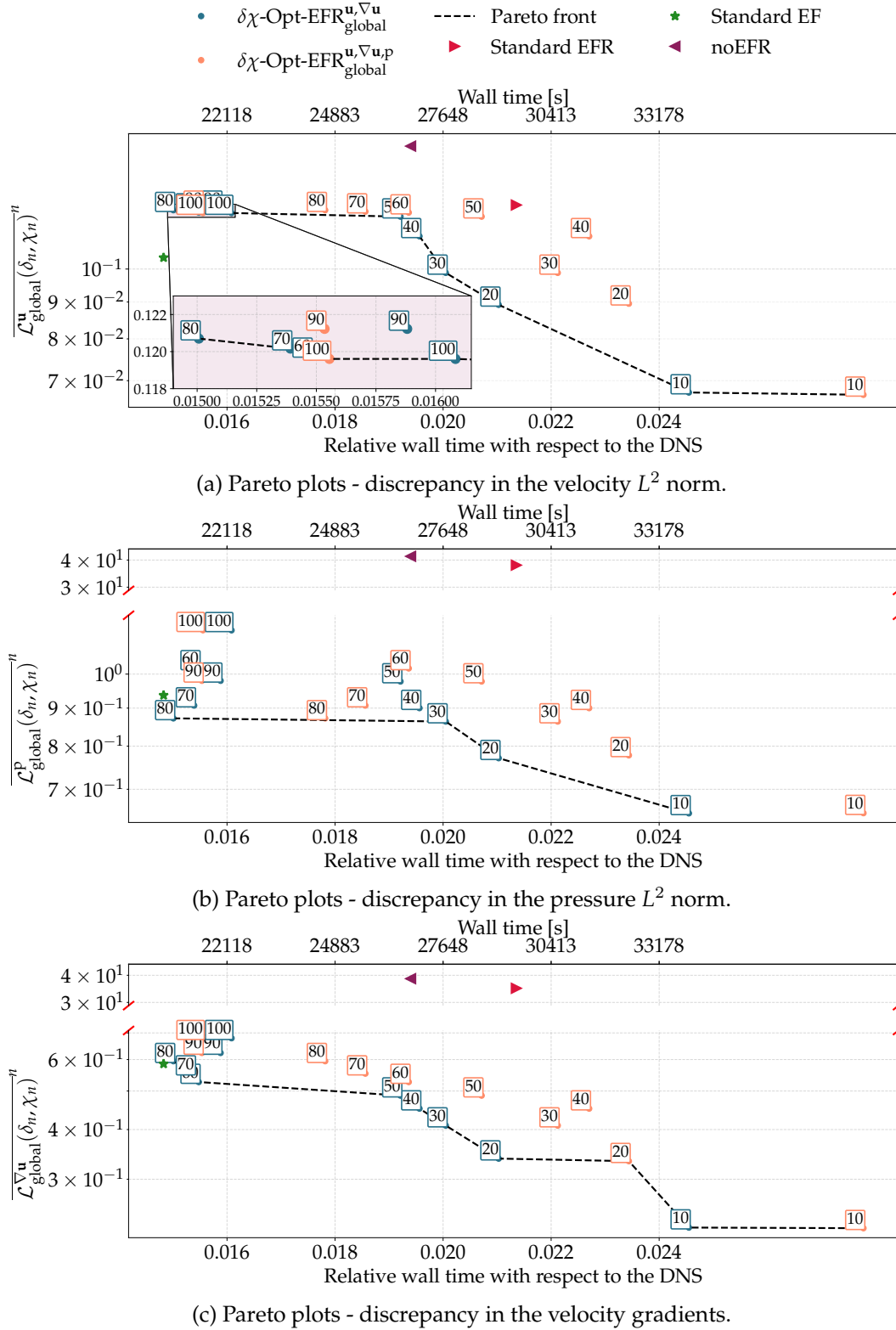


Figure 2.17: Flow past a cylinder test case. Representation of how the optimization step affects the efficiency and accuracy of the velocity and pressure solution in $\delta\chi$ -Opt-EFR $^{\mathbf{u}, \nabla \mathbf{u}}_{\text{global}}$ and $\delta\chi$ -Opt-EFR $^{\mathbf{u}, \nabla \mathbf{u}, \mathbf{p}}_{\text{global}}$ algorithms. The efficiency is represented in terms of the wall clock time. The accuracy is measured in terms of the time-average discrepancy in the velocity (2.17a), pressure (2.17b) and velocity gradients' (2.17c) norm. The numbers identify the number of time steps between two optimizations (N_{opt}).

We investigated three optimization strategies: χ -Opt-EFR, δ -Opt-EF, $\delta\chi$ -Opt-EFR, optimizing either δ or χ or both parameters. The objective functions are defined to minimize the discrepancy between the marginally-resolved Opt-EFR simulation and a reference DNS. The numerical results for the flow past a cylinder at $Re = 1000$ lead to the following main conclusions:

- ▶ Opt-EFR consistently outperforms both noEFR and standard EFR/EF.
- ▶ **Global objective functions** (e.g. energy-related) provide more reliable results than local ones (e.g., MSE), and help in correctly capturing the flow dynamics.
- ▶ Including **velocity gradients** in the objective function is crucial to achieve the highest accuracy.
- ▶ The optimized approach achieves **comparable or reduced computational time** with respect to standard approaches, thanks to the improved numerical stability and the faster convergence of the Newton solver.

The main limitation of the approach is its reliance on reference data and its problem-dependent nature, as the optimization must be repeated for different configurations.

As an outlook, the use of **reinforcement learning** to learn optimal parameter policies offline is addressed in Chapter 3, with a focused study on the tuning of δ within the EF framework. This choice is justified by the results of the present Chapter, which clearly show that the proposed algorithms χ -Opt-EFR, δ -Opt-EF, and $\delta\chi$ -Opt-EFR achieve essentially equivalent performance. Therefore, the following Chapter will focus on the dynamic selection of δ , which is controlled by a RL agent, trained either on representative DNS data or on physics-based information from the ongoing simulation. This enables real-time parameter adaptation without requiring repeated optimization procedures, and allowing for **time extrapolation**.

RL-EF: ADAPTIVE FILTERING VIA REINFORCEMENT LEARNING

This chapter aims to introduce an adaptive version of the **Opt-EFR** algorithm, enabling its application in predictive and time-extrapolation regimes.

The previous chapter proposed the optimization of the **EFR** parameters independently at each time step, heavily relying on the **DNS** reference data and without predictive capability. This preliminary investigation inspired the work [150] presented in this chapter, where the filter parameter is not optimized independently at each time step, but **RL** strategies are employed to learn a time-dependent control policy to properly select the filter value.

RL encompasses a family of machine learning approaches in which an agent learns a policy through direct interaction with an environment, optimizing for a prescribed cumulative reward [351, 158], typically related to the discrepancy with respect to some reference data. Unlike supervised learning, which relies on labeled input–output pairs, **RL** derives optimal actions through sequential trial-and-error exploration. This makes it especially well-suited for control and decision-making tasks where each action shapes subsequent states, providing a natural formulation for time-dependent parameter selection. Within **CFD**, **RL** has seen growing adoption across a range of applications, including flow control, turbulence modeling, subgrid-scale closure, and adaptive numerical stabilization [183, 182, 181, 264, 265, 327, 95, 87, 342, 271, 247]. Collectively, these studies highlight **RL**’s ability to extract nontrivial control strategies directly from high-dimensional flow states, even in strongly nonlinear regimes. Building on these capabilities, the present work employs **RL** to dynamically adjust the filtering intensity within the **EF** framework, recasting parameter selection as an adaptive control problem governed by the evolving flow dynamics.

The key novelties with respect to previous **EF** (or **EFR**) approaches are the following:

- ▶ the introduction of the novel **Reinforcement Learning-based Evolve-Filter (RL-EF) algorithm**, enabling a non-trivial adaptive adjustment of the filter radius in time;
- ▶ a **reference data-free** version of **RL-EF** that learns the filter radius without relying on reference **DNS** data, but only on physics-based indicators, like the equations’ residual, gradient-based measures, or structure-preserving quantities.

We will investigate the performance of **RL-EF** on two 2D test cases: the flow past a cylinder at $Re = 1000$, and the decaying homogeneous isotropic turbulence (**HIT**) in a square

domain at $Re = 4 \times 10^4$. In both cases, our analysis is mainly focused on improving the velocity approximation rather than the pressure one.

This chapter is organized as follows:

- ▶ Section 3.1, which shows the results of the standard EF approach, compared to the reference DNS and to noEF;
- ▶ Section 3.2, which presents the basic knowledge of the RL methods used in this work (subsection 3.2.2), and of our RL-EF algorithm (subsection 3.2.3);
- ▶ Section 3.3, highlighting the results achieved with the novel approach on the two test cases considered;
- ▶ Section 3.4, which draws the conclusions for this work, motivating further extensions.

3.1 The two test cases in FEniCSx: standard EF results

The first test case is the flow past a cylinder at $Re = 1000$, which follows the same setting as described in Section 2.1, namely the same initial and boundary conditions, time stepping and the same FEM spaces. However, in order to couple the FEM solver with a RL environment, we used a different version of the software, FEniCSx [21]. We slightly changed the marginally-resolved and well-resolved meshes, which are represented in Figure 3.1.

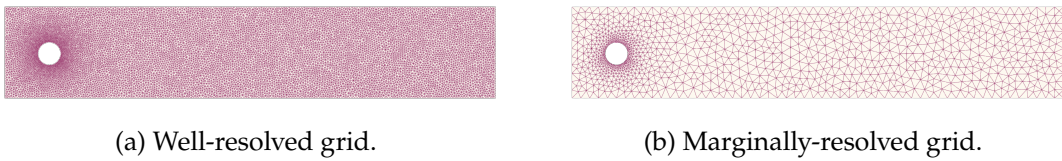


Figure 3.1: *Flow past a cylinder test case.* grids considered for the DNS and for the marginally-resolved simulation in FEniCSx.

The second test case is the 2D decaying HIT at $Re = 40\,000$ on a square domain ($\Omega = [0, 1]^2$), as introduced in section 1.3.1. We follow the same initial and periodic boundary conditions as section 1.3.1, but here we discretize the equations using a FEM in FEniCSx. Differently from the flow past a cylinder, we employ quadrilateral elements on a 256^2 (well-resolved) grid, and on a 64^2 (marginally-resolved) grid. We consider a backward differentiation formula of order 1 (BDF1) as a time integration scheme, and standard $\mathbb{P}^2 - \mathbb{P}^1$ elements for the velocity-pressure coupling. The simulation is performed within the time window $(0, 2]$, i.e., $t_0 = 0$ and $T = 2$, while the time step is fixed and $\Delta t = 1 \times 10^{-3}$ in all the simulations.

Table 3.1 presents the exact number of degrees of freedom for the velocity and pressure fields in both test cases and for both the grids considered, together with the amount of time needed to perform the simulations on both grids¹.

Table 3.1: Degrees of freedom (dofs) for the velocity and pressure fields and wall times needed to perform 4 seconds of simulation, for both the test cases taken into account, and for both the marginally-resolved and well-resolved grids.

Test case	Grid	Velocity dofs	Pressure dofs	Wall time
Flow past a cylinder	Marginally-resolved	9792	1274	≈ 1 hour 4 minutes
	Well-resolved	43609	11036	≈ 4 hours 15 minutes
Decaying HIT	Marginally-resolved	16641	4225	≈ 2 hours 9 minutes
	Well-resolved	263169	66049	≈ 20 hours 42 minutes

Before showing the standard EF and noEF results, it is important to specify that, differently from Chapter 2, we restrict our investigation to the calibration of the filter width $\delta(t)$, and do not consider the relaxation step, namely $\chi = 1$ and $\mathbf{u}_n = \bar{\mathbf{w}}_n$ at each time step t_n . Moreover, regarding the choice of the filtering strategy, differently from the DF used in the previous chapter, we impose here the incompressibility constraint by using a Stokes DF [86, 188], whose continuous form can be found in System (1.10). More precisely, the filter step $\mathbf{u}_n = \mathcal{F}(\mathbf{w}_n)$ at time step t_n can be written as:

$$\begin{cases} -2\delta^2 \Delta \mathbf{u}_n + \mathbf{u}_n + \nabla \pi_n^{\text{aux}} = \mathbf{w}_n, \\ \nabla \cdot \mathbf{u}_n = 0. \end{cases} \quad (3.1)$$

In the above expression, π_n^{aux} represents an auxiliary variable to guarantee the divergence-free constraint for $\mathbf{u}_n$.

Figures 3.2 and 3.3 show representative solutions in time of the velocity magnitude for the filtered DNS², the noEF and standard EF simulations, in the two test cases, respectively.

For the flow past a cylinder case, Figure 3.2 demonstrates that:

- noEF starts to exhibit spurious oscillations in the region nearby the cylinder at $t = 1$, and it blows up after $t = 1.5$;
- standard EF has an overdifusive behaviour, as it manifests a vortex shedding in the cylinder wake only at the end of the simulation, after $t = 2$.

For the HIT case, Figure 3.3 shows that:

¹Wall times refer to the simulations performed on the CINECA cluster Galileo G100, on 2 Intel Xeon Platinum nodes, of 48 threads each. The times refer to the noEFR (marginally-resolved grid) and DNS simulations (well-resolved grid).

²The filtered DNS is obtained by interpolating the DNS solution from the well-resolved grid to the coarser grid.

- ▶ The filtered **DNS** reference flow initially consists of small-scale vortical structures that gradually merge and grow over time, leading to the characteristic decay of turbulence and an overall dissipative behaviour. This evolution differs fundamentally from the flow past a cylinder. In the cylinder case, vortex shedding develops only after an initial transient phase, during which the kinetic energy increases before reaching a periodic regime. By contrast, in the decaying turbulence scenario, the kinetic energy decreases monotonically over time, and the dynamics are entirely governed by the progressive dissipation of structures across scales.
- ▶ The **noEF** solution becomes unstable and blows up after only a few time steps;
- ▶ In contrast, the standard **EF** solution closely follows the filtered **DNS** up to approximately $t = 0.5$. Beyond this point, however, it progressively diverges from the reference solution due to excessive diffusion. This over-diffusive behaviour is reflected in the vortical structures, which appear larger than those of the filtered DNS.

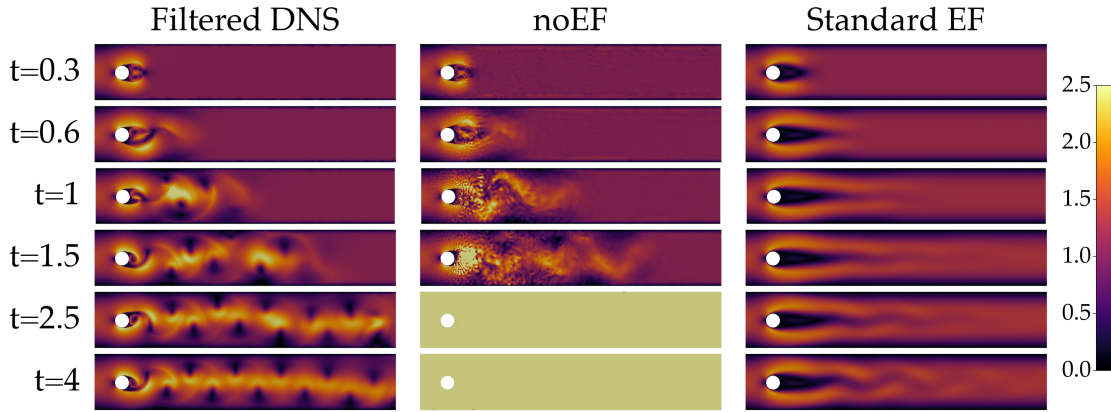


Figure 3.2: *Flow past a cylinder test case.* Time evolution of the velocity solution for the filtered DNS, standard EF, and noEF simulations.

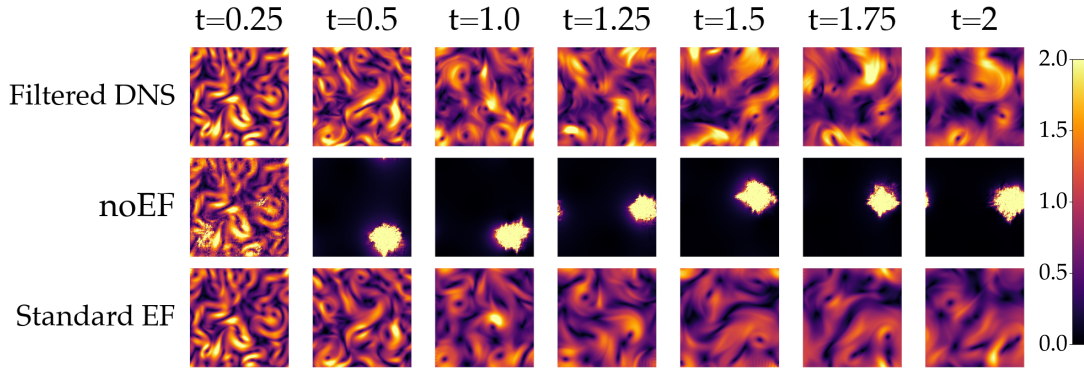


Figure 3.3: *Decaying HIT test case.* Time evolution of the velocity solution for the filtered DNS, standard EF, and noEF simulations.

Remark: on the simulations' software.

Regarding the flow past a cylinder test case, it is important to note that the difference in the computational time between the FEniCS (Table 2.1) and the FEniCSx (Table 3.1) simulations is due to different factors, like the different machines, the number of cores, the software used, and the computational grids. For instance, the well-resolved simulation used in this chapter has 43609 velocity nodes, and it is less refined than the one used in Chapter 2, which has 100676 velocity nodes.

Regarding the decaying HIT test case, it is worth highlighting that the simulation performed with FVM using the library [2] (Figure 1.5), exhibits spurious oscillations but the energy does not grow unbounded. This behaviour is attributed to the energy-conserving properties of the solver, which will be discussed in detail in Chapter 4.

The poor results obtained with the standard literature approaches motivate the use of RL, whose main theoretical background is introduced in Section 3.2.

3.2 Reinforcement Learning for the Evolve-Filter approach

A general and compact description of fully-connected neural networks and multi-layer perceptron is available in Section 3.2.1. Then, the basic knowledge on RL and on the specific algorithm we adopt (Deep Q-Network) is introduced in Section 3.2.2, while its application to the EF formulation (RL-EF) is presented in Section 3.2.3.

3.2.1 Neural networks

We here briefly recall the architecture of a **Multi-Layer Perceptron (MLP)**, a fully-connected basic **Neural Network (NN)** [277, 220]. We consider an input $\mathbf{i} \in \mathbb{R}^{d_{\text{in}} \times B}$, where B is the number of data samples (or *batch size*), and d_{in} is the dimensionality of the input. A fully-connected NN is composed of hidden layers, whose basic units are the *neurons*. The application of a single hidden layer, namely a linear operation, followed by a nonlinear function h (called *activation function*), is expressed as follows:

$$\mathbf{o} = h(\mathbf{W}\mathbf{i} + \mathbf{b}), \quad (3.2)$$

where $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and $\mathbf{b} \in \mathbb{R}^{d_{\text{out}}}$ are the weight matrix and bias vector associated with the hidden layer. The output $\mathbf{o}$ will then have dimension $d_{\text{out}} \times B$. This type of simple NN is typically referred to as *shallow NN*.

A sequential application of hidden layers (linear and nonlinear operations) is instead a *deep NN*. In this general case, we have L hidden layers with weight matrices $\mathbf{W}^{(l)}$ and bias $\mathbf{b}^{(l)}$, with $l = 1, \dots, L$. For example, considering $L = 2$, the output can be expressed as:

$$\mathbf{o} = h \left(\mathbf{W}^{(2)} \left(h(\mathbf{W}^{(1)}\mathbf{i} + \mathbf{b}^{(1)}) \right) + \mathbf{b}^{(2)} \right). \quad (3.3)$$

Figure 3.4 represents an example of deep neural network.

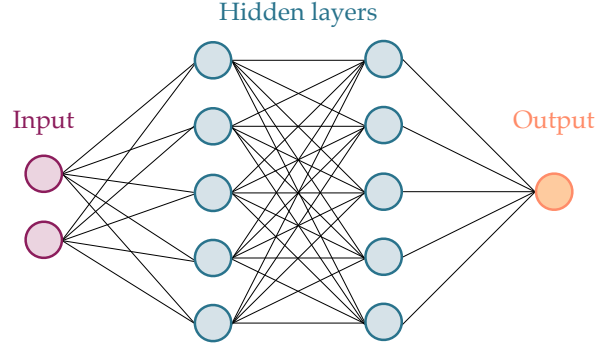


Figure 3.4: Basic illustration of a deep fully-connected neural network architecture (or multi-layer perceptron), with 2 inputs, 1 output, and 2 hidden layers of 5 neurons each.

A neural network is typically indicated with NN_{θ} , where θ is the set of learnable weights, namely $\mathbf{W}^{(l)}, \mathbf{b}^{(l)}, l = 1 \dots, L$. Such parameters are iteratively adjusted during the training process to minimize a predefined loss function. The training proceeds in two main steps: *forward propagation*, where inputs are passed through the network to produce predictions, and *backpropagation*, where the gradients of the loss with respect to each parameter are computed using the chain rule. These gradients are then used by an optimization algorithm, such as stochastic gradient descent or Adam [170], to update the parameters iteratively. Over many iterations (named *epochs*), the network gradually improves its performance by minimizing the loss on the training data.

An example of loss function in supervised learning is the **MSE** discrepancy between the given true data $\mathbf{o}'$ and the model output, namely:

$$\mathcal{L}(\theta) = \frac{1}{B} \sum_{b=1}^B \|\mathbf{o}_b - \mathbf{o}'_b\|_2^2, \text{ with } \mathbf{o} = \text{NN}_{\theta}(\mathbf{i}). \quad (3.4)$$

Other loss functions, like the L^1 loss or combinations of L^2 and L^1 discrepancies, are possible, as well as different optimizers. Other well-known optimizers are variants of Adam like AdamW [209], L-BFGS [204], SGD [11], SOAP [346].

3.2.2 An introduction to RL

We now analyze how **NNs** are used in **RL**.

RL is grounded in the principle of **learning by interaction**: an agent acquires decision-making capabilities through repeated engagement with a dynamic environment. This process is formally cast within the Markov Decision Process (MDP) framework, where an MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively, P is the state transition probability, r the reward function, and $\gamma \in (0, 1]$ the discount factor. The agent–environment interaction unfolds over *episodes*, each comprising a finite sequence of decision *steps*. At decision step t_n , the environment

presents a state $s_n = s(t_n) \in \mathcal{S}$. Upon observing this state, the agent selects an action $a_n = a(t_n) \in \mathcal{A}$ by sampling from a policy $\pi(a_n | s_n)$, which encodes the conditional probability of taking action a_n in state s_n . The environment then transitions to a successor state s_{n+1} according to the transition law $P(s_{n+1} | s_n, a_n)$, and the agent receives a scalar reward signal r_{n+1} . The agent's objective is to find an optimal policy maximising the expected cumulative discounted reward. For a given episode, the **cumulative reward** (or **return**) is defined as

$$R_{\text{train}} = \sum_{n=0}^{N_{\text{train}}-1} \gamma^n r_{n+1},$$

where N_{train} is the number of steps in a training episode. The discount factor γ determines the relative importance assigned to immediate and future rewards. Values of γ close to one encourage long-term planning by placing nearly equal weight on distant rewards, whereas smaller values prioritise immediate gains.

Through sustained interaction with the environment, the agent iteratively refines its policy, eventually acquiring the ability to generalise to previously unseen states and to extrapolate in time beyond the training horizon.

The Deep Q-Network algorithm. In the present work, we employ the **Deep Q-Network (DQN)** algorithm [225], a value-based RL method that focuses on the approximation of the Q-function, also known as the action-value function:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{n+k} \mid s_n = s, a_n = a \right]. \quad (3.5)$$

The operator $\mathbb{E}_\pi[\cdot]$ denotes the expectation over all stochasticity arising from both the policy π and the environment dynamics, i.e., over all future state-action trajectories generated by following π from the initial pair (s, a) . The discounted infinite sum captures the cumulative future reward, with the discount factor γ ensuring convergence while progressively down-weighting the rewards associated with later steps.

Within **DQN**, the Q-function is represented by a deep neural network parametrized by θ , defining a nonlinear mapping

$$Q_\theta : \mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{A}|},$$

which takes the current state as input and outputs the estimated cumulative discounted reward associated with each possible action in the action space $a_1, \dots, a_{|\mathcal{A}|}$. Rather than learning an explicit policy, the agent derives its behaviour implicitly from the Q-function via a greedy selection rule,

$$a_n = \pi(s_n) = \arg \max_{a \in \mathcal{A}} Q_\theta(s_n, a),$$

typically augmented with an exploration strategy during the training phase.

DQN relies on two distinct networks: an *online* Q-Network with parameters θ (Q_θ), updated continuously during training, and a *target* Q-Network with parameters θ^- (Q_{θ^-}),

whose weights are periodically synchronized with those of the online network. The online network parameters are optimized by minimising the loss

$$\mathcal{L}(\theta) = \mathbb{E} \left[\left(r_{n+1} + \gamma \max_{a'} Q_{\theta^-}(s_{n+1}, a') - Q_{\theta}(s_n, a_n) \right)^2 \right].$$

The quantity $r_{n+1} + \gamma \max_{a'} Q_{\theta^-}(s_{n+1}, a')$ is known as the *temporal-difference target*. It combines the immediate reward received after taking action a_n with an estimate of the maximum discounted future return obtainable from the successor state. This estimate is given by the target network Q_{θ^-} .

The use of two separate networks is essential for stabilising the optimization process. While the online network is updated at every training iteration, the target network remains fixed for several updates, providing a slowly varying reference. This separation reduces the harmful feedback loop that would arise if the network were simultaneously used to predict and to define its own learning target.

A detailed treatment of RL theory and methodology can be found in [325].

3.2.3 The RL-EF approach

We apply the RL techniques described in Section 3.2.2 to RL simulations, introducing the RL-EF approach. Specifically, RL-EF is an enhanced EF simulation in which the filter radius δ_n is time-dependent and updated at each time step by a pre-trained RL agent.

In the proposed framework, the action coincides with the filter radius applied at time t_n , i.e., $a_n = \delta_n$, while the observation (state) s_{n+1} corresponds to the vector-valued filtered velocity field $\mathbf{u}_{n+1}(\delta_n)$ obtained using the filter radius selected by the RL agent. The action space $\mathcal{A}$ consists of 50 discrete values, comprising four logarithmically-spaced values in $[1 \times 10^{-10}, 1 \times 10^{-7}]$ and 46 logarithmically-spaced values in $[1 \times 10^{-6}, 1 \times 10^{-3}]$, as schematically illustrated in Figure 3.5. This design reflects the distinct influence that different filter radii have on the velocity field: small values with orders of magnitude ranging from $\mathcal{O}(10^{-10})$ to $\mathcal{O}(10^{-7})$ yield nearly indistinguishable EF solutions, i.e., $\mathbf{u}_n(10^{-10}) \approx \mathbf{u}_n(10^{-7})$, whereas larger values produce a substantially different effect on the solution, i.e. $\mathbf{u}_n(10^{-6}) \not\approx \mathbf{u}_n(10^{-3})$. Consequently, a denser sampling is adopted over the range $[1 \times 10^{-6}, 1 \times 10^{-3}]$.

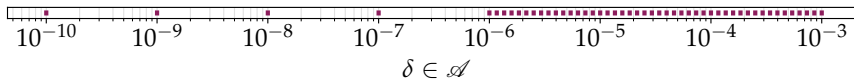


Figure 3.5: Action space used for DQN in RL-EF.

The observation space depends instead on the number of degrees of freedom of the discretization adopted in the EF simulation, namely $\mathcal{S} \equiv \mathbb{R}^{N_u}$, where N_u is the number of velocity nodes of the marginally-resolved grid. In our setting, one RL episode corresponds to a sequence of physical time steps, i.e., the time steps of an EF simulation over a prescribed training time window. A sketch of the RL-EF mechanism is represented in

Figure 3.6. As a result of the training procedure, the RL agent identifies an optimized trajectory $\{(\mathbf{u}_1, \delta_1), \dots, (\mathbf{u}_N, \delta_N)\}$ over the time interval $(0, T]$ (corresponding to N_T time steps), by learning from a reduced training window $(0, T_{\text{train}}]$ (corresponding to N_{train} time steps), with $T_{\text{train}} \ll T$.

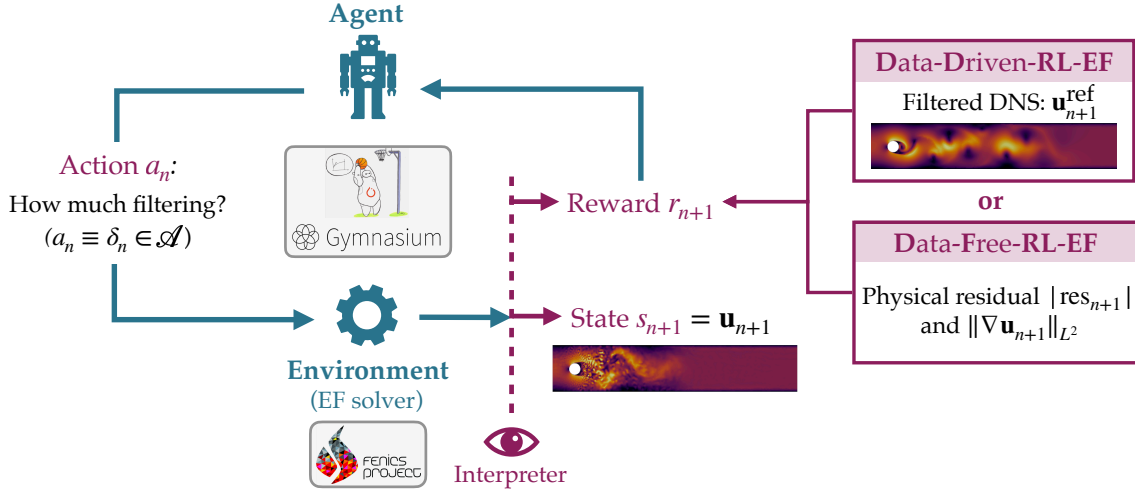


Figure 3.6: Illustrative sketch of the RL-EF procedure.

Remark: on the discrete action space.

Other RL strategies might be employed, such as the Proximal Policy Optimization (PPO) algorithm [301], which allows for a continuous action space. Preliminary investigations using the PPO approach led to poor results in terms of solution accuracy and training stability. However, we do not exclude that a more advanced empirical investigation and fine-tuning of the net architecture might yield more accurate results. Moreover, PPO poses additional challenges compared to DQN, as it features a fundamentally different input–output structure. Indeed, while DQN learns an action–value function from which the optimal action is derived implicitly, PPO directly learns the policy $\pi(a_n | s_n)$, i.e., a probability distribution over the action space. This makes its training more complex, as the algorithm must optimise the policy parameters directly, navigating a higher-dimensional learning problem than that faced by a value-based method such as DQN. For these reasons, in this work we decided to adopt a discrete action space using DQN.

RL rewards. The natural question is:

What is the most appropriate reward expression to enable the agent to learn a suitable filter selection strategy?

We investigated two types of reward: either reference **data-driven**, based on the accuracy

of the **RL-EF** solution with respect to pre-collected reference **DNS** data, or reference **data-free**, based on physics-based quantities, like the equation residual and the velocity gradients of the **RL-EF** solution. Consequently, the general **RL-EF** methodology specializes in **Data-Driven Reinforcement Learning-based Evolve-Filter (DD-RL-EF)** and **Data-Free Reinforcement Learning-based Evolve-Filter (DF-RL-EF)**.

In the data-driven approach, we use as reference the filtered **DNS** velocity solution, indicated as $\mathbf{u}_n^{\text{ref}}$ at time step t_n , just as in Chapter 2. In terms of reward, we followed the lead of previous works [183, 182, 181], which consider the following general form for the reward at each time step t_{n+1} :

$$r_{n+1}^{\text{DD}} = 2 \exp\left(-\frac{1}{\alpha} e_{n+1}\right) - 1, \quad (3.6)$$

where α is a parameter empirically tuned, and e_{n+1} is a metric (specified later) associated with the error of the **RL-EF** solution $\mathbf{u}_{n+1}(\delta_n)$ with respect to the **DNS** reference $\mathbf{u}_{n+1}^{\text{ref}}$. The expression (3.6) is widely used in **RL** to map the error metric e_{n+1} into the interval $[-1, 1]$, thus improving stability and facilitating convergence during training.

In previous works that applied **RL** to **CFD**, the error metric taken into account is typically the relative energy spectrum difference at the first (and more energetic) wave numbers, namely:

$$e_{n+1}^{\text{spectrum}} = \frac{1}{K} \sum_{\kappa=1}^K \left(\frac{\mathcal{E}_{n+1}(\kappa) - \mathcal{E}_{n+1}^{\text{ref}}(\kappa)}{\mathcal{E}_{n+1}^{\text{ref}}(\kappa)} \right)^2, \quad (3.7)$$

where $\mathcal{E}(\kappa)$ and $\mathcal{E}^{\text{ref}}(\kappa)$ are the **RL-EF** and the filtered **DNS** energies, respectively, at the wavenumber κ at time step t_{n+1} . The parameter K denotes the maximum wavenumber retained in the error computation, whose value is typically determined through empirical tuning and varies depending on the test case under consideration. Preliminary experiments showed that the spectrum error (3.7) is not a good metric in our setting, as it led to poor results. We then investigated as a metric the square of the relative error between the **RL-EF** velocity and the reference solution, namely:

$$e_{n+1}^{\text{DD}} = \left(\frac{\|\mathbf{u}_{n+1} - \mathbf{u}_{n+1}^{\text{ref}}\|_2}{\|\mathbf{u}_{n+1}^{\text{ref}}\|_2} \right)^2. \quad (3.8)$$

In all our **DD-RL-EF** tests, the reward is r_{n+1}^{DD} in Equation (3.6), where $\alpha = 1$ and $e_{n+1} = e_{n+1}^{\text{DD}}$.

The reader may have now two questions:

- (i) *What is the number of time steps that should be considered in one RL episode?*
- (ii) *Should each episode start from the beginning of the simulation, e.g. $t = 0$?*

We answer these questions by introducing and comparing two variants for the data-driven setting:

- **DD-RL-EF**, where each episode starts from $t = 0$ and has duration T_{train} . In our experiments, $T_{\text{train}} = T/4$. The cumulative reward for each episode can then be computed as:

$$R^{\text{DD}} = \sum_{n=1}^{N_{\text{train}}} r_n^{\text{DD}}. \quad (3.9)$$

- **Data-Driven Reinforcement Learning-based Evolve-Filter with random initialization (DD-RL-EF-rand)**, where each episode is initialized from a randomly drawn time instant within the training window $n_{\text{rand}} \in \{1, 2, \dots, N_{\text{train}} - 1, N_{\text{train}}\}$. This setting has a reduced episode length compared to the standard **DD-RL-EF** case, which is set to $T_{\text{train}}^{\text{rand}} = T/10$, corresponding to $N_{\text{train}}^{\text{rand}}$ time steps. This choice encourages the agent to experience a broader variety of states compared to episodes systematically initialized at $t = 0$. The cumulative reward reads:

$$R_{\text{rand}}^{\text{DD}} = \sum_{n=n_{\text{rand}}}^{n_{\text{rand}}+N_{\text{train}}^{\text{rand}}} r_n^{\text{DD}}. \quad (3.10)$$

When considering the reference *data-free* setting, the reward reads as follows:

$$r_{n+1}^{\text{DF}} = \frac{1}{2} \left(2 \exp(-\alpha_{\text{res}} |\text{res}_{n+1}|) - 1 \right) + \frac{1}{2} \left(2 \exp \left(- \left| \frac{1}{\alpha_{\text{grad}} (\|\nabla \mathbf{u}_{n+1}\|_2 - \|\nabla \mathbf{u}_n\|_2)} \right| \right) - 1 \right), \quad (3.11)$$

which includes two different contributions:

- $(2 \exp(-\alpha_{\text{res}} |\text{res}_{n+1}|) - 1)$ is directly related to the residual res_{n+1} found using the Newton method when solving the **FEM** system. This formulation yields decreasing rewards as the physical residual grows, i.e., when the **RL-EF** prediction deviates from a solution of the discrete **NSE**. This component penalizes excessively small filter radii δ_n , which fail to provide sufficient stabilization to the system. As evidenced by the **noEF** simulation, such a regime leads a blow-up of the residual.
- $\left(2 \exp \left(- \left| \frac{1}{\alpha_{\text{grad}} (\|\nabla \mathbf{u}_{n+1}\|_2 - \|\nabla \mathbf{u}_n\|_2)} \right| \right) - 1 \right)$ is instead related to the temporal variation in the velocity space gradients. Small variations in the velocity gradients are indicative of overdiffusive behaviour, as observed in **EF** simulations with large filter radii such as $\delta = \eta$. In contrast to the residual term, this component penalizes excessively large values of δ_n . The agent is rewarded more generously when the velocity gradients exhibit greater temporal variation, reflecting a more accurate capture of the flow features and of their expected evolution in time.

The parameters α_{res} and α_{grad} are tuned after preliminary empirical investigations. In our **DF-RL-EF** tests, we consider the reward r_{n+1}^{DF} , with $\alpha_{\text{res}} = 1 \times 10^5$ and $\alpha_{\text{grad}} = 1 \times 10^4$. The cumulative reward for each episode is:

$$R^{\text{DF}} = \sum_{n=1}^{N_{\text{train}}} r_n^{\text{DF}}. \quad (3.12)$$

The reward formulation can also be extended to incorporate weak structure-preserving constraints, such as energy or enstrophy conservation, which rely solely on the **RL-EF** solution itself and require no access to reference data. Along these lines, we introduce the **Structure-Preserving Data-Free Reinforcement Learning-based Evolve-Filter (SP-DF-RL-EF)**, in which the reward is augmented with penalization terms accounting for the energy and enstrophy growth:

$$R_{\text{SP}}^{\text{DF}} = \sum_{n=1}^{N_{\text{train}}} \left[r_n^{\text{DF}} + \mathbf{1}_{\{\mathcal{E}_n > \mathcal{E}_{n-1}\}} \left(\exp \left(-\frac{\mathcal{E}_n - \mathcal{E}_{n-1}}{\alpha_{\mathcal{E}} \mathcal{E}_{n-1}} \right) - 1 \right) + \mathbf{1}_{\{\mathcal{Z}_n > \mathcal{Z}_{n-1}\}} \left(\exp \left(-\frac{\mathcal{Z}_n - \mathcal{Z}_{n-1}}{\alpha_{\mathcal{Z}} \mathcal{Z}_{n-1}} \right) - 1 \right) \right].$$

In the above expression, $\mathbf{1}_{\{\mathcal{E}_n > \mathcal{E}_{n-1}\}}$ and $\mathbf{1}_{\{\mathcal{Z}_n > \mathcal{Z}_{n-1}\}}$ are indicator functions that activate exclusively when energy or enstrophy growth is detected, respectively. In such cases, an additional penalization proportional to the magnitude of the energy and/or enstrophy growth is incorporated into the reward. This approach is particularly tailored for cases like the decaying **HIT** case to (weakly) impose a time-decaying energy and enstrophy. The parameters governing the penalization terms were tuned to $\alpha_{\mathcal{E}} = \alpha_{\mathcal{Z}} = 0.1$.

Table 3.2 provides a summary of the acronyms adopted for the **RL-EF** methodologies introduced in this work.

Table 3.2: Acronyms and main features of the proposed RL-EF approaches.

Acronym	Episode start	Episode length	Cum. reward	Data needed
DD-RL-EF	$t = 0$	$T_{\text{train}} = T/4$	R^{DD}	$\mathbf{u}_n^{\text{ref}}, t_n \in [0, T_{\text{train}}]$
DD-RL-EF-rand	random $t \in [0, T_{\text{train}}]$	$T_{\text{train}}^{\text{rand}} = T/10$	$R_{\text{rand}}^{\text{DD}}$	$\mathbf{u}_n^{\text{ref}}, t_n \in [0, T_{\text{train}} + T_{\text{train}}^{\text{rand}}]$
DF-RL-EF	$t = 0$	$T_{\text{train}} = T/4$	R^{DF}	No reference needed
SP-DF-RL-EF	$t = 0$	$T_{\text{train}} = T/4$	$R_{\text{SP}}^{\text{DF}}$	No reference needed

Remark: on the reward expression.

It is worth noting that, although **DQN** optimises the discounted return (see Equation (3.5)), the cumulative rewards reported in (3.9), (3.10), and (3.12) correspond to the undiscounted sum of the rewards collected during each training episode. We will analyze these metrics, even though are not used during training.

Other questions we address are:

Why not including the pressure discrepancy in the data-driven reward?

The inclusion of the pressure contribution in the **Opt-EFR**_{global}^{u,∇u,p} approaches presented in Chapter 2 was not beneficial to improve the approximation. Hence, we do not investigate the inclusion of the pressure approximation error for the **RL-EF** method.

Why not DF-RL-EF-rand?

In order to maintain a fully reference-free approach, we do not consider the strategy of initializing episodes at a random time step $n = n_{\text{rand}}$, since this would necessitate access to the filtered **DNS** state $\mathbf{u}_{n_{\text{rand}}}^{\text{ref}}$ as the starting point for time integration,

thereby compromising the data-free nature of the framework.

As illustrated in Figure 3.6, the development of **RL-EF** relies on the integration of FEn-iCSx [21, 10, 304, 303] with two main libraries: **Gymnasium** [332, 185], a standard Python-based library for defining reinforcement learning environments, and **Stable Baselines 3** [266], a collection of reliable and well-tested reinforcement learning algorithms built on top of Gymnasium.

3.3 Numerical results of RL-EF

This section is dedicated to showing the numerical results obtained applying the **RL-EF** methodologies on the two test cases presented in Section 3.1. The results include both an investigation in the training stage, with a loss and reward analysis, and the results in the predictive regime $[T_{\text{train}}, T]$.

3.3.1 Flow past a cylinder test case

We start the investigation on the flow past a cylinder test case by examining the **RL** training. The training statistics reported in Figure 3.7 show the cumulative reward over episodes and the **DQN** loss function over the training time steps. As a preliminary remark, it is worth noting that the maximum (*undiscounted*) achievable cumulative reward equals the number of time steps per episode N_{train} . In **DD-RL-EF** and **DF-RL-EF**, $N_{\text{train}} = N_T/4 = 2500$, corresponding to $T_{\text{train}} = 1$ s, while in **DD-RL-EF-rand**, as detailed in Section 3.2.3, $N_{\text{train}}^{\text{rand}} = N_T/10 = 1000$, corresponding to $T_{\text{train}}^{\text{rand}} = 0.4$ s.

A first inspection of the results reveals that all methods converge to values close to their respective maximum cumulative reward, demonstrating that near-optimal policies are successfully learned in all cases. Among them, **DF-RL-EF** reaches the performance plateau earlier and displays smaller fluctuations than **DD-RL-EF**, as highlighted in the zoomed box of Figure 3.7. This behaviour can be attributed to the reward formulation of **DF-RL-EF**, which incorporates both the residual contribution and the temporal variation of the velocity gradients. This structure, combined with the specific choice of the parameters α_{res} and α_{grad} , is deliberately designed to promote stable learning dynamics and large rewards.

A similar trend is observed in **DD-RL-EF-rand**, whose cumulative reward exhibits limited oscillations and rapidly attains its maximum value (1000). In this case, the random initialization of each episode promotes broader state exploration, thereby accelerating the convergence of the agent.

These observations are further confirmed by the loss evolution: the losses associated with **DD-RL-EF-rand** and **DF-RL-EF** decay more rapidly than that of **DD-RL-EF**, reaching $\approx 1 \times 10^{-4}$ before time step 50 000 (which corresponds to 50 episodes for **DD-RL-EF-rand**

and to 20 episodes for **DF-RL-EF**). On the other hand, the **DD-RL-EF** loss stabilizes at the higher plateau of $\approx 1 \times 10^{-3}$.

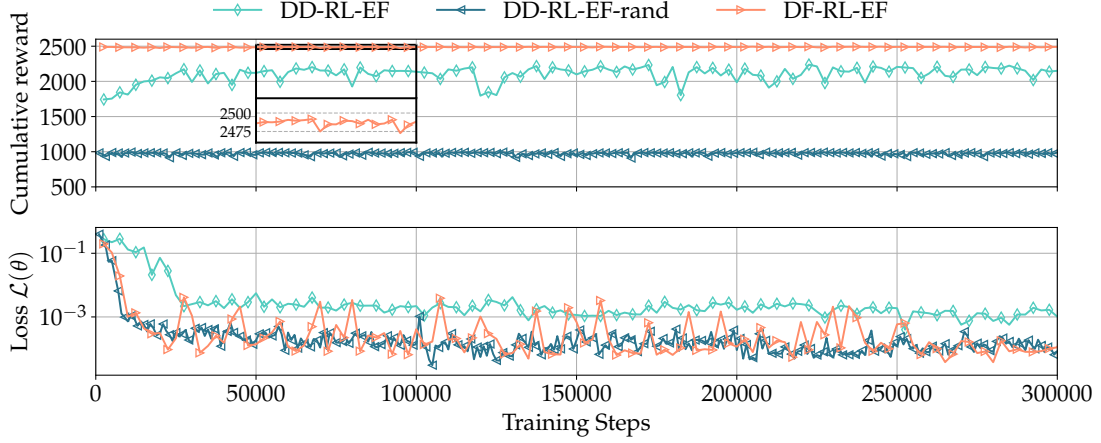


Figure 3.7: *Flow past a cylinder test case*. Cumulative reward and loss value during training steps, for the RL-EF methods.

Once the training is complete, we assess the performance of **RL-EF** in a predictive regime. Figure 3.8 displays: (i) the actions $\delta_n, n = 1, \dots, N_T$ selected by the pre-trained **RL** agent throughout the simulation time (left); (ii) the action frequency (right), addressing the question: *what percentage of the time is a given action chosen by the agent?*

A first observation is that different agents select different actions. However, across all algorithms, the action values exhibit an oscillatory and irregular temporal behaviour. Consistent with the preliminary investigation carried out in [147] (Chapter 2), this suggests that no single fixed optimal δ exists for the entire time window, and that, depending on the current simulation state, the optimal choice at certain time instances is *not to filter*, i.e., selecting δ_n with order of magnitude ranging from $O(10^{-10})$ to $O(10^{-7})$. Indeed, the **RL** agent predominantly selects values smaller than the Kolmogorov length scale η , indicated in Figure 3.8 by dashed grey lines. Furthermore, **DF-RL-EF** tends to select larger values of δ_n overall compared to **DD-RL-EF** and **DD-RL-EF-rand**. This behaviour suggests that the reference-free reward formulation implicitly places greater emphasis on maintaining simulation stability, leading the agent to apply filtering more consistently and to modulate the filter strength dynamically rather than deactivating it. The inclusion of physics-based terms in the reward systematically promotes stabilization, effectively preventing the onset and growth of numerical instabilities.

The above observations are further supported by Figures 3.9 and 3.10, which present a comparison among the **noEF**, **RL-EF**, standard **EF**, and the reference filtered **DNS** solutions. Specifically, Figure 3.9 compares the methods in terms of the velocity norms $\|\mathbf{u}(t)\|_2$ and $\|\nabla \mathbf{u}(t)\|_2$, while Figure 3.10 examines the drag and lift forces acting on the cylinder.

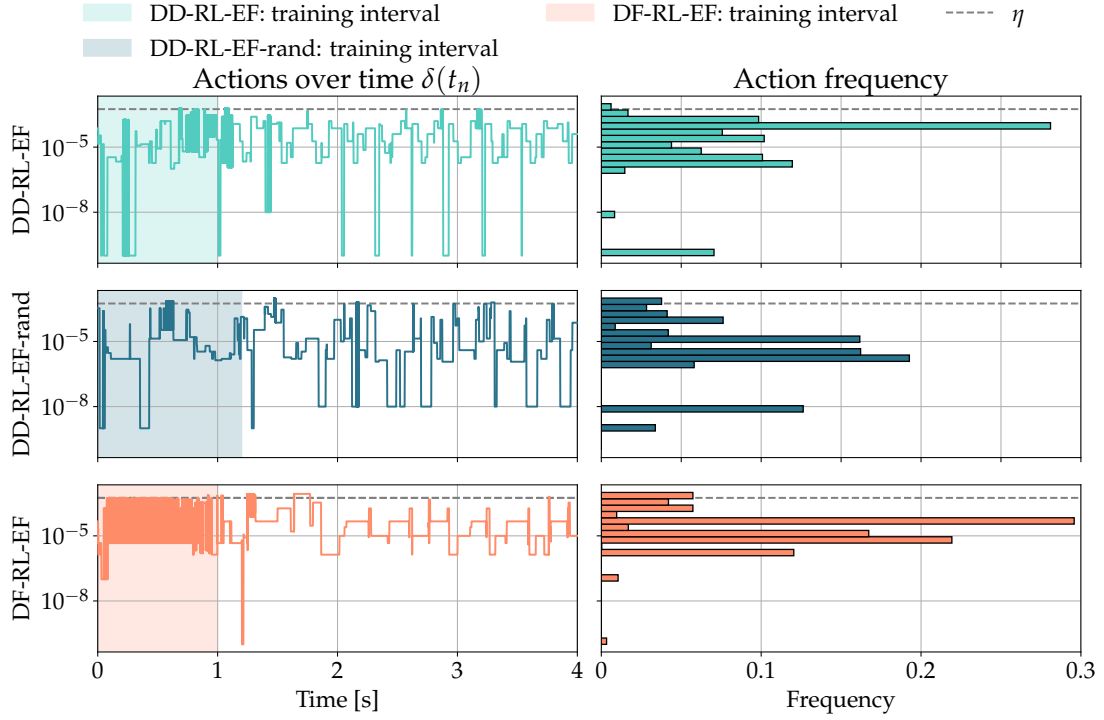


Figure 3.8: *Flow past a cylinder test case*. Action values over testing time, and corresponding frequency, for the RL-EF methods.

The dimensionless lift and drag coefficients (in continuous space) are defined as:

$$\begin{aligned} C_D(t) &= \frac{2}{U^2 L} \int_{\partial\Omega_C} ((2\nu\nabla\mathbf{u}(t) - p(t)\mathbf{I}) \cdot \mathbf{n}_C) \cdot \mathbf{t}_C \, ds, \\ C_L(t) &= \frac{2}{U^2 L} \int_{\partial\Omega_C} ((2\nu\nabla\mathbf{u}(t) - p(t)\mathbf{I}) \cdot \mathbf{n}_C) \cdot \mathbf{n}_C \, ds, \end{aligned} \quad (3.13)$$

where U and L are the characteristic velocity and length scales of the test case, $\mathbf{n}_C$ and $\mathbf{t}_C$ are the unit vectors in the normal and tangential directions to the cylinder, respectively. The cylinder boundary is denoted with $\partial\Omega_C$. In the discrete setting, the boundary integral is approximated by summing the contributions over each boundary edge via Gaussian quadrature, and the stress tensor $2\nu\nabla\mathbf{u} - p\mathbf{I}$ is evaluated using the FEM solution $(\mathbf{u}, p)$ on each boundary element.

Figures 3.9 and 3.10 demonstrate that all RL-EF methods achieve greater accuracy and closer agreement with the filtered DNS reference than both noEF and standard EF.

The overdiffusive character of the standard EF approach is apparent across all considered metrics, manifesting as a nearly flat temporal behaviour without any oscillations. While this ensures numerical stability, it comes at the cost of suppressing physically relevant dynamical features of the flow. The noEF simulation, on the other hand, loses stability and blows up after $t = 1$; its corresponding curve is therefore not reported beyond this time instant in the plots.

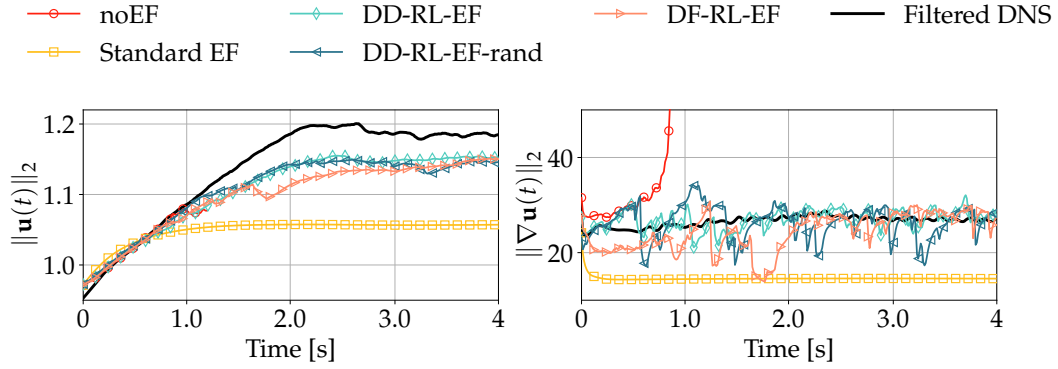


Figure 3.9: *Flow past a cylinder test case*. Time evolution of the velocity and velocity gradients' norms, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods.

All **RL-EF** strategies successfully recover the correct energy evolution and closely track the reference solution. The velocity gradients, however, display temporal oscillations around the reference values. This behaviour is not unexpected and is intrinsic to this class of simulations. Notably, analogous oscillations were reported in Section 2.3 (Figure 2.10), even though in that work each action δ_n was determined as the optimal value resulting from a separate optimization carried out independently at every time step t_n . The presence of oscillations can therefore not be ascribed to suboptimality of the learned policy. Indeed, the setting adopted in the previous Chapter 2 is more favourable, as the filter parameter was computed with full access to the reference data and through a time-local optimization at each step. In the present work, by contrast, the **RL** agent learns a single policy from limited training data and is subsequently required to generalize in a time-extrapolation regime. The observation of similar oscillatory patterns under these more demanding conditions further supports the **robustness** and **effectiveness** of the proposed **RL-EF** framework.

Regarding the aerodynamic forces, the data-driven methods closely reproduce the reference lift and drag coefficients, both in amplitude and frequency. The reference data-free strategy represents an improvement over standard **EF**, although it slightly underestimates the force amplitudes and velocity norms over certain time steps in the interval $[1.5, 2.5]$.

Figure 3.11 shows the energy spectrum $\mathcal{E}(\kappa)$ at selected wavenumbers for the proposed methods, alongside the theoretical turbulence scaling expected in two-dimensional test cases.

At $t = 1$, when vortex shedding is not yet fully developed, all methods display a comparable large-scale energy distribution. Nevertheless, the standard **EF** approach already exhibits a pronounced attenuation of high-wavenumber modes, leading to a premature energy decay at small scales and confirming its overdiffusive character.

At later times ($t = 2$ and $t = 4$), the differences become more pronounced. The **noEF**

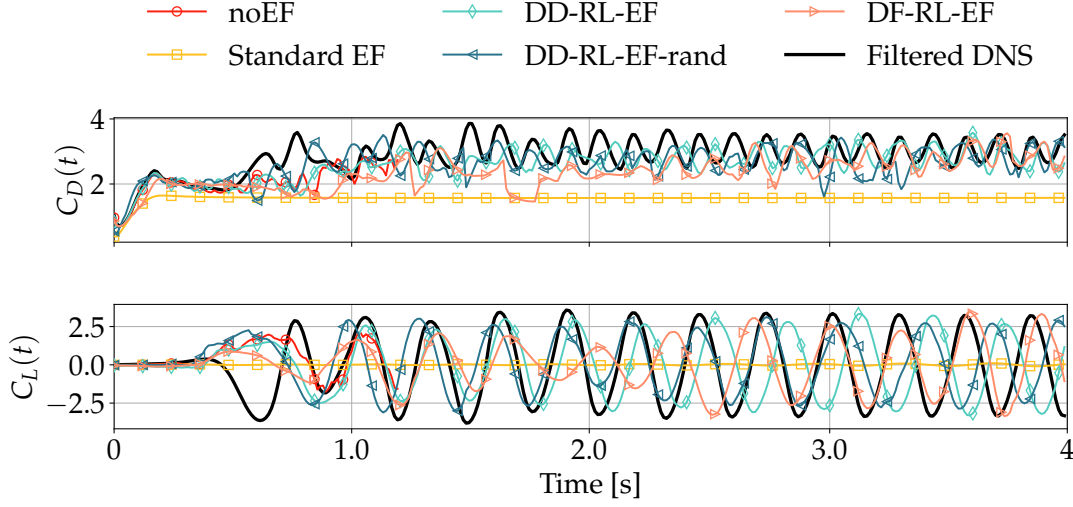


Figure 3.10: *Flow past a cylinder test case*. Time evolution of drag and lift coefficients, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods. Drag and lift forces are computed on the cylinder boundary.

simulation accumulates excessive energy at intermediate and high wavenumbers as a consequence of the blow-up, while standard EF strongly suppresses small-scale activity, producing spectra that are overly steep and fall below the reference solution in the inertial range.

All RL-EF strategies achieve a substantially improved balance between these two extremes. They correctly preserve the large-scale energy content while effectively controlling the high-wavenumber range without introducing excessive diffusion. The resulting spectra remain in close agreement with the reference slope, also in the inertial range where the κ^{-3} scaling is expected.

At $t = 1$ and $t = 2$, the DF-RL-EF spectrum displays lower energy at large wavenumbers, i.e., at small scales, and remains closer to the filtered DNS than the DD-RL-EF strategies. This is directly consistent with the larger δ_n values observed in Figure 3.8, reflecting the tendency of the data-free agent to apply stronger filtering during these stages. This confirms that the physics-based reward effectively promotes stabilization by penalizing the uncontrolled growth of small-scale energy. In particular, it enables the agent to regulate the dissipative range in a physically meaningful manner, preventing energy accumulation at high wavenumbers while preserving the correct large-scale dynamics. Overall, these results demonstrate that a purely physics-informed reward is sufficient to guide the agent toward a spectrally consistent filtering strategy, capable of balancing stability and accuracy across scales.

Furthermore, Figure 3.12 presents the velocity magnitude fields at selected time instances. In comparison with the noEF and standard EF results shown in Figure 3.2, all RL-EF strategies qualitatively reproduce the vortex shedding dynamics with good accuracy, avoiding both the overdiffusive smoothing and the spurious noise observed in the

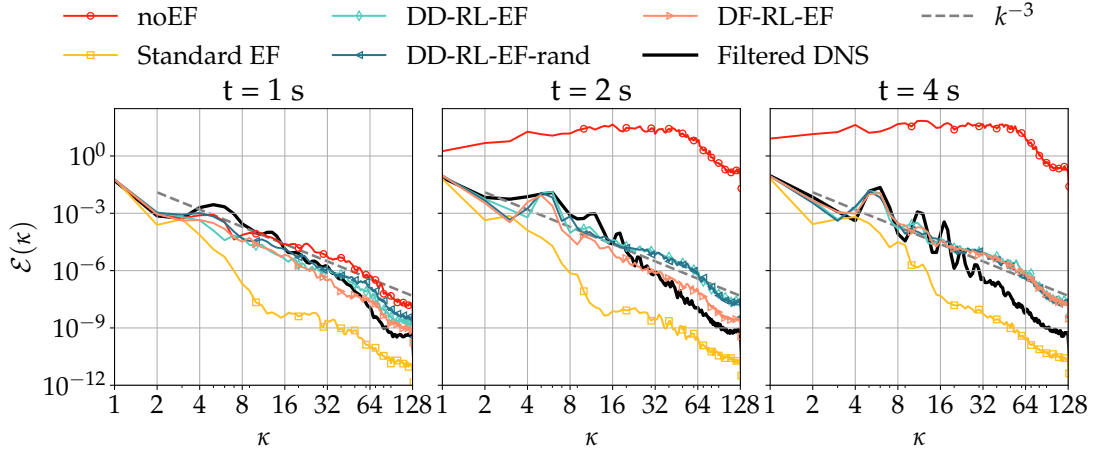


Figure 3.11: *Flow past a cylinder test case.* Energy spectrum at three time instances, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods. The spectrum is computed at the cylinder wake, after “artificially” making the domain periodic.

baseline methods.

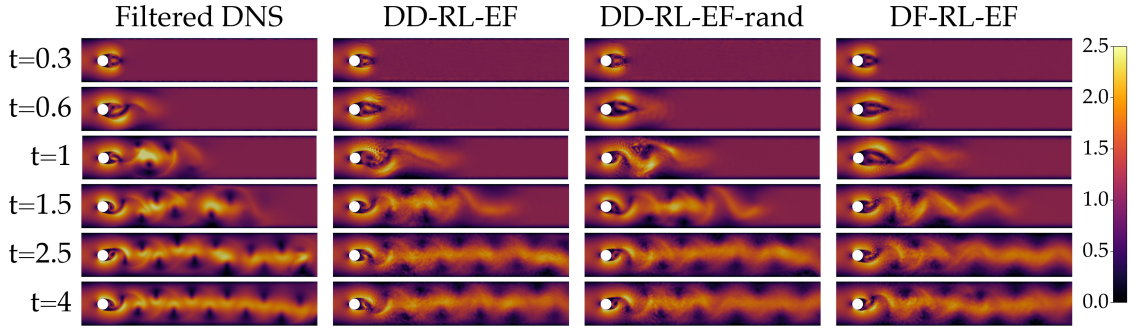


Figure 3.12: *Flow past a cylinder test case.* Time evolution of the velocity solution for the RL-EF simulations, compared with the filtered DNS.

Finally, we compare the methods in terms of global metrics, obtained by averaging in time the accuracy of the **RL-EF** kinetic energy and the energy spectrum across different scales. Global quantities such as energy and enstrophy are adopted in place of the classical relative L^2 error, which is not well-suited to assess the performance of the proposed methodology [147, 322] (see Figure 2.6).

Specifically, we consider:

$$\text{err}_{\mathcal{E}} = \frac{1}{N_T} \sum_{n=1}^{N_T} \left| \frac{\mathcal{E}_n - \mathcal{E}_n^{\text{ref}}}{\mathcal{E}_n^{\text{ref}}} \right|, \quad \text{err}_{\text{spectrum}} = \frac{1}{N_T} \sum_{n=1}^{N_T} \left(\frac{1}{K} \sum_{\kappa=1}^K \log_{10} \left(\frac{\mathcal{E}_n(\kappa)}{\mathcal{E}_n^{\text{ref}}(\kappa)} \right) \right), \quad (3.14)$$

where K is the maximum wavenumber retained in the error computation. In $\text{err}_{\text{spectrum}}$, a $\log_{10}$ -scaled error is employed, as it more effectively captures the relative magnitude of discrepancies spanning several orders of magnitude. This metric would highlight differences in the low-energy, large-wavenumber region of the spectrum, which are often critical for evaluating the fidelity of numerical schemes in convection-dominated flows.

Figure 3.13 reports the above-defined metrics for all **RL-EF** solutions, alongside the **noEF** and standard **EF** baselines. The spectral error $\text{err}_{\text{spectrum}}$ is evaluated at two K values: a larger one retaining all scales, from high-energy small wavenumbers to low-energy large wavenumbers, and a smaller one restricted to the energy-containing large scales only. This distinction allows us to assess the accuracy of the **RL-EF** strategies separately on the dominant large-scale structures (small K) and across the full spectral range (large K).

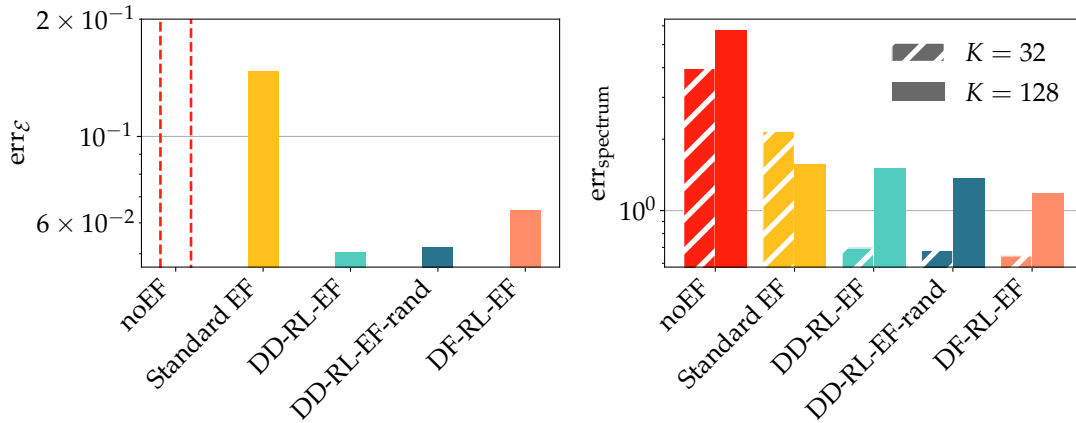


Figure 3.13: *Flow past a cylinder test case*. Error of the kinetic energy averaged in time (err_E), and log-space error of the energy spectrum averaged with respect to the wavenumbers and in time ($\text{err}_{\text{spectrum}}$). The energy spectrum error is computed over two different wavenumbers ranges, namely $K = 32$ and $K = 128$.

For the cylinder test case (Figure 3.13), concerning the energy error err_E , all **RL-EF** methods achieve considerably higher accuracy (approximately 5–6% error) than standard **EF** (approximately 15% error). The **noEF** curve is shown as a dashed red line due to the blow-up of its solution.

Regarding the spectral error, the **RL-EF** methods, and in particular the data-free approach **DF-RL-EF**, attain the lowest error for both values of K . The discrepancy between methods is more pronounced when only the large scales are retained ($K = 32$), suggesting that the choice of filtering strategy has a stronger influence on the accuracy of the energy distribution in the dominant flow structures. In particular, standard **EF** incurs a larger error in the large-scale energy content, as it fails to reproduce the physical vortex shedding. The **RL**-based strategies, by contrast, successfully recover the relevant flow dynamics and better preserve the energy distribution across scales. When the full spectrum is considered ($K = 128$), the differences among **RL** approaches remain limited, with **DF-RL-EF** achieving the lowest overall error. In this regime, standard **EF** performs comparably to the **RL-EF** approaches, as it does not suffer from spurious oscillations or noisy behaviour at small scales.

In conclusion, the following key observations deserve emphasis:

- In the **reference-guided** approaches, reference data are provided only over a limited

time window, namely $[0, 1]$ (or $[0, 1.4]$ for **DD-RL-EF-rand**), during which vortex shedding is not yet fully developed. Despite this restricted, pre-transient training regime, the **RL** agent successfully learns to modulate the filtering intensity from the velocity state alone. Remarkably, the learned policy generalizes to the time-extrapolation regime, maintaining both stability and accuracy of the **RL-EF** simulations even after the flow reaches a fully developed state.

- In the **reference-free** setting, the **RL** agent is trained without any access to reference data, relying exclusively on physics-based information. Nevertheless, it autonomously develops an effective filtering strategy that preserves both stability and accuracy. The resulting performance closely matches that of the data-driven approaches, demonstrating that physically informed rewards alone are sufficient to guide the learning process toward robust and high-quality solutions. This also drastically reduces the computational cost of training, since no reference data or **DNS** solution is required, as further discussed in Section 3.3.3.

Overall, these results underscore the robustness and generalization capability of the proposed **RL** framework, demonstrating that effective stabilization policies can be learned even from limited or entirely absent reference data.

3.3.2 Decaying homogeneous isotropic turbulence test case

Once we have demonstrated the accuracy of **RL-EF** for the flow past a cylinder case, we now address the following question:

*Can the **RL**-based filtering strategy adapt to a regime where no sustained energy production occurs, and where the primary challenge is to preserve the correct decay rate while avoiding excessive dissipation?*

As for the first test case, we start by analyzing the training stability. Figure 3.14 illustrates the cumulative reward and the **DQN** loss function over the course of training episodes for the different **RL-EF** strategies under consideration. Given that this test case is designed to preserve both energy and enstrophy, the analysis is extended to include the **SP-DF-RL-EF** strategy introduced in Section 3.2.3. The maximum achievable cumulative reward is $N_{\text{train}} = N_T/4 = 500$ time steps ($T_{\text{train}} = 0.5$) for all the **RL-EF** methods, with the exception of **DD-RL-EF-rand**, whose episodes are shorter, comprising $N_{\text{train}}^{\text{rand}} = N_T/10 = 200$ time steps ($T_{\text{train}}^{\text{rand}} = 0.2$).

All **RL-EF** strategies converge to a plateau in the proximity of their respective theoretical maximum reward, albeit with markedly different loss decay rates. Notably, and in contrast to the first test case, the overall loss decay is considerably slower, reaching values of the order of $\approx 1 \times 10^{-4}$ only for the **DD-RL-EF** and **DF-RL-EF** strategies. The **DD-RL-EF-rand** and **SP-DF-RL-EF** strategies, on the other hand, exhibit a more oscillatory loss behaviour throughout training.

In the case of **DD-RL-EF-rand**, this instability is consistent with the observations from the first test case and can be attributed to the random initialization of each training episode.

The stochastic nature of the episode reset introduces variability in the initial conditions, which in turn increases dispersion in the learning dynamics. In the structure-preserving setting, the oscillatory behaviour arises from a fundamentally different mechanism: the additional physical constraints embedded in the reward formulation. Although these constraints promote consistency with the underlying physics of the system, they simultaneously increase the complexity of the optimization landscape, resulting in less stable training dynamics and rendering the identification of the optimal filter parameter a more challenging task.

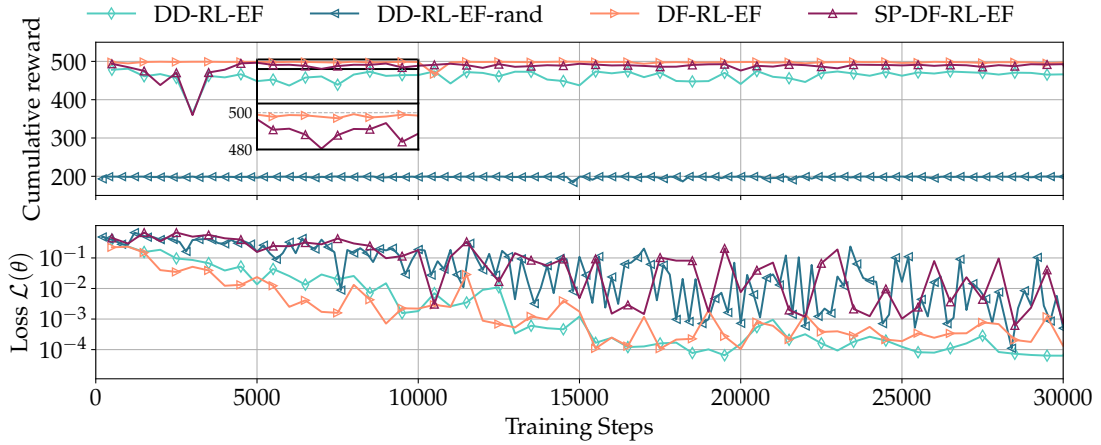


Figure 3.14: *Decaying HIT test case*. Cumulative reward and loss value during training steps, for the RL-EF methods.

We analyze then the performance of the RL-EF methods in predictive regimes. Figure 3.15 illustrates the filtering actions selected by the pre-trained RL agents throughout the simulation, along with the corresponding distributions of the most frequently chosen filter values. In this test case, the agents display a less oscillatory behaviour compared to the previous scenario, with selected action values being sustained over longer consecutive time intervals. The action frequency distributions further reveal that the most commonly selected filter radius δ_n at time t_n coincides with, or lies in close proximity to, the Kolmogorov length scale. This tendency is particularly pronounced in the DD-RL-EF-rand and SP-DF-RL-EF strategies, where the agent selects $\delta_n = \eta \approx 3.53 \times 10^{-4}$ at approximately 40% and 50% of the time steps, respectively. This behaviour suggests that, in a purely dissipative regime, the policy learned by the agent naturally gravitates toward a physically meaningful small-scale cutoff, effectively balancing numerical stabilization with the preservation of the correct energy decay dynamics.

The time evolution of energy and enstrophy, shown in Figure 3.16, provides further insight into the comparative performance of the different strategies. The noEF simulation becomes numerically unstable and diverges shortly after $t = 0.25$, confirming the necessity of a filtering mechanism to ensure long-term stability. The standard EF approach, while stable, yields a smooth and monotonic decay of both energy and enstrophy at the cost of excessive dissipation, resulting in an over-damped response relative to the filtered

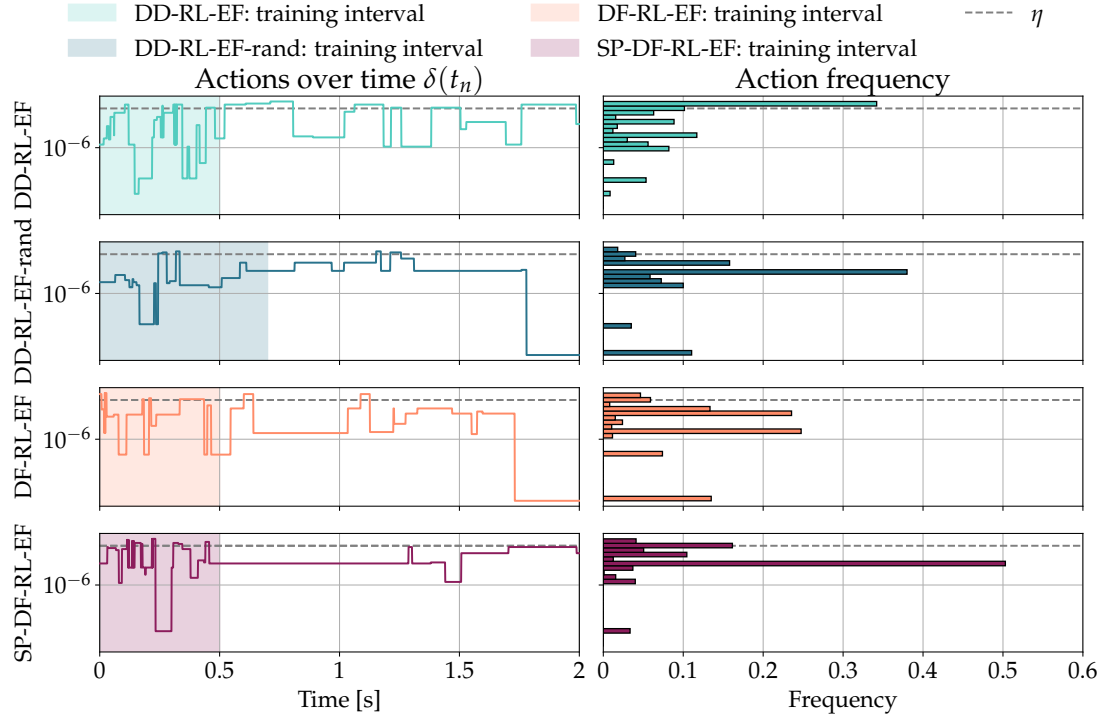


Figure 3.15: *Decaying HIT test case*. Action values over testing time, and corresponding frequency, for the RL-EF methods.

DNS reference.

All RL-EF strategies, with the exception of DD-RL-EF, successfully reproduce the correct energy decay behaviour without introducing the spurious diffusion observed in standard EF. Among these, DD-RL-EF-rand achieves the closest agreement with the filtered DNS in terms of energy evolution. DD-RL-EF, by contrast, tracks the reference solution accurately within the training time window up to $t = 0.5$, but becomes progressively overdiffusive at later times. This degradation in performance is consistent with the larger filter radius values δ_n selected beyond the training horizon, as reported in Figure 3.15, and points to a limited capacity for generalization in the extrapolation regime. Regarding the enstrophy evolution, DD-RL-EF, DD-RL-EF-rand, and DF-RL-EF all exhibit a transient increase, most notably at early times. This behaviour is indicative of a temporary amplification of small-scale activity prior to the onset of the dominant dissipative regime. In the SP-DF-RL-EF case, the weak enforcement of enstrophy conservation yields a more controlled and moderate response. Nonetheless, small residual increases in enstrophy are still observed, which can be attributed to the fact that the conservation constraint is imposed in a penalized, rather than hard, form within the reward function during training.

Figure 3.17 presents a comparison of the energy spectra at $t = 0.5, 1, 1.5$. The noEF simulation, having become unstable at early times, rapidly develops an unphysical spectrum characterized by an excessive accumulation of energy at high wavenumbers. The standard EF approach, conversely, systematically underestimates the energy content at

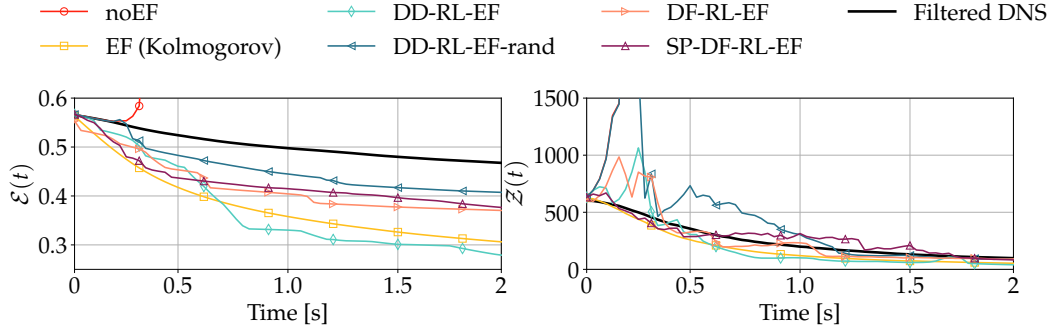


Figure 3.16: *Decaying HIT test case*. Time evolution of the velocity and velocity gradients' norms, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods.

small scales across all reported times, further corroborating its over-dissipative character.

At $t = 0.5$, all **RL-EF** strategies yield spectra that are mutually consistent and in good agreement with the filtered **DNS** reference. A minor exception is noted for **DD-RL-EF-rand**, which exhibits slightly elevated energy levels at intermediate and large wavenumbers, indicative of residual small-scale oscillations. This behaviour is consistent with the higher enstrophy values reported in Figure 3.16, and reflects a degree of enhanced small-scale activity in this strategy. At later times ($t \in [1, 1.5]$), **DD-RL-EF** progressively underestimates the energy content in the high-wavenumber range, in line with the stronger dissipation already identified in the energy and enstrophy evolution. By contrast, all the other methods maintain close agreement with the filtered **DNS** spectrum throughout, accurately reproducing both the large-scale energy content and the small-scale decay behaviour.

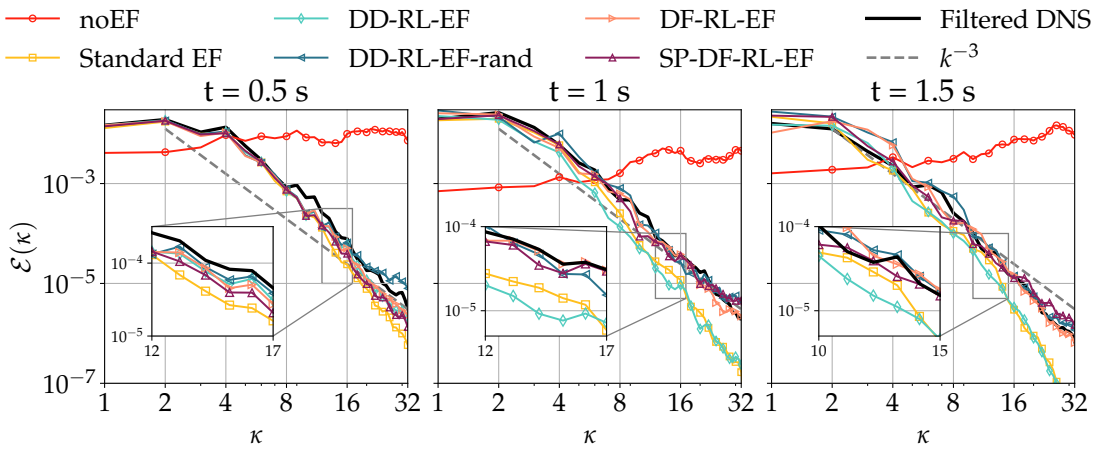


Figure 3.17: *Decaying HIT test case*. Energy spectrum at three time instances, for the filtered DNS, noEF, standard EF, and for the novel RL-EF methods. The spectrum is computed considering the entire squared domain.

A qualitative assessment is provided in Figure 3.18, which compares the **RL-EF** velocity fields against the filtered **DNS** at selected time instants. The visual inspection confirms the quantitative trends discussed above. For $t > 0.5$, the **DD-RL-EF** velocity field displays excessively smoothed vortical structures, consistent with the over-dissipative behaviour identified in both the energy decay and spectral analyses. **DD-RL-EF-rand**, on the other hand, exhibits a noisier small-scale structure at early times (e.g., $t = 0.25$), in agreement with its elevated enstrophy levels and increased high-wavenumber energy content.

The data-free strategies, by contrast, yield a more balanced solution overall. The vortical structures remain well-resolved, exhibiting neither spurious oscillations nor excessive smoothing, and reflect a more effective compromise between numerical stabilization and the faithful preservation of the physical decay dynamics.

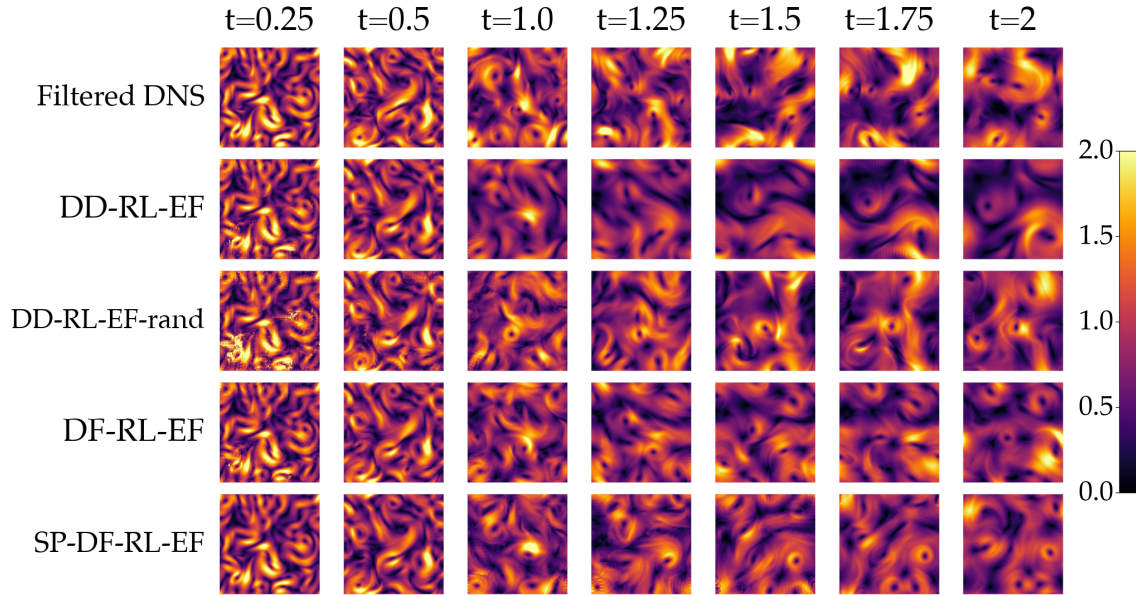


Figure 3.18: *Decaying HIT test case*. Time evolution of the velocity solution for the RL-EF simulations, compared with the filtered DNS.

As in subsection 3.3.1, the analysis is concluded by comparing the methods in terms of the global errors on the kinetic energy and on the energy spectrum, as defined in Equations (3.14). The results are summarized in Figure 3.19. With regard to the kinetic energy error err_E , the figure confirms that **DD-RL-EF-rand** achieves the highest accuracy among all strategies considered, followed by the reference-free methods, **SP-DF-RL-EF** and **DF-RL-EF**. The **DD-RL-EF** strategy, being characterized by excessive diffusion, yields an energy error comparable to that of standard **EF**, reflecting its inability to preserve the correct energy decay dynamics beyond the training horizon.

Concerning the error on the energy spectrum, a distinction is again drawn between the contribution of the large scales, evaluated at maximum wavenumber $K = 8$, and the full range of resolved scales, evaluated at $K = 32$. Unlike the flow past a cylinder (Figure 3.13), the discrepancy between the methods becomes considerably more pronounced

when the full range of scales is taken into account, whereas the differences are less significant when the analysis is restricted to the largest scales alone. This behaviour reflects the intrinsic difficulty of the present test case, where the primary challenge lies in accurately controlling the velocity gradients while avoiding excessive dissipation.

When the full spectral range is considered, a marked distinction emerges between the over-diffusive strategies, namely standard EF and DD-RL-EF, and the remaining approaches. The excessive dissipation inherent to the former group leads to an artificial depletion of energy at high wavenumbers, which in turn results in substantially larger spectral errors at $K = 32$.

When attention is restricted to the large-scale range ($K = 8$), all RL-EF strategies yield spectral errors of comparable magnitude to standard EF, with the notable exception of SP-DF-RL-EF. In this case, the weak enforcement of the physical constraint within the reward function enables the agent to recover the energy spectrum at the dominant scales, leading to improved agreement with the filtered DNS reference even in the absence of training data derived from the reference DNS solution.

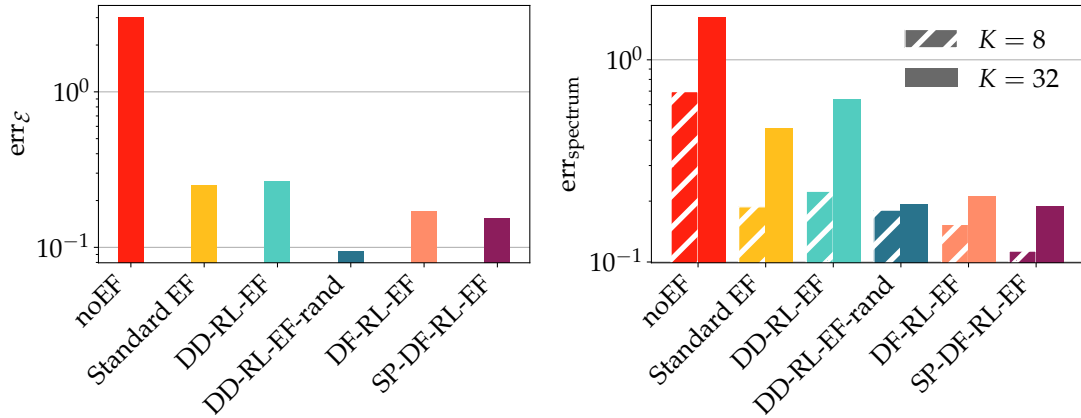


Figure 3.19: *Decaying HIT test case*. Error of the kinetic energy averaged in time (err_ε), and log-space error of the energy spectrum averaged with respect to the wavenumbers and in time ($\text{err}_{\text{spectrum}}$). The energy spectrum error is computed over two different wavenumbers ranges, namely $K = 8$ and $K = 32$.

Overall, this test case demonstrates the capacity of the proposed RL-based filtering framework to adapt to a purely dissipative regime, which is fundamentally different in character from the vortex-shedding scenario examined previously. The results collectively highlight the following key findings:

- Purely **reference-guided** training strategies are susceptible to limited generalization beyond the training window, with performance degrading as the flow dynamics evolve into the extrapolation regime. The introduction of randomization in the training time window can improve robustness with respect to this limitation, at the cost of mild small-scale noise in the solution;

- **Reference-free** reward formulations represent the most promising approach, yielding significantly enhanced stability while maintaining a high level of accuracy in both the spectral and energetic quantities of interest.

3.3.3 Computational details

This section summarizes the key computational aspects of the **RL-EF** experiments. Specifically, Table 3.3 reports the training times for each **RL-EF** variant and both test cases, while Table 3.4 collects the main **RL** hyperparameters to ensure reproducibility. All experiments were carried out on the Galileo G100 cluster at CINECA (Italy), using Intel Xeon Platinum nodes without GPU acceleration. As shown in Table 3.3, the CPU time required for the **DNS** in the decaying **HIT** case considerably exceeds that of the flow past a cylinder, due to the larger number of finite element nodes, as already reported in Table 3.1. Among the methods considered, **DD-RL-EF-rand** exhibits the lowest average CPU time per episode across both test cases. This is a direct consequence of its shorter episode length ($N_{\text{train}}^{\text{rand}} = N/10$, compared to $N_{\text{train}} = N/4$ adopted in all other approaches, as detailed in Table 3.2. However, **DD-RL-EF-rand** incurs a higher cost for collecting the filtered **DNS** data, since episodes are initialized at random points within $[0, T_{\text{train}}]$, necessitating **DNS** data over the extended interval $[0, T_{\text{train}} + T_{\text{train}}^{\text{rand}}]$.

It is also worth emphasizing that the reference-free methods, namely **DF-RL-EF** and **SP-DF-RL-EF**, require no **DNS** data, translating into savings of several hours of computational time. Notably, this reduction in cost does not come at the expense of accuracy; these approaches achieve comparable or even superior solutions relative to their reference-guided counterparts, as evidenced by the global spectrum errors reported in Figures 3.13 and 3.19. Finally, the episode counts listed in Table 3.3 correspond to the point at which the training reward reached a stable plateau, after which the model was saved for subsequent time extrapolation tests.

Table 3.3: Computational training times for RL-EF approaches.

Test case	Algorithm	Average CPU time per episode	Episodes	DNS CPU time
Flow past a cylinder	DD-RL-EF	15 min 38 s	150	63 min 41 s
	DD-RL-EF-rand	6 min 18 s	300	89 min 2 s
	DF-RL-EF	15 min 18 s	200	Not needed
Decaying HIT	DD-RL-EF	32 min 14 s	90	5 h 10 min 30 s
	DD-RL-EF-rand	13 min 11 s	210	7 h 13 min 45 s
	DF-RL-EF	32 min 9 s	90	Not needed
	SP-DF-RL-EF	32 min 14 s	60	Not needed

Table 3.4 reports the hyperparameters needed to train the **RL** agent for reproducibility purposes, and in particular the discount factor used for the Q training, the architecture of the **NN** (which is a standard **MLP**), and other training hyperparameters, like the learning rate, and the optimizer used. We specify that the neural network used in **DQN** in our case is $Q_{\theta} : \mathbb{R}^{N_u} \rightarrow \mathbb{R}^{|\mathcal{A}|}$, where $|\mathcal{A}| = 50$, corresponds to the number of admissible δ values.

Table 3.4: Hyperparameters used for RL training in RL-EF approaches.

Hyperparameter	Value
Discount factor γ	0.99
Learning rate	1×10^{-5}
Network architecture	2 hidden layers of 64 neurons each
Network activation	ReLU
Network optimizer	Adam
Max gradient norm	5
Batch size	128
Target update interval	$5N_{\text{train}}$

3.4 Conclusions and motivations to DD-EF(R) approaches

In this chapter, we presented **RL-EF**: a reinforcement learning framework for the dynamic selection of the filter radius within the **EF** stabilization of convection-dominated incompressible flows. The core idea is to replace heuristic, time-constant prescriptions of the filter width with a learned, state-dependent control policy that continuously adapts to the evolving flow dynamics. The **RL-EF** methodology recasts parameter tuning as a sequential decision-making problem, in which the agent learns a global control strategy over a prescribed training window and subsequently applies it in time extrapolation regimes. Both reference data-driven (**DD-RL-EF**) and fully reference-free (**DF-RL-EF**) reward formulations are investigated, including a structure-preserving variant that incorporates weak energy and enstrophy constraints where required by the specific test case. Numerical investigation on two benchmark problems with fundamentally different dynamics, i.e., the flow past a cylinder at $Re = 1000$ and decaying **HIT** at $Re = 40\,000$, yields the following key results:

- **Stability without over-diffusion.** The learned policies prevent the numerical blow-up observed in marginally-resolved no**EF** simulations, while avoiding the excessive dissipation that characterizes standard **EF** approaches. The **RL**-selected filter radius varies non-trivially in time, confirming that no single constant value remains optimal throughout the entire simulation.
- **Spectral consistency across scales.** The **RL-EF** strategies preserve large-scale flow structures while correctly regulating the dissipative range. The resulting energy spectra remain in close agreement with the reference filtered **DNS**, including in the inertial range, demonstrating that adaptive filtering can recover the correct balance between energy-containing and small-scale modes.
- **Generalization beyond the training window.** In the data-driven setting, the agent is trained exclusively over an early-time window in which the flow has not yet fully

developed. Despite this limited exposure, the learned policy successfully generalizes to later times, encompassing regimes characterized by vortex shedding or long-time turbulent decay. This underscores the ability of **RL** to extract transferable control laws rather than simply interpolating reference data.

- ▶ **Effectiveness of physics-informed rewards.** Most importantly, the reference-free formulation attains performance comparable to its data-driven counterpart. By relying solely on residual-based control and gradient-based indicators, without recourse to computationally expensive **DNS** reference data, the **DF-RL-EF** agent autonomously discovers filtering strategies that are simultaneously stable and spectrally accurate. This demonstrates that physically meaningful reward design alone is sufficient to guide the learning process toward robust stabilization policies, even in the complete absence of reference solutions.

Across both test cases, the learned filtering policies adapt to the qualitative character of the flow regime: oscillatory and highly dynamic in the cylinder wake, and more conservative and persistent in decaying turbulence. This behaviour confirms that the **RL** framework captures problem-specific stabilization requirements rather than imposing a uniform dissipative mechanism. The proposed approach offers a robust and flexible alternative to manual parameter calibration in **EF**-type methods, eliminating fixed tuning rules based on mesh size or Kolmogorov scaling and replacing them with adaptive, state-aware control strategies learned directly from the flow evolution. Several research directions naturally arise from this study. As the primary objective of the present work was to establish a preliminary assessment of the robustness of the proposed approach, these aspects have not been explored here and will be addressed in future investigations:

1. The extension to *three-dimensional* turbulent flows represents a fundamental step toward realistic high-Reynolds-number applications, where increased scale separation and richer nonlinear interactions pose substantially greater stabilization challenges.
2. A particularly promising avenue is the development of *spatially-varying filtering* strategies. In the current formulation, the filter radius is uniform over the computational domain at each time step. However, the optimal level of stabilization is generally highly localized, particularly in flows featuring shear layers, vortex shedding, or boundary layers. Enabling spatial variation of the filter radius could significantly improve accuracy, for instance through multi-agent reinforcement learning, where distinct agents govern local subdomains, or via graph-based neural architectures that operate directly on mesh connectivity to enable localized control.
3. The present framework remains grid-dependent, as the observation space coincides with the discrete velocity degrees of freedom. Constructing a *mesh-independent state*

representation, for example through graph-based encodings, would improve transferability across meshes and resolutions, paving the way toward discretization-agnostic stabilization policies.

4. Adopting a *continuous action space* would eliminate the need to discretize the filter radius and allow finer control of the stabilization mechanism.

One of the inherent limitations of the EF/EF_R approach lies in its dependence on only one (in EF) or two (in EF_R) tunable parameters. Chapter 4 investigates the possibility of overcoming this restriction by scaling the number of controllable parameters up to the full number of degrees of freedom of the simulation, thereby enabling a local prescription of the filtering intensity that can adapt to the heterogeneous stabilization requirements across the computational domain.

DD-EFR: A DATA-DRIVEN ALTERNATIVE TO DIFFERENTIAL FILTERS

While the work presented in Chapters 2 and 3 relies on the **DF** for the filter step, in this chapter we introduce an entirely different filtering concept, constructed directly from reference **DNS** data rather than prescribed through a differential operator. This direction is motivated by the existence of a wide range of **data-driven approaches to flow stabilization**. As introduced in Section 1.4, several research works integrate machine learning into **LES** to model the closure terms and to improve the simulation accuracy. It was found that the performance of such models depends critically on how the network parameters are optimized: when a differentiable solver is available, gradient-based techniques can be employed to train the parameters directly with respect to how well the coarsened **DNS** solution is reproduced. This end-to-end training strategy is often necessary, as optimizing networks to reproduce only the right-hand side — while computationally cheaper — frequently leads to inaccuracies and long-time instabilities [203, 102, 213, 5, 223]. The main limitations of these approaches are the substantial computational cost of generating training data and training the networks, the requirement for a differentiable solver, and the resulting incompatibility with many existing codebases. Furthermore, the learned models are considerably less interpretable than, for example, explicit **DFs**.

For these reasons, in this work we do not adopt machine learning strategies to build our data-driven filter, but we rely instead on a **linear regression approach**.

The following novelties are introduced in this work:

- ▶ The definition of a novel **data-driven linear filter**, which is trained on reference **DNS** data. The resulting evolve-filter strategy exploiting this filter is called **Data-Driven Evolve-Filter (DD-EF)**. Beyond its greater flexibility with respect to the **DF**, **DD-EF** offers several additional advantages. By exploiting the **Fast Fourier Transform (FFT)**, the filter can be applied in a computationally efficient manner, entirely bypassing the need to solve a linear system at each application. Moreover, computing the optimal filter coefficients reduces to a set of independent one-dimensional least squares problems (one for each coefficient), making the parameter identification process significantly cheaper than training a neural network.
- ▶ A clever way to tune the relaxation parameter χ , which leads to a **energy and/or enstrophy-conserving** version of the algorithm.

The methodology is implemented and combined with an energy-conserving finite volume solver using a staggered grid discretization [2, 4], as described in Section 4.1.

A detailed description of the novel methodology can be found in Section 4.2, which is subdivided into two parts: the first part (4.2.1) presents how the data-driven filter is computed, and the second part (4.2.2) presents how a DD-EFR approach can preserve energy and/or enstrophy by appropriately tuning the relaxation parameter.

The methodological developments are followed by a numerical investigation in Section 4.3, where the proposed approach is validated on the decaying HIT test case at $Re = 40\,000$ (the same test case as in Section 1.3.1). The learned filter is also used in the Kolmogorov flow case, which has the same setting as the decaying HIT case, but with an additional forcing term. In this way, the filter is assessed both in temporal extrapolation, as it is applied beyond the time window used for training, and in **extrapolation with respect to the test case**, as it is deployed on flow configurations not seen during the learning stage.

4.1 Preliminaries: the energy-conserving solver and the staggered grid framework

The project is implemented and tested using the library [2, 3], which adopts a finite volume discretization in a staggered grid framework. In this framework, the velocity field is approximated on the cell faces, while the pressure field is approximated at the cell centers, as shown in Figure 4.1a.

Differently from Chapters 2 and 3, where the filtered DNS solution is obtained via interpolation on the coarser grid, in this Chapter we adopt a different mapping from the refined to the coarse grid. In particular, we adopt a face-average filter operation, which is sketched in Figure 4.1b for a 2D structured grid¹. The face-averaging filter can be thought of as a top-hat filter (see Equation (1.8)) acting on the dimensions orthogonal to the velocity components only.

The discretized NSE at time step t_n can be written in the general matrix form as follows:

$$\begin{cases} \mathbf{M}_u \frac{\partial \mathbf{u}_n}{\partial t} = -\mathbf{C}(\mathbf{u}_n)\mathbf{u}_n + \nu \mathbf{D}_u \mathbf{u}_n - \mathbf{G}p_n + \mathbf{f}_n, \\ \mathbf{H}\mathbf{u}_n = \mathbf{0}. \end{cases} \quad (4.1)$$

In the above formulation, $\mathbf{u}_n \in \mathbb{R}^{N_u}$, $p_n \in \mathbb{R}^{N_p}$, for each n , $\mathbf{M} \in \mathbb{R}^{N_u \times N_u}$ is the mass matrix², $\mathbf{C} \in \mathbb{R}^{N_u \times N_u}$ is the convection operator, $\mathbf{D}_u \in \mathbb{R}^{N_u \times N_u}$ is the diffusion operator, $\mathbf{f} \in \mathbb{R}^{N_u}$ is the external forces vector, $\mathbf{G} \in \mathbb{R}^{N_u \times N_p}$ is the pressure gradient operator, $\mathbf{H} \in \mathbb{R}^{N_p \times N_u}$ is the divergence operator. N_u and N_p are the number of degrees of freedom for the velocity and pressure fields, respectively.

¹The procedure can be extended to unstructured grids in a straightforward way [4].

² $\mathbf{M}_u = \mathbf{I}$ in standard finite volume discretizations.

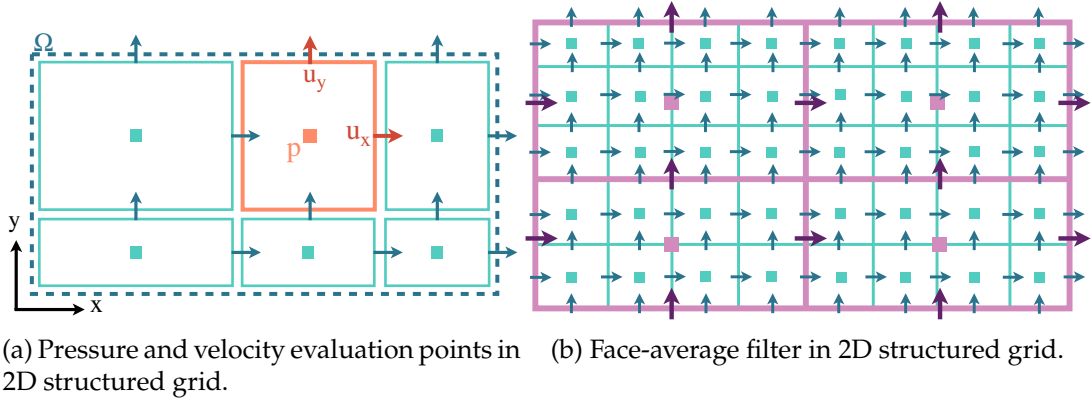


Figure 4.1: Sketch of the fields evaluation and of the face-average filter in the staggered grid framework.

The solver adopted is **energy-conserving**, namely, as happens at the continuous level (Equation (1.4)), it dissipates the kinetic energy, in the absence of external forces and under periodic boundary conditions:

$$\frac{d\mathcal{E}_n}{dt} \leq 0 \quad \forall \text{ time step } t_n. \quad (4.2)$$

This is a crucial property, as it guarantees that, even on under-resolved or marginally resolved grids, the energy in noEFR simulations cannot grow uncontrollably. Conversely, this same property can be leveraged in EF (or EFR) simulations to isolate the effect of the filter and to numerically assess whether it dissipates energy, as will be analyzed in detail in Section 4.2.1.

Proof: the finite-volume solver in the staggered grid framework is energy-conserving.

We here prove that the discrete energy cannot increase. We denote with $\mathbf{M}_u$ and $\mathbf{M}_p$ the element-wise scaling operators containing the velocity and pressure volume sizes. The discrete energy, when $\mathbf{f}_n = \mathbf{0}$, is:

$$\frac{d\mathcal{E}_n}{dt} = \frac{d}{dt} \left(\frac{1}{2} \|\mathbf{u}_n\|_{\mathbf{M}_u}^2 \right) = \frac{d}{dt} \left(\frac{1}{2} \mathbf{u}_n^T \mathbf{M}_u \mathbf{u}_n \right) = -\mathbf{u}_n^T \mathbf{M}_u \mathbf{C}(\mathbf{u}_n) - \mathbf{u}_n^T \mathbf{M}_u \mathbf{G} p_n + \nu \mathbf{u}_n^T \mathbf{M}_u \mathbf{D}_u \mathbf{u}_n.$$

The convective term is skew-symmetric under the incompressibility constraint, namely:

$$\langle \mathbf{C}(\mathbf{u}_n), \mathbf{u}_n \rangle_{\mathbf{M}_u} = \mathbf{u}_n^T (\mathbf{M}_u \mathbf{C}(\mathbf{u}_n) + \mathbf{C}(\mathbf{u}_n) \mathbf{M}_u^T) = 0,$$

This leads to:

$$-\mathbf{u}_n^T \mathbf{M}_u \mathbf{C}(\mathbf{u}_n) \mathbf{u}_n = -\frac{1}{2} \mathbf{u}_n^T (\mathbf{M}_u \mathbf{C}(\mathbf{u}_n) + \mathbf{C}(\mathbf{u}_n) \mathbf{M}_u^T) \mathbf{u}_n = 0.$$

Hence, the convective term is not contributing to the energy variation, just as in the continuous formulation.

The gradient operator is:

$$\mathbf{G} = -\mathbf{M}_u^{-1} \mathbf{H}^T \mathbf{M}_p,$$

where $\mathbf{H}$ is the divergence operator. Hence, exploiting the fact that $\mathbf{H}\mathbf{u}_n = 0$, the pressure contribution becomes:

$$-\mathbf{u}_n^T \mathbf{M}_u \mathbf{G} p_n = \mathbf{u}_n^T \mathbf{M}_u \mathbf{M}_u^{-1} \mathbf{H}^T \mathbf{M}_p p_n = (\mathbf{H}\mathbf{u}_n)^T \mathbf{M}_p p_n = 0.$$

Since the Laplacian matrix is symmetric and negative semidefinite, the viscous contribution to the energy variation can only be negative:

$$\langle \mathbf{D}_u \mathbf{u}_n, \mathbf{u}_n \rangle_{\mathbf{M}_u} = \mathbf{u}_n^T \mathbf{M}_u \mathbf{D}_u \mathbf{u}_n \leq 0 \rightarrow \nu \mathbf{u}_n^T \mathbf{M}_u \mathbf{D}_u \mathbf{u}_n \leq 0.$$

Hence, $\frac{d\mathcal{E}_n}{dt} \leq 0$.

For more details on the solver, we refer the reader to [4].

4.2 DD-EFR methodology

The goal of this section is to introduce the methodological pipeline of [150], sketched in Figure 4.2.

The pipeline is composed of an *offline stage*, i.e., the collection of the reference data and the computation of the optimal data-driven filter $\mathcal{F}^*$ (presented in Section 4.2.1), and an *online stage*, i.e., the DD-EF simulation. During the online simulation, the relaxation parameter χ can be adapted to ensure structure preservation, obtaining two DD-EFR variants, as described in Section 4.2.2.

The ultimate objective is to employ the filter, constructed from snapshots over the time window $[0, T_{\text{train}}]$, in a time-extrapolation regime, i.e., over the interval $[T_{\text{train}}, T]$ with $T \gg T_{\text{train}}$, while maintaining a good accuracy.

4.2.1 Data-driven linear filter

As introduced for the continuous formulation in Section 1.2, an explicit filter operation can be written as a convolution in physical space. In the case of periodic boundaries and uniform grids, we can apply a FFT to the discrete velocity fields. Using the notation $\hat{\cdot}$ to indicate the Fourier coefficients, we can write:

$$\hat{\mathbf{w}} = \mathbf{P}^\dagger \mathbf{w} = \text{FFT}(\mathbf{w}), \text{ and } \mathbf{w} = \mathbf{P} \hat{\mathbf{w}} = \text{FFT}^{-1}(\hat{\mathbf{w}}),$$

where $\mathbf{P}$ is the matrix where the Fourier modes are stored, the † symbol indicates the Hermitian conjugate, and the notation $\hat{\cdot}$ indicates the representation in Fourier space. For non-periodic boundary conditions, this could be extended by the use of lifting functions, like in [276], but in this work we restrict our analysis for simplicity to periodic boundaries.

After applying the FFT, we can write the filter operation as a multiplication in Fourier space [43], obtaining the following relation:

$$\overline{\mathbf{w}} = \mathcal{F} \mathbf{w} = \mathbf{P} \hat{\mathcal{F}} \mathbf{P}^\dagger \mathbf{w}. \quad (4.3)$$

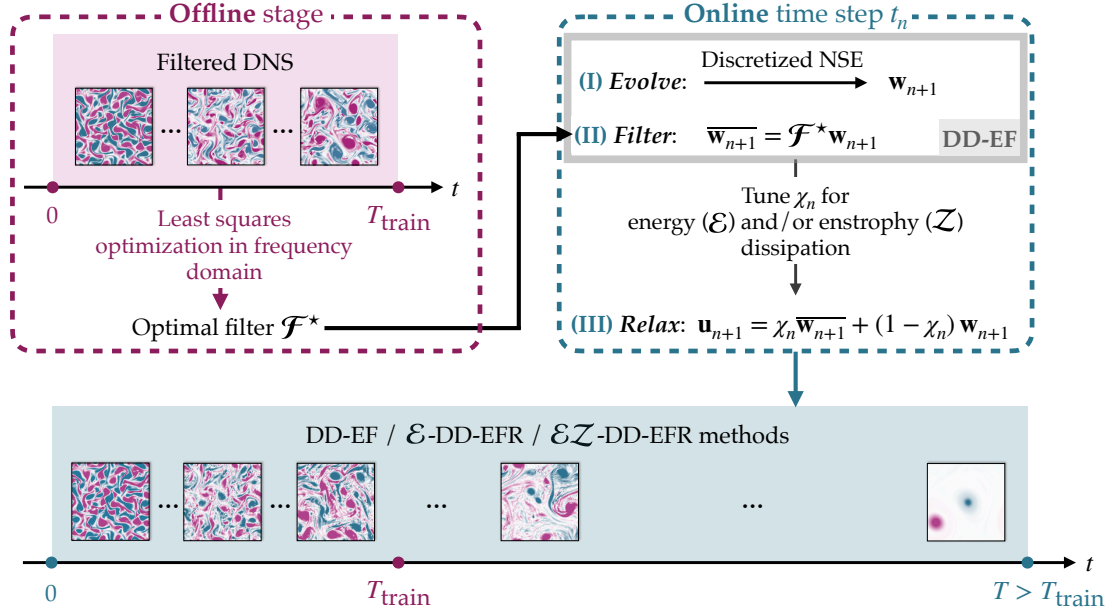


Figure 4.2: Workflow of the DD-EF and DD-EFR frameworks. An optimal filter is learned offline from filtered DNS data, and (optionally) the relaxation parameter is tuned to ensure energy and/or enstrophy dissipation.

The matrix $\widehat{\mathcal{F}} \in \mathbb{C}^{N_u \times N_u}$ is a diagonal matrix, whose entries correspond to the eigenvalues of the filter $\mathcal{F}$.

We now address the question:

How can we derive the optimal entries $\widehat{f}_i^$, $i = 1, \dots, N_u$ from some pre-collected data?*

From now on, we will denote the optimal filter with $\mathcal{F}^*$.

We compute the optimal filter coefficients $\widehat{f}_i^*$ using the reference data $\mathbf{U}^{\text{ref}}$, collected within the time window $[0, T_{\text{train}}]$. We consider a parametrized flow case with parameter μ , given in our tests by the initial condition. For the derivation of the optimal filter, we consider:

- ▶ μ_j , $j = 1, \dots, I_{\text{train}}$ training initial conditions. In the decaying HIT test case, different initial conditions are obtained by changing the seed used to generate the values θ_κ and τ_κ for each wavenumber κ (see the details of the initialization in Section 1.5). Some examples of initial vorticity fields are displayed in Figure 4.3;
- ▶ $n = 1, \dots, N_{\text{train}}$ time instances for each configuration (where N_{train} corresponds to time T_{train}).

Matrix $\mathbf{U}^{\text{ref}}$ collects all the reference filtered DNS snapshots $\mathbf{u}_n^{\text{ref}}(\mu_j)$, with $n = 2, \dots, N_{\text{train}}$, $j = 1, \dots, I_{\text{train}}$. Hence, the matrix $\mathbf{U}^{\text{ref}}$ can be written as:

$$\mathbf{U}^{\text{ref}} = \begin{bmatrix} | & & | & & | & & | \\ \mathbf{u}_2^{\text{ref}}(\mu_1) & \dots & \mathbf{u}_{N_{\text{train}}}^{\text{ref}}(\mu_1) & \dots & \mathbf{u}_2^{\text{ref}}(\mu_{I_{\text{train}}}) & \dots & \mathbf{u}_{N_{\text{train}}}^{\text{ref}}(\mu_{I_{\text{train}}}) \\ | & & | & & | & & | \end{bmatrix}. \quad (4.4)$$

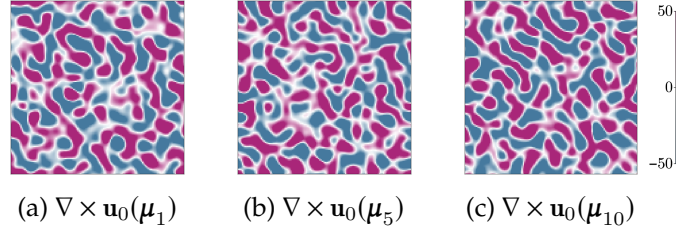


Figure 4.3: *Decaying HIT test case.* The initial vorticity fields at three different random initializations (with seeds 1, 5, and 10). The filtered DNS solutions are considered to train the data-driven filter.

Note that this matrix does not contain the initial state $\mathbf{u}_1^{\text{ref}}$. We also consider another matrix $\mathbf{W}^{\text{ref}}$, which is instead obtained by evolving each filtered DNS velocity solution $\mathbf{u}_n^{\text{ref}}$, $n = 1, \dots, N_{\text{train}} - 1$ for a single time step on the coarse grid. We call the single-time-step-evolved velocities $\mathbf{w}_n^{\text{ref}}$, and we obtain:

$$\mathbf{W}^{\text{ref}} = \begin{bmatrix} | & & | \\ \mathbf{w}_2^{\text{ref}}(\mu_1) & \dots & \mathbf{w}_{N_{\text{train}}}^{\text{ref}}(\mu_{I_{\text{train}}}) \\ | & & | \end{bmatrix} = \text{evolve} \left(\begin{bmatrix} | & & | \\ \mathbf{u}_1^{\text{ref}}(\mu_1) & \dots & \mathbf{u}_{N_{\text{train}}-1}^{\text{ref}}(\mu_{I_{\text{train}}}) \\ | & & | \end{bmatrix} \right). \quad (4.5)$$

Once the matrices $\mathbf{W}^{\text{ref}}$ and $\mathbf{U}^{\text{ref}}$ have been collected, our goal is to find the filter such that the filtered solution on the coarse grid is as close as possible to the reference solution, namely: $\mathcal{F}\mathbf{W}^{\text{ref}} \approx \mathbf{U}^{\text{ref}}$. This approach resembles the **Dynamic Mode Decomposition (DMD)** logic [337, 184], where the learned operator ($\mathcal{F}$) is not the matrix used to evolve the snapshots in time, but the filter applied to the pre-evolved snapshots.

The loss functional we want to minimize, named $\mathcal{L}_{\hat{\mathcal{F}}}$, is the difference between the EF velocities and the reference solutions in the Frobenius norm:

$$\mathcal{L}_{\hat{\mathcal{F}}}(\mathbf{W}^{\text{ref}}, \mathbf{U}^{\text{ref}}) = \|\mathcal{F}\mathbf{W}^{\text{ref}} - \mathbf{U}^{\text{ref}}\|_F^2. \quad (4.6)$$

In Fourier space:

$$\mathcal{L}_{\hat{\mathcal{F}}}(\mathbf{W}^{\text{ref}}, \mathbf{U}^{\text{ref}}) = \|\mathbf{P}^\dagger \mathcal{F} \mathbf{W}^{\text{ref}} - \mathbf{P}^\dagger \mathbf{U}^{\text{ref}}\|_F^2 = \|\widehat{\mathcal{F}} \widehat{\mathbf{W}}^{\text{ref}} - \widehat{\mathbf{U}}^{\text{ref}}\|_F^2, \quad (4.7)$$

where we use the property $\mathcal{F} = \mathbf{P} \widehat{\mathcal{F}} \mathbf{P}^\dagger$.

As previously stated, $\widehat{\mathcal{F}}$ is diagonal and the loss functional becomes:

$$\begin{aligned} \mathcal{L}_{\hat{\mathcal{F}}}(\mathbf{W}^{\text{ref}}, \mathbf{U}^{\text{ref}}) &= \sum_{i=1}^{N_u} \|\widehat{f}_i \widehat{\mathbf{W}}_i^{\text{ref}} - \widehat{\mathbf{U}}_i^{\text{ref}}\|_2^2 = \\ &= \sum_{i=1}^{N_u} (\|\widehat{f}_i\|^2 (\widehat{\mathbf{W}}_i^{\text{ref}})^\dagger \widehat{\mathbf{W}}_i^{\text{ref}} - \widehat{f}_i^\dagger (\widehat{\mathbf{W}}_i^{\text{ref}})^\dagger \widehat{\mathbf{U}}_i^{\text{ref}} - (\widehat{\mathbf{U}}_i^{\text{ref}})^\dagger \widehat{f}_i \widehat{\mathbf{W}}_i^{\text{ref}} + (\widehat{\mathbf{U}}_i^{\text{ref}})^\dagger \widehat{\mathbf{U}}_i^{\text{ref}}), \end{aligned} \quad (4.8)$$

where the superscript i represents the i -th row of the matrices. Differentiating the functional with respect to $\widehat{f}_i^{\dagger 3}$ and setting it to zero, one can find the optimality condition for

³We consider the derivative with respect to $\widehat{f}_i^\dagger$ and not $\widehat{f}_i$ in order to obtain an equation only depending on $\widehat{f}_i$.

the filter coefficients $\widehat{f}_i$:

$$\frac{\partial \mathcal{L}_{\widehat{f}}}{\partial \widehat{f}_i^\dagger} = 2\widehat{f}_i(\widehat{\mathbf{W}}_i^{\text{ref}})^\dagger \widehat{\mathbf{W}}_i^{\text{ref}} - 2(\widehat{\mathbf{W}}_i^{\text{ref}})^\dagger \widehat{\mathbf{U}}_i^{\text{ref}} = 0. \quad (4.9)$$

Solving Equation (4.9) leads to the optimal filter coefficients:

$$\widehat{f}_i^\star = \frac{(\widehat{\mathbf{W}}_i^{\text{ref}})^\dagger \widehat{\mathbf{U}}_i^{\text{ref}}}{(\widehat{\mathbf{W}}_i^{\text{ref}})^\dagger \widehat{\mathbf{W}}_i^{\text{ref}}}, \quad \forall i = 1, \dots, N_u. \quad (4.10)$$

The resulting optimal filter is $\widehat{\mathcal{F}}^\star$ in the Fourier space, and $\mathcal{F}^\star$ in the physical space. We call **DD-EF** the **EF** approach where the filtered velocity field at each time step t_n is computed as:

$$\mathbf{u}_n = \overline{\mathbf{w}}_n = \mathbf{P}^\dagger \widehat{\mathcal{F}}^\star \mathbf{P} \mathbf{w}_n.$$

We remark that the data-driven filter is fixed at all the times considered, and can be built offline using an affordable amount of data.

Stability analysis of the filter. In order to ensure the filter dissipativity, i.e. stability, one should guarantee that the energy decreases after applying the filter at each time step t_n :

$$\|\mathcal{F} \mathbf{w}_n\|_2^2 \leq \|\mathbf{w}_n\|^2. \quad (4.11)$$

If we write the above inequality in Fourier space, we obtain:

$$\sum_{i=1}^{N_u} |\widehat{f}_i|^2 |\widehat{\mathbf{w}}_{i,n}|^2 \leq \sum_{i=1}^{N_u} |\widehat{\mathbf{w}}_{i,n}|^2. \quad (4.12)$$

This condition is not automatically satisfied for our optimal filter in Equation (4.10), since it depends on the reference data $\mathbf{U}^{\text{ref}}$ and $\mathbf{W}^{\text{ref}}$.

We then analyze the effect of the data-driven filter in different training configurations, summarized in Table 4.1. The data are collected from the decaying **HIT** test case, and, since the simulation time step is small ($\Delta t = 5 \times 10^{-4}$), the snapshots are subsampled and taken every 10 time steps in order to reduce the computational burden.

Table 4.1: Data regimes considered to compute the filter matrix for DD-EF simulations. Data refers to the decaying HIT simulation at $Re = 40\,000$.

Train		Test	
T_{train}	I_{train}	T	I_{test}
1 s	1	10 s	5
1 s	10		
3 s	1		
3 s	10		

We analyze the effect of the filter on the velocity field in Figure 4.4 by considering the 2D average of the filter elements $\widehat{\mathcal{F}}^\star$ for both x and y components. We first associate an average filter value to all modes κ with wavenumber magnitude $\kappa = \|\mathbf{k}\| = \sqrt{k_x^2 + k_y^2}$. Such average value is computed as the mean magnitude of the filter coefficients $\widehat{f}_i^\star$ across all modes:

$$\langle |\widehat{\mathcal{F}}^\star(\kappa)| \rangle = \frac{1}{|\mathcal{B}_\kappa|} \sum_{i=1}^{|\mathcal{B}_\kappa|} \widehat{f}_i^\star \mathcal{M}_i(\kappa), \quad (4.13)$$

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

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

and $|\mathcal{B}_\kappa| = \sum_i \mathcal{M}_i(\kappa)$ is the number of Fourier modes in $\mathcal{B}_\kappa$ ⁴. Values exceeding one indicate amplification of the corresponding frequency components, whereas values below one denote attenuation; magnitudes close to one imply that the filter leaves those frequencies largely unaltered. Figure 4.4 shows that the filter slightly amplifies the low-frequency components—those carrying most of the energy—while damping the medium-to-high frequency range. Moreover, the behaviour of the data-driven filter is highly sensitive to the amount of data used in the least-squares problem. In particular, under a *data-scarcity* regime, the filter exhibits more pronounced oscillations, and the optimization may produce values greater than one even at intermediate frequencies (e.g., $\kappa = 9, 10, 14$ for the y component). Such energy injection at intermediate scales can significantly increase the velocity magnitude, potentially leading to numerical instability or even simulation blow-up.

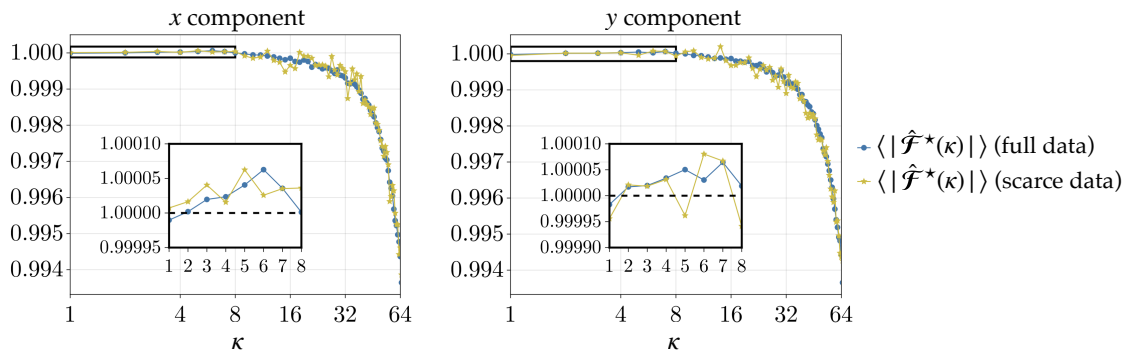


Figure 4.4: 2D average of x and y components of the filter matrix $\widehat{\mathcal{F}}^\star$ for the cases $T_{\text{train}} = 3$ s and $I_{\text{train}} = 10$ (full-data), and $T_{\text{train}} = 1$ s and $I_{\text{train}} = 1$ (scarce-data).

The condition (4.12) is instead satisfied by the **DF**.

⁴ $\langle |\widehat{\mathcal{F}}^\star(\kappa)| \rangle$ can also be seen as a transfer function, evaluated in a 2D test case. The transfer functions of some classical filters are represented in Figure 1.2

Remark: the stability condition for the **DF** filter.

The **DF** matrix in Fourier space has diagonal entries:

$$\mathcal{F}_{ii} = \widehat{f}_i = \frac{1}{1 - 2\delta^2\lambda_i}, \quad i = 1, \dots, N_u. \quad (4.14)$$

In the above expression, λ_i are the eigenvalues of the diffusion operator $\mathbf{D}_u$. Since the eigenvalues λ_i are all real and negative [292], $|\widehat{f}_i| = \frac{1}{1 - 2\delta^2\lambda_i} \leq 1$. The same stability condition has been proved at the continuous level in Section 1.2.

Additionally, note that the **DF** satisfies:

$$|\widehat{f}_i|^2 \leq 0, \quad \forall i = 1, \dots, N_u, \quad (4.15)$$

which is a *sufficient* but not a *necessary* condition to achieve stability. It may indeed lead to overdiffusive behaviour, as it enforces energy dissipation at each frequency. Our goal is instead to guarantee the energy conservation (Equation (4.12)) without overdiffusion, allowing for some frequencies to have an increasing energy.

4.2.2 Energy and enstrophy-conserving approaches

The stability condition (Equation (4.12)) can be enforced by reintroducing the relaxation step, with a smart choice of the relaxation parameter χ at each time step t_n , such that:

$$\|\mathbf{u}_n\|_2^2 = \|(1 - \chi)\mathbf{w}_n + \chi\overline{\mathbf{w}}_n\|_2^2 \leq \|\mathbf{w}_n\|_2^2. \quad (4.16)$$

In these expressions, $\mathbf{w}_n$, $\overline{\mathbf{w}}_n$, and $\mathbf{u}_n$ correspond to the *online* DD-EFR solution approximations at time step t_n , and not to the snapshot matrices $\mathbf{W}^{\text{ref}}$ and $\mathbf{U}^{\text{ref}}$ used to derive the filter $\widehat{\mathcal{F}}^\star$.

Once computed *offline* the filter components as in Equation (4.10), we can adaptively tune $\chi_n = \chi(t_n)$ at each *online* time step. From (4.16), we derive the following inequality:

$$(\chi_n)^2 \tilde{a}_n^\mathcal{E} + 2\chi_n \tilde{b}_n^\mathcal{E} \leq 0, \quad (4.17)$$

where

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

Since $0 \leq \chi_n \leq 1$, to maintain a convex combination in the **EFR** approach, the inequality (4.17) leads to the conditions:

$$\left\{ \begin{array}{ll} \chi_n \leq -\frac{2\tilde{b}_n^\mathcal{E}}{\tilde{a}_n^\mathcal{E}} & \text{if } \tilde{b}_n^\mathcal{E} \leq 0, \\ \chi_n = 0 & \text{otherwise.} \end{array} \right. \quad (4.18)$$

Hence, the online procedure we use to select the optimal relaxation parameter $\chi_{\mathcal{E}\text{-opt}}$ at each time step t_n is the following:

- If constraint (4.16) is satisfied for $\chi_n = 1$:

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

i.e., we only perform a filtering step (DD-EF), and relaxation is not needed.

- If constraint (4.16) is *not* satisfied:

- $\chi_{\mathcal{E}\text{-opt}} = -\frac{2\tilde{b}_n^{\mathcal{E}}}{\tilde{a}_n^{\mathcal{E}}}$, if $\tilde{b}_n^{\mathcal{E}} \leq 0$ (which is the *least dissipative value* that ensures the constraint);
- $\chi_{\mathcal{E}\text{-opt}} = 0$, if $\tilde{b}_n^{\mathcal{E}} > 0$. We simply use the noEFR solution ($\mathbf{w}_n$) without applying any filtering or relaxation. This yields an energy-stable result because the solver is energy-conserving itself. However, the simulation may exhibit spurious oscillations due to the coarse resolution, making it a less desirable option in practice.

Since computing the inner products in the frequency domain does not offer any computational speedup, the computations for $\tilde{a}_n^{\mathcal{E}}$ and $\tilde{b}_n^{\mathcal{E}}$ are all carried out in physical space.

This energy-conserving approach is called **Energy-preserving Data-Driven Evolve-Filter-Relax ($\mathcal{E}$ -DD-EFR)**. The $\mathcal{E}$ -DD-EFR approach does not inherently ensure a monotonic decrease in enstrophy, a property that should hold for the viscous 2D DNS under periodic boundaries and without external forces [245]. However, enstrophy dissipation is important, as an increase in enstrophy often indicates the presence of unphysical spurious oscillations in the solution. This occurs because the vorticity is sensitive to high-frequency modes (large wavenumbers). Enstrophy dissipation may be imposed by the following inequality at each time step t_n :

$$\|\nabla \times \mathbf{u}_n\|_2^2 = \|(1 - \chi)\nabla \times \mathbf{w}_n + \chi\nabla \times \bar{\mathbf{w}}_n\|_2^2 \leq \|\nabla \times \mathbf{w}_n\|_2^2, \quad (4.19)$$

where $(\nabla \times)$ denotes in this case the discrete curl operator. The above expression may be rewritten in a form similar to the one found for the energy-preserving inequality, namely:

$$(\chi_n)^2 \tilde{a}_n^{\mathcal{Z}} + 2\chi_n \tilde{b}_n^{\mathcal{Z}} \leq 0, \quad (4.20)$$

where

$$\tilde{a}_n^{\mathcal{Z}} = \|\nabla \times \mathbf{w}_n - \nabla \times \bar{\mathbf{w}}_n\|_2^2 \quad \text{and} \quad \tilde{b}_n^{\mathcal{Z}} = (\nabla \times \mathbf{w}_n)^T \nabla \times \bar{\mathbf{w}}_n - \|\nabla \times \mathbf{w}_n\|_2^2.$$

Exactly as for the energy inequality, the optimal χ satisfying inequality (4.20) is $\chi_{\mathcal{Z}\text{-opt}} = -2\frac{\tilde{b}_n^{\mathcal{Z}}}{\tilde{a}_n^{\mathcal{Z}}}$. Also in this case, we have:

$$\left\{ \begin{array}{ll} \chi_n \leq -\frac{2\tilde{b}_n^{\mathcal{Z}}}{\tilde{a}_n^{\mathcal{Z}}} & \text{if } \tilde{b}_n^{\mathcal{Z}} \leq 0, \\ \chi_n = 0 & \text{otherwise.} \end{array} \right. \quad (4.21)$$

As for $\tilde{b}_n^{\mathcal{E}}$, $\tilde{b}_n^{\mathcal{Z}}$ may also be negative. In that case, one would simply retain $\chi_{\mathcal{Z}\text{-opt}} = 0$ (noEFR), as in the $\mathcal{E}$ -DD-EFR approach.

We can finally obtain an energy- and enstrophy-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 and less efficient due to the need to compute discrete gradients within the staggered grid framework. We refer to this approach as **Energy- and Enstrophy-preserving Data-Driven Evolve-Filter-Relax (EZ-DD-EFR)**.

4.3 Numerical results of DD-EFR

The novel approaches introduced in Section 4.2 are first tested on the 2D decaying HIT test case (Section 4.3.1), and then on the 2D Kolmogorov flow (Section 4.3.2), both at $Re = 4 \times 10^4$. As previously stated, we adopt the same setting as Section 1.3.1. Time integration is performed using a fourth-order explicit Runge–Kutta scheme [293]. We adopt a 512^2 resolution for the well-resolved grid used to simulate the DNS, and a 128^2 resolution for the marginally-resolved grid.

The DD-EF, $\mathcal{E}$ -DD-EFR, EZ-DD-EFR solutions are compared with other state-of-the-art approaches on the marginally-resolved grid, namely standard EFR, standard EF, and the Smagorinsky model (see section 1.2.1). Each of the state-of-the-art models depends on one or more parameters that require careful calibration and are optimized using a gradient-based method.

The accuracy of the models is evaluated in terms of discrepancy with respect to the filtered DNS in the global kinetic energy $\mathcal{E}_n$, in the global enstrophy $\mathcal{Z}_n$, for $n = 1, \dots, N_T$, and in the energy spectrum. We employ a range of $I_{\text{test}} = 5$ 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. We adopt for the global error in the kinetic energy and in the spectrum the same expressions of Equation (3.14). Similar to the error for the energy, the error in the enstrophy is:

$$\text{err}_{\mathcal{Z}} = \frac{1}{N} \sum_{n=1}^N \left| \frac{\mathcal{Z}_n - \mathcal{Z}_n^{\text{ref}}}{\mathcal{Z}_n^{\text{ref}}} \right|. \quad (4.22)$$

The error expressions are evaluated for all the I_{test} configurations, and then we evaluate the statistics in terms of average value and standard deviation.

4.3.1 Decaying homogeneous isotropic turbulence test case

We start the analysis by considering the *full-data* regime to compute the filter, namely $T_{\text{train}} = 3 \text{ s}$, $I_{\text{train}} = 10$. We performed then the DD-EF and DD-EFR online simulations and investigated the performance in terms of the total kinetic energy and enstrophy trend in time, and of the energy spectrum at fixed time $t = 1 \text{ s}$ (Figure 4.5). In Figures 4.6 and 4.7, we can see the corresponding vorticity fields at fixed times $t = 0.3$ and $t = 1 \text{ s}$. From the plots we can draw the following conclusions:

Remark: parameters' optimization in state-of-the-art filtering methods.

To find the optimal parameters in state-of-the-art methodologies, we employ a *finite-difference stochastic gradient descent*, where the loss function is derived from $I_{\text{train}} = 10$ ensemble-averaged simulations performed in the time interval $[0, T_{\text{train}}]$, $T_{\text{train}} = 1$ s. The optimization seeks to minimize the enstrophy mismatch by adjusting the parameter of interest. More in detail, the loss function is the relative mean squared error (RMSE) in the global enstrophy.

We selected the enstrophy error as objective function because other metrics like the energy error do not provide significant insights on the performance of the method. The global enstrophy instead provides direct insight into small-scale structures and dissipative behaviour. More details on the optimization algorithm are provided in Appendix A of [150].

Table 4.2 reports the optimal values for the parameters. Note that in the standard EFR approach, we adopt a fixed filter radius ($\delta = h$, where h is the mesh size) and we only optimize the relaxation parameter χ .

Table 4.2: Optimized parameters in state-of-the-art approaches methodologies, used for comparison with DD-EF and DD-EFR.

Model	Parameter	Parameter range	Optimal value
Smagorinsky	C_S	$[0, 1]$	0.085
Standard EF	δ	$[0.1 \eta, 10 \eta]$	$1.64 \times 10^{-4} \simeq 0.5 \eta$
Standard EFR	χ (fixed $\delta = h$)	$[0.1 \Delta t, 10 \Delta t]$	$2.3 \times 10^{-3} \simeq 4.5 \Delta t$

- DD-EF and $\mathcal{E}$ -DD-EFR outperform all existing methods in terms of kinetic energy, global enstrophy, and spectrum. The vorticity fields at $t = 0.3$ closely match the filtered DNS in local features (Figure 4.6b and 4.6c), while being *qualitatively similar* to the reference at later times (Figure 4.7b and 4.7c).
- 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 positively contribute to the filter stability. In fact, this approach inherently yields decaying energy and enstrophy, without the need for explicit constraints. Hence, there is no need to explicitly impose enstrophy dissipation.
- 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 produce overdiffusive solutions, whereas the data-driven filter is able to better recover the dissipated energy. From a qualitative standpoint, the vorticity fields appear close to the reference at early times (Figures 4.6e, 4.6f, and

4.6g), but become noticeably overdissipative at later times, such as at $t = 1$ s (Figures 4.7e, 4.7f, and 4.7g).

- The noEFR simulation shows significantly larger enstrophy values in the time interval $[0, 3]$ s, which is also evident from the presence of noise and spurious oscillations in the vorticity field (Figures 4.6d and 4.7d). As time progresses and turbulence dissipates, the noEFR enstrophy decreases and eventually aligns with the filtered DNS behaviour. Interestingly, the noEFR solution provides a good match to the filtered DNS in terms of global kinetic energy among the considered methods. This highlights that relying solely on global kinetic energy as a metric can be misleading, as it may mask deficiencies in capturing fine-scale structures and local flow dynamics, which are more accurately reflected by enstrophy and vorticity-based quantities.

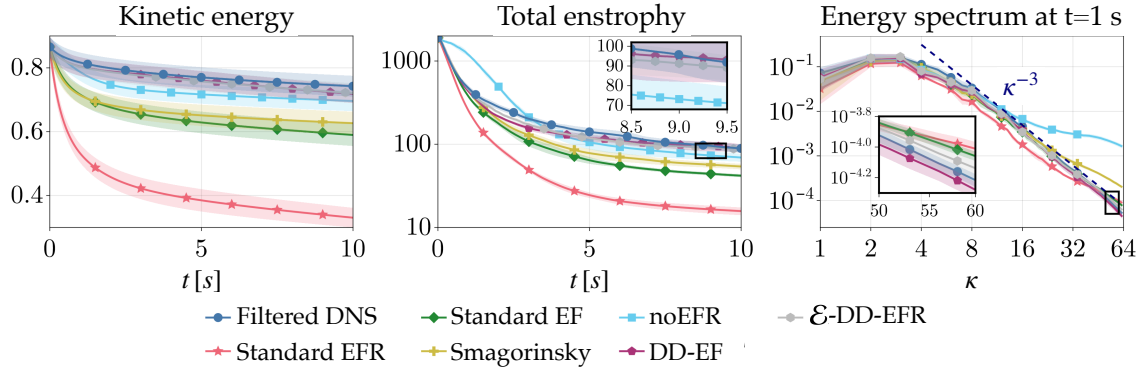


Figure 4.5: *Decaying HIT test case*. Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed methodologies (DD-EF and $\mathcal{E}$ -DD-EFR), noEFR, the state-of-the-art approaches, and the reference filtered DNS. The data-driven filter is computed when $T_{\text{train}} = 3$ s and $I_{\text{train}} = 10$ (full-data regime). For each method, the plots show the average values (solid lines) and the 95% confidence interval among $I_{\text{test}} = 5$ test configurations.

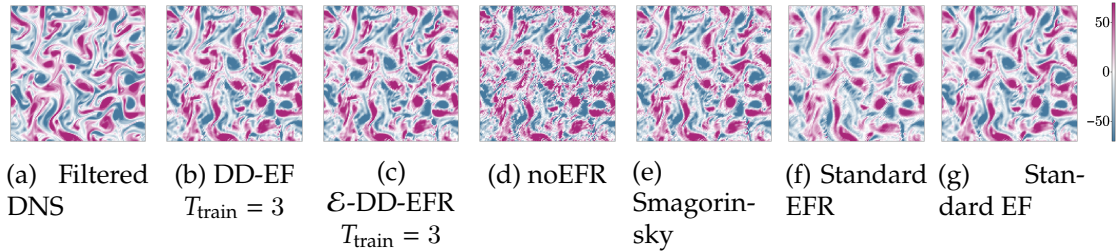


Figure 4.6: *Decaying HIT test case*. Vorticity fields at $t = 0.3$ for different methodologies in one of the I_{test} configurations.

It is also instructive to examine the online time evolution of the optimal relaxation parameter $\chi_{\mathcal{E}\text{-opt}}$. Figure 4.8 reports both the values' distribution and temporal evolution of $\chi_{\mathcal{E}\text{-opt}}$ and a representative test initialization. The parameter predominantly oscillates between 0 and 1. As discussed in Section 4.2.2, enforcing the energy constraint leads

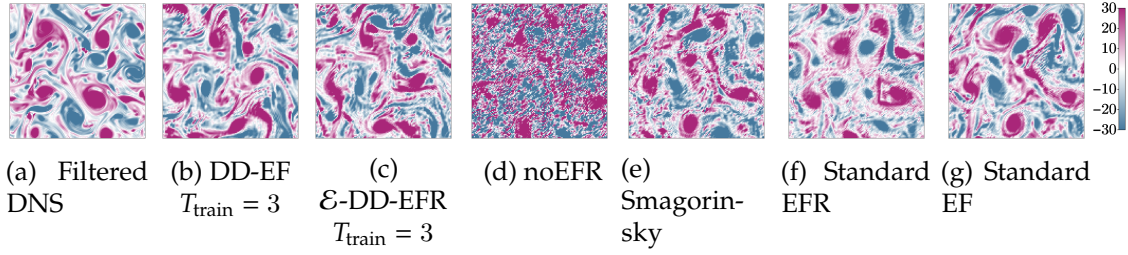


Figure 4.7: *Decaying HIT test case*. Vorticity fields at $t = 1$ for different methodologies in one of the I_{test} configurations.

Remark: on the stability of noEFR solutions.

It is important to note the difference in the noEFR behaviour between Figure 4.5 and Figure 3.16. The simulation remains stable in the framework of the present Chapter because the solver is energy- and enstrophy-conserving by construction. In Chapter 3, the FEM solver does not have the same property. The stability of the evolve step is instead controlled in the RL framework by adding the contribution of the NSE residuals inside the rewards, in the DF-RL-EF approach.

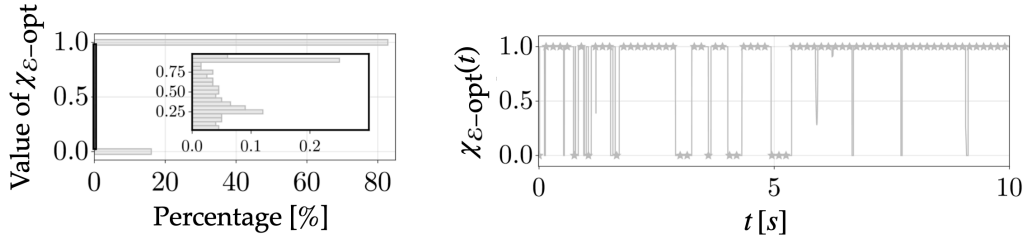


Figure 4.8: *Decaying HIT test case*. Distribution across values and time evolution of the relaxation parameter in the energy-conserving DD-EFR simulation, when the data-driven filter is computed with $T_{\text{train}} = 3$ s, $I_{\text{train}} = 10$ (full-data).

to $\chi_{\varepsilon-\text{opt}} = 0$ whenever $\tilde{b}_n^{\varepsilon} > 0$, since this is the only admissible value satisfying the constraint and corresponds to the noEFR configuration. Moreover, the most frequently observed value (approximately 85% of the time) is $\chi_{\varepsilon-\text{opt}} = 1$, which corresponds to DD-EF. This means that most of the times the filter is already energy-conserving and there is no need to relax. At later times (Figure 4.7), a qualitative analysis reveals that the DD-EF(R) methods preserve more coherent vortex structures compared to state-of-the-art approaches. This is further supported by the spectral analysis at $t = 1$ s, shown in Figure 4.5. In particular, DD-EF(R) exhibit a closer agreement with the filtered DNS at intermediate-to-high wavenumbers, indicating reduced numerical dissipation and improved retention of vortex structures. In contrast, the Smagorinsky model retains excessive energy at large wavenumbers, while standard EFR shows a steeper spectral decay at intermediate wavenumbers.

We now analyse the performance of the filter in the case of *scarce-data* ($T_{\text{train}} = 1$ s, $I_{\text{train}} = 1$) which is the most challenging one. Figure 4.9 shows the statistics for kinetic

energy, enstrophy, and energy spectrum at fixed time, in that specific regime.

Figure 4.9 shows that:

- ▶ **DD-EF** exhibits instability in both energy and enstrophy, which grow and eventually blow up after $t = 5$ s. This confirms that the scarce-data filter can be unstable, as it may inject energy at intermediate wavenumbers, as previously observed from Figure 4.4.
- ▶ **$\mathcal{E}$ -DD-EFR**, which enforces the energy dissipation constraint, shows a stable and monotonically decreasing energy behaviour. However, it becomes inaccurate in reproducing the global enstrophy after $t = 6$ s.
- ▶ **$\mathcal{EZ}$ -DD-EFR** is in this case necessary to stabilize the enstrophy. It remains stable in both energy and enstrophy, and provides an accurate approximation of the filtered DNS in terms of enstrophy. Nevertheless, both **$\mathcal{E}$ -DD-EFR** and **$\mathcal{EZ}$ -DD-EFR** yield overdissipative energy behaviour at large times, likely due to the limited amount of training data used in constructing the filter.

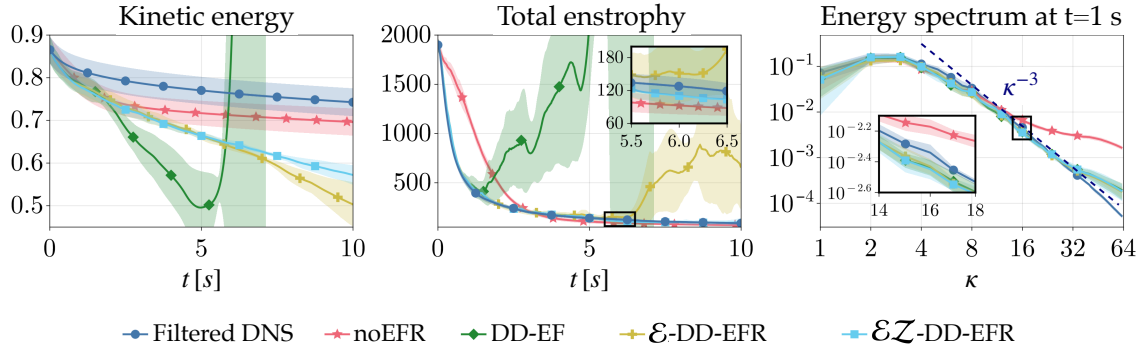


Figure 4.9: *Decaying HIT test case.* Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed methodologies (DD-EF, $\mathcal{E}$ -DD-EFR, $\mathcal{EZ}$ -DD-EFR), noEFR, and the reference filtered DNS. The data-driven filter is computed when $T_{\text{train}} = 1$ s and $I_{\text{train}} = 1$ (scarce-data regime). For each method, the plots show the average values (solid lines) and the 95% confidence interval among $I_{\text{test}} = 5$ test configurations.

Figure 4.10 shows the time evolution of $\chi_{\mathcal{E}\text{-opt}}$ and $\chi_{\mathcal{EZ}\text{-opt}}$ for a representative test initialization. Compared to the full-data case, $\chi_{\mathcal{E}\text{-opt}}$ exhibits more pronounced temporal oscillations. This behaviour reflects the increased instability and error introduced by the limited amount of training data. Furthermore, since $\chi_{\mathcal{EZ}\text{-opt}}$ satisfies both energy and enstrophy dissipation constraints, it displays greater temporal variability. In particular, $\chi_{\mathcal{EZ}\text{-opt}}$ more frequently assumes values within the interval $(0, 1)$, especially for $t > 5$ s, whereas $\chi_{\mathcal{E}\text{-opt}}$ predominantly converges to the extreme values 0 (noEFR) or 1 (DD-EF). Overall, these results indicate that the scarce-data filter may lead to unstable dynamics due to the limited training dataset. However, enforcing the energy and/or enstrophy

constraints (Equations (4.18) and (4.21), respectively) stabilizes the simulation. It is important to note that these constraints are applied *online*, by modifying only the relaxation step, without altering the filter itself. This emphasizes the *flexibility* of the approach, as stability can be enforced without redesigning the underlying filter.

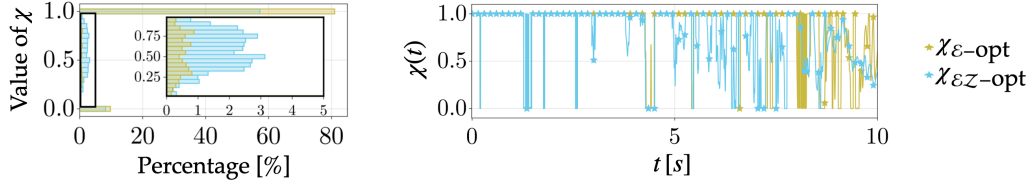


Figure 4.10: *Decaying HIT test case*. Distribution across values and time evolution of the relaxation parameter in the energy-conserving DD-EFR simulation, when the data-driven filter is computed with $T_{\text{train}} = 1$ s, $I_{\text{train}} = 1$ (scarce-data).

Given the observed performance differences between filters trained on full and scarce-datasets, we aim to address the following questions:

(i) *How do scarce-data and full-data filters compare in terms of overall accuracy across different physical metrics?* and

(ii) *To what extent can the introduction of constraints reduce this gap in terms of overall error?*

We answer the questions by measuring the global errors of Equations (3.14) and (4.22) averaged both in times and across the I_{test} test configurations. Instead of considering the statistics over the entire test time interval $[0, T]$, we consider a number of time steps $N_{t=3}$, corresponding to $t = 3$ s. We choose the restricted 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 4.11 display the average and the confidence interval of the errors.

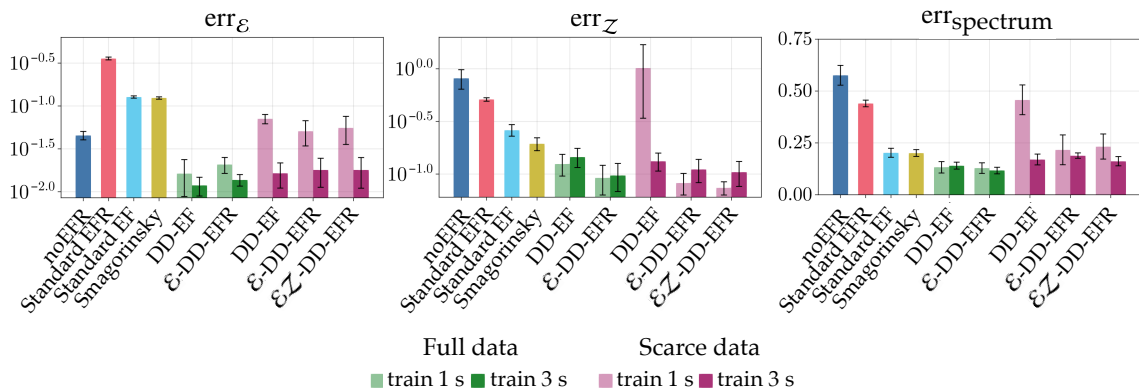


Figure 4.11: *Decaying HIT test case*. Relative error of the total kinetic energy, total enstrophy, and absolute error in logarithmic scale of the spectrum, of the different methodologies with respect to the filtered DNS. The errors are averaged in the time interval $[0, 3]$, and then evaluated as average values and confidence intervals (error bars) among $I_{\text{test}} = 5$ test configurations.

From Figure 4.11, we can draw the following conclusions:

- DD-EF(R) approaches with $T_{\text{train}} = 3$ s provide the best performance across all the considered metrics.
- Among the data-driven methods, **DD-EF** and **$\mathcal{E}$ -DD-EFR** with $T_{\text{train}} = 3$, $I_{\text{train}} = 10$ (full-data setting) achieve the highest accuracy with respect to the filtered **DNS** in terms of global kinetic energy and energy spectrum.
- As already observed in Figure 4.9, **$\mathcal{E}$ -DD-EFR** and **$\mathcal{E}\mathcal{Z}$ -DD-EFR** with $T_{\text{train}} = 1$, $I_{\text{train}} = 1$ (scarce-data setting) significantly improve upon the **DD-EF** results, particularly in terms of enstrophy. In contrast, only marginal improvements are observed for $T_{\text{train}} = 3$, $I_{\text{train}} = 1$, likely because the filter, being trained on a larger dataset, is inherently more stable.
- In both full- and scarce-data settings, the highest accuracy in energy and energy spectrum is achieved for $T_{\text{train}} = 3$ s, consistently with the larger amount of available training data. However, more accurate enstrophy predictions are obtained for $T_{\text{train}} = 1$ s. A possible explanation is that the system exhibits more chaotic dynamics at early times, while dissipation becomes dominant at later stages. Consequently, filters trained over shorter time intervals may better capture sharp gradients compared to those constructed over longer time windows.

Hence, we can answer the previously posed questions as follows:

- (i) Full-data filters generally provide higher accuracy in terms of kinetic energy and energy spectrum.
- (ii) In the scarce-data regime, the introduction of constraints significantly improves enstrophy accuracy, in some cases outperforming even the full-data configurations.

4.3.2 Kolmogorov test case

The second test case is configured similarly to the first, but includes an additional forcing term. To prevent energy decay during long-time simulations, we introduce a Kolmogorov-type body force that injects energy into the system at prescribed wavenumbers, following [4, 58, 202, 173]. The forcing term acts only in the horizontal direction and is defined as

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

where $k_f = 4$ denotes the injection wavenumber. The amplitude c_f is chosen to ensure that the system reaches a statistically stationary energy level high enough to sustain turbulence, while avoiding unbounded energy growth. In our case, we set $c_f = 0.65$. To assess the extrapolation capability of the proposed approach, the DD-EF(R) methods employ the same optimal filter matrix $\widehat{\mathcal{F}}^*$ obtained from the decaying **HIT** test case, thereby performing an explicit extrapolation to this new scenario. For consistency, the state-of-the-art methods are also evaluated using the optimal parameters identified in the first test case.

We start by investigating the DD-EF(R) performance in the full-data regime ($I_{\text{train}} = 10$, $T_{\text{train}} = 3$ s) for the Kolmogorov case. We examine the time evolution of global kinetic energy and enstrophy, as well as the energy spectrum at the fixed time $t = 1$ s, comparing these quantities with the filtered DNS reference (Figure 4.12). The results indicate that:

- ▶ DD-EF and $\mathcal{E}$ -DD-EFR deliver the best overall performance across all considered metrics. While both methods achieve comparable accuracy in energy and enstrophy, $\mathcal{E}$ -DD-EFR shows a closer agreement with the reference spectrum, particularly at high wavenumbers.
- ▶ As expected for the Kolmogorov flow, the reference kinetic energy remains statistically stationary over time, whereas state-of-the-art methods yield overdissipative solutions, consistent with the observations from the first test case.
- ▶ The reference global enstrophy follows a trend similar to the previous case, generally decreasing over time. As before, the noEFR solution exhibits larger gradients and higher enstrophy at early times, which are progressively dissipated for $t > 3$ s.

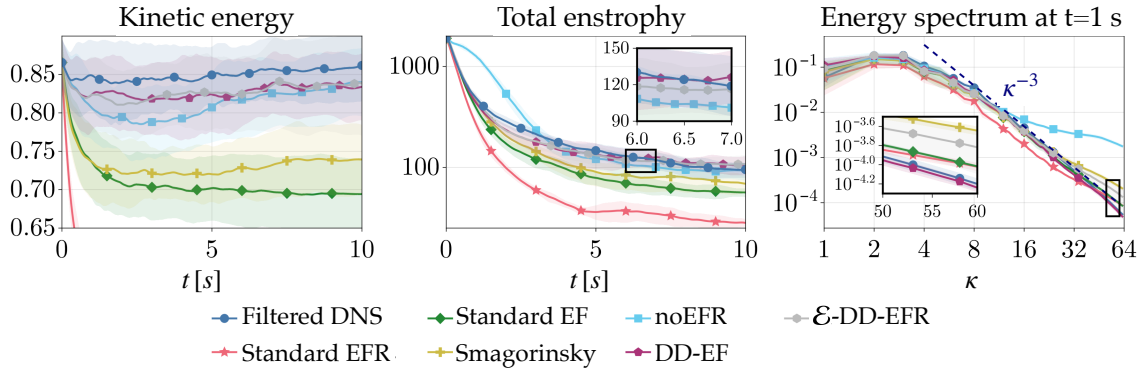


Figure 4.12: *Kolmogorov test case*. Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed methodologies (DD-EF and $\mathcal{E}$ -DD-EFR), noEFR, the state-of-the-art approaches, and the reference filtered DNS. The data-driven filter is computed (from the decaying HIT reference data) when $T_{\text{train}} = 3$ s and $I_{\text{train}} = 10$ (full-data regime). For each method, the plots show the average values (solid lines) and the 95% confidence interval among $I_{\text{test}} = 5$ test configurations.

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 4.8). For the sake of compactness, we skip here this analysis and we consider the filter effect when it is built from the same time window ($T_{\text{train}} = 3$ s) but in a scarce-data regime ($I_{\text{train}} = 1$). In contrast to the results reported for the decaying HIT case for $T_{\text{train}} = 1$ s (Figure 4.9), Figure 4.13 shows that the DD-EF(R) approaches yield more accurate and less dissipative solutions. This improvement is consistent with the larger number of temporal snapshots used in the filter construction. More specifically, DD-EF and $\mathcal{E}$ -DD-EFR exhibit similar trends in the evolution of kinetic energy and enstrophy. However, DD-EF displays instabilities and reduced accuracy in the global enstrophy,

particularly for $t > 6$ s. The enforcement of energy and enstrophy constraints stabilizes the enstrophy at later times and leads to a better agreement with the reference energy spectrum at intermediate wavenumbers. Furthermore, incorporating the enstrophy constraint ($\mathcal{E}\mathcal{Z}$ -DD-EFR) enhances the accuracy of the kinetic energy, especially for $t > 5$ s.

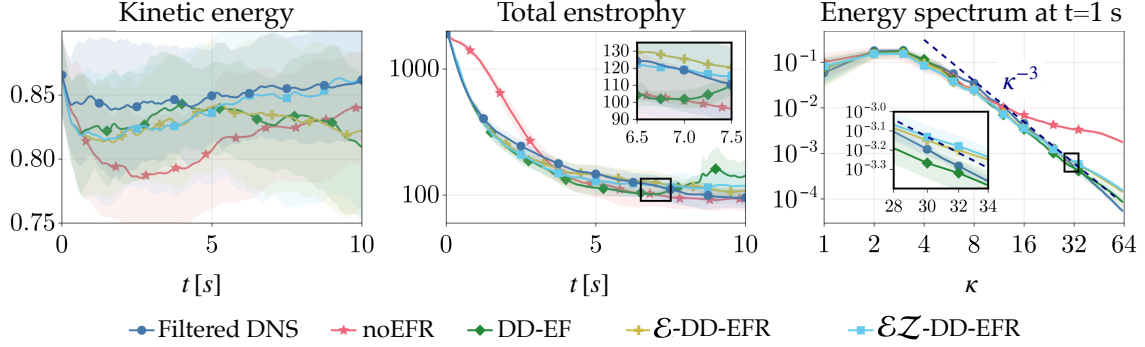


Figure 4.13: *Kolmogorov test case*. Time evolution of total kinetic energy, total enstrophy, and spectrum of the kinetic energy at time $t = 1$ s, for the proposed methodologies (DD-EF, $\mathcal{E}$ -DD-EFR, $\mathcal{E}\mathcal{Z}$ -DD-EFR), noEFR, and the reference filtered DNS. The data-driven filter is computed (from the decaying HIT reference data) when $T_{\text{train}} = 3$ s and $I_{\text{train}} = 1$ (scarce-data regime). For each method, the plots show the average values (solid lines) and the 95% confidence interval among $I_{\text{test}} = 5$ test configurations.

The time evolution of the optimal parameters $\chi_{\mathcal{E}\text{-opt}}$ and $\chi_{\mathcal{E}\mathcal{Z}\text{-opt}}$ is shown in Figure 4.14. Consistent with Figure 4.10, the optimal value is predominantly $\chi = 1$ in both cases. However, $\chi_{\mathcal{E}\mathcal{Z}\text{-opt}}$ exhibits stronger temporal oscillations, particularly at later simulation times. A comparison between Figures 4.10 ($T_{\text{train}} = 1$ s) and 4.14 ($T_{\text{train}} = 3$ s) highlights that a larger training window enhances stability, resulting in reduced oscillations of the optimal χ and fewer intermediate values. Overall, these results indicate that the proposed filtering approach, although trained on a different configuration, maintains its effectiveness in the present test case, both in full- and scarce-data regimes. This supports the method's ability to *extrapolate* and *adapt* to qualitatively similar flow conditions.

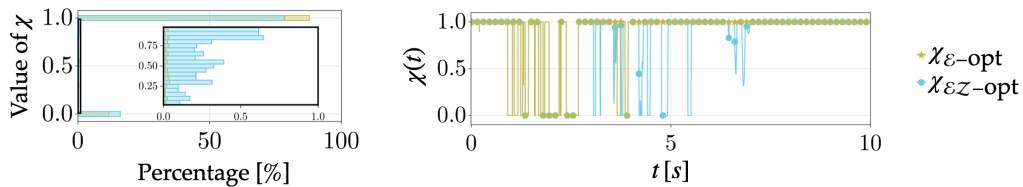


Figure 4.14: *Kolmogorov test case*. Distribution across values and time evolution of the relaxation parameter in the energy-conserving DD-EFR simulation, when the data-driven filter is computed with $T_{\text{train}} = 3$ s, $I_{\text{train}} = 1$ (scarce-data).

As in the first test case, we now compare full- and scarce-data regimes in terms of global errors (see the expressions in (3.14) and (4.22)). Figure 4.15 presents the global error analysis for the Kolmogorov test case, which closely resembles the behaviour observed in Figure 4.11. In this case, since this test case is not dissipating energy in time, we evaluate

the global metrics considering N_T as the total number of time steps, rather than $N_{t=3}$, and we restrict the comparison to the best-performing DD-EF(R) configurations, namely those with $T_{\text{train}} = 3$ s.

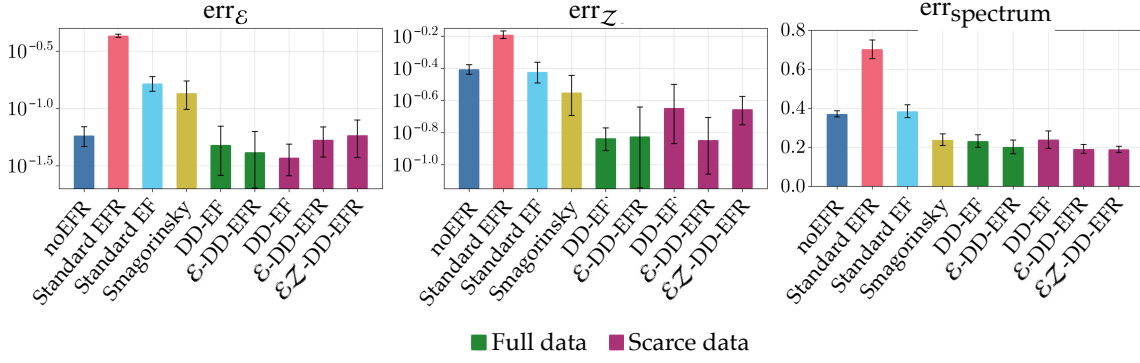


Figure 4.15: *Kolmogorov test case*. Relative error of the total kinetic energy, total enstrophy, and absolute error in logarithmic scale of the spectrum, of the different methodologies with respect to the filtered DNS. The errors are averaged in the time interval $[0, 10]$, and then evaluated as average values and confidence intervals (error bars) among $I_{\text{test}} = 5$ test configurations.

The main difference with respect to Figure 4.11 lies in the noEFR accuracy, which improves when considering N_T instead of $N_{t=3}$. As it no longer exhibits spurious oscillations and noisy behaviour at later times, it is qualitatively closer to the reference solution and has a smaller error than in Figure 4.11.

4.3.3 Efficiency of the methods

This section discusses the computational cost required by the proposed DD-EF(R) methods and compare their efficiency with that of state-of-the-art approaches⁵. The offline and online wall-clock times are reported in Table 4.3. We define the *offline* wall time as the computational cost associated with parameter optimization in state-of-the-art methods (standard EFR/EF and Smagorinsky), while, for DD-EF(R) approaches, the time required to compute the filter matrix $\hat{\mathcal{F}}^*$ ⁶. In particular, in DD-EF(R) the offline cost includes:

- ▶ the time required to advance the filtered DNS solution by one time step, in order to construct the snapshot matrix $\mathbf{W}^{\text{ref}}$;
- ▶ the time needed to solve the least-squares problem and determine the filter components as in Equation (4.10).

The *online* wall time refers to the cost of performing the simulation up to $T = 3$ s with fixed time step $\Delta t = 5 \times 10^{-4}$ s, ensuring a fair comparison across all methods. As shown in Table 4.3, the offline cost associated with parameter tuning in state-of-the-art methods can

⁵The simulations have been performed on an Apple M2 chip (8-core), 16GB unified memory.

⁶The cost of computing the filtered DNS is not included in the *offline* stage in the Table, but it is reported as *online* stage of the filtered DNS.

be significant, depending on the optimization strategy and the amount of data involved. In contrast, the cost of computing the data-driven filter matrix is relatively low, as it only requires solving a least-squares problem. Moreover, the same filter is used across **DD-EF**, **$\mathcal{E}$ -DD-EFR**, and **$\mathcal{E}\mathcal{Z}$ -DD-EFR**, and the offline cost depends solely on the amount of training data (i.e., T_{train} and I_{train}) used in its construction, as detailed in Table 4.3.

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.

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
	12.1 ($T_{\text{train}} = 3$, $I_{\text{train}} = 1$)	
$\mathcal{E}\mathcal{Z}$-DD-EFR	145.4 ($T_{\text{train}} = 3$, $I_{\text{train}} = 10$)	33.9

The online computational cost is significantly higher for standard **EFR/EF** methods, as it requires solving the differential filter equation at each time step. As expected, the **noEFR** configuration is the most efficient, since it does not involve any filtering operation or closure modeling. It is followed by **DD-EF** and the Smagorinsky model in terms of computational cost. The introduction of energy and enstrophy constraints in the data-driven approaches leads to a slight increase in computational time; however, the overall cost remains of the same order of magnitude as **DD-EF**. It is worth emphasizing that these constraints rely exclusively on quantities computed during the online simulation and do not require access to the reference filtered **DNS**. Consequently, the optimization of $\chi(t)$ in **$\mathcal{E}$ -DD-EFR** and **$\mathcal{E}\mathcal{Z}$ -DD-EFR** introduces negligible additional overhead, as it only involves evaluating the system's energy and enstrophy.

Figure 4.16 combines both an accuracy and an efficiency analysis of the methods. The plot reports the global energy and enstrophy errors (averaged across the I_{test} initializations) on the x - and y -axes, respectively. Each point represents the average error, while the size of the surrounding circle is proportional to the online simulation wall time. In this representation, the most accurate methods are located in the bottom-left region of the plot, whereas computationally cheaper methods are associated with smaller circles. Therefore, the optimal trade-off between accuracy and efficiency corresponds to points positioned in the bottom-left corner with minimal circle size. The results clearly show that all **DD-EF(R)** approaches—except for the scarce-data configuration with $T_{\text{train}} = 1$ s and $I_{\text{train}} = 1$ (pink circles)—achieve high accuracy in both energy and enstrophy while being approximately one order of magnitude faster than standard **EFR** and **EF**, consistently with the quantitative results reported in Table 4.3. Even in the most challenging scarce-data setting

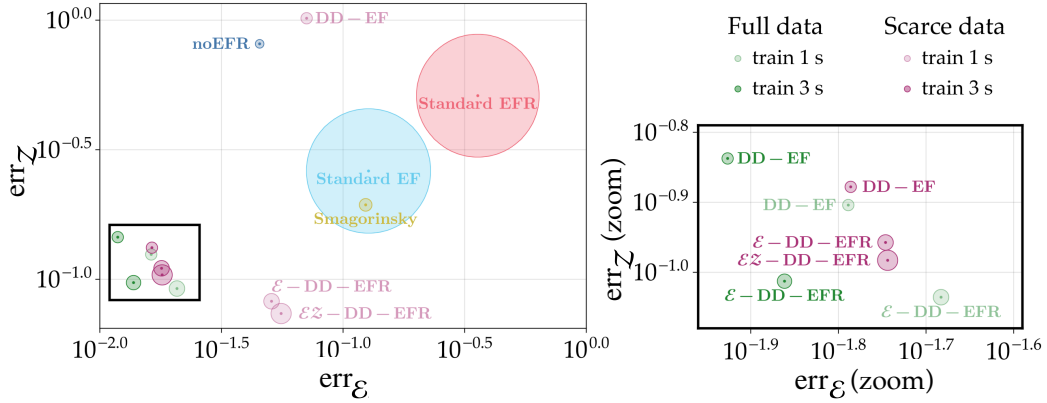


Figure 4.16: *Decaying HIT test case*. Average energy-entropy accuracy of the simulations, where the radius of the circles is proportional to the average wall time needed to perform online simulations.

($T_{\text{train}} = 1$, $I_{\text{train}} = 1$), where the data-driven methods exhibit reduced accuracy, the introduction of constraints in $\mathcal{E}$ -DD-EFR and $\mathcal{E}\mathcal{Z}$ -DD-EFR leads to a substantial improvement in the enstrophy prediction, with virtually no additional computational cost.

Overall, these results demonstrate that the proposed DD-EF(R) approaches provide a markedly superior balance between accuracy and computational efficiency compared to existing methods, consistently outperforming them across the considered regimes.

4.4 Conclusions and outlooks

In this chapter [150], we introduced a novel data-driven extension of the EFR framework for coarse-grid simulation of turbulent flows. First, we learned a **filter operator** via an efficient offline least-squares procedure based on filtered DNS data. The resulting operator is seamlessly integrated within a structure-preserving staggered finite-volume discretization, ensuring consistency with the underlying numerical scheme. Second, we proposed a new strategy for selecting the relaxation parameter $\chi(t)$ by enforcing **global dissipation of energy and/or enstrophy** during the online simulation. The resulting approaches, $\mathcal{E}$ -DD-EFR and $\mathcal{E}\mathcal{Z}$ -DD-EFR, enhance numerical stability while preserving key physical properties of the flow. In particular, $\mathcal{E}$ -DD-EFR enforces global energy dissipation at each time step, whereas $\mathcal{E}\mathcal{Z}$ -DD-EFR further incorporates enstrophy dissipation. These constraints are applied directly to the online simulation data, without requiring reference solutions, and significantly improve robustness, especially in scarce-data regimes. The proposed methodology was validated on two benchmark problems: decaying homogeneous isotropic turbulence and Kolmogorov flow. Notably, the same filter operator, trained exclusively on the first test case, was used in both scenarios. Despite this, the method achieved excellent performance in the second case as well, demonstrating a strong capability to extrapolate across different flow configurations. Compared to traditional

approaches, such as the Smagorinsky model and standard **EFR** strategies, the proposed data-driven methods yield substantial improvements in accuracy, particularly in terms of energy spectra, global kinetic energy, and enstrophy. In addition, the computational cost remains low, as the **DD-EF** approach eliminates the need to solve linear systems (required by the **DF**), while the adaptive evaluation of $\chi(t)$ relies solely on quantities readily available during the simulation. As a result, the method achieves a favourable balance between accuracy and efficiency, outperforming classical techniques in both aspects.

Future work will focus on extending the framework to **non-periodic boundary conditions, unstructured grids, and three-dimensional flows**. In particular, the extension to three-dimensional configurations is conceptually straightforward, as the formulation is dimension-independent, although it entails increased computational cost. For unstructured or non-uniform grids, the reliance on Fourier-based representations must be revisited. In this context, a natural extension is to express the filter in the eigenbasis of the discrete diffusion operator, which generalizes Fourier modes. To address the associated computational burden, model order reduction techniques can be employed. For instance, reduced spectral bases obtained via singular value decomposition or by selecting dominant eigenmodes can significantly decrease the dimensionality of the problem. Moreover, for non-periodic boundary conditions, strategies such as lifting functions—following the approach in [276]—can be adopted to enforce boundary constraints while preserving the structure of the method.

4.4.1 Conclusions to Part I and motivations to Part II

Part I of this thesis proposes two different **surrogate models**, both aimed at improving the performance of standard filtering approaches in the **EFR** framework. We explored both the use of **RL** to adaptively tune the filter width δ during the simulation (Chapter 3), and the use of data to build a different type of filter (Chapter 4). Structure-preserving strategies were explored in both cases, either by weak enforcement in **SP-DF-RL-EF**, or by adaptive tuning in **$\mathcal{E}$ -DD-EFR** and **$\mathcal{E}\mathcal{Z}$ -DD-EFR**.

Both methodologies are based on an **offline-online** decomposition. The expensive offline training stage consists of collecting some reference data (the **DNS** solution on a well-resolved grid) and of training the specific filter strategy proposed (the **RL** training in **RL-EF**, and the filter computation in **DD-EF**). In the online stage, the **EFR** simulations on a marginally-resolved grid are performed.

As discussed in Sections 3.3.3 and 4.3.3, there is a substantial reduction in computational cost from the offline to the online stage. Nevertheless, even when considering a relatively small number of degrees of freedom, the online simulations still require the repeated solution of discretized **PDEs**, which may take **minutes** or even **hours**, especially for long-time integrations or parametric studies.

This motivates the adoption of **model order reduction** strategies, aimed at drastically decreasing the computational burden by projecting the original model onto a low-dimensional subspace. In this framework, the high-fidelity solution is approximated as a combination of a reduced set of basis functions, typically extracted from representative snapshots of the system. By evolving the system in this reduced space, the original system of discretized **PDEs** is transformed into a much smaller and simpler system of ordinary differential equations (ODEs), which can be solved at a significantly lower computational cost while still capturing the dominant flow dynamics.

Part II will focus on enhancing the state-of-the-art projection-based model order reduction approaches by incorporating data-driven filtering or correction terms.

PART II

STABILIZATION APPROACHES FOR REDUCED ORDER METHODS

STABILIZED POD-BASED ROMs: LITERATURE REVIEW

While filtering techniques provide an effective strategy to stabilize marginally-resolved simulations (Part I), the resulting computational cost may still be large in many practical applications. Indeed, even when coarse grids are employed, the numerical solution of the **NSE** requires the resolution of complex nonlinear **PDEs** with a number of degrees of freedom that, in the numerical examples considered in Part I, typically ranges from $O(10^4)$ to $O(10^5)$.

Such computational demands become prohibitive in scenarios where multiple queries are required, such as design optimization, control, or uncertainty quantification. This motivates the introduction of **model order reduction** strategies, whose goal is to significantly decrease both the computational complexity and the dimensionality of the problem, while retaining the essential features of the underlying dynamics [214, 340, 258, 282, 297, 29, 128, 260, 263, 261, 30, 262, 28, 31, 186, 279, 281, 141, 46].

ROMs have been successfully employed in a wide range of applications where repeated and efficient simulations are required. In fluid dynamics, **ROMs** have been extensively used in aerodynamic analysis and design, enabling fast approximations of complex flow phenomena such as vortex shedding and turbulent wakes [234, 313, 33, 51, 353, 48]. In biomedical engineering, **ROMs** have been applied to cardiovascular flows [96, 308, 308, 307, 20, 101, 285, 49], allowing for efficient simulations of blood dynamics in patient-specific geometries, also modeling the fluid-structure interaction [70, 341]. **ROMs** have also been employed in environmental applications, such as ocean circulation [53], atmospheric dynamics [99, 222], pollutant transport and urban flows [323, 167, 217], and geophysical flows [54, 287, 229, 356], where full-order simulations would be computationally prohibitive. Additionally, **ROMs** play a key role in engineering design and optimization [216, 79, 140, 329, 330], where the governing equations must be solved multiple times for varying configurations. More generally, their efficiency makes them particularly suitable for many-query contexts and real-time applications, including control [268, 139, 320], monitoring, and digital twin technologies [15, 115, 165, 164, 124].

Depending on how the reduced model is constructed, **ROMs** can be broadly classified into two categories. **Intrusive ROMs** rely on the explicit knowledge of the governing equations and typically involve a projection of the full-order model onto a reduced basis, as in the case of Galerkin approaches. **Nonintrusive** methods, on the other hand, are purely data-driven and aim at learning the reduced dynamics directly from data, without

requiring access to the underlying equations.

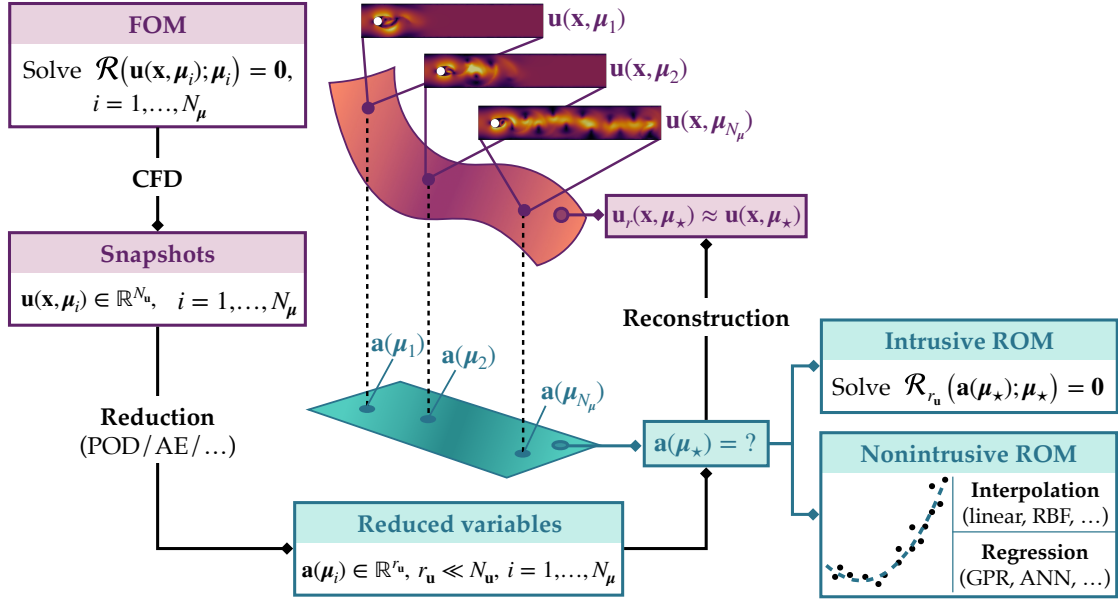


Figure 5.1: Schematic illustration of the general ROM pipeline.

As for the surrogate models presented in Chapters 3 and 4, most ROMs are based on an **offline–online** computational paradigm.

The *offline* stage consists of the construction of a reduced basis and the precomputation of all the quantities required for the efficient evaluation of the reduced model. This phase is typically computationally expensive, as it involves the setup and resolution of the FOM and the generation of high-fidelity solutions (typically named *snapshots*) $\mathbf{u}(\mathbf{x}, \mu_i) \in \mathbb{R}^{N_u}$, $i = 1, \dots, N_\mu$, where μ identifies the set of (physical or geometrical) parameters considered, N_μ is the number of snapshots, and $\mathbf{x}$ indicates the physical coordinates where the snapshots are defined. In real-world applications, the solutions are typically found by solving a system of PDEs, written in residual and compact form:

$$\mathcal{R}(\mathbf{u}(\mathbf{x}, \mu_i); \mu_i) = \mathbf{0}, \quad i = 1, \dots, N_\mu.$$

The snapshots' collection is then followed by the extraction of the dominant modes, for instance, through POD. The reduced representations corresponding to the snapshots can then be written as $\mathbf{a}(\mu_i) \in \mathbb{R}^{r_u}$, where $r_u \ll N_u$ is the reduced dimensionality. The *online* stage, in contrast, is devoted to the rapid evaluation of the ROM for unseen parameter values (i.e., μ_*). This can be achieved either by solving a low-dimensional dynamical system (in *intrusive* ROMs), or by directly approximating the reduced solution via a pre-trained interpolation or regression map $\mu \rightarrow \mathbf{a}$ (in *nonintrusive* ROMs). In the case of projection-based intrusive ROMs, like the **Galerkin-Reduced Order Method (G-ROM)** [179, 208], the

full system of equations is projected onto the reduced basis, obtaining a reduced dynamical system of the form¹:

$$\mathcal{R}_{r_u}(\mathbf{a}(\boldsymbol{\mu}_\star); \boldsymbol{\mu}_\star) = \mathbf{0}.$$

In both the equation-based and purely data-driven methods, the computational cost is drastically reduced with respect to the FOM, enabling efficient many-query predictions.

An illustration summarizing the pipeline of classical ROMs is represented in Figure 5.1.

One of the core questions in ROMs is:

How many snapshots N_μ are needed to achieve a desired accuracy?

The required number of parameter instances N_μ strongly depends on the problem, the parameter-space dimension, and the adopted sampling strategy. In practical applications, accurately capturing the solution variability may require many snapshots, making the offline stage computationally expensive. To reduce this cost, efficient sampling techniques have been developed. In the reduced basis framework, greedy algorithms guided by residual-based a posteriori error estimators [128, 340, 263] iteratively select the most informative parameter samples.

Another important question is:

How is the reduced dimension r_u selected?

For POD-based ROMs, the reduced dimension is determined by the decay of the singular values of the snapshot matrix. The reduced basis is built from the leading modes that capture most of the system energy. Typically, the dimension r_u is chosen so that a prescribed energy threshold is retained, or according to a target approximation accuracy based on projection errors or error indicators. Hence, selecting r_u represents a trade-off between computational efficiency and approximation quality.

It is worth noting, however, that for convection-dominated problems the decay of the singular values may be slow, which may require a relatively large number of modes to achieve an accurate approximation. Part II of the thesis aims at improving the ROM approximation in marginally-resolved regimes, i.e. when the number of modes r_u is enough to capture most of the system's energy, but standard G-ROMs yield inaccurate predictions. As at the full order level (Part I), in this regime intrusive ROMs are often characterized by numerical instabilities and loss of accuracy, which motivates the introduction of stabilization and closure strategies at the reduced level.

Before presenting the novelties of the thesis in the field of model order reduction, an overview of the baseline methods is presented in the following sections:

- Section 5.1 briefly recalls the construction of standard POD-Galerkin ROMs, which we simply indicate as G-ROM;

¹For clarity, we adopt here a compact form for the residual of the reduced system, namely $\mathcal{R}_{r_u}$ only depending on the reduced parametric variable $\mathbf{a}(\boldsymbol{\mu})$. In Section 5.1, this form will be extended to a G-ROM system derived from the discretized NSE and we will also include the reduced pressure equation, namely $\mathcal{R}_{r_p} = \mathbf{0}$.

- Section 5.3 extends the concept of filtering, described in Section 1.2 to **ROMs**;
- Section 5.4 presents different types of closure strategies existing in literature to improve the standard **ROM** formulation. We distinguish here between the turbulence modeling (Section 5.4.1) and the modeling of the interaction among the **POD** resolved and unresolved scales (Section 5.4.2).

5.1 POD-Galerkin ROMs (G-ROMs)

The **POD** [311, 212] is a widely used technique for extracting the low-dimensional structures from high-dimensional data, aiming at identifying a set of orthonormal modes that optimally represent the dominant features of a given field, in terms of energy content.

After the high-fidelity solutions, named *snapshots* are collected into the snapshots' matrix, the **POD** procedure is applied separately to each field of interest. In the context of **CFD**, this typically includes the velocity and pressure fields, and, in the case of **RANS** modeling, as described in Section 5.4.1, additional quantities, such as the eddy viscosity.

For brevity, we indicate the i -th velocity and pressure snapshots as $\mathbf{u}_i = \mathbf{u}(\mathbf{x}, \boldsymbol{\mu}_i)$, and $p_i = p(\mathbf{x}, \boldsymbol{\mu}_i)$, respectively.

We start by considering the velocity and pressure snapshots' matrices:

$$\mathbf{S}_u = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_{N_\mu} \\ | & | & & | \end{bmatrix}, \quad \mathbf{S}_p = \begin{bmatrix} | & | & & | \\ p_1 & p_2 & \dots & p_{N_\mu} \\ | & | & & | \end{bmatrix}, \quad (5.1)$$

where $\mathbf{u}_i \in \mathbb{R}^{N_u}$ and $p_i \in \mathbb{R}^{N_p} \forall i = 1, \dots, N_\mu$. Hence, $\mathbf{S}_u \in \mathbb{R}^{N_u \times N_\mu}$ and $\mathbf{S}_p \in \mathbb{R}^{N_p \times N_\mu}$. We also call P the dimensionality of each $\boldsymbol{\mu}_i$, i.e., the number of parameters considered. In order to keep the offline cost manageable, in the numerical experiments of the next chapters, we will consider a limited number of snapshots, namely $N_\mu \ll N_u$, and in particular $N_\mu \sim \mathcal{O}(10^2) - \mathcal{O}(10^3)$.

The procedure consists in the research for the velocity and pressure **POD** spaces, by solving two optimization problems:

$$\begin{cases} \mathbb{V}_u^r = \arg \min_{i=1, \dots, N_\mu} \frac{1}{N_\mu} \sum_{n=1}^{N_\mu} \|\mathbf{u}_n - \sum_{i=1}^{r_u} (\mathbf{u}_n, \boldsymbol{\phi}_i)_2 \boldsymbol{\phi}_i\|_2^2 & \forall r_u = 1, \dots, N_\mu, \\ \mathbb{V}_p^r = \arg \min_{i=1, \dots, N_\mu} \frac{1}{N_\mu} \sum_{n=1}^{N_\mu} \|p_n - \sum_{i=1}^{r_p} (p_n, \chi_i)_2 \chi_i\|_2^2 & \forall r_p = 1, \dots, N_\mu. \end{cases} \quad (5.2)$$

The **POD** spaces, solutions of (5.2), are:

$$\mathbb{V}_u^r = \text{span}(\{\boldsymbol{\phi}_i\}_{i=1}^{r_u}), \text{ and } \mathbb{V}_p^r = \text{span}(\{\chi_i\}_{i=1}^{r_p}),$$

with $r_u \ll N_u$, $r_p \ll N_p$, and where $\boldsymbol{\phi}_i \in \mathbb{R}^{N_u}$ for $i = 1, \dots, r_u$ and $\chi_i \in \mathbb{R}^{N_p}$ for $i = 1, \dots, r_p$. The modes, $\{\boldsymbol{\phi}_i\}_{i=1}^{r_u}$ for velocity and $\{\chi_i\}_{i=1}^{r_p}$ for pressure, can be equivalently found by solving an eigenvalue problem [179]. In this case, the following correlation matrices are computed:

$$(\mathbf{R}_u)_{ij} = (\mathbf{u}_i, \mathbf{u}_j)_2, \quad (\mathbf{R}_p)_{ij} = (p_i, p_j)_2, \text{ where } \mathbf{R}_u \in \mathbb{R}^{N_\mu \times N_\mu} \text{ and } \mathbf{R}_p \in \mathbb{R}^{N_\mu \times N_\mu}.$$

Each mode can be computed as follows:

$$\phi_i = \frac{1}{N_\mu(\Lambda_u)_{ii}} \sum_{j=1}^{N_\mu} \mathbf{u}_j(\mathbf{V}_u)_{ij}, \quad \chi_i = \frac{1}{N_\mu(\Lambda_p)_{ii}} \sum_{j=1}^{N_\mu} p_j(\mathbf{V}_p)_{ij},$$

where $\mathbf{V}_u$ and $\mathbf{V}_p$ are the matrices whose columns are the eigenvectors of the correlation matrices, whereas Λ_u and Λ_p are diagonal matrices whose entries are the eigenvalues of the correlation matrices.

Once the velocity and pressure modes are obtained, the solution can be projected onto the spaces $\mathbb{V}_u^r$ and $\mathbb{V}_p^r$, obtaining the projections:

$$\mathbf{u}^{\text{proj}}(\mathbf{x}, \mu) = \sum_{i=1}^{r_u} a_i^{\text{proj}}(\mu) \phi_i(\mathbf{x}), \quad p^{\text{proj}}(\mathbf{x}, \mu) = \sum_{i=1}^{r_p} b_i^{\text{proj}}(\mu) \chi_i(\mathbf{x}), \quad (5.3)$$

with $a_i^{\text{proj}}(\mu) = (\mathbf{u}(\mathbf{x}, \mu), \phi_i(\mathbf{x}))_{L^2(\Omega)}$, $\forall i = 1, \dots, r_u$, and $b_i^{\text{proj}}(\mu) = (p(\mathbf{x}, \mu), \chi_i(\mathbf{x}))_{L^2(\Omega)}$, $\forall i = 1, \dots, r_p$. For any reduced dimension r_u and r_p , the projection is the *optimal* solution, namely the one that minimizes the error in the particular norm considered (in our case, the L^2 norm). The error of the projection with respect to the **FOM** solution will be referred to as the *reconstruction error*.

Once the **POD** spaces are constructed, the velocity and pressure field are approximated as a convex combination of the modes:

$$\mathbf{u}(\mathbf{x}, \mu) \approx \mathbf{u}_r(\mathbf{x}, \mu) = \sum_{i=1}^{r_u} a_i(\mu) \phi_i(\mathbf{x}), \quad p(\mathbf{x}, \mu) \approx p_r(\mathbf{x}, \mu) = \sum_{i=1}^{r_p} b_i(\mu) \chi_i(\mathbf{x}). \quad (5.4)$$

Classical Galerkin methods consist of projecting the discretized incompressible **NSE** into the reduced space, which, in the case of **POD**, is spanned by the **POD** modes. In particular, the momentum equation is projected onto each velocity mode ϕ_i , while the continuity equation is projected onto each pressure mode χ_i . After the projection is performed, a dynamical system with $r_u + r_p$ unknown reduced variables is obtained. The unknowns are the velocity and pressure vectors of coefficients, which can be written as

$$\mathbf{a}(\mu) = [a_i(\mu)]_{i=1}^{r_u} \text{ and } \mathbf{b}(\mu) = [b_i(\mu)]_{i=1}^{r_p}.$$

To write the **G-ROM**, we first recall the discretized incompressible **NSE** in the absence of external forces:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot \nu (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) + \nabla p = \mathbf{0}, \\ \nabla \cdot \mathbf{u} = 0. \end{cases} \quad (5.5)$$

Remark: conservative form of the discretized **NSE**.

Unlike Equations (1.19) and (4.1), the convective and viscous terms are written in *conservative form*, namely $(\nabla \cdot (\mathbf{u} \otimes \mathbf{u}))$ and $(\nabla \cdot (\nabla \mathbf{u} + (\nabla \mathbf{u})^T))$. Although equivalent

to the advective and Laplacian forms in the continuous incompressible setting, the conservative formulation is generally preferred in numerical discretizations since it directly derives from integral conservation laws and preserves local conservation properties, especially in finite volume methods.

If we project the momentum equation into each velocity mode and the continuity equation into each pressure mode, we obtain:

$$\begin{cases} \left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot \nu (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) + \nabla p, \phi_i \right)_{L^2(\Omega)} = 0 & \forall i = 1, \dots, r_u, \\ (\nabla \cdot \mathbf{u}, \chi_j)_{L^2(\Omega)} = 0 & \forall j = 1, \dots, r_p. \end{cases} \quad (5.6)$$

In matrix form and considering the decomposition in Equation (5.4), we obtain the reduced dynamical system:

$$\begin{cases} -\mathbf{M}_{\mathbf{u}_r} \dot{\mathbf{a}} - \mathbf{a}^T \mathbf{C}_{\mathbf{u}_r} \mathbf{a} + \nu (\mathbf{B}_r + \mathbf{B}_r') \mathbf{a} - \mathbf{G}_r \mathbf{b} = \mathbf{0}, \\ \mathbf{H}_r \mathbf{a} = \mathbf{0}, \end{cases} \quad (5.7)$$

where the reduced operators $\mathbf{M}_{\mathbf{u}_r} \in \mathbb{R}^{r_u \times r_u}$, $\mathbf{B}_r \in \mathbb{R}^{r_u \times r_u}$, $\mathbf{B}_r' \in \mathbb{R}^{r_u \times r_u}$, $\mathbf{C}_{\mathbf{u}_r} \in \mathbb{R}^{r_u \times r_u \times r_u}$, $\mathbf{G}_r \in \mathbb{R}^{r_u \times r_p}$, $\mathbf{H}_r \in \mathbb{R}^{r_p \times r_u}$ are defined as follows:

$$\begin{aligned} (\mathbf{B}_r)_{ij} &= (\phi_i, \nabla \cdot \nabla \phi_j)_{L^2(\Omega)}, & (\mathbf{B}_r')_{ij} &= (\phi_i, \nabla \cdot (\nabla \phi_j^T))_{L^2(\Omega)}, & (\mathbf{C}_{\mathbf{u}_r})_{ijk} &= (\phi_i, \nabla \cdot (\phi_j \otimes \phi_k))_{L^2(\Omega)}, \\ (\mathbf{G}_r)_{ij} &= (\phi_i, \nabla \chi_j)_{L^2(\Omega)}, & (\mathbf{H}_r)_{ij} &= (\chi_i, \nabla \cdot \phi_j)_{L^2(\Omega)}. \end{aligned} \quad (5.8)$$

The term $\dot{\mathbf{a}}$ indicates the discrete time derivative, which can be expressed in the preferred time integration scheme. For instance, if we adopt a first-order forward Euler scheme, at the time step t_{n+1} we have $\dot{\mathbf{a}}_{n+1} = \frac{\mathbf{a}_{n+1} - \mathbf{a}_n}{t_{n+1} - t_n}$. Note also that $\mathbf{C}_{\mathbf{u}_r}$ is a third-order tensor, which approximates the nonlinear convective operator. Using a third-order tensor may increase the computational cost when the number of degrees of freedom is large. Another way to treat this term is based on *hyper-reduction* techniques. The Empirical Interpolation Method (EIM) [24] and its discrete variant (DEIM) [355] approximate nonlinear operators by projecting them onto a reduced basis constructed from sampled evaluations, thus avoiding computations in the full-order space. Similarly, Gappy-POD [57] reconstructs the nonlinear terms from a limited number of spatial samples, further reducing the cost of their evaluation during the online stage.

Once the reduced dimensionalities r_u and r_p are selected and the reduced operators in (5.8) are assembled, the reduced dynamical system (5.7) can be solved in predictive regimes, namely for unseen parameters $\mu_\star$, which may include time extrapolation and different Reynolds numbers.

However, the reduced model of system (5.7) presents stability issues because the reduced version of the *inf-sup* condition is not satisfied. The instability leads to a lack of accuracy in the pressure approximation and to the need for a tailored velocity-pressure

coupling at the **ROM** level. This motivates the following part, focused on stabilization techniques for **G-ROMs**.

Remark: On other projection-based approaches.

The **Galerkin** approach [104, 14] relies on an *orthogonal* projection, where the residual is enforced to be orthogonal to the same reduced space used to approximate the solution, typically spanned by the **POD** modes obtained from the snapshot matrices. Alternative projection strategies can also be employed. In the **Petrov–Galerkin** framework [252, 360, 63], the trial and test spaces differ, leading to an oblique projection that may improve stability properties. A relevant example is the **Least-Squares Petrov–Galerkin** (LSPG) method [56, 55, 64], which computes the reduced solution by minimizing the residual of the discrete governing equations. More generally, **minimum-residual ROMs** [57] formulate the reduced problem as the minimization of a suitable residual norm. In this thesis, we focus on the **POD–Galerkin** framework. In particular, the reduced bases are assumed to be constructed through **POD**, although alternative approaches, such as greedy algorithms [258, 280, 128] could also be adopted.

5.1.1 Treatment of non-homogeneous Dirichlet boundary conditions

In the **ROM** framework, the treatment of Dirichlet boundary conditions strongly depends on whether they are homogeneous or non-homogeneous. Homogeneous Dirichlet conditions are usually inherited automatically by the reduced space: if the full-order snapshots satisfy the prescribed zero boundary value, then the reduced basis functions, being linear combinations of those snapshots, satisfy the same constraint as well. This is no longer true for non-homogeneous Dirichlet conditions, since a nonzero boundary datum cannot, in general, be represented by linear combinations of basis functions satisfying homogeneous constraints. Hence, the question we try to answer in this section is:

What are the techniques for an explicit treatment of the non-homogeneous Dirichlet boundary conditions?

A common approach is to introduce a **lifting function** [117, 116, 120], so that the solution is decomposed into a known component that matches the prescribed boundary data and an unknown homogeneous component approximated by the reduced basis, namely:

$$\mathbf{u}(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\phi}_0(\mathbf{x}) + \sum_{i=1}^{r_u} a_i(\boldsymbol{\mu}) \boldsymbol{\phi}_i(\mathbf{x}). \quad (5.9)$$

In this way, the reduced basis is constructed only for the homogeneous component ($\mathbf{u} - \boldsymbol{\phi}_0$), so that the resulting reduced modes still belong to the admissible space with zero boundary values. The lifting function $\boldsymbol{\phi}_0$ may be constructed in different ways [269], such as:

- the averaged velocity field over the parametrized snapshot interval;

- ▶ a reference snapshot, like the initial condition in time-dependent problems $\mathbf{u}_0(\mathbf{x})$;
- ▶ an ad-hoc lifting function $\mathbf{u}_g(\mathbf{x})$, for example, by solving an auxiliary full-order problem or by using an analytical function;
- ▶ a combination of snapshots.

Regardless of how ϕ_0 is chosen, it must satisfy $\phi_0 = \mathbf{u}|_{\Gamma_{Dk}}$ for each non-homogeneous boundary Γ_{Dk} . However, its construction is problem-dependent and may affect the accuracy and stability of the reduced model, especially when the boundary data are time-dependent or parameter-dependent.

Alternatively, the boundary condition can be imposed weakly through a **penalty method** [17, 25, 310, 36], by adding to the reduced momentum equation a term that penalizes the mismatch between the **ROM** solution and the prescribed boundary value. We can define this extra-term as:

$$\tau_{BC} \sum_{k=1}^{N_{BC}} (u_{BCk} \Phi_{BCk} - \mathbf{E}_{rk} \mathbf{a}),$$

where N_{BC} is the number of non-homogeneous boundaries, u_{BCk} is the non-homogeneous value of the velocity at boundary Γ_{Dk} , and the remaining matrices are defined as:

$$(\Phi_{BCk})_i = (\phi_i)_{L^2(\Gamma_{Dk})}, \text{ and } (\mathbf{E}_{rk})_{ij} = (\phi_i, \phi_j)_{L^2(\Gamma_{Dk})}. \quad (5.10)$$

Finally, τ_{BC} is a penalization factor whose value can be set by sensitivity analysis or by automatic tuning as presented in [315]. The penalty method has also been adopted in different works like [160, 159, 208, 313, 152, 143, 142].

The lifting approach enforces the boundary condition exactly, whereas the penalty method is often easier to implement and more flexible, but requires a careful choice of the penalty parameter to balance accuracy, conditioning, and stability.

5.2 Pressure stabilization techniques in ROMs

In this part, we mainly focus on two different stabilization techniques aimed at improving the pressure approximation at the **ROM** level.

5.2.1 Supremizer enrichment

The first one is the so-called **supremizer** approach, introduced in [19]. This strategy allows the fulfilment of the inf-sup condition at the reduced level, and consists in enriching the reduced velocity space with r_{sup} additional modes, named *supremizer modes* $[\phi_i^{\text{sup}}]_{i=1}^{r_{\text{sup}}}$, found by applying the **POD** to the supremizer solutions $\mathbf{s}(\mathbf{x}, \mu)$. The supremizer snapshots can be obtained either directly from the pressure modes (*exact* supremizer) or from the pressure snapshots (*approximated* supremizer). In the *exact* enrichment, each supremizer

snapshot is found from each pressure mode χ_i ($s_i = s(\chi_i)$) by solving the system

$$\begin{cases} \Delta s_i = -\nabla \chi_i & \text{in } \Omega, \\ s_i = 0 & \text{on } \partial\Omega. \end{cases} \quad (5.11)$$

On the other hand, in the *approximated* technique, the supremizer solutions are found from the pressure snapshots, basically replacing χ_i with p_i in (5.11), which leads to $s_i = s(p_i)$. In both cases, after the set of supremizer solutions is collected, a standard **POD** procedure is applied to obtain the reduced supremizer space $\mathbb{V}_{\mathbf{u}^{\text{sup}}}^r = \text{span}(\{\phi_i^{\text{sup}}\}_{i=1}^{r_{\text{sup}}})$. The exact approach is more robust in terms of inf-sup condition fulfilment, but it is expensive as it requires the resolution of the full-dimensionality system (5.11) in the on-line stage. On the other hand, in the approximated approach the system can be solved offline, and it provides a good approximation of the supremizer snapshots at a lower online cost. The supremizer stabilization, hence, consists in solving the reduced system (5.7), with additional r_{sup} unknowns, added to the velocity vector of coefficients: $\mathbf{a} = (a_1, \dots, a_{r_{\mathbf{u}}}, a_1^{\text{sup}}, \dots, a_{r_{\text{sup}}}^{\text{sup}})$. More details on the supremizer enrichment can be found in [282, 110, 19].

5.2.2 Pressure Poisson Equation in ROMs

A different type of stabilization is based on the **Pressure Poisson Equation (PPE)** formulation. The **PPE** approach was introduced at the full-order level in [157], where it is used to aid stabilization by adding a dedicated evolution equation for the pressure field. In the full-dimensional problem, the **PPE** is derived by taking the divergence of the momentum equation and applying the divergence-free constraint to eliminate the time derivative $\nabla \cdot \frac{\partial \mathbf{u}}{\partial t} = \frac{\partial(\nabla \cdot \mathbf{u})}{\partial t}$. As the solenoidal constraint is directly embedded into this equation, the **PPE**, namely:

$$\Delta p = -\nabla \cdot (\nabla \cdot (\mathbf{u} \otimes \mathbf{u})) \text{ in } \Omega, \quad (5.12)$$

can be used to replace the continuity equation in the discretized **NSE**. This approach can be extended at the **ROM** level by projecting the **PPE** into each pressure mode χ_i :

$$(\nabla p + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}), \nabla \chi_i)_{L^2(\Omega)} - \nu (\nabla \times \mathbf{u}, \mathbf{n} \times \nabla \chi_i)_{L^2(\Gamma)} - \left(\mathbf{n} \cdot \frac{\partial \mathbf{u}}{\partial t}, \chi_i \right)_{L^2(\Gamma)} = 0 \quad \forall i = 1, \dots, r_p. \quad (5.13)$$

In the above equation, the notation $(\cdot, \cdot)_{L^2(\Gamma)}$ indicates the L^2 dot product on the boundaries $\Gamma = \partial\Omega$, and $\mathbf{n}$ indicates the normal unitary vector to the domain boundary. Note that Equation (5.13) includes the contributions on the boundaries due to the viscous effect and due to the time variation of the velocity boundary conditions [157]. If all the velocity boundary conditions are not time dependent, the term $\left(\mathbf{n} \cdot \frac{\partial \mathbf{u}}{\partial t}, \chi_i \right)_{L^2(\Gamma)}$ vanishes². In all the

²A more detailed derivation of the boundary conditions in the **PPE** in case of turbulence modeling is provided in the Appendix of [145].

test cases we will consider in the next chapters, the velocity is fixed in time at the boundaries. Hence, from now on we consider $\left(\mathbf{n} \cdot \frac{\partial \mathbf{u}}{\partial t}, \chi_i\right)_{L^2(\Gamma)} = 0$. Based on the formulation in (5.13), we can finally write the so-called **PPE-ROM** system in matrix form as follows:

$$\begin{cases} -\mathbf{M}_{\mathbf{u}_r} \dot{\mathbf{a}} - \mathbf{a}^T \mathbf{C}_{\mathbf{u}_r} \mathbf{a} + \nu(\mathbf{B}_r + \mathbf{B}_r') \mathbf{a} - \mathbf{G}_r \mathbf{b} = \mathbf{0}, \\ \mathbf{D}_{\mathbf{p}_r} \mathbf{b} + \mathbf{a}^T \mathbf{C}_{\mathbf{p}_r} \mathbf{a} - \nu \mathbf{N}_r \mathbf{a} = \mathbf{0}, \end{cases} \quad (5.14)$$

where the following reduced operators are used:

$$(\mathbf{D}_{\mathbf{p}_r})_{ij} = (\nabla \chi_i, \nabla \chi_j)_{L^2(\Omega)}, (\mathbf{C}_{\mathbf{p}_r})_{ijk} = (\nabla \chi_i, \nabla \cdot (\phi_j \otimes \phi_k))_{L^2(\Omega)}, (\mathbf{N}_r)_{ij} = (\mathbf{n} \cdot \nabla \chi_i, \nabla \times \phi_j)_{L^2(\Gamma)}. \quad (5.15)$$

The operators have dimensionalities: $\mathbf{D}_{\mathbf{p}_r} \in \mathbb{R}^{r_p \times r_p}$, $\mathbf{C}_{\mathbf{p}_r} \in \mathbb{R}^{r_p \times r_u \times r_u}$, $\mathbf{N}_r \in \mathbb{R}^{r_p \times r_u}$.

Unlike the supremizer enrichment approach, in **PPE-ROMs** the number of reduced unknowns remains unchanged with respect to the non-stabilized system.

5.2.3 Is pressure stabilization enough?

Other approaches to improve the pressure approximation include **residual-based stabilization** techniques [8, 7], which enrich the reduced system with consistent terms depending on momentum or continuity residuals to improve the velocity–pressure coupling, without necessarily enriching the velocity space. Classical examples are **PSPG** [133, 97], **SUPG** [45], and **GLS-ROMs**. A related class is **Local Projection Stabilization (LPS)** [211, 60], which damps the unresolved fluctuations via projection-based operators. Alternative strategies include **least-squares pressure recovery** [16], where the pressure is reconstructed a posteriori from the momentum residual, avoiding a reduced saddle-point system, and **discretize-then-project** approaches [316], which preserve the discrete velocity–pressure coupling by projecting the full-order algebraic system rather than the continuous equations. Finally, **penalty** [32] and **artificial compressibility** [66, 78] methods relax incompressibility through divergence penalization or modified pressure evolution, improving robustness at the cost of additional parameters.

At this point, we address the question:

*Is pressure stabilization enough to stabilize a **G-ROM**?*

The answer is that, even in the presence of stabilization techniques, **G-ROMs** may remain inaccurate or unstable for convection-dominated flows. In this case, modal truncation and Galerkin projection can alter the energy transfer among resolved and unresolved scales, leading to overly energetic or unstable reduced trajectories. In particular, we focus on *under-resolved* and *marginally-resolved* regimes. In analogy with high-fidelity simulations, these regimes refer to the case where the energy retained by the **POD** modes is too low (*under-resolved*) or is enough to capture the relevant dynamics, but **G-ROMs** yield inaccurate results (*marginally-resolved*). Table 5.1 summarizes and compares the meaning of different regimes in both full-order and reduced-order setups.

As confirmed by Table 5.1, **ROMs** may exhibit loss of accuracy in under-resolved or marginally-resolved regimes. While the energetic optimality of the **POD** basis ensures an

Table 5.1: Comparison of resolution regimes in high-fidelity simulations (FOM) and POD-based ROMs.

Regime		Meaning	Regularization
Fully-resolved	FOM	All scales/modes are resolved.	Not needed.
	ROM		May be needed.
Well-resolved	FOM	Most scales/modes are resolved.	Rarely needed.
	ROM		May be needed.
Marginally-resolved	FOM	Most energetic scales/modes are resolved, but there are often numerical issues.	Needed.
	ROM		
Under-resolved	FOM	Important scales/modes are unresolved.	Needed.
	ROM		

efficient representation of the snapshot data, it does not guarantee dynamical accuracy: the discarded modes may still play a crucial role in nonlinear interactions, energy transfer mechanisms, and dissipation processes.

As a result, standard **G-ROMs** may exhibit energy drift, spurious oscillations, or poor long-time predictive performance when the reduced space is insufficient to capture the relevant dynamics. A numerical example of this phenomenon is provided in Figure 5.2, which displays the results obtained using a supremizer (SUP-ROM) and a **PPE** (PPE-ROM) approach in the flow past a cylinder test case at $Re = 50\,000$. This is a benchmark in fluid dynamics and the setup used is the same of [143]. In this case, the number of modes is $r_u = r_{\text{sup}} = r_p = 5$, corresponding to a retained energy greater than 95% for all fields. Figure 5.2 highlights that the velocity and pressure approximations are inaccurate and unphysical, especially in the region nearby the cylinder.

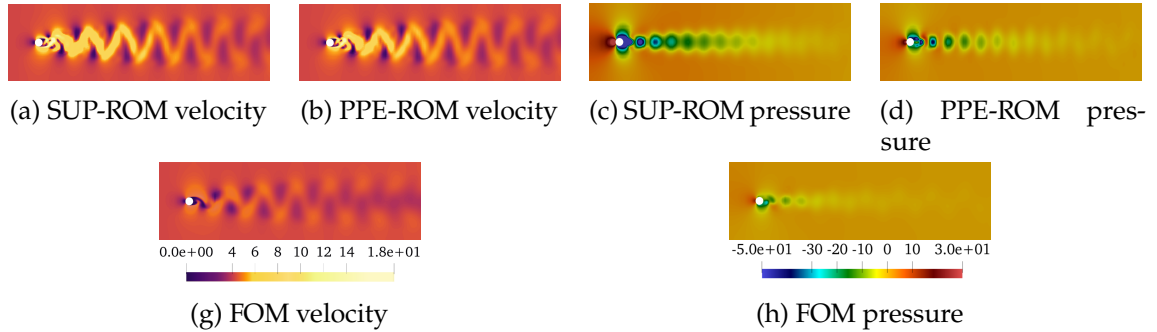


Figure 5.2: *Flow past a cylinder test case.* SUP-ROM, PPE-ROM, and FOM velocity magnitude and pressure fields. The FOM and ROM settings are described in [143]. The fields are at final ROM time and at fixed $Re = 50\,000$.

It is worth noting that stability issues may also arise even in well-resolved settings, due to the lack of intrinsic dissipativity of the Galerkin projection and to numerical or discretization effects.

In the following, we focus on improving accuracy and stability in under-resolved or marginally-resolved regimes by introducing regularization mechanisms, either through filtering strategies (Section 5.3) or dedicated closure techniques (Section 5.4).

The starting point of the next sections is a stabilized ROM for the incompressible NSE in the generic compact form:

$$\begin{cases} \mathcal{R}_{r_u}(\mathbf{a}, \mathbf{b}; \mu) = \mathbf{0}, \\ \mathcal{R}_{r_p}(\mathbf{a}, \mathbf{b}; \mu) = \mathbf{0}, \end{cases} \quad (5.16)$$

which has $r_u + r_p$ reduced unknown variables.

5.3 Filtering approaches for ROMs

The spatial filters introduced in Section 1.2 can be substantially applied in the ROM context into the reduced space. In convection-dominated or turbulent regimes, ROM filtering aims at stabilizing the reduced dynamics by smoothing selected reduced quantities or by regularizing the nonlinear convective interactions.

Remark: ROM spatial filters vs ROM closures. It is important to highlight that the filtered ROMs we describe in this Section have also been named LES-inspired ROMs or LES-ROMs [259]. The reason is that the *spatial filters* are mainly inherited from LES principles, both mathematically and physically.

We emphasize that in [6], the authors consider spatial filters as ROM structural closures, namely closures derived from mathematical arguments. Indeed, as done at the full-order level in Chapter 1, also at the reduced-order level, ROM spatial filters like the DF can be reformulated as closure contributions.

However, it is important to highlight that in [259], LES-ROMs are considered in principle disconnected from ROM closures. Since mainly based on existing physical models, LES-ROMs can in principle operate in different flow problems without retraining. On the other hand, ROM closures typically need to be retrained to new data or settings.

Moreover, ROM spatial filters aim to smooth out different terms in the governing equations, with the goal of alleviating spurious oscillations and enhancing numerical stability. On the other hand, ROM closures typically aim to reintroduce the contribution of the unresolved modes into the reduced system, as will be described in Section 5.4. However, since the primary role of the unresolved small scales is to dissipate energy in the reduced system, ROM closures and filters, even if conceptually different, often overlap [6, 259].

For explicit filters, we can extend the definition of (1.7) as a discrete filter applied to the reduced velocity:

$$\bar{\mathbf{a}} = \mathcal{F}_r(\mathbf{a}), \quad (5.17)$$

where δ is the filter width, just as in the full-order setting.

A first important filter is the **modal filter** or **low-pass filter**, which consists of applying a damping to the **POD** coefficients, namely:

$$\bar{a}_i = \sigma_i a_i, \quad \sigma_i \in [0, 1], \quad \forall i = 1, \dots, r_u.$$

This serves as an energetic stabilizer for the **ROM**, and the weights σ_i can have different shape, like listed in the following.

- ▶ **Sharp cutoff:** $\sigma_i = 1$ if $i \leq r_{\text{cutoff}}$, $\sigma_i = 0$ if $i > r_{\text{cutoff}}$ for a pre-selected reduced dimension r_{cutoff} ;
- ▶ **Exponential damping:** $\sigma_i = e^{-\alpha(i/r_u)^p}$ with pre-selected parameters α and p ;
- ▶ **Algebraic decay:** $\sigma_i = \frac{1}{1+\alpha i^p}$ with pre-selected α and p .

Note that the filtering weights are typically chosen to decrease with the mode index, thereby assigning stronger damping (i.e., smaller coefficients) to higher-order modes. The reason is that higher-order modes are generally less energetic and more affected by truncation-induced inaccuracies. As a consequence, they are more prone to amplify numerical artifacts and spurious oscillations in reduced-order simulations.

Another **ROM** spatial filter is the **projection filter**. Given two fixed dimensions r_u and $r'_u < r$, and the approximation $\mathbf{u}_r \in \text{span}(\phi_1, \dots, \phi_{r_u})$, the projection filter consists of constructing the approximation $\mathbf{u}_{r'} \in \text{span}(\phi_1, \dots, \phi_{r'_u})$ such that: $(\mathbf{u}_{r'}, \phi_i) = (\mathbf{u}_r, \phi_i)$, $\forall i = 1, \dots, r'_u$. This filter is essentially equivalent to a sharp cutoff with $r'_u = r_{\text{cutoff}}$. It was used for the first time in [348], and then in [350, 349] to build a reduced version of the dynamic **SGS** and of the Leray models, respectively.

Another well-known type of filter, already introduced in Equation (1.9), is the **ROM differential filter** [283, 350, 349], which is obtained by Galerkin projection into the velocity modes, yielding the following linear system:

$$(\mathbf{I}_r - 2\delta^2 \mathbf{D}_{u_r}) \bar{\mathbf{a}} = \mathbf{a}, \quad (5.18)$$

where $\mathbf{I}_r \in \mathbb{R}^{r_u \times r_u}$ and $\mathbf{D}_{u_r} \in \mathbb{R}^{r_u \times r_u}$ are the reduced identity and diffusion matrices, respectively.

This approach can be generalized as **ROM higher-order algebraic filter** of the following form:

$$(\mathbf{I}_r - 2\delta^{2m} \mathbf{D}_{u_r}^m) \bar{\mathbf{a}} = \mathbf{a}, \quad (5.19)$$

where m is a positive integer. For $m = 1$, we exactly retrieve the **ROM DF** in Equation (5.18). The exponent m has been introduced [232] to control the percentage of filtering at different wavenumbers, ensuring an increasing energy damping as the wavenumber increases. It was proposed at the reduced level in [119], based on the formulation in [232], and recently used in [333].

A practical example of the effect of the **DF** at the reduced level is provided in Figure 5.3 on the test case of the lid-driven cavity at $Re = 10\,000$. A more detailed presentation of the

test case is provided in [336]. The figure analyzes the filtered solution at different δ values and at different reduced dimensions, obtained applying the DF to the G-ROM solution at a fixed time instance. On the left, the figure displays a graphical comparison among the G-ROM reconstruction, the filtered fields after applying the DF and the projection filter with cutoff dimensions $r'_u = 20\% r_u$ and $r'_u = 40\% r_u$. The graphical comparison shows that increasing values of δ produce a progressively stronger smoothing effect of the DF for all values of r_u . For $r_u = 5$, the projection filter yields a visibly smoother solution, whereas oscillatory structures remain present at higher latent dimensions. This behaviour is likely related to the nature of the POD basis: starting from the third mode, the modes already contain vortical structures and sharper gradients, thereby reproducing small-scale dynamics inherited from the FOM. On the right, the figure shows the effect of the DF directly on the reduced coefficients, which exhibit progressively smaller values as δ increases. At $\delta = 0.1$, the filtered coefficients are indeed close to 0, and the graphical field coincides with the velocity time-average³.

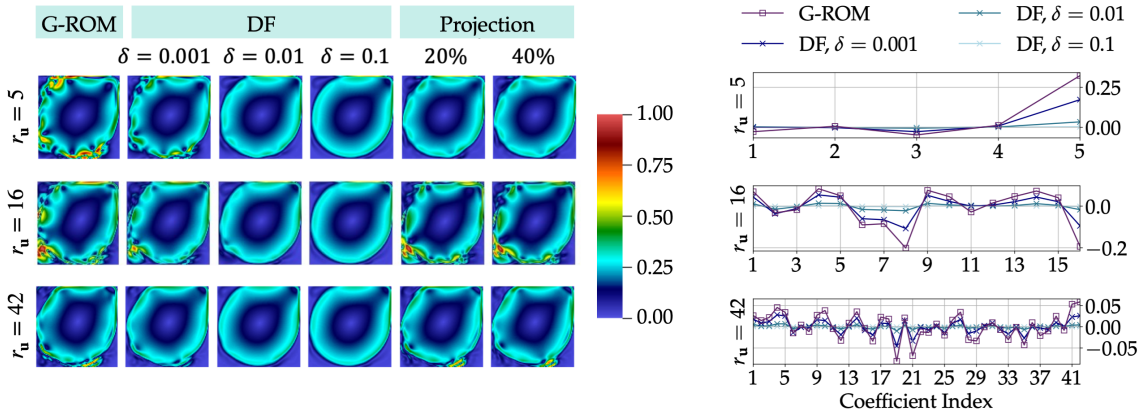


Figure 5.3: *Lid-driven cavity test case*. Example of the effect of the DF with three different δ values at three latent dimensions. The DF is applied for a single time step to a G-ROM solution, as in [336]. The graphical fields (left) and the reduced coefficients value (right) are displayed before (G-ROM) and after applying the filter (DF).

The above-mentioned spatial filters may be used in different LES-ROMs. A LES-ROM example is the EFR-ROM, which provides a reduced-order version of EFR (see Section 1.3 for the full-order formulation). This formulation was introduced in [119, 349], and has been applied to turbulent flows [227], ocean dynamics [113], feedback control [321]. Recent works introduced a deconvolution-based indicator function [112], and investigated the consistency between FOM and ROM parameters [324], i.e. δ and χ .

We here only focus on the Leray-Reduced Order Method (L-ROM), and refer the reader to the extensive review of [259].

³In this case, the zero-th mode is the lifting function, as in (5.9), which simply corresponds to the snapshots' time-average.

5.3.1 Leray-ROM (L-ROM)

L-ROM is the reduced extension of the approach of Equation (1.13), which consists in smoothing out the convective reduced velocity in the nonlinear term, namely:

$$-\mathbf{M}_{\mathbf{u}_r} \dot{\mathbf{a}} - \bar{\mathbf{a}}^T \mathbf{C}_{\mathbf{u}_r} \mathbf{a} + \nu(\mathbf{B}_r + \mathbf{B}_r') \mathbf{a} - \mathbf{G}_r \mathbf{b} = \mathbf{0}, \quad (5.20)$$

Remark: Leray-ROM in conservative form.

We remark that typically the **L-ROM** is written in advective form, where the nonlinear term appears in the reduced system as

$$((\bar{\mathbf{u}}_r \cdot \nabla) \mathbf{u}_r, \phi_i).$$

When writing the nonlinear term in conservative form we obtain:

$$((\bar{\mathbf{u}}_r \cdot \nabla) \mathbf{u}_r, \phi_i) = (\nabla \cdot (\bar{\mathbf{u}}_r \otimes \mathbf{u}_r), \phi_i) - ((\nabla \cdot \bar{\mathbf{u}}_r) \mathbf{u}_r, \phi_i).$$

In incompressible flows, if we employ a consistent divergence-free filter, $\nabla \cdot \bar{\mathbf{u}}_r = 0$. Hence, in conservative matrix form we can write the nonlinear term as $\bar{\mathbf{a}}^T \mathbf{C}_{\mathbf{u}_r} \mathbf{a}$.

5.4 Closure approaches for ROMs

Closure models aim to account, at the reduced level, for the effect of unresolved **POD** modes and for their interaction with the resolved scales. In this sense, they provide a reduced counterpart of the closure problem arising in turbulence modeling [14, 137, 286, 257, 6], which consists of modeling the closure term $\tau_{\mathbf{u}}$ in the momentum equation:

$$\mathcal{R}_{\mathbf{u}}(\mathbf{a}, \mathbf{b}; \mu) + \tau_{\mathbf{u}} = \mathbf{0}. \quad (5.21)$$

Different approaches can be used to model the closure term $\tau_{\mathbf{u}}$:

- ▶ **Functional modeling**, in which physical insight is used to build a model for the closure term like in **Eddy Viscosity-Reduced Order Method (EV-ROM)** [131].
- ▶ **Structural modeling**, in which mathematical methods (e.g., expansions and asymptotics) are used to build a model for the closure term. Examples are the scale-similarity model of [22], the **AD-ROM** approach based on approximating the inverse filtering operation [359, 295].
- ▶ **Stochastic modeling**, where the closure term is modelled by a stochastic process (e.g. Markovian or non-Markovian dynamics with random forcing). Examples are [65, 59, 215].
- ▶ **Data-driven modeling**, in which available numerical (**FOM**) or experimental data is used to build a model for the closure term, like in [357] or in modern machine learning based approaches [288, 289, 294, 74, 145].

We now mainly focus on a type of **functional model**, named **EV-ROM** (Section 5.4.1), and on a type of **data-driven model** (Section 5.4.2), which provide the background methodology for Chapter 7.

5.4.1 Eddy Viscosity-based ROM (EV-ROM)

EV-ROMs introduce additional dissipative contributions in the reduced equations in order to compensate for the missing effect of unresolved or poorly resolved scales. Early **G-ROMs** already used eddy viscosity-type corrections to account for the influence of discarded modes [14]. Later developments include constant or mode-dependent eddy viscosity closures, Smagorinsky-type **POD-ROMs**, dynamically calibrated eddy viscosity models, and optimal nonlinear eddy viscosity laws [286, 257].

Remark: Distinction between LES-based and RANS-based EV-ROMs.

A distinction should be made between **LES**-inspired or modal-truncation eddy viscosity closures and **RANS**-based eddy viscosity **ROMs**. In the former case, the eddy viscosity term is not necessarily the projection of a turbulent-viscosity field already available in the **FOM**. Rather, it is a modelled dissipative correction introduced to represent the effect of discarded **POD** modes and missing nonlinear interactions. In **RANS**-based **EV-ROMs**, instead, the eddy viscosity is inherited from the full-order turbulence model and represents the closure of the Reynolds stresses. Therefore, the **ROM** aims to approximate a quantity already present in the **FOM**, which guarantees consistency. Thus, while in **RANS**-based **ROMs** the eddy viscosity is associated with turbulence closure, in **LES-ROMs** it is primarily associated with modal-truncation closure.

We focus here on **RANS**-based **EV-ROMs**. First, we briefly recall the **RANS** and the **URANS** formulation for time-dependent flow problems. According to the Reynolds decomposition [272], a generic field $s(\mathbf{x}, t)$ can be decomposed into its mean and fluctuating parts, $\tilde{s}(\mathbf{x}, t)$ ⁴ and $s'(\mathbf{x}, t)$, respectively. For unsteady flows, the mean field is computed as:

$$\tilde{s}(\mathbf{x}, t) = \frac{1}{T} \int_t^{t+T} s(\mathbf{x}, \tau) d\tau, \text{ with } T_1 \leq T \leq T_2,$$

where T_1 and T_2 are the fluctuating and mean fields reference time scales. For steady flows, the mean field is not time-dependent and the above expression becomes:

$$\tilde{s}(\mathbf{x}) = \lim_{T \rightarrow \infty} \frac{1}{T} \int_t^{t+T} s(\mathbf{x}, \tau) d\tau.$$

If the Reynolds time-average logic is applied to the discretized incompressible **NSE** using a **FVM**, the discretized solutions $\mathbf{u}$ and p coincide with the Reynolds time-average

⁴Note that we use the notation $\tilde{\cdot}$ for the Reynolds time-average, differently from the standard literature notation $\bar{\cdot}$. We adopt this notation to avoid confusion with the spatial filter notation adopted in the rest of the thesis.

fields, while we express the fluctuating parts with $\mathbf{u}'$ and p' , respectively. Under this hypothesis, we obtain the **URANS** system:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot \nu (\nabla \mathbf{u} + (\nabla \mathbf{u})^T) + \nabla p - \nabla \cdot \Sigma = 0, \\ \nabla \cdot \mathbf{u} = 0. \end{cases} \quad (5.22)$$

If we compare the above System (5.22) with System (5.5), we can immediately see that there is an additional term, $-\nabla \cdot \Sigma$, where $\Sigma = -\overline{\mathbf{u}' \otimes \mathbf{u}'}$ is the Reynolds stress tensor⁵. At the full-order level, in order to close the system, we need to select a technique to model this term. Similarly to **LES** (see section 1.2.1), a well-known approach is modeling the term Σ by introducing the eddy viscosity field ν_t (Boussinesq hypothesis [40]), as follows:

$$\Sigma \approx \nu_t (\nabla \mathbf{u} + (\nabla \mathbf{u})^T). \quad (5.23)$$

Based on this hypothesis, System (5.22) may be written as follows:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot [(\nu + \nu_t)(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)] + \nabla p = 0, \\ \nabla \cdot \mathbf{u} = 0. \end{cases} \quad (5.24)$$

Moreover, by taking the divergence of the momentum equation in System (5.24), we obtain a turbulent version of the **PPE** described in Section 5.2.2. The final **URANS** system using the **PPE** approach can be expressed as follows:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot [(\nu + \nu_t)(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)] + \nabla p = 0, \\ \Delta p = -\nabla \cdot (\nabla \cdot (\mathbf{u} \otimes \mathbf{u})) + \nabla \cdot (\nabla \cdot \nu_t (\nabla \mathbf{u} + (\nabla \mathbf{u})^T)). \end{cases} \quad (5.25)$$

To ensure consistency with respect to the **FOM**, in **RANS**-based **EV-ROMs**, the eddy viscosity field is also introduced at the reduced-order level [131, 129], by applying a **POD** decomposition to the full-order field ν_t . We first collect the eddy viscosity snapshots:

$$\mathbf{S}_{\nu_t} = \begin{bmatrix} | & | & & | \\ \nu_{t1} & \nu_{t2} & \dots & \nu_{tN_\mu} \\ | & | & & | \end{bmatrix}, \quad (5.26)$$

Applying the **POD** to the snapshot matrix $\mathbf{S}_{\nu_t}$, after selecting an appropriate reduced dimension r_{ν_t} , the eddy viscosity modes are extracted: $\{\eta_i\}_{i=1}^{r_{\nu_t}}$. We can then write the following approximation:

$$\nu_t(\mathbf{x}, \boldsymbol{\mu}) \approx \nu_{tr}(\mathbf{x}, \boldsymbol{\mu}) = \sum_{i=1}^{r_{\nu_t}} g_i(\boldsymbol{\mu}) \eta_i(\mathbf{x}), \quad (5.27)$$

⁵The Reynolds stress tensor Σ has a different meaning with respect to the **SGS** stress tensor $\overline{\mathbf{T}}_{\text{SGS}}$ introduced in section 1.2.1. Both terms derive from the convective term, but in different settings (Reynolds time-average, and spatial filtering operation)

where the unknowns are the reduced coefficients: $\mathbf{g}(\boldsymbol{\mu}) = (g_i)_{i=1}^{r_{v_t}}$. The best approximation for a given reduced dimension r_{v_t} is given by the projected v_t , namely:

$$v_t^{\text{proj}}(\mathbf{x}, \boldsymbol{\mu}) = \sum_{i=1}^{r_{v_t}} g_i^{\text{proj}}(\boldsymbol{\mu}) \eta_i(\mathbf{x}), \text{ where } g_i^{\text{proj}}(\boldsymbol{\mu}) = (v_t(\mathbf{x}, \boldsymbol{\mu}), \eta_i(\mathbf{x}))_{L^2(\Omega)} \quad \forall i = 1, \dots, r_{v_t}. \quad (5.28)$$

If we focus on the **PPE** approach, a Galerkin projection of the closed System (5.25) into the velocity and pressure modes leads to the **EV-ROM** formulation:

$$\begin{cases} -\mathbf{M}_{u_r} \dot{\mathbf{a}} - \mathbf{a}^T \mathbf{C}_{u_r} \mathbf{a} + \nu (\mathbf{B}_r + \mathbf{B}_r') \mathbf{a} + \mathbf{g}(\mathbf{T}_{u_r} + \mathbf{T}_{u_r}') \mathbf{a} - \mathbf{G}_r \mathbf{b} = \mathbf{0}, \\ \mathbf{D}_{p_r} \mathbf{b} + \mathbf{a}^T \mathbf{C}_{p_r} \mathbf{a} - \mathbf{g}(\mathbf{T}_{p_r} + \mathbf{T}_{p_r}') \mathbf{a} - \nu \mathbf{N}_r \mathbf{a} = \mathbf{0}, \end{cases} \quad (5.29)$$

where the following tensors have been introduced:

$$\begin{aligned} (\mathbf{T}_{u_r})_{ijk} &= (\boldsymbol{\phi}_i, \nabla \cdot \eta_j \nabla \boldsymbol{\phi}_k)_{L^2(\Omega)}, \quad (\mathbf{T}_{u_r}')_{ijk} = (\boldsymbol{\phi}_i, \nabla \cdot \eta_j (\nabla \boldsymbol{\phi}_k)^T)_{L^2(\Omega)}, \\ (\mathbf{T}_{p_r})_{ijk} &= (\nabla \chi_i, \nabla \cdot \eta_j \nabla \boldsymbol{\phi}_k)_{L^2(\Omega)}, \quad (\mathbf{T}_{p_r}')_{ijk} = (\nabla \chi_i, \nabla \cdot \eta_j (\nabla \boldsymbol{\phi}_k)^T)_{L^2(\Omega)}. \end{aligned} \quad (5.30)$$

The **EV-ROM** system of (5.29) has $r_u + r_p + r_{v_t}$ unknowns (corresponding to the reduced coefficients $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{g}$), but only $r_u + r_p$ equations. Hence, the system is not closed. A way to address the closure, as done in [131], is to approximate the eddy viscosity coefficients $g_i, i = 1, \dots, r_{v_t}$ using an interpolation or regression approach starting from known (or resolved) coefficients, namely $\mathbf{a}$ and $\mathbf{b}$. Since the eddy viscosity is typically correlated with the velocity field, the common choice is to construct a mapping:

$$\mathcal{G} : \mathbb{R}^{r_u} \rightarrow \mathbb{R}^{r_u}, \quad \mathbf{g} = \mathcal{G}(\mathbf{a})$$

or, taking into account time variations in the reduced velocity:

$$\mathcal{G} : \mathbb{R}^{2r_u} \rightarrow \mathbb{R}^{r_u}, \quad \mathbf{g} = \mathcal{G}(\mathbf{a}, \dot{\mathbf{a}}).$$

Another option is to add the parameters' dependency within the mapping to improve the approximation:

$$\mathcal{G} : \mathbb{R}^{r_u + P} \rightarrow \mathbb{R}^{r_u}, \quad \mathbf{g} = \mathcal{G}(\mathbf{a}, \boldsymbol{\mu}).$$

The input and the type of this mapping depend on the considered test case (e.g., steady or unsteady configurations, number of parameters) and on the chosen approximation strategy. In literature, the mapping $\mathcal{G}$ is built either as a **Radial Basis Function (RBF)** [191] interpolation approach [129], or using a neural network regression [361, 143, 145]. In [146] different types of neural network architectures and inputs are investigated. In this thesis, we focus on a mapping of type $\mathcal{G}(\mathbf{a}, \boldsymbol{\mu})$. The target quantity that the approximated $\mathbf{g} = \mathcal{G}(\mathbf{a}, \boldsymbol{\mu})$ aims to reproduce is the projection $\mathbf{g}^{\text{proj}}$ in a pre-defined training set. If we consider the same training set as the **POD**, the loss function is:

$$\mathcal{L}_{\mathcal{G}} = \frac{1}{N_{\boldsymbol{\mu}}} \sum_{i=1}^{N_{\boldsymbol{\mu}}} \|\mathcal{G}(\mathbf{a}, \boldsymbol{\mu}_i) - \mathbf{g}^{\text{proj}}(\boldsymbol{\mu}_i)\|_2^2, \quad (5.31)$$

namely the **MSE**, which is one of the most commonly used loss functions in neural network training, as briefly introduced in Section 3.2.1.

5.4.2 Data-driven closure modeling in ROMs

This part describes closures based on **data-driven modeling**, which infer the missing contribution τ_u in Equation (5.21) from high-fidelity data. Rather than introducing a closure term derived from full-order modeling assumptions (as in **EV-ROMs**), these approaches learn the contribution of the neglected modes from high-fidelity data and incorporate the resulting correction into the reduced-order dynamics. In this sense, data-driven closures may be interpreted as learned *corrections* rather than *closures* in the classical physical sense, although they still play the role of modeling the unresolved modes [287, 6, 143].

This section describes the approach introduced in [357], which constructs a model for the closure term τ_u , based on the following steps.

1. Select a value $d_u > r_u$.

This automatically divides three different scales:

- ▶ $\phi_i, i = 1, \dots, r_u$: the **resolved scales** via the reduced system;
- ▶ $\phi_i, i = r_u + 1, \dots, d_u$: the **modelled scales**, whose contribution is modelled via the closure τ_u ;
- ▶ $\phi_i, i = d_u + 1, \dots, N_\mu$: the **discarded scales**, corresponding to the low-energy modes, whose closure contribution is negligible.

For this hierarchical scale division, this approach has been referred to as **DD-VMS-ROM** [143, 172]. In literature, d_u can be either proportional to the reduced dimension, i.e. hr_u , with $h \in \mathbb{N}$ (for instance, [357] considers $h = 5$), or can be chosen as $d_u = N_\mu$. In the latter case, we want to model the contribution of all the unresolved modes.

2. Compute the *exact* closure term τ_u^{true} as the discrepancy between the momentum equation projected onto d_u modes and onto r_u modes.

This term represents the contribution of the modes $\phi_{r_u+1}, \dots, d_u$ we aim to model and is computed considering the projected (and theoretically optimal) coefficients $a_i^{\text{proj}} = (\mathbf{u}, \phi_i), i = 1, \dots, d_u$. We indicate with $\mathbf{a}_d^{\text{proj}} = (a_i)_{i=1}^{d_u}$ and $\mathbf{a}^{\text{proj}} = (a_i)_{i=1}^{r_u}$ the reduced velocities projected onto d_u and r_u modes, respectively. More in general, the notations $(\cdot)_{d_u}$ and $(\cdot)_d$ indicate the operators or variables computed considering the resolved and modelled scales. The exact closure can be found as:

$$\tau_u^{\text{true}}(\mu) = \overline{\mathcal{R}_{d_u}(\mathbf{a}_d^{\text{proj}}, \mathbf{b}_d^{\text{proj}}; \mu)}^{r_u} - \mathcal{R}_{r_u}(\mathbf{a}^{\text{proj}}, \mathbf{b}^{\text{proj}}; \mu),$$

where the notation $\overline{(\cdot)}^{r_u}$ is a filtering operation corresponding to retaining only the first r_u coefficients. This filter operation coincides with the **ROM** projection filter described in Section 5.3 and it is here employed to have consistent dimensions between $\overline{\mathcal{R}_{d_u}}^{r_u}$ and $\mathcal{R}_{r_u}$. The only contribution to this closure is given by the nonlinear term, say:

$$\tau_u^{\text{true}} = -(\overline{\mathbf{a}_d^{\text{proj}}})^T \mathbf{C}_{ud} \mathbf{a}_d^{\text{proj}} + (\mathbf{a}^{\text{proj}})^T \mathbf{C}_{ur} \mathbf{a}^{\text{proj}}, \quad (5.32)$$

where $\mathbf{C}_{ud} \in \mathbb{R}^{d_u \times d_u \times d_u}$ is defined as $(\mathbf{C}_{ud})_{ijk} = (\phi_i, \phi_j \otimes \phi_k)_{L^2(\Omega)}$, for $i, j, k = 1, \dots, d_u$.

3. Choose a proper closure model $\mathcal{M}_\theta$ for τ_u and learn the optimal parameters θ from data.

If we assume a map of type $\mathcal{M}_\theta(\mathbf{a}, \mu)$, the model parameters can be optimized by solving the minimization problem:

$$\min_{\theta} \frac{1}{N_\mu} \sum_{i=1}^{N_\mu} \|\mathcal{M}_\theta(\mathbf{a}(\mu_i), \mu_i) - \tau^{\text{true}}(\mu_i)\|_2^2. \quad (5.33)$$

We can divide the parameters as $\mu = (t, \mu^{\text{phys, geom}})$, where $\mu^{\text{phys, geom}}$ are physical and/or geometrical parameters. The first contributions on DD-VMS-ROMs consider $\mu = t$ and proposed for τ_u the **quadratic ansatz**:

$$\tau_u^{\text{true}} \approx \tau_u = \mathcal{M}_{(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})}(\mathbf{a}) = \tilde{\mathbf{A}} \mathbf{a} + \mathbf{a}^T \tilde{\mathbf{B}} \mathbf{a}, \quad (5.34)$$

where $\tilde{\mathbf{A}} \in \mathbb{R}^{r_u \times r_u}$ and $\tilde{\mathbf{B}} \in \mathbb{R}^{r_u \times r_u \times r_u}$ are the parameters to be learned. In this case, the three-dimensional tensor $\tilde{\mathbf{B}}$ can be decomposed in $N_{\text{blocks}} = r_u(r_u + 1)/2$ two dimensional blocks and expressed as $\sum_{i=1}^{N_{\text{blocks}}} \tilde{\mathbf{B}}^{(i)} \mathbf{a}^{(i)}$, where $\tilde{\mathbf{B}}^{(i)}$ are two-dimensional matrices and $\mathbf{a}^{(i)}$ are vectors given by quadratic components of the coefficients. This mathematical trick transforms the problem of Equation (5.33) in a linear problem that can be trivially solved using a least squares method, as typically done in the context of operator inference. For the complete mathematical procedure, refer to [357, 251].

Other approaches consider nonlinear models like **neural networks** [358, 74, 146, 145], where the network weights θ can be learned from data via backpropagation (for more details about neural networks see Section 3.2.1). The map with pre-learned weights is here simply named $\mathcal{M}$.

4. Build and solve the data-driven reduced order model with the modelled closure for unseen parameters $\mu_\star$. The closed momentum equation is:

$$\mathcal{R}_{r_u}(\mathbf{a}, \mathbf{b}; \mu_\star) + \mathcal{M}(\mathbf{a}, \mu_\star) = 0. \quad (5.35)$$

Different variants and extensions of the above-described methodology have been proposed throughout the past years, in order to find a solution to the following questions.

Should matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ of the ansatz (5.34) satisfy any physical constraint to enforce dissipation?

This question has been addressed in [226], where physical constraints have been applied to $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$, to ensure that the correction term is dissipative, namely:

$$\mathbf{a}^T [\tilde{\mathbf{A}} \mathbf{a} + \mathbf{a}^T \tilde{\mathbf{B}} \mathbf{a}] \leq 0.$$

This implies solving a constrained least squares problem, which can be trivially implemented. The authors show that the constrained correction yields improved accuracy and stability of the approximation.

If a PPE approach is used, is it convenient to introduce a closure term also for the nonlinear term in the PPE?

We addressed this question in [143] by extending the quadratic ansatz (5.34) to the **PPE-ROM** formulation of Equation (5.14). We introduced an additional pressure closure $\tau_p \in \mathbb{R}^{r_p}$ aimed to reintroduce the contribution of the unresolved pressure modes into the **PPE**. To build this pressure correction, we follow a similar procedure as for τ_u .

In particular, we choose the number of pressure modes $d_p > r_p$ we want to model, and we compute the target pressure closure as follows:

$$\tau_p^{\text{true}} = \overline{(\mathbf{a}_d^{\text{proj}})^T \mathbf{C}_{p_d} \mathbf{a}_d^{\text{proj}}}^{r_p} - (\mathbf{a}^{\text{proj}})^T \mathbf{C}_{p_r} \mathbf{a}^{\text{proj}}. \quad (5.36)$$

The above expression is the contribution of the unresolved modes on the nonlinear part of the **PPE**. In this case, we selected a quadratic ansatz also considering the influence of the pressure coefficients:

$$\begin{bmatrix} \tau_u \\ \tau_p \end{bmatrix} = \tilde{\mathbf{A}} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} + [\mathbf{a}, \mathbf{b}] \tilde{\mathbf{B}} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \quad (5.37)$$

where $\tilde{\mathbf{A}} \in \mathbb{R}^{(r_u+r_p) \times (r_u+r_p)}$ and $\tilde{\mathbf{B}} \in \mathbb{R}^{(r_u+r_p) \times (r_u+r_p) \times (r_u+r_p)}$ can be found by solving a least squares problem just as before. The reduced system for an unseen parameter $\mu_\star = (t_\star)$ including the closures τ_u and τ_p becomes

$$\begin{cases} \mathcal{R}_{r_u}(\mathbf{a}, \mathbf{b}; \mu_\star) + \tau_u(\mathbf{a}, \mathbf{b}) = \mathbf{0}, \\ \mathcal{R}_{r_p}(\mathbf{a}, \mathbf{b}; \mu_\star) + \tau_p(\mathbf{a}, \mathbf{b}) = \mathbf{0}, \end{cases} \quad (5.38)$$

We showed that the addition of the pressure correction significantly improved the approximation of the pressure field.

Are there any theoretical bounds for the error for the corrected G-ROMs?

The work of [171] proves the *verifiability* of the corrected **ROM** with a linear ansatz, namely $\tau_u = \tilde{\mathbf{A}}\mathbf{a}$. The authors proved that small errors in the closure model ($\|\tau_u^{\text{true}} - \tilde{\mathbf{A}}\mathbf{a}\|_2^2$) imply small errors in the **ROM** approximation, which is the concept of verifiability. However, this has not been proved for a quadratic or nonlinear ansatz.

Is it possible to propose a different mapping $\mathcal{M}$ without explicit dependency on the velocity coefficients $\mathbf{a}$?

This problem is addressed in [172], where the closure term is built as a function of the **ROM** residual $\mathcal{R}_{r_u}$, used as an indicator of the unresolved dynamics and truncation error.

Is it possible to extend the quadratic ansatz to turbulent flow cases, within the EV-ROM framework?

This question has been addressed in [142], by considering a correction for the System (5.29). In that case, the following exact closures can be computed by adding the turbulence contribution:

$$\begin{aligned} \tau_u^{\text{true}} &= -\overline{(\mathbf{a}_d^{\text{proj}})^T \mathbf{C}_{u_d} \mathbf{a}_d^{\text{proj}}}^{r_u} + (\mathbf{a}^{\text{proj}})^T \mathbf{C}_{u_r} \mathbf{a}^{\text{proj}} + \\ &\quad + (\mathbf{g}_d^{\text{proj}})^T (\mathbf{T}_{u_d} + \mathbf{T}_{u_d}') \mathbf{a}_d^{\text{proj}} - (\mathbf{g}^{\text{proj}})^T (\mathbf{T}_{u_r} + \mathbf{T}_{u_r}') \mathbf{a}^{\text{proj}}, \\ \tau_p^{\text{true}} &= \overline{(\mathbf{a}_d^{\text{proj}})^T \mathbf{C}_{p_d} \mathbf{a}_d^{\text{proj}}}^{r_p} - (\mathbf{a}^{\text{proj}})^T \mathbf{C}_{p_r} \mathbf{a}^{\text{proj}} + \\ &\quad - (\mathbf{g}_d^{\text{proj}})^T (\mathbf{T}_{p_d} + \mathbf{T}_{p_d}') \mathbf{a}_d^{\text{proj}} + (\mathbf{g}^{\text{proj}})^T (\mathbf{T}_{p_r} + \mathbf{T}_{p_r}') \mathbf{a}^{\text{proj}}. \end{aligned} \quad (5.39)$$

The remaining part of the procedure remains exactly as before.

The works [357, 226, 143, 142] only consider time-dependent problems without any additional physical or geometrical parameters. Is the quadratic ansatz extendable to flow cases with multiple parameters?

In the case where time is the unique parameter, $\mu = (t)$, we only optimize fixed non time-dependent $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$. The quadratic model is extendable to the case $\mu = (t, \mu^{\text{phys, geom}})$ if we consider parametrized matrices $\tilde{\mathbf{A}}(\mu^{\text{phys, geom}})$ and $\tilde{\mathbf{B}}(\mu^{\text{phys, geom}})$, which account for configuration changes. However, this is not trivial and a common choice is to replace the quadratic ansatz with a neural network, which also takes as input the parameters $\mu^{\text{phys, geom}}$, either explicitly or implicitly within the coefficients $\mathbf{a}$.

5.5 Motivation for the proposed approaches

The state-of-the-art filtering and closure approaches developed during the past years, and partially presented in Sections 5.3 and 5.4, have significantly improved the accuracy, robustness, and long-time stability of classical **G-ROMs**. Nevertheless, several challenges remain open, especially in the context of complex turbulent flows, transport problems, nonlinear dynamics, and parameterized flow cases.

On the one hand, the effect of classical **ROM** spatial filters (Section 5.3) often depends on the filter radius, whose value should be tuned case by case. As described in part I, also in **ROMs**, standard literature choices for the filter radius may lead either to overdiffusivity or to insufficient filtering (i.e., spurious oscillations).

On the other hand, the data-driven closure modeling introduced in Section 5.4.2 has proved to be accurate in time-dependent test cases, but has limited extensions to parameterized flow cases, namely when the parameters are not only composed of the physical time t .

These limitations motivate the research questions addressed in part II of this thesis, which aims to further investigate and improve the existing filtering, stabilization, and closure models in the reduced-order framework.

Part II addresses the following questions:

- *Can we employ available **FOM** data to build a stabilization operator that replaces standard spatial filters in **LES-ROMs**?*

In Chapter 6 we introduce a novel operator named **StabOp**, which is different from classical spatial filters, and is built entirely from data. This stabilization operator is used as a mapping from the reduced velocity $\mathbf{a}$ to the filtered velocity $\bar{\mathbf{a}}$.

- *In Chapter 4 a data-driven filter is also introduced. However, the **DD-EF** approach is unstable in time extrapolation settings, and leads to energy blow up. How can we alleviate this issue in **StabOp**?*

StabOp is learned via a PDE-constrained optimization. Within the learning process, the **L-ROM** system is solved for a certain number of training time steps, making

the **L-ROM** energy as close as possible to the **FOM** reference energy. In this way, StabOp incorporates information on the stability of the reduced model, ensuring a more stable and accurate long-time dynamics.

- *How can the approach of [142] be generalized to a parameterized flow case while maintaining stability?*

Chapter 7 provides a parametric extension of the DD-VMS-ROM proposed in [142] using machine learning. More in detail, the DeepONet architecture is employed to separately elaborate the contributions of the flow parameters μ and of the reduced velocity $\mathbf{a}$. Stability can be ensured by introducing a joint training process for both the eddy viscosity map $\mathcal{G}$ and the correction map $\mathcal{M}$. The mutual communication of the functional and data-driven maps during training will help in improving the stability of the **ROM**.

- *Is it possible to extend the closure problem to improve the accuracy of nonintrusive **ROMs**?*
This question has been addressed in Chapter 8, where a correction term of quadratic form (5.34) is learned from data, and directly added to the **ROM** solution. The goal is to model the contribution of the neglected modes, for the first time, in a simplified and purely data-driven setting.

STABOP-L-ROM: DATA-DRIVEN STABILIZATION FOR LERAY-ROMs

Spatial filters have played a central role in the simulation of convection-dominated flows in the past decades, both at the full-order level (see Chapter 1) and in ROMs (see Section 5.3). However, several questions remain to be answered, namely:

What is the most suitable filter for a given application and/or regime?

and

Once we select a filter, how do we choose the parameters (like the filter width δ)?

The present chapter overcomes these issues by replacing the standard spatial filter with a fundamentally different stabilization operator, which is built upon FOM available data. The novel data-driven stabilization operator, which is named **StabOp** [336], is tested on the L-ROM formulation, but can be extended to all LES-ROMs, based on all classical discretization approaches. The construction of StabOp is test-case dependent and is based on two steps: (i) postulating an ansatz (linear, quadratic or nonlinear), and (ii) learn the map via a PDE-constrained optimization which minimizes the discrepancy of a pre-selected Quantity of Interest (QoI) with respect to the FOM reference.

This chapter is divided into the following parts:

- ▶ A preliminary section which clarify the G-ROM and L-ROM formulations in the absence of pressure modeling (Section 6.1);
- ▶ The presentation of StabOp and the algorithm used to compute it (Section 6.2);
- ▶ The numerical investigation on four benchmarks, both two- and three-dimensional, in a convection-dominated regime (Section 6.3).
- ▶ An analysis on the difference between StabOp and standard spatial filters (Section 6.2). We answer the question: *Is StabOp a spatial filter in its classical sense?*
- ▶ A conclusive section, highlighting strengths and weaknesses of StabOp, and future perspectives of this work (Section 6.5).

6.1 Preliminaries: FOM, G-ROM and L-ROM

6.1.1 FOM

The **FOM** used for this work is built using the Spectral Element Method (SEM). In particular, it is constructed by applying a Galerkin projection of the incompressible **NSE** onto the spectral element space with $\mathbb{P}_N - \mathbb{P}_{N-1}$ velocity-pressure coupling. For time discretization, we employ the semi-implicit BDFk/EXTk scheme [94], which uses the k th-order backward differentiation (BDFk) for the time-derivative, k th-order extrapolation (EXTk) for the advection and forcing terms, and an implicit treatment of the dissipation terms. As in [94], we set $k = 3$ so that the imaginary eigenvalues associated with the skew-symmetric advection operator lie within the stability region of the BDFk/EXTk. The resulting full discretization leads to solving a linear unsteady Stokes system at each time step. For additional details on the **FOM** derivation, we refer to [335]. All the test cases considered are time-dependent, and we do not consider any additional physical or geometrical parameters.

6.1.2 G-ROM

The **G-ROM** is built following the procedure described in Section 5.1, i.e. by Galerkin projection of the discretized incompressible **NSE** into a reduced basis functions found with a **POD** procedure. Given a pre-defined latent dimension r_u , the **ROM** space is spanned by the first r_u modes $\{\phi_i\}_{i=1}^{r_u}$. To enforce the non-homogeneous boundary conditions, a lifting function approach is adopted (see Section 5.1.1). The **ROM** approximation is given by:

$$\mathbf{u}_r(\mathbf{x}, t) = \phi_0(\mathbf{x}) + \sum_{i=1}^{r_u} a_i(t) \phi_i(\mathbf{x}). \quad (6.1)$$

By replacing the expression (6.1) and applying a Galerkin projection into each mode ϕ_i , we obtain:

$$\left(\frac{\partial \mathbf{u}_r}{\partial t}, \phi_i \right)_{L^2(\Omega)} + (\nabla \cdot (\mathbf{u}_r \otimes \mathbf{u}_r), \phi_i)_{L^2(\Omega)} + Re^{-1} (\nabla \mathbf{u}_r, \nabla \phi_i)_{L^2(\Omega)} = 0 \quad \forall i = 1, \dots, r_u. \quad (6.2)$$

The **G-ROM** in matrix form reads as:

$$\boxed{\mathbf{M}_{u_r} \dot{\mathbf{a}} + \mathbf{a}^T \mathbf{C}_{u_r} \mathbf{a} + Re^{-1} \mathbf{A}_r \mathbf{a} = \mathbf{0}}, \quad (6.3)$$

where $\mathbf{M}_{u_r}$ and $\mathbf{C}_{u_r}$ are defined in (5.8), while the stiffness operator is defined as: $(\mathbf{A}_r)_{ij} = (\nabla \phi_i, \nabla \phi_j)_{L^2(\Omega)}$.

We note that this formulation is slightly different from that in (5.6), in the following aspects.

- In the case of fixed geometries, the divergence and pressure terms drop out of 6.1 because the ROM basis functions are weakly divergence-free. This formulation is typically called **momentum equation solely ROM** and only approximates the velocity field.

- The formulation is adimensionalized.
- The dissipation term is obtained by applying integration by parts.

6.1.3 L-ROM

Exactly as introduced in section 5.3.1, the **L-ROM** can be written by considering the filtered reduced velocity into the nonlinear term, namely:

$$\boxed{\mathbf{M}_{ur} \dot{\mathbf{a}} + \bar{\mathbf{a}}^T \mathbf{C}_{ur} \mathbf{a} + Re^{-1} \mathbf{A}_r \mathbf{a} = \mathbf{0}}, \quad (6.4)$$

where $\bar{\mathbf{a}}$ is found using the **DF**¹:

$$\bar{\mathbf{a}} = (\mathbf{I}_r + \delta^2 \mathbf{A}_r)^{-1} \mathbf{a}. \quad (6.5)$$

6.2 StabOp: a data-driven stabilization operator

As motivated in Section 5.5, standard spatial filters, like the **DF**, may lead to inaccurate or overly diffusive results. Our goal is to improve the accuracy by replacing the **DF** with a novel data-driven stabilization operator, **StabOp**. We indicate the StabOp map as $\mathcal{S}(\cdot, \boldsymbol{\theta}) : \mathbb{R}^{r_u} \rightarrow \mathbb{R}^{r_u}$, parametrized by the vector $\boldsymbol{\theta}$. The **DF** of Equation (6.5) is hence replaced by:

$$\bar{\mathbf{a}} = \mathcal{S}(\mathbf{a}; \boldsymbol{\theta}). \quad (6.6)$$

The parameters $\boldsymbol{\theta}$ are optimized to minimize the loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N_{\text{train}}} \|\mathcal{E}_{\text{FOM}}(t_i) - \mathcal{E}_{\text{StabOp-L-ROM}}(t_i; \mathbf{a})\|_2^2 + \omega_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2^2, \quad (6.7)$$

where N_{train} are the training time instances, $\mathcal{E}_{\text{FOM}}(t_i)$ and $\mathcal{E}_{\text{StabOp-L-ROM}}(t_i)$ are the kinetic energies associated with the **FOM** and with the **Stabilization Operator strategies for Leray-Reduced Order Method (StabOp-L-ROM)** models at time t_i . The loss in (6.7) considers the discrepancy between the novel model and the **FOM** reference in terms of kinetic energy, but other quantities of interest, e.g. forces acting on objects, may be optimized. The term $\omega_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2^2$ in the loss function is a regularization term which helps in preventing overfitting, and $\alpha_{\boldsymbol{\theta}} > 0$ is a user-defined parameter.

Different ansatz may be considered for the StabOp map $\mathcal{S}$. In this work, we explore three representative cases:

- **Linear form**: $\mathcal{S}(\mathbf{a}; \boldsymbol{\theta}) = \tilde{\mathbf{A}} \mathbf{a} + \tilde{\mathbf{c}}$, with $\tilde{\mathbf{A}} \in \mathbb{R}^{r_u \times r_u}$ and $\tilde{\mathbf{c}} \in \mathbb{R}^{r_u}$. In this case: $\boldsymbol{\theta}$ includes all the elements of $[\tilde{\mathbf{A}}, \tilde{\mathbf{c}}]$;
- **Quadratic form**: $\mathcal{S}(\mathbf{a}; \boldsymbol{\theta}) = \tilde{\mathbf{A}} \mathbf{a} + \mathbf{a}^T \tilde{\mathbf{B}} \mathbf{a} + \tilde{\mathbf{c}}$, with $\tilde{\mathbf{A}} \in \mathbb{R}^{r_u \times r_u}$, $\tilde{\mathbf{B}} \in \mathbb{R}^{r_u \times r_u \times r_u}$ and $\tilde{\mathbf{c}} \in \mathbb{R}^{r_u}$. In this case: $\boldsymbol{\theta}$ includes all the elements of $[\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{c}}]$;

¹Note that in this formulation the factor 2 is omitted.

► **Nonlinear form:** $\mathcal{S}(\mathbf{a}; \boldsymbol{\theta}) = \text{NN}_{\boldsymbol{\theta}}(\mathbf{a})$, where NN is a neural network with l layers and each i -th layer $L^{(i)}$ is the combination of a linear and a nonlinear operation: $L^{(i)} = h \circ [\mathbf{W}^i \quad \mathbf{b}^{(i)}]^T$, applied to $\begin{bmatrix} \mathbf{y}^{(i-1)} \\ 1 \end{bmatrix}$, where $\mathbf{y}^{(i-1)}$ is the output of the previous layer. $\mathbf{y}^{(0)} = \mathbf{a}$, $\mathbf{y}^{(l)} = \bar{\mathbf{a}}$, and the parameters are the compositions of all the layers' parameters (typically named *weights*): $\{(\mathbf{W}^{(i)}, \mathbf{b}^{(i)})_{i=1}^l\}$. h is instead the activation function, which represents the nonlinearity of the NN.

For each model, we compute the optimal parameters $\boldsymbol{\theta}_\star$ by solving a PDE-constrained optimization problem:

$$\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \text{ subject to a solving StabOp-L-ROM:} \quad (6.8)$$

$$\mathbf{M}_{\mathbf{u}_r} \dot{\mathbf{a}} + (\mathcal{S}(\mathbf{a}; \boldsymbol{\theta}))^T \mathbf{C}_{\mathbf{u}_r} \mathbf{a} + \text{Re}^{-1} \mathbf{A}_r \mathbf{a} = \mathbf{0}.$$

The linear, quadratic and nonlinear forms have been used to postulate closure ansatzes, such as in [357, 226, 143]. The above-mentioned works demonstrate that, even with simple parametrizations, the standard ROM accuracy may be improved and complex dynamics can be captured. However, this case is fundamentally different as we solve a PDE-constrained optimization problem, which gives important information on the ROM stability in time.

The StabOp training algorithm is presented in Algorithm 2. The training procedure involves the selection of two hyperparameters: the differential filter radius δ and the L^2 regularization weight ω_θ . The parameter δ controls the amount of filtering applied through the ROM DF, which is used to initialize the operator parameters and guide the training process. We denote by $\mathbb{P}_\delta$ and $\mathbb{P}_\omega$ the discrete search spaces for δ and ω_θ , respectively.

Remark: integration of the DF into StabOp. In challenging numerical experiments, such as those presented in Section 6.3, the initialization of StabOp $\mathcal{S}$, plays a key role in ensuring stable PDE-constrained optimization. In all test cases except the 2D flow past a cylinder problem, where no special initialization was required for stability, we use the differential filter associated with a prescribed filter radius δ , $(\mathbf{I}_r + \delta^2 \mathbf{A}_r)^{-1}$, as an initialization mechanism. For the linear and quadratic model forms, the differential filter is incorporated in two possible ways. In the first approach, the trainable linear operator $\tilde{\mathbf{A}}$ is initialized directly with the differential filter. In the second approach, the model is augmented with an additional linear operator initialized as the differential filter and added explicitly to the model form, i.e. $(\mathbf{I}_r + \delta^2 \mathbf{A}_r)^{-1} + \tilde{\mathbf{A}}$. In both cases, the differential filter remains trainable and is updated during optimization. For the nonlinear model, we employ the augmented formulation, in which the differential filter appears as an explicit additive component alongside the network output, i.e. $(\mathbf{I}_r + \delta^2 \mathbf{A}_r)^{-1} \mathbf{a} + \text{NN}_{\boldsymbol{\theta}}(\mathbf{a})$. In this case, the differential filter operator is kept fixed during training and is not updated during optimization. Different filter radii are considered to generate different filter operators,

each providing a distinct initialization for training the nonlinear model. For each choice of filter radius, the nonlinear model is trained with the corresponding fixed differential filter, and the model achieving the best performance is selected. Although this component could, in principle, be treated as trainable, we keep it fixed in this work and leave the investigation of trainable variants for future study. Overall, this initialization strategy provides a stable starting point for training StabOp, particularly in the under-resolved regime.

For each given (δ, ω_θ) pair, the loss function $\mathcal{L}(\theta)$ is minimized in the training time window $[t_0, T_{\text{train}}]$ using the L-BFGS optimizer, which iteratively updates the model parameters θ using gradients computed via reverse-mode automatic differentiation and a line-search procedure. Hyperparameter selection is based jointly on a validation loss (i.e., evaluated within a validation time window $[T_{\text{train}}, T_{\text{val}}]$) and a growth constraint.

The validation loss is defined to balance accuracy and variability, and it coincides with the widely used R^2 score, namely:

$$R_{\text{val}}^2 = 1 - \frac{\sum_{k=N_{\text{train}}}^{N_{\text{val}}} (\mathcal{E}_{\text{FOM}}(t_k) - \mathcal{E}_{\text{ROM}}(t_k))^2}{\sum_{k=N_{\text{train}}}^{N_{\text{val}}} (\mathcal{E}_{\text{FOM}}(t_k) - \bar{\mathcal{E}}_{\text{val}})^2}, \quad (6.9)$$

with $\bar{\mathcal{E}}_{\text{val}}$ denoting the mean of $\mathcal{E}_{\text{FOM}}$ over the interval $[T_{\text{train}}, T_{\text{val}}]$, can yield misleadingly high values when the validation data exhibit low variability. In such cases, the denominator of R_{val}^2 becomes small, and a dissipative model that stays close to the mean trajectory can achieve a deceptively high score despite underpredicting the temporal fluctuations of the validation data [239, 238]. Motivated by these findings, we augment it with the error in the variance

$$\epsilon_{\text{std, val}} = \frac{|\sigma_{\text{val}}(\mathcal{E}_{\text{ROM}}) - \sigma_{\text{val}}(\mathcal{E}_{\text{FOM}})|}{\sigma_{\text{val}}(\mathcal{E}_{\text{FOM}})}, \quad (6.10)$$

where $\sigma_{\text{val}}(\cdot)$ denotes the standard deviation of the corresponding samples in the validation time window. This yields the following combined validation loss

$$\mathcal{L}_{\text{val}} = -R_{\text{val}}^2 + \mu_{\text{val}} \epsilon_{\text{std, val}}, \quad (6.11)$$

where the parameter μ_{val} balances accuracy and variability: smaller values of μ_{val} favour a high R_{val}^2 , while larger values of μ_{val} prioritize matching the **FOM** variability, thereby preventing false selections that would otherwise lead to overly dissipative models. In all the numerical tests proposed in Section 6.3, we set $\mu_{\text{val}} = 2.5$, selected in a preliminary numerical investigation. The above-mentioned growth constraint is defined as:

$$\frac{\max_{t \in [t_0, T]} |\mathcal{E}_{\text{ROM}}(t_k) - \tilde{\mathcal{E}}_{\text{train}}|}{\max_{1 \leq k \leq N_{\text{train}}} |\mathcal{E}_{\text{FOM}}(t_k) - \tilde{\mathcal{E}}_{\text{train}}|} < \epsilon_{\text{growth}}, \quad (6.12)$$

where $\tilde{\mathcal{E}}_{\text{train}}$ is the average of $\mathcal{E}_{\text{FOM}}$ over T_{train} . This condition requires that the maximum deviation of the **StabOp-L-ROM** predicted $\tilde{\mathcal{E}}_{\text{train}}$ over the entire interval (including training, validation and test) $[t_0, T]$ remains bounded by the largest deviation observed within

the training interval, up to a tolerance ϵ_{growth} . In all the numerical experiments of Section 6.3, we set $\epsilon_{\text{growth}} = 2$.

Remark: integration of the DF into StabOp.

We emphasize that, although **StabOp-L-ROM** and **L-ROM** are simulated over the target interval $[t_0, T]$ during training, no test data are used for hyperparameter tuning. The growth constraint serves exclusively to reject candidate models that fit the validation data but exhibit instability outside the validation region [103].

To ensure a fair comparison of the novel StabOp with respect to standard spatial filters, we optimize the δ value in the **L-ROM**, following the procedure of Algorithm 3. Similarly to Algorithm 2, we consider a PDE-constrained optimization of δ within the training time window $[t_0, T_{\text{train}}]$.

Algorithm 2 Training of StabOp $\mathcal{S}(\mathbf{a}; \boldsymbol{\theta})$.

- 1: **Input:** Initial condition $\mathbf{a}(0)$, ground truth $\mathcal{E}_{\text{FOM}}(t)$ trajectory, learning rate λ_{lr} , number of epochs N_{epochs} , hyperparameter spaces for filter radius $\mathbb{P}_{\delta}$ and L^2 regularization weight $\mathbb{P}_{\omega}$.
 - 2: **for** $(\delta, \omega_{\theta}) \in \mathbb{P}_{\delta} \times \mathbb{P}_{\omega}$ **do**
 - 3: Initialize network parameters $\boldsymbol{\theta}_{(0)}$ using the ROM differential filter with parameter δ
 - 4: **for** $n = 1, \dots, N_{\text{epochs}}$ **do**
 - 5: Solve the StabOp-L-ROM with $\mathcal{S}(\cdot; \boldsymbol{\theta}_{(n-1)})$, and evaluate $\mathcal{E}_{\text{StabOp-L-ROM}}$ over $[t_0, T_{\text{train}}]$
 - 6: Compute the loss: $\mathcal{L}(\boldsymbol{\theta})$
 - 7: Compute the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
 - 8: Update parameters using an L-BFGS step: $\boldsymbol{\theta}_{(n)} \leftarrow \text{LBFGS}(\boldsymbol{\theta}_{(n-1)}; \mathcal{L}, \nabla \mathcal{L})$
 - 9: Solve the StabOp-L-ROM with $\mathcal{S}(\cdot; \boldsymbol{\theta}_{(n)})$ and evaluate and store $\mathcal{E}_{\text{StabOp-L-ROM}}$ over the validation interval $[t_0, T_{\text{val}}]$
 - 10: Evaluate the validation loss $\mathcal{L}_{\text{val}}$ (6.11)
 - 11: **if** $\mathcal{L}_{\text{val}}$ is smaller than the previous one **then**
 - 12: Update the best network parameter $\boldsymbol{\theta}_{\star}$ with $\boldsymbol{\theta}_{(n)}$
 - 13: **end if**
 - 14: **end for**
 - 15: Solve the StabOp-L-ROM with $\mathcal{S}(\cdot; \boldsymbol{\theta}_{\star})$ and evaluate and store $\mathcal{E}_{\text{StabOp-L-ROM}}$ over the target interval $[t_0, T]$
 - 16: **if** the predicted $\mathcal{E}_{\text{StabOp-L-ROM}}$ satisfies the growth constraint (6.12) **then**
 - 17: Evaluate the validation loss $\mathcal{L}_{\text{val}}$
 - 18: **if** $\mathcal{L}_{\text{val}}$ is less than the optimal $\mathcal{L}_{\text{val}}$ **then**
 - 19: Update the final network parameter $\boldsymbol{\theta}$ with $\boldsymbol{\theta}_{\star}$
 - 20: **end if**
 - 21: **end if**
 - 22: **end for**
 - 23: **Output:** Trained parameters $\boldsymbol{\theta}_{\star}$ that define the StabOp $\mathcal{S}$.
-

Algorithm 3 Finding optimal filter radius δ for L-ROM.

```

1: Input: Initial condition  $\mathbf{a}(0)$ , ground truth  $\mathcal{E}_{\text{FOM}}(t)$  trajectory, hyperparameter space
   for filter radius  $\mathbb{P}_\delta$ 
2: for  $\delta \in \mathbb{P}_\delta$  do
3:   Solve L-ROM using filter radius  $\delta$ , and evaluate and store  $\mathcal{E}_{\text{L-ROM}}$  over the target
   interval  $[t_0, T]$ 
4:   if the predicted  $\mathcal{E}_{\text{L-ROM}}$  satisfies the growth constraint (6.12) then
5:     Evaluate the validation loss  $\mathcal{L}_{\text{val}}$  (6.11)
6:     if  $\mathcal{L}_{\text{val}}$  is less than the optimal  $\mathcal{L}_{\text{val}}$  then
7:       Set  $\delta_{\text{optimal}} \leftarrow \delta$ 
8:     end if
9:   end if
10: end for
11: Output: Optimal filter radius  $\delta_{\text{optimal}}$  for L-ROM.
    
```

6.3 Numerical results

This Section illustrates the numerical results obtained by applying the **StabOp-L-ROM** approach to four different test cases:

- ▶ the 2D flow past a cylinder at $Re = 500$ (Section 6.3.1);
- ▶ the 2D lid-driven cavity flow at $Re = 10\,000$ (Section 6.3.2);
- ▶ the 3D flow past a hemisphere at $Re = 2200$ (Section 6.3.3);
- ▶ the 3D minimal channel flow at $Re = 5000$ (Section 6.3.4).

The results are compared in terms of the kinetic energy behaviour in time, where the **FOM** and **ROM** kinetic energies are defined as follows:

$$\mathcal{E}_{\text{FOM}}(t) = \frac{1}{2} \underline{\mathbf{u}}(t)^T \mathbf{M}_{\underline{\mathbf{u}}} \underline{\mathbf{u}}(t), \quad \mathcal{E}_{\text{ROM}}(t) = \frac{1}{2} \mathbf{a}(t)^T \mathbf{M}_{\mathbf{u}} \mathbf{a}(t), \quad (6.13)$$

where $\underline{\mathbf{u}}$ is the spectral element coefficient vector of the velocity field $\mathbf{u}$, and $\mathbf{M}_{\underline{\mathbf{u}}}$ is the **FOM** mass matrix.

Moreover, we compare the proposed **StabOp-L-ROMs** with the **L-ROM** and the **G-ROM** through the energy spectrum computed from the spatial velocity field at a fixed time instant. To this end, the velocity field is first interpolated onto a uniform Cartesian grid, and a Hann window is applied in each spatial direction to reduce spectral leakage while preserving energy. The Fourier transform of each velocity component is then computed. The physical wavenumbers k are defined from the grid spacing as $k_i = \frac{2\pi}{dx_i}$ for $i = 1, \dots, d$, where d denotes the dimensionality of the test case (2 or 3 in the present study). This definition ensures that each mode corresponds to a physical spatial frequency. The spectral energy density is then computed as

$$\mathcal{E}(\mathbf{k}) = \frac{1}{2} \sum_{i=1}^d |\hat{\mathbf{u}}_i|^2,$$

where $\hat{u}_i$ denotes the Fourier transform of the i -th velocity component. Finally, the isotropic energy spectrum is obtained by averaging $\mathcal{E}(\mathbf{k})$ over circular shells in 2D or spherical shells in 3D corresponding to constant wavenumber magnitude $\kappa = |\mathbf{k}|$.

One final analysis done for each test case collects the statistics regarding two fundamental aspects of numerical simulations:

- **Stability**, measured as the fraction of time steps for which the model's kinetic energy remains within a prescribed tolerance relative to the **FOM** kinetic energy. Specifically, at each time step t , the **ROM** energy is checked against the following limits:

$$\mathcal{E}_{\min}(t) < \mathcal{E}_{\text{ROM}}(t) < \mathcal{E}_{\max}(t), \quad (6.14)$$

where the bounds are defined as

$$\mathcal{E}_{\min}(t) = \mathcal{E}_{\text{FOM}}(t) - 2\sigma(\mathcal{E}_{\text{FOM}}), \quad \mathcal{E}_{\max}(t) = \mathcal{E}_{\text{FOM}}(t) + 2\sigma(\mathcal{E}_{\text{FOM}}), \quad (6.15)$$

with $\sigma(\mathcal{E}_{\text{FOM}})$ denoting the standard deviation of the **FOM** kinetic energy.

- **Accuracy**, measured as the fraction of time steps at which the model achieves the smallest instantaneous error among all models considered.

Finally, the computational costs associated with the standard **G-ROM** and **L-ROM**, and with the novel **StabOp-L-ROMs** are evaluated in Section 6.3.5, both at the offline and online stages.

We also specify that, since we consider the adimensionalized **NSE**, the time in the following plots is measured in convective time units (CTUs) and not in physical time.

6.3.1 Flow past a cylinder test case

Our first test problem is the 2D flow past a cylinder at Reynolds number $Re = 500$. The computational domain is $\Omega = [-2.5, 17] \times [-5, 5]$, where the cylinder has unit diameter and is centered at the origin. We consider the time interval $[500, 600]$, where the vortex shedding is already well developed.

The **ROM** basis functions $\{\phi_i\}_{i=1}^{r_u}$ are constructed using **POD** with $N_{\text{train}} = 2001$ snapshots collected over the training interval $[t_0, T_{\text{train}}] = [500, 520]$, corresponding to a sampling frequency of 0.01. The zeroth mode ϕ_0 is defined as the velocity field at $t = 500$, and the **ROM** initial condition is obtained by projecting the lifted snapshot at $t = 500$ onto the reduced space.

We consider four reduced dimensions, namely $r_u = 4, 6, 8$, and 10. These values correspond to a retained energy > 0.99 . Since the case $r_u = 1$ captures only 73.62% of the snapshot energy and is of limited practical relevance, it is not considered. Thus, for the flow past a cylinder test case, we focus exclusively on a *well-resolved* regime. However, it is important to highlight that standard **ROMs** may still lead to unstable dynamics due to

Table 6.1: *Flow past a cylinder test case.* Hyperparameters for the StabOp-L-ROMs (linear, quadratic, and fully nonlinear). Columns labeled NN correspond to hyperparameters specific to the neural-network model, namely the activation function.

r_u	Retained energy	Time window			h (NN)	λ_{lr}	N_{epochs}	ω_{θ}
		Train	Val	Test				
4	0.9952							optimized in $\mathbb{P}_{\omega} = \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}$
6	0.9987	[500, 520]	[520, 540]	[540, 600]	ReLU	0.2	100	
8	0.9993							
10	0.9997							

numerical instabilities, even in well-resolved regimes (see Table 5.1). Table 6.1 summarizes the different ROM regimes considered, and the hyperparameters used for L-ROM and StabOp-L-ROM.

Figures 6.1 and 6.2 display the time behaviour of the kinetic energy for all the ROMs considered, and the corresponding accuracy, in terms of MSE with respect to the FOM counterpart. G-ROM and L-ROM are characterized by unstable energy behaviour in time, at all r_u values except for $r_u = 10$, where the G-ROM has still an increasing energy, while the L-ROM is more stable and closely follows the FOM reference. On the other hand, StabOp-L-ROM provides enhanced stability over time. More in detail, the NN StabOp model has a higher accuracy than the linear and the quadratic models at $r_u = 4$, while it is working similar to the quadratic one at $r_u = 6$. The linear model deviates from the FOM kinetic energy in testing times at $r_u = 6$, underestimating it after $t = 540$. All StabOp-L-ROMs (linear, quadratic and nonlinear) perform similarly at $r_u = 8$ and $r_u = 10$, outperform the standard L-ROM, and accurately reconstruct the FOM kinetic energy qualitative behaviour. This is likely due to the larger number of parameters, and subsequently, the larger expressivity, associated with the data-driven models. It is however important to highlight that the nonlinear model has a higher accuracy than the linear and quadratic models in terms of MSE of the kinetic energy (Figure 6.2).

This is also confirmed by analysing the energy spectrum in Figure 6.3, for all the considered methodologies, at $r_u = 4$ at the latest time instance, i.e., $t = 600$. Since the presence of the cylinder prevents the direct use of FFT-based techniques, the spectrum is computed over the downstream subregion $[1, 17] \times [-5, 5] \subset \Omega$. The StabOp-L-ROM with nonlinear modeling is closer to the FOM energy spectrum, both at the large scales (small wavenumbers), which capture most of the system's energy, and at the small scales (large wavenumbers), which are the most difficult to accurately capture.

The last analysis in Figure 6.4 deals with the time percentage each model is stable and accurate. The most accurate and stable model at all r_u is the StabOp-L-ROM with nonlinear modeling. As previously described, L-ROM is mostly stable only in the case $r_u = 10$. The linear and quadratic StabOp-L-ROMs are mostly stable for larger values of

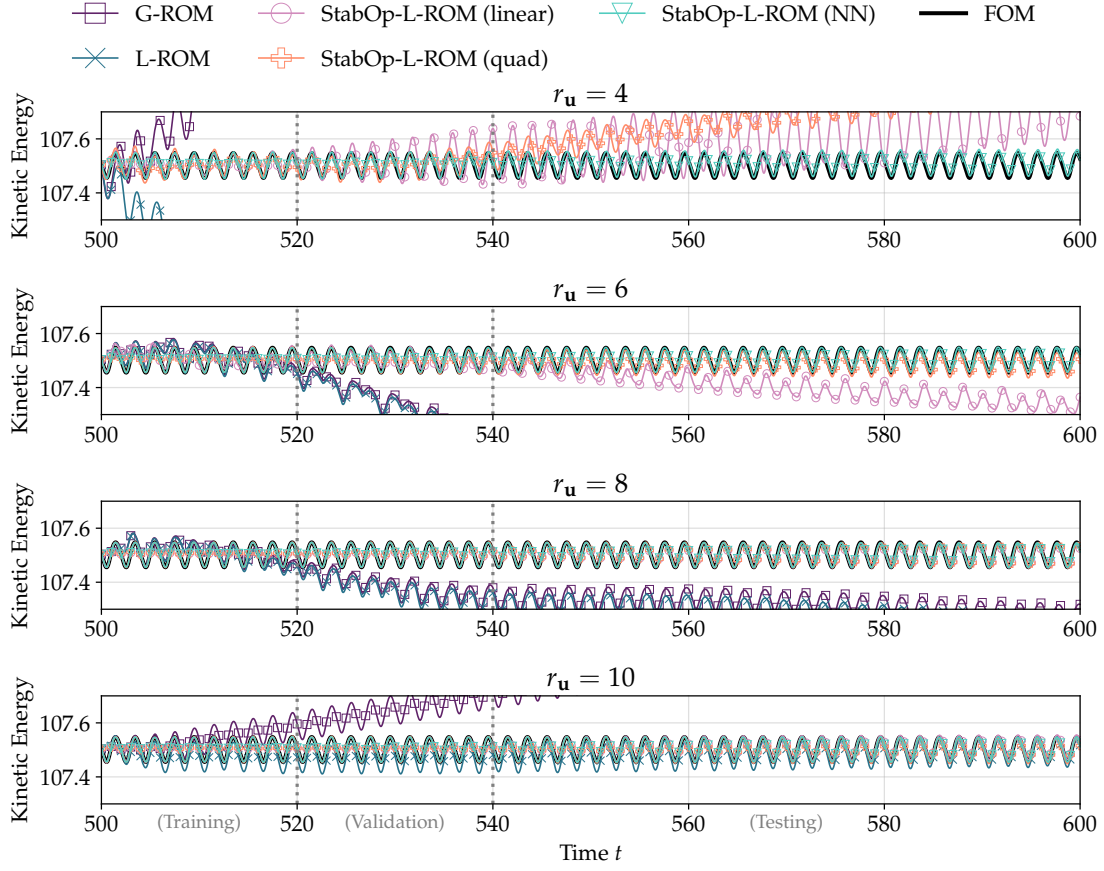


Figure 6.1: *Flow past a cylinder test case*. Kinetic energy variation in time for the FOM reference, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

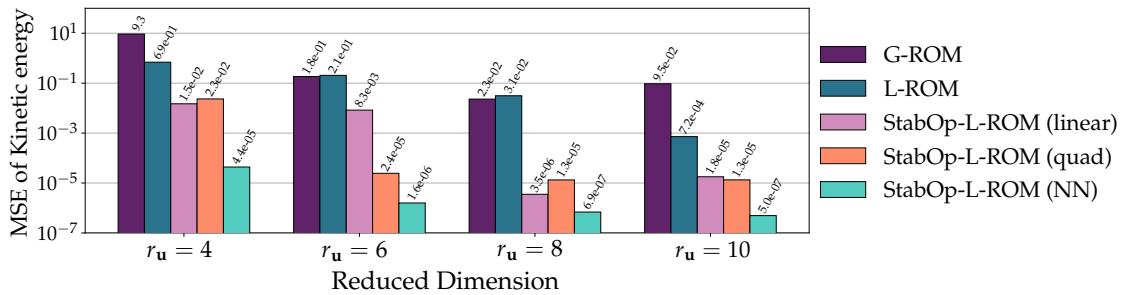


Figure 6.2: *Flow past a cylinder test case*. Mean squared error of kinetic energy with respect to FOM, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

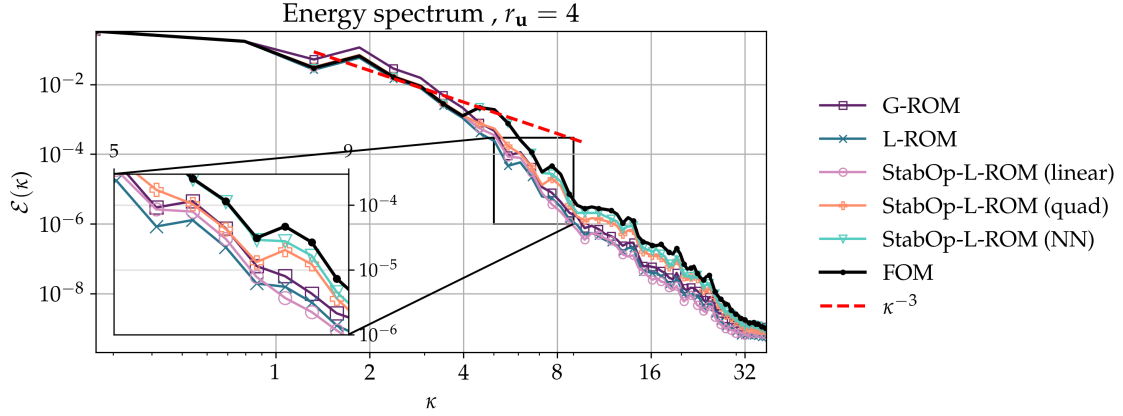


Figure 6.3: *Flow past a cylinder test case*. Energy spectrum at latest time for the reference FOM, standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

r_u , but, as showed in the energy analysis, are either overestimating or underestimating the energy content at the smallest r_u in time extrapolation regimes. On the other hand, **G-ROM** and **L-ROM** are mostly unstable, except for the $r_u = 10$ case, where the **L-ROM** is stable.

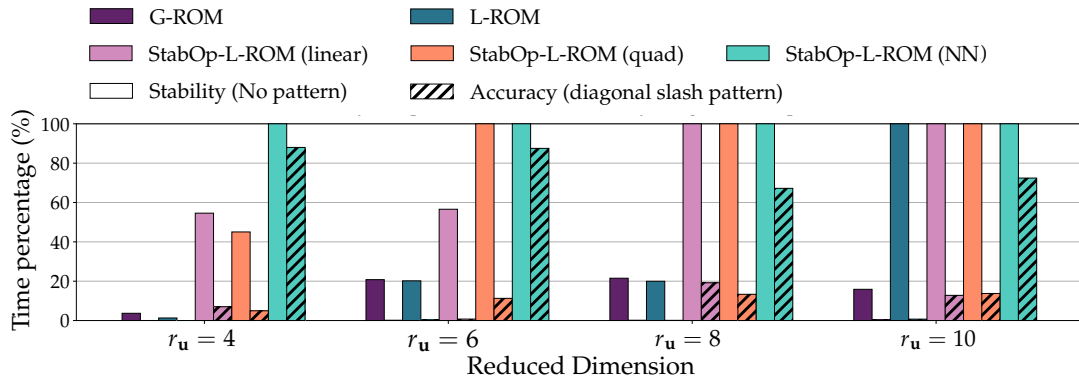


Figure 6.4: *Flow past a cylinder test case*. Stability and accuracy measures for standard G-ROM, L-ROM, and StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

6.3.2 Lid-driven cavity test case

The second test case is the unsteady lid-driven cavity problem at $Re = 10000$ subject to the boundary and initial conditions [162]:

$$\begin{cases} \mathbf{u} = [(1 - x^2)^2, 0]^T & \text{on } \Gamma_{\text{top}} \times \mathbb{R}_+, \\ \mathbf{u} = \mathbf{0} & \text{on } \partial\Omega \setminus \Gamma_{\text{top}} \times \mathbb{R}_+, \\ \mathbf{u} = \mathbf{0} & \text{on } \Omega \times \{0\}, \end{cases} \quad (6.16)$$

where $\Omega = [-1, 1]^2$ and $\Gamma_{\text{top}} = \{\mathbf{x} \in \partial\Omega : y = 1\}$. The FOM simulation is carried out using Nek5000 [93] with 1024 spectral elements of polynomial order $N = 7$. We focus on the time interval [5500, 6300], after the solution reaches a statistically steady state.

The ROM basis functions $\{\phi_i\}_{i=1}^{r_u}$ are constructed via POD using $N_{\text{train}} = 2001$ snapshots from the time interval [5500, 5900], corresponding to a sampling frequency of 0.2. The zeroth mode ϕ_0 is defined to be the average velocity over the training time window, and the ROM initial condition is obtained by projecting the lifted snapshot at $t = 5500$ onto the ROM space. We consider four reduced dimensions, $r_u = 5, 10, 16, 42$, whose regimes are detailed in Table 6.2, along with the hyperparameters of the StabOp models.

Table 6.2: *Lid-driven cavity test case*. Hyperparameters for the StabOp-L-ROMs (linear, quadratic, and fully nonlinear). Columns labeled NN correspond to hyperparameters specific to the neural-network model, namely the activation function.

r_u	Retained energy	Time window			h (NN)	λ_{lr}	N_{epochs}	ω_{θ}	δ
		Train	Val	Test					
5	0.8				Tanh				
10	0.9	[5500, 5900]	[5900, 6100]	[6100, 6300]	Tanh	0.2	200	10^{-8}	optimized in $\mathbb{P}_{\delta} \subset [0.001, 0.4]$
16	0.95				ReLU				
42	0.99				ReLU				

To improve numerical stability and accelerate the optimization process, the loss values are transformed using a logarithmic function, as they are typically small in magnitude. For the linear and quadratic models, the matrix $\tilde{\mathbf{A}}$ is initialized using the differential filter associated with a given δ , while the bias term $\tilde{\mathbf{c}}$ is omitted, as preliminary tests indicated that including it degraded the model performance. For the nonlinear model form, the differential filter is incorporated directly into the network architecture, as specified in Section 6.2. As indicated in the last column of Table 6.2, the filter radius is treated as an additional hyperparameter in all StabOp training procedures, and the search space $\mathbb{P}_{\delta}$ consists of 38 logarithmically spaced values between 0.001 and 0.4.

Figures 6.5 and 6.6 show the time evolution of the kinetic energy, and the MSE of the kinetic energy with respect to the FOM, for all the ROMs. The G-ROM is characterized by unstable behaviour, with increasing or highly oscillating energy at all the r_u values. Conversely, the L-ROM has a more stable behaviour in time, as also confirmed by the stability analysis of Figure 6.8. Even if it is mostly stable in time, the approximated L-ROM energy exhibits small oscillations with mostly time-constant and overdiffusive solution. For this reason, the time percentage the L-ROM is most accurate is low ($\sim 5 - 20\%$). On the other hand, the StabOp-L-ROMs are mostly accurate and stable, with similar performances across the linear, quadratic, and nonlinear models. Differently from the cylinder test case, a more expressive StabOp model does not lead to more accurate energy reconstruction. This is likely due to the complexity of the test case and the training data. Furthermore, the performance is largely determined by the outcome of the optimization

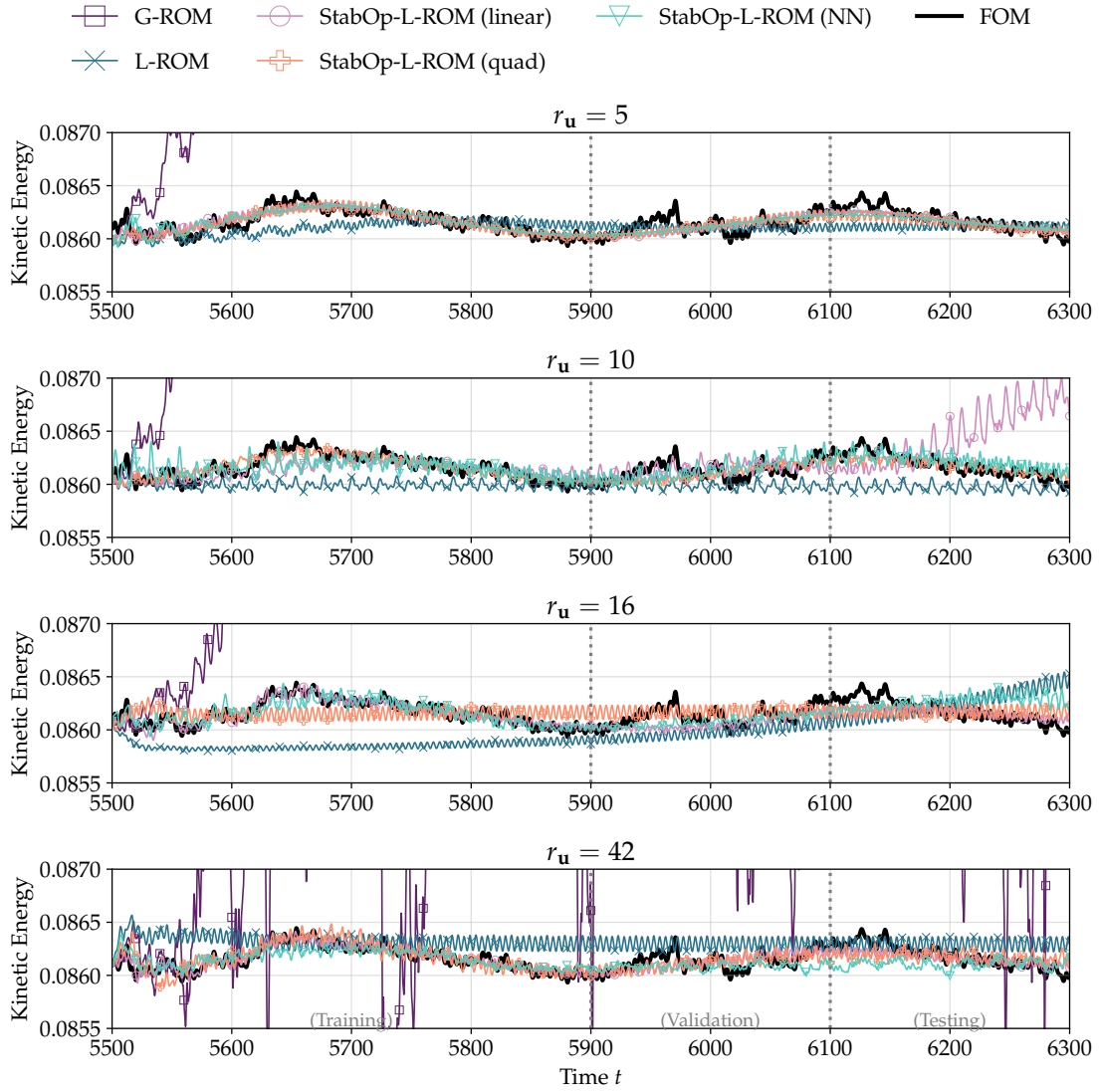


Figure 6.5: *Lid-driven cavity* test case. Kinetic energy variation in time for the FOM reference, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

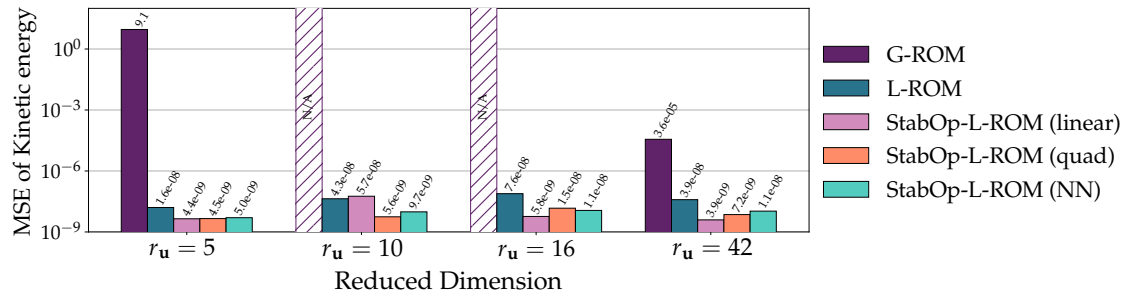


Figure 6.6: *Lid-driven cavity* test case. Mean squared error of kinetic energy with respect to FOM, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

process, which becomes more difficult to navigate as the StabOp model becomes more sophisticated.

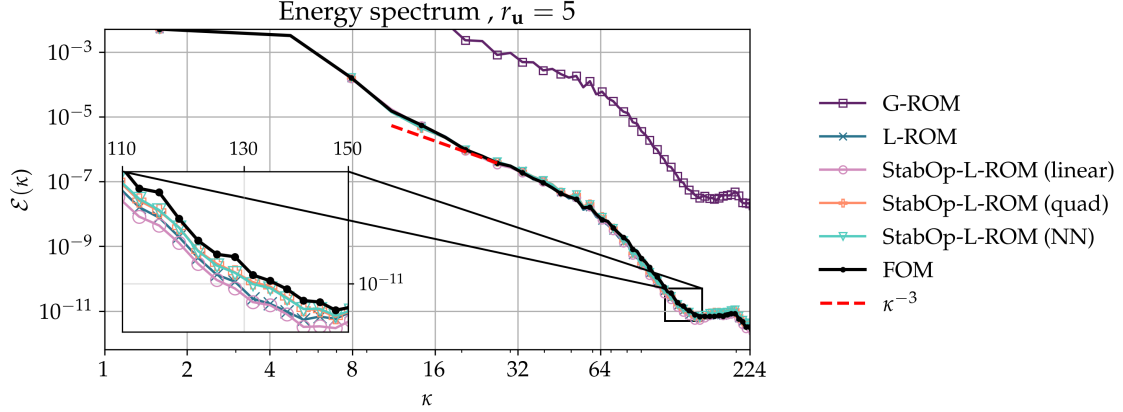


Figure 6.7: *Lid-driven cavity test case*. Energy spectrum at latest time for the reference FOM, standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

The analysis of the energy spectrum in Figure 6.7 confirms that the **G-ROM** overestimates the energy content at all scales, while at the latest time instance the **L-ROM** performs similar to the **StabOp-L-ROMs**. More in detail, at that particular simulation time and at $r_u = 5$, the **StabOp-L-ROM** with quadratic and nonlinear models provide the most accurate results, while the linear **StabOp-L-ROM** performs similar to the standard **L-ROM**.

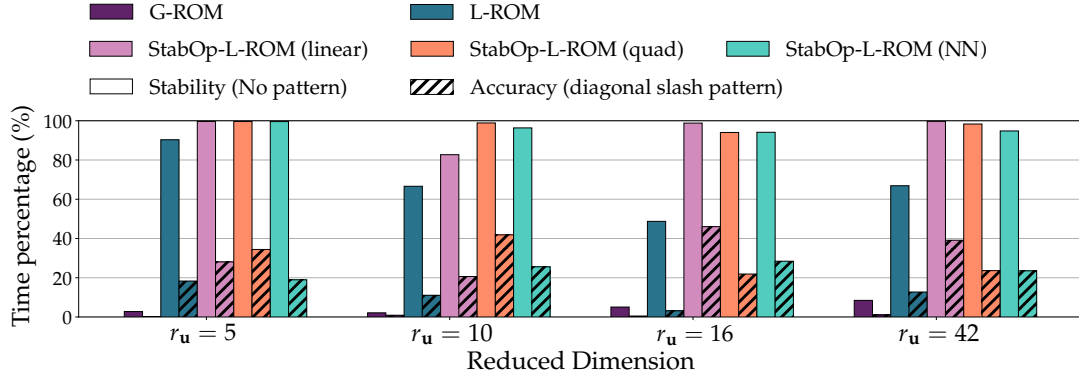


Figure 6.8: *Lid-driven cavity test case*. Stability and accuracy measures for standard G-ROM, L-ROM, and StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

6.3.3 Flow past a hemisphere test case

We next consider the 3D flow past a hemisphere at $Re = 2200$ [338]. The computational domain is a rectangular channel with depth 3.25 (along the z -axis), width 6.4 (along the y -axis), and length 18.2 (along the x -axis). The hemisphere, with radius 0.5, is mounted on a thin cylindrical base of radius 0.5 and height 0.05, which is placed on a smooth flat plate at $z = 0$. Inflow and outflow boundaries are imposed in the x -direction. At the inlet,

a Dirichlet boundary condition is prescribed using a smooth, sine-based inflow profile that approximates a boundary layer structure:

$$u(z) = \begin{cases} \sin\left(\frac{\pi z}{1.2}\right), & z \leq 0.6, \\ 1, & z > 0.6. \end{cases} \quad (6.17)$$

Here, 0.6 is the boundary layer thickness. This inflow condition is not the exact Blasius solution but serves as a smooth analytic approximation that captures the key transition from zero velocity at the wall to the free-stream value away from the wall.

At the outlet, a homogeneous Neumann boundary condition is imposed for the velocity. Symmetry boundary conditions are applied on the lateral boundaries and at the top wall ($z = 3.25$), enforcing zero normal velocity and zero tangential shear stress. A no-slip boundary condition is imposed at the bottom wall ($z = 0$).

The **FOM** simulation is carried out using nekRS [92] with 2042 spectral elements of polynomial order $N = 9$, resulting in about 2×10^6 degrees of freedom. We focus on the time interval $[3000, 3200]$, after the solution reaches a statistically steady state.

The reduced basis functions are constructed via **POD** from $N_{\text{train}} = 2001$ snapshots collected over the time interval $[3000, 3100]$, corresponding to a sampling frequency of 0.05. The zeroth mode (i.e., the lifting function) is defined as the velocity at $t = 3000$, and the **ROM** initial condition is obtained by projecting the lifted snapshot at $t = 3000$ onto the reduced space. We consider three reduced space dimensions, $r_u = 7, 12, 20$, corresponding to different energy contents, as detailed in Table 6.3. The table also highlights the hyperparameters used to train the StabOp models. In particular, the matrix $\tilde{\mathbf{A}}$ in the lin-

Table 6.3: *Flow past a hemisphere test case*. Hyperparameters for the StabOp-L-ROMs (linear, quadratic, and fully nonlinear). Columns labeled NN correspond to hyperparameters specific to the neural-network model, namely the activation function.

r_u	Retained energy	Time window			h (NN)	λ_{lr}	N_{epochs}	ω_{θ}	δ
		Train	Val	Test					
7	0.7								
12	0.75	[3000, 3100]	[3100, 3150]	[3150, 3200]	SiLU	0.2	200	10^{-8}	optimized in $\mathbb{P}_{\delta} \subset [0.001, 0.1]$
20	0.8								

ear and quadratic forms is initialized using the differential filter associated with a given δ , while in the nonlinear StabOp, the **DF** is directly added to the **NN** output and considered during training. Hence, the filter radius of the differential filter is treated as an additional hyperparameter, as shown in Table 6.3, and is optimized among 10 logarithmically spaced values between 0.001 and 0.1.

As done in the previous test cases, Figures 6.9 and 6.10 describe the time evolution and the global **MSE** of the kinetic energy with respect to the **FOM**, at the r_u values taken into account. As in the previous cases, the **G-ROM** is unstable and its energy blows up after a

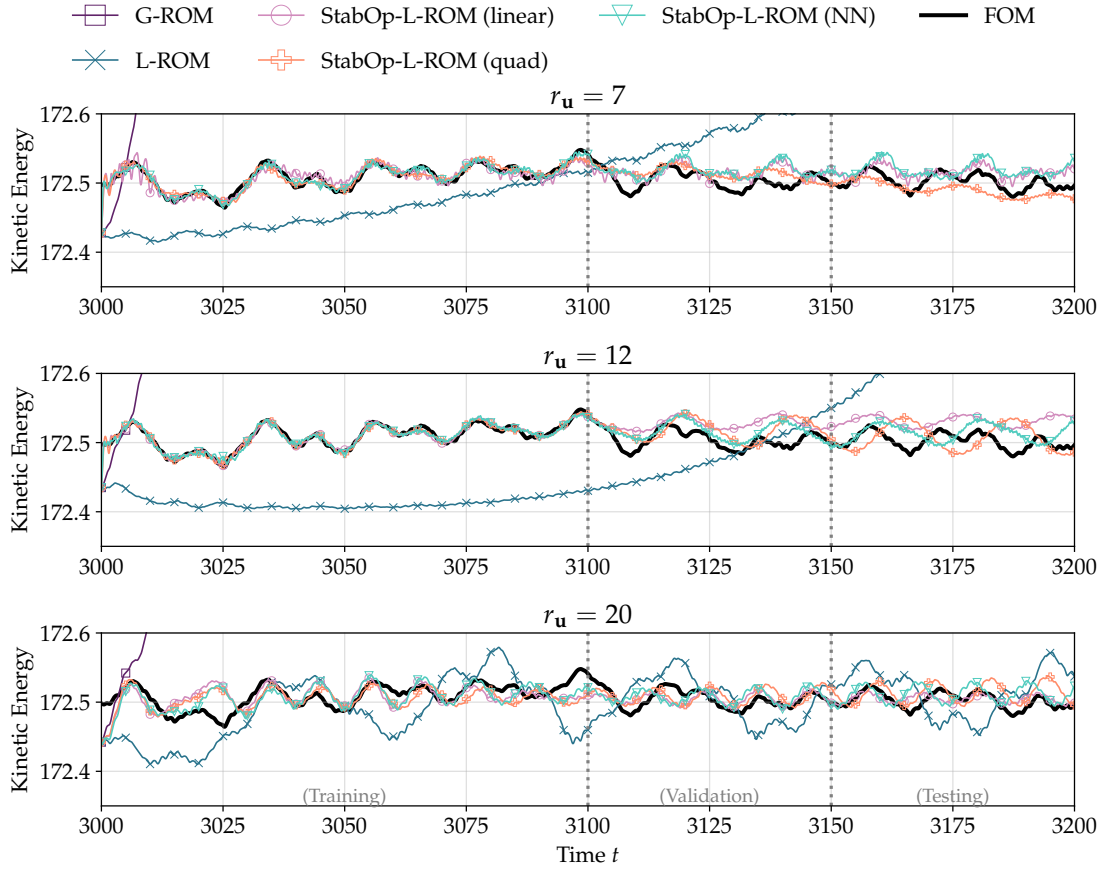


Figure 6.9: *Flow past a hemisphere test case*. Kinetic energy variation in time for the FOM reference, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

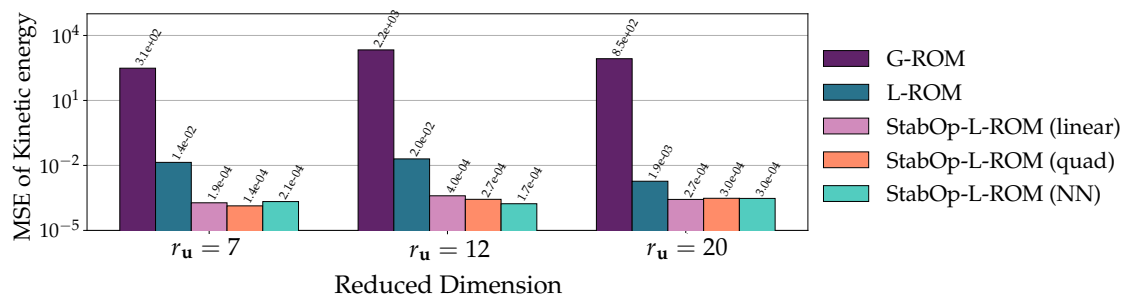


Figure 6.10: *Flow past a hemisphere test case*. Mean squared error of kinetic energy with respect to FOM, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

few time steps. The **L-ROM** is more stable than the **G-ROM** but it is not able to capture the flow dynamics, especially at $r_u = 7, 12$. The stability and accuracy analysis of Figure 6.11 confirms that the **L-ROM** is mostly unstable at $r_u = 7, 12$, while it is stable $\sim 50\%$ of the time at $r_u = 20$. The **StabOp-L-ROMs** are instead mostly stable and accurate, and they are all characterized by similar performance. At all latent dimensions, the **StabOp-L-ROMs** capture the kinetic energy time evolution in training times, while becoming slightly less accurate in validation and test times.

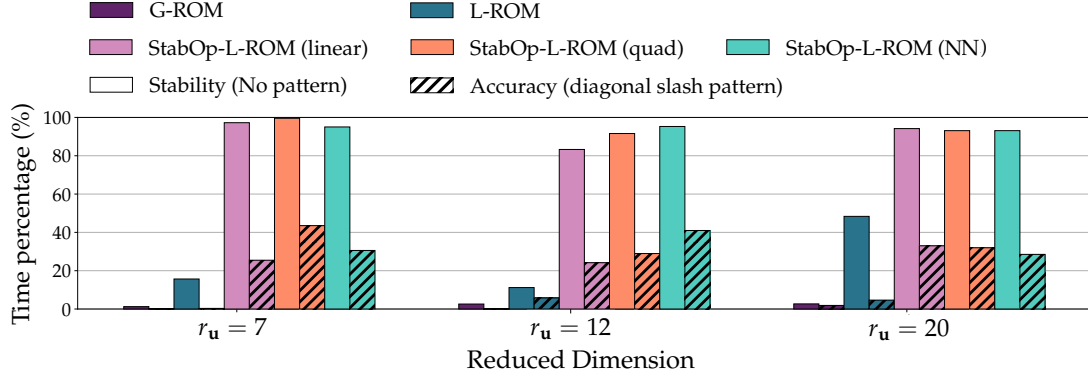


Figure 6.11: *Flow past a hemisphere test case*. Stability and accuracy measures for standard G-ROM, L-ROM, and StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

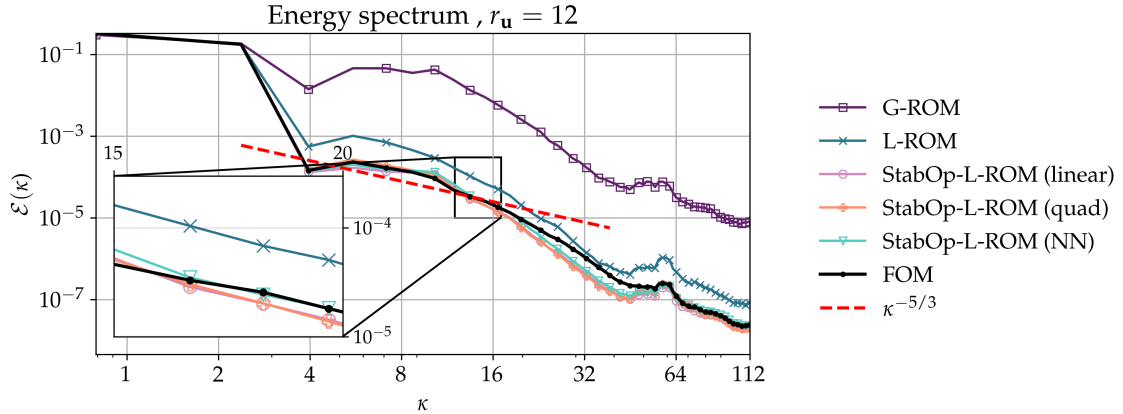


Figure 6.12: *Flow past a hemisphere test case*. Energy spectrum at latest time for the reference FOM, standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

The analysis of the energy spectrum $\mathcal{E}(\kappa)$ in Figure 6.12 at the latest time instance, highlights that the **G-ROM** and the **L-ROM** exhibit unstable and noisy behaviour, as they overestimate the energy content at all wavenumbers. On the other hand, all the **StabOp-L-ROMs** provide a good approximation of the **FOM** energy spectrum. The nonlinear StabOp outperforms the simpler linear and quadratic models at the medium scales, where it well catches the **FOM** energy content. We specify that the spectrum is computed in the region downstream the hemisphere, namely $[2, 15] \times [-3.2, 3.2] \times [0, 3.25]$.

6.3.4 Minimal channel flow test case

We consider the minimal channel flow at $Re = 5000$, which presents strong turbulent features while maintaining simplified flow dynamics, resulting in significantly lower computational costs compared to a full channel flow simulation. Following the setup in [154], the streamwise and spanwise lengths of the channel are set to 0.6π and 0.18π , respectively, and the channel half-height is set to 1.

The FOM simulation is carried out using nekRS [92] with 576 spectral elements of polynomial order $N = 9$, resulting in about 5.76×10^5 degrees of freedom. We focus on the time interval [3000, 4000], after the solution reaches a statistically steady state. This represents a challenging test problem for reduced order modeling, since the G-ROM requires $r_u \geq 400$ modes to capture the kinetic energy of the FOM [334].

The ROM basis functions are constructed via POD from $N_{\text{train}} = 2001$ snapshots collected over the time interval [3000, 3500], corresponding to a sampling frequency of 0.25. The zeroth mode is set to the mean velocity over the training time interval, and the ROM initial condition is obtained by projecting the lifted snapshot at $t = 3000$ onto the ROM space. We consider three latent dimensions, $r_u = 6, 15, 30$, corresponding to smaller energy content than the previous test cases. As already highlighted, this test case is the most challenging one, and more modes would be needed to consider a well-resolved ROM. The energy percentage and the hyperparameters of the StabOp trainings are detailed in Table 6.4. To improve numerical stability and sensitivity during optimization, the loss values are transformed using a logarithmic function, as they are typically small in magnitude.

Table 6.4: *Minimal channel flow test case.* Hyperparameters for the StabOp-L-ROMs (linear, quadratic, and fully nonlinear). Columns labeled linear, quad, and NN correspond to hyperparameters specific to the linear, quadratic or neural-network model, respectively.

r_u	Retained energy	Time window			h (NN)	λ_{lr}	N_{epochs}	ω_θ	δ (linear, quad)
		Train	Val	Test					
6	0.2	[3000, 3500]	[3500, 3750]	[3750, 4000]	SiLU	0.2	200	10^{-8}	optimized in $\mathbb{P}_\delta \subset [0.08, 0.3]$
15	0.3								optimized in $\mathbb{P}_\delta \subset [0.03, 0.2]$
30	0.4								

The same initialization strategies of the flow past a hemisphere are here applied.

The complexity of the test case can be seen from the analysis of the kinetic energy behaviour of Figures 6.13 and 6.14. The time evolution of the kinetic energy of Figure 6.13 highlights the inaccuracy of the G-ROM, which becomes unstable at initial times, and the L-ROM, which is either underestimating the energy behaviour (overdiffusive) or showing more oscillations than the FOM. However, L-ROM performs better and is more stable than in the previous test cases. This is also confirmed by the stability and accuracy analysis of Figure 6.16, e.g., at $r_u = 15$ the L-ROM is stable the $\sim 80\%$ of time. The StabOp-L-ROMs outperform the standard L-ROM by capturing the overall time trend, even if none of the

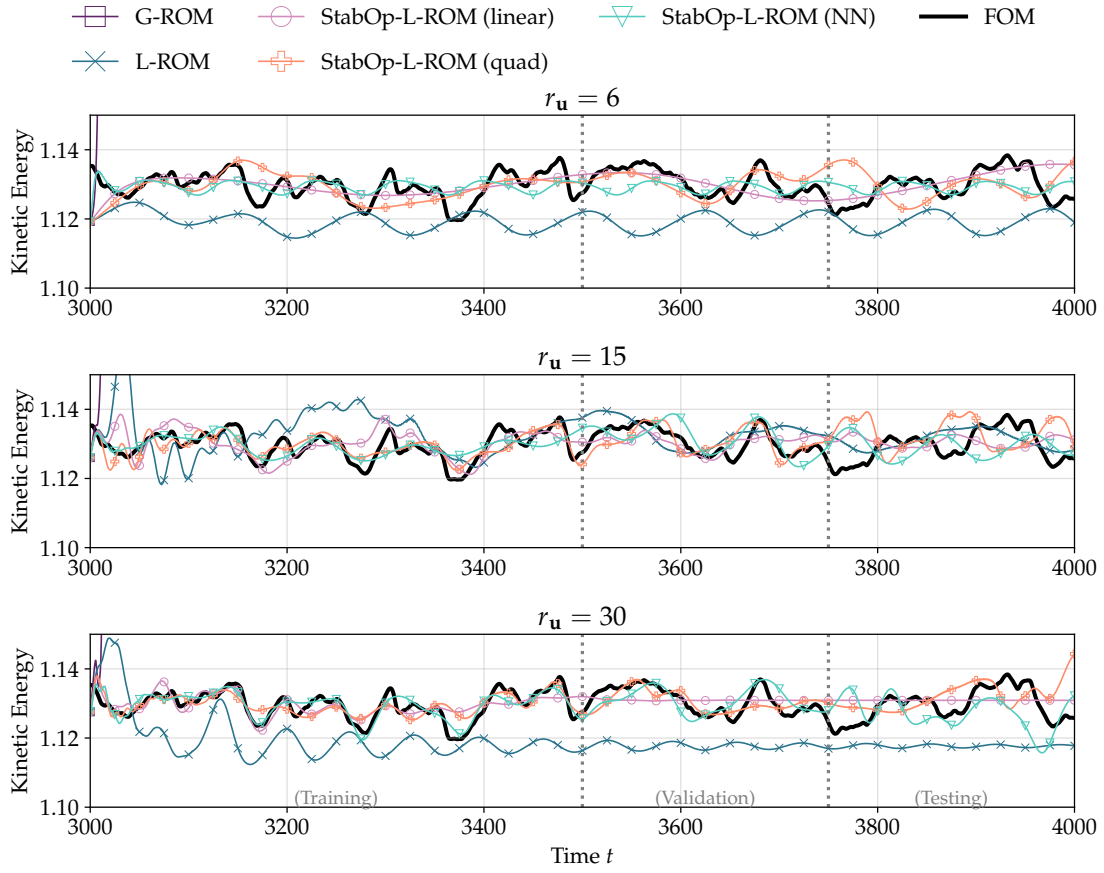


Figure 6.13: *Minimal channel flow test case*. Kinetic energy variation in time for the FOM reference, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

models is able to accurately recover the FOM instant behaviour. The complexity of the test case, and, hence, of the snapshots' data used for training the StabOp models, leads to similar results for the linear, quadratic and nonlinear models (see the MSE of StabOp-L-ROMs in Figure 6.14). We also specify that the final accuracy largely depends on the optimum reached at the end of the optimization process.

The energy spectrum in Figure 6.15 at $r_u = 6$ highlights the fact that all models are still far from the FOM in the energy content, in particular at the medium and large wavenumbers (medium-small scales). However, the StabOp-L-ROM with quadratic model is the most accurate approach in the energy spectrum, measured in the entire domain at the latest time instance.

The numerical investigation can be globally summarized as follows.

- The StabOp-L-ROMs improve the L-ROM in terms of both accuracy and stability both in simple 2D and in more complex 3D test cases. Specifically, the PDE-constrained optimization process ensures the stability of the models over time, also in extrapolation regimes.

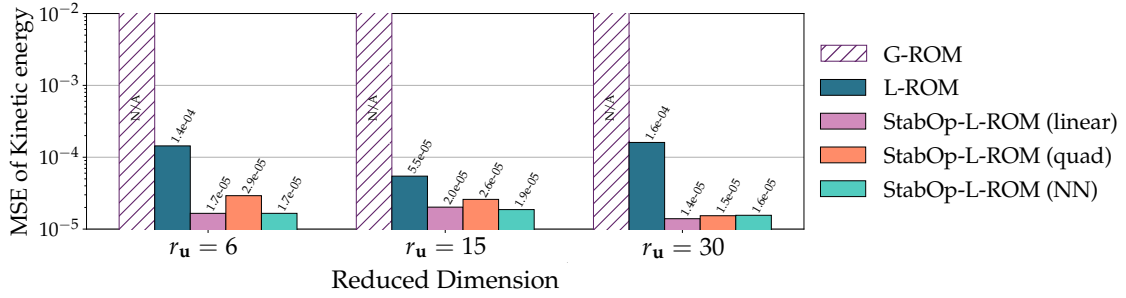


Figure 6.14: *Minimal channel flow test case.* Mean squared error of kinetic energy with respect to FOM, for the standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

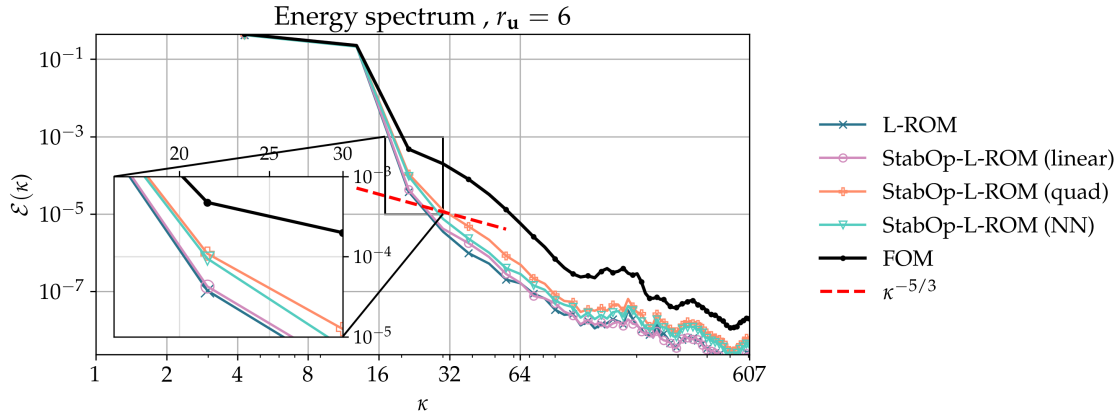


Figure 6.15: *Minimal channel flow test case.* Energy spectrum at latest time for the reference FOM, standard G-ROM, L-ROM, and for the StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

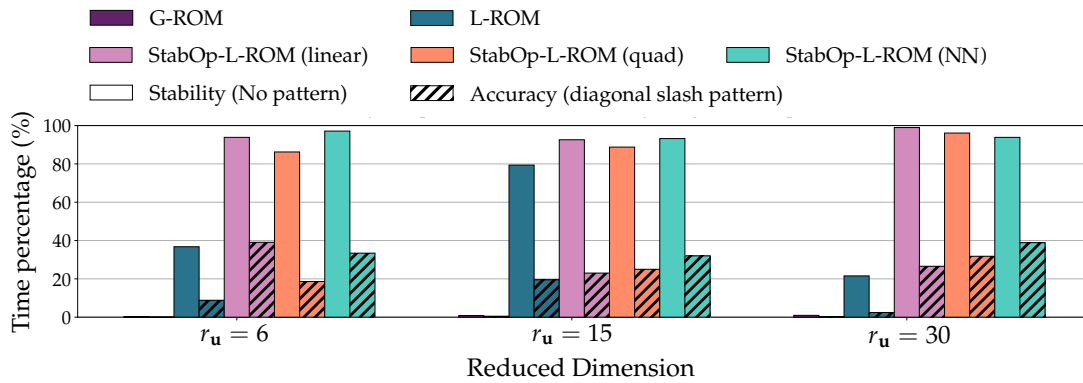


Figure 6.16: *Minimal channel flow test case.* Stability and accuracy measures for standard G-ROM, L-ROM, and StabOp-L-ROM methods (with linear, quadratic or nonlinear modeling).

- The results indicate that the StabOp model expressiveness alone is not sufficient to guarantee improved performance. While the nonlinear, quadratic, and linear formulations exhibit progressively better accuracy in the flow past a cylinder test case, this behaviour is not consistently observed for more complex flows. In such settings, the increased difficulty of the optimization problem can limit the benefits of more sophisticated model forms, making the final performance strongly dependent on the outcome of the training process.

6.3.5 Computational cost

Figure 6.17 summarizes the computational costs of the considered approaches. The left panel reports the average offline training time, while the right panel shows the average online simulation time over the target interval for each test case. For the **L-ROM**, the offline timings are averaged over all reduced dimensions r_u and include the cumulative cost of exploring the filter-radius hyperparameter space $\mathbb{P}_\delta$ to identify the optimal value of δ . For the **StabOp-L-ROMs**, the reported timings are also averaged over all reduced dimensions r_u , but correspond to a single representative choice of hyperparameters. Since the filter radius is used only to initialize the linear StabOp parameters and the operator itself is subsequently learned through PDE-constrained optimization, the reported cost corresponds to a single representative value of δ .

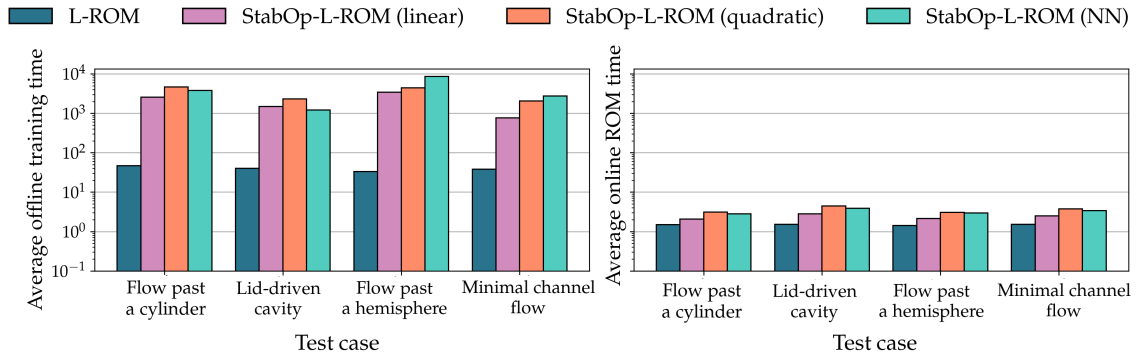


Figure 6.17: CPU time of performing the offline training (left) and online simulation (right) of G-ROM, L-ROM, and all StabOp-L-ROMs.

As expected, the offline cost of the **StabOp-L-ROMs** exceeds that of the **L-ROM** because of the additional PDE-constrained optimization required to construct the StabOp. This cost could be further reduced through roll-out training strategies, in which the optimization is performed over short temporal horizons instead of the entire training interval. Such approaches avoid the construction of long computational graphs and enable the use of first-order stochastic optimization methods, such as Adam, thereby reducing both memory requirements and wall-clock time. Although this strategy sacrifices global-in-time gradient information, it has proven effective for training dynamical-system models and represents a promising avenue for accelerating the construction of the StabOp [168].

The online results show that, despite the higher offline investment, the **StabOp-L-ROMs** incur only a modest increase in simulation cost compared with the **L-ROM**, while remaining computationally inexpensive across all test cases. Direct speedup comparisons with the **FOM** are not reported because the **FOM** simulations are performed in parallel and on GPUs, whereas the ROM construction and training procedures are executed serially on CPUs. Nevertheless, we provide the **FOM** runtimes to quantify the cost of generating the training snapshots.

The **FOM** simulations are performed using Nek5000 [93] for the 2D test cases and NekRS [92] for the 3D test cases. The 2D simulations are executed on 8 MPI ranks and require approximately 339 seconds for the 2D flow past a cylinder and 4399 seconds for the 2D lid-driven cavity. The 3D simulations are carried out on a single NVIDIA TITAN V GPU using NekRS, requiring approximately 1208 seconds for the 3D flow past a hemisphere and 4907 seconds for the 3D minimal channel flow. All **ROMs** are implemented using [161]. The StabOp training procedure has been performed using PyTorch [249] and then integrated into the **L-ROM** solver within [161].

6.4 Is StabOp a spatial filter?

While the previous section numerically investigates the performance of **StabOp-L-ROM** compared with the literature methods, this section answers the question:

Is StabOp a spatial filter?

As described in Section 6.2, we build StabOp as a data-driven linear, quadratic, or nonlinear operator, whose goal is to accurately capture the time evolution of the reference **FOM** energy. However, we do not explicitly mathematically define it as a spatial filter. A spatial filter, like the **DF** [107, 108], as introduced in Chapter 1, aims to inject dissipation, and, therefore, it acts as a smoothing operator. At the reduced order level, the action of a **ROM** spatial filter is the reduction of the magnitude of the **ROM** coefficients, namely at each time step:

$$\bar{a}_i = \mathcal{F}_r(a_i) \leq a_i \quad \forall i = 1, \dots, r_u.$$

We analyse here the effect of the standard **DF** at $\delta = \eta$ (the Kolmogorov length scale), of the projection considering the 20% and the 40% of the modes retained, and of the StabOp with linear, quadratic and nonlinear operators.

The filter effect is analysed by considering an unstable **G-ROM** solution at fixed time and then applying the filter on it (for a single time step). We afterwards analyse both how the qualitative solution has changed, and how the filtered coefficients have been modified by the action of the filter. We repeated this analysis for different r_u values, and for the 2D lid-driven cavity test case (Figures 6.18 and 6.19), the 3D flow past a hemisphere (Figures 6.20 and 6.21), and the 3D minimal channel flow (Figures 6.22 and 6.23).

For the lid-driven cavity test case, the graphical results confirm that the **DF** acts as a filter operator. The projection is also a filter, even if at $r_u = 16$, it still exhibits spurious oscillations. The solution obtained applying StabOp leads to different results depending

on the r_u value considered. At $r_u = 5$, none of the three StabOp versions produce a visibly smoother velocity field, while at $r_u = 16$ and 42 the qualitative fields appear smoother and close to the solution obtained using the DF at $\delta = 0.001$.

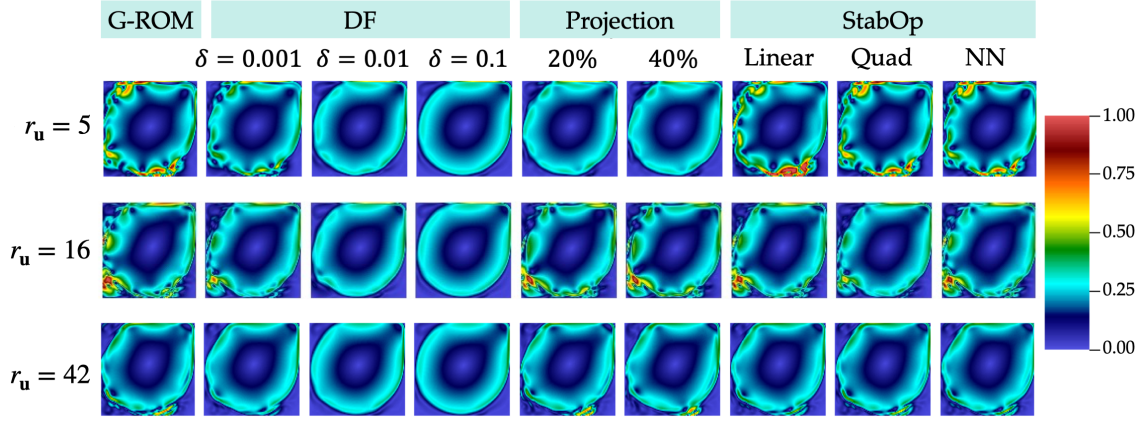


Figure 6.18: *Lid-driven cavity test case*. Velocity fields obtained from the standard G-ROM, and applying the DF, projection, and the StabOp operators to one G-ROM solution.

The quantitative analysis of Figure 6.19 confirms the qualitative results: at $r_u = 5$, the application of the linear StabOp leads to an amplification of the modal contributions rather than a damping; at $r_u = 16$ and 42, all StabOp models except for the linear one leads to damped reduced coefficients, while at $r_u = 42$ all StabOp models act as a filter (i.e., all coefficients have been damped by the StabOp action). At $r_u = 16$ the linear StabOp amplifies the contribution of modes 1, 5, 12 and 15.

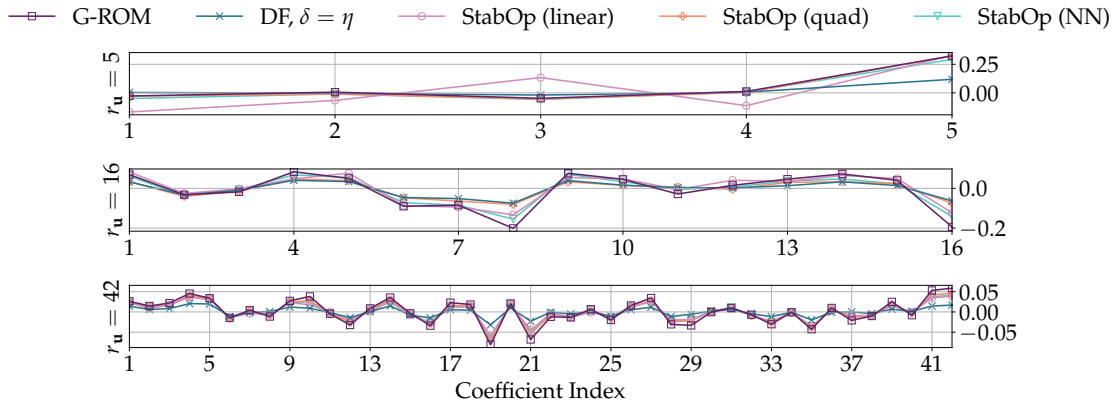


Figure 6.19: *Lid-driven cavity test case*. Reduced coefficients' values at different r_u , for the standard G-ROM, and after applying the DF and the StabOp operators to it.

In the 3D flow past a hemisphere test case (Figure 6.20) the StabOp models produce visibly smoother fields in the case of $r_u = 20$ but not in the case $r_u = 12$. The coefficient values of Figure 6.21 are indeed amplified after the application of all StabOp models at $r_u = 12$, while they are all damped at $r_u = 20$. In the latter case, the StabOp action is similar to the DF action at $\delta = \eta$, which corresponds to the standard literature choice.

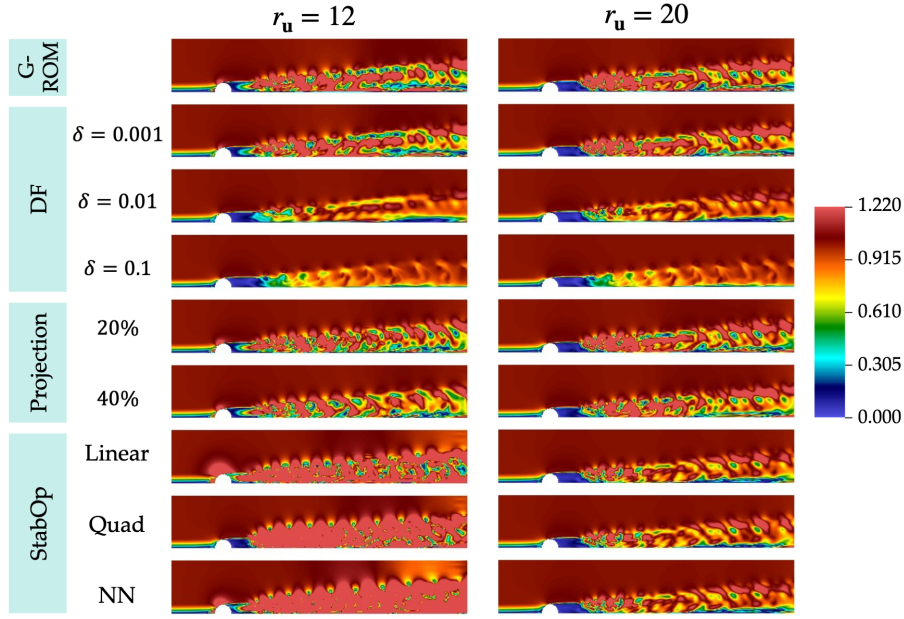


Figure 6.20: *Flow past a hemisphere test case.* Velocity fields obtained from the standard G-ROM, and applying the DF, projection, and the StabOp operators to one G-ROM solution.

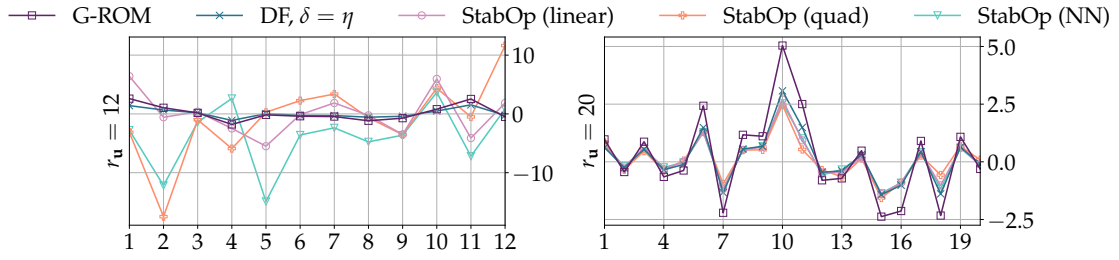


Figure 6.21: *Flow past a hemisphere test case.* Reduced coefficients' values at different r_u , for the standard G-ROM, and after applying the DF and the StabOp operators to it.

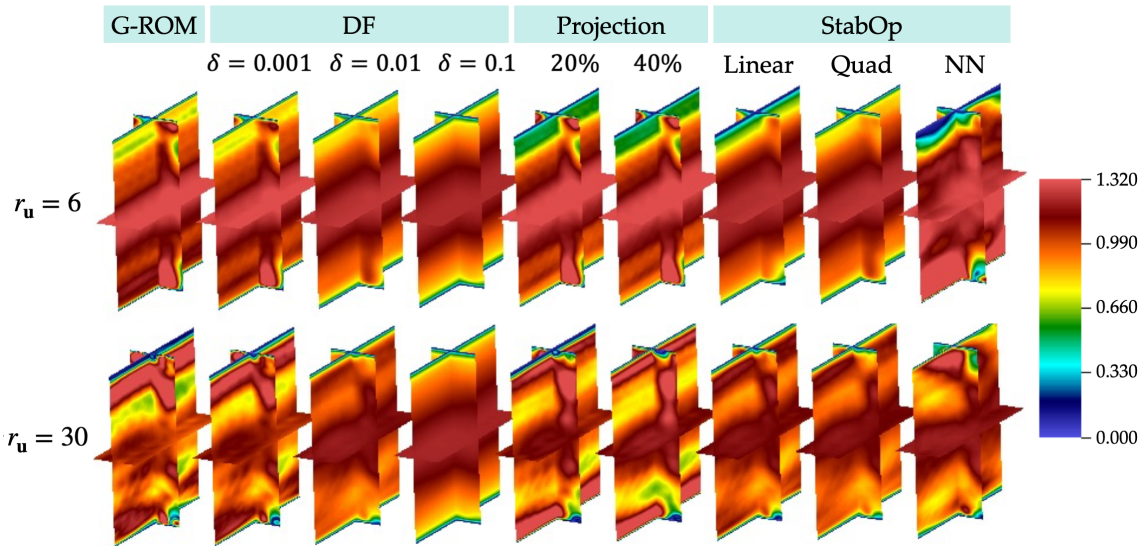


Figure 6.22: *Minimal channel flow test case.* Velocity fields obtained from the standard G-ROM, and applying the DF, projection, and the StabOp operators to one G-ROM solution.

In the 3D minimal channel flow test case, the velocity solutions of Figure 6.22 appear visibly smoother after the application of all StabOp models, except for the nonlinear one. This is also confirmed by the coefficients' values in Figure 6.23. At $r_u = 6$, the coefficients after the application of the nonlinear StabOp behave different from both the linear and quadratic StabOp and from the DF. This is even more visible at $r_u = 30$, where the nonlinear StabOp produces an amplification of the reduced coefficients associated with the modes of indices 12, 13, 16, 20, 21, and 25.

We can conclude that **StabOp does not act as a classical spatial filter**, as its effect changes both depending on the test case, the latent dimension, and on the data used to train the models.

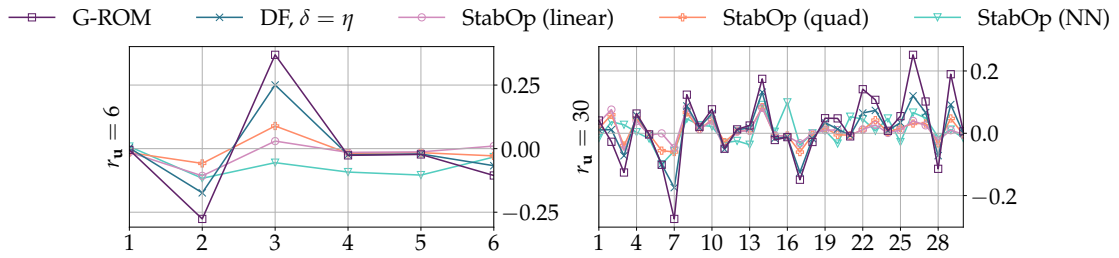


Figure 6.23: *Minimal channel flow test case*. Reduced coefficients' values at different r_u , for the standard G-ROM, and after applying the DF and the StabOp operators to it.

6.5 Conclusions and Motivation to the next chapter

In this chapter, we introduced the **StabOp** framework, a data-driven stabilization strategy that replaces the classical spatial filter used in filter-based **ROM** stabilizations with an operator learned directly from data through PDE-constrained optimization. Unlike traditional approaches, which rely on a prescribed filtering mechanism and the tuning of filter parameters, the proposed framework seeks a stabilization operator that is optimal for a given **ROM** (in our case, the **L-ROM**) and quantity of interest (in our case, the kinetic energy).

We investigated three types of ansatz for StabOp, namely linear, quadratic and nonlinear, and validated the StabOp effect on four convection-dominated flow problems demonstrated that the resulting **StabOp-L-ROM** consistently improves the stability and predictive accuracy of the **L-ROM**. In several cases, the improvement is substantial, yielding one or more orders of magnitude reduction in the prediction error.

Furthermore, the analysis showed that the learned StabOp does not behave as a classical **ROM** filter, suggesting that it represents a fundamentally different stabilization mechanism.

An important observation is that increasing the expressiveness of the StabOp model does not necessarily lead to improved performance. While nonlinear formulations can

outperform linear and quadratic counterparts in some settings, this hierarchy is not consistently observed across all test cases. For more complex flows, the quality of the solution is strongly influenced by the difficulty of the underlying PDE-constrained optimization problem, indicating that model expressiveness and optimization effectiveness must be considered jointly.

Overall, the results highlight the potential of learning stabilization operators directly from data and suggest that classical spatial filtering is not necessarily the optimal paradigm for constructing ROM stabilizations.

Future work should investigate the application of StabOp to other ROM stabilization and closure models, explore alternative training strategies to reduce computational costs.

In this context, while the StabOp framework focuses on learning stabilization mechanisms from data, it does not explicitly address the reconstruction of the unresolved dynamics. This distinction naturally leads to approaches that aim instead at modeling the missing contributions through explicit **correction** or **closure** terms. Such methods, commonly referred to as data-driven correction or closure strategies, are the focus of the following chapters (7 and 8), where neural network-based models are employed to enhance the predictive capability of ROMs by recovering the effects of truncated modes.

DD-EV-ROM: DATA-DRIVEN CLOSURES FOR POD-GALERKIN ROMs

The present chapter focuses on a data-driven stabilization approach for **ROMs**, developed in [145], whose goal is to enhance the reduced approximation by reintroducing the contribution of the neglected modes, as described in Section 5.4.2. More in detail, we investigate convection-dominated flow cases, which are simulated at the full-order level using a **RANS** approach, based on the eddy-viscosity modeling. Moreover, we employ a **PPE** regularization approach at the **ROM** level. Hence, we couple three different frameworks: the **PPE** strategy, the **EV-ROM**, which reintroduces the eddy viscosity into the reduced system (as described in Section 5.4.1), and the **correction** or **closure** modeling, which adds to the **ROM** an extra term accounting for the discarded modes (as described in Section 5.4.2). Previous works like [142] have already coupled turbulence and correction approaches in the **G-ROM** framework, but not in a parametrized setting, as mentioned in Section 5.5.

The novelties of this work are the following.

- ▶ The extension and generalization of the approach introduced in [142] to **parametrized flows** making use of machine learning approaches, and in particular of simplified **neural operators** architectures [210, 155]. The resulting **ROM** is called **Data-Driven Eddy-Viscosity Reduced Order Method (DD-EV-ROM)**.
- ▶ The use of a **coupled training procedure** for the turbulence and for the correction models. This leads to a modified version of **DD-EV-ROM**, aimed to further improve long-time **ROM** stability and accuracy.

Figure 7.1 sketches the pipeline followed in this work to build the **DD-EV-ROM**. We recall that:

- ▶ $\mathcal{G}$ represents the turbulence mapping, namely the map used to approximate the reduced eddy viscosity coefficients g , which is needed to close the reduced system in the **EV-ROM** approach.
- ▶ $\mathcal{M}$ is the correction map, used to model the correction terms. Since we adopt a **PPE** approach, in our case the correction term should include the contributions associated to both the momentum equation and the **PPE**, namely τ_u and τ_p .

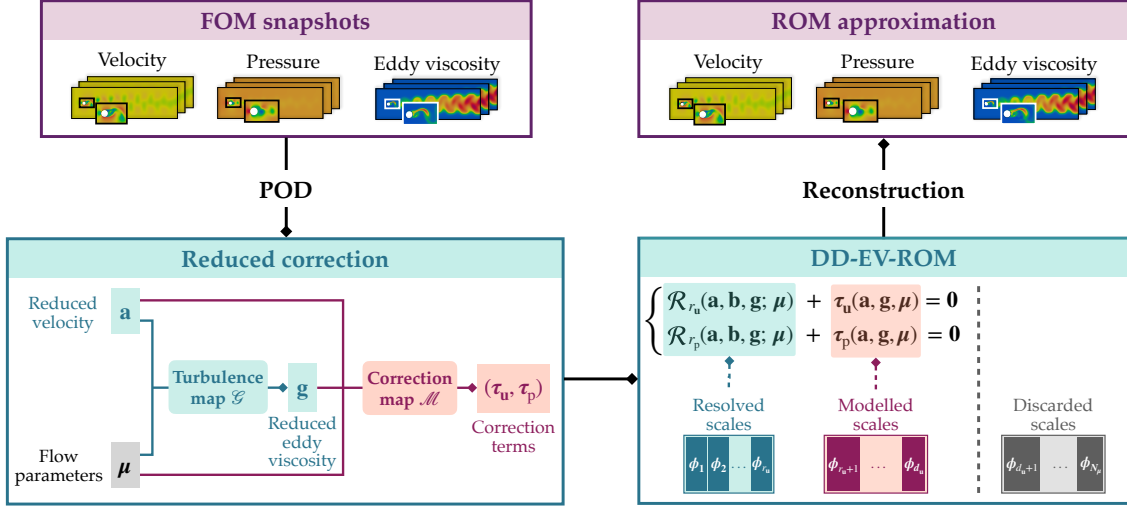


Figure 7.1: Sketch of the pipeline followed to build the DD-EV-ROM model.

Since the general **EV-ROM** using a **PPE** stabilization approach has already been introduced in Section 5.4.1, we refer the reader to it for a literature review about existing methodology.

This chapter is structured as follows:

- ▶ presentation of the **DD-EV-ROM** algorithm (Section 7.1);
- ▶ description of the architectures adopted for the turbulence and correction mappings $\mathcal{G}$ and $\mathcal{M}$ (Section 7.2);
- ▶ numerical investigation on three different test cases in the convection-dominated regime, and at different latent dimensions (Section 7.3);
- ▶ conclusions and motivations for the next chapter, which is a direct extension of the correction approach to a nonintrusive setting (Section 7.4).

7.1 DD-EV-ROM: the algorithm

Before proceeding with the presentation of the **DD-EV-ROM**, we recall that the literature baseline is the **EV-ROM** (Equation (5.29)), where a **PPE** stabilization approach is employed to ensure a good reconstruction of the pressure field. Moreover, we apply a penalty method to enforce the non-homogeneous velocity boundary conditions (see Section 5.1.1 for more details), obtaining the following formulation:

$$\begin{cases} -\mathbf{M}_{u,r}\dot{\mathbf{a}} - \mathbf{a}^T \mathbf{C}_{u,r} \mathbf{a} + \nu(\mathbf{B}_r + \mathbf{B}_r') \mathbf{a} + \mathbf{g}(\mathbf{T}_{u,r} + \mathbf{T}_{u,r}') \mathbf{a} - \mathbf{G}_r \mathbf{b} \\ \quad + \tau_{BC} \sum_{k=1}^{N_{BC}} (u_{BCk} \Phi_{BCk} - \mathbf{E}_{rk} \mathbf{a}) = \mathbf{0}, \\ \mathbf{D}_{p,r} \mathbf{b} + \mathbf{a}^T \mathbf{C}_{p,r} \mathbf{a} - \mathbf{g}(\mathbf{T}_{p,r} + \mathbf{T}_{p,r}') \mathbf{a} - \nu \mathbf{N}_r \mathbf{a} = \mathbf{0}, \end{cases} \quad (7.1)$$

where the reduced operators and the boundary terms are defined in (5.8), (5.15), (5.30), and (5.10).

The **EV-ROM** is a system of $r_u + r_p$ unknowns ($\mathbf{a}$ and $\mathbf{b}$), where at each online time step the reduced eddy viscosity is found by applying the turbulence map $\mathcal{G}$. The penalty coefficient τ_{BC} is optimized through a sensitivity analysis on each test case as in [131, 315]. Bigger values of the penalization factor lead to a stronger enforcement of the boundary conditions. At the same time, large values also lead to a larger condition number and, hence, to less stability.

The **DD-EV-ROM** system is instead a modification of system (7.1), with the addition of the additional data-driven corrections, namely:

$$\begin{cases} -\mathbf{M}_{ur}\dot{\mathbf{a}} - \mathbf{a}^T \mathbf{C}_{ur} \mathbf{a} + \nu(\mathbf{B}_r + \mathbf{B}'_r) \mathbf{a} + \mathbf{g}(\mathbf{T}_{ur} + \mathbf{T}'_{ur}) \mathbf{a} - \mathbf{G}_r \mathbf{b} \\ \quad + \tau_{BC} \sum_{k=1}^{N_{BC}} (u_{BCk} \Phi_{BCk} - \mathbf{E}_{rk} \mathbf{a}) + \boldsymbol{\tau}_u = \mathbf{0}, \\ \mathbf{D}_{pr} \mathbf{b} + \mathbf{a}^T \mathbf{C}_{pr} \mathbf{a} - \mathbf{g}(\mathbf{T}_{pr} + \mathbf{T}'_{pr}) \mathbf{a} - \nu \mathbf{N}_r \mathbf{a} + \boldsymbol{\tau}_p = \mathbf{0}. \end{cases} \quad (7.2)$$

Just as system (7.1), it has $r_u + r_p$ unknowns ($\mathbf{a}$ and $\mathbf{b}$). The reduced eddy viscosity is found with the turbulence mapping $\mathcal{G}$, while the correction terms are computed using the correction map $\mathcal{M}$.

The online stage of **DD-EV-ROM** is detailed in Algorithms 4 (for steady and geometrically-parametrized flows) and 5 (for unsteady and physically-parametrized test cases). Algorithm 5 will be used in the numerical investigation of Sections 7.3.1 and 7.3.2, while Algorithm 4 is employed in the third test case (Section 7.3.3). The main differences are that in the steady case we do not consider the time-derivative term $\mathbf{M}_{ur}\dot{\mathbf{a}}$, and, in the case of parametrized geometries, the reduced operators are parametrized as well, and should be recomputed before solving the reduced system.

Algorithm 4 Pseudo-code for the online stage of DD-EV-ROM in the steady case. The part highlighted in orange corresponds to the test case with geometrical parametrization.

- 1: $\mu_\star$ test parameters, $\mathbf{M}_{ur} = \mathbf{0}$;
compute operators $\mathbf{B}_r(\mu_\star)$, $\mathbf{B}'_r(\mu_\star)$, $\mathbf{C}_{ur}(\mu_\star)$, $\mathbf{C}_{pr}(\mu_\star)$, $\mathbf{G}_r(\mu_\star)$, $\mathbf{D}_{pr}(\mu_\star)$,
- 2: $\mathbf{T}_{ur}(\mu_\star)$, $\mathbf{T}'_{ur}(\mu_\star)$, $\mathbf{N}_r(\mu_\star)$, $\mathbf{T}_{pr}(\mu_\star)$, $\mathbf{T}'_{pr}(\mu_\star)$,
 $\mathbf{E}_{rk}(\mu_\star)$, $\Phi_{BCk}(\mu_\star)$, with $k = 1, \dots, N_{BC}$.
- 3: initial guess $\mathbf{a}_0 = \mathbf{a}_0^{\text{proj}}(\mu_\star)$, $\mathbf{b}_0 = \mathbf{b}_0^{\text{proj}}(\mu_\star)$;
- 4: $i = 0$;
- 5: **repeat**
- 6: $\mathbf{g}_i \leftarrow \mathcal{G}(\mathbf{a}_i)(\mu_\star)$;
- 7: $\boldsymbol{\tau}_i = [\tau_u, \tau_p]_i \leftarrow \mathcal{M}(\mathbf{a}_i, \mathbf{g}_i)(\mu_\star)$;
- 8: compute $\mathbf{a}_{i+1}$, $\mathbf{b}_{i+1}$ by solving system (7.2) using the Newton Method;
- 9: $i \leftarrow i + 1$;
- 10: **until** Convergence or maximum iterations reached
- 11: $\mathbf{a}^{\text{sol}} \leftarrow \mathbf{a}_i$, $\mathbf{b}^{\text{sol}} \leftarrow \mathbf{b}_i$.

Algorithm 5 Pseudo-code for the online stage of algorithms DD-EV-ROM in the unsteady case with only physical parametrization.

```

1:  $\mu_\star^{\text{phys}}$  test physical parameters, except for time;
2: initial guess  $\mathbf{a}^{\text{sol}}(t_0) = \mathbf{a}^{\text{proj}}([t_0, \mu_\star^{\text{phys}}])$ ,  $\mathbf{b}^{\text{sol}}(t_0) = \mathbf{b}^{\text{proj}}([t_0, \mu_\star^{\text{phys}}])$ ;
3: time steps  $t_n$ ,  $n = 1, \dots, N_T$ ;
4: for  $n \in [0, \dots, N_T - 1]$  do
5:    $\mathbf{a}_0 \leftarrow \mathbf{a}^{\text{sol}}(t_{n+1})$ ,  $\mathbf{b}_0 \leftarrow \mathbf{b}^{\text{sol}}(t_{n+1})$ ;
6:    $i = 0$ ;
7:   repeat
8:      $\mathbf{g}_i \leftarrow \mathcal{G}(\mathbf{a}_i)([t_n, \mu_\star^{\text{phys}}])$ ;
9:      $\boldsymbol{\tau}_i = [\boldsymbol{\tau}_u, \boldsymbol{\tau}_p]_i \leftarrow \mathcal{M}(\mathbf{a}_i, \mathbf{g}_i)([t_n, \mu_\star^{\text{phys}}])$ ;
10:    compute  $\mathbf{a}_{i+1}$ ,  $\mathbf{b}_{i+1}$  by solving system (7.2) through the Newton Method;
11:     $i \leftarrow i + 1$ ;
12:  until convergence or maximum iterations reached
13:   $\mathbf{a}^{\text{sol}}(t_{n+1}) \leftarrow \mathbf{a}_i$ ,  $\mathbf{b}^{\text{sol}}(t_{n+1}) \leftarrow \mathbf{b}_i$ .
14: end for

```

7.2 Turbulence and correction maps: neural operators

The main question arising from Sections 5.4.1 and 5.4.2, where the turbulence and correction models have been recalled, is:

What is the most suitable ansatz for the turbulence map $\mathcal{G}$ and for the correction map $\mathcal{M}$?

Different choices are possible, and typically the most suitable ansatz highly depends on the test case considered. Here we focus on **neural operators**, which is an architecture specifically designed to learn operators. The first and simplest neural operator is the **Deep Operator Network (DeepONet)** [210] architecture, displayed in Figure 7.2. Given the observations $\mathbf{u}(x_1), \mathbf{u}(x_2), \dots$, and the parameters $\mathbf{y}$, the goal is to model the operator $\mathcal{O}(\mathbf{u})(\mathbf{y})$. The peculiarity of such model and the difference with respect to fully-connected neural networks is the presence of multiple sub-networks that separately process multiple inputs. In the notation we adopt, the *branch network* $\mathcal{B}$ handles the input $\mathbf{u}$, while the *trunk network* $\mathcal{T}$ takes as input $\mathbf{y}$. The outputs of the two networks, namely $\mathbf{o}_b = \mathcal{B}(\mathbf{u})$ and $\mathbf{o}_t = \mathcal{T}(\mathbf{y})$, that have the same dimension, are then combined with a cross product to obtain the final output $\mathcal{O}(\mathbf{u})(\mathbf{y})$.

For the turbulence map $\mathcal{G}$ we consider a modified version of the **DeepONet**, whose structure is detailed in Figure 7.3. The two inputs are in this case the velocity reduced variables $\mathbf{a} = \{a_i\}_{i=1}^{r_u}$, and the problem parameters $\boldsymbol{\mu} = \{\mu_i\}_{i=1}^P$. Hence, $\mathcal{G} : \mathbb{R}^{r_u+P} \rightarrow \mathbb{R}^{r_v}$. The difference with respect to the **DeepONet** in Figure 7.2 is that the two outputs $\mathbf{o}_b$ and $\mathbf{o}_t$ are then concatenated and processed through another sub-network, here called *reduction network* $\mathcal{H}$. Such procedure allows to have a multi-dimensional output, in our case $\mathcal{G}(\mathbf{a})(\boldsymbol{\mu}) = \mathcal{H}([\mathbf{o}_b, \mathbf{o}_t])$. As introduced in Equation (5.31), the loss function that the

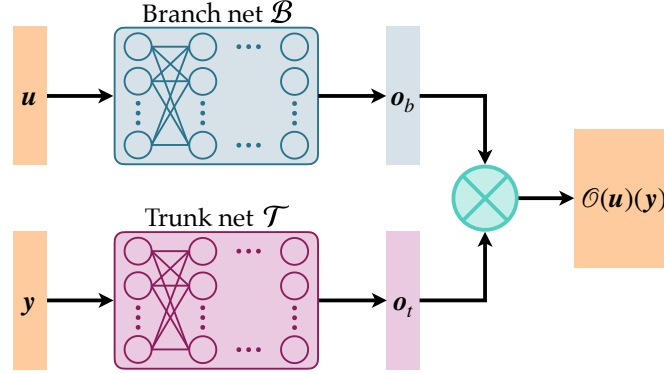
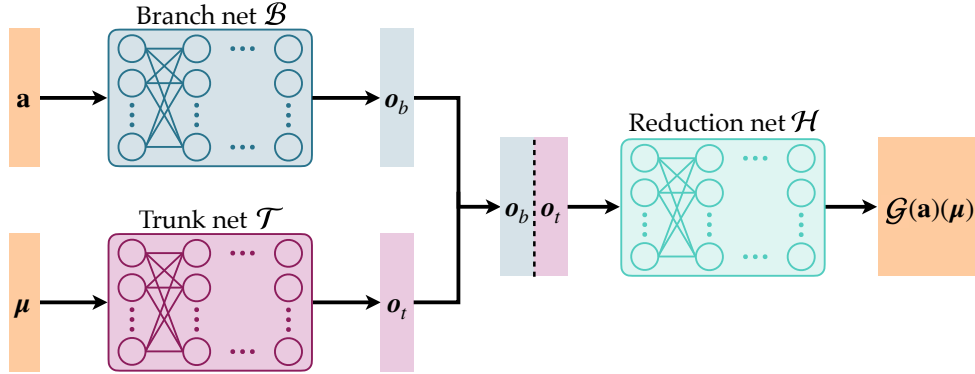


Figure 7.2: Architecture of the classical DeepONet [210].

network minimizes is expressed as follows:

$$\mathcal{L}_{\mathcal{G}} = \frac{1}{N_{\mu}} \sum_{i=1}^{N_{\mu}} \|\mathcal{G}(\mathbf{a})(\mu_i) - \mathbf{g}^{\text{proj}}(\mu_i)\|_2^2. \quad (7.3)$$


 Figure 7.3: Architecture of the turbulence map $\mathcal{G}$, adopted in DD-EV-ROM.

The correction map $\mathcal{M}(\mathbf{a}, \mathbf{g})(\mu)$ is instead modelled through a modified version of the **Multi-Input Operator Network (MIONet)**, whose structure is displayed in Figure 7.4. The architecture relies on three sub-networks separately processing the inputs $\mathbf{a} = \{a_i\}_{i=1}^{r_a}$, $\mathbf{g} = \{g_i\}_{i=1}^{r_g}$, and $\mu = \{\mu_i\}_{i=1}^P$. Three outputs o_{b_1} , o_{b_2} and o_t may be combined in different ways. In the original **MIONet** model [155], the temporary outputs are combined using a tensor product, in our case we use a simplified version and just consider a concatenation of the outputs as input to the reduction network $\mathcal{H}$. The final output is $\mathcal{M}(\mathbf{a}, \mathbf{g})(\mu) = \mathcal{H}([o_{b_1}, o_{b_2}, o_t])$.

The loss function minimized to find the optimal map $\mathcal{M}$, as in Equation (5.33), is:

$$\mathcal{L}_{\mathcal{M}} = \frac{1}{N_{\mu}} \sum_{i=1}^{N_{\mu}} \|\mathcal{M}(\mathbf{a}_i, \mathbf{g}_i)(\mu_i) - \tau^{\text{true}}(\mu_i)\|_2^2, \quad (7.4)$$

where the exact correction term includes both the velocity and pressure contributions: $\tau^{\text{true}} = [\tau_u^{\text{true}}, \tau_p^{\text{true}}]$, where τ_u^{true} and τ_p^{true} include the missing contribution of all nonlinear terms, as in Equation (5.39).

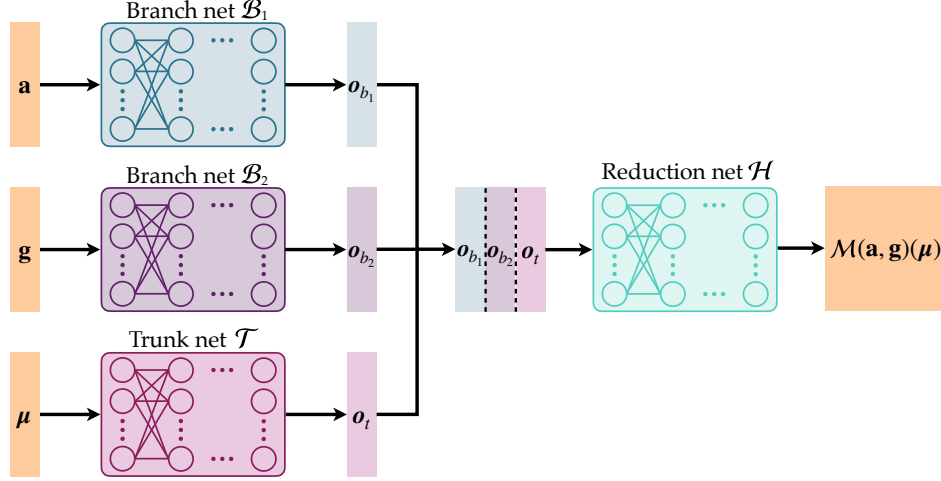


Figure 7.4: Architecture of the correction map $\mathcal{M}$, adopted in DD-EV-ROM.

The choice of adopting deep operator networks was driven by a comparative study conducted in [146], where various neural network architectures were investigated. This study shows that, while multi-layer perceptron networks are simple and effective, they lack a mechanism to properly handle different parametrized inputs, limiting their generalization capability. On the other hand, recurrent-type neural networks, like Long Short-Term Memory (LSTM), perform well in time-dependent problems, but might not be suitable for steady problems. **DeepONet** and **MIONet**, by contrast, are well-suited for multi-input operator learning and show superior robustness in different parametric settings [210].

Moreover, we investigate a modified training procedure with multiple contributions within the loss function, namely:

$$\mathcal{L}_{\mathcal{M}}^{\star} = \mathcal{L}_{\mathcal{M}} + \mathcal{L}_{\mathcal{G}} + \mathcal{L}_{\mathcal{MG}}, \quad (7.5)$$

where $\mathcal{L}_{\mathcal{M}}$ is in Equation (7.4), $\mathcal{L}_{\mathcal{G}}$ is in Equation (7.3), and $\mathcal{L}_{\mathcal{MG}}$ is computed as follows:

$$\mathcal{L}_{\mathcal{MG}} = \frac{1}{N_{\mu}} \sum_{i=1}^{N_{\mu}} \|\tau_i^{\text{true}}(\mathbf{a}_i^{\text{proj}}, \mathcal{G}(\mathbf{a}_i^{\text{proj}})(\mu_i), \mu_i) - \mathcal{M}(\mathbf{a}_i, \mathcal{G}(\mathbf{a}_i)(\mu_i))(\mu_i)\|_2^2. \quad (7.6)$$

In the modified training case, the weights of the pre-trained network $\mathcal{G}$ and of network $\mathcal{M}$ are optimized at the same time.

We introduce here the following notation:

- We call **DD-EV-ROM** the ROM where:
 - $\mathcal{G}$ minimizes the loss in Equation (7.3);

- $\mathcal{M}$ minimizes the loss in Equation (7.4).
- We call **Data-Driven Eddy-Viscosity Reduced Order Method with coupled training (DD-EV-ROM[★])** the ROM where:
 - $\mathcal{G}$ is pre-trained with loss function (7.3);
 - the pre-trained $\mathcal{G}$ and $\mathcal{M}$ are trained together to minimize the modified loss function in Equation (7.5).

The coupled training procedure considered in **DD-EV-ROM[★]** aims to guarantee higher accuracy, especially for test parameters and in extrapolation settings.

Remark: why DD-EV-ROM[★]?

It is important to describe the potential of the DD-EV-ROM[★] and the reason why the standard DD-EV-ROM may be inaccurate.

Indeed, considering only the contribution in (7.4) may cause lack of accuracy in the approximation of the correction term, especially in test and extrapolation settings. The reason is that network $\mathcal{M}$ is trained only relying on the *exact* coefficients $\mathbf{a}^{\text{proj}}$ and $\mathbf{g}^{\text{proj}}$. However, it may happen that, due to bad conditioning, when solving the online system (7.2), the velocity solution $\mathbf{a}^{\text{sol}}$ differs from the projected coefficients $\mathbf{a}^{\text{proj}}$, then

$$\|\mathbf{a}^{\text{sol}} - \mathbf{a}^{\text{proj}}\|_2 \neq 0.$$

Since the network $\mathcal{G}$ is also trained relying on the exact coefficients $\mathbf{a}^{\text{proj}}$, we might have:

$$\|\mathcal{G}(\mathbf{a}^{\text{sol}})(\boldsymbol{\mu}) - \mathbf{g}^{\text{proj}}\|_2 \neq 0,$$

and this might lead to inaccuracy and uncontrolled energy growth also in the EV-ROM model. Equivalently, for network $\mathcal{M}$:

$$\begin{aligned} & \|\mathcal{M}(\mathbf{a}^{\text{sol}}, \mathcal{G}(\mathbf{a}^{\text{sol}})(\boldsymbol{\mu}))(\boldsymbol{\mu}) - \boldsymbol{\tau}^{\text{true}}(\mathbf{a}^{\text{proj}}, \mathbf{g}^{\text{proj}}, \boldsymbol{\mu})\|_2 = \\ & \left\| \left[\mathcal{M}(\mathbf{a}^{\text{sol}}, \mathcal{G}(\mathbf{a}^{\text{sol}})(\boldsymbol{\mu}))(\boldsymbol{\mu}) - \boldsymbol{\tau}^{\text{true}}(\mathbf{a}^{\text{proj}}, \mathcal{G}(\mathbf{a}^{\text{sol}})(\boldsymbol{\mu}), \boldsymbol{\mu}) \right] + \right. \\ & \left. \left[\boldsymbol{\tau}^{\text{true}}(\mathbf{a}^{\text{proj}}, \mathcal{G}(\mathbf{a}^{\text{sol}})(\boldsymbol{\mu}), \boldsymbol{\mu}) - \boldsymbol{\tau}^{\text{true}}(\mathbf{a}^{\text{proj}}, \mathbf{g}^{\text{proj}}, \boldsymbol{\mu}) \right] \right\|_2 \gg 0. \end{aligned} \quad (7.7)$$

The expression in Equation (7.7) contains two distinct contributions:

- The first term $\mathcal{M}(\mathbf{a}^{\text{sol}}, \mathcal{G}(\mathbf{a}^{\text{sol}})(\boldsymbol{\mu}))(\boldsymbol{\mu}) - \boldsymbol{\tau}^{\text{true}}(\mathbf{a}^{\text{proj}}, \mathcal{G}(\mathbf{a}^{\text{sol}})(\boldsymbol{\mu}), \boldsymbol{\mu})$ is related to the generalization capability of network $\mathcal{M}$.
- The second term $\boldsymbol{\tau}^{\text{true}}(\mathbf{a}^{\text{proj}}, \mathcal{G}(\mathbf{a}^{\text{sol}})(\boldsymbol{\mu}), \boldsymbol{\mu}) - \boldsymbol{\tau}^{\text{true}}(\mathbf{a}^{\text{proj}}, \mathbf{g}^{\text{proj}}, \boldsymbol{\mu})$ is related to the performance of network $\mathcal{G}$. It is also important to highlight that the *exact* closure term (the **MIONet**'s target) depends on the eddy viscosity variables $\mathbf{g}$, as in expression (5.39).

The difference between the two types of training can be seen in Subsection 7.3.2, where even small perturbations in the velocity solution lead to inaccurate results for mapping $\mathcal{M}$. Moreover, in the **DD-EV-ROM** this may be reflected in inaccurate pressure approximations, as the reduced pressure $\mathbf{b}^{\text{sol}}$ comes after the residual minimization of the data-driven enriched reduced system.

7.3 Numerical results

In this section, the **DD-EV-ROM** and **DD-EV-ROM*** methods are compared with the literature approach **EV-ROM** in terms of accuracy and stability over time. We consider three different two-dimensional test cases:

- the unsteady flow past a cylinder with parametrized viscosity (Section 7.3.1);
- the unsteady channel-driven cavity flow with parametrized viscosity (Section 7.3.2);
- the steady backward facing step flow with parametrized geometry (Section 7.3.3).

The first two test cases are characterized by physical parameters, namely t and ν , while the third test case has three geometrical parameters. In all the test cases, we analyse the following aspects:

- (i) Description of the test case, and **POD** eigenvalues decay;
- (ii) Comparison among **EV-ROM**, **DD-EV-ROM**, and **DD-EV-ROM*** in terms of relative L^2 errors in training and test configurations, in different selected regimes $(r_u, r_p, r_{\nu t})$, associated with specific percentages of retained energy (as done in Chapter 6).
- (iii) Graphical representation of the results at the final online time instance for a selected regime, for an unseen configuration $\mu_\star$.

To evaluate the performance of the novel methods with respect to the literature baseline, we compute the average reduction of the relative error of **DD-EV-ROM** and **DD-EV-ROM*** with respect to the **EV-ROM** baseline, which corresponds to the gain in accuracy obtained with the novel correction approach. We denote it as “gain_s” for the generic field s , expressed as:

$$\begin{aligned} \text{gain}_s &= \frac{1}{N_\mu} \sum_{i=1}^{N_\mu} \frac{E_s^{\text{DD-EV-ROM}}(\mu_i) - E_s^{\text{EV-ROM}}(\mu_i)}{E_s^{\text{EV-ROM}}(\mu_i)}, \\ \text{gain}_s^* &= \frac{1}{N_\mu} \sum_{i=1}^{N_\mu} \frac{E_s^{\text{DD-EV-ROM}^*}(\mu_i) - E_s^{\text{EV-ROM}}(\mu_i)}{E_s^{\text{EV-ROM}}(\mu_i)}, \end{aligned} \tag{7.8}$$

for the **DD-EV-ROM** and **DD-EV-ROM***, respectively.

Regarding the offline stage, all simulations have been carried out using the open-source software OpenFOAM [153, 240], which employs the finite-volume discretization method. The **POD** operators are extracted using the library ITHACA-FV [138, 314], while the **ROM** integrated with the neural networks is solved using Python and PyTorch [249].

The computational times needed to perform the offline and online stages are deeply analysed in Section 7.3.4, which also provides a final discussion of the results obtained.

7.3.1 Flow past a cylinder test case

We consider a parametrized version of the periodic flow past a cylinder, where the parameters considered are time and the dynamic viscosity ν , that indirectly parametrizes the Reynolds number.

The computational domain is initialized with the velocity field $\mathbf{u} = (5, 0)$, pressure field $p = 0$, and eddy viscosity field $\nu_t = 2.35 \times 10^{-3}$ throughout the entire domain. The domain, including the boundaries notation, and the computational grid are displayed in Figure 7.5.

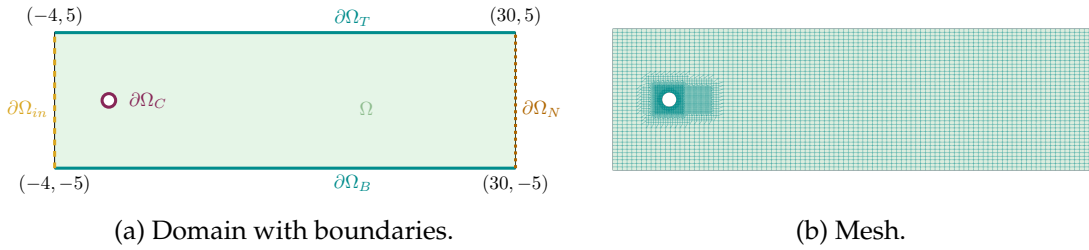


Figure 7.5: Flow past a cylinder test case. Domain and mesh considered.

The boundary conditions, inspired by previous studies [314, 131, 143, 142], are prescribed as follows:

$$\begin{aligned}
 \text{On } \partial\Omega_{in} : \quad & \begin{cases} \mathbf{u} = (5, 0), \\ \nabla p \cdot \mathbf{n} = 0, \\ \nu_t = 2.35 \times 10^{-3}; \end{cases} & \text{On } \partial\Omega_T \cup \partial\Omega_B : \quad & \begin{cases} \mathbf{u} \cdot \mathbf{n} = 0, \\ \nabla p \cdot \mathbf{n} = 0, \\ \nabla \nu_t \cdot \mathbf{n} = 0; \end{cases} \\
 \text{On } \partial\Omega_N : \quad & \begin{cases} \nabla \mathbf{u} \cdot \mathbf{n} = 0, \\ p = 0, \\ \nu_t = 2.35 \times 10^{-3}; \end{cases} & \text{On } \partial\Omega_C : \quad & \begin{cases} \mathbf{u} = \mathbf{0}, \\ \nabla p \cdot \mathbf{n} = 0, \\ \nu_t \text{ wall function of type nutkWallFunction.} \end{cases}
 \end{aligned}$$

The full-order simulations are performed over a range of 5 viscosity values corresponding to Reynolds numbers between 2×10^4 and 6×10^4 . For each viscosity value, 500 snapshots are collected, yielding a total of 2500 snapshots, corresponding to the time window [20, 22], when the flow is periodic. The snapshots are stored every 0.004 s. The penalty coefficient used for the weak imposition of the boundary conditions is set to $\tau_{BC} = 10^3$ and is found following the procedure described in [315]. All reduced-order simulations employ a second-order backward differentiation formula (BDF2) for time integration.

The parameters' range both used to train the **POD** and to test the methods, and the resulting cumulative **POD** eigenvalues are represented in Figure 7.6.

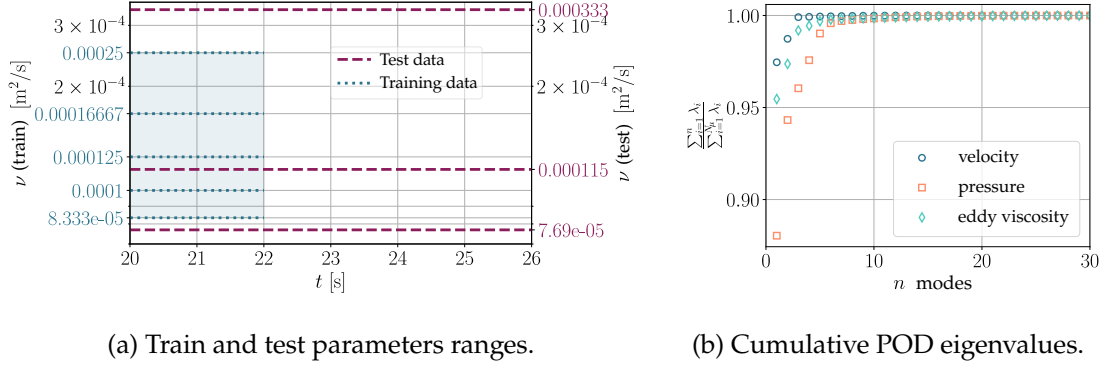


Figure 7.6: *Flow past a cylinder test case*. Train and test parameters for the ROMs, and cumulative eigenvalues obtained from the POD.

The **POD** eigenvalue analysis reveals a rapid decay of the modal energy, particularly for the velocity field. Nevertheless, as previously discussed, the standard **POD**–Galerkin formulation may still exhibit limited accuracy in the marginally-resolved regime. The remainder of this section presents a quantitative comparison of the relative errors obtained with **EV-ROM**, **DD-EV-ROM**, and **DD-EV-ROM*** for several combinations of velocity, pressure, and eddy-viscosity modes, denoted by the triplet (r_u, r_p, r_{v_t}) . These modal configurations are selected according to the **POD** energy content retained by each reduced basis, ranging from strongly compressed to well resolved reduced-order representations, as reported in Table 7.1. As already noticed in the previous chapter (Section 6.3.1), the flow past a cylinder is a simple test case, as a few number of modes is needed to capture most of the system's energy.

Table 7.1: *Flow past a cylinder test case*. Regimes considered for ROMs for the velocity, pressure and eddy viscosity fields.

Retained energy [%]	r_u	r_p	r_{v_t}
95	1	3	1
99	3	5	3
99.5	3	6	5
99.6	3	7	5
99.7	3	8	6
99.8	3	9	7
99.9	3	12	13
99.99	10	24	30

To quantify the impact of the data-driven closure terms, we consider the average reduction of the relative error defined in Equation (7.8) over the time interval $[20, 22]$. The analysis is performed for the velocity, pressure, and eddy-viscosity fields across all modal configurations considered. Both results on training and testing parameters are reported

in Figure 7.7. Overall, both DD-EV-ROM and DD-EV-ROM^{*} consistently outperform the baseline EV-ROM across all modal regimes. Several observations can be drawn from the results.

- ▶ The largest improvements are obtained for the pressure field. This can be attributed to the use of a PPE formulation together with a dedicated pressure correction term. Furthermore, pressure approximations produced by the baseline EV-ROM are generally less accurate than the corresponding velocity approximations, resulting in larger pressure closure terms τ_p that can be more effectively learned by the neural-network-based closure model.
- ▶ The performance obtained on the training and testing sets is comparable. This indicates that the closure mapping $\mathcal{M}$ generalizes well and remains effective even in extrapolation scenarios, without exhibiting overfitting.
- ▶ The average error reduction tends to increase as more modes are retained in the reduced basis. At first glance, this behaviour may appear counterintuitive, since larger corrections might be expected in more compressed reduced spaces. However, the baseline EV-ROM becomes increasingly prone to instabilities as the latent dimension grows, leading to less accurate predictions. Consequently, the contribution of the data-driven correction becomes more significant in well resolved reduced-order spaces.

Figure 7.8 presents a qualitative comparison between the approximations obtained with EV-ROM, DD-EV-ROM, and DD-EV-ROM^{*}, using the modal configuration $(r_u, r_p, r_{v_t}) = (10, 24, 30)$. The reduced-order solutions are compared against the corresponding full-order reference solutions for an unseen parameter value. The analysis focuses on the eddy viscosity field, but similar comparisons can be made for the velocity and pressure fields. The results clearly demonstrate the improvement in accuracy provided by DD-EV-ROM^{*}. The eddy viscosity field is visually very close to the projection, indicating that the reduced-order approximation approaches the best achievable representation within the selected reduced basis. Moreover, for all considered fields, DD-EV-ROM^{*} consistently provides more accurate predictions than DD-EV-ROM, highlighting the benefit of the coupled training procedure.

7.3.2 Channel-driven cavity flow test case

We here consider the unsteady flow inside a cavity-shaped channel, which is characterized by a different behaviour with respect to the first test case. More in detail, the flow is here driven through a channel that features a protrusion on the bottom wall. Vortices of various sizes form mainly nearby the cavity. The vortices have irregular and non-periodic pattern, differently from the flow past a cylinder. Also in this case, we consider a parametrized setup, having as parameters the time and the viscosity ν .

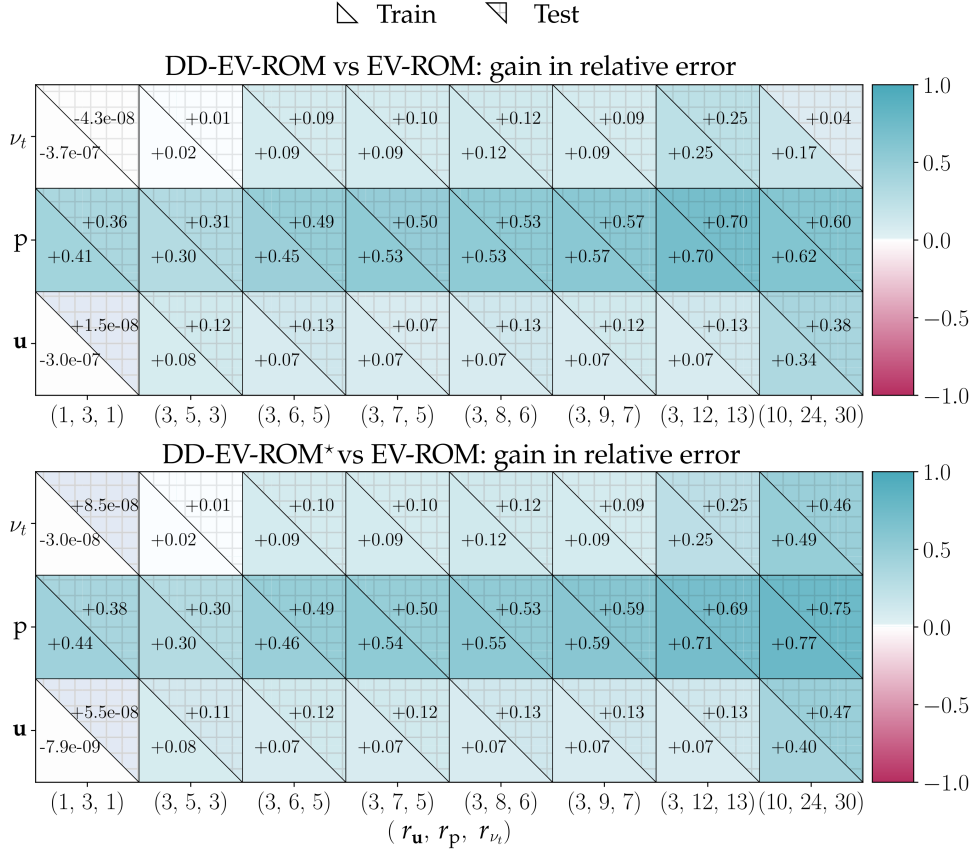


Figure 7.7: *Flow past a cylinder test case*. Gain in relative error at different latent dimensions and for all the fields considered.

The domain with specified boundaries, and the computational grid are reported in Figure 7.9. We consider as initial conditions $\mathbf{u} = (0.1, 0)$, $p = 0$, and $\nu_t = 6 \times 10^{-4}$. Employing the notation of Figure 7.9, we consider the following boundary conditions:

$$\begin{aligned}
 \text{On } \partial\Omega_T : \begin{cases} \nabla \mathbf{u} \cdot \mathbf{n} = 0, \\ \nabla p \cdot \mathbf{n} = 0, \\ \nu_t = 1 \times 10^{-8}; \end{cases} & \quad \text{On } \partial\Omega_B : \begin{cases} \mathbf{u} = \mathbf{0}, \\ \nabla p \cdot \mathbf{n} = 0, \\ \nu_t = 1 \times 10^{-8}; \end{cases} \\
 \text{On } \partial\Omega_{in} : \begin{cases} \mathbf{u} = (0.1, 0), \\ \nabla p \cdot \mathbf{n} = 0, \\ \nu_t = 6 \times 10^{-4}; \end{cases} & \quad \text{On } \partial\Omega_N : \begin{cases} \nabla \mathbf{u} \cdot \mathbf{n} = 0, \\ p = 0, \\ \nabla \nu_t \cdot \mathbf{n} = 0. \end{cases}
 \end{aligned}$$

For the **POD** construction, we consider 5 viscosity values corresponding to Reynolds numbers in the range $[1 \times 10^3, 2 \times 10^4]$. Snapshots are collected every 0.05 s over a simulation horizon of 10 seconds, resulting in a total of 1000 snapshots. Specifically, 200 snapshots are retained from each full-order simulation. For this test case, the penalty coefficient

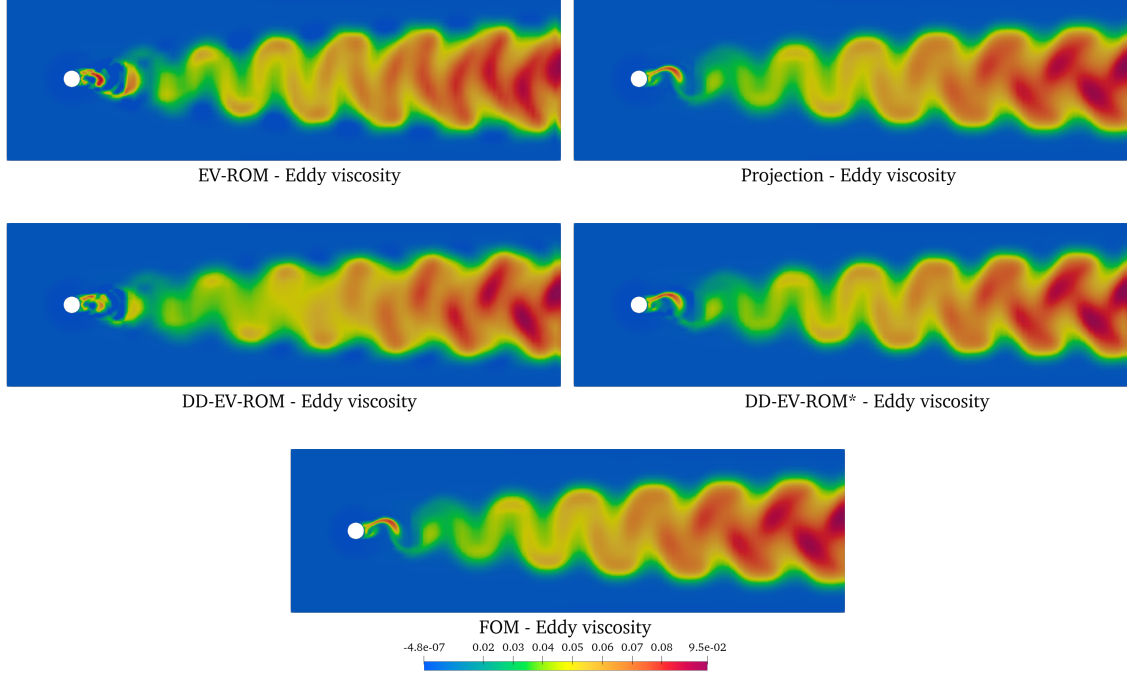


Figure 7.8: *Flow past a cylinder test case*. Eddy viscosity approximation obtained from EV-ROM, DD-EV-ROM, DD-EV-ROM*, and projection at final online time instance ($t = 8$) and $\nu = 1.15 \times 10^{-4} \text{ m}^2 \text{ s}^{-1}$, compared with the FOM ground truth. For the ROM the modal regime is $(r_u, r_p, r_{v_t}) = (10, 24, 30)$.

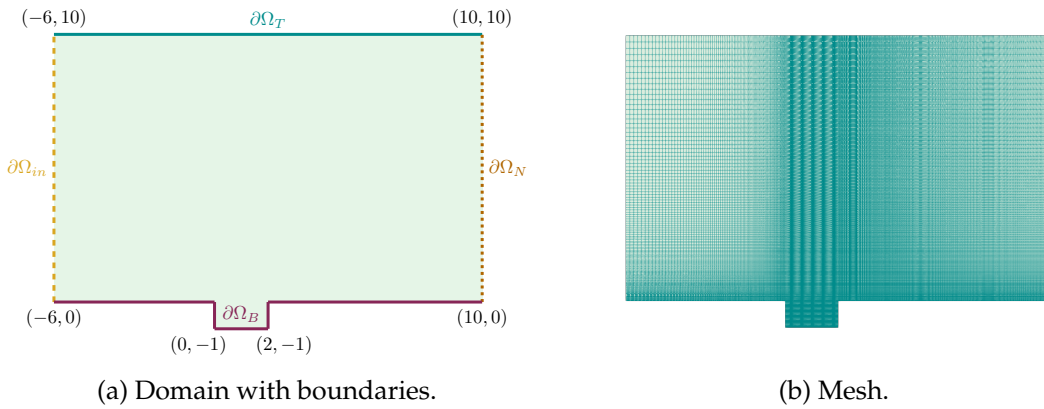


Figure 7.9: *Channel-driven cavity flow test case*. Domain and mesh considered.

is set to $\tau_{BC} = 1$, and optimized using the method of [315]. The ROMs employ a first-order implicit Euler time integration scheme, consistently with the full-order discretization adopted in OpenFOAM. The train and test parameters' values and the resulting POD cumulative eigenvalues are displayed in Figure 7.10. As in the first test case, a few number of modes is necessary to achieve a large percentage of retained energy. In particular,

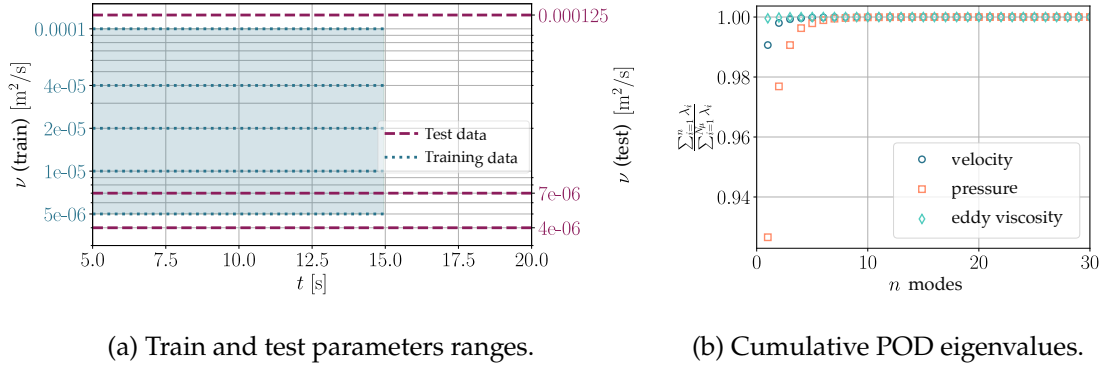


Figure 7.10: *Channel-driven cavity flow test case*. Train and test parameters for the ROMs, and cumulative eigenvalues obtained from the POD.

we perform the ROM analysis considering the reduced regimes reported in Figure 7.2, selected following energy criteria.

Table 7.2: *Channel-driven cavity flow test case*. Regimes considered for ROMs for the velocity, pressure and eddy viscosity fields.

Retained energy [%]	r_u	r_p	r_{v_t}
95	1	2	1
99	1	3	1
99.5	2	4	1
99.7	2	5	1
99.9	3	7	1

The global error analysis is performed in Figure 7.11, which shows the gain metric of Equation (7.8) for the velocity, pressure and eddy viscosity fields, at different modal combinations for training and test configuration. This test case is more challenging than the cylinder one, since it is not periodic and complex vortex structures are generated in the region nearby the cavity. We can indeed draw the following considerations.

- **DD-EV-ROM*** always provides improved approximations with respect to **EV-ROM**.
- The main difference with respect to the previous test case is that the two proposed methodologies perform significantly different. In particular, **DD-EV-ROM*** always provides improved approximations, while the **DD-EV-ROM** shows worse results than **EV-ROM** for some under-resolved modal regimes, like $(r_u, r_p, r_{v_t}) = (1, 2, 1)$ and $(r_u, r_p, r_{v_t}) = (2, 4, 1)$.

- The results obtained for the velocity and for the eddy viscosity are similar to the results of **EV-ROM**. This is likely due to the fact that the **EV-ROM** predictions are very close to the projection, which is the best result we can achieve. Hence, there is no hope to improve the **EV-ROM** result, if it coincides with the projection.
- Training and test results are always similar for both methods. This suggests that no overfitting is observed for the **DD-EV-ROM**. It happens indeed that the training errors are large even if the mapping $\mathcal{M}$ reaches small loss values. The reason is that, even if the reduced velocity solution is close to the projected field, the problem may be ill-conditioned. Therefore, small perturbations $\|\mathbf{a}^{\text{sol}} - \mathbf{a}^{\text{proj}}\|_2$ may lead to large output differences $\|\tau^{\text{true}}(\mathbf{a}^{\text{sol}}, \mathcal{G}(\mathbf{a}^{\text{sol}})(\mu), \mu) - \tau^{\text{true}}(\mathbf{a}^{\text{proj}}, \mathbf{g}^{\text{proj}}, \mu)\|_2$, and the *standard* training procedure of $\mathcal{M}$ is not robust enough to capture the correct mapping.

From Figure 7.11, we can conclude the positive performance of **DD-EV-ROM*** over **DD-EV-ROM** in most of the modal regimes.

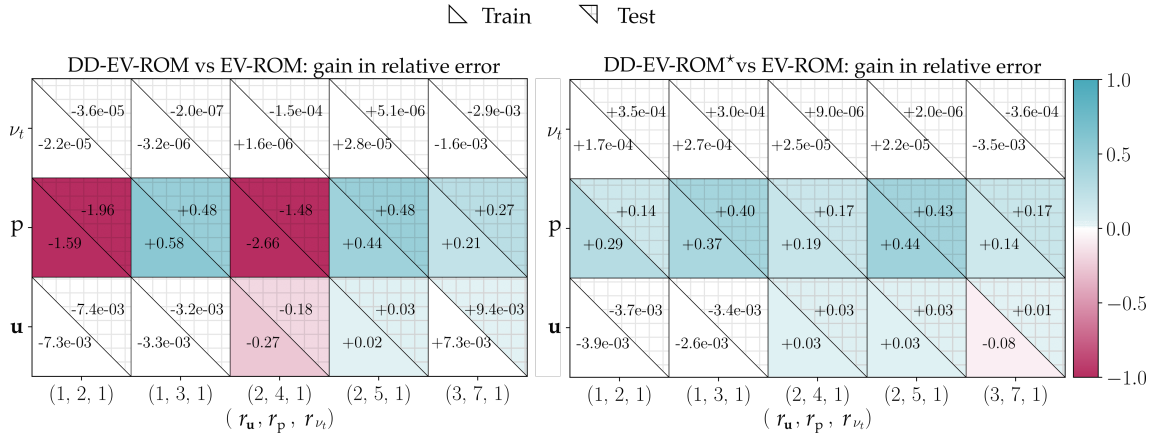


Figure 7.11: *Channel-driven cavity flow test case*. Gain in relative error at different latent dimensions and for all the fields considered.

Finally, a graphical comparison among the pressure **ROM** approximations is provided in Figure 7.12. The figure shows the pressure absolute errors with respect to the **FOM** reference in a zoomed region of the domain nearby the cavity for $(r_u, r_p, r_{v_t}) = (2, 5, 1)$, at the final online time instance and for an unseen viscosity value. It shows that the **DD-EV-ROM** and **DD-EV-ROM*** improve the results of the baseline **EV-ROM**.

In particular, the approximation error of **DD-EV-ROM** is qualitatively similar to the reconstruction error, which is the best result we can achieve with our method.

7.3.3 Backward-facing step test case

The backward-facing step test case is characterized by a geometrical parameterization, with three parameters, namely the step slope α , the inlet height h_1 , and the global height of the domain h_2 , represented in Figure 7.13. Differently from the other two test cases, the flow is steady. Therefore, we focus on the error analysis of the stationary solution with respect to the **FOM** counterpart, at different parametrized geometries.

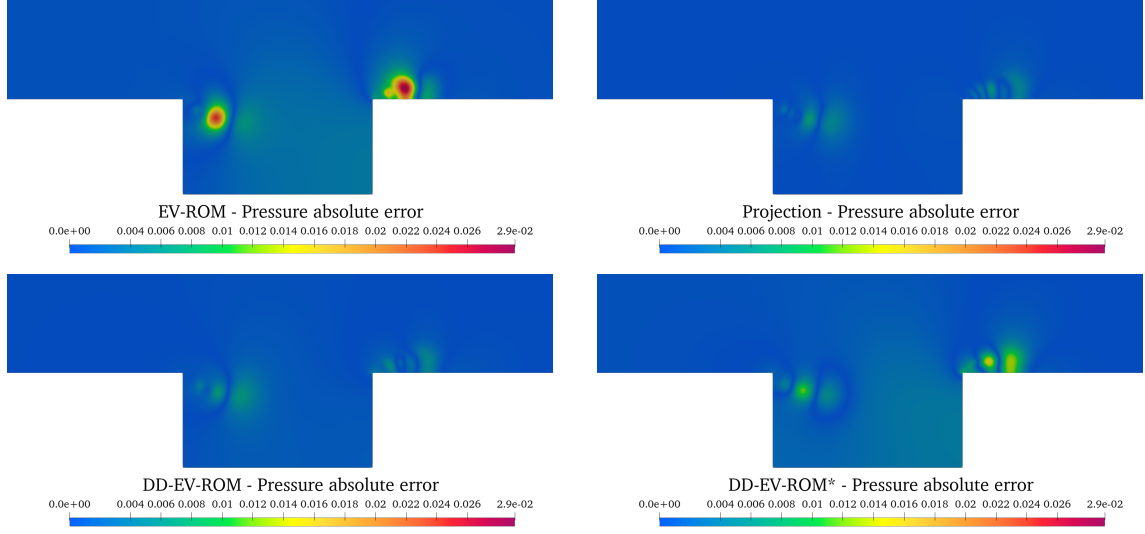


Figure 7.12: *Channel-driven cavity flow test case*. Absolute error in the pressure approximation obtained from EV-ROM, DD-EV-ROM, DD-EV-ROM*, and projection at final on-line time instance ($t = 15$), at $\nu = 7 \times 10^{-6} \text{ m}^2 \text{ s}^{-1}$. For the ROM the modal regime is $(r_u, r_p, r_{v_t}) = (2, 5, 1)$.

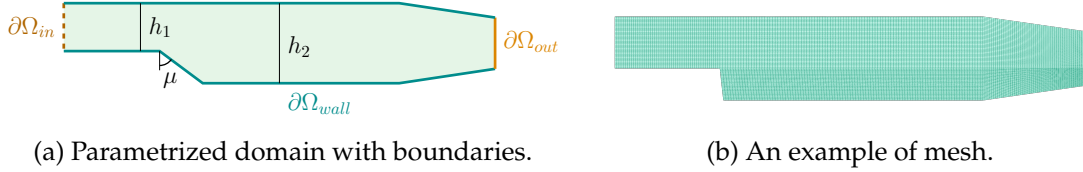
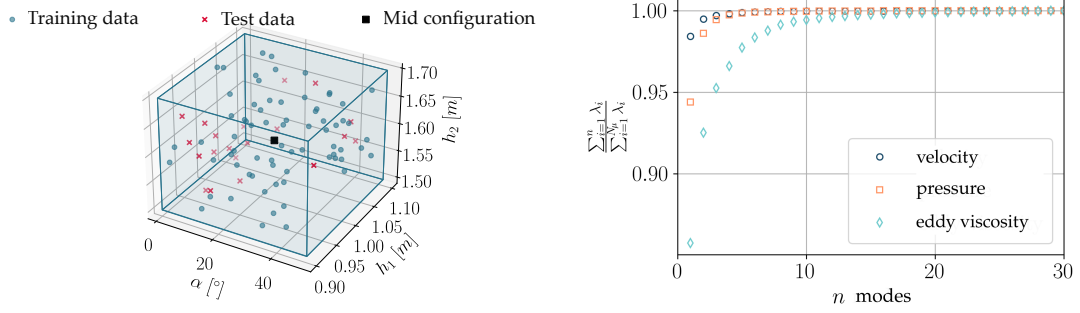


Figure 7.13: *Backward-facing step test case*. Parametrized domain and mesh considered.

Figure 7.13 displays the parametrized domain, and the grid used for a single set of geometrical parameters. The offline simulations are performed considering as initial conditions $\mathbf{u} = (1, 0)$, $p = 0$ and $\nu_t = 8.52 \times 10^{-4}$ in the whole domain, while the boundary conditions read as follows.

$$\begin{aligned} \text{On } \partial\Omega_{in} : \quad & \begin{cases} \mathbf{u} = (1, 0), \\ \nabla p \cdot \mathbf{n} = 0, \\ \nu_t = 8.52 \times 10^{-4}; \end{cases} & \text{On } \partial\Omega_{out} : \quad & \begin{cases} \nabla \mathbf{u} \cdot \mathbf{n} = 0, \\ p = 0, \\ \nu_t = 8.52 \times 10^{-4}; \end{cases} \\ \text{On } \partial\Omega_{wall} : \quad & \begin{cases} \mathbf{u} = 0, \\ \nabla p \cdot \mathbf{n} = 0, \\ \nu_t \text{ wall function of type nutkWallFunction.} \end{cases} \end{aligned}$$

We consider a specific set of training parameters for the **POD** $\{\mu_i\}_{i=1}^{N_\mu}$, where $\mu_i = (\alpha_i, h_{1_i}, h_{2_i})$. $N_\mu = 70$ is the number of simulations with deformed mesh, while we test the approach on 20 test parameters. The training, test, and mid-configuration parameters, and the **POD** cumulative eigenvalues trend are displayed in Figure 7.14.



(a) Train and test parameters ranges.

(b) Cumulative POD eigenvalues.

 Figure 7.14: *Backward-facing step test case*. Train and test parameters for the ROMs, and cumulative eigenvalues obtained from the POD.

It is important to specify that the matrices $\mathbf{S}_u$, $\mathbf{S}_p$ and $\mathbf{S}_{v_t}$ include not only the stationary fields but also the *intermediate states*. By *intermediate states* we refer to snapshots extracted at intermediate iterations of the steady-state Navier–Stokes solver in OpenFOAM, which is performed using the SIMPLE algorithm. As previously done in [361], these snapshots are taken before full convergence, and are used to enrich the snapshot matrices with additional flow states. We call $n_{\text{iter}}(\mu_i)$ the global number of iterations for each parameters' set, and T_i the number of intermediate states retained as **POD** snapshots. We decided to take one snapshot every five iterations of the SIMPLE algorithm, namely $T_i = \frac{n_{\text{iter}}(\mu_i)}{5}$. The value of T_i depends on the specific geometry, namely on the parameters' set μ_i , and in the simulations we performed, $T_i \in [139, 196]$. Being $N_\mu = 70$, the global number of snapshots is $\sum_{i=1}^{N_\mu} T_i = 11717$. Considering a generic field s , we have the corresponding snapshots' matrix:

$$\mathbf{S}_s = \{s^{(1)}(\mathbf{x}_1, \mu_1), \dots, s^{(T_1)}(\mathbf{x}_1, \mu_1), \dots, s^{(1)}(\mathbf{x}_{N_\mu}, \mu_{N_\mu}), \dots, s^{(T_{N_\mu})}(\mathbf{x}_{N_\mu}, \mu_{N_\mu})\},$$

where $s^{(j)}(\mathbf{x}_i, \mu_i)$ indicates the j -th intermediate state for the field s on the deformed mesh $\mathbf{x}_i$ corresponding to parameter μ_i .

As the mesh has fixed size $N_p = 14040$, matrices $\mathbf{S}_p$ and $\mathbf{S}_{v_t}$ belong to $\mathbb{R}^{N_c \times \sum_{i=1}^{N_\mu} T_i} \equiv \mathbb{R}^{14040 \times 11717}$, while $\mathbf{S}_u \in \mathbb{R}^{2N_p \times \sum_{i=1}^{N_\mu} T_i} \equiv \mathbb{R}^{28080 \times 11717}$.

This strategy is here employed to ensure better convergence of the **ROM** solver and good **ROM** accuracy. Indeed, considering only the final steady states may lead to **ROM** instability, since the reduced solver (as the full order counterpart) visits different intermediate solutions. On the other hand, considering intermediate solutions within the snapshots' matrices leads to superior robustness and stability of the reduced system.

In this geometrically-parametrized test case, the basis is computed in the mid-configuration, corresponding to parameters $\mu_{\text{mid}} = (\alpha_{\text{mid}}, h_1, h_2) = (25^\circ, 1 \text{ m}, 1.6 \text{ m})$, as done in [361]. As can be seen from Figure 7.14, this test case exhibits a significantly slower decay for the eddy viscosity field. The **ROM** performance is then analysed in the modes' combination reported in Table 7.3. As for the other test cases, the criterion considered to select the

modes values (r_u, r_p, r_{v_t}) is the retained energy. Indeed, as the Table shows, the value of r_{v_t} is always bigger than r_u and r_p .

Table 7.3: *Backward-facing step test case.* Regimes considered for ROMs for the velocity, pressure and eddy viscosity fields.

Retained energy [%]	r_u	r_p	r_{v_t}
95	1	2	3
99	2	3	8
99.5	3	4	12
99.6	3	4	14
99.7	3	4	16
99.8	4	5	20

For this test case, the optimized penalty coefficient considered for the boundary conditions is $\tau_{BC} = 1$.

Figure 7.15 represents the average error reduction in training and test configurations for the three fields of interest, in different regimes (also specified in Table 7.3).

From the figure, we can draw the following considerations:

- There is no clear trend in the average error reduction, differently from the first two test cases. For instance, in the first test case, we noticed that the average error reduction increases as the number of modes increases and we justified this because of the worse performance of the baseline **EV-ROM**. In the second test case, we noticed that only the pressure has a consistent error reduction, while the other fields are substantially unchanged with respect to the **EV-ROM**, since it was already close to the projection and could not be improved more. Here, the fact that there is not a clear trend is linked to the **complexity of the snapshots' manifold**. Moreover, the **POD** basis is computed in a mid-configuration and this may lead to lower **ROM** accuracy for parameters far from the mid-configuration (even in the training set).
- The error reduction obtained for the eddy viscosity field is in general linked to the velocity one. For instance, when $(r_u, r_p, r_{v_t}) = (1, 2, 3)$, the gain in accuracy is close to zero for both velocity and eddy viscosity, while for $(r_u, r_p, r_{v_t}) = (3, 4, 16)$ the two fields have similar error values. This is due to the structure of the turbulence map $\mathcal{G}$, that has inputs $(\mathbf{a}, \boldsymbol{\mu})$. Therefore, $\mathbf{a}^{\text{sol}} \simeq \mathbf{a}^{\text{proj}}$ leads to $\mathcal{G}(\mathbf{a}^{\text{sol}}, \boldsymbol{\mu}) \simeq \mathbf{g}^{\text{proj}}$.
- In general, the **DD-EV-ROMs** always improve the accuracy of the standard **EV-ROM**, except for the velocity at $(r_u, r_p, r_{v_t}) = (3, 4, 14)$. However, even in this case, this is compensated by an improvement in the pressure approximation. Furthermore, we noticed that the increasing error of the velocity does not reflect in a increasing error for the eddy viscosity.
- The results of **DD-EV-ROM** and **DD-EV-ROM*** are similar, with slightly better performances of the **DD-EV-ROM*** method, as in $(r_u, r_p, r_{v_t}) = (4, 5, 20)$.

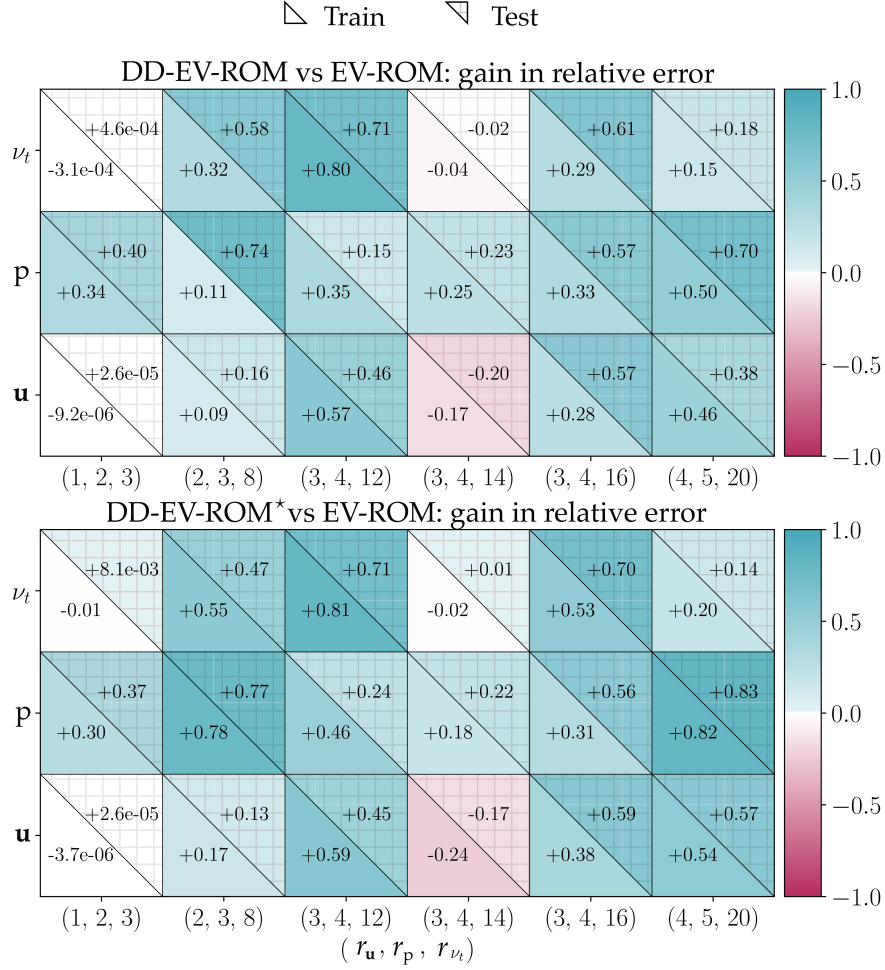


Figure 7.15: *Backward-facing step test case*. Gain in relative error at different latent dimensions and for all the fields considered.

Qualitative results in terms of eddy viscosity approximation, for a test parameter, namely $(\alpha, h_1, h_2) = (6.3^\circ, 0.99 \text{ m}, 1.58 \text{ m})$, in the regime $(r_u, r_p, r_{\nu_t}) = (3, 4, 12)$, are represented in Figure 7.16. The standard EV-ROM fails in accurately approximating this field, generating unphysical oscillations in the bottom part of the domain. The addition of the correction terms allows to almost reach the projection accuracy.

7.3.4 Computational time and Discussion

The wall time required for the different stages of the computational pipeline is summarized in Table 7.4, which reports both offline and online costs.¹ The results can be interpreted by separating the overall workflow into four main components:

- **Full-order simulations (FOM).** The table reports the computational cost of both a single simulation and the complete set of high-fidelity runs. For the unsteady test

¹All computations are performed on a single core of an Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz with 16GB of RAM.

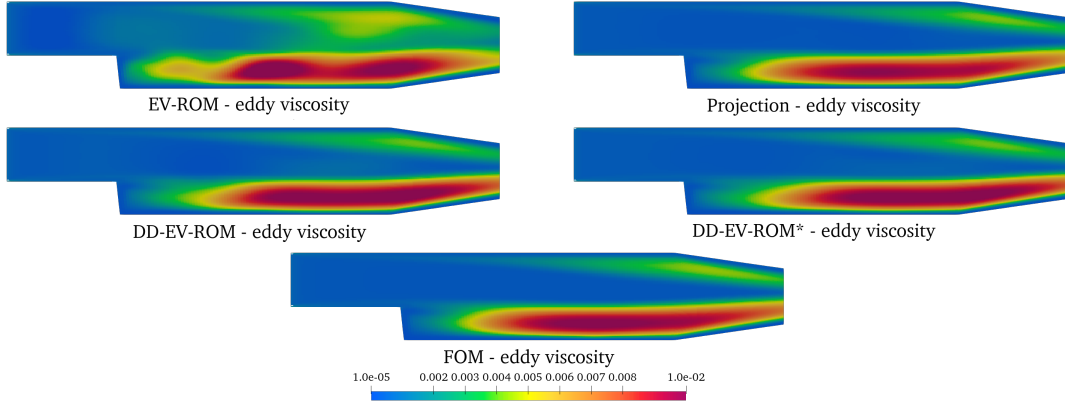


Figure 7.16: *Backward-facing step test case*. Eddy viscosity approximation obtained from EV-ROM, DD-EV-ROM, DD-EV-ROM*, and projection compared with the FOM ground truth, at the test parameters $(\alpha, h_1, h_2) = (6.3^\circ, 0.99 \text{ m}, 1.58 \text{ m})$. For the ROM the modal regime is $(r_u, r_p, r_{v_t}) = (3, 4, 12)$.

Table 7.4: Wall times needed to perform the offline and online stages of the ROM pipeline, in the three cases considered. One-simulation FOM and ROM times are intended for 1 s of simulation in the unsteady test cases, and for the steady simulation in the backstep test case. For neural networks' training and ROMs, we report minimum and maximum wall times indicated as $t_{\min} - t_{\max}$. Indeed, in such cases the wall time depends on the reduced dimensions considered, namely on the combination (r_u, r_p, r_{v_t}) .

Offline/online stages		Flow past a cylinder	Channel-driven cavity	Backstep flow
Offline	FOM (one simulation)	4 min 48 s	2 min 11 s	~ 9 min
	FOM (total)	48 min	59 min 10 s	~ 10 h
	POD and assemble operators	30 min	5 h 53 min	17 h 33 min
	Training $\mathcal{G}$	15–16 min	13–14 min	44–45 min
	Training $\mathcal{M}$ (DD-EV-ROM)	16–17 min	17–30 min	64–66 min
	Coupled training (DD-EV-ROM*)	66–77 min	51–59 min	3 h 53 min–4 h 10 min
Online	Assemble operators	–	–	5–9 s
	EV-ROM (one simulation)	0.45–0.93 s	0.22–0.26 s	0.05–0.07 s
	DD-EV-ROMs (one simulation)	0.64–1.19 s	0.27–0.29 s	0.05–0.16 s

cases, the reported values correspond to one second of physical time integration, while for the steady test case they correspond to a single converged simulation. The three configurations exhibit comparable single-run costs. The larger total cost observed in the steady backward-facing step case is mainly due to the higher number of parameter instances, with $N_\mu = 70$.

- ▶ **POD construction and operator assembly.** The cost of assembling reduced operators depends strongly on the mesh size, as it involves $L^2(\Omega)$ inner products. Although the channel-driven cavity case features the largest mesh and would therefore be expected to be the most expensive, the highest computational cost is instead observed in the backstep flow case. This is due to the parameter-dependent nature of the operators. In this setting, the reduced operators must be assembled for each parameter instance in the training set, and related terms are required to compute the exact closure contributions used as targets for the correction map $\mathcal{M}$.
- ▶ **Coupled training of $\mathcal{G}$ and $\mathcal{M}$.** The training stage is more expensive for the backstep flow case. This is due to the need to evaluate the eddy viscosity approximation $\mathcal{G}(\mathbf{a}_i^{\text{proj}})(\mu_i)$ at every training epoch, which is then used to update the target closure terms entering the loss function in Equation (7.6).
- ▶ **Reduced-order simulations (ROMs).** The baseline **EV-ROM** and the proposed data-driven variants exhibit comparable online costs. The data-driven approaches incur a slightly higher computational cost due to the evaluation of the closure mapping, which introduces an additional term of the form $\tau = \mathcal{M}(\mathbf{a}, \mathbf{g})(\mu)$. In the last test case, an additional overhead is present due to the need to assemble parameter-dependent reduced operators during the online stage. Despite this additional cost, all data-driven models achieve substantial speed-ups with respect to the full-order simulations. The observed acceleration is approximately $300\times$ in the flow past a cylinder case, $250\times$ in the channel-driven cavity flow, and $80\times$ in the backstep case. The reduced speed-up of the last test case is attributable to the additional cost associated with parameter-dependent operator assembly, as already mentioned before.

Overall, the proposed **DD-EV-ROMs** provide speed-ups of order $\mathcal{O}(10^2)$ across all test cases. More in detail, the geometrically-parametrized test case requires larger offline effort due to the geometric parametrization and the need to assemble parameter-dependent operators. Even in this more challenging setting, the **ROMs** maintain a favourable computational advantage over the full-order simulations while improving accuracy relative to the baseline **EV-ROM**. In particular, the pressure field benefits most from the inclusion of the closure strategy, where the **PPE** formulation incorporates a dedicated closure term. Finally, we address the comparison between **DD-EV-ROM** and **DD-EV-ROM*** in terms of accuracy and robustness. Across the three test cases considered, the following trends emerge:

- ▶ In the cylinder test case, both training strategies behave similarly. Time-dependent error trends indicate that **DD-EV-ROM*** exhibits slightly improved temporal stability, particularly in extrapolation regimes.
- ▶ In the cavity test case, the differences between the two approaches become more pronounced. For certain modal configurations, the **DD-EV-ROM** may become unstable, whereas the alternative training strategy enhances both accuracy and stability. In this case, the latter acts not only as a performance improvement but also as a stabilizing mechanism for the reduced system.
- ▶ In the backstep configuration, both approaches yield comparable results. This case remains the most challenging due to the complexity of the underlying solution manifold.

Overall, although the **DD-EV-ROM*** introduces additional computational cost, it consistently provides improved robustness and, in challenging regimes, superior stability compared to the **DD-EV-ROM**.

Finally, Table 7.5 displays the details of the architectures of the **DeepONet** used for the turbulence model $\mathcal{G}$ and of the **MIONet** used for the correction model $\mathcal{M}$. In both models, the learning rate decays by a factor $\gamma = 0.2$ every $N_{\text{step}} = 3000$ epochs. In the **DD-EV-ROM***, after a pre-training (following the hyperparameters in Table 7.5), both models are trained together for additionally 20000 epochs, minimizing the loss function expressed in $\mathcal{L}_{\mathcal{M}}^*$.

Table 7.5: Hyperparameters of DeepONet $\mathcal{G}$ and MIONet $\mathcal{M}$ networks, considered for the turbulence and closure mappings, respectively.

Mapping	Network	Hidden layers	Non-linearity	Learning rate	Stopping epoch
$\mathbf{g} = \mathcal{G}(\mathbf{a})(\boldsymbol{\mu})$	$\mathcal{B}$	[20, 20, 20]	Softplus	$(1 \times 10^{-3})\gamma^{\lfloor \frac{t_{\text{epoch}}}{N_{\text{step}}} \rfloor}$	20000
	$\mathcal{T}$	[20, 20, 20]	Softplus		
	$\mathcal{H}$	[20, 20, 20]	Softplus		
$\boldsymbol{\tau} = \mathcal{M}(\mathbf{a}, \mathbf{g})(\boldsymbol{\mu})$	$\mathcal{B}_1$	[20, 20, 20]	Softplus	$(1 \times 10^{-3})\gamma^{\lfloor \frac{t_{\text{epoch}}}{N_{\text{step}}} \rfloor}$	20000
	$\mathcal{B}_2$	[20, 20, 20]	Softplus		
	$\mathcal{T}$	[20, 20, 20]	Softplus		
	$\mathcal{H}$	[20, 20, 20]	Softplus		

7.4 Conclusions and Motivation to the next chapter

The methodology presented in this chapter is designed to improve the accuracy of **POD-Galerkin ROMs** by recovering, through machine-learning techniques, the contribution of the modes discarded during the projection process in a parametrized setting. As for

Chapter 6, the proposed framework mostly operates in a *marginally-resolved* regime, where the retained modes capture most of the system energy but standard ROMs are typically inaccurate. As a baseline model, we consider the EV-ROM, which incorporates an eddy-viscosity closure at the reduced level to maintain consistency with the full-order RANS formulation, following the approach proposed in [131] (and briefly recalled in Section 5.4.1). The proposed strategy builds upon previous closure methodologies developed in [357, 226, 228, 142], where additional correction terms were introduced into the reduced system to account for the discarded dynamics. In those studies, the closure terms were modeled through quadratic functions of the reduced coefficients and were restricted to non-parametric settings (see Section 5.4.2 for more details).

The first contribution of the present work is the **parametric extension** of this framework through the use of neural operators. A DeepONet mapping $\mathcal{G}$, referred to as the *turbulence map*, is employed to approximate the reduced eddy-viscosity coefficients from the reduced velocity representation and the flow parameters. A MIONet mapping $\mathcal{M}$, denoted as the *correction map*, predicts the reduced closure terms using the reduced velocity coefficients, the reduced eddy-viscosity coefficients, and the parameter vector as inputs. These two components define the proposed DD-EV-ROM framework.

The second contribution is a **coupled training strategy** for the mappings $\mathcal{G}$ and $\mathcal{M}$. By explicitly linking the approximation of the turbulent viscosity with the prediction of the closure terms during training, the resulting framework, denoted as DD-EV-ROM*, achieves improved robustness and stability.

The proposed methodologies are assessed on three benchmark problems: the unsteady flow past a cylinder, the unsteady channel-driven cavity flow, and the steady backward-facing step flow with geometric parametrization. The results demonstrate that both DD-EV-ROM and DD-EV-ROM* consistently improve upon the baseline EV-ROM across a wide range of modal regimes and for unseen configurations, confirming the predictive capabilities of the proposed approach. Among the variables considered, the pressure field exhibits the most significant benefit from the introduction of the closure terms, where a dedicated correction is incorporated within the reduced PPE. Furthermore, the coupled-training strategy underlying DD-EV-ROM* provides additional robustness in highly unsteady and challenging reduced-order settings, often yielding more accurate and stable solutions than the standard DD-EV-ROM formulation.

The methodology presented in this chapter is developed within the framework of **intrusive** reduced-order modeling, where the governing equations are explicitly available and can be projected onto a reduced space. In this setting, the proposed correction strategy is designed to compensate for the effects of mode truncation by leveraging data-driven approximations of the unresolved dynamics.

While most of the existing literature on closure modeling for ROMs has focused on intrusive formulations, the intrusive setting has some inherent limitations:

- In practical applications, the access to the governing equations may be limited or

completely unavailable.

- ▶ In many real-world scenarios, only a restricted set of measurements or simulation data is available, making intrusive projection-based approaches difficult or impossible to apply.
- ▶ Even when enhanced through data-driven correction terms, the reduced dynamics remain governed by a projected system of equations, whose stability properties may deteriorate as the simulation evolves in time, leading to a progressive degradation of the solution quality.

Beyond the specific formulation considered here, the underlying idea is more general: *exploiting data to recover information that is lost during the reduction process.*

The next chapter adopts this broader viewpoint and investigates how the concept of data-driven correction can be extended to a completely data-driven and nonintrusive framework. Rather than augmenting a projected dynamical system, available data is used to directly correct the solution.

CORRECTED-ROMs: AN EXTENSION OF CLOSURES TO NONINTRUSIVE ROMs

The goal of this chapter is to extend the concept of *closure* or *correction* to the framework of completely data-driven, nonintrusive ROM. The term *closure* is traditionally used in CFD and reduced-order modeling to denote an additional contribution introduced into the governing equations in order to account for unresolved effects or missing information. In this context, closure terms are incorporated directly into the mathematical model and are used to recover the influence of scales, variables, or dynamics that are not explicitly represented in the resolved system.

In the present chapter, however, we consider a different setting. Since we focus on fully data-driven nonintrusive ROM and no explicit governing equations are available, the objective is to improve the accuracy of the POD-based approximation by learning a suitable correction directly from data. The learned quantity is therefore added to the reduced-order prediction itself, rather than to the governing equations. The ultimate goal remains the same as in the intrusive setting: to compensate for the loss of information induced by mode truncation and improve the predictive capabilities of the baseline linear POD approximation, especially in case of nonlinear dynamics, e.g., turbulent flows. In this context, several authors have proposed **quadratic ROMs** as an extension of the classical linear framework [357, 106, 23, 302, 151]. In these approaches, the reduced solution is augmented by a quadratic contribution built from the retained POD coefficients, with the aim of recovering part of the information associated with the neglected modes. From this perspective, quadratic ROMs can be interpreted as correction models for POD, since they seek to reconstruct unresolved dynamics through interactions among the resolved modes. There is, however, no unique strategy for determining such a correction. A common approach consists in identifying the corresponding quadratic operator from data through a linear least-squares problem, as previously done in the field of operator inference [106, 251]. This procedure is computationally efficient and often provides excellent agreement with the training data. However, the resulting models may suffer from overfitting and limited generalization capabilities, especially when the identification problem is ill-conditioned or when predictions are required for previously unseen configurations. We named this approach **Quadratic Correction via Least-Squares (Quad-LS)** and consider it as a baseline for comparison.

To address these limitations, we propose a novel data-driven framework in which the

quadratic correction operator is no longer identified through a linear least-squares procedure, but is instead learned as a nonlinear mapping from data. More specifically, the key novelties of this work can be summarized as follows:

- ▶ We introduce a neural-network-based approximation of the quadratic correction operator. In particular, we employ architectures inspired by the **DeepONet** [210] and the **MIONet** [155], which were adopted in Chapter 7 with a similar scope.
- ▶ We develop a continuous-in-space formulation, referred to as **Quadratic Correction via DeepONet (QuadNet)**, in which the spatial coordinates are explicitly incorporated into the model through collocation points. This enables the correction to be evaluated continuously over the computational domain rather than being tied to a fixed spatial discretization.
- ▶ We further extend the methodology to parametrized problems through the **Quadratic Correction via MIONet (QuadNet- μ)** architecture, which augments the network inputs with the physical or geometrical parameters defining the problem. This allows the learned correction to vary continuously across the parameter space and generalize to unseen configurations.

A graphical version of the pipeline followed to build the corrected ROM is displayed in Figure 8.1.

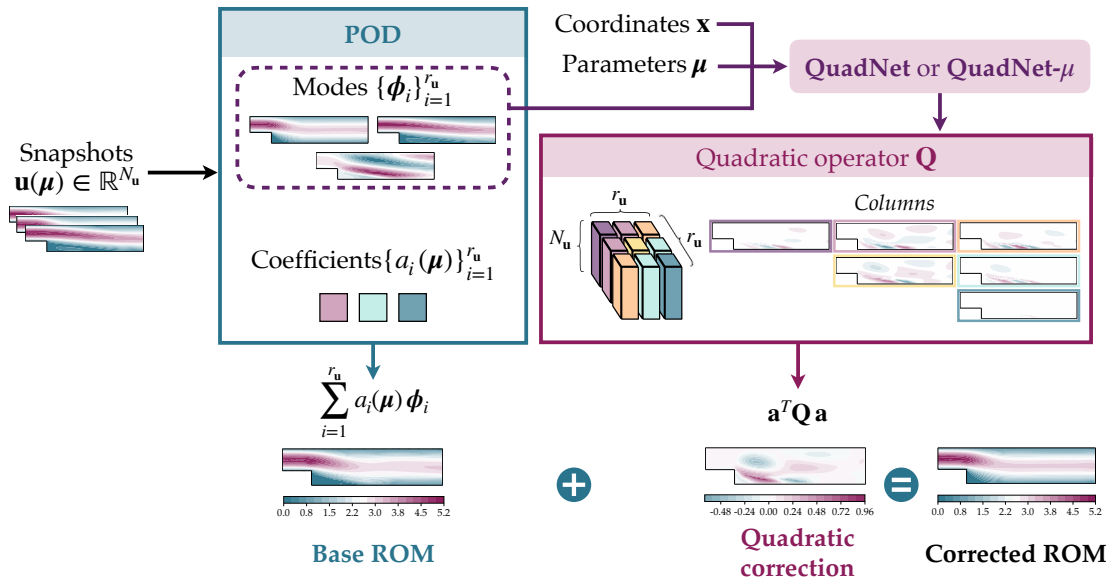


Figure 8.1: Graphical abstract to build a corrected ROM, which enhances the standard POD-based nonintrusive ROM with a data-driven machine-learning-based quadratic correction.

This chapter is subdivided as follows:

- ▶ a brief section dedicated to nonintrusive **ROMs** (Section 8.1);

- ▶ the presentation of the quadratic correction approaches **Quad-LS**, **QuadNet** and **QuadNet- μ** (Section 8.2);
- ▶ the numerical investigation on two test cases, namely the two-dimensional backward-facing step flow, and the lid-driven cavity flow (Section 8.3);
- ▶ the final conclusive Section 8.4 for both the present chapter and the whole Part II, motivating the development of fully-nonlinear **ROMs** in Part III.

8.1 Preliminaries: nonintrusive ROMs

This section is dedicated to briefly recalling the basic knowledge of nonintrusive **ROMs**. As introduced in Chapter 5 and illustrated in Figure 5.1, the starting point is the collection of the high-fidelity snapshots in the snapshots' matrix. We here only focus on the velocity field, and hence consider the velocity snapshots' matrix:

$$\mathbf{S}_u = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_{N_\mu} \\ | & | & & | \end{bmatrix},$$

where $\mathbf{u}_i = \mathbf{u}(\mathbf{x}, \mu_i)$, $i = 1, \dots, N_\mu$ are the discrete velocity snapshots at each parameters' instance $\mu_i \in \mathbb{R}^P$. We recall that each discrete velocity is evaluated at the same mesh points $\mathbf{x} \in \mathbb{R}^{N_u}$. If we consider geometrical parametrization, as in the backstep test case of Chapter 7, different grids are possible, as long as the underlying family of domains preserves topological consistency.

Once the high-fidelity manifold is populated, non-intrusive **ROM** consist of three main stages:

- ▶ **Projection** of the solution manifold onto a low-dimensional space of dimension r_u .
- ▶ Prediction of the reduced representation corresponding to unseen parameter values through **interpolation or regression techniques**.
- ▶ **Backmapping**, namely the reconstruction of the high-dimensional solution from its reduced representation.

The first projection step, namely **POD**, is here conducted by computing the Singular Value Decomposition (SVD)¹ of the snapshots' matrix $\mathbf{S}_u$, namely:

$$\mathbf{S}_u = \mathbf{\Phi} \mathbf{\Sigma} \mathbf{V}^T,$$

where $\mathbf{\Phi}$ contains some pre-selected **POD** modes $\{\phi_i\}_{i=1}^{r_u}$, $\mathbf{\Sigma}$ is the diagonal matrix of singular values, and $\mathbf{V}$ contains the corresponding right singular vectors. The solution can then be approximated as

$$\mathbf{u}_r(\mathbf{x}, \mu) = \sum_{i=1}^{r_u} a_i(\mu) \phi_i(\mathbf{x}). \quad (8.1)$$

¹In the intrusive setting of Section 5.1, the **POD** is instead computed via the correlation matrix.

The computational efficiency of **POD** stems from the separation of the spatial dependence, encoded in the modes, and the parametric dependence, encoded in the reduced coefficients.

To predict the solution for an unseen parameter value $\mu_\star$, several approximation strategies can be employed, including **RBF** interpolation [1, 50], **Gaussian Process Regression (GPR)** [67, 242], and neural-network-based regressors [127]. In this chapter, we adopt an **RBF** interpolation strategy, defined as

$$\mathbf{a}(\mu_\star) = \sum_{i=1}^{N_\mu} \omega_i \varphi(\|\mu_\star - \mu_i\|), \quad (8.2)$$

where $\varphi(\cdot)$ denotes the radial basis kernel, ω_i are the interpolation weights, and μ_i are the training parameters, with $i = 1, \dots, N_\mu$. The interpolation weights are determined by enforcing exact reconstruction of the training coefficients, obtained by projecting the snapshots onto the reduced space:

$$\sum_{i=1}^{N_\mu} \omega_i \varphi(\|\mu_j - \mu_i\|) = \mathbf{a}^{\text{proj}}(\mu_j) = \Phi^T \mathbf{u}(\mu_j), \quad j = 1, \dots, N_\mu.$$

The **RBF** interpolation is particularly attractive due to its flexibility, its ability to handle scattered data, and its good approximation properties. The choice of the kernel function depends on the characteristics of the problem and the complexity of the reduced manifold. In the numerical experiments presented in this chapter, we employ both the thin-plate spline and linear kernels:

$$\varphi(d) = d^2 \log(d), \quad \varphi(d) = d. \quad (8.3)$$

Once the reduced coefficients corresponding to an unseen parameter are predicted, the full-order approximation is recovered through the **POD** expansion of Equation 8.1.

8.2 Corrected-ROMs

As discussed in the previous section, the approximation provided by the **POD** expansion of Equation 8.1 may become inaccurate when only a limited number of modes is retained, particularly in scarce-data settings. To mitigate this limitation, we augment the reduced-order approximation with a correction term, obtaining

$$\mathbf{u}(\mathbf{x}, \mu) \approx \sum_{i=1}^r a_i(\mu) \phi_i(\mathbf{x}) + \tau(\mathbf{x}; \mu). \quad (8.4)$$

The correction term $\tau(\mathbf{x}; \mu) \in \mathbb{R}^{N_u}$ is modelled through a fully data-driven mapping, whose objective is to approximate the exact reconstruction error of the **POD** approximation, namely

$$\tau(\mathbf{x}, \mu) \approx \tau^{\text{true}}(\mathbf{x}; \mu) = \mathbf{u}(\mathbf{x}, \mu) - \mathbf{u}_r(\mathbf{x}, \mu), \quad (8.5)$$

where $\mathbf{u}_r$ is the **POD** approximation of Equation (8.1). The purpose of the proposed model (8.5) is therefore to recover, through a learned correction, the information lost during the dimensionality-reduction process. Since the exact correction depends on the available reduced-order information, the corresponding approximation can be expressed in terms of the retained **POD** modes, the reduced coefficients, and, in a parametrized setting, the problem parameters. In general, we seek a mapping of the form

$$\tau = \tau(\phi_1, \dots, \phi_{r_u}, a_1, \dots, a_{r_u}; \mu).$$

In this section, we investigate three different types of correction, all based on a quadratic ansatz with respect to the reduced coefficients:

- ▶ **Quad-LS**: a non-trainable approach based on the identification of a quadratic correction operator through a least-squares procedure, following the methodology proposed in [106].
- ▶ **QuadNet**: a **DeepONet**-inspired architecture that learns the correction term from the **POD** modes $(\phi_1, \dots, \phi_{r_u})$ and the reduced coefficients $(a_1, \dots, a_{r_u})$.
- ▶ **QuadNet- μ** : a **MIONet**-inspired extension of **QuadNet** that additionally incorporates the parameter vector μ among the network inputs. The architecture receives $(\phi_1, \dots, \phi_{r_u})$, $(a_1, \dots, a_{r_u})$, and μ , allowing the correction to vary continuously across the parameter space.

8.2.1 Quad-LS (state of the art)

The **Quad-LS** strategy aims at approximating the correction term with the quadratic form:

$$\tau(\mu) = \mathbf{a}^T(\mu) \mathbf{Q} \mathbf{a}(\mu). \quad (8.6)$$

This strategy is employed in [251, 106], and relies on the determination of the quadratic operator $\mathbf{Q}$ via least-squares optimization. In fact, the procedure to determine $\mathbf{Q}$ is very general, and can be applied to the inference of quadratic operators in different contexts.

It is important to highlight that in Equation (8.6), $\tau(\mu) = \tau(\mathbf{x}; \mu)$ is the N_u -dimensional vector with the values of the correction at each mesh node, while $\mathbf{Q} \in \mathbb{R}^{r_u \times N_u \times r_u}$ is a tensor that satisfies the following minimization problem:

$$\min_{\mathbf{Q} \in \mathbb{R}^{r_u \times N_u \times r_u}} \sum_{j=1}^{N_\mu} \|\tau^{\text{true}}(\mu_j) - \mathbf{a}(\mu_j)^T \mathbf{Q} \mathbf{a}(\mu_j)\|_2^2. \quad (8.7)$$

The minimization problem in (8.7) can be re-written and solved as a least squares problem exploiting the symmetry of the quadratic form $\mathbf{a}^T \mathbf{Q} \mathbf{a}$. We can indeed write the quadratic form as a double summation:

$$\sum_{i=1}^{r_u} \sum_{j=1}^{r_u} a_i(\mu) a_j(\mu) Q_{ij}. \quad (8.8)$$

We introduce a vector $\tilde{\mathbf{a}}(\boldsymbol{\mu}) \in \mathbb{R}^s$, which contains all pairwise products of the components of $\mathbf{a}$ with themselves, without repetition. Hence it has dimension $s = \frac{r_u(r_u+1)}{2}$ and is defined as

$$\tilde{\mathbf{a}}(\boldsymbol{\mu}) = \begin{bmatrix} \mathbf{a}^{(1)}(\boldsymbol{\mu}) \\ \vdots \\ \mathbf{a}^{(r_u)}(\boldsymbol{\mu}) \end{bmatrix}, \text{ where } \mathbf{a}^{(i)}(\boldsymbol{\mu}) = a_i(\boldsymbol{\mu}) \begin{bmatrix} a_1(\boldsymbol{\mu}) \\ \vdots \\ a_i(\boldsymbol{\mu}) \end{bmatrix} \in \mathbb{R}^i. \quad (8.9)$$

We can also write $\mathbf{Q}$ in block form, as

$$\hat{\mathbf{Q}} = \begin{bmatrix} \hat{\mathbf{Q}}^{(1)}, \dots, \hat{\mathbf{Q}}^{(r_u)} \end{bmatrix} \in \mathbb{R}^{N_u \times s}, \quad (8.10)$$

where $\hat{\mathbf{Q}}^{(i)} \in \mathbb{R}^{N_u \times i}$. The minimization in Equation (8.7) can be then written as:

$$\min_{\hat{\mathbf{Q}} \in \mathbb{R}^{N_u \times s}} \sum_{j=1}^{N_\mu} \|\boldsymbol{\tau}^{\text{true}}(\boldsymbol{\mu}_j) - \hat{\mathbf{Q}}\tilde{\mathbf{a}}(\boldsymbol{\mu}_j)\|_2^2. \quad (8.11)$$

We refer the reader to [251] for the extended least squares derivation.

An important characteristic of this approach is that it identifies a single quadratic operator $\mathbf{Q}$ from the available training data. Since $\mathbf{Q}$ is fixed and does not explicitly depend on the problem parameters, its predictive capabilities may be limited when extrapolating to unseen configurations. In particular, the approximation accuracy can deteriorate when the training manifold used to solve the minimization problem (8.7) does not adequately capture the complexity and variability of the underlying high-fidelity solution manifold.

8.2.2 QuadNet

To overcome the limited generalization capabilities of the **Quad-LS** approach, we introduce a novel machine-learning-based correction strategy. The proposed model is defined as

$$\tau(\boldsymbol{\mu}) = \mathbf{a}^T(\boldsymbol{\mu}) \mathbf{Q}(\boldsymbol{\Phi})(\mathbf{x}) \mathbf{a}(\boldsymbol{\mu}), \quad (8.12)$$

where the explicit dependence on $\mathbf{x}$ emphasizes that the correction is represented as a continuous function in space. The quadratic operator $\mathbf{Q}(\boldsymbol{\Phi})(\mathbf{x})$ is approximated through a **DeepONet**-inspired architecture, illustrated in Figure .

The **DeepONet**, introduced in the previous chapter (see Figure 7.2) is characterized by the presence of multiple subnetworks that process different inputs independently before combining the extracted information. In the present setting, the r_u **POD** modes are provided as input to the *branch network* $\mathcal{B}$, while the corresponding d -dimensional mesh coordinates are supplied to the *trunk network* $\mathcal{T}$. More specifically, we define the mappings

$$\mathcal{B} : \mathbb{R}^{r_u} \rightarrow \mathbb{R}^l, \quad \mathcal{T} : \mathbb{R}^d \rightarrow \mathbb{R}^l,$$

so that each **DeepONet** input consists of the values of the r_u retained modes evaluated at a given spatial location together with the coordinates of that location. The outputs of the branch and trunk networks are subsequently merged and processed by a third network,

$$\mathcal{H} : \mathbb{R}^l \rightarrow \mathbb{R}^s,$$

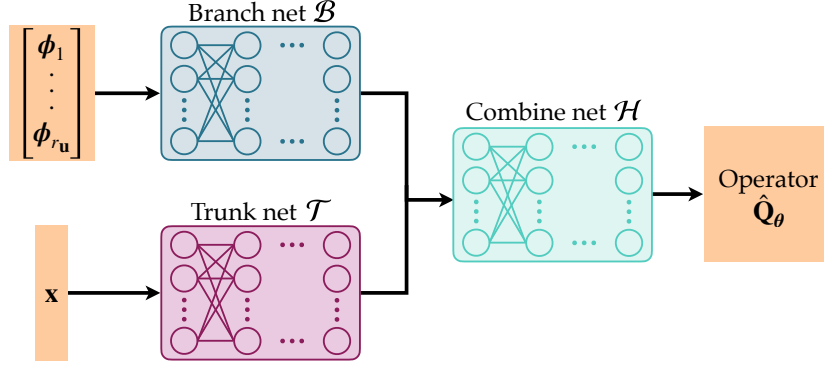


Figure 8.2: Schematic representation of the architecture adopted for QuadNet.

whose output is the matrix $\hat{\mathbf{Q}}_{\theta}(\Phi)(\mathbf{x})$, corresponding to the block representation introduced in 8.10. Here, θ denotes the set of trainable parameters of the neural architecture. Although the latent dimension l can in principle be chosen arbitrarily, numerical experiments indicate that selecting $l = s$ provides a good balance between accuracy and computational cost.

The main advantage of **QuadNet** is its continuous dependence on the spatial coordinates. As a consequence, the learned quadratic operator can be evaluated at locations that do not belong to the original computational mesh. Furthermore, the network can be trained using only a subset of the mesh points, significantly reducing both memory requirements and training costs. The impact of the number of training points is investigated in the numerical experiments through a dedicated sensitivity analysis. The network parameters are identified by minimizing the loss function

$$\mathcal{L}(\theta) = \frac{1}{N_{\mu}} \sum_{i=1}^{N_{\mu}} \frac{\|\tau^{\text{true}}(\mu_i) - \hat{\mathbf{Q}}_{\theta}(\Phi)(\mathbf{x}) \tilde{\mathbf{a}}(\mu_i)\|_{l_2}^2}{\|\tau^{\text{true}}(\mu_i)\|_{l_2}^2}. \quad (8.13)$$

The use of a relative-error loss, rather than the most common mean squared error, is motivated by the need to reduce the sensitivity of the training process to variations in the magnitude of the correction terms. In practice, the corrections may be very small or may exhibit significantly different scales across the dataset, making the optimization more challenging. Normalizing the error with respect to the correction norm alleviates this issue and generally leads to a more robust training procedure. Although the normalization factors may be costly to compute for large meshes, they are evaluated only once before training and therefore have a negligible impact on the overall computational cost.

8.2.3 QuadNet- μ

Both the **Quad-LS** and **QuadNet** models proposed are fixed in the parametric space. We propose a further parametrized extension of **QuadNet**, named **QuadNet- μ** , which approximates the correction as:

$$\tau(\mu) = \mathbf{a}^T(\mu) \mathbf{Q}(\Phi, \mu)(\mathbf{x}) \mathbf{a}(\mu). \quad (8.14)$$

In this case, the neural network used to compute $\mathbf{Q}(\Phi, \mu)(\mathbf{x})$ has a **MIONet**-inspired structure, which is characterized by more than two subnetworks. The proposed architecture is indeed composed of two branch networks and one trunk network, that separately process the modes Φ , the parameter μ , and the spatial coordinates $\mathbf{x}$. More precisely, the first branch network defines a map $\mathcal{B}_1 : \mathbb{R}^{r_u} \mapsto \mathbb{R}^l$, whose input is an r_u -dimensional vector representing the r_u modes evaluated at a given point in the domain; the second branch network is a map $\mathcal{B}_2 : \mathbb{R}^P \mapsto \mathbb{R}^l$, whose input is a P -dimensional vector representing the parameters of the problem (in our case $P = 1$); finally, the trunk network is a map $\mathcal{T} : \mathbb{R}^d \mapsto \mathbb{R}^l$, whose input is the d -dimensional vector of spatial coordinates for a given point in the domain. As for the **QuadNet**, a final subnetwork $\mathcal{O} : \mathbb{R}^l \mapsto \mathbb{R}^s$ outputs the block version of the quadratic operator, namely $\hat{\mathbf{Q}}_\theta(\Phi, \mu)(\mathbf{x})$, where θ are the parameters of the network learned during training. The **QuadNet- μ** architecture is represented in Figure 8.3. This model is trained considering the same loss function as in Equation

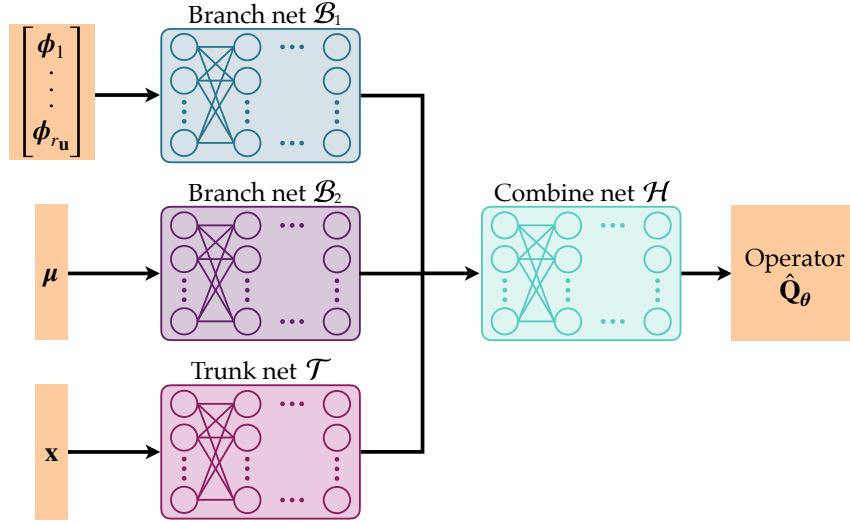


Figure 8.3: Schematic representation of the architecture adopted for QuadNet- μ .

(8.13), where the network output also incorporates the parameter dependence, namely $\hat{\mathbf{Q}}_\theta(\Phi, \mu)(\mathbf{x})$.

Remark: technical note on QuadNet and QuadNet- μ . It is important to highlight that the quadratic expression $\mathbf{a}^T \mathbf{Q} \mathbf{a}$ is not used in practice, neither in the **NN** training nor in the online evaluations of the quadratic correction. We indeed approximate the correction with the equivalent quadratic form $\hat{\mathbf{Q}} \tilde{\mathbf{a}}$, where $\hat{\mathbf{Q}} = \mathbf{QuadNet}(\Phi)(\mathbf{x})$ or $\hat{\mathbf{Q}} = \mathbf{QuadNet-}\mu(\Phi, \mu)(\mathbf{x})$.

8.3 Numerical results

The proposed quadratic corrections are numerically tested on two different two-dimensional test cases: the steady-state backward-facing step flow, and the lid-driven cavity flow, both

with varying velocity magnitude at the inlet.

The domains, boundaries notation and computational grids associated with both test cases are represented in Figure 8.4. In both cases, the governing equations are the incompressible NSE (Equation (1.1)), in their steady-state form in the first test case.

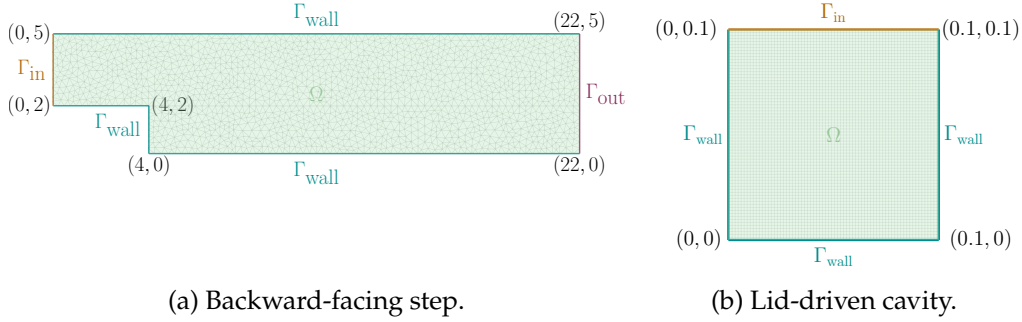


Figure 8.4: Domain and mesh considered in both the test cases.

In the backward-facing step flow case, the equations are complemented with no-slip conditions at the top and bottom boundaries, homogeneous Neumann conditions for the Cauchy stress tensor at the outlet, and Dirichlet conditions at the inlet, with a parabolic Poiseuille profile for the velocity. The full system of equations is given by

$$\begin{cases} \mathbf{u} = \mathbf{0} & \text{on } \Gamma_{\text{wall}}, \\ -p\mathbf{n} + \nu \nabla \mathbf{u} \cdot \mathbf{n} = \mathbf{0} & \text{on } \Gamma_{\text{out}}, \\ \mathbf{u} = \left(\frac{\mu}{2.25}(y-2)(5-y), 0 \right) & \text{on } \Gamma_{\text{in}}, \end{cases} \quad (8.15)$$

where we consider one parameter $\mu \in [1, 80]$, defining the amplitude of the parabolic profile. Moreover, y refers to the vertical component of the coordinates.

The full-order solution is computed using the FEM, implemented in the open-source Python package FEniCS [207], considering the velocity-pressure coupling $\mathbb{P}^2 - \mathbb{P}^1$ polynomial spaces. The domain of Figure 8.4a is discretized with 3091 triangular elements, and the discrete problem is solved for 500 values of μ . For each μ the discrete velocity is saved at the $N_u = 1639$ vertices of the triangulation.

In the lid-driven cavity test case, the boundary conditions here are non-slip conditions on Γ_{wall} and Dirichlet conditions on Γ_{in} , where we impose a constant, horizontal velocity, whose magnitude is given by a parameter. In particular:

$$\begin{cases} \mathbf{u} = \mathbf{0} & \text{in } \Omega \times \{0\}, \\ \mathbf{u} = \mathbf{0} & \text{on } \Gamma_{\text{wall}} \times [t_0, T], \\ \mathbf{u} = (\mu, 0) & \text{on } \Gamma_{\text{in}} \times [t_0, T], \end{cases} \quad (8.16)$$

where $t_0 = 0$ s, $T = 5$ s, $\nu = 10^{-5} \text{ m}^2 \text{ s}^{-1}$, and the parameter $\mu \in [0.5, 1] \text{ m s}^{-1}$ is the velocity magnitude at the top boundary. The Reynolds number for this problem can be defined as

$Re = \frac{\mu L}{\nu}$, where $L = 0.1$ m is the width (and height) of the domain. With this range of the parameter μ , the Reynolds number varies in the range $[5 \times 10^3, 10^4]$, which corresponds to a convection-dominated flow, in a transitional state from laminar to turbulent.

The lid-driven cavity flow is simulated using the **FVM** with the open-source software OpenFOAM [153, 230, 300]. The spatial domain is discretized with a uniform grid of $70 \times 70 \times 1$ hexahedral cells, and the time domain is discretized at steps $\Delta t = 1 \times 10^{-4}$ s.

It is important to stress that we do not consider time as a parameter. The dataset is indeed built by solving the problem for 150 values of μ and saving the snapshot at the final time 5 s. The velocity field is evaluated at the $N_u = 5041$ domain points.

The numerical results are structured as follows:

- ▶ A description of the hyperparameters of the different models is provided in 8.3.1;
- ▶ We analyze the performance of the novel corrections (8.3.2), considering fixed number of modes r_u and of snapshots N_μ (which we call *full* data regime);
- ▶ We also investigate the accuracy of our model in two challenging data regimes:
 - *partial* data (Section 8.3.3), where the models are trained with only a subset of the mesh nodes, while keeping fixed the values of r_u and N_μ ;
 - *scarce* data (Section 8.3.4), where the networks are trained considering a smaller set of parameters' instances N_μ , for different values of r_u .
- ▶ Analysis of the computational cost of all the models, in Section 8.3.5.

8.3.1 Setup of Corrected-ROMs

As briefly mentioned above, we investigate the novel **ROM** methodologies in different regimes, namely *full*, *partial*, and *scarce* data. The number of modes r_u , the number of training snapshots N_μ (used for both the **POD** and the **QuadNet** and **QuadNet- μ** trainings), and the percentage of spatial coordinates used to train the **QuadNet- μ** architecture (only in the partial data case) are outlined in Table 8.1.

Additionally, regarding the **RBF** interpolation used for the mapping $\mu \rightarrow \mathbf{a}$, we adopt a linear kernel for the backward-facing step, and a thin plate spline kernel for the lid-driven cavity test case. Moreover, once the **POD**, **RBF**, and the correction model are trained, we select 100 and 50 unseen μ instances (in the backstep and cavity test cases, respectively) to compare the methodologies. It is important to remark that we consider the same test instances in all the regimes (full, scarce, and partial data).

Regarding the **Quad-LS** methodology, the minimization problem (8.7) can be easily solved with the least squares solver implemented in PyTorch. However, we highlight that for high-dimensional problems (with larger number of degrees of freedom N_u), the size of tensor $\hat{\mathbf{Q}}$ may rapidly grow, leading to increasing computational cost and memory for the least-squares problem.

Table 8.1: Setup of all the ROMs considered, in different regimes (full, partial, and scarce data). The last column refers to the percentage of spatial coordinates considered for the training of QuadNet- μ .

Test case	Regime	N_μ	N_u	r_u	N_u fraction (QuadNet- μ training)
Backward-facing step	Full data	400	1639	3	100%
	Partial data	400	1639	3	[10, 20, 50]%
	Scarce data	[10, 20, 30, 40, 50, 100, 150, 200]	1639	[3, 5, 7, 9]	100%
Lid-driven cavity	Full data	100	5041	3	100%
	Partial data	100	5041	3	[10, 20, 50]%
	Scarce data	[10, 20, 30, 40, 50, 100]	5041	[3, 5, 7, 9]	100%

Table 8.2: Summary of the network parameters for the QuadNet model.

	Architecture				Training		
	Input	Hidden	Output	Activation	lr	Min. loss	Max. epochs
$\mathcal{B}$	r_u	7×20	s	Tanh	$O(10^{-3})$	10^{-2}	20000
$\mathcal{T}$	$d = 2$	7×20	s	Tanh			
$\mathcal{H}$	s	/	s	Tanh			

Concerning the QuadNet and QuadNet- μ models, for all the subnetworks (see the architecture of Figures 8.2 and 8.3) we adopt a simple feed-forward multi-layer perceptron structure, whose hyperparameters are summarized in Tables 8.2 and 8.3. Preliminary

Table 8.3: Summary of the network parameters for the QuadNet- μ model.

	Architecture				Training		
	Input	Hidden	Output	Activation	lr	Min. loss	Max. epochs
$\mathcal{B}_1$	r_u	7×20	s	Tanh	$O(10^{-3})$	10^{-2}	20000
$\mathcal{B}_2$	$P = 1$	7×20	s	Tanh			
$\mathcal{T}$	$d = 2$	7×20	s	Tanh			
$\mathcal{H}$	s	/	s	Tanh			

investigations showed that deeper branch and trunk networks lead to improved training convergence and model accuracy. For this reason, we consider 7 hidden layers of 20 neurons each for all the branch and trunk networks. The combine network is instead composed of a single layer which combines the outputs of branch and trunk networks to obtain the operator $\hat{\mathbf{Q}}$. We recall that $s = r_u \frac{r_u+1}{2}$, and that increasing latent dimensions r_u leads to large number of parameters of the NNs. This point, together with the additional computational cost required by the NNs training, is discussed in Section 8.3.5.

8.3.2 Full data

In the full data regime, the decay of the **POD** singular values obtained in both the test cases by applying the SVD to the velocity snapshots' matrix $\mathbf{S}_u$ is represented in Figure 8.5. The lid-driven cavity test case exhibits a slower decay, as it is convection-dominated and highly nonlinear. This is something expected a priori, since the range considered for the Reynolds number incorporates the laminar-to-turbulence transition regime.

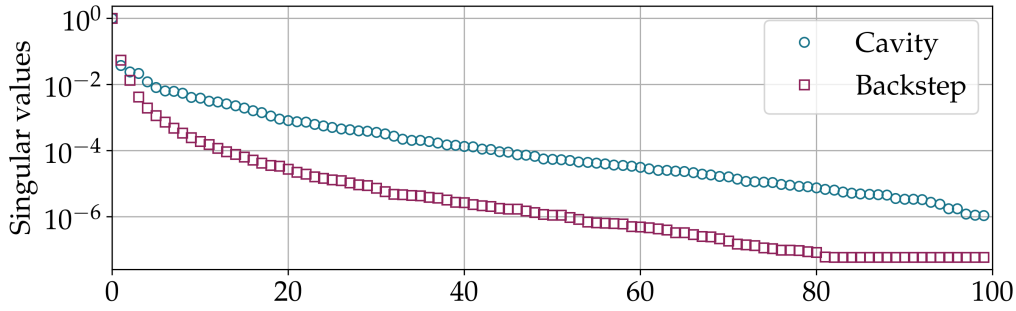


Figure 8.5: POD singular values decay in the full data regime, for both the test cases considered. The values are normalized to the first (largest) singular value.

We start comparing the models in terms of accuracy, both from a graphical point of view (Figures 8.6 and 8.7 for the backstep and cavity, respectively) and in terms of relative l^2 errors with respect to the **FOM** (Tables 8.4 and 8.5 for the backstep and cavity, respectively). In the graphical representation, we do not include the **QuadNet** as it performs worse than the **QuadNet- μ** model. More in detail, the figures show the **FOM** solution at an unseen parameter $\mu_\star$, and the graphical error fields $\mathbf{u}_r(\mathbf{x}, \mu_\star) - \mathbf{u}(\mathbf{x}, \mu_\star)$ obtained with the standard **POD-RBF**, and applying the corrections, namely employing **Quad-LS** and **QuadNet- μ** .

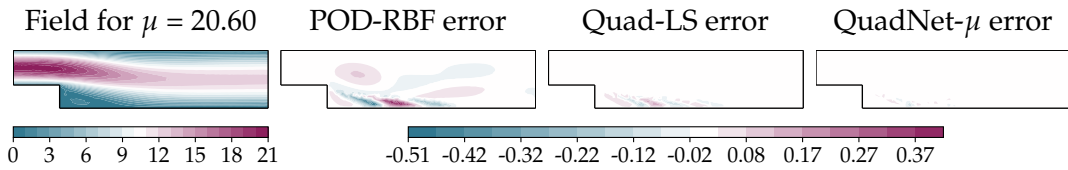


Figure 8.6: *Backward-facing step test case*. Velocity FOM solution, and errors of POD-RBF, Quad-LS and QuadNet- μ predictions with respect to the FOM, at an unseen parameter.

The tables show instead the relative l^2 errors:

$$E_u(\mu) = \frac{\|\mathbf{u}_r(\mathbf{x}, \mu) - \mathbf{u}(\mathbf{x}, \mu)\|_{l^2}}{\|\mathbf{u}(\mathbf{x}, \mu)\|_{l^2}}.$$

The terms “Avg”, “Median” and “Std” refer to the average, median and standard deviation across the parameters considered.

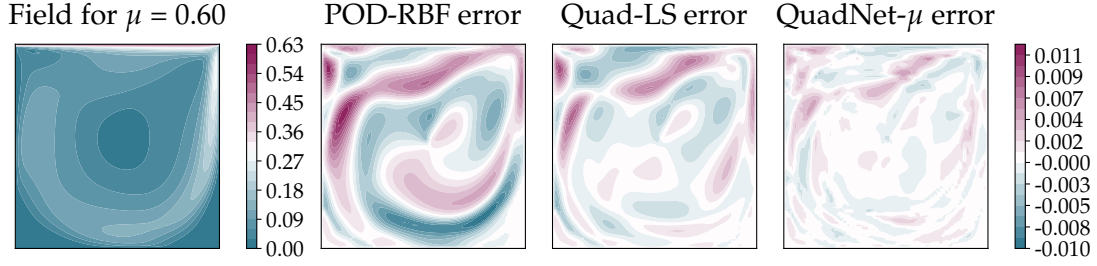


Figure 8.7: *Lid-driven cavity test case*. Velocity FOM solution, and errors of POD-RBF, Quad-LS and QuadNet- μ predictions with respect to the FOM, at an unseen parameter.

The results show that, in both test cases, the quadratic corrections improve the accuracy of the baseline linear ROM. The most accurate model is QuadNet- μ , which consistently outperforms both Quad-LS and QuadNet on the training and test datasets. This result is expected, since QuadNet- μ is local in the parameter space, whereas the other two models are global.

The benefits of a local model are particularly evident in the backstep test case, where both the average errors and their standard deviations are significantly smaller. Indeed, the flow exhibits markedly different behaviours across the parameter range considered, and the reduced variance achieved by QuadNet- μ suggests that it is better able to capture these variations than the other approaches. In contrast, this effect is less pronounced for the lid-driven cavity case. Although the flow is convection-dominated throughout the entire parameter range and is reproduced less accurately by a linear manifold, its overall structure remains relatively unchanged.

Furthermore, this space-continuous model can be trained using only a reduced set of selected sample points, leading to a more efficient training procedure while preserving a high level of accuracy, as will be discussed in Section 8.3.3.

Table 8.4: *Backward-facing step test case*. Comparison of models performance in terms of average, median and standard deviation of the velocity relative error.

	Train error			Test error		
	Avg(E_u)	Std(E_u)	Median(E_u)	Avg(E_u)	Std(E_u)	Median(E_u)
POD-RBF	0.015	0.023	0.0059	0.015	0.022	0.006
Quad-LS	0.005	0.013	0.0015	0.0033	0.0062	0.0015
QuadNet	0.006	0.012	0.0015	0.0060	0.0099	0.0015
QuadNet- μ	0.0014	0.0029	0.00036	0.0013	0.0023	0.00036

8.3.3 Partial data

In this section we investigate how the space-continuity of NN-based models can be leveraged to reduce the number of collocation points used during training, with the aim of

Table 8.5: *Lid-driven cavity test case*. Comparison of models performance in terms of average, median and standard deviation of the velocity relative error.

	Train error			Test error		
	Avg(E_u)	Std(E_u)	Median(E_u)	Avg(E_u)	Std(E_u)	Median(E_u)
POD-RBF	0.030	0.010	0.027	0.033	0.010	0.030
Quad-LS	0.0154	0.0055	0.014	0.0176	0.0052	0.015
QuadNet	0.0154	0.0062	0.013	0.0182	0.0057	0.017
QuadNet- μ	0.0091	0.0035	0.0086	0.0108	0.0032	0.010

lowering the computational cost of the offline phase. Since **QuadNet- μ** has shown the best overall performance, we focus on this model.

Restricting the training data to a subset of collocation points introduces an additional challenge: selecting an appropriate subset from the full mesh. A straightforward approach is to sample points uniformly at random across the domain, ensuring broad coverage. While this strategy is generally effective, an analysis of the correction fields shows that they are often highly localized, being non-negligible only in small regions of the domain. This suggests that uniform sampling may be suboptimal, as it does not prioritise regions where the correction is large or rapidly varying, while oversampling areas where it is nearly zero.

To address this issue, we propose a sampling strategy based on a Boltzmann-like probability distribution assigned to each mesh node. Specifically, the probability of selecting the i -th node is defined as

$$\pi_i \propto \exp(-\tilde{\tau}/(\bar{\tau}_i + \varepsilon)), \quad (8.17)$$

where $\bar{\tau}_i = \frac{1}{N_\mu} \sum_{j=1}^{N_\mu} \tau^{\text{true}}(\mathbf{x}_i, \mu_j)$ denotes the average correction at node i over all training snapshots, $\tilde{\tau}$ is a suitable normalization factor (such as the mean or maximum norm of the corrections), and ε is a small positive constant introduced to avoid division by zero. This construction increases the probability of selecting nodes associated with larger corrections, while reducing the likelihood of sampling regions where the correction is typically small. A comparison between uniform and Boltzmann-like sampling strategies is provided in Figure 8.8, where the proposed approach is shown to achieve the intended effect.

The applicability of **QuadNet- μ** in the 10, 20, and 50% partial data regimes is investigated in terms of relative errors in Figure 8.9, which shows the median and the 5 – 95% quantiles. As expected, the error decreases as the number of collocation points increases, and the most accurate result is achieved when the full mesh is considered. However, it is important to highlight that in the backstep test case, the 50%-partial data configuration improves the **Quad-LS** accuracy. In the lid-driven cavity case, the difference between the three partial data configurations is not as large as for the backstep. This is likely due to the complexity of the solution, where the local features are not captured if less collocation points are considered. It is however important to remark that:

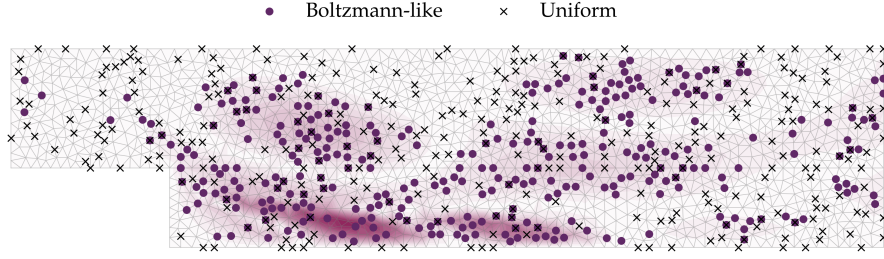


Figure 8.8: *Backward-facing step test case*. Comparison between samples of points drawn from different distribution. The **purple** points are sampled from a Boltzmann-like distribution, while the black crosses are sampled from a uniform distribution. A correction field is plotted in the background for reference.

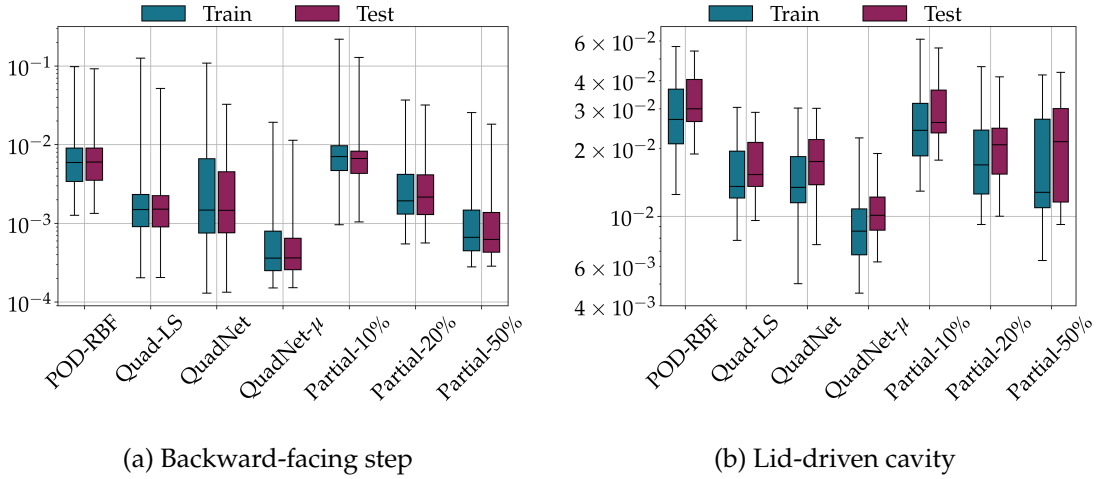


Figure 8.9: Boxplots comparing the accuracy of the models considered in both the test cases. The label *Partial data* ($x\%$) refers to the results obtained using QuadNet- μ with the $x\%$ training snapshots.

- it is possible to choose a more effective sampling strategy to select the collocation points, but this goes beyond the scope of this work;
- considering 10% of the mesh nodes reduces the training time of roughly 10% with respect to the full data configuration. We expect larger time saving in the case of larger N_u .

8.3.4 Scarce data

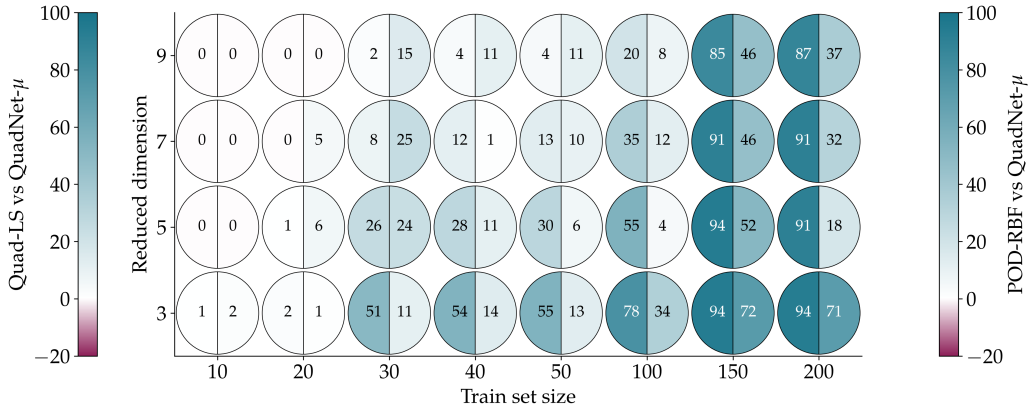
Finally, we test the model's performance when data is scarce, i.e., with smaller N_μ values, and for different latent dimensions r_u . This scenario is particularly relevant in real-world applications, where performing a large number of FOMs is computationally prohibitive.

We focus on the best-performing model of the previous experiments (QuadNet- μ), and measure the reduction of the relative error with respect to the baseline methods, i.e., **POD-RBF** and **Quad-LS**. Just as in the previous chapter (see Equation (7.8)), this *gain* variable

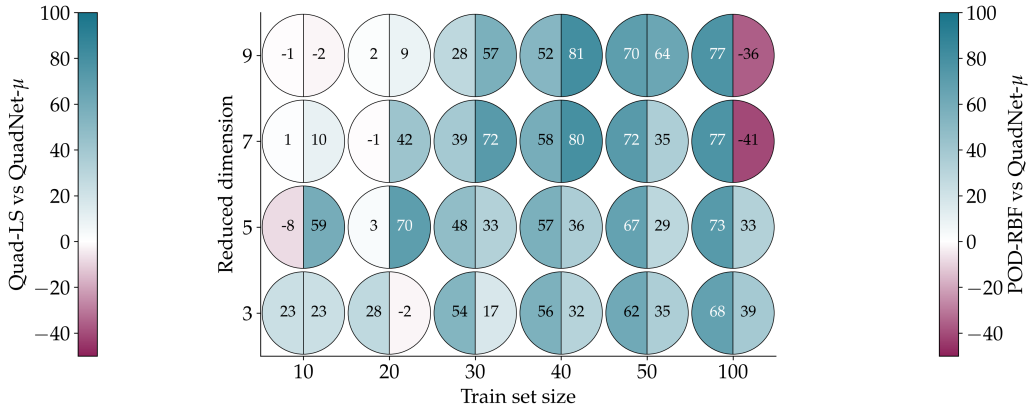
is computed as:

$$\frac{1}{N_{\mu}^{\text{test}}} \sum_{i=1}^{N_{\mu}^{\text{test}}} \frac{E_{\mathbf{u}}^{\text{base}}(\mu_i) - E_{\mathbf{u}}^{\text{QuadNet-}\mu}(\mu_i)}{E_{\mathbf{u}}^{\text{base}}(\mu_i)},$$

where N_{μ}^{test} is the number of test parameters, and the superscript “base” refers either to **POD-RBF** or to **Quad-LS**. Figure 8.10 quantifies the gain metric with respect to both baseline models and both test cases.



(a) Backward-facing step.



(b) Lid-driven cavity.

Figure 8.10: Comparison of QuadNet- μ errors with POD-RBF and Quad-LS. The left half of the circles shows the relative difference in error between QuadNet- μ and POD-RBF, while the right half shows the difference between QuadNet- μ and Quad-LS.

We find that **QuadNet- μ** outperforms the baseline models in almost all cases, often reducing the error by more than 90%. A few exceptions can be observed, mainly for configurations with a small number of training snapshots and a large number of **POD** modes. In these situations, two factors may limit the effectiveness of the quadratic correction:

- ▶ when only a few snapshots are available, the **POD** basis may not provide an accurate representation of the solution manifold, and the quadratic correction can amplify the resulting approximation errors;

- ▶ as the number of modes approaches the number of training snapshots, the **POD** reconstruction of the training data becomes nearly exact, causing the correction fields to become very small. Consequently, the network learns corrections of negligible magnitude, and the resulting quadratic **ROM** behaves similarly to the linear one. This is something already observed also in Chapter 7 (see for example Figure 7.7).

Compared with **Quad-LS**, **QuadNet- μ** is generally more accurate, with only a few exceptions occurring for large training datasets and relatively high-dimensional reduced spaces. As the number of POD modes increases, the neural networks become larger and more difficult to train, making the results more sensitive to hyperparameter choices and weight initialization. Nevertheless, even in the cases where **QuadNet- μ** does not achieve the lowest error, it typically remains competitive and often still outperforms **Quad-LS**.

As expected, all models benefit from larger training datasets and richer reduced spaces. However, **Quad-LS** tends to struggle when data are scarce, producing substantially larger errors than both **POD-RBF** and **QuadNet- μ** . As more training snapshots become available, the performance of both the quadratic approaches improves, with **QuadNet- μ** outperforming **Quad-LS** in most configurations. Furthermore, **QuadNet- μ** appears to be more robust, exhibiting a smaller spread in prediction errors, as evidenced by the narrower percentile bands shown in 8.10. Although **Quad-LS** occasionally achieves slightly lower errors for specific parameter combinations, the differences remain small and the overall trends consistently favour **QuadNet- μ** .

8.3.5 Computational cost

We can now perform a comparison among the three quadratic models in their computational cost.

As already discussed, one of the main distinctions between **Quad-LS** and the **NN**-based approaches is that the former is discrete in space, whereas the latter provide a continuous representation. Consequently, the quadratic operator in **Quad-LS** is characterized by $s \times N_\mu$ parameters, corresponding to approximately 10 000 parameters for the backward-facing step and 30 000 for the lid-driven cavity. In contrast, from Tables 8.2 and 8.3 we find that **QuadNet** and **QuadNet- μ** contain approximately 5700 and 8400 trainable parameters, respectively, making both models effectively smaller than **Quad-LS**.

A smaller model is advantageous from a memory standpoint, particularly for large-scale applications where the computational mesh is substantially larger than in the present test cases. However, in our experiments this reduction in model size did not translate into noticeable computational savings. While the online evaluation costs of the three approaches are comparable, the offline training phase is considerably more expensive for the **NN**-based models. The training time² of **QuadNet** and **QuadNet- μ** ranges from approximately 8 to 25 minutes, depending on the network size and the amount of training

²The machine is a workstation equipped with an 11th Gen Intel(R) Core(TM) i7-11700 CPU, 32 GB of RAM, and an NVIDIA Quadro RTX 4000 GPU with 8 GB of memory.

data. In contrast, training **Quad-LS** requires only a few milliseconds, typically less than 10.

Nevertheless, this additional offline cost becomes much less significant when compared with the cost of generating full-order simulations. For instance, in the lid-driven cavity test case, a single full-order simulation requires approximately 3 minutes. Therefore, the time needed to train a neural network corresponds to the cost of generating only about 3 to 8 additional full-order solutions, depending on the value of r_u . In practice, if **Quad-LS** and **QuadNet- μ** are trained on the same dataset, the extra training cost of **QuadNet- μ** can be amortized after only 3 to 8 additional online queries. In a realistic many-query setting, such an overhead is likely to be negligible.

8.4 Conclusions

In this chapter, based on the work of [68], we introduced a new class of non-intrusive **ROMs** based on neural operator architectures, with the goal of overcoming some of the limitations of classical linear approaches. In particular, we proposed two neural-network-based models, **QuadNet** and **QuadNet- μ** , built upon the **DeepONet** and **MIONet** frameworks, respectively, to learn quadratic corrections to an existing surrogate model. The proposed approaches consistently improved the accuracy of standard **POD-based ROMs** and of the baseline quadratic correction methodology (**Quad-LS**) [251, 106]. Numerical experiments on benchmark fluid dynamics problems showed error reductions of up to 90% with respect to **POD-RBF** and up to 80% with respect to **Quad-LS**, while requiring a **limited number of training samples** and relatively **compact** models. These results highlight the ability of neural operators to effectively represent complex nonlinear dynamics and correction fields. Furthermore, the **continuous dependence** of the proposed models on both the spatial coordinates and the physical parameters provides additional flexibility and generalization capabilities to practical scenarios (like scarce or partial data availability).

Remark: parallelism between **DD-EV-ROM** (Chapter 7) and the **Corrected-ROM** (Chapter 8).

The work presented in this chapter is the natural nonintrusive extension of the approach of Chapter 7. Table 8.6 highlights similarities and differences among the models, in terms of the exact correction we want to learn, and the correction map used to model it. For the **Corrected-ROM** we focus on the parametrized (and more accurate) map **QuadNet- μ** .

Table 8.6: Comparison between **DD-EV-ROM** and **Corrected-ROM**.

		DD-EV-ROM	Corrected-ROM
τ^{true}	Definition	$\left[\begin{array}{c} \overline{\mathcal{R}_{d_u}^{r_u}} - \mathcal{R}_{r_u} \\ \overline{\mathcal{R}_{d_p}^{r_u}} - \mathcal{R}_{r_p} \end{array} \right]$	$\mathbf{u} - \mathbf{u}_r$
	Dimension	$r_u + r_p$ (<i>reduced</i>)	N_u (<i>full</i>)
Correction map	Notation	$\mathcal{M}$	QuadNet- μ
	Architecture	MIONet (Figure 7.4)	MIONet (Figure 8.3)
	Input	$\mathbf{a}, \mathbf{g}, \mu$	$\phi_1, \dots, \phi_{r_u}, \mathbf{x}, \mu$
	Input dim.	$r_u + r_{v_t} + P$ (<i>reduced</i>)	$r_u + d + P$ (<i>reduced</i>)
	Output	$\tau \approx \tau^{\text{true}}$	$\hat{\mathbf{Q}}$ s.t. $\hat{\mathbf{Q}}\tilde{\mathbf{a}} \approx \tau^{\text{true}}$
	Output dim.	$r_u + r_p$ (<i>reduced</i>)	$r_u \frac{r_u+1}{2}$ (<i>reduced</i>)

The table shows that the definition of the exact correction is different among the models, as it is a classical *closure* approach in DD-EV-ROM and a simple discrepancy with respect to the ground truth in the Corrected-ROM. Moreover, there is a difference in the dimension of the correction itself. On the one hand, in the intrusive approach it is an extra-term added into the reduced system, and has indeed a *reduced* dimension. On the other hand, in the nonintrusive approach it directly corrects the ROM approximation and we need indeed *full* dimensionality. Finally, in the intrusive approach the output of the neural network is the correction itself, while in the nonintrusive model, the output is the quadratic operator. This allows the nonintrusive model to keep a quadratic form, while the intrusive one adopts a fully nonlinear function.

8.4.1 Conclusions to Part II and motivations to Part III

In Part II, we have presented three projects aimed at enhancing the accuracy of existing ROMs using different filtering and closure (or correction) approaches. The use of machine-learning allowed to build nonlinear functions able to accurately model stabilization operators (in the StabOp case of Chapter 6) and closure terms (in Chapters 7 and 8). In all three projects presented in Part II, the starting point is a standard POD-based ROM. Such models are supported by a well-established mathematical framework, which provides rigorous results on approximation properties, stability, and error estimation. The machine-learning components introduced in Chapters 6, 7, and 8 are not intended to replace this framework, but rather to augment it through nonlinear stabilization, closure, or correction operators.

However, the introduction of data-driven components also raises important theoretical and practical challenges. Once machine-learning-based terms are added, either at the

level of the reduced dynamical system or directly to the reconstructed solution, the behaviour of the resulting model becomes significantly harder to analyse. In general, classical guarantees inherited from projection-based reduced-order modeling no longer apply automatically. For instance, the stability of the reduced system may be compromised, as observed in the original **DD-EV-ROM** formulation when the correction map is trained independently of the turbulence map. Likewise, rigorous error bounds become difficult to establish, since the overall approximation now depends on the interaction between the projection-based model and a learned nonlinear correction.

Moreover, these approaches remain fundamentally constrained by the underlying **POD** representation. The machine-learning components can compensate for modeling errors and missing dynamics, but they cannot overcome the intrinsic limitations of the reduced basis. In the case of **DD-EV-ROMs**, for example, the reconstruction error associated with the **POD** projection represents a lower bound on the achievable approximation accuracy. Consequently, when the **POD** basis itself provides a poor representation of the solution manifold—as may occur in complex real-world applications, in the presence of scarce training data, or when only partial observations are available—even sophisticated closure models may not be sufficient.

These considerations have motivated a growing body of recent research devoted to surrogate modeling approaches in which **POD** is no longer regarded as the fundamental representation tool. Instead, nonlinear machine-learning architectures are employed to learn low-dimensional latent representations directly from data. Such methods offer the potential to overcome some of the limitations of linear subspace approximations and constitute the focus of Part III.

PART III

TOWARDS NONLINEAR REDUCED ORDER MODELS

NONLINEAR ROMs AND CONVOLUTIONAL ARCHITECTURES: LITERATURE REVIEW

Although the **POD** provides an optimal low-rank approximation in the sense of the L^2 -projection onto linear subspaces and it relies on a strong mathematical theory, its accuracy is fundamentally constrained by the complexity of the underlying solution manifold $\mathcal{A}$.

From an approximation theoretic perspective, the **POD** constructs the best possible approximation within a linear subspace of pre-selected dimension n . Consequently, the performance of **POD** is fundamentally linked to the Kolmogorov n -width of the solution manifold $\mathcal{A}$ [69, 214, 255]. The n -width quantifies the best possible approximation error achievable by any n -dimensional linear subspace $\mathbb{V}$ and is defined as:

$$d_n(\mathcal{A}) = \inf_{\dim(\mathbb{V})=n} \sup_{u \in \mathcal{A}} \inf_{v \in \mathbb{V}} \|u - v\|. \quad (9.1)$$

When the Kolmogorov n -width decays rapidly, the solution manifold is well approximated by a low-dimensional linear space, and **POD** yields efficient **ROMs**. However, in many nonlinear and parameter-dependent **PDEs**, the Kolmogorov width decays slowly, indicating that any accurate representation would require a prohibitively large number of basis functions. This would make the reduced space high-dimensional and therefore computationally expensive or even infeasible in practice due to the cost of generating sufficiently many snapshots.

This intrinsic limitation motivates the development of **nonlinear reduction strategies**. Several nonlinear approaches for **ROMs** have been explored in the past years. We list here a far-from-exhaustive list of nonlinear reduction approaches:

- ▶ **nonlinear manifold approximations**, like quadratic manifolds [151, 23, 106], briefly recalled in Chapter 8, polynomial manifolds [105], kernel **POD** [299];
- ▶ **geometric transformation-based methods**, like registration methods [326, 90], shifted **POD** [270, 246], optimal transport [166], the method of freezing [236];
- ▶ **manifold interpolation methods**, such as Grassmann or geodesic interpolation [12, 13];
- ▶ **data-driven nonlinear maps**, like **AEs** [132, 347].

In this thesis, we only focus on the last class, i.e., autoencoders, which will be the basis to construct the model of Chapter 10. The remainder of this chapter is organized as follows:

- ▶ a brief description and state-of-the-art review of **AE**-based model reduction methodologies (Section 9.1);
- ▶ a brief section recalling the structure of a **convolutional** layer and of its continuous extension, and how it is employed in **AEs** (Section 9.2);
- ▶ a final section motivating the novel approach for nonlinear model order reduction of Chapter 10.

9.1 Nonlinear dimensionality reduction via AutoEncoders

In this section, we revise the most common **AE**-based methodologies for model order reduction, which are schematized in Figure 9.1.

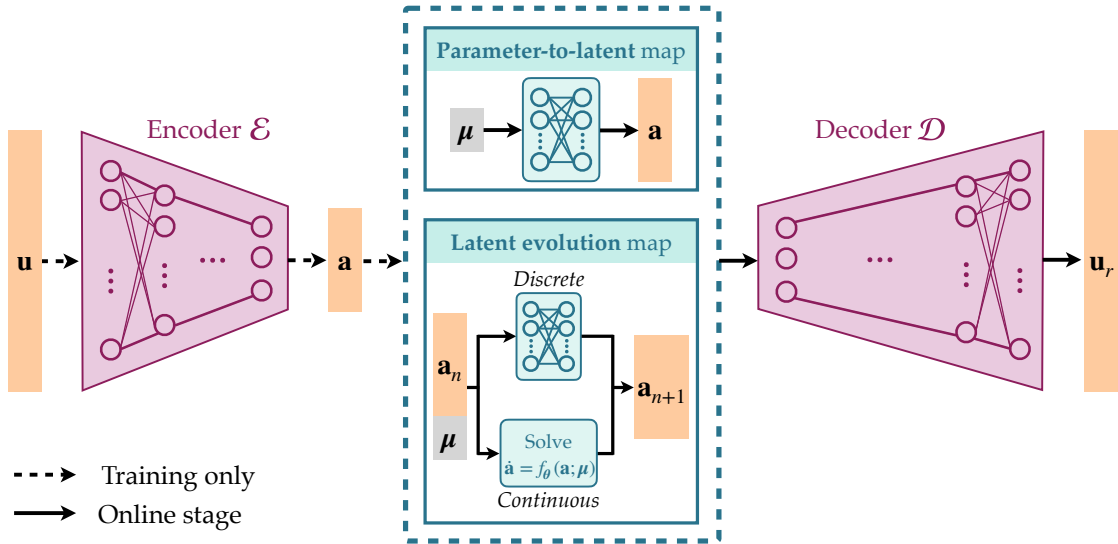


Figure 9.1: An illustrative scheme of the most common nonlinear model order reduction methodologies.

AEs have emerged as one of the most successful approaches for nonlinear reduced-order modelling. Unlike linear projection methods, **AEs** learn a nonlinear low-dimensional manifold directly from data through two parametric mappings: an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. The encoder $\mathcal{E} : \mathbb{R}^N \rightarrow \mathbb{R}^r$ compresses a high-dimensional state $\mathbf{u} \in \mathbb{R}^N$ into a set of r latent coordinates $\mathbf{a} = (a_i)_{i=1}^r$, while the decoder $\mathcal{D} : \mathbb{R}^r \rightarrow \mathbb{R}^N$ reconstructs the original solution from the latent representation. Formally, we can write:

$$\mathbf{u} \approx \mathbf{u}_r = \text{AE}(\mathbf{u}) = (\mathcal{D}_\varphi \circ \mathcal{E}_\theta)(\mathbf{u}),$$

where φ and θ are the network parameters that are jointly optimized by minimizing a reconstruction loss over a set of N_μ training snapshots, typically:

$$\mathcal{L}_{\text{AE}}(\varphi, \theta) = \frac{1}{N_\mu} \sum_{i=1}^{N_\mu} \|\mathbf{u}(\mu_i) - (\mathcal{D}_\varphi \circ \mathcal{E}_\theta)(\mathbf{u}(\mu_i))\|^2. \quad (9.2)$$

The simplest architecture of an AE consists of two fully-connected multilayer perceptrons (MLPs) (as represented in Figure 9.1), one implementing the encoder and the other the decoder, as introduced in [132]. While several architectural variants have been proposed, including convolutional AEs [193], graph [122] and graph convolutional AEs [254], fully-connected networks remain the standard choice for problems where the state is naturally represented as a vector. Convolutional architectures, which better exploit spatial locality and translational invariance, will be discussed in the following section.

Since the decoder is responsible for reconstructing a high-dimensional nonlinear solution manifold from a compressed latent representation, it often requires a larger expressive capacity than the encoder, which is commonly achieved through deeper or wider architectures (see Figure 9.1). Conversely, several works have investigated constrained autoencoder architectures, such as with symmetric, orthogonal, or biorthogonal encoder-decoder weights [243, 44, 199]. Rather than increasing the expressive power of the network, these approaches introduce architectural constraints to improve properties such as invertibility and interpretability. These additional constraints, however, generally lead to a more difficult optimization problem and may hinder the convergence of the training procedure.

Once the nonlinear manifold has been learned, an important question naturally arises: *How can the latent coordinates be computed during the online stage, when the full-order solution is no longer available?*

During the training stage, this issue does not arise explicitly, since the latent representation is readily obtained by encoding high-fidelity snapshots through the encoder. However, in the online phase, the encoder can no longer be used, as the full state is precisely the unknown quantity that the ROM aims to approximate. Consequently, an additional mechanism is required to estimate or evolve the latent variables using only accessible information, such as the initial condition, system parameters, or past latent states (in the case of time-dependent problems). From this perspective, three main classes of latent coordinate mappings have been proposed in the literature (as represented in Figure 9.1):

- **Parameter-to-latent maps**, where a mapping

$$\mu \mapsto \mathbf{a}$$

is learned via an interpolation or regression-type map. This approach is typical of nonintrusive surrogate models, where the goal is to learn a direct mapping from the parameter space to the latent manifold. Classical strategies include RBF interpolation, GPR [121], neural network regressors [127], operator-learning approaches such as DeepONet [210] and other neural operators when interpreted in reduced coordinates. Examples of works applying such methods in AE-based ROMs are [101, 100, 98].

- **Discrete-latent time-evolution maps**, where the evolution of the system for time-dependent problems can be modelled directly through the mapping

$$(\mathbf{a}_n, \boldsymbol{\mu}) \mapsto \mathbf{a}_{n+1}.$$

In this way, the original high-dimensional evolution is replaced with a learned map on the reduced coordinates. This class includes autoregressive neural networks, recurrent architectures such as Long-Short Term Memory networks (LSTMs) [221, 306], as well as more structured dynamical representations such as Koopman-based models, DMD variants [298, 337, 184], and sparse identification of nonlinear dynamics [47]. Examples of works exploiting these approaches in model order reduction are [344, 244, 163, 71, 231].

- **Continuous-latent time-evolution maps**, where the underlying vector field $f_\theta(\mathbf{a}, \boldsymbol{\mu})$ governing the latent evolution is directly learned, and the reduced-order ODE

$$\dot{\mathbf{a}} = f_\theta(\mathbf{a}, \boldsymbol{\mu})$$

is solved. The latent trajectory is then obtained by numerical integration of this learned model. This perspective is closely related to neural ODEs and their extensions to parameterized and latent dynamical systems [61, 194]. Examples of application of such methods in the field of model order reduction are [274, 195, 201, 88, 89, 331].

9.2 Convolutional architecture

While fully-connected AEs treat the input state as a one-dimensional vector, many reduced-order modelling applications involve solutions naturally defined on one-, two-, or three-dimensional spatial domains. In these cases, flattening the solution into a vector neglects the underlying spatial organization and local correlations of the physical field of interest. CNNs overcome this limitation by exploiting local connectivity and weight sharing, allowing the network to extract hierarchical spatial features while drastically reducing the number of trainable parameters.

A **convolutional layer** applies a set of learnable filters (or kernels) that are convolved with the input feature maps. We consider an input tensor $\mathbf{S} \in \mathbb{R}^{C_{\text{in}} \times H_{\text{in}} \times W_{\text{in}}}$, which represents an image of dimension H_{in} (height) $\times W_{\text{in}}$ (width) with C_{in} input channels. For example, in the case of a red-green-blue (RGB) image, we can naturally distinguish 3 input channels corresponding to the three main colours, while in the case of a d -dimensional vector field $C_{\text{in}} = 2$. We also consider a convolutional kernel $\mathbf{K} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times H_K \times W_K}$, where H_K and W_K are the height and width of the kernel, and C_{out} is the desired number of output channels. To simplify the notation we indicate as $\mathbf{S}^{(c)} \in \mathbb{R}^{H_{\text{in}} \times W_{\text{in}}}$ the c -th channel of the input (adopting the same notation for the output), while we denote as $\mathbf{K}^{(d,c)} \in \mathbb{R}^{H_K \times W_K}$ the kernel matrix corresponding to the input channel c and to the output channel d . The

application of a convolutional layer with kernel $\mathbf{K}$ on the input feature $\mathbf{S}$ produces the output:

$$\mathbf{Y}^{(d)} = h \left(\sum_{c=1}^{C_{\text{in}}} \left(\mathbf{K}^{(d,c)} * \mathbf{S}^{(c)} + b^{(d)} \right) \right), \quad (9.3)$$

where $b^{(d)}$ is the scalar bias associated with the d -th output channel, h is the nonlinear activation function, and the symbol $*$ indicates a convolution (or *cross-correlation*) operation, namely:

$$\left(\mathbf{K}^{(d,c)} * \mathbf{S}^{(c)} \right)_{i,j} = \sum_{p=1}^{H_K} \sum_{q=1}^{W_K} K_{p,q}^{(d,c)} S_{i+p,j+q}^{(c)}. \quad (9.4)$$

An example of discrete convolution is represented in Figure 9.2a. The convolutional layer is characterized by four hyperparameters: kernel size, stride, padding, and dilation. The kernel size defines the size of the local region considered by each convolution operation, whereas the stride determines how the kernel moves across the input and controls the resulting spatial resolution. The padding defines how boundary conditions are handled during convolution, ensuring either dimensional preservation or controlled reduction of the spatial domain. Finally, the dilation allows for an enlarged receptive field without increasing the number of parameters, enabling the capture of multi-scale spatial features.

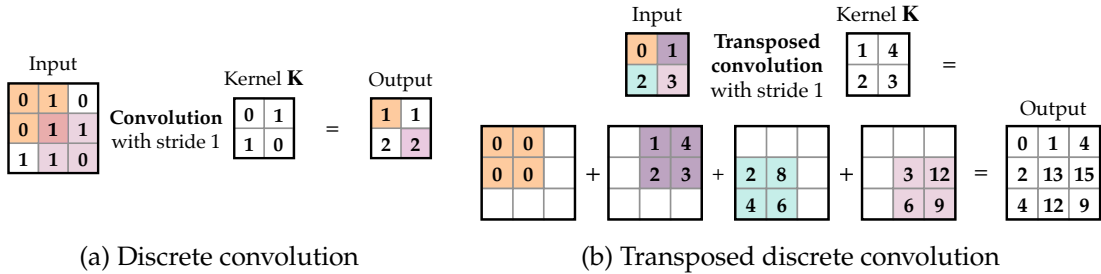


Figure 9.2: Examples of direct and transposed discrete convolution operations.

CNNs are constructed by composing multiple convolutional blocks, each typically consisting of a convolutional layer followed by a nonlinear activation. In addition, **CNN** architectures typically employ pooling operators, strided convolutions, and upsampling layers to control the spatial resolution of feature maps. Pooling operations, such as *max* or *average* pooling, provide fixed downsampling mechanisms, while strided convolutions offer a learnable alternative. These components are often complemented by normalization layers to stabilize the training process and improve convergence.

Architectures such as **LeNet**, [192] **AlexNet** [178], **VGGNet** [309], **ResNet** [125], and **U-Net** [275] have progressively demonstrated the effectiveness of convolutional structures.

The convolutional layers may also be used into an **AE** architecture for dimensionality reduction in high-dimensional spatial problems, as they exploit local correlations and hierarchical structure in the data. The resulting model is called **Convolutional AutoEncoder**

(CAE) and typically contains a sequence of convolutional layers and activations (encoder), and of transposed convolutional layers and activations (decoder). The **transposed convolutional** layers are indeed used to perform a learnable upsampling, as represented in Figure 9.2b. Unlike standard convolutional layers, which map high-resolution inputs to lower-dimensional feature maps, transposed convolutions propagate information from a low-dimensional representation to a higher-resolution output by distributing each input activation over a larger spatial region. This operation can be interpreted as the transpose of a linear convolution operator and is widely employed in convolutional autoencoders for reconstructing high-dimensional physical fields.

Moreover, fully-connected layers are often introduced after a flattening operation that converts convolutional feature maps into one-dimensional vectors, thereby enabling global interactions among all extracted spatial features, particularly useful in classification tasks.

While CNNs are widely exploited for image-like data, their direct application in CFD settings typically relies on the assumption that the input field is defined on a structured mesh, so that it can be represented as a regular tensor with a well-defined spatial ordering. This requirement constitutes a limitation in practical engineering applications, where simulations are often performed on unstructured meshes. In such cases, the raw solution must be remapped onto a structured grid before being processed by convolutional architectures, potentially introducing interpolation errors and additional computational cost.

The following part presents the **continuous** version of convolutional layers [73], which can then be used for unstructured inputs.

9.2.1 Continuous Convolutional architecture

We here recall the continuous convolutional layer model, which was recently proposed in [73].

A **continuous convolutional** layer extends the classical convolutional operation by replacing discrete convolutional kernels with **learnable continuous functions**. To simplify the notation, we consider here a two-dimensional convolution, a scalar input function, i.e., $C_{\text{in}} = d = 2$, and a scalar output function, i.e., $C_{\text{out}} = 1$. At first, we consider an irregular grid of coordinates $\{\mathbf{x}_j\}_{j=1}^N = \{(x_j, y_j)\}_{j=1}^N \in \Omega \subset \mathbb{R}^2$, being N the number of degrees of freedom. In contrast with the discrete convolution defined in (9.4), the continuous convolution (considering $d = 2$ dimensions) can be defined as:

$$\mathcal{Y}(x, y) = (\mathcal{S} * \mathcal{K})(x) = \int_{\mathbb{X}} \int_{\mathbb{Y}} \mathcal{S}(x + \tau_x, y + \tau_y) \cdot \mathcal{K}(\tau_x, \tau_y) d\tau_x d\tau_y, \quad (9.5)$$

where $\mathcal{S} : \Omega \rightarrow \mathbb{R}$ represents the continuous version of the input tensor $\mathbf{S}$ defined into the domain Ω . $\mathcal{K} : \mathbb{X} \times \mathbb{Y} \rightarrow \mathbb{R}$ is the continuous kernel. Note that $\mathbb{X} \times \mathbb{Y} \neq \mathbb{R}^2$, as the filter is defined on a smaller domain than the input function. In particular, we typically consider $\mathbb{X} = [0, 1]$ and $\mathbb{Y} = [0, 1]$. The continuous kernel is approximated using a trainable MLP,

while the integral can be approximated as a discrete sum, as follows:

$$\mathcal{Y}(\tilde{x}_i, \tilde{y}_i) = \sum_{x_i \in \mathbb{X}} \sum_{y_i \in \mathbb{Y}} \mathcal{S}(x_i + \tau_x, y_i + \tau_y) \cdot \mathcal{K}(x_i, y_i), \quad (9.6)$$

where (τ_x, τ_y) are the strides positions, while $(\tilde{x}_i, \tilde{y}_i)$ are the points obtained by taking the centroid of the filter position mapped onto the Ω domain. A schematic illustration of the continuous convolutional layer is represented in Figure 9.3 [73].

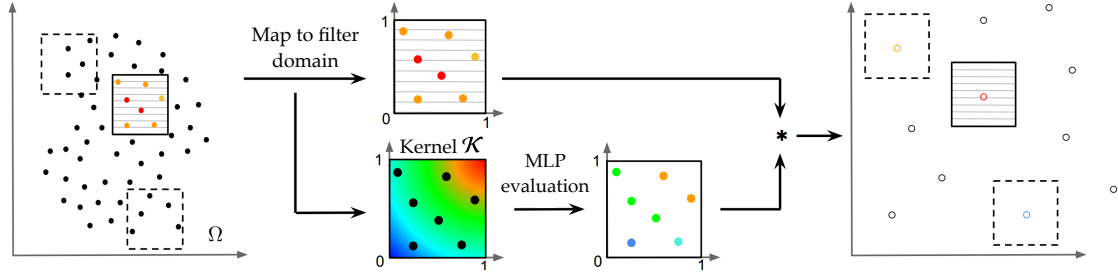


Figure 9.3: Sketch of the continuous convolutional operation [73]: the unstructured points are mapped into the filter domain $\mathbb{X} \times \mathbb{Y} = [0, 1]^2$, then a MLP kernel $\mathcal{K}$ is used to approximate the filter values, and finally a convolution between the mapped input values and the filter values is computed.

We also remark some advantages and computational details of the continuous convolutional layer [73]:

- ▶ it works for a general domain Ω , e.g., the connectivity is not a requirement;
- ▶ the stride positions do not have to cover the entire domain;
- ▶ the result can be generalized to **different grids** as the model itself is space-continuous;
- ▶ the implementation is optimized by stacking the mapped input, so that the **MLP** is called only once before the for loop.

However, as we will see in Chapter 10, training a **Continuous Convolutional Neural Network (CCNN)** is typically more time-demanding than training a discrete **CNN**. The reason lies in the mapping function used to compute the points $(\tilde{x}_i, \tilde{y}_i)$. It is here implemented using linear search, but the use of more advanced algorithms based on efficient data structures may decrease the training time.

The question that may automatically arise is:

How is the continuous convolutional layer employed in the field of model order reduction?

As in [73], it can be adopted in combination with both standard convolutional layers and **MLP** architectures to form a **Continuous Convolutional AutoEncoder (CCA)**. Moreover, the transposed convolutional operation may be trivially implemented within the decoder part [83]. Differently from standard **CAEs**, the presence of a first continuous convolutional layer in the encoder (and of the transposed continuous convolution as a last layer of the decoder) allows the **CCA** to be applied to solutions defined on different grids.

9.3 Motivation for the proposed approach

As briefly outlined in this chapter, several researchers are working on *nonlinear ROMs*. However –to the best of our knowledge– most of the existing works focus on applying a nonlinear compression directly on the solution field, e.g. in a purely data-driven fashion, without employing the full-order operators. The question that we pose ourselves is then: *Is it possible to define a nonlinear transformation, as a nonlinear generalization of POD-Galerkin approaches, that allows to compress full-order operators?*

We propose an *operator compression* framework in the field of nonlinear model order reduction by exploiting neural networks to compress the full-order operators.

For instance, we consider a full-order operator $\mathbf{A} \in \mathbb{R}^{N \times N}$, where N is the number of FOM degrees of freedom.

In POD-Galerkin approaches, the corresponding reduced operator is computed as $\mathbf{A}_r = \mathbf{V}^T \mathbf{A} \mathbf{V}$, where $\mathbf{V} \in \mathbb{R}^{N \times r}$ and r is the pre-selected reduced dimension.

We propose to replace the linear projection with a nonlinear **Convolutional Encoder** (CE) mapping $\mathbf{A}_r = \text{CE}(\mathbf{A})$. Once all the reduced operators are nonlinearly compressed, we can solve the reduced equations with standard techniques (linear or nonlinear solvers), and finally decode the reduced solution into the full-order space with a neural network.

After introducing the nonlinear mappings required by the proposed framework, several natural questions arise.

Why learning reduced operators instead of directly learning the solution?

Unlike most nonlinear ROMs, which directly learn a low-dimensional representation of the solution field, our goal is to learn a reduced representation of the operators defining the governing equations. This allows the reduced model to retain the algebraic structure of the original problem: rather than replacing the solver with a neural network, we still solve a reduced system of equations. Consequently, the resulting ROM remains **interpretable** and closer in spirit to projection-based methods.

Can the structural properties of the full-order model be preserved?

Since the compression is performed at the operator level, it is possible to design the nonlinear mappings so that important mathematical properties of the full-order operators are maintained at the ROM level. Depending on the operator considered, one may enforce symmetry, positive definiteness, or other algebraic constraints on the reduced operators, either as a *weak* constraint (an additional term in the loss function) or as a *hard* constraint (by employing a specific NN architecture). Preserving these properties improves the FOM-ROM consistency, and contributes to the robustness of the numerical method, as showed in the context of DMD [18].

Is the training procedure computationally expensive?

Training the neural networks constitutes an additional offline cost. However, convolutional architectures exploit parameter sharing: the same convolutional kernels are applied throughout the computational domain, so the number of trainable parameters depends

primarily on the kernel size rather than on the dimension of the discretized problem. This makes convolution-based maps significantly more scalable than fully connected architectures.

What happens when the full-order operators become extremely large?

For complex test cases, a large value of N may be required to achieve an accurate simulation, making the storage and compression of the resulting full-order matrices prohibitively expensive in terms of memory. However, PDEs' operators such as mass, diffusion, and convection matrices are inherently sparse and exhibit strong locality. These features can be naturally exploited through the **continuous convolution** operation introduced in Section 9.2.1, which are designed to operate on unstructured representations and can directly handle the sparse structure of differential operators. In particular, instead of considering the full matrix representation, only the non-zero entries of the discrete operators are processed, allowing the network to learn local patterns while avoiding the storage of large dense matrices. This substantially reduces the memory footprint and improves the scalability of the proposed approach for large-scale problems.

CONVOLUTIONAL OPERATOR COMPRESSION STRATEGIES FOR ROMs

This chapter investigates the use of convolutional architectures for the compression of differential operators, with the goal of building a novel nonlinear **ROM** framework.

As motivated in Section 9.3, the main goal of this chapter is to investigate how to construct a **nonlinear ROM** by learning compressed representations of full-order operators rather than directly approximating the solution field.

We propose a preliminary framework based on trainable neural network mappings that perform nonlinear operator compression, while maintaining the structure of a classical reduced problem formulation. This study represents an initial step towards the development of nonlinear operator-based **ROMs**, exploring the potential of neural networks to extend classical projection-based approaches beyond linear subspace approximations.

The chapter is subdivided in the following parts:

- ▶ Section 10.1, where the novel **ROM** structure is presented;
- ▶ Section 10.2, which performs a preliminary investigation on three two-dimensional test cases, namely a Poisson problem with two parameters, an advection-dominated test case, and a Burgers test case. A final discussion of the results and of the computational costs is provided in Section 10.2.4.
- ▶ A conclusive part highlighting the main findings and possible future research directions is finally presented in Section 10.3.

10.1 Methodology

Here we introduce the novel nonlinear **ROM** proposed in this work, which is sketched in Figure 10.1. As briefly anticipated in Section 9.3, our goal is to propose a nonlinear version of the well-known projection-based intrusive **ROM** approaches, based on neural networks.

We start by considering a full-order system in the form $\mathbf{A}\mathbf{u} = \mathbf{f}$, where $\mathbf{A} \in \mathbb{R}^{N \times N}$, $\mathbf{u} \in \mathbb{R}^N$, and $\mathbf{f} \in \mathbb{R}^N$. We aim to compress the **FOM** into a reduced order system of prescribed dimension $r \ll N$. We seek three different mappings:

- ▶ a nonlinear convolutional encoder **CE**: $\mathbf{A} \mapsto \mathbf{A}_r$, with $\mathbf{A}_r \in \mathbb{R}^{r \times r}$. To learn this nonlinear mapping, we employ **CNNs**. Originally developed for structured data such

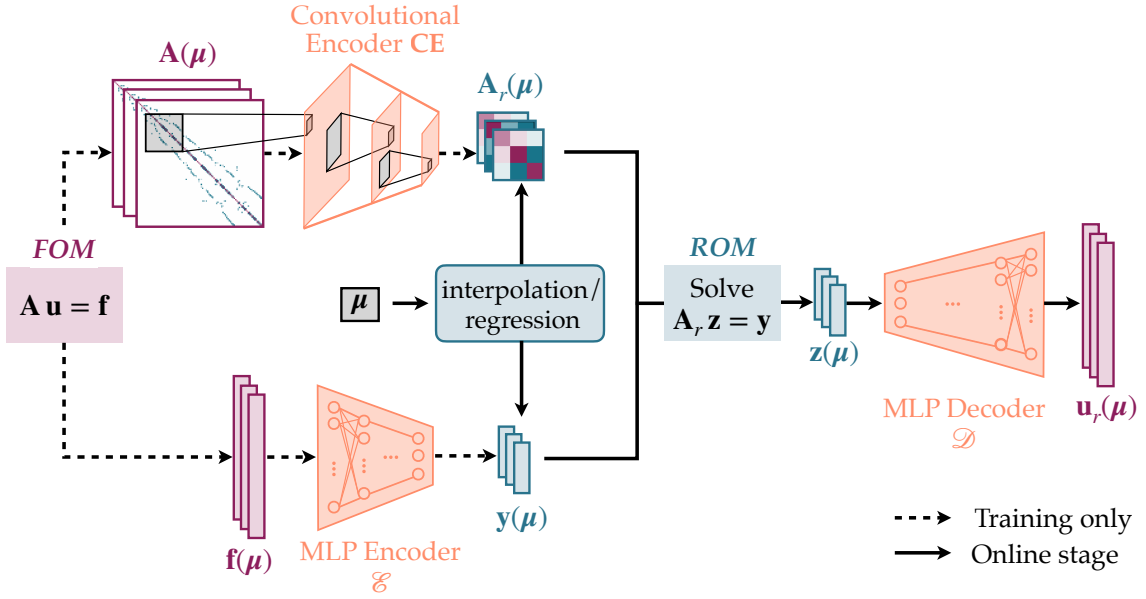


Figure 10.1: Schematic illustration of the proposed CE-AE approach, where a convolutional neural network is used to compress the full-order discrete operators.

as images, CNNs are naturally suited to discrete differential operators, which can also be interpreted as structured inputs;

- ▶ a nonlinear encoder $\mathcal{E} : f \mapsto y$, with $y \in \mathbb{R}^r$ is the latent representation of the full-order field f ;
- ▶ a nonlinear decoder $\mathcal{D} : z \mapsto u_r$, where z is found by solving the reduced r -dimensional system $A_r z = y$. In a classical ROM fashion, the goal is to obtain a decoded solution $u_r \in \mathbb{R}^N$ as close as possible to the full-order reference u .

Both the matrix A and the vector-valued fields u and f are in general parameter-dependent, namely $A(\mu)$, $f(\mu)$, and $u(\mu)$.

We consider as encoder $\mathcal{E}$ and decoder $\mathcal{D}$ classical MLP neural networks, but other choices are possible, such as convolutional neural networks [193] or continuous convolutional neural networks [73].

Moreover, we consider the MLP encoder and decoder as parts of the same AE, e.g., having a symmetric architecture in the layers and activation functions, but different choices are possible. More in detail, about the AE architecture, we remark that the encoder and decoder weights are not symmetric but jointly trained without any structural constraint.

Remark: about the solution encoder.

It is important to highlight that we compress the right-hand side term f with the same encoder used to compress the solution u , as the two vectors formally belong to the same vector space. However, this assumption may introduce an excessive

inductive bias in the training process. Indeed, although $\mathbf{u}$ and $\mathbf{f}$ belong to the same vector space, their spatial structures and relevant features can differ significantly. Enforcing a shared encoder therefore restricts the admissible latent space and may prevent the network from learning the most suitable reduced representation for the solution, negatively affecting the convergence.

In standard **AEs**, the loss function that is typically adopted is the discrepancy between the reconstructed solution, namely, adopting the notation of Figure 10.1:

$$\mathcal{L}_{\text{AE}} = \frac{1}{N_\mu} \sum_{i=1}^{N_\mu} \|\mathbf{u}(\mu_i) - \mathbf{u}_r(\mu_i)\|_2^2, \quad (10.1)$$

where N_μ is the number of training snapshots, and μ is the solution set of parameters.

The novel architecture has instead a loss function in the form

$$\mathcal{L}_{\text{CE-AE}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{AE}}, \quad (10.2)$$

where we distinguish the following contributions.

- The *intrusive* contribution $\mathcal{L}_{\text{CE}}$, namely the error made by solving the reduced system. In particular, it is computed as follows.
 1. Compute the reduced operator $\mathbf{A}_r = \text{CE}(\mathbf{A})$, and the reduced vector $\mathbf{y} = \mathcal{E}(\mathbf{f})$;
 2. Solve the reduced system $\mathbf{A}_r \mathbf{z} = \mathbf{y}$ and find $\mathbf{z}$.
 3. Backmap the reduced solution $\mathbf{z}$ to the full-order space using the decoder $\mathcal{D}$, and compute the *intrusive* loss as:

$$\mathcal{L}_{\text{CE}} = \frac{1}{N_\mu} \sum_{i=1}^{N_\mu} \|\mathbf{u} - \mathcal{D}(\mathbf{z})\|_2^2. \quad (10.3)$$

- The *nonintrusive* contribution $\mathcal{L}_{\text{AE}}$, representing the **AE** = $\mathcal{D} \circ \mathcal{E}$ reconstruction error solely. This contribution is the one of Equation (10.1), with $\mathbf{u}_r = (\mathcal{D} \circ \mathcal{E})(\mathbf{u})$.

We call the proposed approach **Convolutional Encoder-AutoEncoder (CE-AE)**.

Remark: About the compression of more complex FOM systems. The proposed logic may be extended to more complex systems, like (5.12), by applying the nonlinear compression to each full-order operator appearing in the governing equations. In the presence of two or more operators, we can modify our approach in two different ways: (i) consider different CEs, which separately compress the given operators; (ii) consider a unique CE with more input channels, each processing one operator. More complex nonlinear **FOMs** in the form $\mathcal{R}(\mathbf{u}; \mathbf{x}, \mu) = \mathbf{0}$ lead to a **ROM** of the form $\mathcal{R}_r(\mathbf{z}; \mu) = \mathbf{0}$. In this case, the exact resolution of the **ROM** system can be handled

in two ways:

- ▶ Iteratively solve the nonlinear system using a Newton method. In this case, the reduced solution $\mathbf{z}$ would be the *exact* one, and the loss contribution remains the same as in Equation (10.3).
- ▶ Replace the loss contribution of Equation (10.3) with the residual norm, leading to:

$$\mathcal{L}_{\text{CE}} = \frac{1}{N_\mu} \sum_{i=1}^{N_\mu} \|\mathcal{R}_r(\mathbf{z}; \mu_i)\|_2^2. \quad (10.4)$$

This transforms the model in a **Physics-Informed Neural Network (PINN)** [267] at the reduced-order level, as done in previous works like [169, 62, 75, 130].

In the case of parametrized operators $\mathbf{A}(\mu)$ and/or right-hand side vectors $\mathbf{f}(\mu)$, the previously described architecture cannot be directly employed for the evaluation of unseen parameter configurations $\mu_\star$. Indeed, the proposed method requires access to the **FOM** quantities, namely the operators and/or the full-order vectors, which are not available at new parameter values.

Therefore, an additional parametrization step is required to enable prediction at unseen configurations. As illustrated in Figure 10.1, we introduce a parametric mapping $\mu \mapsto \mathbf{A}_r$ and/or $\mu \mapsto \mathbf{y}$, which directly predicts the corresponding latent representation of the operators and vectors from the parameter space.

In this way, the reduced-order quantities can be efficiently reconstructed without requiring the assembly or evaluation of the underlying **FOM** system at the new configuration $\mu_\star$. In this case we adopt a **RBF** interpolation approach, as described in Equation (8.2) of Section 8.1, but different approaches (like **GPR** or **NN**) are possible. We will indicate the proposed **ROM**, coupled with the **RBF** prediction at unseen states as **CE-AE-RBF**.

Additionally, in convection-dominated cases the full-order convection matrix is typically $\mathbf{C}(\mathbf{u})$, hence a mapping $\mathbf{z} \mapsto \mathbf{C}_r$ is here proposed, as we will see in the last test case considered (Section 10.2.2). In time-dependent test cases, this also allows the generalization of the approach to different time step sizes, as the time instance t is not an explicit input of the reduced map.

Remark: About the stability of the reduced system $\mathbf{A}_r \mathbf{z} = \mathbf{y}$. We remark that in all the proposed convolutional approaches and in all the test cases, we consider as output matrix the sum of the encoder output matrix, and of the **stabilization term** $\epsilon \mathbf{I}_r$, where $\mathbf{I}_r$ is the identity matrix of dimension $r \times r$ and $\epsilon = 1 \times 10^{-4}$. The addition of this regularization term enhances the numerical stability and improves the conditioning of the reduced system $\mathbf{A}_r \mathbf{z} = \mathbf{y}$. This system is solved both in the offline stage, during the training process, and in the online stage, when the solution

is predicted for unseen configurations.

Another possible strategy to prevent poor conditioning is to augment the loss function with a regularization term based on the condition number of the reduced operator. Nevertheless, the repeated computation of the condition number during training would significantly increase the computational cost of each optimization step. For this reason, we do not adopt this approach in the present work.

10.1.1 The efficient use of continuous convolution for sparse matrices

In real-world scenarios, the **FOM** system usually involves a very large number of degrees of freedom N , potentially reaching the order of millions. As a result, the corresponding **FOM** matrices may have prohibitively large dimensions ($N \times N$), preventing their direct manipulation and storage. This limitation can make the adoption of standard convolutional architectures impractical, as they often require memory resources that scale unfavourably with the input size.

However, it is worth noting that **FOM** operators are typically sparse, meaning that the majority of their entries are equal to zero.

The question now is: *How can we exploit this property to overcome the memory limitations associated with the direct treatment of full matrices?*

In this work we consider the continuous convolutional architecture (introduced in Section 9.2.1), which is specifically designed to operate on irregularly sampled data and can naturally exploit the sparsity structure of the operators.

In the original paper on continuous convolutional layers [73], the input consists of the spatial coordinates (e.g., (x, y) in a two-dimensional setting) together with the corresponding values of the physical field evaluated at those locations $\mathbf{u}(x, y)$. Following the same idea, we interpret the discrete operator as the realization of an underlying continuous field. Instead of providing the full $N \times N$ matrix as input, we consider only the entries corresponding to non-zero elements, providing their indices (i, j) and the associated values. Consequently, the input dimensionality is reduced from $N \times N$ to $3 \times N_{\text{nz}}$, where $N_{\text{nz}} \ll N$ denotes the number of non-zero entries and is typically much smaller than N , depending on the structure of the considered operator. As represented in Figure 10.2, one (or more) continuous convolutional layers are used to compress the original **FOM** operator to an intermediate dimension, which can be then further compressed by standard convolutional layers.

The nonlinear **ROM** approach using the continuous convolutional encoder architecture is here named **Continuous Convolutional Encoder-AutoEncoder (CCE-AE)**. Analogously to the standard convolutional **ROM (CE-AE)**, we refer to the efficient **ROM** version using the **RBF** interpolation as **CCE-AE-RBF**. This formulation offers two important advantages. First, it allows the proposed approach to be applied to different types of operators, such as mass, diffusion, or stiffness matrices, without requiring a modification of

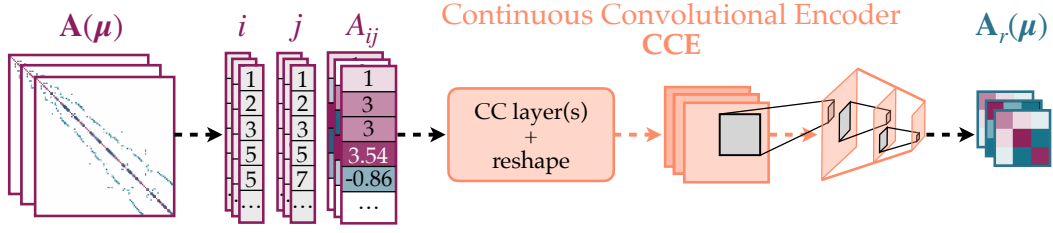


Figure 10.2: Sketch on the operator compression strategy using the continuous convolutional architecture, coupled with a standard convolutional network.

the architecture. Second, since the input is defined through the coordinates of the non-zero entries rather than through a fixed matrix representation, the method can naturally handle operators arising from different discretizations and computational grids.

Furthermore, if a **CCAE** is adopted also for the solution field, the entire framework could be extended to become **independent of the underlying discretization grid**. In this case, both the operator and the solution would be represented through continuous latent mappings, enabling the application of the reduced model across different grids and full-order discretizations. This extension is not investigated in the present work and is left as a direction for future research.

10.2 Numerical results

Here we evaluate the performance of the proposed approach in three different test cases:

- ▶ a Poisson equation on a square with two parameters incorporated into the source term (Section 10.2.1);
- ▶ a parametrized advection-diffusion model defined on a square domain, with one parameter defining the diffusivity (Section 10.2.2);
- ▶ a Burgers problem in a 2D backstep domain, where the time is the unique parameter considered (Section 10.2.3).

For each test case, we assess the performance of the proposed **CE-AE** approach (and, for parametrized problems, of the **CE-AE-RBF** framework) by comparing it with standard linear reduction techniques, such as **POD** and **POD-RBF**, as well as nonlinear approaches based on **AE** and **AE-RBF**. We specify that we exploit the same **AE** architecture both in the **CE-AE** and in the **AE** solely, to guarantee a fair comparison.

The accuracy of the different methods is evaluated through the relative L^2 error of the reconstructed solution with respect to the corresponding **FOM** solution, together with a

qualitative assessment of the obtained approximations. Furthermore, we provide a comparison of the loss behaviour of standard **AEs** and the proposed **CE-AE** approaches, investigating an ensemble of different **NNs** with the same architecture and different weights' initialization.

Finally, we analyse the reduced operators obtained through the proposed nonlinear compression strategy and discuss their underlying properties.

10.2.1 Poisson test case

The first test case is a Poisson non-affine equation on the square domain $\Omega = [-1, 1]^2$, with two parameters $\mu = (\mu_1, \mu_2) \in [-1, 1]^2$, written in the following form:

$$\begin{cases} \Delta u = g(x; \mu) \text{ in } \Omega, \\ u = 0 \text{ on } \partial\Omega, \end{cases} \quad \text{with } g(x; \mu) = \exp(-2(x - \mu_1)^2 - 2(y - \mu_2)^2). \quad (10.5)$$

We can then obtain the following weak formulation:

$$a(u, v) = f(v; \mu), \text{ where } a(u, v) = \int_{\Omega} \nabla u \cdot \nabla v \, dx, \quad f(v; \mu) = \int_{\Omega} g(x, \mu) \, dx,$$

for each test function $v \in H_0^1(\Omega)$. Using a FE discretization, the above formulation can be written as a linear system with parametrized right-hand side:

$$\mathbf{A}u(\mu) = f(\mu), \quad (10.6)$$

with $\mathbf{A} \in \mathbb{R}^{N \times N}$, $f \in \mathbb{R}^N$, and $N = 638$. The FE grid has been built using the *gmsh* utility, and the FE simulation is performed using FEniCS [206, 206, 9].

We collect a set of 30 train and 15 test snapshots, and then perform the **POD** considering the training data. Figure 10.3 represents the values of the train, test, and validation parameters, together with the **POD** singular values decay. The validation set, composed of 15 snapshots, is used during the training of the **NNs** solely to monitor the different components of the loss function and assess the model's generalization performance.

Although the singular values exhibit an exponential decay, the decay rate is relatively slow, indicating that a large number of **POD** modes is required to capture a significant fraction of the solution energy.

Following the same rationale adopted in the previous chapters of this thesis, the objective is to achieve accurate **ROM** predictions in *under-resolved* or *marginally-resolved* regimes, namely by considering a reduced number of degrees of freedom. For this reason, we analyse the results obtained considering $r = 2, 3$, and 4. The diffusion matrix $\mathbf{A}$ in Equation (10.6), whose sparsity pattern is represented in Figure 10.4, is symmetric and positive-definite, namely $\mathbf{A}^T = \mathbf{A}$ and $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0$, $\forall \mathbf{x} \neq \mathbf{0}$.

In the classical reduced basis framework, the corresponding reduced matrix $\mathbf{A}_r = \mathbf{V}^T \mathbf{A} \mathbf{V}$ preserves the symmetry and positive-definiteness of the **FOM** matrix $\mathbf{A}$, because the discrete modes matrix $\mathbf{V}$ is an orthogonal matrix. We then pose ourselves the following question:

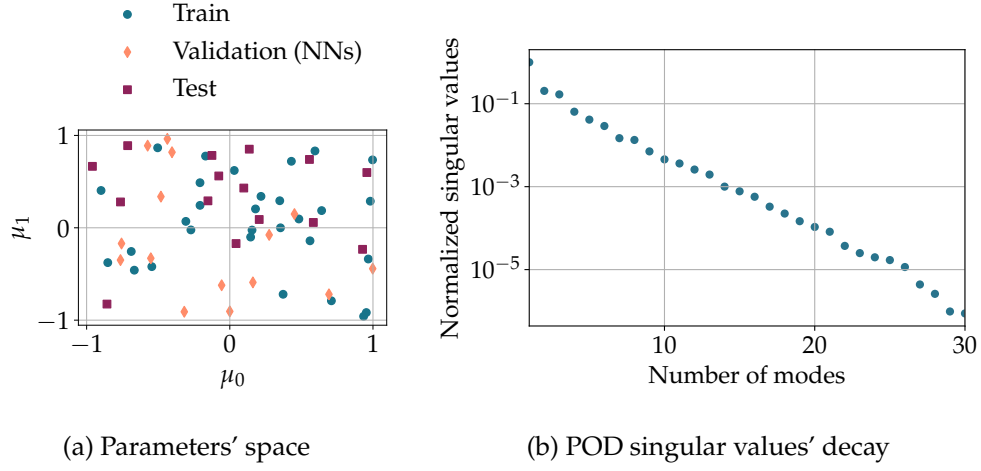


Figure 10.3: *Poisson test case*. Train and test parameters (left), and the POD singular values decay (right).

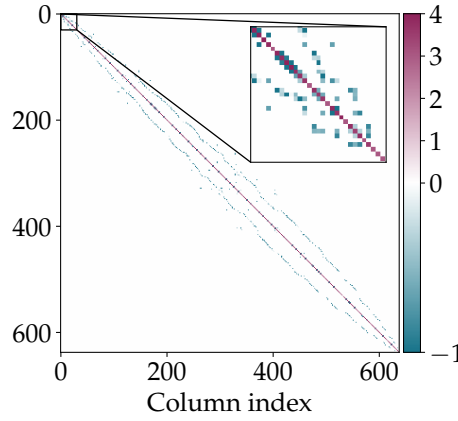


Figure 10.4: *Poisson test case*. Full-order matrix $\mathbf{A}$.

Is the optimal reduced matrix $\mathbf{A}_r$ symmetric positive-definite?

To numerically answer this question, we investigate two compression variants, where the final reduced matrix is expressed in two different ways:

- $\mathbf{A}_r = \mathbf{A}'_r + \epsilon \mathbf{I}_r$, where $\mathbf{A}'_r$ is the encoder output, and $\epsilon = 1 \times 10^{-4}$. The stabilization term $\epsilon \mathbf{I}_r$ improves numerical stability and conditioning when solving the reduced system, as already mentioned in Section 10.1. The resulting model is simply referred to as **Convolutional Encoder (CE)**;
- $\mathbf{A}_r = (\mathbf{A}'_r)^T \mathbf{A}'_r + \epsilon \mathbf{I}_r$, which is by construction symmetric and positive-definite. This model is called **Symmetric positive definite Convolutional Encoder (S-CE)**.

Before discussing the results, we first analyse the training behaviour of the different models by comparing the validation loss evolution of the **AE**, **CE-AE**, and **Symmetric positive definite Convolutional Encoder-AutoEncoder (S-CE-AE)** approaches. In particular,

Figure 10.5 reports the loss trends over the training epochs for 10 neural networks sharing the same architecture but initialized with different random weights. The curves are presented in terms of their average value and associated confidence interval, in order to assess both the convergence behaviour and the variability induced by the initialization.

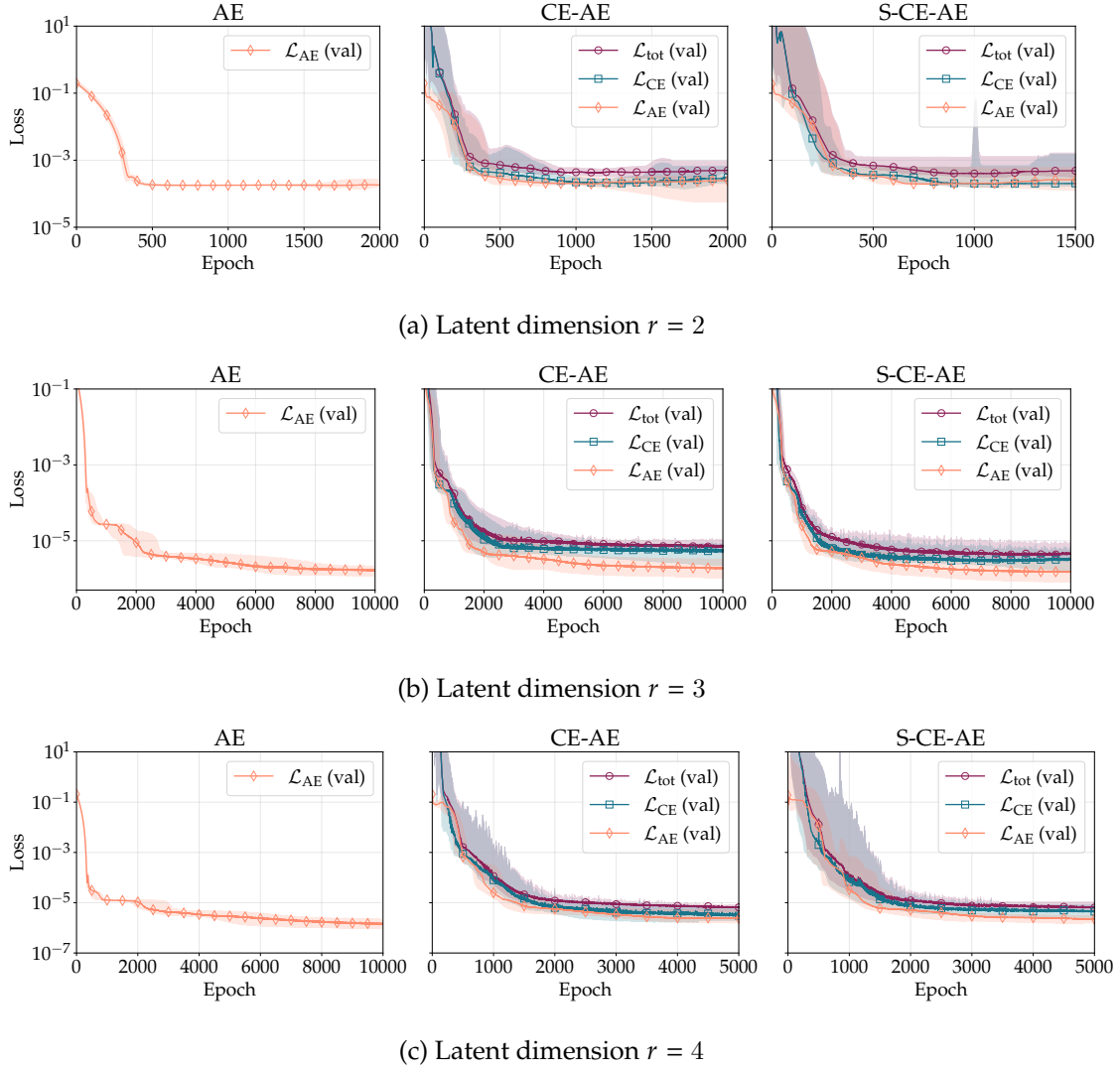


Figure 10.5: *Poisson test case*. Loss trend across epochs for a standard AE, and for the novel models CE-AE and S-CE-AE, at different latent dimensions, namely $r = 2, 3$, and 4.

For all the considered latent dimensions $r = 2, 3, 4$, we observe that the standard AE training exhibits, in general, a lower variability compared to the proposed CE-AE and S-CE-AE approaches. This behaviour is expected, since the latter models introduce an additional loss contribution, thus imposing a stronger constraint on the optimization process and potentially increasing the sensitivity to the initial conditions.

Moreover, the average value of the loss component associated with the AE reconstruction, $\mathcal{L}_{\text{AE}}$, at convergence is comparable among the three approaches for all the considered latent dimensions. However, for $r = 2$ and $r = 3$, the minimum values achieved during

training, represented by the lower bound of the confidence intervals, are lower for both the **CE-AE** and **S-CE-AE** models. This suggests that the inclusion of the additional loss term $\mathcal{L}_{\text{CE}}$ can improve the convergence of the autoencoder reconstruction loss, allowing the network to reach lower reconstruction errors compared to the standard **AE** training alone.

The reduced matrices obtained with the proposed approaches are graphically represented in Figure 10.6. The matrices are normalized between 0 and 1 to enable a qualitative comparison of their structures.

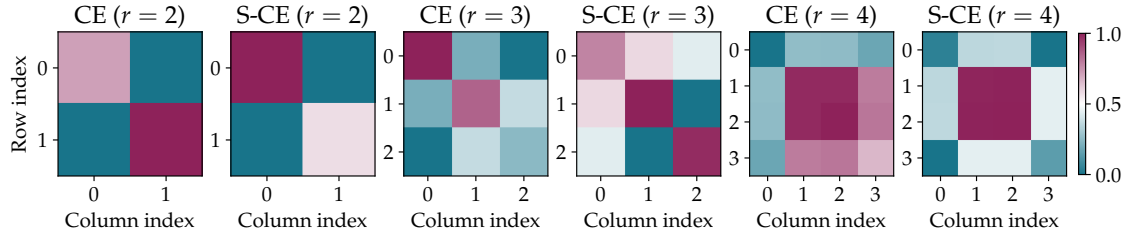


Figure 10.6: *Poisson test case*. Reduced matrices $\mathbf{A}_r$, normalized in range $[0, 1]$. The matrices are obtained by compressing the full-order discrete matrix $\mathbf{A}$ with the convolutional encoder **CE** and with its symmetric version **S-CE**, at different latent dimensions $r = 2, 3$, and 4.

We first consider the case with latent dimension $r = 2$. The reduced matrices obtained by the **CE** and **S-CE** models at convergence exhibit a similar structure, up to a permutation of the reduced degrees of freedom, which in this case corresponds to a transposition of the matrix. This behaviour can be attributed to the lack of an intrinsic ordering of the latent variables learned by the **AE**: different training runs may identify equivalent reduced representations associated with different latent coordinate arrangements.

Interestingly, both reduced operators are symmetric, despite the fact that no symmetry constraint is explicitly enforced in the **CE** architecture. The same behaviour is observed also for larger latent dimensions ($r = 3$ and $r = 4$), suggesting that the optimization process naturally drives the network towards symmetric reduced operators. This indicates that the optimal reduced representation identified by the neural network tends to preserve the structural properties of the original **FOM** operator, similarly to standard projection-based linear reduction techniques.

A similar correspondence between the global structures of the reduced operators can also be observed for the case $r = 4$, further confirming the consistency of the learned representations across different latent dimensions.

The relative L^2 test errors obtained by the proposed approaches, together with the corresponding **POD**-based results, are reported in Figure 10.7. The figure shows the box plots of the relative L^2 errors obtained using the best-performing network among the different training realizations considered in Figure 10.5. The selected network is then evaluated over a set of unseen parameter configurations (represented in Figure 10.3).

For the **POD** framework, we consider three different error contributions: the reconstruction error (obtained by directly compressing and reconstructing the solution field), the **POD-RBF** error, and the **POD-Empirical Interpolation Method (EIM)** error.

The **POD-EIM** approach is a projection-based intrusive method in which the reduced quantities (the right-hand side vector y in the present case) at unseen parameter configurations are approximated through an affine expansion of pre-computed **EIM** interpolation modes. In this work, we consider $r_{\text{EIM}} = 2r$ interpolation modes, although different choices are possible depending on the desired accuracy and computational cost.

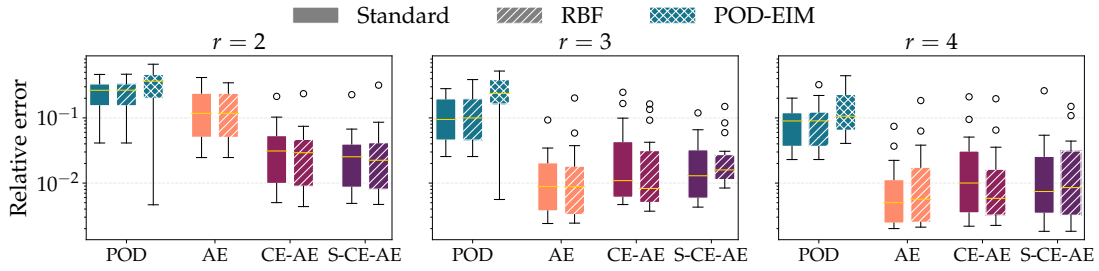


Figure 10.7: *Poisson test case*. Relative L^2 test errors of different approximations with respect to the FOM reference, at the latent dimensions $r = 2, 3$, and 4 .

From the plots, we can draw the following conclusions.

- For $r = 2$, the proposed approaches achieve an accuracy improvement of approximately one order of magnitude compared to the standard **AE**. This result is further supported by the qualitative comparison reported in Figure 10.8, which shows that both the **POD** and the standard **AE** fail to accurately generalize the solution prediction to unseen configurations.
- At $r = 3$ and $r = 4$, the proposed approaches achieve results comparable to the **AE-RBF**.

As already discussed when analysing the loss behaviour, the inclusion of the convolutional encoder and the additional loss contribution does not negatively affect the performance of the autoencoder. Instead, the proposed formulation provides an additional regularizing effect, which stabilizes the optimization process and allows the network to reach comparable or even improved accuracy with respect to the standard **AE**.

10.2.2 Advection-diffusion test case

Here, we are interested in a parametrized advection-diffusion problem, defined on the unit square $\Omega = [0, 1]^2$, written as follows:

$$\begin{cases} -10^{-\mu} \Delta u + \beta \cdot \nabla u = 1 & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases} \quad (10.7)$$

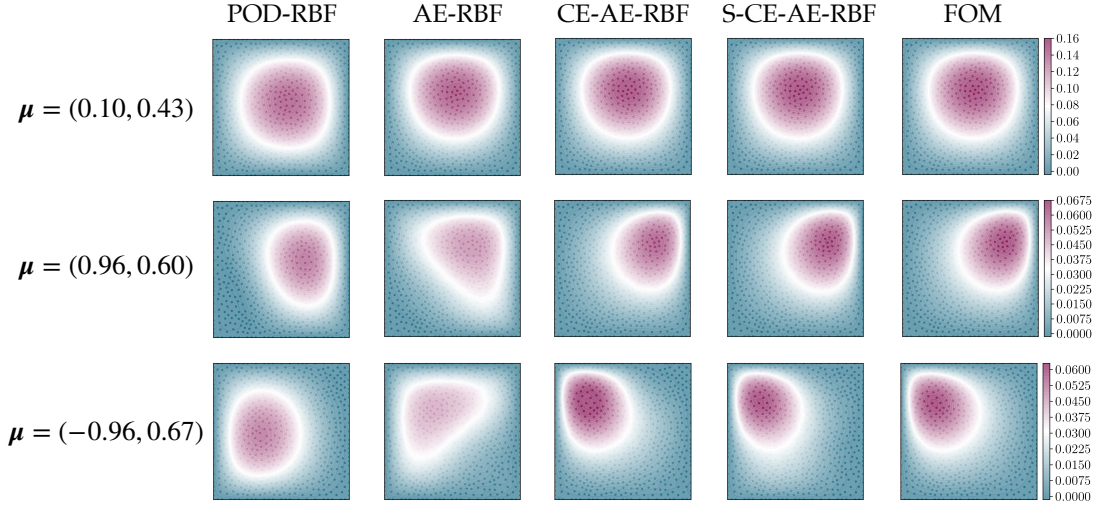


Figure 10.8: *Poisson test case*. Approximated solutions for POD-RBF, AE-RBF, CE-AE-RBF, and S-CE-AE-RBF, compared with the FOM reference, at latent dimension $r = 2$, and considering three test parameters.

where $\beta = (1, 1)$ is the advection velocity field, while the parameter μ determines the Péclet number of the problem. Increasing μ reduces the diffusion coefficient, leading to advection-dominated regimes, whereas smaller values of μ correspond to diffusion-dominated ones. The corresponding weak form can be written as:

$$a(u, v; \mu) = f(v), \quad \forall v \in H_0^1(\Omega),$$

with

$$a(u, v; \mu) = \int_{\Omega} 10^{-\mu} \nabla u \cdot \nabla v \, dx + \int_{\Omega} (\beta \cdot \nabla u) v \, dx, \quad f(v) = \int_{\Omega} v \, dx.$$

Since the above formulation might become unstable at some μ values, we apply the SUPG stabilization, obtaining the following formulation:

$$\begin{aligned} a(u, v; \mu) + a_{\text{SUPG}}(u, v; \mu) &= f(v) + f_{\text{SUPG}}(v), \\ \text{with } a_{\text{SUPG}} &= \tau_{\text{SUPG}} \int_{\Omega} [-10^{-\mu} (\Delta u + \beta \cdot \nabla u) \cdot (h_x \beta \cdot \nabla v)] \, dx, \\ \text{and } f_{\text{SUPG}} &= \tau_{\text{SUPG}} \int_{\Omega} (h_x \beta \cdot \nabla v) \, dx, \end{aligned} \quad (10.8)$$

where τ_{SUPG} is a parameter identifying the weight of the additional term ($\tau_{\text{SUPG}} = 0.5$ in our case), while h_x is a function defining the diameter associated with each element.

After discretizing the SUPG formulation using a FEM approach, we obtain the following linear system:

$$(\mathbf{A}_1 + 10^{-\mu} \mathbf{A}_2) \mathbf{u} = \mathbf{f}. \quad (10.9)$$

The full-order quantities are defined as follows:

- ▶ $(\mathbf{A}_1)_{ij} = \int_{\Omega} [(\beta \cdot \nabla \phi_j) \phi_i + \tau_{\text{SUPG}}(\beta \cdot \nabla \phi_j) \cdot (h_x \beta \cdot \nabla \phi_i)] dx;$
- ▶ $(\mathbf{A}_2)_{ij} = \int_{\Omega} [\nabla \phi_j \cdot \nabla \phi_i + \tau_{\text{SUPG}}(-\Delta \phi_j) \cdot (h_x \beta \cdot \nabla \phi_i)] dx;$
- ▶ $(f)_i = \int_{\Omega} [\phi_i + \tau_{\text{SUPG}}(h_x \beta \cdot \nabla \phi_i)] dx.$

The notation ϕ_i refers to the **FEM** basis functions, while the full dimensionality is in this case $N = 2500$.

The **FOM** formulation of Equation (10.9) allows for a fully separation of the non-parametrized and parametrized matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, which are represented in their sparsity pattern in Figure 10.9. As in the previous test case, we can recognize patterns in the matrices, e.g.,

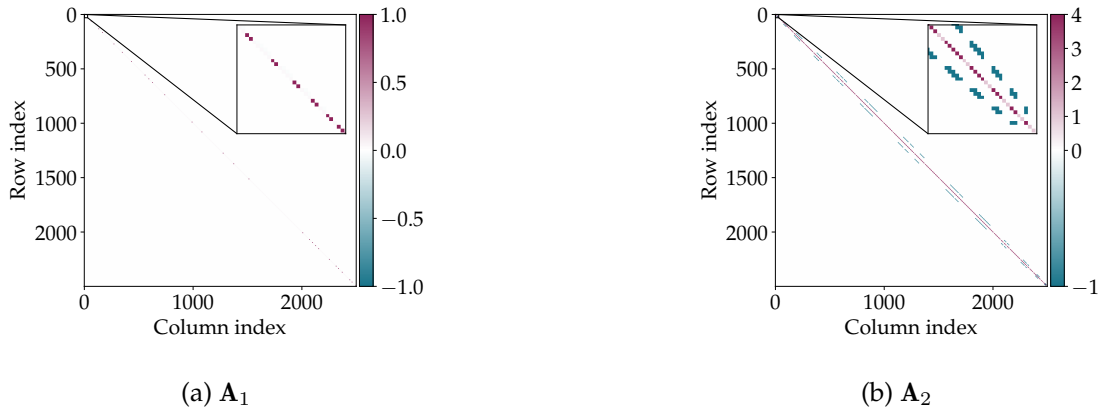


Figure 10.9: *Advection-diffusion test case*. Full-order matrices $\mathbf{A}_1$ and $\mathbf{A}_2$.

$\mathbf{A}_2$ is symmetric. However, since in the Poisson case the *structure-preserving* approach (**S-CE-AE**) and the standard **CE-AE** converged to similar results, we proceed now without imposing any constraint.

The resulting **ROM** is written as:

$$(\mathbf{A}_{r1} + 10^{-\mu} \mathbf{A}_{r2}) \mathbf{z} = \mathbf{y}, \quad (10.10)$$

with $r = 2, 3$, or 4 as latent dimension.

Hence, in this test case we do not need the addition of an interpolation or regression strategy that maps the parameters' space into the reduced quantities.

The parameter values considered for training, validation (only in models involving **NNs**) and testing, together with the **POD** singular values' decay, are represented in Figure 10.10. We consider 20 training parameters, and 5 parameters both for validation and test. The singular values' decay is similar to the previous test case, highlighting that we need a large number of modes to capture the solution accurately.

As done in the first test case, we analyse in Figure 10.11 the loss behaviour during training iterations, at $r = 2, 3, 4$, for the standard **AE** and for the **CE-AE** model. In all the latent dimensions considered, the **CE-AE** loss reaches values smaller than more than one order of magnitude with respect to the standard **AE**. This phenomenon suggests that the convergence is improved by the novel **ROM** architecture proposed. Moreover, the

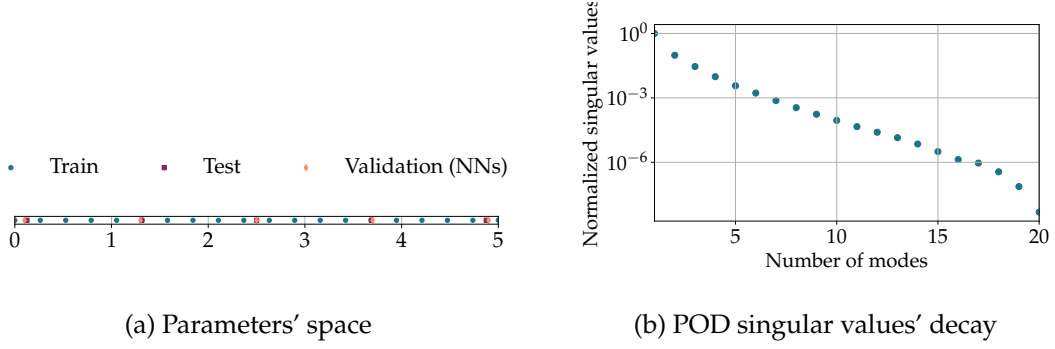


Figure 10.10: *Advection-diffusion test case*. Train, validation (for NNs), and test parameters (left), and the POD singular values decay (right).

addition of the loss contribution $\mathcal{L}_{\text{CE}}$ acts as a *stabilizer* for the **AE** itself, which converges to a more accurate latent representation.

The reduced matrices obtained by **CE** compression, and then normalized in the range $[0, 1]$ are represented in Figure 10.12.

At both $r = 2$ and $r = 3$, the **CE** converges to symmetric matrices $\mathbf{A}_{r2}$, reproducing the **FOM** properties. This is not equally satisfied at $r = 4$, where the increased latent dimension may lead to more variability in the converged solution. Moreover, at $r = 2$ all the components of $\mathbf{A}_{r2}$ have the same value, suggesting that it is treated as one additional degree of freedom, properly affecting the effect of the parameters on the latent solution. The **POD**, **AE** and **CE-AE** solutions are quantitatively compared in terms of the relative test L^2 error in Figure 10.13. The corresponding fields are graphically compared in Figure 10.14 for the **POD-RBF**, **AE-RBF**, and **CE-AE** methods. The results in both figures have been obtained considering the most accurate **AE** and **CE-AE** from the ensemble considered to generate Figure 10.11. At all latent dimensions, the **CE-AE** is significantly more accurate than the standard **POD** and **AE**, and it is characterized by a smaller variability of the error among the test parameters considered. Interestingly, the **AE** has similar accuracy to the **POD**, while it is even less accurate at $r = 4$.

The accuracy improvement can be also noticed qualitatively in Figure 10.14, which shows the approximated solutions at $r = 2$. Here, we can see that the **AE** and the **POD** fail to accurately approximate the solution at small μ values (first row of the figure), while the **CE-AE** is able to reproduce the **FOM** field similar to larger μ values.

10.2.3 Burgers test case

The final test case is a viscous Burgers equation in a 2D backstep domain modeled in OpenFOAM [153, 240] using the **FVM**. The domain and FV grid are represented in Figure 10.15, including the notation adopted for the boundaries.

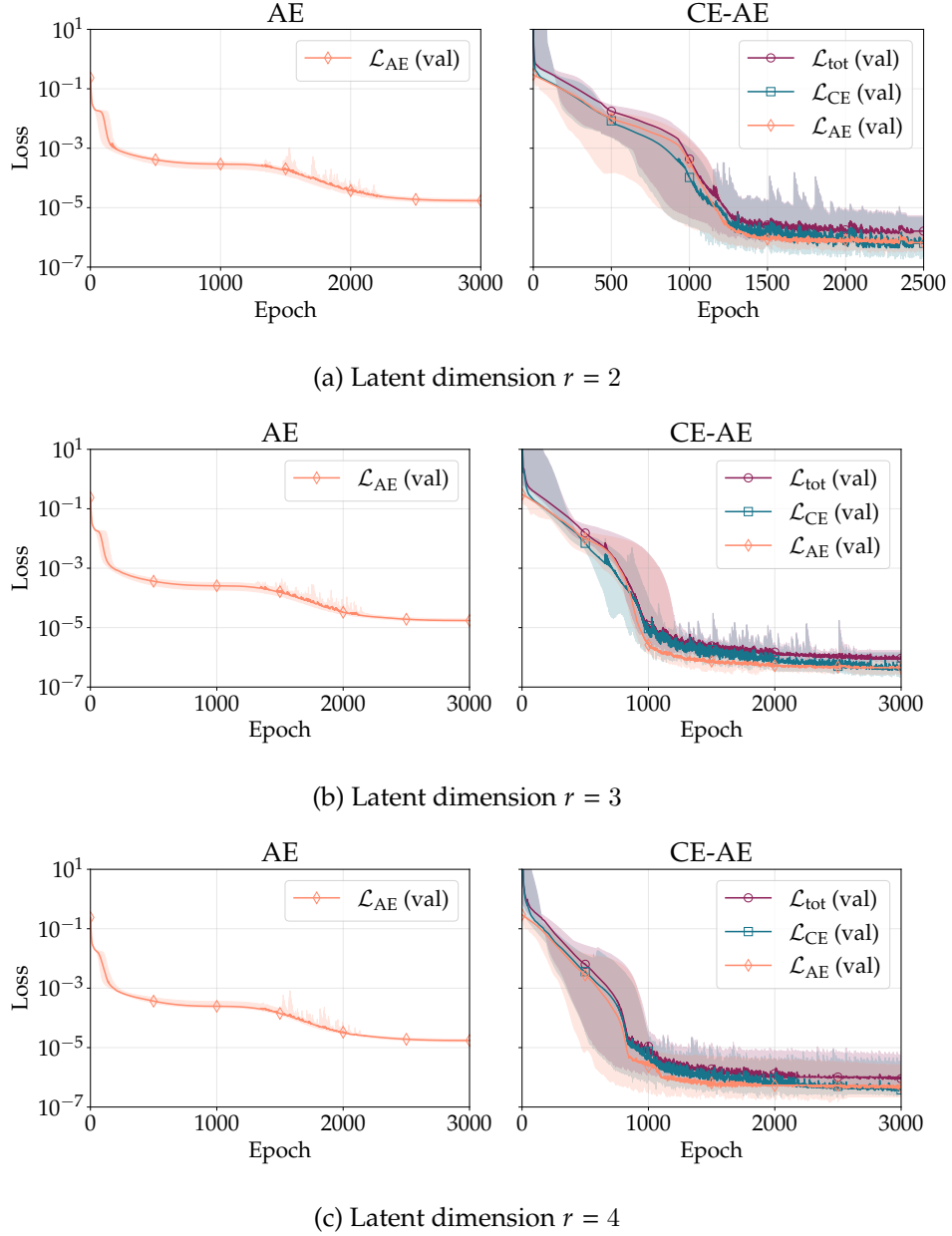


Figure 10.11: *Advection-diffusion test case*. Loss trend across epochs for a standard AE, and for the novel model CE-AE, at different latent dimensions, namely $r = 2, 3$, and 4.

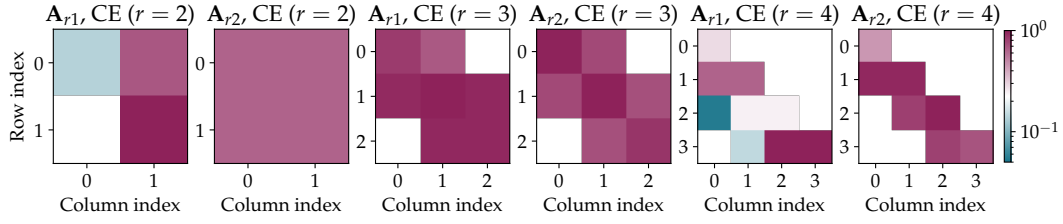


Figure 10.12: *Advection-diffusion test case*. Reduced matrices $\mathbf{A}_{r1}$ and $\mathbf{A}_{r2}$, normalized in range $[0, 1]$. The matrices are obtained by compressing the full-order discrete matrix $\mathbf{A}_1$ and $\mathbf{A}_2$ with the convolutional encoder CE, at different latent dimensions $r = 2, 3$, and 4.

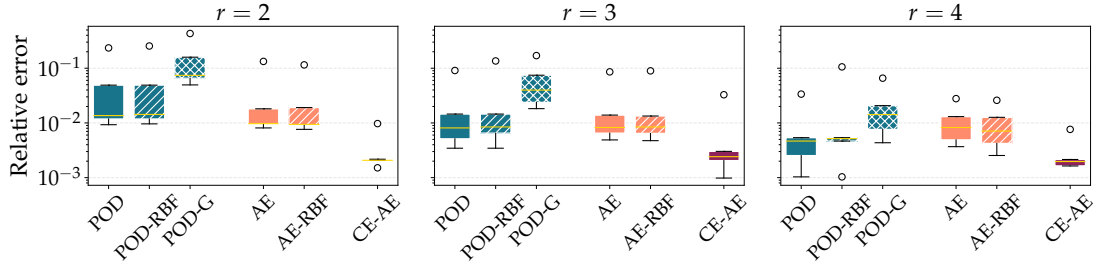


Figure 10.13: *Advection-diffusion test case*. Relative L^2 test errors of different approximations with respect to the FOM reference, at the latent dimensions $r = 2, 3$, and 4.

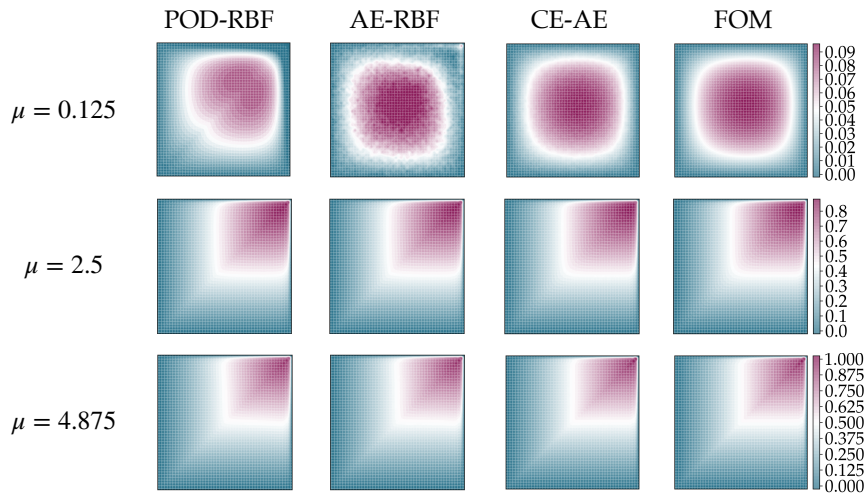


Figure 10.14: *Advection-dominated test case*. Approximated solutions for POD-RBF, AE-RBF, and CE-AE, compared with the FOM reference, at latent dimension $r = 2$, and considering three test parameters.

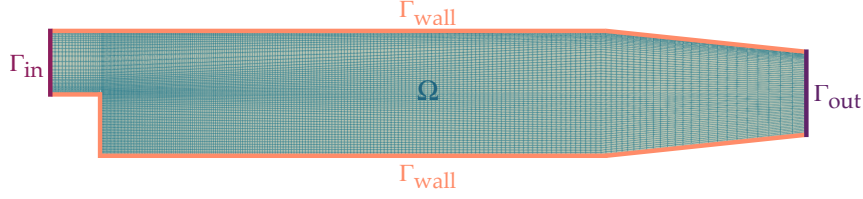


Figure 10.15: *Burgers test case*. Domain and computational grid adopted, with boundaries' notation.

The problem in continuous form can be written as follows:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} - \nu \Delta \mathbf{u} = \mathbf{0} & \text{in } \Omega, \\ \mathbf{u} = (1, 0) & \text{on } \Gamma_{\text{in}}, \\ \nabla \mathbf{u} \cdot \mathbf{n} = 0 & \text{on } \Gamma_{\text{out}}, \\ \mathbf{u} = \mathbf{0} & \text{on } \Gamma_{\text{wall}}. \end{cases} \quad (10.11)$$

The continuous problem may be discretized as follows:

$$\mathbf{M}_{\mathbf{u}} \frac{\mathbf{u}_{n+1} - \mathbf{u}_n}{\Delta t} + \mathbf{C}_{\mathbf{u}}(\mathbf{u}_n) \mathbf{u}_{n+1} - \nu \mathbf{D}_{\mathbf{u}} \mathbf{u}_{n+1} = \mathbf{f}_{\mathbf{C}}(\mathbf{u}_n) + \mathbf{f}_{\mathbf{D}}, \quad (10.12)$$

where $\mathbf{M}_{\mathbf{u}}$ is the mass matrix, $\mathbf{C}_{\mathbf{u}}(\mathbf{u})$ is the discrete convection operator, $\mathbf{D}_{\mathbf{u}}$ is the laplacian operator¹, while $\mathbf{f}_{\mathbf{D}}$ and $\mathbf{f}_{\mathbf{C}}$ are the known vectors associated with the laplacian and convection terms, respectively. In practice, the finite-volume discretization of each operator gives rise not only to a sparse matrix but also to a source vector, which incorporates the boundary conditions. In a more compact form, we can rewrite Equation (10.12) as follows:

$$(\mathbf{M}_{\mathbf{u}} + \Delta t (\mathbf{C}_{\mathbf{u}}(\mathbf{u}_n) + \mathbf{D}_{\mathbf{u}})) \mathbf{u}_{n+1} = \mathbf{f}_{\mathbf{C}} + \mathbf{f}_{\mathbf{D}} + \mathbf{M}_{\mathbf{u}} \mathbf{u}_n. \quad (10.13)$$

Differently to the previous cases, the FOM number of degrees of freedom is now larger ($N = 24450$, with 12250 cells), making the memory demand to compress the operators $\mathbf{M}_{\mathbf{u}}$, $\mathbf{C}_{\mathbf{u}}$ and $\mathbf{D}_{\mathbf{u}} \in \mathbb{R}^{N \times N}$ prohibitively high. For this reason, we compress the FOM matrices using a continuous convolutional encoder to exploit the matrices sparsity, as introduced in Section 10.1.1.

The resulting ROM system can be written as follows:

$$(\mathbf{M}_{\mathbf{u}_r} + \Delta t (\mathbf{C}_{\mathbf{u}_r}(\mathbf{z}_n) + \mathbf{D}_{\mathbf{u}_r})) \mathbf{z}_{n+1} = \mathbf{y}_{\mathbf{C}}(\mathbf{z}_n) + \mathbf{y}_{\mathbf{D}} + \mathbf{M}_{\mathbf{u}_r} \mathbf{z}_n. \quad (10.14)$$

To clarify how the reduced system is constructed, we describe the pseudo-code of the offline stage in Algorithm 6 and the online prediction in Algorithm 7. In the **offline** stage, the logic adopted is the same as in the previous test cases, with some differences:

- There are multiple matrices to compress, and we adopt the same convolutional encoder. Other choices are possible (different channels, multiple encoders) but we postpone this investigation to future work.

¹We adopt the same notation as in the previous chapters.

- ▶ We only adopt an intrusive loss $\mathcal{L}_{\text{CCE}}$ to train the **CCE-AE** model. In our preliminary investigations, we noticed that including both the losses contributions led to slow or no convergence during training. Therefore, we decided to proceed without including the purely data-driven contribution into the loss.
- ▶ We deal with a time-dependent problem with fixed $\Delta t = 1 \times 10^{-4}$, which is kept unchanged in the **ROM** formulation (Equation (10.14)).
- ▶ The **RBF** interpolation (second part of Algorithm 6) is applied to the time-dependent matrix $\mathbf{C}_{\text{ur}}$ and vector $\mathbf{y}_{\text{Cr}}$. Since in the **FOM** such quantities are functions of the **FOM** state $\mathbf{u}_n$, we reproduce the same maps at the reduced level, namely $\mathbf{C}_{\text{ur}}(\mathbf{z}_n)$ and $\mathbf{y}_{\text{Cr}}(\mathbf{z}_n)$. Therefore, we distinguish two **RBF** interpolators, one for the matrices (**RBF_{mat}**), which interpolates the matrices element-wise, and one for the vector (**RBF_{vec}**), which take as input the latent state rather than the parameter (e.g., time). This allows the construction of more interpretable latent maps, as we will see in the results.
- ▶ The input used to train the **RBF** maps is composed of the latent trajectory $\mathbf{z}_n^{\text{train}}$, $n = 1, N_\mu$, where the $\mathbf{z}^{\text{train}}$ have been found by compressing the **FOM** quantities with the pre-trained CCE and $\mathcal{E}$, and solving the reduced system.

In the **online** stage, we compress the time-invariant matrices $\mathbf{M}_{\text{ur}}$ and $\mathbf{D}_{\text{ur}}$ using the CCE, while the time-dependent $\mathbf{C}_{\text{ur}}(t_n)$ are predicted at each time step t_n using the pre-trained **RBF_{mat}** map having as input the reduced solution $\mathbf{z}_n$. The same applies to the reduced vectors: $\mathbf{y}_{\text{D}}$ is found once by compressing the time-invariant vector $\mathbf{f}_{\text{D}}$, while the time-dependent vector $\mathbf{y}_{\text{C}}(t_n)$ is found as **RBF_{vec}**($\mathbf{z}_n$) at each time step.

As previously mentioned, the only parameter is time, considering the first $N_\mu = 50$ training time steps, and testing on the window $[0, N_T]$, with $N_T = 100$. The time steps t_n , with $n \in [50, 75]$ are used during training for validation. If we apply the **POD** to the snapshots containing the first $N_\mu = 50$ snapshots, we obtain the singular values' decay represented in Figure 10.16.

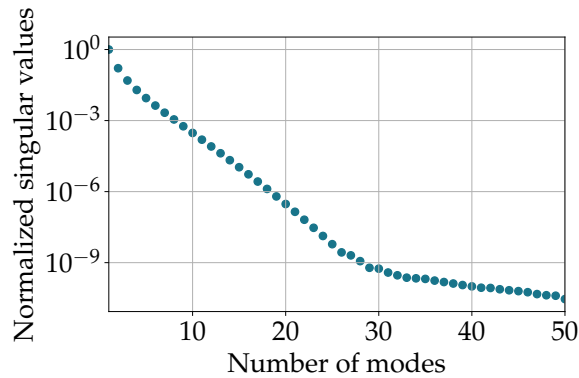


Figure 10.16: *Burgers test case*. The POD singular values decay.

Algorithm 6 Pseudo-code for the offline training of the CCE-AE-RBF approach.

Phase I: CCE-AE training

- 1: **Input data:** snapshots $\mathbf{u}_n$, $n = 1, \dots, N_\mu$; matrices $\mathbf{M}_u$, $\mathbf{D}_u$, and $\mathbf{C}_u(\mathbf{u}_n)$, $n = 1, \dots, N_\mu$; vectors $\mathbf{f}_D$, $\mathbf{f}_C(\mathbf{u}_n)$, $n = 1, \dots, N_\mu$;
- 2: Continuous convolutional encoder CCE; MLP encoder $\mathcal{E}$ and decoder $\mathcal{D}$; number of training epochs n_{epochs} ;
- 3: **for** $e \in [1, \dots, n_{\text{epochs}}]$ **do**
- 4: **Initial FOM state:**
- 5: $\mathbf{M}_{ur} = \text{CCE}(\mathbf{M}_u)$;
- 6: $\mathbf{D}_{ur} = \text{CCE}(\mathbf{D}_u)$;
- 7: $\mathbf{y}_D = \mathcal{E}(\mathbf{f}_D)$;
- 8: $\mathbf{z}_0 = \mathcal{E}(\mathbf{u}_0)$;
- 9: **for** $n \in [0, \dots, N_\mu - 1]$ **do**
- 10: $\mathbf{C}_{ur} \leftarrow \text{CCE}(\mathbf{C}_u(\mathbf{u}_n))$;
- 11: $\mathbf{y}_C \leftarrow \mathcal{E}(\mathbf{f}_C(\mathbf{u}_n))$;
- 12: $\mathbf{A}_r \leftarrow \mathbf{M}_{ur} + \Delta t(\mathbf{C}_{ur} + \mathbf{D}_{ur})$;
- 13: $\mathbf{y} \leftarrow \mathbf{y}_C + \mathbf{y}_D + \mathbf{M}_{ur}\mathbf{z}_n$;
- 14: Compute $\mathbf{z}_{n+1}$ as the solution of $\mathbf{A}_r\mathbf{z}_{n+1} = \mathbf{y}$;
- 15: **end for**
- 16: Compute the loss $\mathcal{L}_{\text{CCE}} = \frac{1}{N_\mu} \sum_{n=1}^{N_\mu} \|\mathcal{D}(\mathbf{z}_n) - \mathbf{u}_n\|_2^2$.
- 17: Backpropagate $\mathcal{L}_{\text{CCE}}$ and update the trainable parameters of CCE, $\mathcal{E}$, and $\mathcal{D}$;
- 18: **end for**

Phase II: RBF training

- 19: **Collect input:** pre-trained $\mathcal{E}$ and CCE;
 - 20: $\mathbf{M}_{ur} = \text{CCE}(\mathbf{M}_u)$;
 - 21: $\mathbf{D}_{ur} = \text{CCE}(\mathbf{D}_u)$;
 - 22: $\mathbf{y}_D = \mathcal{E}(\mathbf{f}_D)$;
 - 23: $\mathbf{C}_{ur}^{\text{train}}(t_n) = \text{CCE}(\mathbf{C}_u(\mathbf{u}_n))$ for $n = 0, \dots, N_\mu$;
 - 24: $\mathbf{y}_C^{\text{train}}(t_n) = \mathcal{E}(\mathbf{f}_C(\mathbf{u}_n))$ for $n = 0, \dots, N_\mu$;
 - 25: $\mathbf{z}_0^{\text{train}} = \mathcal{E}(\mathbf{u}_0)$;
 - 26: **for** $n \in [0, \dots, N_\mu - 1]$ **do**
 - 27: $\mathbf{A}_r \leftarrow \mathbf{M}_{ur} + \Delta t(\mathbf{C}_{ur} + \mathbf{D}_{ur})$;
 - 28: $\mathbf{y} \leftarrow \mathbf{y}_C + \mathbf{y}_D + \mathbf{M}_{ur}\mathbf{z}_n^{\text{train}}$;
 - 29: Compute $\mathbf{z}_{n+1}^{\text{train}}$ as the solution of $\mathbf{A}_r\mathbf{z}_{n+1}^{\text{train}} = \mathbf{y}$;
 - 30: **end for**
 - 31: **Input (RBF train):** $[\mathbf{z}_1^{\text{train}}, \dots, \mathbf{z}_{N_\mu}^{\text{train}}], [\mathbf{y}_C^{\text{train}}(t_1), \dots, \mathbf{y}_C^{\text{train}}(t_{N_\mu})],$
 $[\mathbf{C}_{ur}^{\text{train}}(t_1), \dots, \mathbf{C}_{ur}^{\text{train}}(t_{N_\mu})];$
 - 32: Train $\text{RBF}_{\text{mat}} : \mathbf{z}_n^{\text{train}} \mapsto \mathbf{C}_{ur}^{\text{train}}(t_n)$;
 - 33: Train $\text{RBF}_{\text{vec}} : \mathbf{z}_n^{\text{train}} \mapsto \mathbf{y}_C^{\text{train}}(t_n)$.
-

Algorithm 7 Pseudo-code for the online prediction of the CCE-AE-RBF approach.

- 1: **Input:** Pre-trained continuous convolutional encoder CCE, MLP encoder $\mathcal{E}$ and decoder $\mathcal{D}$;
 - 2: **Initial FOM state:** $\mathbf{u}_0 = \mathbf{u}(t_0)$, time steps $t_n, n = 1, \dots, N_T$
 - 3: $\mathbf{M}_{ur} = \text{CCE}(\mathbf{M}_u)$;
 - 4: $\mathbf{D}_{ur} = \text{CCE}(\mathbf{D}_u)$;
 - 5: $\mathbf{y}_{D_r} = \mathcal{E}(\mathbf{f}_D)$;
 - 6: $\mathbf{z}_0 = \mathcal{E}(\mathbf{u}_0)$;
 - 7: **for** $n \in [0, \dots, N_T - 1]$ **do**
 - 8: $\mathbf{C}_{ur} \leftarrow \text{RBF}_{\text{mat}}(\mathbf{z}_n)$;
 - 9: $\mathbf{y}_{C_r} \leftarrow \text{RBF}_{\text{vec}}(\mathbf{z}_n)$;
 - 10: $\mathbf{A}_r \leftarrow \mathbf{M}_{ur} + \Delta t(\mathbf{C}_{ur} + \mathbf{D}_{ur})$;
 - 11: $\mathbf{y} \leftarrow \mathbf{y}_C + \mathbf{y}_D + \mathbf{M}_{ur}\mathbf{z}_n$;
 - 12: Compute $\mathbf{z}_{n+1}$ as the solution of $\mathbf{A}_r\mathbf{z}_{n+1} = \mathbf{y}$.
 - 13: Reconstruct the solution as $\mathcal{D}(\mathbf{z}_{n+1})$.
 - 14: **end for**
-

Figure 10.17 shows the validation loss decay during training, comparing the training process of a standard AE and of the proposed CCE-AE approach. Since this test case is particularly challenging, both the AE and CCE-AE validation losses converge to relatively high values (approximately 10^{-2}). This indicates that the extrapolation task is inherently difficult, as even the unconstrained AE struggles to accurately reconstruct unseen samples. However, we can observe that, also in this case, for the smallest latent dimension ($r = 2$), the CCE-AE validation loss converges to a lower value than that of the AE. This suggests that coupling the autoencoder with the consistency constraint improves the robustness of the training process.

We now analyse the predictive accuracy of the proposed model, by testing it in a time extrapolation setting, namely at $t_n, n \in [75, 100]$. Figure 10.18 represents the time evolution of the relative L^2 errors of the approximated solution with respect to the FOM solution, for different POD-based, and AE-based approaches. As can be seen, linear approaches such as the POD fail to accurately reconstruct the solution in an under-resolved regime. Introducing nonlinearities significantly improves the reconstruction capability: within the training window, the AE achieves reconstruction errors one order of magnitude smaller than those of the POD for both $r = 2$ and $r = 3$. Nevertheless, when the solution evolves beyond the training regime, the AE fails to preserve the correct temporal dynamics, resulting in inaccurate extrapolations. Introducing reduced-order interpolation through RBF (POD-RBF and AE-RBF) further deteriorates the reconstruction accuracy. This can be attributed to the difficulty of approximating the mapping from the temporal variable t to the reduced coefficients $\mathbf{z}(t)$. In fact, the reduced trajectories may exhibit complex and nonlinear behaviours that cannot be accurately represented by the chosen interpolation approach. Consequently, these approaches become unreliable for prediction beyond the training regime and cannot be effectively employed for extrapolation.

A POD-Galerkin approach is also analysed. We adopt an *inefficient* formulation, in

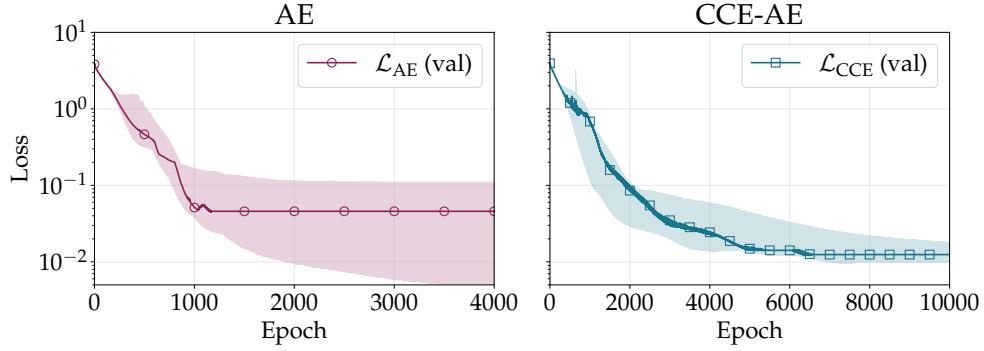
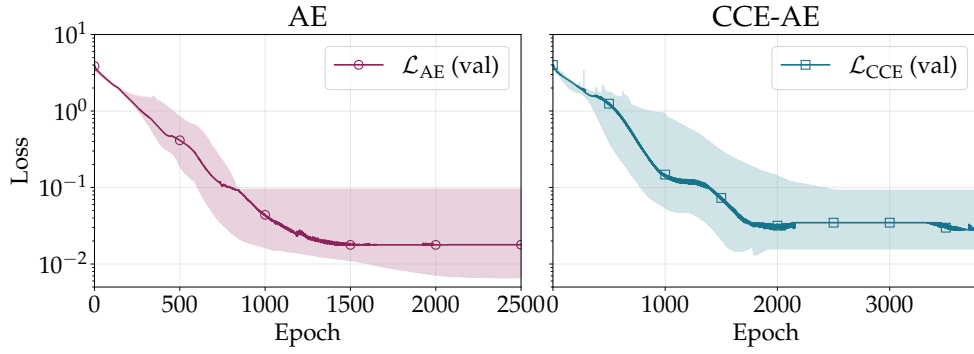
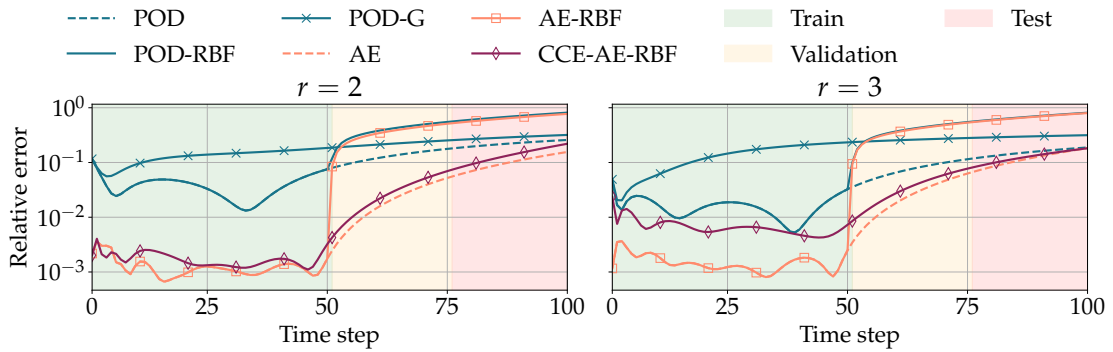

 (a) Latent dimension $r = 2$

 (b) Latent dimension $r = 3$

 Figure 10.17: *Burgers test case*. Loss trend across epochs for a standard AE, and for the novel model CCE-AE, at different latent dimensions, namely $r = 2$ and 3.

 Figure 10.18: *Burgers test case*. Time variation of the relative L^2 errors with respect to the FOM solution, obtained using standard methodologies (POD, POD-RBF, POD-G, AE, AE-RBF), and the novel CCE-AE-RBF approach.

which the full-order quantities $\mathbf{C}_u$ and $\mathbf{f}_C$ are projected onto the **POD** modes, also in the predictive regime. Despite the use of full-order quantities, the method already exhibits large errors from the beginning of the prediction window, indicating that the reduced formulation is not able to accurately capture the solution dynamics.

Finally, the **CCE-AE-RBF** approach is considered. Although the error still increases over time beyond the training window, the proposed interpolation strategies $\mathbf{z}(t) \mapsto \mathbf{y}_C(t)$ and $\mathbf{z}(t) \mapsto \mathbf{C}_{u,r}(t)$ provide a more accurate approximation of the reduced quantities, keeping the prediction error closer to the reconstruction error. This suggests that the main limitation is not introduced by the **RBF** interpolation itself, but rather by the difficulty of the autoencoder in representing and propagating the nonlinear dynamics of the solution.

We can summarize the main findings here:

- **The main source of error appears to be related to the autoencoder representation.**
Due to the strongly nonlinear nature of the Burgers dynamics, the latent space learned by the **AE** is not sufficiently accurate to preserve the temporal evolution of the solution in the predictive regime. Therefore, the observed degradation is mainly attributed to the difficulty of the underlying nonlinear reduction rather than to the proposed architecture.
- **The **RBF** interpolation strategy provides a more suitable approximation framework.**
Unlike approaches based on the direct mapping $t \mapsto \mathbf{z}(t)$, the proposed method interpolates the reduced operators as a function of the latent coordinates. This mapping is more consistent with the reduced-order formulation and introduces a smaller additional error, allowing the predictions to remain closer to the intrinsic reconstruction capability of the CCE-AE.

Finally, Figure 10.19 compares the graphical approximations obtained by using different **ROMs** at fixed latent dimension $r = 3$, at different time steps (in the predictive setting). As expected, the **POD-RBF** and **AE-RBF** approximations are totally unphysical, and are not following the time evolution of the **FOM** solution. On the other hand, the **CCE-AE-RBF** approach, although characterized by some spurious oscillations, is able to correctly reproduce the overall behaviour of the **FOM**, suggesting that the physics-based **ROM** formulation provides a more accurate representation of the underlying dynamics. The oscillations may be related to the limited capacity of the decoder, which may require a larger architecture (e.g., more layers and neurons). Different decoder architectures will be explored in future works.

10.2.4 Computational details

This section describes the computational aspects of the proposed methods, including the architecture and hyperparameters' setup of the neural networks, and the training cost (number of parameters and wall time).

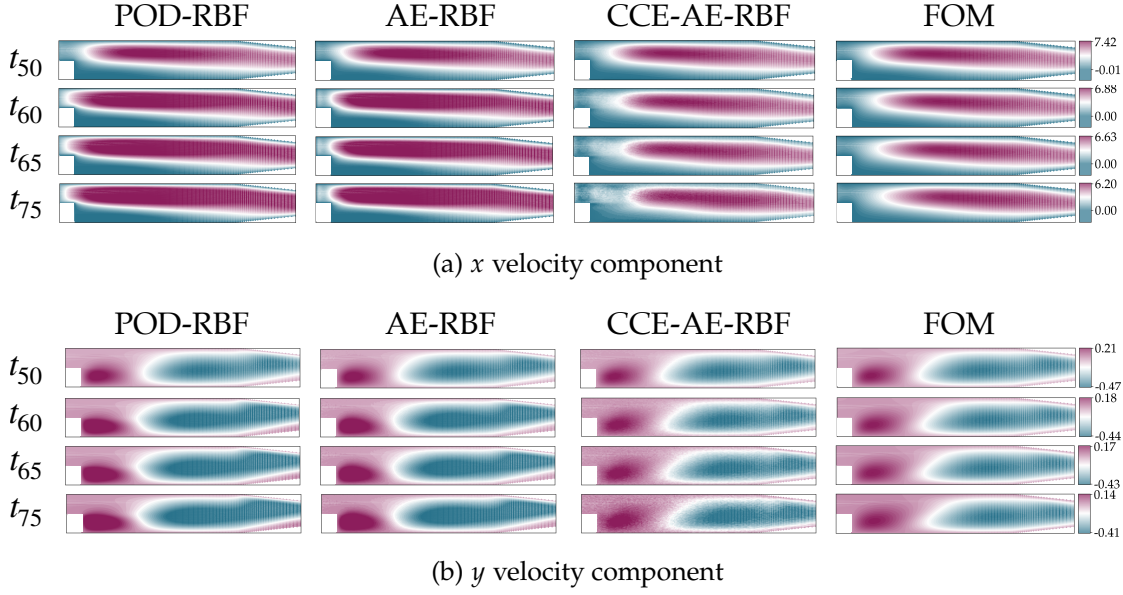


Figure 10.19: *Burgers test case*. Solution fields for the x (top) and y (bottom) components, found using POD-RBF, AE-RBF, CCE-AE-RBF, compared with the FOM. The fields are shown at different time steps in a predictive regime, at the fixed latent dimension $r = 3$.

The training process has been performed using the SOAP optimizer [346] using one GPU NVIDIA RTX A2000 (12 GB VRAM), CUDA 12.4.

The AE is a deep fully-connected neural network with input and output dimensions equal to N . For the first two test cases, the encoder consists of hidden layers with $[100, 30, r]$ neurons, whereas for the Burgers test case we adopt a more compact architecture, $[10, r]$. In both cases, the decoder has a symmetric architecture to the encoder, and we adopt the Softplus as activation function. Preliminary numerical experiments showed that increasing the network size for the Burgers problem does not provide any significant improvement in reconstruction accuracy, while it increases the risk of overfitting. Therefore, a smaller architecture is adopted for this test case.

The convolutional encoders used to compress the FOM matrices have different architectures, depending on the test case and on the value of r . The architectures of the CEs and CCEs are summarized in the Tables 10.1, 10.2, and 10.3, for the three test cases, respectively.

The proposed architectures include a sequence of 2D convolutional layers, activation functions, max pooling operations, and batch normalization. A continuous convolutional layer is considered in the Burgers test case (Table 10.3) to efficiently process sparse operators. Here, N_b denotes the batch size. In the previous test cases, $N_b = 1$ since no parametrized operators were considered. The quantity N_{nz} represents the number of non-zero entries of the operator. As described in Section 10.1.1, the operator is represented using three input columns corresponding to the row indices, column indices, and values of the non-zero entries. The filter of the continuous convolutional layer ContConv1 is a deep fully-connected neural network of input dimension 2 (row and column indexes), hidden

Table 10.1: *Poisson test case*. Architectures of CE architectures, in terms of kernel, stride, padding, and activation function for each layer.

Layer	Input dim	Output dim	Kernel size	Stride	Padding	Activation
Conv1	(1, 638, 638)	(1, 127, 127)	(10, 10)	(5, 5)	(1, 1)	Softplus
Conv2	(1, 127, 127)	(1, 42, 42)	(5, 5)	(3, 3)	(1, 1)	Softplus
Conv3	(1, 42, 42)	(1, 20, 20)	(5, 5)	(2, 2)	(1, 1)	Softplus
Conv4 ($r = 2$)	(1, 20, 20)	(1, 7, 7)	(4, 4)	(3, 3)	(1, 1)	Softplus
Conv5 ($r = 2$)	(1, 7, 7)	(1, 4, 4)	(3, 3)	(2, 2)	(1, 1)	Softplus
Conv6 ($r = 2$)	(1, 4, 4)	(1, 2, 2)	(3, 3)	(2, 2)	(1, 1)	$\times$
Conv4 ($r = 3$)	(1, 20, 20)	(1, 10, 10)	(3, 3)	(2, 2)	(1, 1)	Softplus
Conv5 ($r = 3$)	(1, 10, 10)	(1, 5, 5)	(3, 3)	(2, 2)	(1, 1)	Softplus
Conv6 ($r = 3$)	(1, 5, 5)	(1, 3, 3)	(3, 3)	(2, 2)	(1, 1)	$\times$
Conv4 ($r = 4$)	(1, 20, 20)	(1, 7, 7)	(4, 4)	(3, 3)	(1, 1)	Softplus
Conv5 ($r = 4$)	(1, 7, 7)	(1, 4, 4)	(3, 3)	(2, 2)	(1, 1)	$\times$

Table 10.2: *Advection-diffusion test case*. Architectures of CE architectures, in terms of kernel, stride, padding, and activation function for each layer.

Layer	Input dim	Output dim	Kernel size	Stride	Padding	Activation
Conv1	(1, 2500, 2500)	(1, 623, 623)	(10, 10)	(4, 4)	(0, 0)	Softplus
MaxPool1	(1, 623, 623)	(1, 154, 154)	(8, 8)	(4, 4)	$\times$	$\times$
Conv2	(1, 154, 154)	(1, 50, 50)	(5, 5)	(3, 3)	(0, 0)	Softplus
MaxPool2	(1, 50, 50)	(1, 16, 16)	(3, 3)	(3, 3)	$\times$	$\times$
Conv3 ($r = 2$)	(1, 16, 16)	(1, 7, 7)	(3, 3)	(2, 2)	(0, 0)	$\times$
MaxPool3 ($r = 2$)	(1, 7, 7)	(1, 2, 2)	(4, 4)	(2, 2)	$\times$	$\times$
Conv3 ($r = 3$)	(1, 16, 16)	(1, 7, 7)	(3, 3)	(2, 2)	(0, 0)	$\times$
MaxPool3 ($r = 3$)	(1, 7, 7)	(1, 3, 3)	(3, 3)	(2, 2)	$\times$	$\times$
Conv3 ($r = 4$)	(1, 16, 16)	(1, 14, 14)	(3, 3)	(1, 1)	(0, 0)	$\times$
MaxPool3 ($r = 4$)	(1, 14, 14)	(1, 4, 4)	(4, 4)	(3, 3)	$\times$	$\times$

layers with neurons $[10, 10, 10]$, and output dimension 1 (the continuous filter value).

Table 10.3: *Burgers test case*. Architectures of CCE architectures, in terms of kernel, stride, padding, and activation function for each layer.

Layer	Input dim	Output dim	Kernel size	Stride	Padding	Activation
ContConv1	$(N_b, 1, N_{nz}, 3)$	$(N_b, 1, 25, 3)$	$(4890, 4890)$	$\times$	$\times$	$\times$
Reshape	$(N_b, 1, 25, 3)$	$(N_b, 3, 5, 5)$	$\times$	$\times$	$\times$	$\times$
Conv1 ($r = 2$)	$(N_b, 3, 5, 5)$	$(N_b, 1, 2, 2)$	$(3, 3)$	$(2, 2)$	$(0, 0)$	Softplus
BatchNorm1 ($r = 2$)	$(N_b, 1, 2, 2)$	$(N_b, 1, 2, 2)$	$\times$	$\times$	$\times$	$\times$
MaxPool1 ($r = 3$)	$(N_b, 3, 5, 5)$	$(N_b, 3, 4, 4)$	$(2, 2)$	$(1, 1)$	$\times$	$\times$
Conv1 ($r = 3$)	$(N_b, 3, 4, 4)$	$(N_b, 1, 3, 3)$	$(2, 2)$	$(1, 1)$	$(0, 0)$	Softplus
BatchNorm1 ($r = 3$)	$(N_b, 1, 3, 3)$	$(N_b, 1, 3, 3)$	$\times$	$\times$	$\times$	$\times$

All the neural networks are trained with a learning rate 1×10^{-3} , and using an early stopping based on the validation loss value. In all the test cases, a weight decay regularization with coefficient 1×10^{-2} was employed to improve the generalization capability of the model.

Finally, we present in Table 10.4 the computational cost needed by the novel **CE-AE**, compared with the cost of a standard **AE** with the same architecture employed for the **AE** in the novel model. Regarding the computational time, for the first two test cases the

Table 10.4: Training times, number of parameters, and accuracy in terms of average test error, of the AE, CE-AE, S-CE-AE (for the second test case), and CCE-AE models. The test errors are computed by interpolating the latent variables with an RBF approach.

Test case	Model	Average training time			Number of parameters			Average L^2 test error		
		$r = 2$	$r = 3$	$r = 4$	$r = 2$	$r = 3$	$r = 4$	$r = 2$	$r = 3$	$r = 4$
1	AE	24.6	381	370	$\sim 134K$			0.152	0.026	0.026
	CE-AE	221	606	116	+190	+183	+180	0.041	0.035	0.025
	S-CE-AE	245	527	109	+190	+183	+180	0.045	0.032	0.028
2	AE	920	999	1078	$\sim 516K$			0.032	0.024	0.010
	CE-AE	1331	1461	1882	+219	+212	+219	0.0035	0.0082	0.0030
3	AE	55.1	39		$\sim 513K$			0.50	0.52	
	CCE-AE	9889	3454		+399	+384		0.085	0.084	

proposed models exhibit a slightly higher computational cost compared to the **AE**, while

remaining of the same order of magnitude. This increase is mainly due to the additional convolutional encoder, which introduces only a limited number of trainable parameters compared to the **AE**. Therefore, the increase in model complexity remains marginal.

Conversely, for the Burgers' case, the computational cost is higher due to two main factors: the presence of the continuous convolutional layer, which involves more expensive routines (e.g., the filter mapping function), and the time-loop integration within the training procedure, which requires the reduced dynamics to be evaluated at each epoch. Nevertheless, also in this case, the number of additional trainable parameters introduced with respect to the **AE** remains negligible.

Despite the additional computational effort, the proposed approaches achieve improved accuracy compared to the **AE** in the testing phase, except for the first test case with $r = 3$.

10.3 Conclusions and Future Perspectives

In this chapter, we introduced a novel perspective on **nonlinear model order reduction** based on the compression of full-order operators through convolutional neural network architectures. Unlike most existing nonlinear **ROM** methodologies, which focus on learning nonlinear representations of the solution manifold, the proposed framework aims at learning compact representations of the operators defining the governing equations. This preserves the algebraic structure of the reduced problem, allowing the reduced system to be solved through standard numerical techniques while extending classical projection-based **ROMs** beyond linear or quadratic subspaces.

The proposed methodology was assessed on three benchmark problems of increasing complexity. The numerical experiments demonstrated that the operator-compression strategy is capable of producing accurate **ROMs** while maintaining a compact latent representation. Moreover, the results highlighted that incorporating the solution of the reduced system directly into the training process improves the quality of the learned latent space with respect to a standard fully-connected **AE**.

Interestingly, even without explicitly enforcing structural constraints, the proposed architectures naturally learned reduced operators that reflected the algebraic properties of the corresponding full-order operators. In the Poisson test case, symmetric full-order operators consistently led to nearly symmetric reduced operators, indicating that these structural features can emerge naturally during training.

Furthermore, incorporating the reduced-system solution into the training objective acts as an effective **stabilization mechanism** for the autoencoder optimization. Although it introduces an additional loss contribution, it guides the optimization towards more informative latent representations, leading to improved training stability and, in several cases, to lower reconstruction losses than those achieved by a standard autoencoder.

Finally, the introduction of continuous convolutional layers provided an effective strategy for exploiting the sparsity of differential operators, making the approach scalable to

large discretized systems and naturally applicable to operators defined on unstructured meshes.

Although the proposed framework should be regarded as a preliminary step towards nonlinear operator-based ROMs, the obtained results suggest several promising research directions. A first natural extension consists in applying the methodology to **more realistic and large-scale CFD problems**, including the incompressible Navier–Stokes equations and multiphysics applications. Furthermore, more expressive parameter-to-latent mappings, such as neural networks, could replace the interpolation strategies adopted in this work, improving the generalization capability over complex parameter spaces.

Finally, **combining continuous convolutional networks for both the operators and the solution fields** would provide a fully discretization-independent framework, enabling nonlinear ROMs that naturally generalize across different computational grids.

CONCLUSIONS

In this thesis, we investigated the development of several **stabilization strategies** for convection-dominated flows, spanning from filtering approaches to closure models.

We investigate these techniques in different computational settings, from coarse-grid simulations to reduced order models. Both cases tackle the same underlying challenge, namely the accurate approximation of under-resolved flow dynamics, arising either from insufficient spatial resolution (in full-order simulations) or from severe dimensionality reduction in **ROMs**.

To address this issue, we proposed a series of data-driven and machine-learning-based methodologies that enhance classical stabilization techniques, and share a common objective: *replacing fixed modeling assumptions with models capable of learning from data while preserving, whenever possible, the mathematical structure of the governing equations.*

The **first part** of the thesis focused on the stabilization of marginally-resolved simulations through the **EFR** framework. Rather than considering the filtering operation as a fixed numerical ingredient, this work demonstrated that the optimal filter radius and relaxation parameter are not fixed quantities, but rather depend on the flow characteristics, the simulation regime, and the evolving dynamics of the system. These optimal parameters can be identified either through optimization techniques or by learning suitable policies from data and from the numerical simulations themselves. In particular:

- ▶ We introduced the **Opt-EFR** algorithm, which optimizes the parameters at each time step to obtain an **EFR** solution as close as possible to a given reference **DNS**. This investigation revealed that the optimal parameter values vary significantly in time, and that in most cases the optimal strategy corresponds to applying no filtering.
- ▶ We extended the **Opt-EFR** approach to enable efficient extrapolation in time by employing an **RL** agent to predict the optimal filter parameters. We explored both a data-driven reward, based on **DNS** data, and a data-free reward, based on physics-based metrics computed from the evolving coarse simulation. Both **RL-EF** methodologies achieved accurate and efficient long-time extrapolation in two-dimensional convection-dominated flows.

These developments showed that stabilization can be interpreted as a dynamic process, whose optimal behaviour depends on the instantaneous flow configuration instead of pre-determined empirical parameters.

Furthermore, we introduced a fully data-driven evolve-filter algorithm (**DD-EF**), where the classical differential filter is replaced by a data-driven operator learned from **DNS**

simulations and combined with a structure-preserving relaxation procedure enforcing energy and enstrophy preservation. This approach provided more accurate and robust approximations than traditional filters, while reducing the computational cost and improving its suitability for practical applications.

Building upon these ideas, the **second part** of the thesis extended the use of data-driven strategies to reduced-order modeling, targeting the improvement of existing stabilization and correction techniques for under-resolved **ROMs**, where a reduced number of modes is employed.

First, similarly to **DD-EF**, we replaced the classical differential **ROM** filter in the **L-ROM** formulation with a data-driven stabilization operator (**StabOp**). We investigated linear, quadratic, and nonlinear formulations, all of which provided improved accuracy compared with classical **ROM** filters, including in challenging three-dimensional turbulent flows such as the minimal channel flow.

We then investigated a complementary stabilization strategy based on data-driven closure (or *correction*) models, aimed at compensating for the effects of unresolved scales. More specifically:

- ▶ With the **DD-EV-ROM**, we extended existing closure strategies for **POD-Galerkin ROMs** to parametrized settings by exploiting neural operators. The combination of data-driven correction and turbulence models, both represented through neural operators, resulted in stable and accurate predictions for unseen configurations.
- ▶ We further generalized the neural operator-based correction approach to non-intrusive **POD-based ROMs**, with the **QuadNet** and **QuadNet- μ** architectures. In this framework, the correction term is directly applied to the reduced approximation rather than to the reduced equations, enabling its use in scenarios where the underlying governing equations are not available.

The obtained results consistently demonstrated that nonlinear data-driven corrections significantly improve the predictive capabilities of reduced-order models across different flow configurations and parameter regimes, while preserving their computational efficiency.

The **third part** of the thesis explored a broader perspective on **nonlinear model reduction**. The previous investigations showed that, although data-driven stabilization and correction strategies can significantly improve the predictive capabilities of classical **ROMs**, they remain constrained by the underlying linear **POD** representation. To overcome this limitation, we moved beyond the paradigm of correcting existing reduced models and investigated the direct learning of compact nonlinear representations of the **FOM** operators defining the governing equations.

This shift represents a conceptual evolution from learning stabilization terms to learning numerical operators themselves, extending the projection-based philosophy of classical **ROMs**, such as **POD-Galerkin** methods, towards nonlinear latent representations learned from data.

The proposed operator-compression framework combines classical **AEs** with convolutional encoders, exploiting the intrinsic locality and sparsity of differential operators to construct compact latent representations while preserving the algebraic structure required to solve the reduced system. Moreover, the coupling between the learned operators and the intrusive reduced formulation acts as an additional regularization mechanism during training, improving the stability of the learning process and guiding the neural architecture towards physically meaningful representations.

Taken together, the contributions presented throughout this thesis outline a coherent evolution in the role of machine learning within **CFD**. Rather than replacing physics-based numerical methods, machine learning is employed as a mathematical tool to identify quantities that are traditionally prescribed empirically, including stabilization parameters, filtering operators, closure models, and eventually the numerical operators themselves. From this perspective, stabilization progressively evolves from a *fixed numerical correction* into a *learnable mathematical component* of the computational model.

Methodological considerations

The methodologies developed throughout this thesis operate at different levels of the computational model, ranging from **FOM** to **ROM**, and therefore exhibit different advantages and limitations. Their suitability depends on several factors, including the availability of high-fidelity data and of the governing equations, the computational cost and training requirements, the online efficiency, and the possibility of incorporating mathematical or structure-preserving constraints. A summary of the main advantages, limitations, and suitable application scenarios of each methodology introduced in the different chapters is reported in Table 11.

The proposed methodologies should not, however, be regarded as standalone or mutually exclusive approaches, as they can be combined or transferred across different levels. For example, the data-driven filter introduced within **DD-EF** can also be constructed in **POD-Galerkin ROMs**. In this setting, the resulting linear filter is closely related to the linear formulation considered in **StabOp-L-ROM**. The two approaches nevertheless differ in the training procedure: in **StabOp-L-ROM**, the optimization is performed over the time evolution of the reduced model, whereas in **DD-EF** the filter is learned independently of the time-integration loop.

Conversely, the data-driven stabilization operator learned within **StabOp-L-ROM** could in principle also be applied at the **FOM** level as a mapping from the velocity field to its filtered counterpart, for instance within either the **EFR** or Leray formulations.

Similarly, the adaptive filter developed through **RL-EF** can naturally be extended to the reduced-order setting. In this case, the state provided to the **RL** agent could be represented by the reduced velocity coefficients rather than by the full-order field, while the reward could be defined by comparing the **ROM-EF** solution with the projection of the reference **DNS** onto the **POD** space. This would provide an adaptive **ROM** stabilization

Table 11: Summary of the main methodologies developed in this thesis, including their main advantages and limitations, and the scenarios in which each approach is most suitable.

Method	Main advantages	Main limitations	When suitable
Opt-EFR (Chapter 2)	Ideal prediction of optimal δ and χ	Needs all DNS data	Preliminary analysis of ideal δ and χ values
RL-EF (Chapter 3)	Needs no/limited DNS data Possible generalization to different settings (ICs, Re , ...)	Needs access to the solver and communication with the RL environment	Long-time simulations in case of no/limited DNS data
DD-EF(R) (Chapter 4)	Efficient filter Generalization to different settings (ICs, Re , ...) Energy/enstrophy-conserving	Needs DNS data Requires large memory in the case of large grids	Simulations with limited grid size, with available DNS data (currently only with periodic boundary conditions)
StabOp-L-ROM and DD-EV-ROM (Chapters 6 and 7)	Efficient training of NNs Generalization to different settings (ICs, Re , ...)	Needs access to equations and data (snapshots) Not energy-stable, might blow up	Cases with available data and equations, and where standard POD -Galerkin fails
Corrected-ROM (Chapter 8)	Needs no access to equations Efficient (real-time)	Needs FOM data (snapshots) Less interpretable	Cases with available data, in case of no access to equations
CE-AE-ROM (Chapter 10)	Nonlinear compression of both solutions and operators Preserves the operators' structure Interpretable	Needs access to equations Methodological complexity Currently less mature	Cases with available data and equations, where POD is not accurate

strategy in which the filter parameters are learned according to the instantaneous reduced state.

A similar strategy can be investigated for **Opt-EFR**, where the optimal filter parameters could be identified directly at the reduced level by comparing the **ROM-EFR** solution with the **POD** projection of the reference **DNS**.

Future perspectives

As partially anticipated in the previous section, several research directions naturally arise from the present work.

- ▶ Building upon the results of the first part of the thesis, a natural research direction is the extension of the proposed methodologies to increasingly complex configurations, including fully three-dimensional turbulent flows, for both the **RL-EF** and **DD-EF** approaches. Another important objective is to improve the generalization capability of these methods, first across different computational grids and subsequently across different test cases. For **RL-EF**, this could be achieved by considering space-continuous filter values and exploiting the flexibility of graph neural networks. For **DD-EF**, a possible strategy is to replace the matrix–vector multiplication with a space-continuous trainable nonlinear operator. Finally, an interesting perspective is the development of consistent **EFR-ROM** formulations based on the **DD-EF** and **RL-EF** methodologies. A first step towards the calibration of the filter at the reduced level has already been explored in the Master’s thesis [296].
- ▶ For the methodologies presented in the second part of the thesis, namely **DD-EV-ROM**, **StabOp-L-ROM**, and **QuadNet- μ** , a promising direction is the incorporation of mathematical and physical constraints directly into the learning process. Enforcing properties such as conservation laws, energy stability, symmetry, and other operator-specific structures would improve both the robustness and the interpretability of the resulting data-driven reduced-order models.
- ▶ The nonlinear **ROM** framework introduced in the last part of the thesis naturally opens several research perspectives. A primary objective is its extension to more complex systems, such as the Stokes and Navier–Stokes equations, investigating how both the solution manifold and the associated full-order operators can be efficiently compressed while preserving the accuracy of the reduced dynamics. In parallel, we plan to further develop the continuous convolutional architecture, with the goal of improving its efficiency and extending its applicability to simulations performed on different computational grids.

The overarching goal of this thesis is to bridge classical numerical methods and machine learning, leveraging their complementary strengths to develop accurate, robust, and

efficient surrogate models. This research vision has also been fostered through the collaborations established during the PhD, including the joint activities with Prof. Cinnella and collaborators [149, 362], where similar challenges have been addressed in the context of nonintrusive ROMs. Looking ahead, these research directions naturally align with the objectives of the **ROSA** (Reduced Order and Surrogate methods for advanced Applications) project, recently funded through the Italian FIS programme, which will focus on the development of next-generation surrogate modeling techniques for reliable and efficient simulations of real-world and industrial systems. The methodologies developed in this thesis will therefore constitute a starting point for future research within this broader framework, contributing to the design of accurate and data-enhanced computational models for complex engineering applications.

LIST OF THESIS PUBLICATIONS

- ▶ **A. Ivagnes**, M. Strazzullo, M. Girfoglio, T. Iliescu, and G. Rozza. “Data-Driven Optimization for the Evolve-Filter-Relax Regularization of Convection-Dominated Flows”. In: *International Journal for Numerical Methods in Engineering* 126.9 (2025), DOI: [10.1002/nme.70042](https://doi.org/10.1002/nme.70042).
Candidate contribution: Conceptualization, Methodology, Formal analysis, Software, Writing - original draft.
- ▶ **A. Ivagnes**, M. Strazzullo, and G. Rozza. “Reinforcement Learning-Based Filters for Convection-Dominated Flows: Reference-Free and Reference-Guided Training”. In: *arXiv preprint arXiv:2603.02086*, under review (2026).
Candidate contribution: Conceptualization, Methodology, Software, Writing - original draft.
- ▶ **A. Ivagnes**, T. Van Gastelen, S. D. Agdestein, B. Sanderse, G. Stabile, and G. Rozza. “A new data-driven energy-stable evolve-filter-relax model for turbulent flow simulation”. In: *Computer Methods in Applied Mechanics and Engineering* 450 (2026), p. 118654, DOI: [10.1016/j.cma.2025.118654](https://doi.org/10.1016/j.cma.2025.118654).
Candidate contribution: Writing – original draft, Software, Methodology, Conceptualization.
- ▶ P.-H. Tsai, **A. Ivagnes**, A. Quaini, T. Iliescu, and G. Rozza. “StabOp: A Data-Driven Stabilization Operator for Reduced Order Modeling”. In: *Computer Methods in Applied Mechanics and Engineering* 457 (2026), DOI: [10.1016/j.cma.2026.118990](https://doi.org/10.1016/j.cma.2026.118990).
Candidate contribution: Methodology, Formal analysis, Software, Conceptualization, Writing – review and editing, Writing – original draft.
- ▶ **A. Ivagnes**, G. Stabile, and G. Rozza. “Data-driven Closure Strategies for Parametrized Reduced Order Models via Deep Operator Networks”. In: *Computer Methods in Applied Mechanics and Engineering* 455 (2026), p. 118900, DOI: [10.1016/j.cma.2026.118900](https://doi.org/10.1016/j.cma.2026.118900).
Candidate contribution: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft.
- ▶ G. Codega, **A. Ivagnes**, N. Demo, and G. Rozza. “Machine Learning-based quadratic closures for non-intrusive Reduced Order Models”. In: *SIAM Journal on Scientific Computing* 48.3 (2026): C505-C525, DOI: [10.1137/25M1766759](https://doi.org/10.1137/25M1766759).
Candidate contribution: Supervision, Conceptualization, Methodology, Writing - review and editing.

BIBLIOGRAPHY

- [1] F. M. A. Acosta. “Radial basis function and related models: an overview”. In: *Signal Processing* 45.1 (1995), pp. 37–58 (cit. on p. 173).
- [2] S. D. Agdestein, S. Ciarella, B. Sanderse, and R. Hoekstra. *IncompressibleNavier-Stokes.jl*. Version 2.1.0. 2024-11-09. DOI: [10.5281/zenodo.14217342](https://doi.org/10.5281/zenodo.14217342) (cit. on pp. 5, 17, 47, 73).
- [3] S. D. Agdestein and B. Sanderse. “A differentiable software suite for accelerated simulation of turbulent flows”. In: *arXiv preprint arXiv:2604.18536* (2026) (cit. on pp. 5, 17, 73).
- [4] S. D. Agdestein and B. Sanderse. “Discretize first, filter next: Learning divergence-consistent closure models for large-eddy simulation”. In: *Journal of Computational Physics* 522 (2025-02), p. 113577. ISSN: 0021-9991. DOI: [10.1016/j.jcp.2024.113577](https://doi.org/10.1016/j.jcp.2024.113577) (cit. on pp. 17, 21, 73, 75, 88).
- [5] S. D. Agdestein and B. Sanderse. “Learning filtered discretization operators: non-intrusive versus intrusive approaches”. In: *arXiv preprint arXiv:2208.09363* (2022) (cit. on p. 72).
- [6] S. E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, and B. R. Noack. “On closures for reduced order models—A spectrum of first-principle to machine-learned avenues”. In: *Physics of Fluids* 33.9 (2021) (cit. on pp. 108, 111, 115).
- [7] S. Ali, F. Ballarin, G. Rozza, et al. “A reduced basis stabilization for the unsteady Stokes and Navier-Stokes equations”. In: *Advances in Computational Science and Engineering* 1.2 (2023), pp. 180–201. DOI: [10.3934/acse.2023008](https://doi.org/10.3934/acse.2023008) (cit. on p. 106).
- [8] S. Ali, F. Ballarin, and G. Rozza. “Stabilized reduced basis methods for parametrized steady Stokes and Navier–Stokes equations”. In: *Computers & Mathematics with Applications* 80.11 (2020), pp. 2399–2416. DOI: [10.1016/j.camwa.2020.03.019](https://doi.org/10.1016/j.camwa.2020.03.019) (cit. on p. 106).
- [9] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. “The FEniCS project version 1.5”. In: *Archive of numerical software* 3.100 (2015) (cit. on pp. 5, 24, 206).
- [10] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. “Unified form language: A domain-specific language for weak formulations of partial differential equations”. In: *ACM Transactions on Mathematical Software (TOMS)* 40.2 (2014), pp. 1–37 (cit. on pp. 5, 55).

- [11] S.-i. Amari. “Backpropagation and stochastic gradient descent method”. In: *Neurocomputing* 5.4-5 (1993), pp. 185–196 (cit. on p. 48).
- [12] D. Amsallem and C. Farhat. “An online method for interpolating linear parametric reduced-order models”. In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2169–2198 (cit. on p. 191).
- [13] D. Amsallem and C. Farhat. “Interpolation method for adapting reduced-order models and application to aeroelasticity”. In: *AIAA journal* 46.7 (2008), pp. 1803–1813 (cit. on p. 191).
- [14] N. Aubry, P. Holmes, J. L. Lumley, and E. Stone. “The dynamics of coherent structures in the wall region of a turbulent boundary layer”. In: *Journal of fluid Mechanics* 192 (1988), pp. 115–173 (cit. on pp. 103, 111, 112).
- [15] G. Aversano, M. Ferrarotti, and A. Parente. “Digital twin of a combustion furnace operating in flameless conditions: reduced-order model development from CFD simulations”. In: *Proceedings of the Combustion Institute* 38.4 (2021), pp. 5373–5381. DOI: [10.1016/j.proci.2020.06.045](https://doi.org/10.1016/j.proci.2020.06.045) (cit. on p. 97).
- [16] M. Azaiez, T. C. Rebollo, M. Oulghelou, and I. S. Muñoz. “Least-squares pressure recovery in reduced order methods for incompressible flows”. In: *Journal of Computational Physics* 519 (2024), p. 113397 (cit. on p. 106).
- [17] I. Babuška. “The finite element method with penalty”. In: *Mathematics of computation* 27.122 (1973), pp. 221–228 (cit. on p. 104).
- [18] P. J. Baddoo, B. Herrmann, B. J. McKeon, J. N. Kutz, and S. L. Brunton. “Physics-informed dynamic mode decomposition (piDMD)”. In: *arXiv preprint arXiv:2112.04307* (2021) (cit. on p. 198).
- [19] F. Ballarin, A. Manzoni, A. Quarteroni, and G. Rozza. “Supremizer stabilization of POD–Galerkin approximation of parametrized steady incompressible Navier–Stokes equations”. In: *International Journal for Numerical Methods in Engineering* 102.5 (2015), pp. 1136–1161 (cit. on pp. 104, 105).
- [20] C. Balzotti, P. Siena, M. Girfoglio, G. Stabile, J. Dueñas-Pamplona, J. Sierra-Pallares, I. Amat-Santos, and G. Rozza. “A reduced order model formulation for left atrium flow: an atrial fibrillation case”. In: *Biomechanics and Modeling in Mechanobiology* 23.4 (2024), pp. 1411–1429 (cit. on p. 97).
- [21] I. A. Baratta, J. P. Dean, J. S. Dokken, M. Habera, J. HALE, C. N. Richardson, M. E. Rognes, M. W. Scroggs, N. Sime, and G. N. Wells. “DOLFINx: the next generation FEniCS problem solving environment”. In: (2023) (cit. on pp. 5, 44, 55).
- [22] J. Bardina, J. Ferziger, and W. Reynolds. “Improved subgrid-scale models for large-eddy simulation”. In: *13th fluid and plasmadynamics conference*. 1980, p. 1357 (cit. on p. 111).

- [23] J. Barnett and C. Farhat. “Quadratic approximation manifold for mitigating the Kolmogorov barrier in nonlinear projection-based model order reduction”. In: *Journal of Computational Physics* 464 (2022), p. 111348. ISSN: 0021-9991. DOI: [10 . 1016 /j . jcp . 2022 . 111348](https://doi.org/10.1016/j.jcp.2022.111348) (cit. on pp. 170, 191).
- [24] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. “An ‘empirical interpolation’ method: application to efficient reduced-basis discretization of partial differential equations”. In: *Comptes Rendus Mathematique* 339.9 (2004), pp. 667–672 (cit. on p. 102).
- [25] J. W. Barrett and C. M. Elliott. “Finite element approximation of the Dirichlet problem using the boundary penalty method”. In: *Numerische Mathematik* 49.4 (1986), pp. 343–366 (cit. on p. 104).
- [26] G. K. Batchelor. “Computation of the energy spectrum in homogeneous two-dimensional turbulence”. In: *The Physics of Fluids* 12.12 (1969), pp. II–233 (cit. on p. 10).
- [27] Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. Hughes, A. Reali, and G. Scovazzi. “Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows”. In: *Computer methods in applied mechanics and engineering* 197.1-4 (2007), pp. 173–201 (cit. on p. 20).
- [28] P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira. *Model Order Reduction: Volume I: System-and Data-Driven Methods and Algorithms*. De Gruyter, 2021 (cit. on p. 97).
- [29] P. Benner, S. Gugercin, and K. Willcox. “A survey of projection-based model reduction methods for parametric dynamical systems”. In: *SIAM review* 57.4 (2015), pp. 483–531 (cit. on p. 97).
- [30] P. Benner, M. Ohlberger, A. Cohen, and K. Willcox. *Model reduction and approximation: theory and algorithms*. SIAM, 2017 (cit. on p. 97).
- [31] P. Benner, W. Schilders, S. Grivet-Talocia, A. Quarteroni, G. Rozza, and L. Miguel Silveira. *Model Order Reduction: Volume 2: Snapshot-Based Methods and Algorithms*. De Gruyter, 2020 (cit. on p. 97).
- [32] M. Bercovier and M. Engelman. “A finite element for the numerical solution of viscous incompressible flows”. In: *Journal of Computational Physics* 30.2 (1979), pp. 181–201 (cit. on p. 106).
- [33] M. Bergmann and L. Cordier. “Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models”. In: *Journal of Computational Physics* 227.16 (2008), pp. 7813–7840 (cit. on p. 97).
- [34] L. C. Berselli, T. Iliescu, and W. J. Layton. *Mathematics of large eddy simulation of turbulent flows*. Vol. 2. Springer, 2006 (cit. on p. 23).

-
- [35] L. Bertagna, A. Quaini, and A. Veneziani. “Deconvolution-based nonlinear filtering for incompressible flows at moderately large Reynolds numbers”. In: *International Journal for Numerical Methods in Fluids* 81.8 (2016), pp. 463–488 (cit. on pp. 20, 38).
- [36] K. Bizon and G. Continillo. “Reduced order modelling of chemical reactors with recycle by means of POD-penalty method”. In: *Computers & chemical engineering* 39 (2012), pp. 22–32 (cit. on p. 104).
- [37] G. Boffetta and R. E. Ecke. “Two-dimensional turbulence”. In: *Annual review of fluid mechanics* 44.1 (2012), pp. 427–451 (cit. on p. 10).
- [38] A. Boral, Z. Y. Wan, L. Zepeda-Núñez, J. Lottes, Q. Wang, Y.-F. Chen, J. Anderson, and F. Sha. “Neural Ideal Large Eddy Simulation: Modeling Turbulence with Neural Stochastic Differential Equations”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., 2023, pp. 69270–69283. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/dabaded617b3be96c3ed161498a7d71c-Paper-Conference.pdf (cit. on p. 21).
- [39] J. Borggaard, T. Iliescu, and J. P. Roop. “A bounded artificial viscosity large eddy simulation model”. In: *SIAM Journal on Numerical Analysis* 47.1 (2009), pp. 622–645 (cit. on p. 13).
- [40] J. Boussinesq. *Essai sur la théorie des eaux courantes*. Paris: Imprimerie Nationale, 1877 (cit. on pp. 2, 13, 113).
- [41] A. L. Bowers and L. G. Rebholz. “Numerical study of a regularization model for incompressible flow with deconvolution-based adaptive nonlinear filtering”. In: *Computer Methods in Applied Mechanics and Engineering* 258 (2013), pp. 1–12 (cit. on p. 20).
- [42] A. Bowers, L. Rebholz, A. Takhirov, and C. Trenchea. “Improved accuracy in regularization models of incompressible flow via adaptive nonlinear filtering”. In: *International Journal for Numerical Methods in Fluids* 70.7 (2012), pp. 805–828 (cit. on p. 20).
- [43] R. N. Bracewell. *The Fourier Transform and Its Applications*. 3rd. New York: McGraw-Hill, 2000 (cit. on p. 75).
- [44] S. Brivio and N. R. Franco. “Deep Symmetric Autoencoders from the Eckart-Young-Schmidt Perspective”. In: *arXiv preprint arXiv:2506.11641* (2025) (cit. on p. 193).
- [45] A. N. Brooks and T. J. R. Hughes. “Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations”. In: *Computer Methods in Applied Mechanics and Engineering* 32.1–3 (1982), pp. 199–259. DOI: [10.1016/0045-7825\(82\)90071-8](https://doi.org/10.1016/0045-7825(82)90071-8) (cit. on p. 106).

- [46] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022 (cit. on p. 97).
- [47] S. L. Brunton, J. L. Proctor, and J. N. Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937 (cit. on p. 194).
- [48] S. L. Brunton, C. W. Rowley, and D. R. Williams. “Reduced-order unsteady aerodynamic models at low Reynolds numbers”. In: *Journal of Fluid Mechanics* 724 (2013), pp. 203–233 (cit. on p. 97).
- [49] M. Bucelli, A. Zingaro, P. C. Africa, I. Fumagalli, L. Dede’, and A. Quarteroni. “A mathematical model that integrates cardiac electrophysiology, mechanics, and fluid dynamics: Application to the human left heart”. In: *International journal for numerical methods in biomedical engineering* 39.3 (2023), e3678 (cit. on p. 97).
- [50] M. D. Buhmann. “Radial basis functions”. In: *Acta numerica* 9 (2000), pp. 1–38 (cit. on p. 173).
- [51] T. Bui-Thanh, K. Willcox, and O. Ghattas. “Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications”. In: *AIAA journal* 46.10 (2008), pp. 2520–2529 (cit. on p. 97).
- [52] J. M. Burgers. “A mathematical model illustrating the theory of turbulence”. In: *Advances in applied mechanics* 1 (1948), pp. 171–199 (cit. on p. 1).
- [53] Y. Cao, J. Zhu, Z. Luo, and I. M. Navon. “Reduced-order modeling of the upper tropical pacific ocean model using proper orthogonal decomposition”. In: *Computers & mathematics with Applications* 52.8-9 (2006), pp. 1373–1386 (cit. on p. 97).
- [54] M. A. Cardoso, L. J. Durlofsky, and P. Sarma. “Development and application of reduced-order modeling procedures for subsurface flow simulation”. In: *International journal for numerical methods in engineering* 77.9 (2009), pp. 1322–1350 (cit. on p. 97).
- [55] K. Carlberg, M. Barone, and H. Antil. “Galerkin v. least-squares Petrov–Galerkin projection in nonlinear model reduction”. In: *Journal of Computational Physics* 330 (2017), pp. 693–734 (cit. on p. 103).
- [56] K. Carlberg, C. Bou-Mosleh, and C. Farhat. “Efficient non-linear model reduction via a least-squares Petrov–Galerkin projection and compressive tensor approximations”. In: *International Journal for numerical methods in engineering* 86.2 (2011), pp. 155–181 (cit. on p. 103).
- [57] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem. “The GNAT method for nonlinear model reduction: effective implementation and application to computational fluid dynamics and turbulent flows”. In: *Journal of Computational Physics* 242 (2013), pp. 623–647 (cit. on pp. 102, 103).

- [58] G. J. Chandler and R. R. Kerswell. “Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow”. In: *Journal of Fluid Mechanics* 722 (2013), pp. 554–595 (cit. on p. 88).
- [59] M. D. Chekroun, H. Liu, and S. Wang. *Stochastic parameterizing manifolds and non-Markovian reduced equations: stochastic manifolds for nonlinear SPDEs II*. Springer, 2014 (cit. on p. 111).
- [60] D. Chen, Q. Li, and H. Song. “Error analysis of a stable reduced order model based on the proper orthogonal decomposition method for the Allen–Cahn–Navier–Stokes system”. In: *Computer Methods in Applied Mechanics and Engineering* 401 (2022), p. 115661 (cit. on p. 106).
- [61] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. “Neural ordinary differential equations”. In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 194).
- [62] W. Chen, Q. Wang, J. S. Hesthaven, and C. Zhang. “Physics-informed machine learning for reduced-order modeling of nonlinear problems”. In: *Journal of computational physics* 446 (2021), p. 110666 (cit. on p. 203).
- [63] Z. Chen and Y. Xu. “The Petrov–Galerkin and Iterated Petrov–Galerkin Methods for Second-Kind Integral Equations”. In: *SIAM Journal on Numerical Analysis* 35.1 (1998), pp. 406–434 (cit. on p. 103).
- [64] Y. Choi and K. Carlberg. “Space–time least-squares Petrov–Galerkin projection for nonlinear model reduction”. In: *SIAM Journal on Scientific Computing* 41.1 (2019), A26–A58 (cit. on p. 103).
- [65] A. J. Chorin and F. Lu. “Discrete approach to stochastic parametrization and dimension reduction in nonlinear dynamics”. In: *Proceedings of the National Academy of Sciences* 112.32 (2015), pp. 9804–9809 (cit. on p. 111).
- [66] A. J. Chorin. “A numerical method for solving incompressible viscous flow problems”. In: *Journal of computational physics* 135.2 (1997), pp. 118–125 (cit. on p. 106).
- [67] W. Chu, Z. Ghahramani, and C. K. Williams. “Gaussian processes for ordinal regression.” In: *Journal of machine learning research* 6.7 (2005) (cit. on p. 173).
- [68] G. Codega, A. Ivagnes, N. Demo, and G. Rozza. “Machine Learning-based quadratic closures for non-intrusive Reduced Order Models”. In: *SIAM Journal on Scientific Computing* 48.3 (2026), pp. C505–C525 (cit. on pp. 4, 187).
- [69] A. Cohen and R. DeVore. “Approximation of high-dimensional parametric PDEs”. In: *Acta Numerica* 24 (2015), pp. 1–159 (cit. on p. 191).
- [70] C. M. Colciago, S. Deparis, and A. Quarteroni. “Comparisons between reduced order models and full 3D models for fluid–structure interaction problems in haemodynamics”. In: *Journal of Computational and Applied Mathematics* 265 (2014), pp. 120–138 (cit. on p. 97).

- [71] P. Conti, G. Gobat, S. Fresca, A. Manzoni, and A. Frangi. “Reduced order modeling of parametrized systems through autoencoders and SINDy approach: continuation of periodic solutions”. In: *Computer Methods in Applied Mechanics and Engineering* 411 (2023), p. 116072 (cit. on p. 194).
- [72] D. Coscia, A. Ivagnes, N. Demo, and G. Rozza. “Physics-Informed Neural networks for Advanced modeling”. In: *Journal of Open Source Software* 8.87 (2023), p. 5352 (cit. on p. 5).
- [73] D. Coscia, L. Meneghetti, N. Demo, G. Stabile, and G. Rozza. “A continuous convolutional trainable filter for modelling unstructured data”. In: *Computational Mechanics* 72.2 (2023), pp. 253–265 (cit. on pp. 196, 197, 201, 204).
- [74] Z. Dar, J. Baiges, and R. Codina. “Artificial neural network based correction for reduced order models in computational fluid mechanics”. In: *Computer Methods in Applied Mechanics and Engineering* 415 (2023), p. 116232 (cit. on pp. 111, 116).
- [75] H. Dave, L. Cotteleer, and A. Parente. “Physics-informed non-intrusive reduced-order modeling of parameterized dynamical systems”. In: *Computer Methods in Applied Mechanics and Engineering* 443 (2025), p. 118045 (cit. on p. 203).
- [76] P. Davidson. *Turbulence: an introduction for scientists and engineers*. Oxford university press, 2015 (cit. on p. 10).
- [77] V. DeCaria, W. Layton, A. Pakzad, Y. Rong, N. Sahin, and H. Zhao. “On the determination of the grad-div criterion”. In: *Journal of Mathematical Analysis and Applications* 467.2 (2018), pp. 1032–1037 (cit. on p. 38).
- [78] V. DeCaria, T. Iliescu, W. Layton, M. McLaughlin, and M. Schneier. “An artificial compression reduced order model”. In: *SIAM Journal on Numerical Analysis* 58.1 (2020), pp. 565–589 (cit. on p. 106).
- [79] N. Demo, M. Tezzele, A. Mola, and G. Rozza. “Hull shape design optimization with parameter space and model reductions, and self-learning mesh morphing”. In: *Journal of Marine Science and Engineering* 9.2 (2021), p. 185 (cit. on p. 97).
- [80] N. Demo, M. Tezzele, and G. Rozza. *EZyRB: Easy Reduced Basis method*. Version 1.1.1. URL: <https://github.com/mathLab/EZyRB> (cit. on p. 5).
- [81] N. Demo, M. Tezzele, and G. Rozza. “EZyRB: Easy reduced basis method”. In: *Journal of Open Source Software* 3.24 (2018), p. 661 (cit. on p. 5).
- [82] S. Deparis, M. A. Fernández, and L. Formaggia. “Acceleration of a fixed point algorithm for fluid-structure interaction using transpiration conditions”. In: *ESAIM: Modélisation mathématique et analyse numérique* 37.4 (2003), pp. 601–616 (cit. on p. 20).
- [83] P. developers. *PINA: A Unified Framework for Scientific Machine Learning*. Version 0.2. URL: <https://github.com/mathLab/PINA> (cit. on p. 197).

-
- [84] S. developers. *SMITHERS: a Mathematical Interdisciplinary Toolbox for Helping Engineers Researchers and Scientist*. Version 0.0.1. URL: <https://github.com/mathLab/Smither> (cit. on p. 5).
 - [85] A. Dunca and Y. Epshteyn. “On the Stolz–Adams deconvolution model for the large-eddy simulation of turbulent flows”. In: *SIAM Journal on Mathematical Analysis* 37.6 (2006), pp. 1890–1902 (cit. on p. 20).
 - [86] V. J. Ervin, W. J. Layton, and M. Neda. “Numerical analysis of filter-based stabilization for evolution equations”. In: *SIAM Journal on Numerical Analysis* 50.5 (2012), pp. 2307–2335 (cit. on pp. 12, 15, 23, 45).
 - [87] D. Fan, L. Yang, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis. “Reinforcement learning for bluff body active flow control in experiments and simulations”. In: *Proceedings of the National Academy of Sciences* 117.42 (2020), pp. 26091–26098 (cit. on p. 43).
 - [88] N. Farenga, S. Fresca, S. Brivio, and A. Manzoni. “On latent dynamics learning in nonlinear reduced order modeling”. In: *Neural Networks* 185 (2025), p. 107146 (cit. on p. 194).
 - [89] N. Farenga, S. Fresca, and A. Manzoni. “Neural latent dynamics models”. In: *The Symbiosis of Deep Learning and Differential Equations II*. 2022 (cit. on p. 194).
 - [90] A. Ferrero, T. Taddei, and L. Zhang. “Registration-based model reduction of parameterized two-dimensional conservation laws”. In: *Journal of Computational Physics* 457 (2022), p. 111068 (cit. on p. 191).
 - [91] J. H. Ferziger, M. Perić, and R. L. Street. *Computational methods for fluid dynamics*. Vol. 3. Springer, 2002 (cit. on p. 1).
 - [92] P. Fischer, S. Kerkemeier, M. Min, Y.-H. Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, et al. “NekRS, a GPU-accelerated spectral element Navier–Stokes solver”. In: *Parallel Comput.* 114 (2022), p. 102982 (cit. on pp. 5, 134, 137, 141).
 - [93] P. Fischer, J. Kruse, J. Mullen, H. Tufo, J. Lottes, and S. Kerkemeier. “Nek5000—open source spectral element CFD solver”. In: *Argonne National Laboratory, Mathematics and Computer Science Division, Argonne, IL* (2008). URL: <https://nek5000.mcs.anl.gov/index.php/MainPage> (cit. on pp. 5, 131, 141).
 - [94] P. Fischer, M. Schmitt, and A. Tomboulides. “Recent developments in spectral element simulations of moving-domain problems”. In: *Recent progress and modern challenges in applied mathematics, modeling and computational science* (2017), pp. 213–244 (cit. on p. 121).
 - [95] B. Font, F. Alcántara-Ávila, J. Rabault, R. Vinuesa, and O. Lehmkuhl. “Deep reinforcement learning for active flow control in a turbulent separation bubble”. In: *Nature communications* 16.1 (2025), p. 1422 (cit. on p. 43).

- [96] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*. Vol. 1. Springer Science & Business Media, 2010 (cit. on p. 97).
- [97] L. P. Franca and T. J. R. Hughes. “Two classes of mixed finite element methods”. In: *Computer Methods in Applied Mechanics and Engineering* 69.1 (1988), pp. 89–129. doi: [10.1016/0045-7825\(88\)90168-5](https://doi.org/10.1016/0045-7825(88)90168-5) (cit. on p. 106).
- [98] N. Franco, A. Manzoni, and P. Zunino. “A deep learning approach to reduced order modelling of parameter dependent partial differential equations”. In: *Mathematics of Computation* 92.340 (2023), pp. 483–524 (cit. on p. 193).
- [99] C. Franzke and A. J. Majda. “Low-order stochastic mode reduction for a prototype atmospheric GCM”. In: *Journal of the atmospheric sciences* 63.2 (2006), pp. 457–479 (cit. on p. 97).
- [100] S. Fresca, L. Dede’, and A. Manzoni. “A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs”. In: *Journal of Scientific Computing* 87.2 (2021), p. 61 (cit. on p. 193).
- [101] S. Fresca, A. Manzoni, L. Dedè, and A. Quarteroni. “Deep learning-based reduced order models in cardiac electrophysiology”. In: *PloS one* 15.10 (2020), e0239416 (cit. on pp. 97, 193).
- [102] H. Frezat, J. L. Sommer, R. Fablet, G. Balarac, and R. Lguensat. “A posteriori learning of quasi-geostrophic turbulence parametrization: an experiment on integration steps”. In: *arXiv preprint arXiv:2111.06841* (2021) (cit. on p. 72).
- [103] C. Gahr, I.-G. Farcaş, and F. Jenko. “Scientific machine learning based reduced-order models for plasma turbulence simulations”. In: *Phys. Plasmas* 31.11 (2024) (cit. on p. 125).
- [104] B. G. Galerkin. “Rods and plates. Series occurring in various questions concerning the elastic equilibrium of rods and plates”. In: *Engineers Bulletin (Vestnik Inzhenerov)* 19 (1915), pp. 897–908 (cit. on p. 103).
- [105] R. Geelen, L. Balzano, S. Wright, and K. Willcox. “Learning physics-based reduced-order models from data using nonlinear manifolds”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 34.3 (2024) (cit. on p. 191).
- [106] R. Geelen, S. Wright, and K. Willcox. “Operator inference for non-intrusive model reduction with quadratic manifolds”. In: *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115717. ISSN: 0045-7825. DOI: <https://doi.org/10.1016/j.cma.2022.115717> (cit. on pp. 170, 174, 187, 191).
- [107] M. Germano. “Differential filters for the large eddy numerical simulation of turbulent flows”. In: *The Physics of fluids* 29.6 (1986), pp. 1755–1757 (cit. on pp. 2, 11, 141).

- [108] M. Germano. “Differential filters of elliptic type”. In: *The Physics of fluids* 29.6 (1986), pp. 1757–1758 (cit. on pp. 2, 11, 141).
- [109] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot. “A dynamic subgrid-scale eddy viscosity model”. In: *Physics of fluids a: Fluid dynamics* 3.7 (1991), pp. 1760–1765 (cit. on p. 20).
- [110] A.-L. Gerner and K. Veroy. “Certified reduced basis methods for parametrized saddle point problems”. In: *SIAM Journal on Scientific Computing* 34.5 (2012), A2812–A2836 (cit. on p. 105).
- [111] S. Ghosal, T. S. Lund, P. Moin, and K. Akselvoll. “A dynamic localization model for large-eddy simulation of turbulent flows”. In: *Journal of fluid mechanics* 286 (1995), pp. 229–255 (cit. on p. 21).
- [112] M. Girfoglio, A. Quaini, and G. Rozza. “A hybrid projection/data-driven reduced order model for the Navier-Stokes equations with nonlinear filtering stabilization”. In: *Journal of Computational Physics* 486 (2023), p. 112127 (cit. on p. 110).
- [113] M. Girfoglio, A. Quaini, and G. Rozza. “A linear filter regularization for POD-based reduced-order models of the quasi-geostrophic equations”. In: *Comptes Rendus. Mécanique* 351.S1 (2023), pp. 457–477 (cit. on p. 110).
- [114] M. Girfoglio, A. Quaini, and G. Rozza. “Pressure stabilization strategies for a LES filtering reduced order model”. In: *Fluids* 6.9 (2021), p. 302 (cit. on p. 20).
- [115] H. Gong, S. Cheng, Z. Chen, and Q. Li. “Data-enabled physics-informed machine learning for reduced-order modeling digital twin: application to nuclear reactor physics”. In: *Nuclear Science and Engineering* 196.6 (2022), pp. 668–693 (cit. on p. 97).
- [116] W. Graham, J. Peraire, and K. Tang. “Optimal control of vortex shedding using low-order models. Part II—model-based control”. In: *International journal for numerical methods in engineering* 44.7 (1999), pp. 973–990 (cit. on p. 103).
- [117] W. Graham, J. Peraire, and K. Tang. “Optimal control of vortex shedding using low-order models. Part I—open-loop model development”. In: *International Journal for Numerical Methods in Engineering* 44.7 (1999), pp. 945–972 (cit. on p. 103).
- [118] Y. Guan, A. Chattopadhyay, A. Subel, and P. Hassanzadeh. “Stable a posteriori LES of 2D turbulence using convolutional neural networks: Backscattering analysis and generalization to higher Re via transfer learning”. In: *Journal of Computational Physics* 458 (2022), p. 111090. ISSN: 0021-9991. DOI: [10.1016/j.jcp.2022.111090](https://doi.org/10.1016/j.jcp.2022.111090) (cit. on p. 21).
- [119] M. Gunzburger, T. Iliescu, M. Mohebbujaman, and M. Schneier. “An Evolve-Filter-Relax Stabilized Reduced Order Stochastic Collocation Method for the Time-Dependent Navier-Stokes Equations”. In: *SIAM/ASA Journal on Uncertainty Quantification* 7.4 (2019), pp. 1162–1184 (cit. on pp. 109, 110).

- [120] M. D. Gunzburger, J. S. Peterson, and J. N. Shadid. “Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data”. In: *Computer methods in applied mechanics and engineering* 196.4-6 (2007), pp. 1030–1047 (cit. on p. 103).
- [121] M. Guo and J. S. Hesthaven. “Reduced order modeling for nonlinear structural analysis using Gaussian process regression”. In: *Computer methods in applied mechanics and engineering* 341 (2018), pp. 807–826 (cit. on p. 193).
- [122] X. Han, H. Gao, T. Pfaff, J.-X. Wang, and L.-P. Liu. “Predicting physics in mesh-reduced space with temporal attention”. In: *arXiv preprint arXiv:2201.09113* (2022) (cit. on p. 193).
- [123] F. H. Harlow, J. E. Welch, et al. “Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface”. In: *Physics of fluids* 8.12 (1965), p. 2182 (cit. on p. 17).
- [124] D. Hartmann, M. Herz, and U. Wever. “Model order reduction a key technology for digital twins”. In: *Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing*. Springer, 2018, pp. 167–179 (cit. on p. 97).
- [125] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on p. 195).
- [126] N. Heavner. “Locally chosen grad-div stabilization parameters for finite element discretizations of incompressible flow problems”. In: *SIAM Undergraduate Research Online (SIURO)* 7 (2017), SO1278 (cit. on p. 39).
- [127] J. Hesthaven and S. Ubbiali. “Non-intrusive reduced order modeling of nonlinear problems using neural networks”. In: *Journal of Computational Physics* 363 (2018-02). doi: 10.1016/j.jcp.2018.02.037 (cit. on pp. 173, 193).
- [128] J. S. Hesthaven, G. Rozza, B. Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*. Vol. 590. Springer, 2016 (cit. on pp. 97, 99, 103).
- [129] S. Hijazi. “Reduced order methods for laminar and turbulent flows in a finite volume setting: projection-based methods and data-driven techniques”. PhD thesis. International School for Advanced Studies (SISSA), 2020 (cit. on pp. 113, 114).
- [130] S. Hijazi, M. Freitag, and N. Landwehr. “POD-Galerkin reduced order models and physics-informed neural networks for solving inverse problems for the Navier–Stokes equations”. In: *Advanced Modeling and Simulation in Engineering Sciences* 10.1 (2023), p. 5 (cit. on p. 203).
- [131] S. Hijazi, G. Stabile, A. Mola, and G. Rozza. “Data-driven POD-Galerkin reduced order model for turbulent flows”. In: *Journal of Computational Physics* 416 (2020), p. 109513 (cit. on pp. 111, 113, 114, 148, 154, 168).

- [132] G. E. Hinton and R. R. Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *science* 313.5786 (2006), pp. 504–507 (cit. on pp. 5, 191, 193).
- [133] T. J. R. Hughes, L. P. Franca, and M. Balestra. “A new finite element formulation for computational fluid dynamics: V. A stable Petrov–Galerkin formulation of the Stokes problem accommodating equal-order interpolations”. In: *Computer Methods in Applied Mechanics and Engineering* 59.1 (1986), pp. 85–99. doi: 10.1016/0045-7825(86)90025-3 (cit. on p. 106).
- [134] T. J. Hughes. “Multiscale phenomena: Green’s functions, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods”. In: *Computer methods in applied mechanics and engineering* 127.1-4 (1995), pp. 387–401 (cit. on p. 19).
- [135] T. J. Hughes, G. R. Feijóo, L. Mazzei, and J.-B. Quincy. “The variational multiscale method—a paradigm for computational mechanics”. In: *Computer methods in applied mechanics and engineering* 166.1-2 (1998), pp. 3–24 (cit. on p. 19).
- [136] T. J. Hughes, L. Mazzei, and K. E. Jansen. “Large eddy simulation and the variational multiscale method”. In: *Computing and visualization in science* 3.1 (2000), pp. 47–59 (cit. on p. 19).
- [137] T. Iliescu and Z. Wang. “Variational multiscale proper orthogonal decomposition: Navier-stokes equations”. In: *Numerical Methods for Partial Differential Equations* 30.2 (2014), pp. 641–663 (cit. on p. 111).
- [138] ITHACA-FV (*In real Time Highly Advanced Computational Applications for Finite Volumes*). URL: <https://ithaca-fv.github.io/ITHACA-FV> (cit. on pp. 5, 154).
- [139] K. Ito and S. S. Ravindran. “Reduced basis method for optimal control of unsteady viscous flows”. In: *International Journal of Computational Fluid Dynamics* 15.2 (2001), pp. 97–113 (cit. on p. 97).
- [140] A. Ivagnes, N. Demo, and G. Rozza. “A shape optimization pipeline for marine propellers by means of reduced order modeling techniques”. In: *International Journal for Numerical Methods in Engineering* 125.7 (2024), e7426 (cit. on p. 97).
- [141] A. Ivagnes, M. Khamlich, P. Siena, and G. Rozza. “in Computational Fluid Dynamics: An Overview of Methods”. In: *Emerging Technologies in Computational Sciences for Industry, Sustainability and Innovation: Math to Product* (2025), p. 1 (cit. on p. 97).
- [142] A. Ivagnes, G. Stabile, A. Mola, T. Iliescu, and G. Rozza. “Hybrid data-driven closure strategies for reduced order modeling”. In: *Applied Mathematics and Computation* 448 (2023), p. 127920 (cit. on pp. 104, 117–119, 146, 154, 168).
- [143] A. Ivagnes, G. Stabile, A. Mola, T. Iliescu, and G. Rozza. “Pressure data-driven variational multiscale reduced order models”. In: *Journal of Computational Physics* 476 (2023), p. 111904 (cit. on pp. 104, 107, 114, 115, 117, 118, 123, 154).

- [144] A. Ivagnes, G. Stabile, and G. Rozza. “A Physics-Based Nonlinear Model Order Reduction Framework Based on Operator Compression”. In preparation. 2026 (cit. on p. 5).
- [145] A. Ivagnes, G. Stabile, and G. Rozza. “Data-driven Closure Strategies for Parametrized Reduced Order Models via Deep Operator Networks”. In: *Computer Methods in Applied Mechanics and Engineering* 455 (2026), p. 118900 (cit. on pp. 4, 105, 111, 114, 116, 146).
- [146] A. Ivagnes, G. Stabile, and G. Rozza. “Parametric Intrusive Reduced Order Models enhanced with Machine Learning Correction Terms”. In: *arXiv preprint arXiv:2406.04169* (2024) (cit. on pp. 4, 114, 116, 151).
- [147] A. Ivagnes, M. Strazzullo, M. Girfoglio, T. Iliescu, and G. Rozza. “Data-Driven Optimization for the Evolve-Filter-Relax Regularization of Convection-Dominated Flows”. In: *International Journal for Numerical Methods in Engineering* 126.9 (2025). DOI: [10.1002/nme.70042](https://doi.org/10.1002/nme.70042) (cit. on pp. 2, 23, 39, 40, 56, 60).
- [148] A. Ivagnes, M. Strazzullo, and G. Rozza. “Reinforcement Learning-Based Filters for Convection-Dominated Flows: Reference-Free and Reference-Guided Training”. In: *arXiv preprint arXiv:2603.02086* (2026) (cit. on p. 3).
- [149] A. Ivagnes, N. Tonicello, P. Cinnella, and G. Rozza. “Enhancing non-intrusive reduced-order models with space-dependent aggregation methods”. In: *Acta Mechanica* (2024), pp. 1–30 (cit. on pp. 5, 232).
- [150] A. Ivagnes, T. Van Gastelen, S. D. Agdestein, B. Sanderse, G. Stabile, and G. Rozza. “A new data-driven energy-stable evolve-filter-relax model for turbulent flow simulation”. In: *Computer Methods in Applied Mechanics and Engineering* 450 (2026), p. 118654 (cit. on pp. 3, 43, 75, 83, 93).
- [151] S. Jain, P. Tiso, J. B. Rutzmoser, and D. J. Rixen. “A quadratic manifold for model order reduction of nonlinear structural dynamics”. In: *Computers & Structures* 188 (2017), pp. 80–94. ISSN: 0045-7949. DOI: [10.1016/j.compstruc.2017.04.005](https://doi.org/10.1016/j.compstruc.2017.04.005) (cit. on pp. 170, 191).
- [152] A. Janon, M. Nodet, and C. Prieur. “Certified reduced-basis solutions of viscous Burgers equation parametrized by initial and boundary values”. In: *ESAIM: Mathematical Modelling and Numerical Analysis* 47.2 (2013), pp. 317–348 (cit. on p. 104).
- [153] H. Jasak. “Error analysis and estimation for the finite volume method with applications to fluid flows.” In: (1996) (cit. on pp. 5, 154, 179, 213).
- [154] J. Jiménez and P. Moin. “The minimal flow unit in near-wall turbulence”. In: *J. Fluid Mech.* 225 (1991), pp. 213–240 (cit. on p. 137).
- [155] P. Jin, S. Meng, and L. Lu. “MIONet: Learning multiple-input operators via tensor product”. In: *SIAM Journal on Scientific Computing* 44.6 (2022), A3490–A3514 (cit. on pp. 146, 150, 171).

-
- [156] V. John, A. Linke, C. Merdon, M. Neilan, and L. G. Rebholz. “On the divergence constraint in mixed finite element methods for incompressible flows”. In: *SIAM review* 59.3 (2017), pp. 492–544 (cit. on p. 38).
 - [157] H. Johnston and J.-G. Liu. “Accurate, stable and efficient Navier–Stokes solvers based on explicit treatment of the pressure term”. In: *Journal of Computational Physics* 199.1 (2004), pp. 221–259 (cit. on p. 105).
 - [158] L. P. Kaelbling, M. L. Littman, and A. W. Moore. “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285 (cit. on p. 43).
 - [159] I. Kalashnikova and M. F. Barone. “Efficient non-linear proper orthogonal decomposition/Galerkin reduced order models with stable penalty enforcement of boundary conditions”. In: *International Journal for Numerical Methods in Engineering* 90.11 (2012), pp. 1337–1362 (cit. on p. 104).
 - [160] I. Kalashnikova and M. F. Barone. “On the stability and convergence of a Galerkin reduced order model (ROM) of compressible flow with solid wall and far-field boundary treatment”. In: *International journal for numerical methods in engineering* 83.10 (2010), pp. 1345–1375 (cit. on p. 104).
 - [161] K. Kaneko, P.-H. Tsai, N. Christensen, and P. Fischer. *NekROM*. <https://github.com/Nek5000/NekROM>. Version v0.2.0. 2026. DOI: [10.5281/zenodo.18487297](https://doi.org/10.5281/zenodo.18487297) (cit. on pp. 5, 141).
 - [162] K. Kaneko, P.-H. Tsai, and P. Fischer. “Towards model order reduction for fluid-thermal analysis”. In: *Nuclear Engineering and Design* 370 (2020), p. 110866 (cit. on p. 130).
 - [163] J. N. Kani and A. H. Elsheikh. “DR-RNN: A deep residual recurrent neural network for model reduction”. In: *arXiv preprint arXiv:1709.00939* (2017) (cit. on p. 194).
 - [164] M. G. Kapteyn, D. J. Knezevic, D. Huynh, M. Tran, and K. E. Willcox. “Data-driven physics-based digital twins via a library of component-based reduced-order models”. In: *International Journal for Numerical Methods in Engineering* 123.13 (2022), pp. 2986–3003 (cit. on p. 97).
 - [165] M. G. Kapteyn, D. J. Knezevic, and K. Willcox. “Toward predictive digital twins via component-based reduced-order models and interpretable machine learning”. In: *AIAA scitech 2020 forum*. 2020, p. 0418 (cit. on p. 97).
 - [166] M. Khamlich, F. Pichi, and G. Rozza. “Optimal Transport-Inspired Deep Learning Framework for Slow-Decaying Kolmogorov-Width Problems: Exploiting Sinkhorn Loss and Wasserstein Kernel”. In: *SIAM Journal on Scientific Computing* 47.2 (2025), pp. C235–C264 (cit. on p. 191).
 - [167] M. Khamlich, G. Stabile, G. Rozza, L. Környei, and Z. Horváth. “A physics-based reduced order model for urban air pollution prediction”. In: *Computer Methods in Applied Mechanics and Engineering* 417 (2023), p. 116416 (cit. on p. 97).

- [168] H. Kim, V. Shankar, V. Viswanathan, and R. Maulik. “Generalizable data-driven turbulence closure modeling on unstructured grids with differentiable physics”. In: *Computers & Fluids* (2026), p. 107200 (cit. on p. 140).
- [169] Y. Kim, Y. Choi, D. Widemann, and T. Zohdi. “A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder”. In: *Journal of Computational Physics* 451 (2022), p. 110841 (cit. on p. 203).
- [170] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 48).
- [171] B. Koc, C. Mou, H. Liu, Z. Wang, G. Rozza, and T. Iliescu. “Verifiability of the data-driven variational multiscale reduced order model”. In: *Journal of Scientific Computing* 93.2 (2022), p. 54 (cit. on p. 117).
- [172] B. Koc, S. Rubino, T. Chacón Rebollo, and T. Iliescu. “Residual-based data-driven variational multiscale reduced order models for parameter-dependent problems”. In: *Computational and Applied Mathematics* 44.6 (2025), p. 308 (cit. on pp. 115, 117).
- [173] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. “Machine learning–accelerated computational fluid dynamics”. In: *Proceedings of the National Academy of Sciences* 118.21 (2021), e2101784118 (cit. on pp. 17, 21, 88).
- [174] A. N. Kolmogorov. “The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers”. In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 434.1890 (1991), pp. 9–13 (cit. on p. 9).
- [175] A. N. Kolmogorov. *Turbulence: the legacy of AN Kolmogorov*. Cambridge university press, 1995 (cit. on p. 10).
- [176] A. N. Kolmogorov. “A refinement of previous hypotheses concerning the local structure of turbulence in a viscous incompressible fluid at high Reynolds number”. In: *Journal of Fluid Mechanics* 13.1 (1962), pp. 82–85 (cit. on p. 9).
- [177] R. H. Kraichnan. *Inertial ranges in two-dimensional turbulence*. Tech. rep. 1967 (cit. on p. 10).
- [178] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012) (cit. on p. 195).
- [179] K. Kunisch and S. Volkwein. “Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics”. In: *SIAM Journal on Numerical analysis* 40.2 (2002), pp. 492–515 (cit. on pp. 98, 100).
- [180] M. Kurz and A. Beck. “A machine leaning framework for LES closure terms”. In: *Electronic Transactions on Numerical Analysis* 56 (2022), pp. 117–137 (cit. on p. 21).
- [181] M. Kurz, A. Beck, and B. Sanderse. “Harnessing equivariance: Modeling turbulence with graph neural networks”. In: *arXiv preprint arXiv:2504.07741* (2025) (cit. on pp. 21, 43, 52).

- [182] M. Kurz, P. Offenhäuser, and A. Beck. “Deep reinforcement learning for turbulence modeling in large eddy simulations”. In: *International journal of heat and fluid flow* 99 (2023), p. 109094 (cit. on pp. 17, 21, 43, 52).
- [183] M. Kurz, P. Offenhäuser, D. Viola, O. Shcherbakov, M. Resch, and A. Beck. “Deep reinforcement learning for computational fluid dynamics on HPC systems”. In: *Journal of Computational Science* 65 (2022), p. 101884 (cit. on pp. 21, 43, 52).
- [184] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016 (cit. on pp. 77, 194).
- [185] A. Kwiatkowski, M. Towers, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, K. Andreas, M. Krimmel, A. KG, et al. “Gymnasium: a standard interface for reinforcement learning environments”. In: (2024) (cit. on p. 55).
- [186] T. Lassila, A. Manzoni, A. Quarteroni, and G. Rozza. “Model order reduction in fluid dynamics: challenges and perspectives”. In: *Reduced Order Methods for modeling and computational reduction* (2014), pp. 235–273 (cit. on p. 97).
- [187] W. Layton, C. C. Manica, M. Neda, M. Olshanskii, and L. G. Rebholz. “On the accuracy of the rotation form in simulations of the Navier–Stokes equations”. In: *Journal of Computational Physics* 228.9 (2009), pp. 3433–3447 (cit. on p. 38).
- [188] W. Layton, C. C. Manica, M. Neda, and L. G. Rebholz. “Numerical analysis and computational testing of a high accuracy Leray-deconvolution model of turbulence”. In: *Numerical Methods for Partial Differential Equations: An International Journal* 24.2 (2008), pp. 555–582 (cit. on pp. 12, 38, 45).
- [189] W. Layton, L. G. Rebholz, and C. Trenchea. “Modular nonlinear filter stabilization of methods for higher Reynolds numbers flow”. In: *Journal of Mathematical Fluid Mechanics* 14.2 (2012), pp. 325–354 (cit. on pp. 12, 13, 15, 20).
- [190] W. J. Layton and L. G. Rebholz. *Approximate deconvolution models of turbulence: analysis, phenomenology and numerical analysis*. Vol. 2042. Springer Science & Business Media, 2012 (cit. on pp. 2, 11).
- [191] D. Lazzaro and L. B. Montefusco. “Radial basis functions for the multivariate interpolation of large scattered data sets”. In: *Journal of Computational and Applied Mathematics* 140.1-2 (2002), pp. 521–536 (cit. on p. 114).
- [192] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 195).
- [193] K. Lee and K. T. Carlberg. “Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders”. In: *Journal of Computational Physics* 404 (2020), p. 108973 (cit. on pp. 5, 193, 201).

- [194] K. Lee and E. J. Parish. “Parameterized neural ordinary differential equations: Applications to computational physics problems”. In: *Proceedings of the Royal Society A* 477.2253 (2021), p. 20210162 (cit. on p. 194).
- [195] M. Lehtimäki, L. Paunonen, and M.-L. Linne. “Accelerating neural odes using model order reduction”. In: *IEEE Transactions on Neural Networks and Learning Systems* 35.1 (2022), pp. 519–531 (cit. on p. 194).
- [196] A. Leonard. “Energy cascade in large-eddy simulations of turbulent fluid flows”. In: *Advances in geophysics*. Vol. 18. Elsevier, 1975, pp. 237–248 (cit. on p. 13).
- [197] J. Leray. “Essai sur les mouvements plans d’un liquide visqueux que limitent des parois”. In: *Journal de Mathématiques pures et appliquées* 13 (1934), pp. 331–418 (cit. on p. 13).
- [198] J. Leray. “Sur le mouvement d’un liquide visqueux emplissant l’espace”. In: *Acta mathematica* 63.1 (1934), pp. 193–248 (cit. on pp. 4, 13).
- [199] G. Li Causi, N. Tonicello, L. Magri, and G. Rozza. “Convolutional Symmetric AutoEncoders: enhancing latent stability via differential geometry”. In: *arXiv preprint arXiv:2607.00669* (2026) (cit. on p. 193).
- [200] D. Lilly. “A proposed modification of the Germano subgrid-scale eddy viscosity model”. In: *Phys. Fluids A* 4.3 (1992), pp. 633–635 (cit. on p. 20).
- [201] A. J. Linot and M. D. Graham. “Data-driven reduced-order modeling of spatiotemporal chaos with neural ordinary differential equations”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 32.7 (2022) (cit. on p. 194).
- [202] B. List, L.-W. Chen, K. Bali, and N. Thuerey. “Differentiability in unrolled training of neural physics simulators on transient dynamics”. In: *Computer Methods in Applied Mechanics and Engineering* 433 (2025), p. 117441 (cit. on p. 88).
- [203] B. List, L.-W. Chen, and N. Thuerey. “Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons”. In: *Journal of Fluid Mechanics* 949 (2022), A25 (cit. on pp. 21, 72).
- [204] D. C. Liu and J. Nocedal. “On the limited memory BFGS method for large scale optimization”. In: *Mathematical programming* 45.1 (1989), pp. 503–528 (cit. on p. 48).
- [205] A. Logg, K.-A. Mardal, and G. Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*. Vol. 84. Springer Science & Business Media, 2012 (cit. on pp. 5, 24).
- [206] A. Logg and G. N. Wells. “DOLFIN: Automated finite element computing”. In: *ACM Transactions on Mathematical Software (TOMS)* 37.2 (2010), pp. 1–28 (cit. on pp. 5, 24, 206).
- [207] A. Logg, G. N. Wells, and J. Hake. “DOLFIN: a C++/Python finite element library”. In: *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Springer, 2012, pp. 173–225 (cit. on pp. 5, 24, 178).

- [208] S. Lorenzi, A. Cammi, L. Luzzi, and G. Rozza. “POD-Galerkin method for finite volume approximation of Navier–Stokes and RANS equations”. In: *Computer Methods in Applied Mechanics and Engineering* 311 (2016), pp. 151–179 (cit. on pp. 98, 104).
- [209] I. Loshchilov and F. Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017) (cit. on p. 48).
- [210] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. “Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators”. In: *Nature machine intelligence* 3.3 (2021), pp. 218–229 (cit. on pp. 146, 149–151, 171, 193).
- [211] G. Lube, G. Rapin, and J. Løwe. “Local projection stabilization for incompressible flows: equal-order vs. inf-sup stable interpolation”. In: *Electronic Transactions on Numerical Analysis* 25 (2006), pp. 106–122 (cit. on p. 106).
- [212] J. L. Lumley. “The structure of inhomogeneous turbulent flows”. In: *Atmospheric Turbulence and Radio Wave Propagation*. Ed. by A. M. Yaglom and V. I. Tatarski. Nauka, 1967, pp. 166–178 (cit. on pp. 3, 100).
- [213] J. F. MacArt, J. Sirignano, and J. B. Freund. “Embedded training of neural-network subgrid-scale turbulence models”. In: *Phys. Rev. Fluids* 6 (5 2021), p. 050502. DOI: [10.1103/PhysRevFluids.6.050502](https://doi.org/10.1103/PhysRevFluids.6.050502) (cit. on p. 72).
- [214] Y. Maday and E. M. Rønquist. “A reduced-basis element method”. In: *Journal of scientific computing* 17.1 (2002), pp. 447–459 (cit. on pp. 97, 191).
- [215] A. J. Majda, J. Harlim, and B. Gershgorin. “Mathematical strategies for filtering turbulent dynamical systems”. In: *Discrete Contin. Dyn. Syst* 27.2 (2010), pp. 441–486 (cit. on p. 111).
- [216] A. Manzoni, A. Quarteroni, and G. Rozza. “Shape optimization for viscous flows by reduced basis methods and free-form deformation”. In: *International Journal for Numerical Methods in Fluids* 70.5 (2012), pp. 646–670 (cit. on p. 97).
- [217] S. Masoumi-Verki, F. Haghighat, and U. Eicker. “A review of advances towards efficient reduced-order models (ROM) for predicting urban airflow and pollutant dispersion”. In: *Building and Environment* 216 (2022), p. 108966 (cit. on p. 97).
- [218] J. Mathieu and J. Scott. *An introduction to turbulent flow*. Cambridge University Press, 2000 (cit. on p. 1).
- [219] R. Maulik, O. San, A. Rasheed, and P. Vedula. “Subgrid modelling for two-dimensional turbulence using neural networks”. In: *Journal of Fluid Mechanics* 858 (2018), pp. 122–144. ISSN: 1469-7645. DOI: [10.1017/jfm.2018.770](https://doi.org/10.1017/jfm.2018.770) (cit. on p. 21).
- [220] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133 (cit. on p. 47).

- [221] L. R. Medsker, L. Jain, et al. "Recurrent neural networks". In: *Design and applications* 5.64-67 (2001), p. 2 (cit. on p. 194).
- [222] P. M. Mehta and R. Linares. "A methodology for reduced order modeling and calibration of the upper atmosphere." In: *Space Weather: The International Journal of Research & Applications* 15.10 (2017) (cit. on p. 97).
- [223] H. Melchers, D. Crommelin, B. Koren, V. Menkovski, and B. Sanderse. "Comparison of neural closure models for discretised PDEs". In: *Computers & Mathematics with Applications* 143 (2023), pp. 94–107 (cit. on p. 72).
- [224] C. Meneveau, T. S. Lund, and W. H. Cabot. "A Lagrangian dynamic subgrid-scale model of turbulence". In: *Journal of fluid mechanics* 319 (1996), pp. 353–385 (cit. on p. 21).
- [225] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533 (cit. on p. 49).
- [226] M. Mohebujjaman, L. G. Rebholz, and T. Iliescu. "Physically constrained data-driven correction for reduced-order modeling of fluid flows". In: *International Journal for Numerical Methods in Fluids* 89.3 (2019), pp. 103–122 (cit. on pp. 116, 118, 123, 168).
- [227] C. Mou, E. Merzari, O. San, and T. Iliescu. "An Energy-Based Lengthscale for Reduced Order Models of Turbulent Flows". In: *Nuclear Engineering and Design* 412 (2023), p. 112454 (cit. on p. 110).
- [228] C. Mou, B. Koc, O. San, L. G. Rebholz, and T. Iliescu. "Data-driven variational multiscale reduced order models". In: *Computer Methods in Applied Mechanics and Engineering* 373 (2021), p. 113470 (cit. on p. 168).
- [229] C. Mou, Z. Wang, D. R. Wells, X. Xie, and T. Iliescu. "Reduced order models for the quasi-geostrophic equations: A brief survey". In: *Fluids* 6.1 (2020), p. 16 (cit. on p. 97).
- [230] F. Moukalled, L. Mangani, M. Darwish, et al. *The finite volume method in computational fluid dynamics*. Vol. 113. Springer, 2016 (cit. on p. 179).
- [231] N. T. Mücke, S. M. Bohté, and C. W. Oosterlee. "Reduced order modeling for parameterized time-dependent PDEs using spatially and memory aware deep learning". In: *Journal of Computational Science* 53 (2021), p. 101408 (cit. on p. 194).
- [232] J. S. Mullen and P. F. Fischer. "Filtering techniques for complex geometry fluid flows". In: *Communications in numerical methods in engineering* 15.1 (1999), pp. 9–18 (cit. on p. 109).
- [233] M. Navier. "Sur les Lois du Mouvement des fluides, Mémoires de l'Académie Royale des Sciences". In: *T. VI. P. 389 (année 1827)* (1823), p. 389 (cit. on pp. 1, 8).

- [234] B. R. Noack, K. Afanasiev, M. Morzyński, G. Tadmor, and F. Thiele. “A hierarchy of low-dimensional models for the transient and post-transient cylinder wake”. In: *Journal of Fluid Mechanics* 497 (2003), pp. 335–363 (cit. on pp. 4, 97).
- [235] G. Novati, H. L. De Laroussilhe, and P. Koumoutsakos. “Automating turbulence modelling by multi-agent reinforcement learning”. In: *Nature Machine Intelligence* 3.1 (2021), pp. 87–96 (cit. on p. 21).
- [236] M. Ohlberger and S. Rave. “Nonlinear reduced basis approximation of parameterized evolution equations via the method of freezing”. In: *Comptes Rendus Mathématique* 351.23-24 (2013), pp. 901–906 (cit. on p. 191).
- [237] L. Onsager. “Statistical hydrodynamics”. In: *Il Nuovo Cimento (1943-1954)* 6.Suppl 2 (1949), pp. 279–287 (cit. on p. 10).
- [238] C. Onyutha. “A hydrological model skill score and revised R-squared”. In: *Hydrol. Res.* 53.1 (2022), pp. 51–64 (cit. on p. 124).
- [239] C. Onyutha. “From R-squared to coefficient of model accuracy for assessing “goodness-of-fits””. In: *Geosci. Model Dev. Discuss.* 2020 (2020), pp. 1–25 (cit. on p. 124).
- [240] *OpenFOAM - Free CFD Software*. URL: <https://openfoam.org> (cit. on pp. 5, 154, 213).
- [241] P. Orlandi. *Fluid flow phenomena: a numerical toolkit*. Vol. 55. Springer Science & Business Media, 2000 (cit. on p. 17).
- [242] G. Ortali, N. Demo, G. Rozza, et al. “A Gaussian process regression approach within a data-driven POD framework for engineering problems in fluid dynamics”. In: *Mathematics in Engineering* 4.3 (2022), pp. 1–16 (cit. on p. 173).
- [243] S. E. Otto, G. R. Macchio, and C. W. Rowley. “Learning nonlinear projections for reduced-order modeling of dynamical systems using constrained autoencoders”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 33.11 (2023) (cit. on p. 193).
- [244] S. E. Otto and C. W. Rowley. “Linearly recurrent autoencoder networks for learning dynamics”. In: *SIAM Journal on Applied Dynamical Systems* 18.1 (2019), pp. 558–593 (cit. on p. 194).
- [245] A. Palha and M. Gerritsma. “A mass, energy, enstrophy and vorticity conserving (MEEVC) mimetic spectral element discretization for the 2D incompressible Navier–Stokes equations”. In: *Journal of Computational Physics* 328 (2017), pp. 200–220. ISSN: 0021-9991. DOI: [10.1016/j.jcp.2016.10.009](https://doi.org/10.1016/j.jcp.2016.10.009) (cit. on p. 81).
- [246] D. Papapicco, N. Demo, M. Girfoglio, G. Stabile, and G. Rozza. “The neural network shifted-proper orthogonal decomposition: a machine learning approach for non-linear reduction of hyperbolic equations”. In: *Computer Methods in Applied Mechanics and Engineering* 392 (2022), p. 114687 (cit. on p. 191).

- [247] R. Paris, S. Beneddine, and J. Dandois. “Robust flow control and optimal sensor placement using deep reinforcement learning”. In: *Journal of Fluid Mechanics* 913 (2021), A25 (cit. on p. 43).
- [248] J. Park and H. Choi. “Toward neural-network-based large eddy simulation: application to turbulent channel flow”. In: *Journal of Fluid Mechanics* 914 (2021), A16. doi: [10.1017/jfm.2020.931](https://doi.org/10.1017/jfm.2020.931) (cit. on p. 21).
- [249] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019) (cit. on pp. 141, 154).
- [250] J. Pedlosky. *Geophysical fluid dynamics*. Springer Science & Business Media, 2013 (cit. on p. 10).
- [251] B. Peherstorfer and K. Willcox. “Data-driven operator inference for nonintrusive projection-based model reduction”. In: *Computer Methods in Applied Mechanics and Engineering* 306 (2016), pp. 196–215 (cit. on pp. 116, 170, 174, 175, 187).
- [252] G. I. Petrov. “Application of the method of Galerkin to a problem involving the stationary flow of a viscous fluid”. In: *Prikl. Matem. Mekh* 4.3 (1940), pp. 3–12 (cit. on p. 103).
- [253] Philip, E and Elizabeth, WONG. “Sequential quadratic programming methods”. In: *UCSD Department of Mathematics Technical Report NA-10-03 August* (2010) (cit. on p. 29).
- [254] F. Pichi, B. Moya, and J. S. Hesthaven. “A graph convolutional autoencoder approach to model order reduction for parametrized PDEs”. In: *Journal of Computational Physics* 501 (2024), p. 112762 (cit. on p. 193).
- [255] A. Pinkus. *N-widths in Approximation Theory*. Springer Science & Business Media, 2012 (cit. on p. 191).
- [256] S. B. Pope. “Turbulent flows”. In: *Measurement Science and Technology* 12.11 (2001), pp. 2020–2021 (cit. on p. 1).
- [257] B. Protas, B. R. Noack, and J. Østh. “Optimal nonlinear eddy viscosity in Galerkin models of turbulent flows”. In: *Journal of Fluid Mechanics* 766 (2015), pp. 337–367 (cit. on pp. 111, 112).
- [258] C. Prud’Homme, D. V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. T. Patera, and G. Turinici. “Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods”. In: *J. Fluids Eng.* 124.1 (2002), pp. 70–80 (cit. on pp. 97, 103).

-
- [259] A. Quaini, O. San, A. Veneziani, and T. Iliescu. “Bridging large eddy simulation and reduced-order modeling of convection-dominated flows through spatial filtering: Review and perspectives”. In: *Fluids* 9.8 (2024), p. 178 (cit. on pp. 20, 21, 108, 110).
- [260] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: an introduction*. Springer, 2015 (cit. on p. 97).
- [261] A. Quarteroni and G. Rozza. “Numerical solution of parametrized Navier–Stokes equations by reduced basis methods”. In: *Numerical methods for partial differential equations: an international journal* 23.4 (2007), pp. 923–948 (cit. on p. 97).
- [262] A. Quarteroni, G. Rozza, et al. *Reduced order methods for modeling and computational reduction*. Vol. 9. Springer, 2014 (cit. on p. 97).
- [263] A. Quarteroni, G. Rozza, and A. Manzoni. “Certified reduced basis approximation for parametrized partial differential equations and applications”. In: *Journal of Mathematics in Industry* 1.1 (2011), p. 3 (cit. on pp. 97, 99).
- [264] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi. “Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control”. In: *Journal of fluid mechanics* 865 (2019), pp. 281–302 (cit. on p. 43).
- [265] J. Rabault, F. Ren, W. Zhang, H. Tang, and H. Xu. “Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization”. In: *Journal of Hydrodynamics* 32.2 (2020), pp. 234–246 (cit. on p. 43).
- [266] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. “Stable-baselines3: Reliable reinforcement learning implementations”. In: *Journal of machine learning research* 22.268 (2021), pp. 1–8 (cit. on p. 55).
- [267] M. Raissi, P. Perdikaris, and G. E. Karniadakis. “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations”. In: *Journal of Computational physics* 378 (2019), pp. 686–707 (cit. on p. 203).
- [268] S. Ravindran. “A reduced-order approach for optimal control of fluids using proper orthogonal decomposition”. In: *International journal for numerical methods in fluids* 34.5 (2000), pp. 425–448 (cit. on p. 97).
- [269] J. Reddy. “On penalty function methods in the finite-element analysis of flow problems”. In: *International Journal for Numerical Methods in Fluids* 2.2 (1982), pp. 151–171 (cit. on p. 103).
- [270] J. Reiss, P. Schulze, J. Sesterhenn, and V. Mehrmann. “The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena”. In: *SIAM Journal on Scientific Computing* 40.3 (2018), A1322–A1344 (cit. on p. 191).

- [271] F. Ren, J. Rabault, and H. Tang. “Applying deep reinforcement learning to active flow control in weakly turbulent conditions”. In: *Physics of Fluids* 33.3 (2021) (cit. on p. 43).
- [272] O. Reynolds. “An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels”. In: *Philosophical Transactions of the Royal society of London* 174 (1883), pp. 935–982 (cit. on pp. 1, 112).
- [273] L. F. Richardson. *Weather prediction by numerical process*. Franklin Classics, 1922 (cit. on p. 8).
- [274] C. J. Rojas, A. Dengel, and M. D. Ribeiro. “Reduced-order model for fluid flows via neural ordinary differential equations”. In: *arXiv preprint arXiv:2102.02248* (2021) (cit. on p. 194).
- [275] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 (cit. on p. 195).
- [276] H. Rosenberger and B. Sanderse. “No pressure? Energy-consistent ROMs for the incompressible Navier-Stokes equations with time-dependent boundary conditions”. In: *Journal of Computational Physics* 491 (2023), p. 112405 (cit. on pp. 75, 94).
- [277] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386 (cit. on p. 47).
- [278] W. Rozema, H. J. Bae, and R. W. Verstappen. “Local dynamic gradient Smagorinsky model for large-eddy simulation”. In: *Physical Review Fluids* 7.7 (2022), p. 074604 (cit. on p. 21).
- [279] G. Rozza, F. Ballarin, L. Scandurra, and F. Pichi. “Real time reduced order computational mechanics”. In: *SISSA springer series, Springer Cham* (2024) (cit. on p. 97).
- [280] G. Rozza, D. B. P. Huynh, and A. T. Patera. “Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: application to transport and continuum mechanics”. In: *Archives of computational methods in engineering* 15.3 (2008), pp. 229–275 (cit. on p. 103).
- [281] G. Rozza, G. Stabile, and F. Ballarin. *Advanced reduced order methods and applications in computational fluid dynamics*. SIAM, 2022 (cit. on p. 97).
- [282] G. Rozza and K. Veroy. “On the stability of the reduced basis method for Stokes equations in parametrized domains”. In: *Computer methods in applied mechanics and engineering* 196.7 (2007), pp. 1244–1260 (cit. on pp. 97, 105).
- [283] F. Sabetghadam and A. Jafarpour. “ α regularization of the POD-Galerkin dynamical systems of the Kuramoto–Sivashinsky equation”. In: *Applied Mathematics and Computation* 218.10 (2012), pp. 6012–6026 (cit. on p. 109).

- [284] P. Sagaut. *Large eddy simulation for incompressible flows: an introduction*. Springer, 2006 (cit. on pp. 1, 14).
- [285] M. Salvador, M. Strocchi, F. Regazzoni, C. M. Augustin, L. Dede', S. A. Niederer, and A. Quarteroni. "Whole-heart electromechanical simulations using latent neural ordinary differential equations". In: *NPJ Digital Medicine* 7.1 (2024), p. 90 (cit. on p. 97).
- [286] O. San and T. Iliescu. "A stabilized proper orthogonal decomposition reduced-order model for large scale quasigeostrophic ocean circulation". In: *Advances in Computational Mathematics* 41.5 (2015), pp. 1289–1319 (cit. on pp. 111, 112).
- [287] O. San and R. Maulik. "Extreme learning machine for reduced order modeling of turbulent geophysical flows". In: *Physical Review E* 97.4 (2018), p. 042322 (cit. on pp. 97, 115).
- [288] O. San and R. Maulik. "Neural network closures for nonlinear model order reduction". In: *Advances in Computational Mathematics* 44.6 (2018), pp. 1717–1750 (cit. on p. 111).
- [289] O. San, S. Pawar, and A. Rasheed. "Variational multiscale reinforcement learning for discovering reduced order closure models of nonlinear spatiotemporal transport systems". In: *Scientific Reports* 12.1 (2022), p. 17947 (cit. on p. 111).
- [290] O. San and A. E. Staples. "High-order methods for decaying two-dimensional homogeneous isotropic turbulence". In: *Computers & Fluids* 63 (2012), pp. 105–127 (cit. on p. 17).
- [291] B. Sanderse. "Energy-conserving discretization methods for the incompressible Navier-Stokes equations : application to the simulation of wind-turbine wakes". English. Phd Thesis 2 (Research NOT TU/e / Graduation TU/e). Centrum voor Wiskunde en Informatica, 2013. ISBN: 978-90-386-3338-1. DOI: [10.6100/IR750543](https://doi.org/10.6100/IR750543) (cit. on p. 17).
- [292] B. Sanderse. "Non-linearly stable reduced-order models for incompressible flow with energy-conserving finite volume methods". In: *Journal of Computational Physics* 421 (2020), p. 109736 (cit. on p. 80).
- [293] B. Sanderse and B. Koren. "Accuracy analysis of explicit Runge–Kutta methods applied to the incompressible Navier–Stokes equations". In: *Journal of Computational Physics* 231.8 (2012), pp. 3041–3063 (cit. on p. 82).
- [294] B. Sanderse, P. Stinis, R. Maulik, and S. E. Ahmed. "Scientific machine learning for closure models in multiscale problems: A review". In: *Foundations of Data Science* 7.PNNL-SA-195444 (2024) (cit. on p. 111).
- [295] A. Sanfilippo, I. Moore, F. Ballarin, and T. Iliescu. "Approximate deconvolution Leray reduced order model for convection-dominated flows". In: *Finite Elements in Analysis and Design* 226 (2023), p. 104021 (cit. on p. 111).

- [296] M. G. de la Santa Vinuela. “Machine Learning-enhanced Variational MultiScale Evolve and Filter Strategies Reduced Order Models for Turbulent Flows”. Master’s thesis. Madrid, Spain: Universidad Politecnica de Madrid, 2026 (cit. on p. 231).
- [297] W. H. Schilders, H. A. Van der Vorst, and J. Rommes. *Model order reduction: theory, research aspects and applications*. Vol. 13. Springer, 2008 (cit. on p. 97).
- [298] P. J. Schmid. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of fluid mechanics* 656 (2010), pp. 5–28 (cit. on p. 194).
- [299] B. Schölkopf, A. Smola, and K.-R. Müller. “Nonlinear component analysis as a kernel eigenvalue problem”. In: *Neural computation* 10.5 (1998), pp. 1299–1319 (cit. on p. 191).
- [300] R. Schreiber and H. B. Keller. “Driven cavity flows by efficient numerical techniques”. In: *Journal of Computational Physics* 49.2 (1983), pp. 310–333 (cit. on p. 179).
- [301] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal Policy Optimization Algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017) (cit. on p. 51).
- [302] P. Schwerdtner and B. Peherstorfer. “Greedy construction of quadratic manifolds for nonlinear dimensionality reduction and nonlinear model reduction”. In: *SIAM Journal on Mathematics of Data Science* 8.3 (2026), pp. 793–819 (cit. on p. 170).
- [303] M. W. Scroggs, I. A. Baratta, C. N. Richardson, and G. N. Wells. “Basix: a runtime finite element basis evaluation library”. In: *Journal of Open Source Software* 7.73 (2022), p. 3982 (cit. on pp. 5, 55).
- [304] M. W. Scroggs, J. S. Dokken, C. N. Richardson, and G. N. Wells. “Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes”. In: *ACM Transactions on Mathematical Software (TOMS)* 48.2 (2022), pp. 1–23 (cit. on pp. 5, 55).
- [305] V. Shankar, V. Puri, R. Balakrishnan, R. Maulik, and V. Viswanathan. “Differentiable physics-enabled closure modeling for Burgers’ turbulence”. In: *Machine Learning: Science and Technology* 4.1 (2023), p. 015017 (cit. on p. 21).
- [306] A. Sherstinsky. “Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network”. In: *Physica d: Nonlinear phenomena* 404 (2020), p. 132306 (cit. on p. 194).
- [307] P. Siena, P. C. Africa, M. Girfoglio, and G. Rozza. “On the accuracy and efficiency of reduced order models: Towards real-world applications”. In: *Advances in Applied Mechanics* 59 (2024), pp. 245–288 (cit. on p. 97).
- [308] P. Siena, M. Girfoglio, F. Ballarin, and G. Rozza. “Data-driven reduced order modelling for patient-specific hemodynamics of coronary artery bypass grafts with physical and geometrical parameters”. In: *Journal of Scientific Computing* 94.2 (2023), p. 38 (cit. on p. 97).

- [309] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on p. 195).
- [310] S. Sirisup and G. E. Karniadakis. “Stability and accuracy of periodic flow solutions obtained by a POD-penalty method”. In: *Physica D: Nonlinear Phenomena* 202.3-4 (2005), pp. 218–237 (cit. on p. 104).
- [311] L. Sirovich. “Turbulence and the dynamics of coherent structures. Parts I–III”. In: *Quarterly of Applied Mathematics* 45.3 (1987), pp. 561–590 (cit. on pp. 3, 100).
- [312] J. Smagorinsky. “General circulation experiments with the primitive equations: I. The basic experiment”. In: *Monthly weather review* 91.3 (1963), pp. 99–164 (cit. on pp. 2, 13).
- [313] G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, G. Rozza, et al. “POD-Galerkin reduced order methods for CFD using Finite Volume Discretisation: vortex shedding around a circular cylinder”. In: *Communications in Applied and Industrial Mathematics* 8.1 (2017), pp. 210–236 (cit. on pp. 5, 97, 104).
- [314] G. Stabile and G. Rozza. “Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier–Stokes equations”. In: *Computers & Fluids* 173 (2018), pp. 273–284 (cit. on pp. 5, 154).
- [315] S. K. Star, G. Stabile, F. Belloni, G. Rozza, J. Degroote, et al. “Extension and comparison of techniques to enforce boundary conditions in Finite Volume POD-Galerkin reduced order models for fluid dynamic problems”. In: *arXiv preprint arXiv:1912.00825* (2019) (cit. on pp. 104, 148, 154, 159).
- [316] S. K. Star, B. Sanderse, G. Stabile, G. Rozza, and J. Degroote. “Reduced order models for the incompressible Navier-Stokes equations on collocated grids using a ‘discretize-then-project’ approach”. In: *International Journal for Numerical Methods in Fluids* 93.8 (2021), pp. 2694–2722 (cit. on p. 106).
- [317] G. G. Stokes. “On the theories of the internal friction of fluids in motion and of the equilibrium and motion of elastic solids”. In: *Transactions of the Cambridge Philosophical Society* 8 (1845), pp. 287–319 (cit. on pp. 1, 8).
- [318] S. Stolz and N. A. Adams. “An approximate deconvolution procedure for large-eddy simulation”. In: *Physics of Fluids* 11.7 (1999), pp. 1699–1701 (cit. on p. 20).
- [319] S. Stolz, N. A. Adams, and L. Kleiser. “An approximate deconvolution model for large-eddy simulation with application to incompressible wall-bounded flows”. In: *Physics of fluids* 13.4 (2001), pp. 997–1015 (cit. on p. 20).
- [320] M. Strazzullo. “Model Order Reduction for Nonlinear and Time-Dependent Parametric Optimal Flow Control Problems”. PhD thesis. SISSA, 2021 (cit. on p. 97).

- [321] M. Strazzullo, F. Ballarin, T. Iliescu, and C. Canuto. “New Feedback Control and Adaptive Evolve-Filter-Relax Regularization for the Navier-Stokes Equations in the Convection-Dominated Regime”. In: *arXiv preprint arXiv:2307.00675* (2023) (cit. on p. 110).
- [322] M. Strazzullo, F. Ballarin, T. Iliescu, and T. C. Rebollo. “Variational multiscale evolve and filter strategies for convection-dominated flows”. In: *Computer Methods in Applied Mechanics and Engineering* 438 (2025), p. 117811 (cit. on p. 60).
- [323] M. Strazzullo, F. Ballarin, R. Mosetti, and G. Rozza. “Model reduction for parametrized optimal control problems in environmental marine sciences and engineering”. In: *SIAM Journal on Scientific Computing* 40.4 (2018), B1055–B1079 (cit. on p. 97).
- [324] M. Strazzullo, M. Girfoglio, F. Ballarin, T. Iliescu, and G. Rozza. “Consistency of the full and reduced order models for evolve-filter-relax regularization of convection-dominated, marginally-resolved flows”. In: *International Journal for Numerical Methods in Engineering* 123.14 (2022), pp. 3148–3178 (cit. on pp. 23, 110).
- [325] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. Adaptive Computation and Machine Learning. Cambridge, Massachusetts and London, England: The MIT Press, 2020 (cit. on p. 50).
- [326] T. Taddei. “A registration method for model order reduction: data compression and geometry reduction”. In: *SIAM Journal on Scientific Computing* 42.2 (2020), A997–A1027 (cit. on p. 191).
- [327] H. Tang, J. Rabault, A. Kuhnle, Y. Wang, and T. Wang. “Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning”. In: *Physics of Fluids* 32.5 (2020) (cit. on p. 43).
- [328] A. E. Tejada-Martinez and K. E. Jansen. “A dynamic Smagorinsky model with dynamic determination of the filter width ratio”. In: *Physics of Fluids* 16.7 (2004), pp. 2514–2528 (cit. on p. 21).
- [329] M. Tezzele. “Data-driven parameter and model order reduction for industrial optimisation problems with applications in naval engineering”. PhD thesis. SISSA, 2021 (cit. on p. 97).
- [330] M. Tezzele, L. Fabris, M. Sidari, M. Sicchiero, and G. Rozza. “A multifidelity approach coupling parameter space reduction and nonintrusive POD with application to structural optimization of passenger ship hulls.” In: *International Journal for Numerical Methods in Engineering* 124.5 (2023) (cit. on p. 97).
- [331] L. Tomada, F. Pichi, and G. Rozza. “Latent Dynamics Graph Convolutional Networks for model order reduction of parameterized time-dependent PDEs”. In: *Journal of Computational Physics* (2026), p. 115150 (cit. on p. 194).

- [332] M. Towers, A. Kwiatkowski, J. Balis, G. De Cola, T. Deleu, M. Goulão, K. Andreas, M. Krimmel, A. Kg, R. Perez-Vicente, et al. “Gymnasium: A standard interface for reinforcement learning environments”. In: *Advances in Neural Information Processing Systems* 38 (2026) (cit. on p. 55).
- [333] P. H. Tsai, P. Fischer, and T. Iliescu. “A time-relaxation reduced order model for the turbulent channel flow”. In: *Journal of Computational Physics* 521 (2025), p. 113563 (cit. on p. 109).
- [334] P. Tsai, P. Fischer, and E. Solomonik. “Accelerating Galerkin Reduced-Order Models for Turbulent Flows with Tensor Decomposition”. In: *arXiv preprint arXiv:2311.03694* (2023) (cit. on p. 137).
- [335] P.-H. Tsai and P. Fischer. “Parametric model-order-reduction development for unsteady convection”. In: *Frontiers in Physics* 10 (2022), p. 903169 (cit. on p. 121).
- [336] P.-H. Tsai, A. Ivagnes, A. Quaini, T. Iliescu, and G. Rozza. “StabOp: A Data-Driven Stabilization Operator for Reduced Order Modeling”. In: *Computer Methods in Applied Mechanics and Engineering* 457 (2026) (cit. on pp. 4, 110, 120).
- [337] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. “On dynamic mode decomposition: Theory and applications”. In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421 (cit. on pp. 77, 194).
- [338] H. M. Tufo, P. Fischer, M. E. Papka, and K. Blom. “Numerical simulation and immersive visualization of hairpin vortices”. In: *Proceedings of the 1999 ACM/IEEE conference on Supercomputing*. 1999, 62–es (cit. on p. 133).
- [339] Van Reeuwijk, M and Jonker, HJJ and Hanjalić, K. “Incompressibility of the Leray- α model for wall-bounded flows”. In: *Physics of Fluids* 18.1 (2006) (cit. on p. 38).
- [340] K. Veroy and A. T. Patera. “Certified real-time solution of the parametrized steady incompressible Navier–Stokes equations: rigorous reduced-basis a posteriori error bounds”. In: *International Journal for Numerical Methods in Fluids* 47.8-9 (2005), pp. 773–788 (cit. on pp. 97, 99).
- [341] J. Vierendeels, L. Lanoye, J. Degroote, and P. Verdonck. “Implicit coupling of partitioned fluid–structure interaction problems with reduced order models”. In: *Computers & structures* 85.11-14 (2007), pp. 970–976 (cit. on p. 97).
- [342] R. Vinuesa, O. Lehmkuhl, A. Lozano-Durán, and J. Rabault. “Flow control in wings and discovery of novel approaches via deep reinforcement learning”. In: *Fluids* 7.2 (2022), p. 62 (cit. on p. 43).
- [343] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. In: *Nature methods* 17.3 (2020), pp. 261–272 (cit. on p. 30).

- [344] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos. “Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks”. In: *Proceedings. Mathematical, physical, and engineering sciences* 474.2213 (2018), p. 20170844 (cit. on p. 194).
- [345] A. Vreman. “An eddy-viscosity subgrid-scale model for turbulent shear flow: Algebraic theory and applications”. In: *Physics of fluids* 16.10 (2004), pp. 3670–3681 (cit. on p. 20).
- [346] N. Vyas, D. Morwani, R. Zhao, I. Shapira, D. Brandfonbrener, L. Janson, and S. Kakade. “SOAP: Improving and stabilizing shampoo using adam for language modeling”. In: *International Conference on Learning Representations*. Vol. 2025. 2025, pp. 93423–93444 (cit. on pp. 48, 222).
- [347] Y. Wang, H. Yao, and S. Zhao. “Auto-encoder based dimensionality reduction”. In: *Neurocomputing* 184 (2016), pp. 232–242 (cit. on p. 191).
- [348] Z. Wang, I. Akhtar, J. Borggaard, and T. Iliescu. “Proper orthogonal decomposition closure models for turbulent flows: a numerical comparison”. In: *Computer Methods in Applied Mechanics and Engineering* 237 (2012), pp. 10–26 (cit. on p. 109).
- [349] D. Wells, Z. Wang, X. Xie, and T. Iliescu. “An evolve-then-filter regularized reduced order model for convection-dominated flows”. In: *International Journal for Numerical Methods in Fluids* 84.10 (2017), pp. 598–615 (cit. on pp. 109, 110).
- [350] D. R. Wells. “Stabilization of POD-ROMs”. PhD thesis. Virginia Tech, 2015 (cit. on p. 109).
- [351] M. A. Wiering and M. Van Otterlo. “Reinforcement learning”. In: *Adaptation, learning, and optimization* 12.3 (2012), p. 729 (cit. on p. 43).
- [352] D. C. Wilcox et al. *Turbulence modeling for CFD*. Vol. 2. DCW industries La Canada, CA, 1998 (cit. on p. 1).
- [353] K. Willcox and J. Peraire. “Balanced model reduction via the proper orthogonal decomposition”. In: *AIAA journal* 40.11 (2002), pp. 2323–2330 (cit. on p. 97).
- [354] Wilson, RB. “A simplicial method for convex programming”. In: *Harvard University: Cambridge, MA, USA* (1963) (cit. on p. 29).
- [355] D. Xiao, F. Fang, A. G. Buchan, C. C. Pain, I. M. Navon, J. Du, and G. Hu. “Non-linear model reduction for the Navier–Stokes equations using residual DEIM method”. In: *Journal of Computational Physics* 263 (2014), pp. 1–18 (cit. on p. 102).
- [356] D. Xiao, C. Heaney, L. Mottet, F. Fang, W. Lin, I. Navon, Y. Guo, O. Matar, A. Robins, and C. Pain. “A reduced order model for turbulent flows in the urban environment using machine learning”. In: *Building and Environment* 148 (2019), pp. 323–337 (cit. on p. 97).

- [357] X. Xie, M. Mohebujjaman, L. G. Rebholz, and T. Iliescu. “Data-driven filtered reduced order modeling of fluid flows”. In: *SIAM Journal on Scientific Computing* 40.3 (2018), B834–B857 (cit. on pp. [111](#), [115](#), [116](#), [118](#), [123](#), [168](#), [170](#)).
- [358] X. Xie, C. Webster, and T. Iliescu. “Closure learning for nonlinear model reduction using deep residual neural network”. In: *Fluids* 5.1 (2020), p. 39 (cit. on p. [116](#)).
- [359] X. Xie, D. Wells, Z. Wang, and T. Iliescu. “Approximate deconvolution reduced order modeling”. In: *Computer Methods in Applied Mechanics and Engineering* 313 (2017), pp. 512–534 (cit. on p. [111](#)).
- [360] C.-C. Yu and J. C. Heinrich. “Petrov—Galerkin method for multidimensional, time-dependent, convective-diffusion equations”. In: *International Journal for numerical methods in engineering* 24.11 (1987), pp. 2201–2215 (cit. on p. [103](#)).
- [361] M. Zancanaro, M. Mrosek, G. Stabile, C. Othmer, and G. Rozza. “Hybrid neural network reduced order modelling for turbulent flows with geometric parameters”. In: *Fluids* 6.8 (2021), p. 296 (cit. on pp. [114](#), [162](#)).
- [362] P. Zappi, A. Ivagnes, N. Tonicello, and G. Rozza. “Efficient and Accurate Surrogate Modeling of Turbulent Flows via Space-Dependent Aggregation and Reduced Order Models”. In: *arXiv preprint arXiv:2603.15816* (2026) (cit. on pp. [5](#), [232](#)).