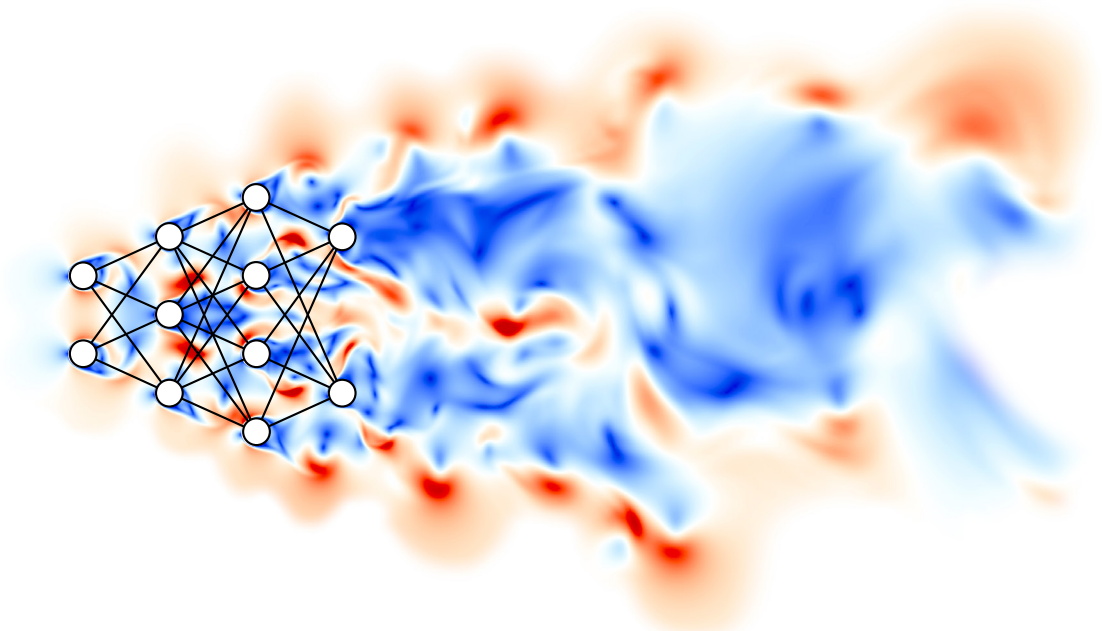
KTH ROYAL INSTITUTE
OF TECHNOLOGY

Doctoral Thesis in Engineering Mechanics

Time, space and control: deep-learning applications to turbulent flows

LUCA GUASTONI



Stockholm, Sweden 2023

Time, space and control: deep-learning applications to turbulent flows

LUCA GUASTONI

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Doctor of Philosophy on Monday the 12th June 2023, at 10:00 a.m. in F3, KTH, Lindstedtsvägen 26 & 28, Stockholm.

Doctoral Thesis in Engineering Mechanics
KTH Royal Institute of Technology
Stockholm, Sweden 2023

TRITA-SCI-FOU 2023:27
ISBN 978-91-8040-601-7

Printed by: Universitetsservice US-AB, Sweden 2023

Time, space and control: deep-learning applications to turbulent flows

Luca Guastoni
FLOW, Engineering Mechanics, KTH Royal Institute of Technology
SE–100 44 Stockholm, Sweden

Abstract

In the present thesis, the application of deep learning and deep reinforcement learning to turbulent-flow simulations is investigated. Deep-learning models are trained to perform temporal and spatial predictions, while deep reinforcement learning is applied to a flow-control problem, namely the reduction of drag in an open channel flow.

Long short-term memory (LSTM, Hochreiter & Schmidhuber 1997) networks and Koopman non-linear forcing (KNF) models are optimized to perform temporal predictions in two reduced-order-models of turbulence, namely the nine-equations model proposed by Moehlis *et al.* (2004) and a truncated proper orthogonal decomposition (POD) of a minimal channel flow (Jiménez & Moin 1991). In the first application, both models are able to produce accurate short-term predictions. Furthermore, the predicted system trajectories are statistically correct. KNF models outperform LSTM networks in short-term predictions, with a much lower training computational cost. In the second task, only LSTMs can be trained successfully, predicting trajectories that are statistically accurate.

Spatial predictions are performed in two turbulent flows: an open channel flow and a boundary-layer flow. Fully-convolutional networks (FCNs) are used to predict two-dimensional velocity-fluctuation fields at a given wall-normal location using wall measurements (and vice versa). Thanks to the non-linear nature of these models, they provide a better reconstruction performance than optimal linear methods like extended POD (Borée 2003).

Finally, we show the potential of deep reinforcement learning in discovering new control strategies for turbulent flows. By framing the fluid-dynamics problem as a multi-agent reinforcement-learning environment and by training the agents using a location-invariant deep deterministic policy-gradient (DDPG) algorithm, we are able to learn a control strategy that achieves a remarkable 30% drag reduction, improving over existing strategies by about 10 percentage points.

Key words: turbulence, deep learning, deep reinforcement learning, flow control.

Tid, rum och kontroll: djupinlärningsapplikationer för turbulenta flöden

Luca Guastoni
FLOW, Institutionen för Mekanik, Kungliga Tekniska högskolan,
SE-100 44 Stockholm, Sverige

Sammanfattning

I den förinställda avhandlingen undersöks tillämpningen av djupinlärning och djupförstärkningsinlärning på turbulenta flödessimuleringar. Modeller för djupinlärning tränas för att utföra tids- och rumsförutsägelser, medan djupförstärkningsinlärning tillämpas på ett flödeskontrollproblem, nämligen minskningen av motståndet i ett öppet kanalflöde.

Long short-term memory (LSTM, Hochreiter & Schmidhuber 1997) nätverk och Koopman non-linear forcing (KNF) modeller optimeras för att utföra tidsförutsägelser i två turbulensmodeller med reducerad ordning, nämligen nio-ekvationsmodellen som föreslagits av Moehlis *et al.* (2004) och en trunkerad proper orthogonal decomposition (POD) av en minimal kanalströmning (Jiménez & Moin 1991). I den första applikationen kan båda modellerna producera korrekta korttidsförutsägelser, dessutom är de förutsagda trajektorierna statistiskt korrekta. KNF-modeller överträffar LSTM-nätverk i kortsiktiga förutsägelser, med en mycket lägre utbildningsberäkningskostnad. I den andra uppgiften kan endast LSTM nätverken tränas framgångsrikt, med trajektorier som är statistiskt korrekta.

Spatiala förutsägelser utförs i två turbulenta flöden, ett öppet kanalflöde och en gränsskikt. Fully-convolutional networks (FCN) används för att förutsäga tvådimensionella hastighetsfluktuationsfält vid givet avstånd från väggen med hjälp av väggmätningar (och vice versa). Tack vare deras icke-linjär karaktär ger dessa modeller bättre rekonstruktionsprestanda än optimala linjära metoder som extended POD (Borée 2003).

Slutligen visar vi potentialen med djup förstärkningsinlärning för att upptäcka nya kontrollstrategier i turbulenta flöden. Genom att inrama strömningsmekaniska problemet som en förstärknings-inlärningsmiljö med flera agenter och genom att träna agenterna med hjälp av en positionsinvariant deep deterministic policy gradient (DDPG) algoritm, kan vi lära oss en kontrollstrategi som uppnår en anmärkningsvärd 30% minskning av luftmotståndet, vilket jämfört med existerande strategier är en förbättring med cirka 10 procentenheter.

Nyckelord: turbulens, djupinlärning, djupförstärkningsinlärning, flödeskontroll.

Preface

This thesis describes different applications of deep learning and deep reinforcement learning to turbulent-flow simulations. The goals of these applications are temporal predictions, spatial reconstructions or flow control. In the first part, we describe the numerical simulations that have been performed, as well as the data-driven tools that have been tested for each application. A theoretical introduction to the tools that have been used to compare the performance of the different models is also provided. Since we can identify three different applications, the papers in the second part are organized by topic, rather than following the chronological order of publication. Note that paper 6 does not belong to any of the deep-learning applications mentioned before; however, the simulation was initially performed to sample the data for spatial predictions, hence the paper is included in that group. The publications in the second part are adjusted to comply with the present thesis format for consistency, but their contents have not been altered as compared with their original counterparts.

Cover: Representation (with artistic license) of a fully-connected neural network in a fluid flow. Simulation performed by Pol Suarez.

Paper 1. P. A. Srinivasan, L. Guastoni, H. Azizpour, P. Schlatter & R. Vinuesa, 2019. *Predictions of turbulent shear flows using deep neural networks.* Phys. Rev. Fluids **4** (5), 054603.

Paper 2. H. Eivazi, L. Guastoni, P. Schlatter, H. Azizpour & R. Vinuesa, 2021. *Recurrent neural networks and Koopman-based frameworks for temporal predictions in a low-order model of turbulence.* Int. J. Heat Fluid Flow **90**, 108816.

Paper 3. G. Borrelli, L. Guastoni, H. Eivazi, P. Schlatter, & R. Vinuesa, 2022. *Predicting the temporal dynamics of turbulent channels through deep learning.* Int. J. Heat Fluid Flow **96**, 109010.

Paper 4. L. Guastoni, M. P. Encinar, P. Schlatter, H. Azizpour & R. Vinuesa, 2020. *Prediction of wall-bounded turbulence from wall quantities using convolutional neural networks.* J. Phys.: Conf. Ser. **1522**, 012022.

Paper 5. L. Guastoni, A. Güemes, A. Ianiro, S. Discetti, P. Schlatter, H. Azizpour & R. Vinuesa, 2020. *Convolutional-network models to predict wall-bounded turbulence from wall quantities.* J. Fluid Mech. **928**, A27.

Paper 6. A. G. Balasubramanian, L. Guastoni, P. Schlatter, & R. Vinuesa, 2023. *Direct numerical simulation of a zero-pressure-gradient thermal turbulent boundary layer up to $Pr = 6$.* Submitted to J. Fluid Mech.

Paper 7. L. Guastoni, A. G. Balasubramanian, A. Güemes, A. Ianiro, S. Discetti, P. Schlatter, H. Azizpour & R. Vinuesa, 2023. *Fully-convolutional networks for velocity-field predictions based on the wall heat flux in turbulent boundary layers.* Submitted to J. Fluid Mech.

Paper 8. A. G. Balasubramanian, L. Guastoni, P. Schlatter, H. Azizpour & R. Vinuesa, 2023. *Predicting the wall-shear stress and wall pressure through convolutional neural networks.* Submitted to Int. J. Heat Fluid Flow.

Paper 9. L. Guastoni, J. Rabault, P. Schlatter, H. Azizpour & R. Vinuesa, 2023. *Deep reinforcement learning for turbulent drag reduction in channel flows.* Eur. Phys. J. E **46** (4), 27.

June 2023, Stockholm
Luca Guastoni

Division of work between authors

The main advisor for the project is Ricardo Vinuesa (RV), with Hossein Azizpour (HA) and Philipp Schlatter (PS) as co-advisors.

Paper 1. The timeseries dataset was generated by Premanand Alathur Srinivasan (PAS), who also trained the MLP and LSTM network models, under the supervision of Ricardo Vinuesa (RV), Hossein Azizpour (HA) and Philipp Schlatter (PS). The ordinary-differential-equations model was re-implemented by Luca Guastoni (LG) to generate a higher-Reynolds-number dataset, for paper-review purposes. The LSTM models were trained with the new dataset by LG, with input from RV, HA and PS. Poincaré maps were computed by PAS and Lyapunov exponents by LG. The paper has been written by PAS, RV and LG, with comments provided by HA and PS.

Paper 2. The Koopman nonlinear forcing (KNF) models were implemented by Hamidreza Eivazi (HE). The LSTM performing multiple predictions at training time was implemented by LG. The training-stopping criterion was proposed by LG. Both HE and LG were advised by RV, HA and PS. The robustness analysis of the KNF and LSTM models was performed by HE. The paper was written by HE and LG. RV, PS and HA revised the manuscript and provided input.

Paper 3. The simulation was set up by Giuseppe Borrelli (GB), who also computed the proper orthogonal decomposition (POD) and trained the LSTM models. LG, RV, PS and HA provided input on the implementation and training. The predictions analysis was performed by GB. The paper was written by GB with input from LG and RV.

Paper 4. The open-channel-flow simulation setup was performed by LG, with input from RV and PS. The neural-network models were designed and trained by LG, with feedback from RV and HA. The linear-stochastic-estimation (LSE) model for comparison was implemented by Miguel P. Encinar. The post-processing was done by LG. The paper was written by LG, with the help of RV, HA and PS.

Paper 5. The numerical simulations were performed by LG, with input from RV and PS. The FCN model was implemented and trained by LG, with the help of RV and HA. The FCN-POD model was designed and trained by Alejandro Güemes (AG), with the support of Andrea Ianiro (AI) and Stefano Discetti (SD). The analysis of the results was performed by LG and AG, with feedback from RV, HA, PS, AI and SD. The paper was written by LG and AG. Comments were provided by RV, HA, PS, AI and SD.

Paper 6. The simulation was set up by Arivazhagan G. Balasubramanian (AGB), with input from LG, PS and RV. The dataset was analyzed by AGB. LG, RV, PS and HA provided feedback on the post-processing results. The paper was written by AGB and LG. RV revised the manuscript and provided input.

Paper 7. The dataset was generated by AGB, with input from LG, PS and RV. The network model was implemented and trained by AGB, starting from the previous code implementation by LG. LG, RV and HA provided feedback on the network design. The post-processing of the results was performed by AGB, with input from LG, RV and HA. The paper was written by LG and AGB. RV revised the manuscript and provided input.

Paper 8. The network architecture was designed by AGB, with input from LG, PS, HA and RV. The training of the model and the post-processing of the results were performed by AGB. The paper was written by AGB. LG and RV edited the manuscript.

Paper 9. The general idea was conceived by RV. The deep reinforcement learning (DRL) interface for the flow simulation was developed by LG, with input by RV, PS and Jean Rabault (JR). The simulations and post-processing of the results were performed by LG. JR, RV and HA provided feedback. The paper was written by LG and edited by RV and JR. Comments were provided by RV, JR and HA.

Conferences

Part of the work in this thesis has been presented at the following international conferences. The presenting author is underlined.

Luca Guastoni, Prem A. Srinivasan, Hossein Azizpour, Philipp Schlatter, and Ricardo Vinuesa. *On the use of recurrent neural networks for predictions of turbulent flows.* 11th International Symposium on Turbulence and Shear Flow Phenomena, TSFP. Southampton, United Kingdom, 2019.

Luca Guastoni, Prem A. Srinivasan, Hossein Azizpour, Philipp Schlatter, and Ricardo Vinuesa. *Predictions of turbulent shear flows by neural networks and application to off-wall boundary conditions.* 17th European Turbulence Conference. Torino, Italy, 2019.

Luca Guastoni, Alejandro Güemes, Andrea Ianiro, Stefano Discetti, Philipp Schlatter, Hossein Azizpour and Ricardo Vinuesa. *Predictions in wall-bounded turbulence through convolutional-network models using wall quantities.* 73rd Annual Meeting of the APS Division of Fluid Dynamics. Chicago, IL - Virtual, 2020.

Arivazhagan G. Balasubramanian, Luca Guastoni, Alejandro Güemes, Andrea Ianiro, Stefano Discetti, Philipp Schlatter, Hossein Azizpour and Ricardo Vinuesa. *Predicting the near-wall region of turbulence through convolutional neural networks.* 13th International ERCOFTAC Symposium on Engineering Turbulence Modelling and Measurements. Rhodes, Greece - Hybrid, 2021.

Luca Guastoni, Jean Rabault, Ali Ghadirzadeh, Philipp Schlatter, Hossein Azizpour and Ricardo Vinuesa. *Deep reinforcement learning*

for active drag reduction in wall turbulence. 74th Annual Meeting of the APS Division of Fluid Dynamics. Phoenix, AZ, 2021.

Luca Guastoni, Arivazhagan G. Balasubramanian, Alejandro Güemes, Andrea Ianiro, Stefano Discetti, Philipp Schlatter, Hossein Azizpour and Ricardo Vinuesa. *Non-intrusive sensing in turbulent boundary layers via deep fully-convolutional neural networks*. 12th International Symposium on Turbulence and Shear Flow Phenomena, TSFP. Osaka, Japan - Virtual, 2022.

Luca Guastoni, Jean Rabault, Ali Ghadirzadeh, Philipp Schlatter, Hossein Azizpour and Ricardo Vinuesa. *Active drag reduction in turbulent open channel flow using deep reinforcement learning*. 75th Annual Meeting of the Division of Fluid Dynamics. Indianapolis, IN, 2022.

Luca Guastoni, Jean Rabault, Philipp Schlatter, Ricardo Vinuesa and Hossein Azizpour. *Discovering drag reduction strategies in wall-bounded turbulent flows using deep reinforcement learning*. International Conference on Learning Representations, ICLR. Kigali, Rwanda, 2023.

Contents

Part II - Papers

Part I

Overview and summary

Chapter 1

Introduction

Turbulence defines the complex motion of a fluid, characterized by sudden changes in velocity and pressure. Even if the fluid motion can be described analytically by a deterministic system of equations (named after Claude-Louis Navier and George Gabriel Stokes), it can exhibit chaotic behaviour, resulting from the non-linear terms in these equations. The onset of the chaotic behavior in fluid flows is related to the Reynolds number Re, which represents the relative importance of inertial and viscous forces. The different mechanisms that lead to turbulence are not reviewed here, instead, we focus only on fully-developed turbulent flows (even though we do try to re-laminarize them in chapter 5).

To date, a complete analytical treatment of the turbulent flow has proven to be very difficult, if not impossible (Benzi & Vulpiani 2022). Hence, we resort to experimental and numerical investigations to infer the fundamental physical mechanisms underlying turbulence. The very first studies relied on experimental evidence like the one described by Reynolds (1883). Even the Kolmogorov theory of turbulence (Kolmogorov 1941) was considered purely phenomenological by his own author (Sinai 2003). The direct numerical simulation (DNS) by Kim *et al.* (1987) showed that the flow physics could be accurately reproduced with numerical tools, and since then DNSs have become a fundamental turbulence research tool.

Turbulence characterizes fluid flows in a wide range of circumstances, both in nature and in engineering applications. Its widespread occurrence and importance in natural and industrial processes determine a strong interdisciplinary interest, and it is studied in different contexts, ranging from aviation to medical devices, from astrophysics to geophysics. At the same time, the study of turbulence integrates elements from a number of different research fields, including statistical mechanics and applied mathematics. With a syncretistic approach, methods and practices from other fields have been applied to turbulence research and refined in an effort to obtain new insights into the physics of turbulent flows. As an example, a large part of the progress in the understanding of turbulence is related to the application and development of numerical methods for fluid-flow simulations, as well as the methods designed to analyse the data produced with these computations.

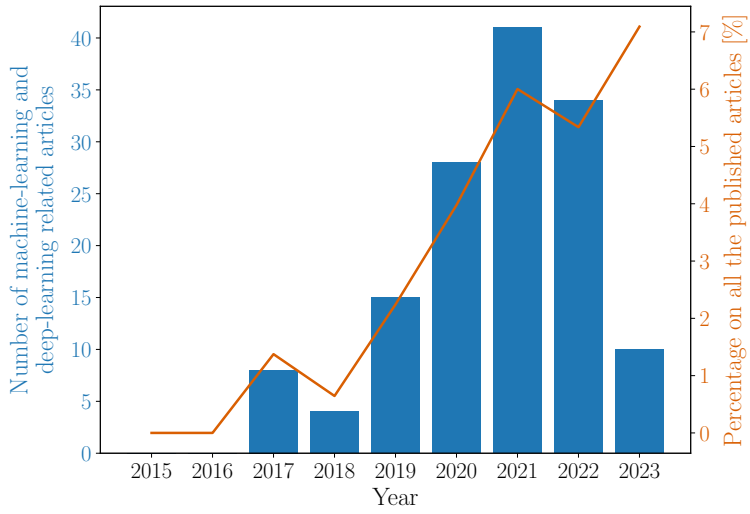


Figure 1.1: Publications on *Physical Review Fluids* or *Physical Review Letters* about fluid dynamics mentioning either 'machine learning' or 'deep learning' in the title or abstract. The axis on the right indicates the percentage with respect to the overall number of published manuscripts about fluid dynamics in these journals each year.

In recent years, machine-learning (ML) and more specifically deep-learning (DL) techniques have emerged as effective tools to perform classification and regression tasks, leveraging large amounts of data. Thanks to the intrinsic non-linearity and domain-agnostic nature of these algorithms, they have been applied to different domains and problems, allowing the development of novel data-driven solutions for natural language processing (NLP), computer vision and robotics. More recently, these techniques have also been applied in natural sciences and in this thesis we explore the use of deep learning and deep reinforcement learning in turbulent-flow simulations. This work is part of a larger effort from the research community, as testified by the increasing number of publications on this topic (figure 1.1). There are different areas in which ML and DL can be used to enhance CFD, as reviewed by Vinuesa & Brunton (2022). Here, three different yet connected application domains are explored: time, space and control.

Deep neural networks are trained to perform different types of predictions in chaotic or turbulent systems. Turbulence is a phenomenon with non-trivial features in both time and space, for this reason, we analyse the two problems separately using specific tools for each case, as described in chapters 3 and 4, respectively. The deep-learning models used for temporal and spatial predictions

are all trained using a *supervised-learning* approach. For each input sample, the corresponding correct output (also called *ground truth*) is provided. The (mean-squared) difference between the model output and the ground truth is calculated and the learnable parameters of the models are optimized using their gradient with respect to the error.

When deep learning is combined with reinforcement learning, the same network models can be used for flow-control purposes, by training them to approximate quantities of interest in the reinforcement-learning framework described in chapter 5. Differently from the deep-learning algorithms used for prediction, the discovery of novel control strategies with DRL is not supervised, meaning that no ground truth is provided and the agent interacts with the system to be controlled, identifying by trial and error the best-performing actuation strategies. Note that it should not be considered *unsupervised* either, since unsupervised learning algorithms focus only on discovering a hidden structure in the data (Sutton & Barto 2018). Most of the applications of machine-learning-based techniques to flow control are very recent, but they have shown promising results in progressively more complex settings (Vignon *et al.* 2023), towards applications of great economical and environmental impact such as the control around wings (Vinuesa *et al.* 2022).

Thesis structure. After this introduction, the remainder of part I of the thesis provides a description of the numerical simulations and the three deep-learning and deep-reinforcement learning applications investigated. Chapter 2 introduces the numerical methods used to simulate the turbulent flows and the cases from which the datasets for training and testing are sampled. Chapter 3 compares the different architectures used for temporal predictions, while chapter 4 is devoted to the description of the models used for spatial reconstructions. In this latter chapter, the implications of training choices such as the definition of the loss function are also discussed. (Deep) reinforcement learning is introduced in chapter 5, along with the details of its application to reduce the drag in wall-bounded turbulent flows. Concluding remarks and future directions are outlined in chapter 6. Part II includes the papers submitted to or published in peer-reviewed journals, organized by topic.

Chapter 2

Numerical simulations

Training and test datasets need to be provided in order to optimize the neural-network models. Nowadays it is possible to leverage existing datasets made available by other researchers, most notably the John Hopkins datasets (Li *et al.* 2008) and TurbRot (Biferale *et al.* 2020). One important advantage of this approach is that it facilitates the reproducibility of the results. However, this comes at the cost of lower flexibility in terms of sampling frequency and sampled quantities. Depending on the application, the required datasets might not be available or sufficiently large. In these cases, the dataset generation represents the initial part of the work, during which a numerical experiment is set up to sample inputs and outputs for the model (in our supervised-learning scenario). When the dataset is not readily available, the outcome of the numerical simulation has its own scientific value beyond the mere accumulation of samples (see **Paper 6**).

In some studies (see **Paper 1** and **Paper 2**) we used a reduced-order model for turbulent flows, namely the nine-equations model by Moehlis *et al.* (2004). In this case, the integration of the equation is performed with a multi-step Runge–Kutta scheme, which is relatively computationally inexpensive. In all the other works, we performed direct numerical simulations (DNSs) to sample the necessary data. In these simulations, the governing equations are integrated without resorting to any turbulence modelling or filtering, and the temporal and spatial resolutions need to be sufficient to simulate all the relevant time and lengthscales.

In all our simulations, we consider an incompressible flow. Its evolution is governed by the Navier–Stokes equations, which can be written in a non-dimensional form as:

$$\nabla \cdot \mathbf{u} = 0, \tag{2.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla P = \frac{1}{Re}\nabla^2 \mathbf{u}. \tag{2.2}$$

Under the assumption of an incompressible flow these two equations are decoupled from the energy equation, hence the pressure P just acts as a Lagrange multiplier to enforce the incompressibility constraint.

We considered wall-bounded flows in relatively simple geometries. This allowed us to use an established pseudo-spectral method to solve the Navier–Stokes equations. In particular, we used SIMSON (Chevalier *et al.* 2007), an efficient and parallelized implementation of the numerical method. The flow is simulated in domain Ω of size $L_x \times L_y \times L_z$, where x, y and z indicate the streamwise, wall-normal and spanwise directions. The corresponding velocity components are denoted by (u, v, w).

The numerical method does not solve directly for the three velocity components. Following the approach that was first proposed by Kim *et al.* (1987), the continuity and the momentum equations, (2.1) and (2.2) respectively, are replaced with two scalar equations. The change of variables allows for the projection of the momentum equation into a divergence-free space, eliminating the need for the Lagrange multiplier P. The resulting equations are a function of the wall-normal component of the vorticity η and the wall-normal component of the velocity v:

$$\begin{cases} \dfrac{\partial \nabla^2 v}{\partial t} = h_v + \dfrac{1}{Re}\nabla^2\nabla^2 v \\ \dfrac{\partial \eta}{\partial t} = h_\eta + \dfrac{1}{Re}\nabla^2 \eta, \end{cases} \tag{2.3}$$

with:

$$h_v = \frac{\partial}{\partial y}\left[\frac{\partial}{\partial x}\left(\mathbf{u}\cdot\nabla u\right) + \frac{\partial}{\partial z}\left(\mathbf{u}\cdot\nabla w\right)\right] - \left(\frac{\partial^2}{\partial^2 x} + \frac{\partial^2}{\partial^2 z}\right)\left(\mathbf{u}\cdot\nabla v\right), \tag{2.4}$$

and:

$$h_\eta = \frac{\partial}{\partial x}(\mathbf{u}\cdot\nabla w) - \frac{\partial}{\partial z}(\mathbf{u}\cdot\nabla u). \tag{2.5}$$

Once v and η are known, the wall-parallel velocity components are subsequently computed by using the definition of η and the continuity equation (2.1):

$$\begin{cases} \dfrac{\partial u}{\partial x} + \dfrac{\partial w}{\partial z} = -\dfrac{\partial v}{\partial y} \\ \dfrac{\partial u}{\partial z} - \dfrac{\partial w}{\partial x} = \eta. \end{cases} \tag{2.6}$$

Periodic boundary conditions are considered in the streamwise and spanwise directions. This implies that the equations can be discretized using a truncated Fourier-series expansion in the wall-parallel directions. Chebyshev polynomials are used in the wall-normal direction. Throughout this thesis, N_x and N_z indicate the number of modes used in the streamwise and spanwise directions, while N_y refers to the highest order of polynomials included in the Chebyshev series expansion. The time advancement is performed in a semi-implicit way, using a third-order Runge–Kutta method for the nonlinear terms and a second-order Crank–Nicolson algorithm for the linear ones.

In this thesis, we consider three different wall-bounded flows: standard channel flow, open channel flow and boundary-layer flow. In all cases, we

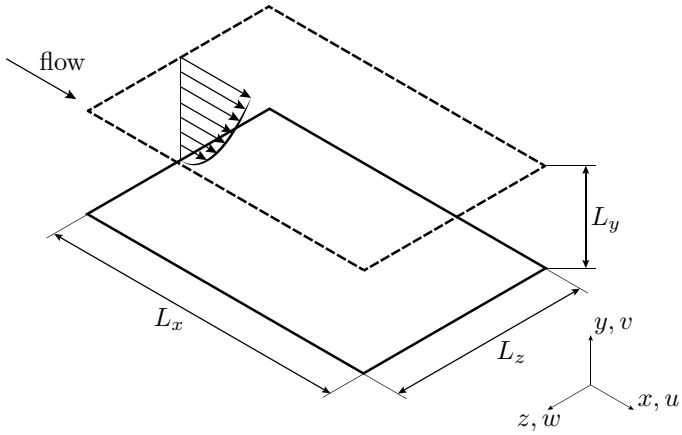


Figure 2.1: Computational domain and frame of reference for the DNS of a turbulent open channel flow. From Guastoni *et al.* (2021).

consider a Dirichlet condition on the lower boundary. This way, it is possible to impose an arbitrary velocity distribution:

$$v|_{y=0} = v_{\text{wall}}(x, z, t). \tag{2.7}$$

Note that v_{wall} can be set to zero, obtaining a no-slip condition to simulate the wall. Alternatively, a non-zero velocity distribution can be chosen, in order to simulate blowing and suction at the wall. The other boundary condition in the wall-normal direction depends on the flow case being simulated.

2.1. Channel flows

Among wall-bounded flows, channel flows are one of the simplest examples of *internal flows*, along with pipe flows. As mentioned before, we consider periodic boundary conditions in the spanwise direction. Provided that L_z is large enough, a channel flow is equivalent to the flow in a duct with an infinite aspect ratio (Pope 2000). The flow can be simulated considering two walls at the top and bottom boundaries of the domain. We refer to this case as *standard channel flow*. This setting is commonly chosen in literature and it has been the reference case for fundamental turbulence research in the last decades.

Alternatively, it is possible to impose a symmetry condition at the top boundary, while keeping a wall at the bottom. When these boundary conditions are chosen, an *open channel flow* is simulated, as depicted in figure 2.1. This can be considered to be equivalent to the lower half of a standard channel flow, even though the different boundary conditions affect the flow close to the top boundary. In wall-bounded flows, the presence of an additional wall significantly affects the flow. In standard channel flows, the effect of each wall can extend beyond the centerline of the channel (Lozano-Durán *et al.* 2012). For this, open

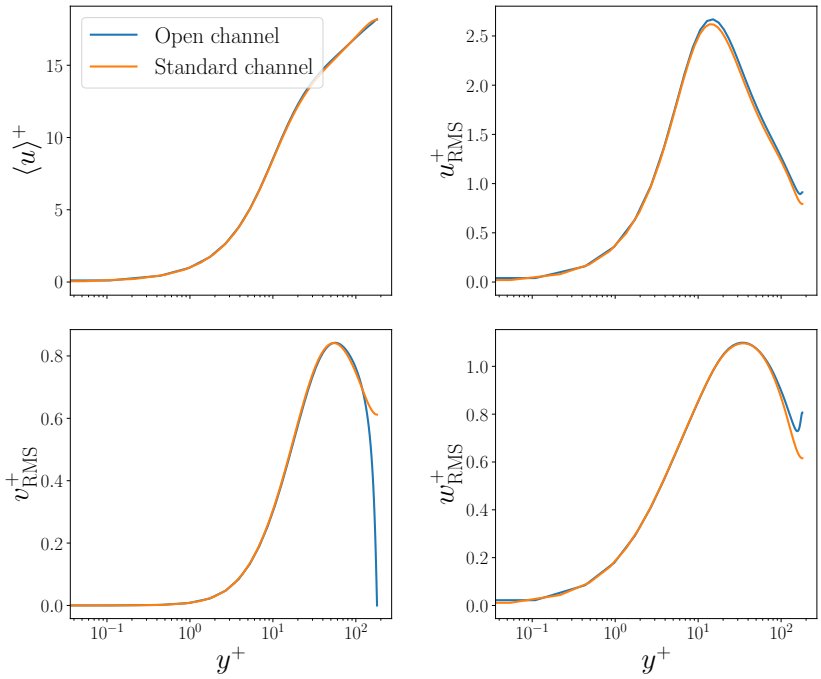


Figure 2.2: Comparison of first- and second-order statistics between the standard channel flow and the open channel flow.

channel flows are more suitable to analyze the effect of an individual wall on the fluid flow. On the other hand, the domain size limits the maximum size of the turbulent structures that can develop in the flow, and the open-channel simulations lack the larger structures that can be observed in the standard channel flow.

The channel flow simulations are performed at $Re_\tau = 180$ and $Re_\tau = 550$, as specified in each study. This Reynolds number is based on the friction velocity u_τ and the characteristic length δ. The friction velocity is defined as $u_\tau = \sqrt{\tau_w/\rho}$, where τ_w is the wall-shear stress and ρ is the fluid density. The characteristic length is the semi-height of the domain ($L_y/2$) in the standard channel flow or the height L_y in the open channel flow.

In both the standard and open channel, the flow is statistically independent of x and z. The differences between the two channel-flow configurations are evident when the statistics of the fully-developed flows are compared (figure 2.2). The mean streamwise velocity $\langle u \rangle$ is averaged in time and the homogeneous directions. The fluctuation intensity for each velocity component is defined as $u_{i,\mathrm{RMS}} = \sqrt{\langle (u_i - \langle u_i \rangle)^2 \rangle}$. Note that '+' denotes the scaling in terms of either

the friction velocity u_τ or the viscous length $\ell^* = \nu/u_\tau$ (where ν is the fluid kinematic viscosity). The mean profile of the velocity is largely unaffected by the choice of the channel-flow simulation. On the one hand, the streamwise and spanwise fluctuation intensities differ only in the vicinity of the channel centerline. On the other hand, the wall-normal fluctuations are the most affected, since the boundary condition imposed at the upper wall is related to the wall-normal derivative of the velocity v.

As mentioned before, the size of the domain plays an important role in the turbulent features that can be simulated. The cost of the simulations grows very quickly with the Reynolds number if all the physically-relevant features of the flow have to be resolved. For this, the choice of domain size is a result of the trade-off between the largest simulated turbulent structures and the overall computational cost. One approach to minimize the computational cost is described by Jiménez & Moin (1991). The domain size is limited to the minimal flow unit in which a self-sustained turbulent flow is possible. In this computational box, only a single low-speed streak is simulated. The small scales in the minimal flow unit are adequately represented since the resolution in a smaller box is correspondingly better in a smaller domain, when the same number of points is considered (Jiménez & Moin 1991). When comparing the statistics with those of a channel flow computed in a larger domain, the statistical moments are in good agreement in the near-wall region, while large discrepancies can be observed in the outer layer. This approach can also be advantageous when the number of simulations is large, for instance when the fluid solver is coupled with a reinforcement-learning algorithm, as described in **Paper 9**.

2.2. Turbulent boundary layer

The other flow case we considered is a turbulent boundary layer, which is an example of external flow. Differently from channel flows, the boundary layer is a spatially-developing flow, hence all the quantities that characterize the flow exhibit a dependence on the streamwise coordinate. For example, the reference length δ is the boundary-layer thickness and it increases with x.

Also in this case, the boundary conditions need to be defined. The flow is periodic in the spanwise direction. Since the boundary layer is a spatially-developing flow, a fringe forcing at the outlet region is necessary to impose periodic boundary conditions also in the streamwise direction. This allows us to use the same numerical method used for channel flows. The fringe forcing is defined as:

$$\mathbf{f} = \lambda(x)(\mathcal{U} - \mathbf{u}), \tag{2.8}$$

where $\lambda(x)$ is a smooth top-hat function that is non-zero only in the fringe region and $\mathcal{U}(x, y)$ is a base flow, for instance, the Falkner–Skan–Cooke (FSC) flow. The turbulent-boundary-layer simulations in this thesis are performed with a no-slip boundary condition at the wall. On the upper boundary, a Neumann boundary condition is imposed, in order to approximate an undisturbed flow

at an infinite distance from the lower boundary. In particular, the boundary condition is defined as

$$\left.\frac{\partial \mathbf{u}}{\partial y}\right|_{y=y_L} = \left.\frac{\partial \mathcal{U}}{\partial y}\right|_{y=y_L}, \tag{2.9}$$

where $\mathcal{U}(x, y)$ is the same base flow used for the fringe forcing.

Our work involving boundary-layer simulations is mainly targeting non-intrusive sensing as detailed in chapter 4. To this end, different passive scalars were simulated and sampled from the turbulent-boundary-layer DNS. Passive scalars do not alter the velocity flow field but they are advected and diffused following the governing equation:

$$\frac{\partial \theta}{\partial t} = -\mathbf{u} \cdot \nabla \theta + \frac{1}{Re\, Pr} \nabla^2 \theta + f_\theta, \tag{2.10}$$

where Pr is the Prandtl number and f_θ represents the fringe forcing applied to enforce periodic boundary conditions in the streamwise direction. For all the passive scalars, Dirichlet boundary conditions are imposed at the wall:

$$\theta|_{y=0} = 0, \tag{2.11}$$

and at the upper boundary:

$$\theta|_{y=y_L} = 1. \tag{2.12}$$

A linear profile is considered for the scalars at the inlet.

The highest Prandtl number considered in this simulation is $Pr = 6$. To date, no turbulent-boundary-layer simulation with scalars at such a high Prandtl is available in the literature. Therefore, the statistical characterization of the scalar flow field in a turbulent boundary layer is reported in **Paper 6**.

Chapter 3

Time: recurrent networks and Koopman non-linear forcing framework

The first application of deep-learning techniques we consider is the prediction of the temporal evolution of a system. The objective of these applications is to predict accurately the short-term behaviour, while reproducing correctly the long-term statistics of the system. Natural systems are characterized by a wide range of spatio-temporal scales and identifying the dynamics at all these scales is a first step to predict the future behaviour of the system. For this, different approaches for this have been proposed in the recent years (including SINDy, Brunton *et al.* 2016), however, knowing the underlying equations that govern the dynamics might not be sufficient to perform accurate predictions, as the system may be high dimensional and/or may exhibit chaotic behaviour. Besides classical approaches such as Kalman filtering (Kalman 1960) and its extensions for nonlinear applications (Einicke & White 1999; Julier & Uhlmann 1997), in the last years, the availability of large datasets fostered the development of data-driven prediction techniques. The performance of machine-learning models like multilayer perceptrons (MLP, Rumelhart *et al.* 1985) has been assessed, however, better results can be obtained by models that can take advantage of the sequential nature of time series data, like the recurrent neural networks (RNNs) described in section 3.1. During the training, these models develop an internal representation of the dynamics that can be later used to produce new state-space trajectories.

Even if the training of recurrent neural networks can be performed directly on the system variables, it can be challenging for the model to capture all relations between the different variables, if a high-dimensional system is considered. To address this issue, one approach is to perform dimensionality reduction. Data-driven techniques like proper orthogonal decomposition (POD) provide a set of functions (or *modes*) that define a linear reduced-order representation of the system, based on the information available in the sampled data. The corresponding time coefficients are also computed to reconstruct the original data. Training a neural-network model to predict the time coefficients of a limited number of active modes is simpler as the decomposition takes care of extracting the most important features from the data, as shown in **Paper 3**. Furthermore, this approach is less computationally expensive.

Alternatively, the same data can be used to derive a model of the system dynamics, by projecting the variables into a different manifold. For instance, Koopman theory allows for the representation of nonlinear dynamics in terms of an infinite-dimensional linear operator. In practice, the system dynamics are projected onto a finite-dimensional space in which the dynamics are approximately linear. Once this operator is known, the future state of the system is computed by applying the operator on the current system state, without the need to train neural networks for the predictions. These dimensionality-reduction approaches have been subsequently modified in order to model the non-linear interactions in the system dynamics. In particular, **Paper 2** and **Paper 3** analyze the performance of a Koopman framework in which non-linearities are modelled as external forcing (Khodkar & Hassanzadeh 2021, also described in section 3.2). One of the main advantages of the Koopman framework is that its theoretical foundation can be connected to classical approaches for the analysis of dynamical systems (Brunton *et al.* 2022).

It is worth noting that a reduced-order representation of the data can also be obtained by training an autoencoder network (Kramer 1991). This network architecture consists of two parts, an *encoder* and a *decoder*. The encoder maps the high-dimensional inputs into a low-dimensional *latent space*. The decoder reconstructs the input starting from the low-order representation. Note that the learnt representations are inherently non-linear, hence potentially more than linear dimensionality-reduction techniques (*e.g.* POD). Autoencoders can be combined with recurrent neural networks: the encoder acts as a learnt feature extractor and the recurrent network is trained to predict temporal sequences in the latent space. The two networks are trained at the same time, in order to optimize the reduced-order representation for the sequence prediction in the latent space.

The prediction of sequential data is relevant in several physical applications, but also in natural language processing (NLP). Several deep-learning models have been developed for translation and text generation over the years. The first sequence-to-sequence models for translation were based on feed-forward recurrent neural networks, but they could only be used on fixed-length sequences. This limitation was removed with the introduction of models based on autoencoders and recurrent neural networks (Cho *et al.* 2014), however, the input sentence is still encoded in a fixed-length vector and this can become a limiting factor when long sentences have to be translated. To address this issue, an attention mechanism was added to the models (Bahdanau *et al.* 2014). Given the recurrent nature of these models, the training of the network is difficult to parallelize and the scalability with respect to the size of the training dataset is negatively affected. For this reason, the research in the latest years focussed on transformers (Vaswani *et al.* 2017). These networks scale much better than the previous models since they are based only on attention mechanisms. While we did not test this architecture for the prediction of time series in fluid mechanics, we mention it for completeness, given its importance in machine learning

applications. This architecture can be adapted to perform spatial predictions and reconstructions, as detailed in chapter 4.

3.1. Recurrent neural networks

Recurrent neural networks are neural networks that are specifically designed to learn from sequential data. They are equipped with feedback connections that are able to store representations of the latest inputs $\boldsymbol{\chi}_t$ in their internal state $\mathbf{h}_t$ and use this information to compute the output $\boldsymbol{\zeta}_t$. Recurrent neural networks can be described using the following equations:

$$\begin{cases} \mathbf{h}_t = f_{\mathbf{h}}(\boldsymbol{\chi}_t, \mathbf{h}_{t-1}, \boldsymbol{\zeta}_{t-1}; \theta_{\mathbf{h}}) \\ \boldsymbol{\zeta}_t = f_{\boldsymbol{\zeta}}(\mathbf{h}_t; \theta_{\boldsymbol{\zeta}}), \end{cases} \tag{3.1}$$

where $f_{\mathbf{h}}$ and $f_{\boldsymbol{\zeta}}$ can be non-linear functions if they include activation functions such as hyperbolic tangent tanh or the rectified linear unit (ReLU, Nair & Hinton 2010). Here $\theta_{\mathbf{h}}$ and $\theta_{\boldsymbol{\zeta}}$ indicate the learnable parameters of the two functions, respectively. Different architectures are specified based on the definition of the internal state function $f_{\mathbf{h}}$ and output function $f_{\boldsymbol{\zeta}}$.

The feedback connections in the network architecture allow for predicting time series, however, the learning algorithm needs to be adapted to take the feedback connections into account. Neural networks are usually trained using backpropagation algorithms, and in order to train recurrent neural networks, backpropagation-through-time is used. The loss function to minimize is:

$$\mathcal{L} = \frac{1}{T} \sum_{i=1}^{T} l(\mathbf{y}_i, \boldsymbol{\zeta}_i), \tag{3.2}$$

where T is the length of the considered sequence and $l(\mathbf{y}_i, \boldsymbol{\zeta}_i)$ is the loss computed at the i-th timestep. In order to update the model parameters it is necessary to compute the gradient with respect to $\theta_{\mathbf{h}}$ and $\theta_{\boldsymbol{\zeta}}$. In particular, the first gradient can be expanded as:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta_{\mathbf{h}}} &= \frac{1}{T} \sum_{i=1}^{T} \frac{\partial l(\mathbf{y}_i, \boldsymbol{\zeta}_i)}{\partial \theta_{\mathbf{h}}} \\ &= \frac{1}{T} \sum_{i=1}^{T} \frac{\partial l(\mathbf{y}_i, \boldsymbol{\zeta}_i)}{\partial \boldsymbol{\zeta}_i} \frac{\partial f_{\boldsymbol{\zeta}}(\mathbf{h}_i; \theta_{\boldsymbol{\zeta}})}{\partial \mathbf{h}_i} \frac{\partial \mathbf{h}_i}{\partial \theta_{\mathbf{h}}}, \end{aligned} \tag{3.3}$$

here the term $\partial \mathbf{h}_i / \partial \theta_{\mathbf{h}}$ needs to be recursively computed backward in time, since $\mathbf{h}_i$ is a function of $\mathbf{h}_{i-1}$. Typically the backpropagation is not performed over the entire sequence, but only on the given number of timesteps. In this case, the algorithm is referred to as truncated back-propagation through time. Note that this choice affects the length of the sequence that is used by the RNN to make its next prediction.

Different recurrent architectures have been used to predict the temporal evolution of different dynamical systems with clear performance improvements

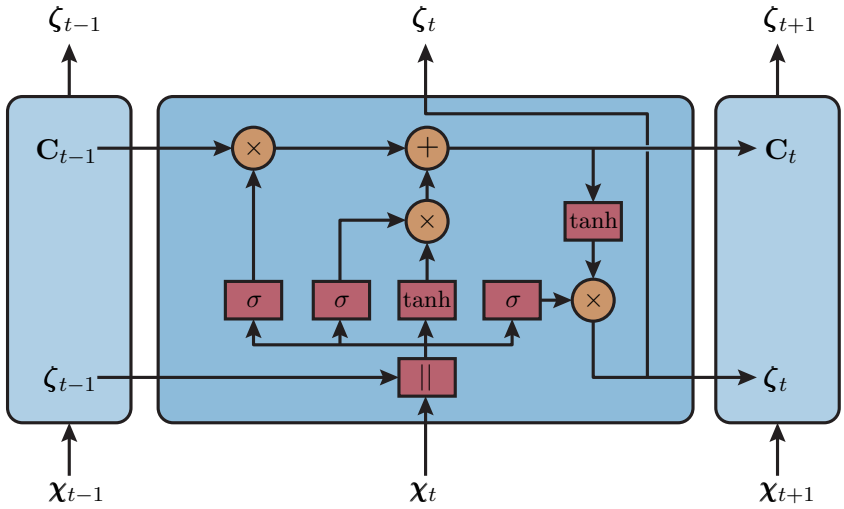


Figure 3.1: Long short-term memory network cell, adapted from Olah (2015).

over simple feed-forward neural networks, as shown in **Paper 1**. In particular, we focused on long short-term memory (LSTM) networks, which are designed to relieve the problem of vanishing gradients in vanilla recurrent neural networks. This is done by introducing different paths for the input and hidden state at the previous timestep to affect the output and the current hidden state, as shown in figure 3.1. These signal paths are called *gates*. The equations of the LSTM, as defined by Hochreiter & Schmidhuber (1997), are:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\boldsymbol{\chi}_t, \boldsymbol{\zeta}_{t-1}] + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i[\boldsymbol{\chi}_t, \boldsymbol{\zeta}_{t-1}] + \mathbf{b}_i) \\
\widetilde{\mathbf{C}}_t &= \tanh(\mathbf{W}_f[\boldsymbol{\chi}_t, \boldsymbol{\zeta}_{t-1}] + \mathbf{b}_f) \\
\mathbf{C}_t &= \mathbf{f}_t \otimes \mathbf{C}_{t-1} + \mathbf{i}_t \otimes \widetilde{\mathbf{C}}_t \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\boldsymbol{\chi}_t, \boldsymbol{\zeta}_{t-1}] + \mathbf{b}_o) \\
\boldsymbol{\zeta}_t &= \mathbf{o}_t \otimes \tanh(\mathbf{C}_{t-1}),
\end{aligned} \tag{3.4}$$

where $\otimes$ indicates the Hadamard product and σ is a sigmoid activation function. Note that $\mathbf{i}_t$, $\mathbf{o}_t$ and $\mathbf{f}_t$ represent the *input*, *output* and *forget* gates, respectively. The output $\boldsymbol{\zeta}_t$ corresponds to the hidden state $\mathbf{h}_t$, and $\mathbf{C}_t$ is the LSTM *cell state*. In this network architecture, both the cell state and the hidden state (or output) from the previous timestep are used to compute the new LSTM states.

Our approach to recurrent-neural-networks training is to provide a sequence of length p $(\boldsymbol{\chi}_0, \ldots, \boldsymbol{\chi}_{p-1})$ to the network to compute the following data point $\boldsymbol{\zeta}_p$. Note that $\boldsymbol{\zeta}_p = \boldsymbol{\chi}_p$ if the prediction target is the future state of the system described by $\boldsymbol{\chi}$, however, the output can also be a different variable if we aim to predict a temporal signal based on another one, for example, the velocity within the flow field, using wall-pressure measurements as input. Depending on the application, the inference procedure might differ from the one used for training. At inference time, after the first prediction step, the ground-truth value for $\boldsymbol{\chi}_p$ is not available, hence the predicted value $\widehat{\boldsymbol{\chi}}_p$ from the previous evaluation of the recurrent network needs to be used as part of the input. With this autoregressive approach, an error whose magnitude depends on the prediction error during the training is introduced. Such an error can play an important role in dynamical systems that exhibit chaotic behaviour. In these systems, a small perturbation of the system state can lead, over time, to a completely different trajectory in the state space. The accumulation of errors due to sequential evaluations of the network can then determine the prediction of a different trajectory.

In order to address this issue, it is also possible to modify the training to mimic the inference procedure. In this case, n_p successive predictions are performed for each training step: given the initial ground-truth temporal sequence of length p, the network evaluation yields $\widehat{\boldsymbol{\chi}}_p$. This value is concatenated with the temporal sequence and the last p data points are used to perform the prediction $\widehat{\boldsymbol{\chi}}_{p+1}$. This procedure is repeated n_p times and it increases the robustness of the model to the uncertainty in the input sequence, as shown in **Paper 2**. This approach has some affinity with training a neural-network model with *noisy* data. Noise can be used as means to train models that are robust with respect to data inaccuracy or uncertainty. The main difference is that the performance of the models is usually degraded when noisy data are used. In this case, the higher medium-term accuracy is instead improved thanks to the similarity between the training and inference procedure. Note, however, that remedies like the one just described can only delay the inevitable increase of the error in the predictions obtained with autoregressive models.

3.2. Non-linear dimensionality reduction: Koopman non-linear forcing

Koopman-operator theory was first introduced by Koopman (1931). We consider a time-discrete, non-linear dynamical system described by the operator $\mathcal{F}$:

$$\mathbf{x}_{t+1} = \mathcal{F}(\mathbf{x}_t), \tag{3.5}$$

where $\mathbf{x}$ is defined in the state space $\mathcal{M} \subseteq \mathbb{R}^n$. We introduce a linear, infinite-dimensional operator $\mathcal{K}$, called *Koopman* operator, which enables describing the system dynamics. The Koopman operator does not act directly on the state variables $\mathbf{x}$, instead, it is applied on functions of the state space called *observables* $\mathbf{g}(\mathbf{x})$. With this approach, it is possible to write the evolution of the system as:

$$\mathbf{g}(\mathbf{x}_{t+1}) = \mathcal{K}\mathbf{g}(\mathbf{x}_t) = \mathbf{g}(\mathcal{F}(\mathbf{x}_t)). \tag{3.6}$$

The formulation of the theory in terms of system observables can be used to define a data-driven approach to the linearization of a dynamical system.

Since the Koopman operator is infinite-dimensional, it is necessary to approximate it with a suitable finite-dimensional representation. The first step in this direction can be performed by applying a modal decomposition to the operator, as proposed by Mezić (2005). The resulting infinite sum of *Koopman modes* does represent exactly the operator. Alternatively, it is possible to approximate the operator with a truncated sum of relevant modes. Dynamic mode decomposition (DMD) was first proposed by Schmid (2010) as a means to identify fluid structures based on snapshots sampled from the flow. The connection between DMD and the Koopman operator was first reported by Rowley *et al.* (2009), describing how DMD represents a data-driven algorithm to compute the Koopman modes. The DMD algorithm provides the linear operator $\mathbf{A}$ that best approximates the temporal evolution of the system state:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t. \tag{3.7}$$

Crucially, not only DMD provides a set of (non-orthogonal) modes, but also a linear model that describes the evolution of the modes over time, which can be used to perform predictions in time.

Several extensions have been proposed in an effort to model the nonlinear dynamics, both for the Koopman operator and DMD. Early attempts (Carleman 1932) investigated the possibility to define the observables in Koopman theory as the state variables $\mathbf{x}$ and their higher powers $\mathbf{x}^n$. Note, however, that the dynamics of the higher powers of $\mathbf{x}$ contain even higher powers of $\mathbf{x}$. This recurrence leads to an infinite-dimensional system, preventing the construction of a finite-dimensional representation of the operator. Furthermore, the resulting infinite-dimensional system is not equivalent to the initial nonlinear system, as the former admits solutions that are not solutions of the finite system (Steeb 1981). In DMD, linear measurements can be complemented with nonlinear functions of the measurements (extended DMD, Williams *et al.* 2015). Alternatively, nonlinear effects can be modelled as an external forcing (Koopman nonlinear forcing or KNF, Khodkar & Hassanzadeh 2021), as shown in **Paper 2** and **Paper 3**. In this case, the resulting model is:

$$\mathbf{x}_{m+1} = \mathbf{A}\mathbf{x}_m + \mathbf{B}\mathbf{f}_m, \tag{3.8}$$

where $\mathbf{f}_m$ represents a library of candidate nonlinear functions, such as polynomials or trigonometric functions. The identification of the nonlinear dynamics in the data is performed with the SINDy algorithm (Brunton *et al.* 2016). Note that the computation of $\mathbf{A}$ is based on the Hankel-DMD algorithm proposed in Brunton *et al.* (2017), as detailed in **Paper 2**.

The main difference with respect to RNNs is that all the dataset is processed at once when computing the model matrices. The size of the dataset is limited by the amount of data that we are able to process at the same time. The computation of the matrices in KNF is more computationally-efficient

since neural networks process the entire dataset several times during training. Different tools work better in different scenarios: as detailed in **Paper 2**, the KNF framework performs better than RNNs on data from the nine-equations model. By contrast, LSTM networks perform better in the prediction of the temporal coefficients of POD modes in a channel flow (see **Paper 3**).

Chapter 4

Space: convolutional networks and transformers

Since the early works of Taylor (1935), spatial correlations have been used as a tool to develop theories and models of turbulence. Space (and time) correlations have been studied first in experiments and later in numerical simulations (Biferale *et al.* 2011). Kim *et al.* (1987) extensively compared the spatial correlations of the different velocity components with experimental data to validate their first DNS. We define the spatial correlation of two quantities as:

$$\mathcal{R}_{ij}(\mathbf{x}, \mathbf{r}, t) = \langle u_i(\mathbf{x} + \mathbf{r}, t) u_j(\mathbf{x}, t) \rangle, \tag{4.1}$$

where $\mathbf{r} = (\delta x, \delta y, \delta z)$ indicates the spatial distance for which the correlation is being computed. Note that the correlations do not depend on the position $\mathbf{x}$ in the homogeneous directions. The correlation of the different quantities sharply decreases when $\|\mathbf{r}\|$ increases, as shown in several research works (including Nakagawa & Nezu 1981, for open-channel flows). By leveraging the spatial correlation between two variables, it is possible to create a model that predicts one variable based on the knowledge of the spatial distribution of the other.

Our objective is to carry out *non-intrusive* sensing of the flow by predicting the flow field using measurements of different flow quantities at the wall as inputs. This is motivated by the fact that these wall measurements can be obtained without perturbing the flow, whereas placing probes within the flow can be more complicated and, in full-scale high-Reynolds-number applications, even technologically unfeasible. When designing the neural-network model to perform non-intrusive sensing, the correlation function between input and output variables (*e.g.* the wall-shear stress and the velocity away from the wall, Marusic & Heuer 2007) suggests the use of neural-network layers that can extract localized features of the flow. At the same time, we consider flows that are homogeneous at least in one direction, hence a model that embeds the translational equivariance of the input and output fields is desirable. Both requirements can be satisfied by using convolutional layers, as detailed in **Paper 5**. These layers represent the fundamental building block of convolutional neural networks (CNNs, LeCun *et al.* 1989), that have also been used in other fluid-dynamics applications such as the design of closure models in large-eddy simulations (LESs, Beck *et al.* 2019) or flow control (Park & Choi 2020).

The use of convolution-based neural network models for non-intrusive sensing was first tested on the fields sampled from an open-channel-flow simulation. The velocity fluctuations at a given wall-normal location are predicted based on the streamwise and spanwise components of the wall-shear stress and the wall pressure. The prediction of two-dimensional fluctuation velocity fields from wall-measurement fields of the same dimensionality motivates the use of fully-convolutional-network (FCN) architectures. The main difference between the FCN architecture and CNN architecture is that the former does not include any fully-connected layer. Depending on the predicted output, we define two different architectures, named FCN and FCN-POD. In the first case, the network directly predicts the output field, whereas in the second case, we first apply proper orthogonal decomposition (POD) on the training dataset to obtain a set of orthonormal basis functions $\boldsymbol{\phi}_i(\boldsymbol{x})$, so that the flow field can be represented as a linear combination of these functions:

$$\boldsymbol{u}(\boldsymbol{x},t) \approx \sum_{i=1}^{N_m} \psi_i(t)\sigma_i\boldsymbol{\phi}_i(\boldsymbol{x}), \tag{4.2}$$

where N_m is the total number of POD modes and σ_i is the root-squared energy contribution to the i-th mode. The FCN-POD model is trained to predict the temporal POD coefficients $\psi_i(t)$. On the one hand, the initial decomposition of the flow fields allows the FCN-POD model to leverage the information stored in the spatial modes to perform the prediction. On the other hand, the number of POD modes required to reconstruct all the energy in the flow field is very large and many modes have a relatively small energy contribution. To address this, one approach is to reconstruct only a portion of the overall energy by using a limited number of modes. Alternatively, the flow field can be divided into smaller subdomains, which share the same orthogonal basis decomposition but have different temporal coefficients. In this second approach, the energy associated with the structures larger than the subdomain is included in the first, most-energetic mode. These limitations of the spatial representation can negatively affect the FCN-POD overall performance, even if the temporal coefficients are correctly predicted, as shown in **Paper 5**. While the FCN model does not suffer from these drawbacks, reconstructing the flow field based on the information at the wall becomes progressively more difficult as the distance between the flow field and the wall increases. Furthermore, the use of the FCN architecture allows the strict enforcement of periodic boundary conditions, applying periodic padding to the input fields in the corresponding directions. This is not possible with the FCN-POD model. A more comprehensive comparison of the performance of the two architectures is reported in **Paper 5**.

The FCN architecture has been tested both for predictions in an open channel flow and turbulent boundary layer. In the latter case, the output flow field is predicted using different input fields. Based on the input data, three types of predictions are defined: *type-I* predictions consider the streamwise and spanwise wall-shear-stress components, as well as the wall-pressure fields as

inputs. In *type-II* predictions, the streamwise wall-shear stress is substituted with the heat flux field of a passive scalar. In *type-III* predictions, only the heat-flux field is considered as input. The objective of the study described in **Paper 7** is to test progressively more difficult scenarios, using fewer wall inputs, which are also less correlated with the target field. *Type-III* predictions are designed to replicate an experimental scenario in which only the wall heat-flux field is sampled. Furthermore, it should be noted that the predictions in the turbulent boundary layer are more challenging than the ones performed in the open channel because the boundary layer is a spatially-developing flow.

The FCN architecture has been tested also for the opposite task, which consists of predicting the wall-shear stress based on the flow field at a given distance from the wall, as reported in **Paper 8**. This task is more challenging for the FCN network than non-intrusive sensing since convolution operations are equivalent to filters applied to the input data. Therefore, it is difficult for the network to predict the high-frequency content in the target fields at the wall.

Note that the use of convolutional layers has another important limitation, independently from the learning task. In fact, these layers assume a fixed spacing between the input points that are convolved. This limits the possibility to use a previously-trained neural network on other data (also known as *transfer learning*, Pan & Yang 2009). The training of a neural network can be effectively sped up by initializing it with the learnable parameters of a network trained on another flow-field dataset provided that the physical spacing (*e.g.* Δx^+ and Δz^+) between the input points is approximately the same. Within these constraints, transfer learning has been applied to the training at different wall-normal locations (**Paper 4**) and different Reynolds numbers (**Paper 5**).

Other deep-learning models have been tested for spatial predictions and reconstructions; for instance, the term *generative adversarial networks* (GANs, Goodfellow *et al.* 2014) indicates two networks that are trained concurrently: a *generative* network (or *generator*) predicts the output and a *discriminative* network needs to determine whether the output belongs to the generator distribution or the reference (ground-truth) distribution. When two convolution-based networks are trained in this framework, the generator network provides statistically-accurate results in a variety of tasks such as super-resolution (Güemes *et al.* 2021) and gap reconstructions (Buzzicotti *et al.* 2021).

In all our spatial-prediction studies, the assessment of the prediction performance is based on the instantaneous predictions and the accuracy in reproducing the statistical moments of the velocity field, as detailed in section 4.1. Furthermore, the energy distributions of the DNS and predicted fields are compared using power-spectral densities of the velocity components, which represent the spectral decompositions of the corresponding autocorrelation functions, as described in section 4.2. Finally, convolutional networks are not the only architecture that can be used for spatial predictions. Thanks to their scalability, transformers have started to be trained for computer-vision tasks like

image recognition and segmentation. Section 4.3 describes how the original architecture, designed for signal transduction, can be modified to work on two-dimensional flow fields.

4.1. Mean-squared error and the statistics

The training of the neural network for the prediction and reconstruction of spatial fields is based on the minimization of the mean-squared error between the prediction and the target. Depending on the approach, the target can be the flow field (in the FCN case), or the temporal coefficients obtained by the projection of the field on an orthonormal basis of spatial functions (in the FCN-POD case). There are two important implications related to this choice of the loss function: the first is related to the information reconstructed by the network when the mean-squared error of the predictions is minimized, and the second is related to the performance of the model when it is analyzed from a statistical perspective.

Implication I: when the mean-squared error is used a loss function, the network learns first the large-scale features, and then progressively optimizes the tunable parameters to minimize the error at finer and finer scales (Xu *et al.* 2019). This depends on the definition of the MSE: if we are computing the error in the velocity fields, this quantity is equivalent to the energy missing from the predicted fields. Larger scales contain more energy and their reconstruction has a larger contribution to the overall error. Furthermore, the error is averaged in the different spatial directions. This affects the quality of the predictions in spatially-developing flows if no weighting is introduced. As shown in figure 4.1, the error is space-dependent: it is higher at the extrema and lower at the center of the field. More accurate predictions can be obtained if a smaller streamwise-location (and corresponding Reynolds number) range is considered.

Implication II: even if the trained networks are optimized to reconstruct the instantaneous behaviour of the flow, it is desirable that the statistical properties of the flow are correctly predicted for statistically-stationary flows. This could be addressed by adding the individual statistical moments to the loss function, however, this approach has three main drawbacks: first, the inclusion of a specific statistical quantity does not guarantee a higher accuracy for the other statistics. Second, the loss is a scalar quantity, so the relative importance of the different terms has to be tuned by performing several, independent training runs. Third, the loss is computed on every mini-batch in order to update the model parameters, hence the statistical quantities in the loss need to be computed as well. The typical size of the batch does not allow the computation of converged statistics, preventing the comparison with the flow statistics computed on the entire training dataset. This problem becomes more relevant, for progressively higher statistical moments included in the loss definition. For these reasons, our approach for spatial predictions is to focus on the instantaneous accuracy of the predictions, while the statistical accuracy is only assessed *a-posteriori* on the test dataset. In our works, we consider the prediction of the velocity

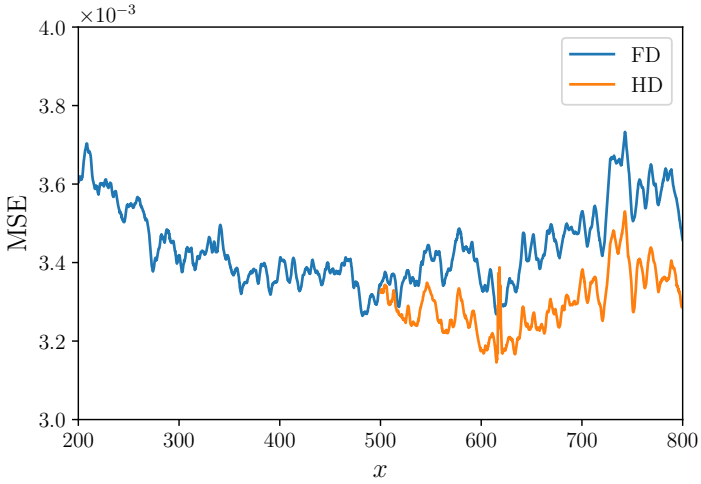


Figure 4.1: MSE in the predictions as a function of the streamwise location for *type-III* predictions at $y^+ = 30$. FD indicates the *full-domain* samples (larger streamwise range), while HD indicates *half-domain* samples. The curve represents the average over the spanwise direction and the samples in the test dataset. From Guastoni *et al.* (2022)

fluctuations as the target for the neural-network models, hence we calculate the error in the fluctuation intensities for the different velocity components, defined as:

$$E^+_{\mathrm{RMS}}(u) = \frac{\left| u^+_{\mathrm{RMS,Pred}} - u^+_{\mathrm{RMS,DNS}} \right|}{u^+_{\mathrm{RMS,DNS}}}, \tag{4.3}$$

for the streamwise component, and similarly for the other two components. Here, the subscripts 'DNS' and 'Pred' refer to the reference and predicted profiles, respectively.

4.2. Spectra

Further insight into the prediction accuracy can be gained by looking at the energy distribution at different lengthscales. To this end, the pre-multiplied power-spectral densities of the different quantities can be computed and analyzed. The power spectrum describes the energy associated with each Fourier mode. For the open channel, since the flow is homogeneous in both wall-parallel directions, it is defined as:

$$\mathcal{P}_{ij}(k_x, k_z) = \mathcal{F}(u_i)\mathcal{F}^*(u_j), \tag{4.4}$$

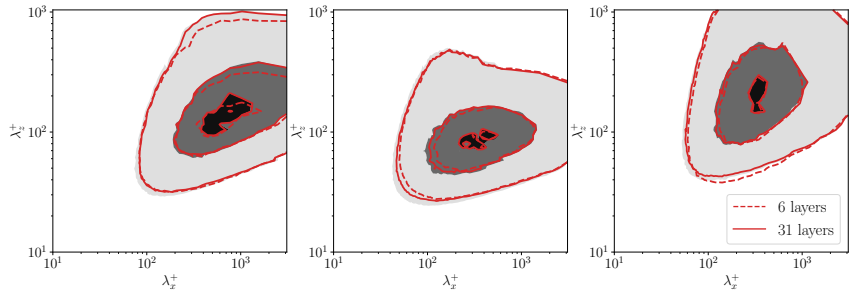


Figure 4.2: Pre-multiplied two-dimensional power-spectral densities for *type-I* predictions at $y^+ = 30$. The three columns represent $k_z k_x \phi_{uu}$ (left), $k_z k_x \phi_{vv}$ (center), $k_z k_x \phi_{ww}$ (right). The contour levels contain 10%, 50% and 90% of the maximum DNS power-spectral density. Shaded contours refer to the reference data, while contour lines refer to the FCN with 6 layers (dashed) and with 31 layers (solid), respectively. Adapted from **Paper 7**.

where $\mathcal{F}(\cdot)$ is the two-dimensional Fourier transform and * indicates the complex conjugate. The power-spectral density is defined as:

$$\begin{aligned} \phi_{ij}(k_x, k_z) &= \frac{\mathcal{P}_{ij}}{\mathrm{d}k_x \mathrm{d}k_z} \quad \text{or} \\ \psi_{ij}(\lambda_x, \lambda_z) &= \frac{\mathcal{P}_{ij}}{\mathrm{d}\lambda_x \mathrm{d}\lambda_z} = \phi_{ij}\left(\frac{k_x^2 k_z^2}{2\pi}\right), \end{aligned} \tag{4.5}$$

since $\mathrm{d}\lambda_i = 2\pi/\mathrm{d}k_i$. Note that the power-spectral density is related to the correlation function in equation (4.1), since ϕ_{ij} is the Fourier transform of $\mathcal{R}_{ij}$. Note that in spatially-developing-flow simulations, only the spanwise direction is homogeneous. The power spectral density is then computed as a function of λ_z and the period λ_t using the Welch method (Welch 1967) to approximate the Fourier transform of a non-periodic signal. Note that λ_t and λ_x can be related by the Taylor's frozen turbulence hypothesis (Taylor 1935).

In non-intrusive sensing applications, the amount of energy in the predicted fields is lower than that in the ground-truth fields, and the difference becomes more pronounced the farther the target field is from the wall. The pre-multiplied spectra provide information about how the predicted energy is distributed in the wavelength space. The power-spectral density of the predictions is related to the FCN architecture. Figure 4.2 compares the predictions of two networks that have approximately the same number of learnable parameters, but they are organized in a different number of convolutional layers. The higher is the depth of the network, the higher is the compositional capability of the FCN, *i.e.* the ability to combine simple features identified by the previous layers into more and more complex ones. This translates into better predictions of the larger-scale features.

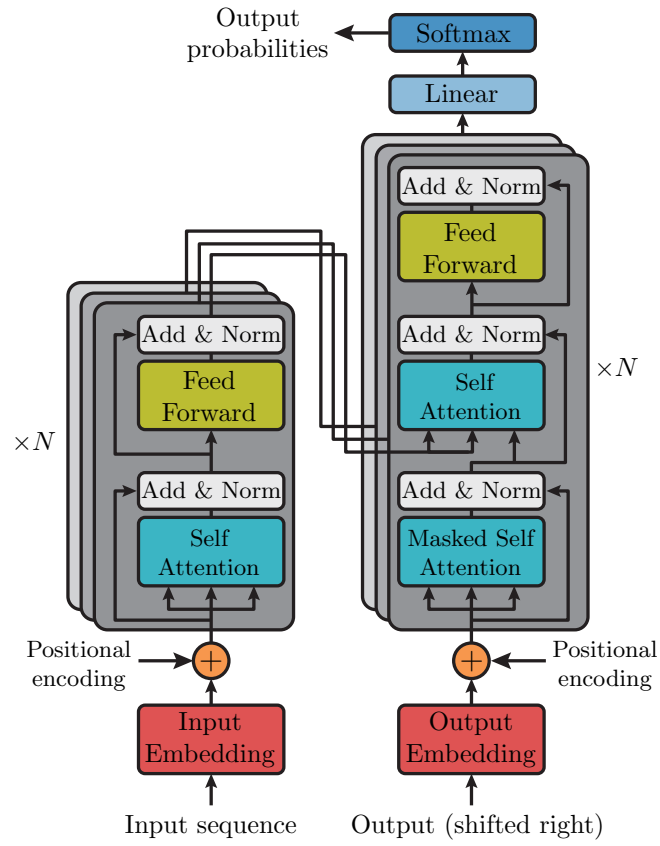


Figure 4.3: Transformer architecture, adapted from Vaswani *et al.* (2017).

4.3. Transformers for spatial predictions

Transformers were first introduced for NLP, and since then they have found application in both computer-vision tasks and time series forecasting (Lim & Zohren 2021). The transformer architecture was initially designed for sequence transduction (*e.g.* language translation), hence it requires a source sequence and a target sequence. The architecture consists of two parts, an *encoder* and a *decoder*, as shown in figure 4.3, which is reminiscent of the autoencoder neural-network architecture. The encoder receives as input the source sequence, whereas the decoder receives the target sequence as input, shifted to the right with respect to the output. In the figure, FFN indicates a feed-forward neural network, consisting of a dense layer with a ReLU activation function, followed by a second dense layer.

Both the encoder and the decoder include layers devoted to the computation of attention. The implementation of attention mechanisms without resorting to convolutional or recurrent layers allows the transformer model to scale better, enabling the training on much larger datasets. Attention in transformers is defined as a scaled dot-product:

$$A(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}}\right)\mathbf{V}. \tag{4.6}$$

Here $\mathbf{Q} \in \mathbb{R}^{n_i \times n}$, $\mathbf{K} \in \mathbb{R}^{n_i \times n}$ and $\mathbf{V} \in \mathbb{R}^{n_i \times n_v}$ are termed *queries*, *keys* and *values*, respectively. n is the query and key dimension, while n_v is the value dimension. n_i indicates the number of input elements. Depending on the inputs, different types of attention can be defined:

- **Encoder self-attention:** $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are computed based on the input sequence.
- **Decoder self-attention:** it takes the target sequence as input. In this case, the attention computation is masked, meaning that position i can be influenced only by positions $j \leq i$.
- **Encoder/decoder attention:** Both the encoded ($\mathbf{K}$, $\mathbf{V}$) and the decoded sequences ($\mathbf{Q}$) are considered.

Both the attention and the FFN layers are equipped with residual connections. Layer normalization is performed after the addition of the original and processed input of the layer.

After their introduction for language translation, transformers were adapted to image-recognition tasks with minimal modifications. The vision transformer (ViT, Dosovitskiy *et al.* 2020) has two main implementation differences with respect to the original transformer model. The first difference is in the way the input is processed before being passed to the encoder (also called *tokenization*): the input image is divided into two-dimensional patches, which are then flattened and linearly projected into a D-dimensional vector, before being fed to the first attention layer. Second, since there is no target sequence, the decoder part of the network is removed, similarly to the BERT model (Devlin *et al.* 2018) used in language modelling. Alternatively, it is possible to use only the decoder part of the original transformer model, as in generative pre-trained transformers (GPTs, Radford & Narasimhan 2018). Furthermore, additional small modifications are introduced, for instance, layer normalization is performed before the multi-head attention and feed-forward layers, and residual connections are added without layer normalization.

4.3.1. *Use case: generation of wall-parallel turbulent flow fields using GPT models*[1]

Large language models (LLM) are very large transformer models that have shown impressive results in the generation of text. The network architecture

[1]Work in collaboration with Federico Baldassarre (main contributor), Ricardo Vinuesa and Hossein Azizpour

is domain agnostic and it can also be used for the generation of images (Chen *et al.* 2020). A similar approach can find application in fluid-dynamics problems, for instance, in the generation of turbulent velocity fields. This can help reduce the computational cost of a spatially-evolving flow simulation by providing suitable inlet conditions, as reported by Yousif *et al.* (2022). Here we describe a preliminary exploration of the possibility to generate two-dimensional wall-parallel flow fields from an open-channel flow simulation. Before generating the new fields, a reduced-order representation is learned using a vector-quantized variational autoencoder (VQ-VAE, van den Oord *et al.* 2017). Hierarchical autoencoders (Fukami *et al.* 2020) and β-variational autoencoders (Eivazi *et al.* 2022) have also been used to learn reduced-order representations of flow fields. The use of discrete codes instead of a high-dimensional continuous latent space is more computationally-efficient and avoids the posterior collapse (Lucas *et al.* 2019). Note that, in our approach, we do not learn a representation of the entire flow field at once, instead we divide the input image in patches of size 8×8, in a way that reminds ViT input-processing. In order to produce new samples, we train an auto-regressive GPT model (implemented as GPT-NeoX, Black *et al.* 2022) starting from the latent-space representation. The transformer learns the conditional probability of the codes in the quantized latent space, given the previous ones in raster order (left-to-right, top-to-bottom). After training, three different parameters can be tuned for sampling:

- **Temperature** T**:** higher temperature implies a higher variety in the samples, lower temperature generates fields that are closer to the ones used for training.
- **Top** k: the generator picks among the top N_k tokens with the highest probability. Note that $N_k \leq \overline{N}_k$, where $\overline{N}_k$ represents the number of tokens used for latent-space representation. In our experiments $\overline{N}_k = 256$.
- **Top** p: the generator picks among the tokens whose probabilities add up to N_p ($N_p \leq 1$).

We analyze the turbulent fields sampled using $T = 2.0$, $N_k = 16$ and $N_p = 0.8$. The fields sampled from a DNS simulation and generated by the network can be qualitatively compared in figure 4.4. Even if the transformer is able to generate fields that are visually accurate and that do not show visual artifacts, the amount of energy in the individual components has a different distribution when compared to the DNS fields. A more quantitative analysis can be performed by observing the spectra in figure 4.5: the energy in the streamwise component is distributed over a wider range of wavelengths, both in the streamwise and spanwise directions. The wall-normal component shows the worse energy reconstruction, with all the energy concentrated within a limited range of scales, whereas the power-spectral density in the spanwise component is in good agreement with the one of the DNS data.

On the one hand, these first results highlight the potential of these architectures to generate synthetic turbulence data, on the other hand, they confirm

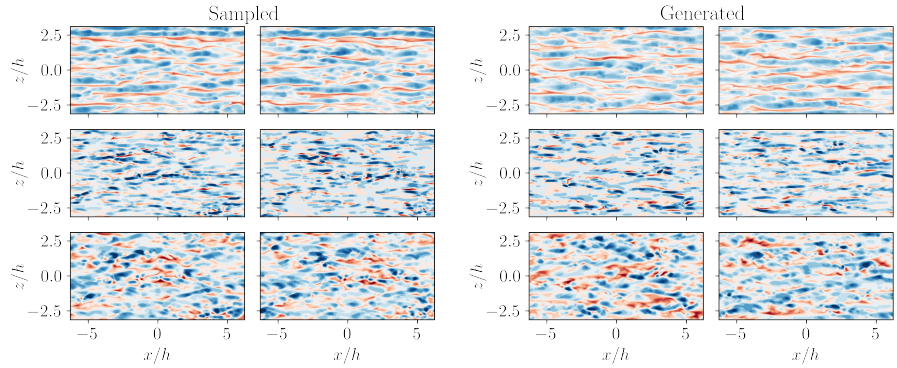


Figure 4.4: Comparison of the flow fields obtained by sampling a DNS (first two columns from the left) and the ones generated by the transformer (last two columns from the left). The rows represent the streamwise (top), wall-normal (middle) and spanwise (bottom) velocity components.

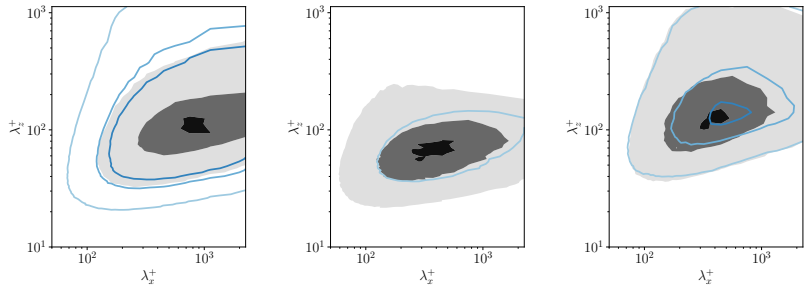


Figure 4.5: Pre-multiplied two-dimensional power-spectral densities of the generated fields. The three columns represent $k_z k_x \phi_{uu}$ (left), $k_z k_x \phi_{vv}$ (center), $k_z k_x \phi_{ww}$ (right). The progressively-darker contour levels contain 10%, 50% and 90% of the maximum DNS power-spectral density. Shaded contours refer to the reference data, while contour lines refer to the generated data.

the importance of a comprehensive quantitative analysis in order to verify that the physical features of the flow are correctly reproduced.

Chapter 5

Control: deep reinforcement learning

The possibility to manipulate a fluid flow to modify its physical characteristics has been the objective of intense research efforts for a very long time. Brunton & Noack (2015) discuss the use of fletching on the backs of arrows as an early example of flow-control device. Fletching is a passive-control system as it does not require external energy for its operation. By contrast, active systems rely on powered actuators to control the flow. Passive-control systems are simpler to deploy, however, active-control strategies are typically more efficient and we focus mostly on this latter category. Active-flow-control solutions can be further divided into *open-loop* or *closed-loop* strategies. The former operate independently from the instantaneous system dynamics, based on predetermined control patterns, while the latter adjust the actuation according to the controlled-system state. In order to design closed-loop active-control systems, sensors need to be positioned to measure the relevant flow properties. A system model might be required to characterize the flow, based on the limited information provided by the sensors. Furthermore, the physical mechanisms that we aim to manipulate need to be identified and the actuators to perform such a manipulation need to be designed. A large number of research works have focused on one or more of these aspects, however, several modern engineering applications are characterized by turbulent flows and the possibility to control such flows is hindered by their stochastic and non-linear nature.

Data-driven methods like genetic programming (GP, Banzhaf *et al.* 1998) and deep reinforcement learning (DRL, Sutton & Barto 2018) can be used to automatize the steps of the control design process, such as the sensors placement (Paris *et al.* 2021) and the discovery of control mechanisms (Rabault *et al.* 2019; Gautier *et al.* 2015), both in numerical and experimental settings (Benard *et al.* 2016). **Paper 9** focuses on the use of DRL to discover novel drag-reduction strategies in turbulent flows; the reinforcement-learning (RL) framework and algorithms are introduced in sections 5.1 and 5.2, respectively.

The application of RL and DRL in fluid mechanics is not as developed as traditional supervised learning. In fact, only recently researchers have started utilizing these algorithms to solve flow control problems. The applications range from drag reduction (Rabault *et al.* 2019) to mesh optimization (Lorsung & Barati Farimani 2023), from the maximization of the efficiency of agents

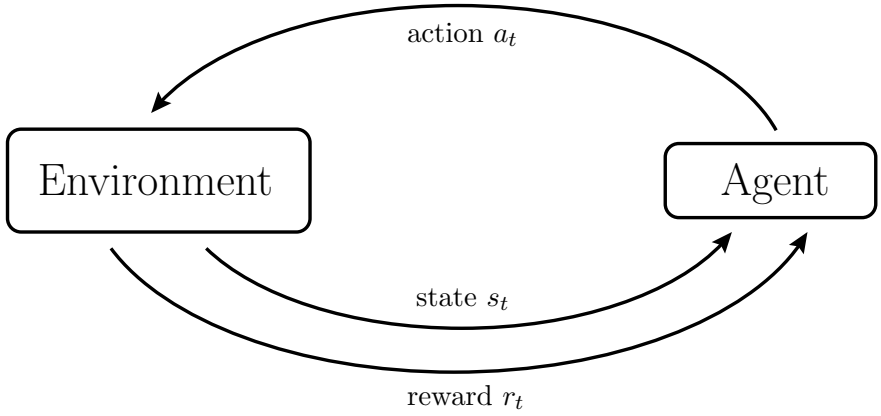


Figure 5.1: The reinforcement-learning framework

swimming in a turbulent flow (Biferale *et al.* 2019; Verma *et al.* 2018) to turbulence modelling (Novati *et al.* 2021; Kurz *et al.* 2023).

The first application of DRL for drag reduction is the work by Rabault *et al.* (2019), in which the flow around a cylinder is modified by means of two actuators, perpendicular to the main direction of the flow, with the objective to reduce the overall drag. The Reynolds number based on the mean streamwise velocity and the cylinder diameter is $Re = 100$. Despite its apparent simplicity, this two-dimensional flow case has become a benchmark for other DRL control studies (Tang *et al.* 2020; Ren *et al.* 2021), including the work by Varela *et al.* (2022), which highlights the non-trivial dependency of the optimal drag-reduction policy on the Reynolds number, as well as the possibility to identify different Reynolds-dependent strategies using DRL. The problem of reducing the drag of the flow around a cylinder has a fundamental character and its solution provides phenomenological and implementation insights that can be used for other flow cases. Given the interdisciplinary nature of flow control, the availability of such benchmarks helps extend the use of novel control design techniques like DRL by providing a quantitative performance comparison with other control solutions. In our work, the proposed benchmark is an open channel flow, that shares many physical features with the standard channel flow, reference case for numerical turbulence research since the work by Kim *et al.* (1987).

5.1. Reinforcement learning fundamentals

Reinforcement learning is a mathematical framework introduced by Sutton (1988) which is used solve Markov decision processes (MDPs). In this framework, the learning is based on the interaction of an *agent* with an *environment*. The latter is described by its state $s_t \in \mathcal{S}$ at any given time t. At each step, the agent

performs an action $a_t \in \mathcal{A}$ in the environment, which evolves to a different state based on a given transition function $P(s_{t+1}|s_t, a_t)$. Then the agent receives an instantaneous reward $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$. The information exchanged between the environment and the agent is summarized in figure 5.1. A sequence of states and actions defines a trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$, even though the term *episode* can also be interchangably used to refer to a trajectory. The objective of reinforcement learning is to define a policy function $\pi : \mathcal{S} \to \mathcal{A}$, that maximizes the expected return $\mathbb{E}[R(\tau)]$, with $R(\tau)$ defined as:

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t. \tag{5.1}$$

Here $\gamma \leq 1$ is a discount factor to prioritize rewards that are closer in time, and T is the length of the trajectory τ. Typically, if T is finite $\gamma = 1$, otherwise $\gamma < 1$. The expected return is computed as an average over all possible trajectories. In the most general case, the transition function P and the policy function π are stochastic. Hence, in order to formulate an equation for the expected return, we first need to define the trajectory probability:

$$P(\tau|\pi) = P(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t), \tag{5.2}$$

where $P(s_0)$ is the probability distribution of the initial state. The expected return is then:

$$\mathbb{E}[R(\tau)] = \int_{\tau} P(\tau|\pi)R(\tau)\mathrm{d}\tau. \tag{5.3}$$

The expected return provides a *value* indication when conditioned on a specific state or state-action pair. In particular, we can define a state value function:

$$V^{\pi}(s) = \mathbb{E}[R(\tau)|s_0 = s], \tag{5.4}$$

as the expected return when the initial state is s and the agent acts according to a policy π. Similarly, the action-value function is defined as:

$$Q^{\pi}(s, a) = \mathbb{E}[R(\tau)|s_0 = s, a_0 = a]. \tag{5.5}$$

In this case, the first action that is taken is a and then the agent acts according to the policy π. Note that a does not have to be sampled from the policy π. The policy π^* that maximizes $Q(s, a)$ is called *optimal* policy. Importantly, by knowing the optimal Q-function $Q^*(s, a)$, it is possible to choose the optimal action for a given state:

$$a^*(s) = \arg\max_a Q^*(s, a). \tag{5.6}$$

For this, several RL algorithms include the approximation of the Q-function, as further detailed in section 5.2.

5.2. Deep-reinforcement-learning algorithms

Different DRL algorithms can be used to design the control policy. Here only *model-free* DRL algorithms are reviewed since current research and applications focus on these algorithms, however, there exist *model-based* algorithms as well. The latter category of algorithms leverages a model of the environment to choose the next action and also to analyze different possible scenarios and plan in advance the next sequence of actions. An existing environment model can be used or it can be learned during the agent training. Model-based algorithms have a higher sample efficiency than model-free algorithms, meaning that they require fewer interactions with the environment during training in order to reach a comparable level of performance. Note, however, that the policies learnt with these algorithms may take advantage of limitations of the environment model, providing a worse performance when deployed in the actual environment.

Model-free DRL algorithms can be classified based on the quantity that is being approximated during the agent training. A first group of algorithms is based on equation (5.6): the agent learns an approximation $Q(s, a|\theta^Q)$ of the optimal action-value function Q^*, where θ^Q are the learnable parameters. Then the actions are chosen with the maximization of the Q-function:

$$a(s) = \arg\max_a Q(s, a|\theta^Q). \tag{5.7}$$

One example of Q-learning is deep Q-networks (DQN, Mnih *et al.* 2013). A second group of algorithms learns directly a policy $\mu(s)$, parameterized with θ^μ. One example is the policy-gradient algorithm (Williams 1992). Typically, policy learning is performed *on-policy*, meaning that the policy updates are computed based on interactions collected using the most recent policy. By contrast, Q-learning methods are *off-policy* algorithms, and the optimization takes advantage of any interaction available, independently from the policy used to choose the action. This characteristic makes Q-learning algorithms more sample-efficient, however, they are typically less stable because the optimization is not targeting directly a function that computes the actions to perform, based on the current state. It is also possible to combine the two approaches: proximal policy optimization (PPO, Schulman *et al.* 2017) and deep deterministic policy gradient (DDPG, Lillicrap *et al.* 2015) are included in this category. The former learns a stochastic policy and it has been used for previous flow-control applications (Belus *et al.* 2019). We apply the latter to our turbulent-flow-control problem. DDPG is an off-policy algorithm, which concurrently learns both a policy function and a Q-function. For this, a *critic* neural network $Q(s, a|\theta^Q)$ and an *actor* neural network are trained $\mu(s|\theta^\mu)$. θ^Q and θ^μ represent the learnable parameters of the two networks, respectively. The training is performed according to algorithm 1. In this algorithm, a replay buffer $\mathcal{B}$ is used to store the transitions (s_i, a_i, r_i, s_{i+1}). The updates of the learnable parameters are performed on mini-batches randomly sampled from the replay buffer. Furthermore, duplicates of both the actor and critic network are used and we refer to them as *target networks*. Smaller updates are applied to the

learnable parameters of the target networks at each iteration. Using them to compute the targets y_i (see algorithm 1) stabilizes the learning process.

Algorithm 1: DDPG algorithm

Initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$.
Initialize replay buffer $\mathcal{B}$
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
for episode = 1, N **do**
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \text{clip}(\mu(s_t|\theta^\mu) + \epsilon, a_{\min}, a_{\max})$, with $\epsilon \sim \mathcal{N}$
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
    Randomly sample a minibatch of B transitions (s_i, a_i, r_i, s_{i+1}) from $\mathcal{B}$
    Compute targets:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$$

    Update critic by minimizing the loss:

$$L = \frac{1}{B}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{B}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \rho\theta^Q + (1-\rho)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \rho\theta^\mu + (1-\rho)\theta^{\mu'}$$

  end for
end for

5.3. Turbulent drag reduction as a reinforcement-learning problem

The algorithms described in section 5.2 can be applied to flow-control problems. For this, it is necessary to formulate them as reinforcement-learning problems, defining state observations, agent actions, and rewards. In **Paper 9**, we apply DRL to find strategies to reduce the turbulent drag in an open channel flow.

The agent computes the next action based on the observation of the current state of the environment. Typically, the observation is only a partial representation of the state; in our case the agent receives the velocity fluctuations in the streamwise and wall-normal directions at a given distance from the wall

($y^+ = 15$). Note that the agent knowledge of the environment is limited by the observation provided, hence this choice implicitly defines a prior on the features of the flow state that can be leveraged by the control system. As an example, a sensing plane farther away from the wall will reduce the capability of the agent to characterize and control the near-wall turbulent structures. Classical approaches rely on the observation of the velocity fluctuations (Choi *et al.* 1994), however other quantities can be sampled as well (*e.g.* the vorticity, Buzzicotti *et al.* 2020).

In our drag-reduction problem, a wall-normal velocity distribution is applied at the wall to control the flow. The actions are usually chosen within a prescribed range. It is important to highlight that the next action value can be significantly different from the current one and a sudden variation can cause numerical-stability problems. In order to prevent this issue, a smooth (linear or exponential) transition between the two action values can be implemented.

In **Paper 9** we define the reward as the percentage reduction of the wall-shear stress, averaged over the entire wall. The instantaneous reward signal may not be sufficiently informative because of the stochastic nature of turbulence. Spatial averaging helps filter out the high-frequency content of the reward signal. Furthermore, a temporal averaging based on a moving window can also be implemented, further regularizing the signal. In this case, the size of the averaging window is a hyperparameter of the reinforcement-learning problem. Note that the actuation frequency is an additional hyperparameter of the problem, as it imposes a lower bound to the timescale of the phenomena that can be controlled by the agent.

5.3.1. *Quadrant analysis*

Once a control strategy has been devised, different tools can be used to analyze the controlled flow. Here we describe *quadrant analysis*, which was used in **Paper 9** to compare the flow controlled with opposition control (Choi *et al.* 1994) and with a DRL policy. Quadrant analysis was first introduced by Wallace *et al.* (1972) to analyze experimental data and propose a categorization of the different contributions to the Reynolds shear stresses. The streamwise and wall-normal fluctuations are divided into four categories, based on the sign of the fluctuations: Q1 ($+u'$,$+v'$), Q2 ($-u'$,$+v'$), Q3 ($-u'$,$-v'$), and Q4 ($+u'$,$-v'$). Note that the fluctuations are not evenly-distributed among the quadrants in wall-bounded turbulence, as shown in figure 5.2. Q2 and Q4 events are associated with motions that can be observed in turbulent flows, denoted as ejections and sweeps, respectively. Near the wall, there is a predominance of Q4 events over Q2 motions, and farther from the wall the opposite is observed. Q1 and Q3 events are called outward and inward interactions respectively, and they are comparatively less frequent. Q2 and Q4 motions are *gradient-type* interactions, since they are indicative of vertical momentum fluxes related to fluid motions up (Q2) and down (Q4) the mean-velocity gradient with smaller (Q2) or larger (Q4) streamwise momentum than average. Q1 and Q3 are instead

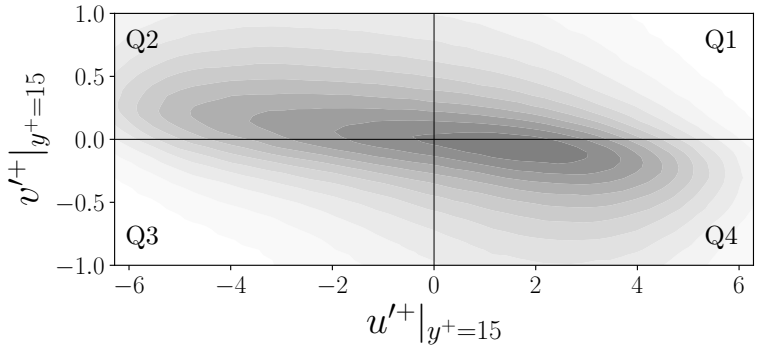


Figure 5.2: Joint probability distribution function $P(u', v')$ at $y^+ = 15$ in a turbulent open channel flow with $Re_\tau = 180$. A darker colour corresponds to a higher probability.

called *counter-gradient-type* interactions. Quadrant analysis is very effective in revealing the connection between the distribution of the fluctuations and the turbulent features that characterize the flow. Therefore, we will use it to identify how the flow physics is affected by the different control strategies.

Chapter 6

Conclusions and outlook

In this thesis, several possible deep-learning and deep-reinforcement-learning applications are analyzed. Whether they are used for temporal predictions or spatial reconstructions, neural-network models have shown the ability to outperform other existing methods. One reason for this is the non-linear nature of these models, which allows them to provide more expressive representations of the input-output relations that they approximate.

Historically, turbulence research relied on robust, yet mathematically-simple models for prediction, reconstruction and control. Neural networks have a very large number of parameters, and larger models typically can generalize better (Novak *et al.* 2018). This counter-intuitive property allows us to train very large models without the need to estimate the number of learnable parameters (or *model complexity*) for a specific task. Despite the encouraging results, we have not explored the full potential of these data-driven models for turbulence prediction and control. Here we review some future research directions for the different applications.

Temporal predictions. The temporal evolution of different chaotic systems can be reproduced in a statistically-consistent manner using recurrent neural networks or KNF. It is worth noting that, while turbulent motions are chaotic, not all chaotic motions are turbulent. Recurrent neural networks provide promising results in the prediction of data-driven reduced-order models of turbulence; however, turbulence has a unique statistical characterization (Biferale *et al.* 2004) that is very difficult to model or reproduce. The ability of neural networks or other data-driven models to generate temporal trajectories with correct multi-time and multi-scale statistics still needs to be assessed.

The possibility to train a model to learn and reproduce the essential nature of turbulence is an important and appealing endeavour, however, neural networks can find application in the generation of other temporally-correlated data for turbulence simulations. For instance, Yousif *et al.* (2022) trained a transformer network, along with a super-resolution GAN, to generate inflow data for spatially-developing turbulent-boundary-layer simulations.

Spatial predictions. Convolution-based models can accurately reconstruct turbulent velocity flow fields starting from wall measurements and vice versa, thanks to the inductive biases in the network architecture. These non-linear

neural-network models outperform optimal linear models such as extended POD (Borée 2003), or equivalently, linear-stochastic estimation (Adrian 1988). In this thesis, the FCN models are trained and tested only on flow fields sampled from DNSs. However, these models have great potential for experimental and engineering applications, allowing the turbulent-flow characterization without the need of probing the flow in an intrusive way. Experimental data are subject to noise and typically have a much lower resolution than DNS data. The effect of such experimental limitations needs to be assessed before the deployment of these techniques in realistic scenarios.

Note that we consider for our predictions only two-dimensional fields. If three-dimensional quantities need to be predicted, physical constraints should be enforced directly in the predictions. For instance, predicting v and η instead of the velocity components allows us to enforce the incompressibility condition, while simultaneously reducing the number of outputs for the network. Further in the direction of increasing the complexity of the predictions, the combination of space and time predictions with a single neural-network model is another appealing research direction. One possible approach is to treat time as an additional dimension of the data and apply convolution operations both in space and time. Alternatively, it is possible to take advantage of the characteristics of temporal data by using specific layers such as convolutional LSTMs (Shi *et al.* 2015), used in weather forecasting.

DRL in fluid mechanics. The application of DRL in fluid mechanics is in a comparatively early stage with respect to that of deep learning. Our research represents only a first step towards the design of turbulent-flow control systems with these tools. The discovery of a drag-reducing policy that outperforms opposition control should be regarded as a first proof-of-concept of the potential of these techniques. It is worth noting that the reported DRL policy is only one of the different strategies learnt during the interaction of the agent with the flow. Policies that obtain similar rewards may rely on different physical mechanisms. Similarly to other approaches to the discovery of control strategies (Duriez *et al.* 2017), the analysis of these policies can provide insights on the way the flow features that are affected by the control, thus advancing our understanding of turbulence dynamics and how to manipulate it. Considering DRL not only as a control technique but also as a research tool can be regarded as part of the paradigm shift from 'understanding $\rightarrow$ modeling $\rightarrow$ control' to 'control $\rightarrow$ modeling $\rightarrow$ understanding' hinted by Noack (2019).

The extension of the numerical application of DRL to higher Reynolds numbers represents the next step towards the deployment of DRL-based solutions in engineering applications, such as drag reduction or separation control on wings (Vinuesa *et al.* 2022). As highlighted by Marusic *et al.* (2021), the best-performing drag-reduction strategies can exhibit a Reynolds-number dependency. Even if DRL algorithms are able to identify different strategies at different Reynolds numbers (Varela *et al.* 2022), the computational cost of the learning at a high Reynolds number is prohibitively high. While the refinement

of the numerical methods and the improvement of DRL-CFD interfaces can help to accelerate the agent training, alternative learning approaches may offer more effective solutions. For instance, Linot *et al.* (2023) trained a DRL agent using a reduced-order model of the flow case to be controlled. This solution is computationally efficient, but it shares some of the drawbacks of model-based DRL, described in chapter 5. Alternatively, different DRL algorithms, characterized by a higher sample efficiency, can be tested. One example is *offline RL* proposed by Levine *et al.* (2020), however, the possibility to sample efficiently all the different flow behaviour and configurations needs to be investigated. RL agents are trained to maximize the expected return $\mathbb{E}[R(\tau)]$, which implies approximating the instantaneous reward function $\mathcal{R} : \mathcal{S}^2 \times \mathcal{A} \to \mathbb{R}$. The use of a multi-agent approach, with several agents sharing the same policy and operating locally, can drastically improve the learning time (as shown in **Paper 9**), while enforcing physical properties such as the translational invariance of the policy.

The studies gathered in this thesis represent first, exploratory steps in the application of deep-learning and deep-reinforcement-learning models to turbulent-flow simulations. Despite the fact that turbulence is a three-dimensional phenomenon that exhibits temporal and spatial dependencies, we only considered these aspects separately. Furthermore, our models were trained to perform in flows with specific parameters (*e.g.* the Reynolds number). As of today, there is no established methodology to predict the model behaviour in different flow settings. Training and testing the models in new conditions require sampling new simulations, which become more and more expensive as we get closer to real-world engineering applications. This simulation cost can only partially be reduced using techniques like transfer learning. For this, the range of applicability of these techniques is still to be assessed.

While deep learning and deep reinforcement learning will not replace the simulation of turbulent flows altogether, they are able to provide non-linear models that can improve our ability to predict and simulate fluid flows. Fluid dynamics has high importance in strategic fields like energy production, terrestrial and airborne transportation and the attitude towards novel approaches is rather prudent (Spalart & McLean 2011). The implementation of new solutions for these fields involves strong robustness requirements, which can be met by including physical constraints in the neural-network-models response, but also adding a quantification of the uncertainty in the predictions.

Numerical approaches to data analysis (DMD) and system identification (SINDy) have been developed to solve fluid-dynamics problems and then extended to other fields and disciplines. By contrast, deep learning and deep reinforcement learning have first developed solutions for natural-language processing, computer vision and robotics, and we are currently exploring their application to fluid dynamics. These applications present unique challenges for these data-driven models, which need to be adapted to enforce physical constraints and deliver physically-accurate predictions. The refinement of existing

methods and the development of new models can lead to further advancements that can be mutually beneficial for both fields, machine learning and fluid dynamics.

Acknowledgements

This thesis is the result of almost 5 years of work, but it would not exist if all the people who supported me inside and outside of work had not been there. First I would like to thank my supervisors: Ricardo, Hossein and Philipp, who gave me the opportunity to pursue this PhD. Ricardo, for your constant feedback and support: your encouragement and enthusiasm have been essential during these years. Hossein, for your invaluable help while moving my first step with machine learning. Philipp, for your expert guidance on numerics and SIMSON.

I am also thankful to all the collaborators that helped shape my research work: Ari, Jean, Alejandro, and Hamidreza to name a few. Despite having spent a non-negligible part of the PhD working from home, the time in the office has always been very pleasant and enriching. For this, I thank my office mates and the other researchers in the department, including the ones that left and the ones that recently joined. Apologies for not mentioning all of you by name. A special thanks goes to Pol, for helping me design the thesis cover.

During these years in Sweden, I have been constantly surrounded by friends who made me feel at home. Thank you, Alma, Fermín and Jordi for all the lasagnas together. Thank you, Rocco, Lucia and Giovanni, for your sincere friendship and support, whenever needed. Thanks also to my first contact in Sweden, Giacomo, for introducing me to the right people at the right moment, including Carina, Sara and Giorgia.

Sweden is an amazing place to do a PhD, but also to spend time outdoors. I was lucky enough to do it while climbing and I would like to thank all the people that have shared this passion with me: Federico R., Ilaria, Marco C. E., Klaus and Maria. Thanks for pushing me to reach greater heights in this relatively-flat country. Thanks also to Emmi, Mercedes and Lavinia, for bringing a delightful musical twist to Stockholm's life. Few individuals could have also been mentioned earlier since I shared many different experiences with them: thank you Federico B., Marco and Andrea for all the moments together, life in Sweden would not have been the same without your company.

My heartfelt gratitude goes to all my friends in Italy, whom I am eager to meet again every time I travel back to Milan. Grazie alla mia famiglia, per il loro amore e supporto incondizionato. Finally, last but definitely not least, thank you, Chiara, for all the things that I was able to discover with you.

Bibliography

Adrian, R. 1988 Stochastic estimation of organized turbulent structure: Homogeneous shear flow. *J. Fluid Mech.* **190**, 531–559.

Bahdanau, D., Cho, K. & Bengio, Y. 2014 Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Banzhaf, W., Nordin, P., Keller, R. E. & Francone, F. D. 1998 *Genetic programming: an introduction*. Morgan Kaufmann.

Beck, A. D., Flad, D. G. & Munz, C.-D. 2019 Deep neural networks for data-driven LES closure models. *J. Comput. Phys.* **398**, 108910.

Belus, V., Rabault, J., Viquerat, J., Che, Z., Hachem, E. & Reglade, U. 2019 Exploiting locality and translational invariance to design effective deep reinforcement learning control of the 1-dimensional unstable falling liquid film. *AIP Advances* **9** (12), 125014.

Benard, N., Pons-Prats, J., Periaux, J., Bugeda, G., Braud, P., Bonnet, J. & Moreau, E. 2016 Turbulent separated shear flow control by surface plasma actuator: Experimental optimization by genetic algorithm approach. *Experiments in Fluids* **57** (2), 22.

Benzi, R. & Vulpiani, A. 2022 Multifractal approach to fully developed turbulence. *Rendiconti Lincei. Scienze Fisiche e Naturali* **33** (3), 471–477.

Biferale, L., Boffetta, G., Celani, A., Devenish, B. J., Lanotte, A. & Toschi, F. 2004 Multifractal statistics of lagrangian velocity and acceleration in turbulence. *Phys. Rev. Lett.* **93**, 064502.

Biferale, L., Bonaccorso, F., Buzzicotti, M., Clark Di Leoni, P. & Gustavsson, K. 2019 Zermelo's problem: optimal point-to-point navigation in 2D turbulent flows using reinforcement learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29** (10), 103138.

Biferale, L., Bonaccorso, F., Buzzicotti, M. & Clark di Leoni, P. 2020 TURB-Rot. A large database of 3D and 2D snapshots from turbulent rotating flows. `http://smart-turb.roma2.infn.it`. *arXiv:2006.07469*.

Biferale, L., Calzavarini, E. & Toschi, F. 2011 Multi-time multi-scale correlation functions in hydrodynamic turbulence. *Physics of Fluids* **23** (8), 085107.

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B. & Weinbach, S. 2022 GPT-NeoX-20B: An open-source autoregressive language model. *arXiv:2204.06745*.

Borée, J. 2003 Extended proper orthogonal decomposition: a tool to analyse correlated events in turbulent flows. *Exp. Fluids* **35** (2), 188–192.

Brunton, S. L., Brunton, B. W., Proctor, J. L., Kaiser, E. & Kutz, J. N. 2017 Chaos as an intermittently forced linear system. *Nat. Commun.* **8** (1), 19.

Brunton, S. L., Budišić, M., Kaiser, E. & Kutz, J. N. 2022 Modern koopman theory for dynamical systems. *SIAM Review* **64** (2), 229–340.

Brunton, S. L. & Noack, B. R. 2015 Closed-Loop Turbulence Control: Progress and Challenges. *Applied Mechanics Reviews* **67** (5), 050801.

Brunton, S. L., Proctor, J. L. & Kutz, J. N. 2016 Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences* **113** (15), 3932–3937.

Buzzicotti, M., Biferale, L. & Toschi, F. 2020 Statistical properties of turbulence in the presence of a smart small-scale control. *Phys. Rev. Lett.* **124**, 084504.

Buzzicotti, M., Bonaccorso, F., Di Leoni, P. C. & Biferale, L. 2021 Reconstruction of turbulent data with deep generative models for semantic inpainting from turb-rot database. *Phys. Rev. Fluids* **6**, 050503.

Carleman, T. 1932 Application de la théorie des équations intégrales linéaires aux systémes d'équations différentielles non linéaires. *Acta Mathematica* **59**, 63 – 87.

Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D. & Sutskever, I. 2020 Generative pretraining from pixels. In *Proceedings of the 37th International Conference on Machine Learning* (ed. H. D. III & A. Singh), *Proceedings of Machine Learning Research*, vol. 119, pp. 1691–1703. PMLR.

Chevalier, M., Schlatter, P., Lundbladh, A. & Henningson, D. 2007 A pseudospectral solver for incompressible boundary layer flows. *Tech. Rep.*. TRITA-MEK 2007:07. KTH Mechanics, Stockholm, Sweden.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. 2014 Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Choi, H., Moin, P. & Kim, J. 1994 Active turbulence control for drag reduction in wall-bounded flows. *Journal of Fluid Mechanics* **262**, 75–110.

Devlin, J., Chang, M., Lee, K. & Toutanova, K. 2018 BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. & Houlsby, N. 2020 An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv:2010.11929*.

Duriez, T., Brunton, S. L. & Noack, B. R. 2017 *Machine Learning Control – Taming Nonlinear Dynamics and Turbulence*. Springer.

Einicke, G. & White, L. 1999 Robust extended kalman filtering. *IEEE Transactions on Signal Processing* **47** (9), 2596–2599.

Eivazi, H., Le Clainche, S., Hoyas, S. & Vinuesa, R. 2022 Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows. *Expert Systems with Applications* **202**, 117038.

Fukami, K., Nakamura, T. & Fukagata, K. 2020 Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. *Physics of Fluids* **32** (9), 095110.

Gautier, N., Aider, J.-L., Duriez, T., Noack, B. R., Segond, M. & Abel, M.

2015 Closed-loop separation control using machine learning. *Journal of Fluid Mechanics* **770**, 442–457.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. 2014 Generative adversarial networks. *arXiv:1406.2661*.

Guastoni, L., Balasubramanian, A. G., Güemes, A., Ianiro, A., Discetti, S., Schlatter, P., Azizpour, H. & Vinuesa, R. 2022 Non-intrusive sensing in turbulent boundary layers via deep fully-convolutional neural networks. *arXiv:2208.06024*.

Guastoni, L., Güemes, A., Ianiro, A., Discetti, S., Schlatter, P., Azizpour, H. & Vinuesa, R. 2021 Convolutional-network models to predict wall-bounded turbulence from wall quantities. *J. Fluid Mech.* **928**, A27.

Güemes, A., Discetti, S., Ianiro, A., Sirmacek, B., Azizpour, H. & Vinuesa, R. 2021 From coarse wall measurements to turbulent velocity fields through deep learning. *Physics of Fluids* **33** (7), 075121.

Hochreiter, S. & Schmidhuber, J. 1997 Long short-term memory. *Neural Comput.* **9**, 1735–1780.

Jiménez, J. & Moin, P. 1991 The minimal flow unit in near-wall turbulence. *Journal of Fluid Mechanics* **225**, 213–240.

Julier, S. J. & Uhlmann, J. K. 1997 New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI* (ed. I. Kadar), , vol. 3068, pp. 182 – 193. International Society for Optics and Photonics, SPIE.

Kalman, R. E. 1960 A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering* **82** (Series D), 35–45.

Khodkar, M. & Hassanzadeh, P. 2021 A data-driven, physics-informed framework for forecasting the spatiotemporal evolution of chaotic dynamics with nonlinearities modeled as exogenous forcings. *Journal of Computational Physics* **440**, 110412.

Kim, J., Moin, P. & Moser, R. 1987 Turbulence statistics in fully developed channel flow at low reynolds number. *Journal of Fluid Mechanics* **177**, 133–166.

Kolmogorov, A. 1941 The Local Structure of Turbulence in Incompressible Viscous Fluid for Very Large Reynolds' Numbers. *Akademiia Nauk SSSR Doklady* **30**, 301–305.

Koopman, B. O. 1931 Hamiltonian systems and transformation in Hilbert space. *Proc. Natl. Acad. Sci.* **17** (5), 315–318.

Kramer, M. A. 1991 Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal* **37** (2), 233–243.

Kurz, M., Offenhäuser, P. & Beck, A. 2023 Deep reinforcement learning for turbulence modeling in large eddy simulations. *International Journal of Heat and Fluid Flow* **99**, 109094.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W. & Jackel, L. 1989 Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*. Morgan-Kaufmann.

Levine, S., Kumar, A., Tucker, G. & Fu, J. 2020 Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv:2005.01643*.

Li, Y., Perlman, E., Wan, M., Yang, Y., Meneveau, C., Burns, R., Chen,

S., Szalay, A. & Eyink, G. 2008 A public turbulence database cluster and applications to study lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence* **9**, N31.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D. 2015 Continuous control with deep reinforcement learning. *arXiv:1509.02971*.

Lim, B. & Zohren, S. 2021 Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **379** (2194), 20200209.

Linot, A. J., Zeng, K. & Graham, M. D. 2023 Turbulence control in plane couette flow using low-dimensional neural ODE-based models and deep reinforcement learning. *arXiv:2301.12098*.

Lorsung, C. & Barati Farimani, A. 2023 Mesh deep Q network: A deep reinforcement learning framework for improving meshes in computational fluid dynamics. *AIP Advances* **13** (1), 015026.

Lozano-Durán, A., Flores, O. & Jiménez, J. 2012 The three-dimensional structure of momentum transfer in turbulent channels. *J. Fluid Mech.* **694**, 100–130.

Lucas, J., Tucker, G., Grosse, R. B. & Norouzi, M. 2019 Understanding posterior collapse in generative latent variable models. In *DGS@ICLR*.

Marusic, I., Chandran, D., Rouhi, A., Fu, M. K., Wine, D., Holloway, B., Chung, D. & Smits, A. J. 2021 An energy-efficient pathway to turbulent drag reduction. *Nature Communications* **12** (5805).

Marusic, I. & Heuer, W. D. C. 2007 Reynolds number invariance of the structure inclination angle in wall turbulence. *Phys. Rev. Lett.* **99**, 114504.

Mezić, I. 2005 Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dyn.* **41** (1), 309–325.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. 2013 Playing Atari with deep reinforcement learning. *arXiv:1312.5602*.

Moehlis, J., Faisst, H. & Eckhardt, B. 2004 A low-dimensional model for turbulent shear flows. *New J. Phys.* **6**, 56.

Nair, V. & Hinton, G. E. 2010 Rectified linear units improve restricted Boltzmann machines. In *Proc. of the 27th Int. Conf. on Machine Learning (ICML-10)*, pp. 807–814.

Nakagawa, H. & Nezu, I. 1981 Structure of space-time correlations of bursting phenomena in an open-channel flow. *Journal of Fluid Mechanics* **104**, 1–43.

Noack, B. R. 2019 Closed-loop turbulence control-from human to machine learning (and retour). In *Fluid-Structure-Sound Interactions and Control* (ed. Y. Zhou, M. Kimura, G. Peng, A. Lucey & L. Huang), pp. 23–32. Singapore: Springer Singapore.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J. & Sohl-Dickstein, J. N. 2018 Sensitivity and generalization in neural networks: an empirical study. *arXiv:1802.08760*.

Novati, G., de Laroussilhe, H. L. & Koumoutsakos, P. 2021 Automating turbulence modeling by multi-agent reinforcement learning. *Nature Machine Intelligence* **3**, 87–96.

Olah, C. 2015 Understanding LSTM Networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

van den Oord, A., Vinyals, O. & kavukcuoglu, k. 2017 Neural Discrete Representation Learning. In *Neural Information Processing Systems*, , vol. 30. Curran Associates, Inc.

Pan, S. J. & Yang, Q. 2009 A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* **22** (10), 1345–1359.

Paris, R., Beneddine, S. & Dandois, J. 2021 Robust flow control and optimal sensor placement using deep reinforcement learning. *Journal of Fluid Mechanics* **913**, A25.

Park, J. & Choi, H. 2020 Machine-learning-based feedback control for drag reduction in a turbulent channel flow. *Journal of Fluid Mechanics* **904**, A24.

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

Rabault, J., Kuchta, M., Jensen, A., Réglade, U. & Cerardi, N. 2019 Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics* **865**, 281–302.

Radford, A. & Narasimhan, K. 2018 Improving language understanding by generative pre-training. *Preprint*.

Ren, F., Rabault, J. & Tang, H. 2021 Applying deep reinforcement learning to active flow control in weakly turbulent conditions. *Physics of Fluids* **33** (3), 037121.

Reynolds, O. 1883 XXIX. 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. *Philosophical Transactions of the Royal Society of London* **174**, 935–982.

Rowley, C. W., Mezić, I., Bagheri, S., Schlatter, P. & Henningson, D. S. 2009 Spectral analysis of nonlinear flows. *J. Fluid Mech.* **641**, 115–127.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. 1985 Learning internal representations by error propagation. *Tech. Rep.*. California Univ San Diego La Jolla Inst for Cognitive Science.

Schmid, P. J. 2010 Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* **656** (November), 5–28.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. 2017 Proximal policy optimization algorithms. *arXiv:1707.06347*.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W. & Woo, W. 2015 Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv:1506.04214*.

Sinai, Y. G. 2003 Preface. In: A. Vulpiani and R. Livi. The Kolmogorov legacy in physics. Berlin: Springer.

Spalart, P. R. & McLean, J. D. 2011 Drag reduction: enticing turbulence, and then an industry. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **369** (1940), 1556–1569.

Steeb, W.-H. 1981 Linearization procedure and nonlinear systems of differential and difference equations. In *Nonlinear Phenomena in Chemical Dynamics* (ed. C. Vidal & A. Pacault), pp. 275–275. Berlin, Heidelberg: Springer Berlin Heidelberg.

Sutton, R. S. 1988 Learning to predict by the methods of temporal differences. *Machine Learning* **3** (1), 9–44.

Sutton, R. S. & Barto, A. G. 2018 *Reinforcement Learning: An Introduction*, 2nd edn. The MIT Press.

Tang, H., Rabault, J., Kuhnle, A., Wang, Y. & Wang, T. 2020 Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Physics of Fluids* **32** (5), 053605.

Taylor, G. I. 1935 Statistical theory of turbulence. *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences* **151** (873), 421–444.

Varela, P., Suárez, P., Alcántara-Ávila, F., Miró, A., Rabault, J., Font, B., García-Cuevas, L. M., Lehmkuhl, O. & Vinuesa, R. 2022 Deep reinforcement learning for flow control exploits different physics for increasing Reynolds number regimes. *Actuators* **11** (12).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. 2017 Attention is all you need. *arXiv:1706.03762*.

Verma, S., Novati, G. & Koumoutsakos, P. 2018 Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences* **115** (23), 5849–5854.

Vignon, C., Rabault, J. & Vinuesa, R. 2023 Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions. *Physics of Fluids* **35** (3), 031301.

Vinuesa, R. & Brunton, S. 2022 Enhancing computational fluid dynamics with machine learning. *Nature Computational Science* **2**, 358–366.

Vinuesa, R., Lehmkuhl, O., Lozano-Durán, A. & Rabault, J. 2022 Flow control in wings and discovery of novel approaches via deep reinforcement learning. *Fluids* **7** (2), 62.

Wallace, J. M., Eckelmann, H. & Brodkey, R. S. 1972 The wall region in turbulent shear flow. *Journal of Fluid Mechanics* **54** (1), 39–48.

Welch, P. 1967 The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics* **15** (2), 70–73.

Williams, M. O., Kevrekidis, I. G. & Rowley, C. W. 2015 A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science* **25** (6), 1307–1346.

Williams, R. J. 1992 Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8** (3), 229–256.

Xu, Z.-Q. J., Zhang, Y. & Xiao, Y. 2019 Training behavior of deep neural network in frequency domain. In *Neural Information Processing* (ed. T. Gedeon, K. W. Wong & M. Lee), pp. 264–274. Cham: Springer International Publishing.

Yousif, M. Z. G., Zhang, M.-T., Yu, L., Vinuesa, R. & Lim, H. 2022 A transformer-based synthetic-inflow generator for spatially developing turbulent boundary layers. *Journal of Fluid Mechanics* **957**.

Part II

Papers

Summary of the papers

Paper 1

Predictions of turbulent shear flows using deep neural networks

In this work, the possibility to predict the temporal evolution of turbulent flows using neural networks is explored. In particular, a nine-equation shear-flow model is used to generate the training and test data. Two different neural-network types are considered: the multilayer perceptron (MLP) and the long short-term memory (LSTM) network. Different architectures for each type are trained, varying the number of layers, number of units per layer, dimension of the input, weight initialization and activation functions. LSTM networks are better-suited to analyze sequential data, hence they perform better than all the tested MLP networks. LSTM can correctly reproduce the temporal evolution of the model, both in terms of turbulence statistics and of the dynamical behavior of the system (characterized by Poincaré maps and Lyapunov exponents).

Paper 2

Recurrent neural networks and Koopman-based frameworks for temporal predictions in a low-order model of turbulence

The capabilities of recurrent neural networks and Koopman-based frameworks are assessed in the prediction of temporal dynamics of the low-order model of near-wall turbulence by Moehlis *et al.* (New J. Phys. **6**, 56, 2004). Two different data-driven models are considered: long-short-term memory (LSTM) networks and a newly-developed Koopman-based framework called Koopman with nonlinear forcing (KNF). Both can predict the long-term behaviour of the dynamical system with relative errors in the statistics (mean and the fluctuations) below 1%. The KNF framework outperforms the LSTM network when it comes to short-term predictions and the model tuning requires significantly less computational power. Since the LSTM training is performed on the instantaneous predictions, the model is not optimized to reproduce the statistics of the dynamical system. By using a train-stopping criterion based on the computed statistics, it is possible to achieve excellent statistical reconstruction with minimal loss of accuracy in the instantaneous predictions.

Paper 3

Predicting the temporal dynamics of turbulent channels through deep learning

Two data-driven models, a long short-term memory (LSTM) network and a Koopman non-linear forcing model, are optimized to reproduce the temporal evolution of a minimal turbulent channel flow. We first obtain a data-driven model based on a modal decomposition in the Fourier domain (which we denote as FFT-POD) of the time series sampled from the flow. LSTM and KNF models are trained to predict the temporal dynamics of the minimal-channel-flow modes. Tests with different configurations highlight the limits of the KNF method compared to the LSTM, given the complexity of the flow under study. Long-term predictions with LSTM networks show excellent agreement from the statistical point of view, with errors below 2% for the best models with respect to the reference. The chaotic behaviour of the trained LSTM models is compared with that of the original simulation using Lyapounov exponents, while the dynamic behaviour is analyzed using Poincaré maps.

Paper 4

Prediction of wall-bounded turbulence from wall quantities using convolutional neural networks

A fully-convolutional neural-network (FCN) model is used to predict the streamwise velocity fields at several wall-normal locations by taking as input the streamwise and spanwise wall-shear-stress planes in a turbulent open channel flow. For each sampled wall-normal location ($y^+ = 15,\ 30,\ 50$), we trained different networks to predict the wall-parallel flow field, using the data sampled from a direct numerical simulation at friction Reynolds number $Re_\tau = 180$. Thanks to their non-linearity, the neural networks outperform the linear methods existing in literature, both in terms of instantaneous accuracy and turbulence statistics. When a higher time step between the samples Δt_s^+ is considered, the prediction accuracy increases at all the considered wall-normal locations, as long as the network capacity is sufficient to generalize over the dataset. Training time can be effectively reduced, by a factor of 4, via a *transfer-learning* method that initializes the network parameters using the optimized parameters of a previously-trained network.

Paper 5

Convolutional-network models to predict wall-bounded turbulence from wall quantities

Two models based on convolutional neural networks are trained to predict the two-dimensional instantaneous velocity-fluctuation fields at different wall-normal locations in a turbulent open-channel flow, using the wall-shear-stress components and the wall pressure as inputs. The first model is a fully-convolutional neural network (FCN) which directly predicts the fluctuations, while the second one reconstructs the flow fields using a linear combination of orthonormal basis

functions, obtained through proper orthogonal decomposition (POD), hence named FCN-POD. Both models are trained using data from direct numerical simulations (DNS) at friction Reynolds numbers $Re_\tau = 180$ and 550. Being able to predict the nonlinear interactions in the flow, both models show better predictions than the extended proper orthogonal decomposition (EPOD), which establishes a linear relation between input and output fields. The performance of the models is compared based on predictions of the instantaneous fluctuation fields, turbulence statistics and power-spectral densities. Transfer learning can be used for the FCN model, using the model parameters learned from $Re_\tau = 180$ dataset to initialize those of the model that is trained on the $Re_\tau = 550$ dataset. After training the initialized model at the new Re_τ, our results indicate the possibility to match the reference-model performance up to $y^+ = 50$, with as little as 25% of the original training data.

Paper 6

Direct numerical simulation of a zero-pressure-gradient thermal turbulent boundary layer up to $Pr = 6$

In this paper, we report the analysis of a direct numerical simulation (DNS) of an incompressible zero-pressure-gradient turbulent boundary layer is performed with the Reynolds number based on momentum thickness Re_θ up to 1080. Four passive scalars, characterized by the Prandtl numbers $Pr = 1, 2, 4, 6$, are simulated with constant Dirichlet boundary conditions. To the best of our knowledge, the present DNS dataset provides the thermal boundary layer with the highest Prandtl number available in the DNS literature. Turbulence statistics for the flow and thermal fields are computed and compared with available numerical simulations at similar Reynolds numbers. The mean flow and temperature profiles, root-mean squared (RMS) velocity and temperature fluctuations, turbulent heat flux, turbulent Prandtl number and higher-order statistics agree well with the numerical data reported in the literature. Furthermore, the pre-multiplied two-dimensional spectra of the velocity and of the passive scalars are computed, providing a quantitative description of the energy distribution at the different lengthscales for various wall-normal locations. The energy distribution of the heat-flux fields at the wall is concentrated on longer temporal structures and exhibits a different footprint at the wall, with increasing Prandtl number.

Paper 7

Fully-convolutional networks for velocity-field predictions based on the wall heat flux in turbulent boundary layers

The effectiveness of fully-convolutional neural networks (FCNs) in predicting the instantaneous state of a fully-developed turbulent flow at different wall-normal locations using wall-measured quantities has been analyzed in Guastoni *et al.* (2021). In that study, wall-shear-stress distributions were used as inputs and they can be challenging to measure in experiments. To address this issue, we introduce a model that utilizes the heat-flux field at the wall from a

passive scalar as input. The study considered four different Prandtl numbers $Pr = \nu/\alpha = (1, 2, 4, 6)$, where the thermal diffusivity and kinematic viscosity of the scalar quantity are represented by α and ν, respectively. The fields are sampled from the DNS of a turbulent boundary layer, since accurate heat-flux measurements can be performed in experimental settings. This study represents the first step towards the implementation of a *non-intrusive* sensing approach for the flow in practical applications, for instance for flow control.

Paper 8

Predicting the wall-shear stress and wall pressure through convolutional neural networks

In this study, we assess the capability of convolution-based neural networks to predict the wall quantities in a turbulent open channel flow, starting from measurements above the wall. Two different fully-convolutional networks are tested, called FCN and R-Net (the latter takes the name from the residual connections that are present in the network architecture). At $Re_\tau = 550$, both FCN and R-Net can take advantage of the self-similarity in the logarithmic region of the flow and predict the velocity-fluctuation fields at $y^+ = 50$ using the velocity-fluctuation fields at $y^+ = 100$ as input with about 10% error in prediction of streamwise-fluctuation intensity. R-Net models perform better than the FCN models when trained to predict the wall-shear-stress and wall-pressure fields using the velocity-fluctuation fields. At $y^+ = 50$, R-Net predictions have an error around 10% in the fluctuations intensity at both $Re_\tau = 180$ and 550. These results are an encouraging starting point to develop neural-network-based approaches for modelling turbulence near the wall in numerical simulations, especially in the context of wall-modeled large-eddy simulations (WMLESs).

Paper 9

Deep reinforcement learning for turbulent drag reduction in channel flows

The control of turbulent fluid flows represents a problem in several engineering applications. The chaotic, high-dimensional, non-linear nature of turbulence hinders the possibility to design robust and effective control strategies. In this work, we apply deep reinforcement learning to a three-dimensional turbulent open-channel flow, a canonical-flow example that is often used as a study case in turbulence, aiming to reduce the friction drag in the flow. By casting the fluid-dynamics problem as a multi-agent reinforcement-learning environment and by training the agents using a location-invariant deep deterministic policy gradient algorithm, we are able to obtain a control strategy that achieves a remarkable 43% and 30% drag reduction in a minimal and a larger channel (at a friction Reynolds number of 180), respectively, outperforming the classical opposition control by around 20 and 10 percentage points, respectively.