École polytechnique fédérale de Lausanne

Master thesis spring 2023

Master in Mechanical Engineering

Turbulent heated jet: scientific machine learning inference of velocity and pressure fields from the temperature field

Author:

Gabriel Bessette

Supervisors:

Pr. Julio Soria, Head of the Laboratory for Turbulence Research in Aerospace and Combustion, Monash University

Pr. Tobias Schneider, Head of the Laboratory of Emergent Complexity in Physical Systems, EPFL

Acknowledgements

I want to thank supervisors, Pr. J. Soria and Pr. T. Schneider for their trust, support, availability and cooperation during the entire project. Especially to Pr. J. Soria, for hosting me at the Laboratory for Turbulence Research in Aerospace and Combustion (LTRAC) at Monash University, Clayton, and for providing access to computational resources.

I would also like to thank C. Atkinson, V. Kitsios and any other member of the LTRAC for their advice and guidance throughout my stay at Monash University.

Finally, I am thankful towards my entire family for their constant support and for giving me the chance to take part in this project far away from home, in Australia.

Abstract

To investigate new ways of predicting the flow of a 3D turbulent heated jet, this project aims at developing and evaluating scientific machine learning models for the inference of velocity and pressure components, by only knowing the flow's temperature field. It explores and evaluates the performance of Physics-Informed Neural Networks (PINN) and provides guidelines for the inference of these flow variables.

Due to the difficulty in measuring features of a turbulent flow (Dahm et al. [1]), PINNs could be used as a model to predict velocity and pressure by only knowing some characteristics of the flow. Recent advances in experimental techniques have shown that temperature fields could be accurately measured with the Back-Oriented Schlieren method (Shoaib [2]) and therefore provide valuable data for the retrieval of velocities and pressure terms. As a first step, PINN models have been studied with reference DNS data, for validation purposes.

This report includes the study of a 3D turbulent heated round jet at high Reynolds number (Re=10'000) for three different PINN models: a first model only driven by target temperature data, a second one considering additional boundary conditions and a final model accounting for noisy target values. For these three configurations, relative errors have been computed and compared to ground truth DNS data from Karami et al. [3].

Table of Contents

List of Figures

List of Tables

Nomenclature

Accents & Subscripts	Description
x	non-dimensional quantities
x^*	dimensional quantities
x_o^*	reference quantities
x_{tgt}	target quantities
$\tilde{x}$	noisy quantities
$\bar{x}$	mean quantities
$\hat{x}$	axis
x_{norm}	normalised quantities
x_{min}	minimum quantities
x_{max}	maximum quantities
x_t	quantities at iteration t
$x^{(i)}$	quantities of layer i in a neural network
x_F	quantities belonging to the family of functions F
x_T	training quantities

Roman Symbols

Symbol	Description	Units
a	speed of sound	m/s
b	neural network's bias	-
d	inlet orifice diameter	m
e	specific energy	J
F	family of functions represented by a neural network architecture	-
L_{data}	data loss	-
L_{res}	residual loss	-
L_{tot}	total loss	-
m	first order momentum in Adam optimizer	-
N	number of data points	-
N_r	number of grid points in the r dimension	-
N_t	number of grid points in the time dimension	-
N_θ	number of grid points in the θ dimension	-
N_z	number of grid points in the z dimension	-
n_{lvl}	noise level	-
p	pressure	Pa
r	radial coordinate	m
s	weight's standard deviation	-
t	time	s
T	temperature	K
U	average velocity	m/s
u_r	velocity in radial direction	m/s
u_θ	velocity in azimutal direction	m/s
u_z	velocity in axial direction	m/s
V	PINN's estimated variable $(T, u_z, u_r, u_\theta, p)$	-
V_{out}	PINN's output variable $(T, u_z, u_r, u_\theta, p)$	-

Symbol	Description	Units
v	second order momentum in Adam optimizer	-
w	neural network's weight	-
x	output of a layer in a neural network	-
z	axial coordinate	m

Greek Symbols

Symbol	Description	Units
α	loss weighting factor	[0,1]
α_1	loss weighting factor 1	[0,1]
α_2	loss weighting factor 2	[0,1]
α_d	momentum diffusivity	m^2/s
β_1	Adam optimiser parameter	[0,1]
β_2	Adam optimiser parameter	[0,1]
Γ	domain	-
γ	heat capacity ratio	-
Δ	difference	-
ε_{app}	approximation error	-
ε_{gen}	generalisation error	-
ε_{opt}	optimisation error	-
ϵ_p	pressure relative error	%
ϵ_T	temperature relative error	%
ϵ_{u_r}	radial velocity relative error	%
ϵ_{u_θ}	azimutal velocity relative error	%
ϵ_{u_z}	axial velocity relative error	%
η	learning rate	-
θ	azimutal coordinate	-
μ	dynamic viscosity	$kg/m/s$
ν	kinematic viscosity	m^2/s
ρ	density	kg/m^3
Ω	studied domain	-
$\partial\Omega$	studied domain's boundaries	-

Dimensionless Numbers

Symbol	Description	Formula
Ma	Mach number	$Ma = \frac{u}{a}$
Pe	Péclet Number	$Pe = Pr \cdot Re$
Pr	Prandtl Number	$Pr = \frac{\nu}{\alpha_d}$
Re	Reynolds Number	$Re = \frac{\rho u d}{\mu}$

Acronyms

Abbreviation	Expansion
Adam	Adaptive Moment Estimation
DNS	Direct Numerical Simulation
GD	Gradient Descent
LTRAC	Laboratory for Turbulence Research in Aerospace and Combustion
MDP	Multi-Dimensional Perceptron
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NS	Navier-Stokes
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
RANS	Reynolds Averaged Navier-Stokes
SGD	Stochastic Gradient Descent

1 Introduction

This project represents an initial step in the development of a scientific machine learning model aimed at estimating velocities and pressure within a 3D turbulent heated jet based solely on its temperature field. It is a preliminary investigation under the incompressibility assumption which explores the concept of Physics-Informed Neural Networks to predict flow variables of a jet on a reduced domain, and uses DNS data (Karami [3]) as reference values. Additionaly, it gathers useful information about optimisations with PINNs and provides some helpful guidelines to solve encountered issues.

The studied jet has been chosen thanks to the availability of high fidelity simulated data at the Laboratory for Turbulence Research in Aerospace and Combustion (LTRAC) at Monash University in Clayton. It has been selected for its turbulent behaviour ($Re = 10'000$) as well as for its three dimensional simple geometry.

Throughout this work, a first method has been investigated to study the ability of a neural network to predict the velocity and pressure components of the jet by only feeding temperature target values. However, this approach made it impossible to retrieve accurate estimations of the flow's velocity and pressure terms. Indeed, specifying additional boundary conditions has turned out to be detrimental in the model's inference, and this technique has been studied as a second step. Finally, the PINN has been evaluated on its ability to handle noisy targets in the optimisation process, as a way to simulate experimental target values.

This report first dives into the literature revolving around PINNs applied to fluid mechanics problems. It then provides a description of jet flows along with their modelisation in chapter 3. Numerical methods that have been used for this work and the corresponding simulation setup are further described in sections 4 and 5. At last, results are discussed in section 6 and some flow visualisations are illustrated in the Appendix.

2 Previous Studies

Methods for inferring velocity components in turbulent flows using limited scalar field data have been previously developed, with one example being the scalar imaging velocimetry technique proposed by Dahm et al. [1]. This technique enables the extraction of the velocity field by inversion of the exact conserved scalar transport equation. It is done in three steps: first by collecting accurate 4D scalar fields of the flow to be able to write its transport equation, second by retrieving the velocity component projected in the direction of the scalar's gradient at every point in space and time, and finally, by extracting the full velocity vector using matrix inversion. However, these intermediate steps require to find suitable regions of the flow where inversions can be performed. This is sometimes tedious and cumbersome, therefore another approach involving neural networks has been investigated.

Since the early work of Funahashi [4] and Cybenko [5] on the development of the universal approximation theorem, awareness has been raised on the ability to use neural networks as function approximators. Indeed, according to Hornik et al. [6], a 2-layer neural network is sufficient to approximate any continuous function on a compact domain. Along with recent developments in open source Python libraries on machine learning modeling such as Pytorch [7] or Tensorflow [8], and with recent improvements in computational power and in the ability to run optimisations on GPUs (NVIDIA cuda [9]), neural networks have become very popular. Spanning a large range of applications in various domains such as biology for disease identification (Mazlan et al. [10]), visual recognition with computer vision (Voulodimos et al. [11]), or business for data-based forecasting (Hassani et al. [12]), neural networks have been solicited, not only for classification or regression tasks, but also for modeling and generative tasks.

Among these applications, recent advances in scientific machine learning have opened the door to new numerical methods for solving partial differential equations (PDE). Indeed, Raissi et al. [13] proposed a framework that solves PDEs with deep neural networks through the minimization of residuals and inclusion of target data points in the optimisation process. Initially built to compensate for the lack of available data in neural networks' training, Physics-Informed Neural Networks (PINN) utilize both concepts of supervised and unsupervised learning by separating the loss function into two terms: a first term which pushes the model to fit target values, and another term, that can be considered as additional constraints in the optimisation's loss formulation, that eventually drives the neural network's outputs to satisfy physics equations. Although PINNs require some adequate parameter optimisation to model complex problems and are sometimes prone to fail for complicated modeling tasks (Krishnapriya et al. [14]), they have been widely used thanks to their grid independent property and to their ability to solve inverse problems. In particular, PINNs have been used in fluid mechanics to solve Navier-Stokes equations of motion.

Different neural architectures and multiple techniques have been investigated to adapt PINNs for solving fluid problems. A first method consists in providing boundary conditions on the outer simulation domain as the supervised data loss term, and in considering Navier-Stokes equations as the unsupervised loss term (as residuals). Using this method, Eivazi et al.

[15] have achieved accurate results for predicting the Falkner–Skan laminar boundary layer, with less than 1% L_2 relative error on each of the flow's variables. The same authors have also modified the output of the PINN by including Reynolds-stress components, in order to estimate turbulent flow features. Simulations have been performed on a zero-pressure-gradient boundary layer, on an adverse-pressure-gradient boundary layer, on a turbulent flow over a NACA4412 airfoil, and on a flow over a periodic hill, with less than 4% L_2 relative error obtained for the streamwise velocity component. However, other variables were not accurately captured for these turbulent flows, underlining the difficulties PINNs have to solve complex problems. To improve the PINN's accuracy for turbulent channel flows, Jin et al. [16] have used an additional data driven loss term in the PINN's loss formulation that accounts for initial conditions. Additionally, they have tested a vecolicty-vorticty formulation of Navier-Stokes equations and applied Wang et al. [17] method of dynamical weight updates during training, which were beneficial to improve the solution's accuracy. The authors obtained similar results compared to Eivazi et al. [15], with low relative error for the velocity in the stream-wise direction and higher error for all other variables, increasing throughout time, especially for pressure.

To further improve the convergence of PINNs and better understand their loss landscape, Gopakumar et al. [18] have shown that feeding sparse or coarse data inside the simulation domain could act as a regulator, pushing the network towards the vicinity of the solution. Hence adding an extra data loss term defined on the entire simulation domain can greatly improve the accuracy of the solution. This has also been shown in the work by Kag et al. [19] on the simulation of 2D turbulence using stream functions and a spectral decomposition, where only 0.1% of the training data inside the domain was used as a regulator. The PINN model in this case successfully captured the statistics of large scale modes and improved the overall accuracy of the solution.

Raissi et al. [20] pushed the idea of data acting as a regulator to its limit, by inferring velocities and pressure of low Reynolds incompressible 2D and 3D flows around a cylinder, by only considering data targets from a passive scalar, without information on any of the other flow variables. However, this was only achieved on a specific domain in the cylinder's wake. As a matter of fact, inlet boundary conditions were necessary to solve Navier-Stokes on the entire simulation domain. However, Cai et al. [21] managed to predict velocity and pressure components of a 2D and 3D convective heated flow under the Boussinesq assumption, by considering only target temperature values on the entire domain. Along with a study on parameter optimisation including the neural network's size, the spatial and temporal resolution of temperature target values, and added noise on these targets, Cai et al. [22] have demonstrated the PINN's ability to predict accurate results and handle noisy targets.

With this in mind, this project examines the PINN's ability to infer velocities and pressure of a 3D heated turbulent jet flow at high Reynolds number, by only providing temperature values in the data loss formulation, and investigates the effect of noisy targets on the accuracy of the solution.

3 Theory

This chapter introduces jet flows and provides insights to how these flows are resolved, especially through Direct Numerical Simulations (DNS). In the aim of reducing the complexity of the problem formulation, assumptions on the flow's governing equations are also being discussed.

3.1 Jet Flow

Jet flows belong the family of shear flows including channel flows, boundary layers, wake flows and others (Ball et al. [23]). They are mostly solicited in combustion processes but also play an important role in other chemical processes such as cooling, drying and mixing. Thanks to their turbulent nature, jets have been studied in turbulence research and have provided insights into the formulation of turbulent models (Abdel-Rahman [24]). Different types of jet flows exist, but for the scope of this study, we will first consider simple round jet flows and further discuss about variable density jet flows (Chaissaing et al. [25]).

Round free jet flows, or canonical jets, are flows emanating from a nozzle and merging into another medium. Jets are mainly defined by their Reynolds number Re_d, which strongly depends on the nozzle's diameter d, the average velocity U, the fluid's density ρ and viscosity μ.

$$Re_d = \frac{\rho^* \cdot U^* \cdot d^*}{\mu^*} \tag{1}$$

Generally oriented along a fixed direction $\hat{z}$, they evolve through space and time by interacting with the other medium, generating vortices through the development of shear layer instabilities (Kelvin-Helmotz instabilities). The structure of these flows is well known and axisymmetric round jets can be decomposed into several regions depending on the axial distance to their inlet ($z = \frac{z^*}{d^*}$) and on the radial distance relative to their centerline ($r = \frac{r^*}{d^*}$). An illustration is provided below:

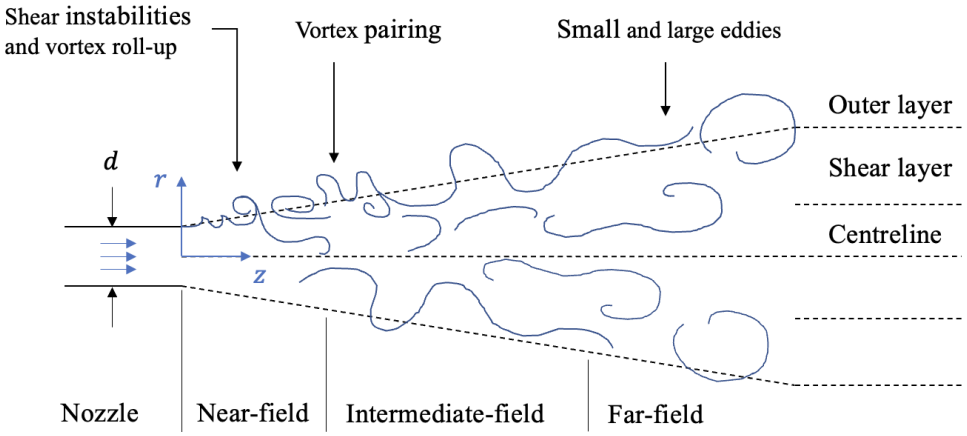

Figure 3.1: Jet flow characteristics and regions

According to Ball et al. [23] and Abdel-Rahman [24], a jet can be decomposed into three main regions: the near-field, the intermediate field and the far-field. The near-field usually lies within $0 \leq z \leq 7$, where the flow mixes with the surrounding fluid and causes the development of turbulence. Vortex roll-up and vortex pairing phenomenons appear in this mixing zone. The fluid then progresses to the intermediate-field, where non-isotropic turbulence structures evolve and interact. Finally, the jet reaches the far-field region $z \geq 70$, where it preserves self-similarity properties: the number of independent variables describing the flow can be reduced and the flow statistics only depend on a certain combination of these variables (Ghosal et al. [26]). Thanks to its reduction in statistical complexity, this region appears to be the most studied region (Wygnanski et al. [27], Hussein et al. [28]).

Jets can further be broken down into three additional regions in the radial direction $\hat{r}$. The closest zone to the $\hat{z}$ axis is called the center-line zone and corresponds to the region where the flow experiences maximum axial mean velocity. The shear layer delimits the zone where vortices develop, evolve and interact to form large eddies. Finally, the outer region corresponds to the domain where the large eddies decay into smaller eddies.

Variable density jets are more complex than canonical jets and allow for variations in density. On top of the continuity and momentum equations needed to describe incompressible flows, these jets require an additional equation for concentration of each species along with an energy equation (Chaissaing et al. [25]).

This study focuses on a turbulent heated jet flow. A numerical method for modelling and simulating this flow is described in the following section.

3.2 Direct Numercial Simulation

Out of different numerical simulation techniques used to solve turbulent heated flows, Direct Numerical Simulation is the most accurate method. Indeed, DNS does not require any modelisation of the flow such as turbulence models introduced in Reynolds-Averaged Navier-Stokes simulations (Blazek [29]). DNS strongly relies on the fluid's equations of motion, governed by Navier-Stokes equations (NS), without any prior assumption. DNS can capture accurate velocity gradients at small scales and achieve this by solving NS on a fine grid. However, this makes DNS computationally expensive compared to other methods and hence less popular.

In this study, accurate DNS data of a 3D turbulent heated jet flow has been acquired from previous work by Karami et al. [3]. The non-dimensionalised equations of motion of the studied flow are listed in equation (3) and include the continuity equation, three momentum equations as well as an energy equation. Further information about the DNS setup is described in [3] and characteristics of the flow are tabulated in section 5.1.1.

The normalisation has been performed according to Karami et al. [3], with superscript '*' representing dimensional values and subscript 'o' being ambient conditions:

$$\begin{aligned} \mathbf{x} &= \frac{\mathbf{x}^*}{d^*},\ t = \frac{t^* a_o^*}{d^*},\ \mathbf{u} = \frac{\mathbf{u}^*}{a_o^*},\ p = \frac{p^*}{\rho_o^* a_o^{*2}},\ T = \frac{T^*}{T_o^*} \\ \rho &= \frac{\rho^*}{\rho_o^*},\ \mu = \frac{\mu^*}{\mu_o^*},\ Pr = \frac{\nu^*}{\alpha^*},\ Re = \frac{\rho_o^* a_o^* d^*}{\mu_o^*} \end{aligned} \tag{2}$$

In the above, $\mathbf{x}$ refers to the spatial vector, t to the time, $\mathbf{u}$ to the velocity vector, p to the pressure, T to the temperature, ρ to the density, μ to the dynamic viscosity given by $\frac{\mu^*}{\mu_o^*} = (\frac{T^*}{T_o^*})^{0.7}$, Pr to the Prandtl number and Re to the Reynolds number. Additionally, a defines the speed of sound and d the jet diameter. The corresponding equations of motion that have been used in the DNS are described below, with ∇ and $\nabla\cdot$ describing the gradient and divergence operators, respectively:

$$\begin{aligned} \frac{\partial \rho}{\partial t} &= -\nabla \cdot (\rho \mathbf{u}) \\ \frac{\partial \rho \mathbf{u}}{\partial t} &= -\nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p\mathbb{I} - \mu\mathbb{T}) \\ \frac{\partial \rho e}{\partial t} &= -\nabla \cdot [\mathbf{u}(\rho e + p) - \mathbf{Q} - \mu\mathbb{T} \cdot \mathbf{u}] \end{aligned} \tag{3}$$

The deviatoric stress tensor, $\mathbb{T}$, the heat flux, $\mathbf{Q}$, and the density-weighted energy are defined in equations (4), with γ specifying the heat capacity ratio:

$$\begin{aligned} \mathbb{T} &= [\nabla \mathbf{u} + (\nabla \mathbf{u})^T - \frac{2}{3}(\nabla \cdot \mathbf{u})\mathbb{I}] \\ \mathbf{Q} &= \mu \frac{\gamma}{\gamma - 1} \frac{1}{Pr} \nabla T \\ \rho e &= \frac{1}{2}\rho \mathbf{u} \cdot \mathbf{u} + \frac{p}{\gamma - 1} \end{aligned} \tag{4}$$

3.3 Incompressibility Assumption

Referring to the equations of motion of a turbulent heated jet in equation (3), when incompressibility is assumed, the evolution equations can be expressed as follows:

$$\begin{aligned} e_1 &: \frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \frac{1}{Pe} \nabla^2 T = 0 \\ e_2 - e_4 &: \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - \frac{1}{Re} \nabla^2 \mathbf{u} = 0 \\ e_5 &: \nabla \cdot \mathbf{u} = 0 \end{aligned} \tag{5}$$

where Pe refers to the Peclet number representing the ratio of the advective transport rate over the diffusive transport rate. This ratio is defined by the product of the Prandtl number and the Reynolds number $Pe = Pr \cdot Re$. In the above, e_1 refers to the heat equation, e_2 to e_4 represent the three-dimensional momentum equations, and e_5 stands for the continuity equation.

It is important to note that the energy term has been removed in equations (5). Moreover, although temperature is not needed to find a solution to an incompressible flow, a heat equation (e_1) has been added to account for available DNS temperature data. Temperature therefore acts as a passive scalar (Warhaft [30]) and does not affect the properties of the incompressible flow.

Since the turbulent heated jet is a round jet, cylindrical coordinates (z, r, θ) have been chosen. The velocity vector $\mathbf{u}$ has been decomposed accordingly into (u_z, u_r, u_θ). Similarily, the momentum equations e_2, e_3 and e_4 have been respectively directed in the axial direction, $\hat{z}$, in the radial direction, $\hat{r}$, and in the azimutal direction, $\hat{\theta}$. The exact formulation used for equations e_1 to e_5 is written in the Appendix, refer to system of equations (26).

4 Numerical Methods

This section explores the main numerical techniques used for the jet flow's model inference. It provides basic understanding on neural networks, on how PINNs are constructed and dives into some essential aspects of neural networks' optimisation.

4.1 Neural Networks

From classification to regression and generation, neural networks span a large range of different tasks and have many different architectures (Multi-Layer Perceptrons, Recurrent Neural Networks, Convolutional Neural Networks, Transformers...). To provide useful background on deep neural networks that have been used in this project, focus will first be made on Multi-Dimensional Perceptrons (MDP) and on Multi-Layer Perceptrons (MLP).

4.1.1 Multi-Layer Perceptrons

Multi-Dimensional Perceptrons are maps taking input data in a certain dimension $\mathbb{R}^C$ and generating multi-dimensional outputs in $\mathbb{R}^D$. They are represented by a linear combination of inputs and composed by an activation function, which is most of the time non-linear. The linear combination itself is performed by multiplication of weights and addition of biases to the inputs. Weights and biases are the model's parameters. Let $f(x; w, b)$ be the MDP applied on input x, and σ be the activation function. Weights are denoted by w and biases by b. This MDP can be formulated as:

$$f(x; w, b) = \sigma(w \cdot x + b),\ w \in \mathbb{R}^{C \times D},\ b \in \mathbb{R}^C \tag{6}$$

Multi-Layer Perceptrons, also called Feed-Forward Networkds (FFN), are the composition of MDPs and extend to several layers. One layer corresponds to one composition of an activation function with the output of the previous layer, whereas the number of neurons in one layer refers to the layer's output dimension. This way, neural networks are characterized by their depth, or number of layers, and their width, or number of neurons per layer. MLPs are referred as deep neural networks when their number of layers are consequent, and these have been used throughout this study. Illustrations are provided on figure 4.1, and a general expression of a L-layer MLP is provided below with superscript 'l' referring to the l-th layer:

$$\begin{aligned} x^{(0)} &= x \\ x^{(l)} &= \sigma(w^{(l-1)} \cdot x^{(l-1)} + b^{(l-1)}),\ \forall l = 1...L \\ f(x; w, b) &= x^{(L)} \end{aligned} \tag{7}$$

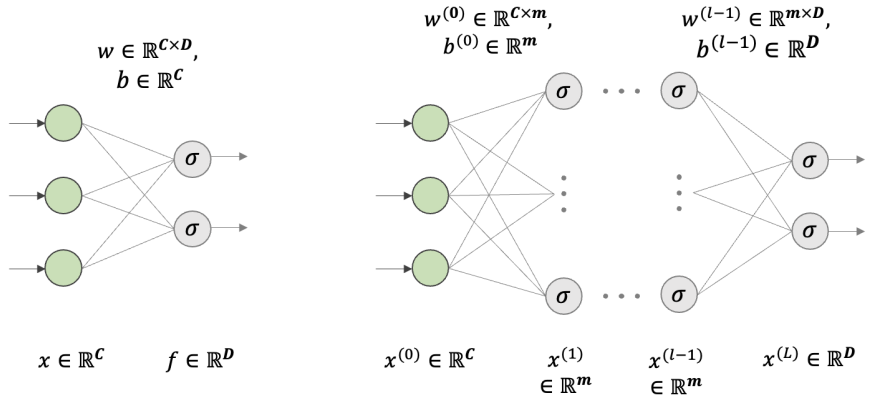

Figure 4.1: Mulit-Dimensional Perceptron (left) and Multi-Layer Perceptron (right), with input dimension C, output dimension D and hidden layers of dimension m

4.1.2 Training a Neural Network

Neural networks are used to optimize an objective function or more commonly called, a loss function. Minimizing the loss function is done via training of the network and involves two processes that are repeated iteratively: a forward pass and a backward pass. During the forward pass, the neural network composes activation functions with weights and biases to transform the inputs. This step corresponds to the mapping of the inputs by the functions described in equations (7). The loss is then usually constructed with the neural network's outputs and with some available data. In order to minimize this loss, the model's weights and biases are updated during the backward pass using gradients. Indeed, thanks to backpropagation or auto-differentiation, gradients of the loss with respect to the model's parameters are efficiently computed using the chain rule (Rumelhart et al. [31]). These gradients are used in the weights and biases' updates following a specific rule defined by the optimiser, refer to section 4.5. Training is performed until convergence is reached after a certain number of iterations. The neural network eventually learns features or relationships already present in the target data, to eventually provide a model that best fits them.

4.1.3 Errors in Neural Networks

According to Lu et al. [32], the error made between the neural network's outputs and the ground truth can be split into three types of errors: an approximation error ε_{app}, a generalisation error ε_{gen}, and an optimisation error ε_{opt}.

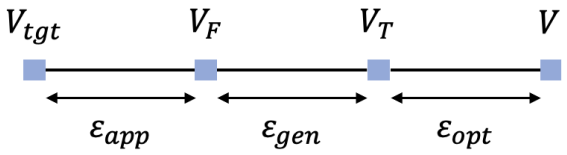

Figure 4.2: Error decomposition in neural networks

Although Lu et al. [32] have shown that a feed-forward network can simultaneously and uniformly approximate any function and its partial derivatives, a neural network is first limited by the family of functions F it can represent, due to its fixed architecture. Indeed, because of its fixed number of neurons, number of layers and activation functions, it can only provide the best function V_F in this family of functions close to the ground truth V_{tgt}. The difference between V_F and V_{tgt} is called the approximation error ε_{app}.

Moreover, since the neural network is trained on a finite dataset T with a limited number of points, it can only best approximate the ground truth on these data points. The error made between the best theoretical approximation on this training set V_T and the best fit in the family of functions F is called the generalization error ε_{gen}.

Finally, an error ε_{opt} needs to be considered during the optimisation process and corresponds to the error between the best theoretical approximation on a fixed training dataset V_T and the actual output of the neural network V. This error derives from the loss function's complexity and the optimisation setup: a neural network is only trained for a fixed number of iterations, with a certain optimizer, for a given loss function. Hence, most of the time during training, a neural network will not be able to find a global optimum and will only reach a local minimum.

4.2 Physics-Informed Neural Networks

Physics-Informed Neural Networks are a subclass of neural networks. They have been introduced by Raissi et al. [13] in 2019 and have been designed to compensate for the lack of available data when training neural networks. They have the potential to solve partial differential equations given fixed model parameters and also allow to solve some inverse problems. In order to achieve this, they take advantage of automatic differentiation already employed in neural networks' backward pass to compute partial derivatives with respect to some inputs. This allows PINNs to accurately calculate derivatives during training while approximating solutions of non-linear partial differential equations.

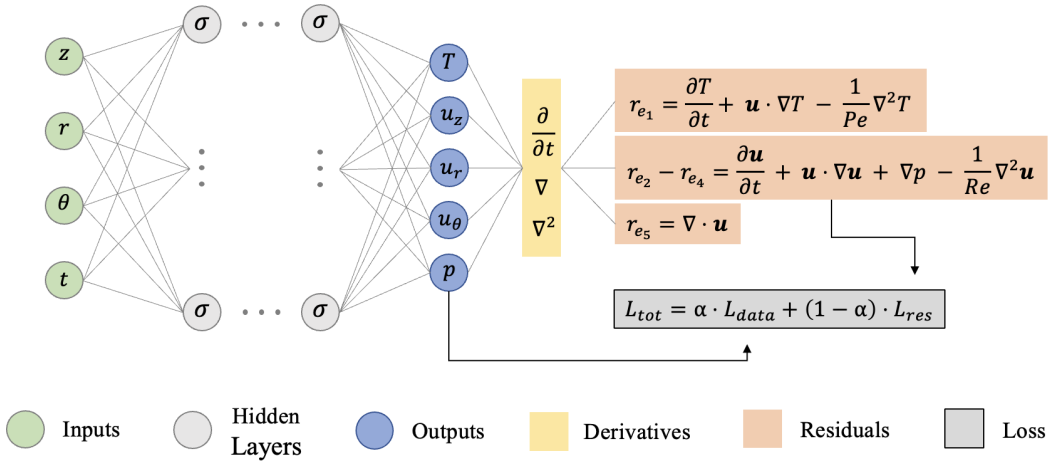

Figure 4.3: Visualisation of the PINN's architecture

PINNs rely on the architecture of feed-forward neural networks, or MLPs, with the inclusion of a particular loss function L_{tot} composed of two terms: a data-driven loss term L_{data}, commonly used in supervised learning methods, and a physics-driven loss term L_{res}. Each of these terms are error indicators on the PINN's ability to fit the provided data values, and on its aptitude to satisfy the physics equations. The main objective of the optimisation is therefore to reduce these errors, and update the weights of the PINN illustrated on figure 4.3, such that the total loss L_{tot} is minimised.

$$L_{tot} = \alpha \cdot L_{data} + (1 - \alpha) \cdot L_{res} \tag{8}$$

In this study, L_{data} and L_{res} have been balanced out using a weighting pre-factor α. The latter has been defined on the $[0, 1]$ interval and has allowed to give more importance to either of the loss terms during training. The total loss L_{tot} expressed in equation (8) is therefore a convex combination of L_{data} and L_{res}. Moreover, these terms have been computed using the Mean Squared Error (MSE), which is the default loss metric used for PINNs, according to Raissi et al. [13]. The MSE loss, or L_2-norm, is formulated in equation (9) by considering an arbitrary domain Γ containing a collection of N data points:

$$\|\cdot\|_{2\ \Gamma} = \frac{1}{N} \sum_{i=1}^{N} (\cdot)^2 \tag{9}$$

In the total loss formulation (8), the data-driven term, denoted by L_{data}, represents the error between the PINN's estimated values V and the known target values V_{tgt}, a priori obtained from DNS data. A general expression for L_{data} is provided in equation (10), with Ω denoting the entire domain of interest, and $\partial\Omega$ referring to the grid points on the boundaries of the domain.

$$L_{data} = \left\| \frac{V - V_{tgt}}{std(V_{tgt})} \right\|_{2\,\Omega} + \left\| \frac{V - V_{tgt}}{std(V_{tgt})} \right\|_{2\,\partial\Omega} \tag{10}$$

Since different optimisations have been investigated, the appropriate data loss for each method is further described in section 5.3. Note that in the above equation, the errors have been normalized by the standard deviation of the target values, $std(V_{tgt})$, to eventually drive each output of the neural network towards the same precision relative to the original target distribution in the DNS data.

The second term, L_{res}, is the residual loss and reflects the error made by the neural network when approximating the physics equations e_1 to e_5. It is important to note that only estimates of the left hand side of e_1 to e_5 can be computed during the optimisation. The difference between these estimations and the true value (zero) of the evolution equations is referred to as residuals. In other words, residuals, denoted by r_{e_1} to r_{e_5}, have been respectively defined by the left hand side of equations e_1 to e_5. Zero residual values would then imply that the corresponding physics equations are satisfied. Hence, minimising L_{res} would eventually reduce the residuals in magnitude and therefore drive the neural network to find an approximated solution of Navier-Stokes' equations.

For this problem, accounting for the convective heat equation e_1 and together with the incompressible Navier-Stokes equations $e_2 - e_5$ described in section 3.3, the residual loss has been defined over domain Ω as:

$$L_{res} = \|r_{e_1,norm}\|_{2\ \Omega} + \|r_{e_2,norm}\|_{2\ \Omega} + \|r_{e_3,norm}\|_{2\ \Omega} + \|r_{e_4,norm}\|_{2\ \Omega} + \|r_{e_5,norm}\|_{2\ \Omega} \tag{11}$$

$$\begin{aligned} r_{e_1,norm} &= \frac{r_{e_1}}{\left\|\frac{\partial T}{\partial t}\right\|_1} \quad r_{e_2,norm} = \frac{r_{e_2}}{\left\|\frac{\partial u_z}{\partial t}\right\|_1} \quad r_{e_3,norm} = \frac{r_{e_3}}{\left\|\frac{\partial u_r}{\partial t}\right\|_1} \\ r_{e_4,norm} &= \frac{r_{e_4}}{\left\|\frac{\partial u_\theta}{\partial t}\right\|_1} \quad r_{e_5,norm} = \frac{r_{e_5}}{\sqrt{\frac{\partial u_r}{\partial r}^2 + (\frac{1}{r}\frac{\partial u_\theta}{\partial \theta})^2 + \frac{\partial u_z}{\partial z}^2}} \end{aligned} \tag{12}$$

The residuals have been normalised with respect to the L_1-norm $\|\cdot\|_1$ of the temporal derivative components in equations e_1 to e_4, as these are the driving terms in the jet flow. For the continuity equation, the normalisation has been performed by the root-mean square of the spatial derivatives' components in equation e_5. These scalings have been chosen such that each term in the above equations have the same importance during the optimization. They have also been applied to improve the balance between the loss terms in equation (8), and to give more meaning to the weighting pre-factor α.

4.3 Input and Output Normalisation

Normalization is an essential step for the convergence of neural networks and has been employed on several occasions. Indeed, according to Chen et al. [33], normalising the neural network's inputs and outputs by the statistics of the flow greatly improves the forward and backward passes, improving the network's ability to model a large range of values present in the data. This is mainly due to the architecture of deep neural networks as well as to the *tanh* activation functions with values defined on the $[-1, 1]$ interval.

4.3.1 Input

Inputs for every dimension (z, r, θ, t) have been normalised with respect to their minimum and maximum values according to equation (13). The transformed neural network inputs, x_{in}, therefore lie within the $[-1, 1]$ interval:

$$x_{in} = \frac{x - (x_{max} + x_{min})/2}{(x_{max} - x_{min})/2} \tag{13}$$

4.3.2 Output

Although outputs from the DNS data are already normalised from a physics point of view, an additional normalization step has been performed on the outputs V_{out} of the neural network. Their values have been scaled down to the $[-1, 1]$ interval using the flow's mean, $\bar{V}_{tgt}$, and standard deviation, $std(V_{tgt})$, evaluated over the studied domain:

$$V = V_{out} \cdot std(V_{tgt}) + \bar{V}_{tgt} \tag{14}$$

4.4 Initialisation

Different types of weights' initialization are used in neural networks but one of the most famous and practical one is Xavier's initialisation method (Glorot et al. [34]). Xavier's initialization aims at controlling the variance of activations across layers of a neural network such that the gradients are prevented from exploding or vanishing. It is a compromise between keeping the variance of activations constant during the forward pass and keeping the variance of gradients with respect to activations constant during the backward pass.
To satisfy the above conditions, and using a uniform distribution for the initialization of the weights in each layer $w^{(l)} \sim U(-s, s)$, the weights' standard deviation s, in each layer l, should be expressed as:

$$s = gain * \sqrt{\frac{6}{n^{(l)} + n^{(l+1)}}} \tag{15}$$

with $n^{(l)}$ corresponding to the number of neurons in layer l, and $n^{(l+1)}$ corresponding to the number of neurons in layer $l + 1$. The *gain* strongly depends on the activation functions used. For example, for *tanh* activation functions, the gain should be set at:

$$gain = \sqrt{\frac{5}{3}} \tag{16}$$

Additionally, biases should be initialized with 0 value across the layers of the neural network.

4.5 Optimiser

The optimiser sets the update rule of the weights and biases during the training phase of a neural network. Three main optimisers are described below: Gradient Descent (GD), Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam).

During training, a neural network iteratively computes the loss for every input sample and updates the weights of the neural network in order to minimise the loss function L. The latter can be written as the sum of all individual loss terms l_n for each input sample n in the training data set containing N samples:

$$L = \sum_{n=1}^{N} l_n \tag{17}$$

Gradient Descent's method uses the gradient of the loss function with respect to the weights, ∇L, in order to update the neural network's weights w. The method was first introduced by Cauchy [35] for solving nonlinear equations. The GD update rule is given below:

$$w_{t+1} = w_t - \eta \nabla L_t(w_t) \tag{18}$$

with subscript t indicating the current iteration, and η referring to the learning rate. Its application to neural networks has been very popular in recent years giving way to several other methods, such as SGD introduced by Robbins et al. [36], and the Adam optimizer by Kingma et al. [37].

Stochastic Gradient Descent is a modified version of GD, where the input samples and the loss are divided into batches. Training a neural network by splitting the input data into mini-batches has appeared to be very efficient in terms of optimization while reducing the training computational cost. At every iteration, SGD randomly selects one mini-batch of input samples of size b, accumulates the losses $l_{n,t}$ for every data in this mini-batch, and then updates the weights of the neural network. SGD repeats the update rule in equation (19) for a certain number of epochs, with one epoch corresponding to the number of iterations performed to account for all batches.

$$w_{t+1} = w_t - \eta \sum_{n=1}^{b} \nabla l_{n,t}(w_t) \tag{19}$$

Because SGD randomly selects input batches at each epoch, it helps the loss evading local minima, allowing to reduce the generalisation error.

Following these advantages, the Adam optimizer is a variant of SGD and additionally exploits the statistics of previous steps to compensate for the anisotropy of gradients across the neural networks layers. This method is an adaptative method which computes individual learning rates for each weight separately by combining momentum with pre-coordinate re-scaling. Adam optimizer is the standard optimizer used in PINNs (Raissi et al. [13]), with few parameters (β_1, β_2), and has been chosen for its great convergence properties.

$$\begin{aligned}
g_t &= \sum_{n=1}^{b} \nabla l_{n,t}(w_t) \\
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
\hat{m}_t &= \frac{m_t}{1-\beta_1} \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{v}_t &= \frac{v_t}{1-\beta_2} \\
w_{t+1} &= w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t
\end{aligned} \tag{20}$$

4.6 Gaussian Noise

To study the influence of noisy target values, refer to sections 5.3 and 6, Gaussian noise with zero mean and unit standard deviation has been applied on temperature target values as well as on velocity and pressure boundary conditions. The added noise is expressed in equation (24) and is inspired from Cai et al. [21], [22]:

$$noise = n_{lvl} * std(V_{tgt}) * Gauss(0, 1) \tag{21}$$

In the former equation, the noise level n_{lvl} is defined by the ratio of the noise amplitude over the standard deviation of each corresponding target term V_{tgt} (temperature, velocity or pressure) over the entire studied domain.

5 Simulation Setup

In this section, the simulation setup is described and includes details on the data, the optimisation methods as well as on the evaluation metrics used for the different simulations.

5.1 Problem definition

5.1.1 Data and Flow Characteristics

The main focus of this thesis is the study of a turbulent heated jet flow at high Reynolds. Thanks to the Laboratory for Turbulence Research in Aerospace and Combustion (LTRAC), high fidelity DNS data has been provided and considered ground truth for every simulation. Details on the DNS are provided in section 3.2 whereas characteristics of the jet flow are listed in the table below:

d	a_0^*	ρ_0^*	T_0^*	γ	Re	Pr
0.015 $[m]$	340.26 $[m/s]$	1.225 $[kg/m^3]$	288.16 $[K]$	1.4	10′000	0.7

Table 5.1: Ambient values and flow characteristics

with d^* the nozzle diameter, a_0^* the ambient speed of sound, ρ_0^* the ambient density, T_0^* the ambient temperature, γ the specific heat ratio, Re the Reynolds number and Pr Prandtl's number. It is important to note that throughout this work, the DNS data obtained has not been modified and has been kept in its non-dimensional form. Furthermore, dimensional results and visualizations of temperature, velocity and pressure profiles in their dimensional forms have been retrieved by multiplication of the aforementioned ambient values, and are illustrated in the Appendix.

5.1.2 Domain

The jet's simulation domain is presented in red on figure 5.1, where the ground truth temperature has been plotted on part of the available DNS grid. A 3D visualisation of the geometry is also illustrated in cylindrical coordinates on figure 5.2, and a reference plane, the rz-plane has been defined at its center, to facilitate observations of flow quantities. For simplification, this domain has been chosen in the incompressibility region of the flow close to the jet axis. This considerably reduced the problem's complexity and enabled to work with incompressible Navier-Stokes equations (26), requiring the computation of fewer derivatives during training. The incompressibility region has been chosen at a specific distance far away from the jet's inlet, such that the average mach number Ma was strictly smaller than 0.3 along the z-axis, on a rθ-plane of diameter d.

5.1.3 Grid

Temperature, velocity and pressure values have been retrieved on each point of the DNS grid, which has been set as the training and evaluation grid for every simulation performed.

Although the mesh is not perfectly uniform and includes different stretches in the $\hat{z}$ and $\hat{r}$ directions, the studied subdomain has been chosen such that $r \leq d$ and such that the non-dimensional spacing between grid points are approximately of the same order of magnitude in space and time. The average spacing of the non-dimensional grid is shown in the table below, for each dimension:

	Δz	Δr	$\Delta \theta$	Δt
Mean grid spacing [-]	0.017	0.007	0.043	0.020

Table 5.2: Mean grid spacing in each dimension

Moreover, due to computational and time limitations, the region of interest has been chosen small relative to the entire available DNS data. Indeed, the domain consists of 96′200 points distributed over 10 time-steps on the $N_z \times N_r \times N_\theta \times N_t = 26 \times 37 \times 10 \times 10$ grid. Its shape along with the grid points' distribution are shown on figure 5.2.

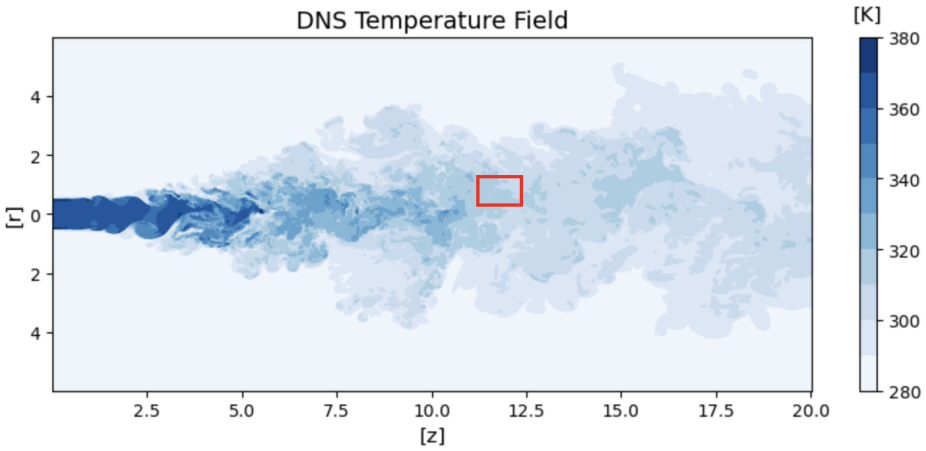

Figure 5.1: Jet's ground truth temperature (rz-plane) and simulation domain (red)

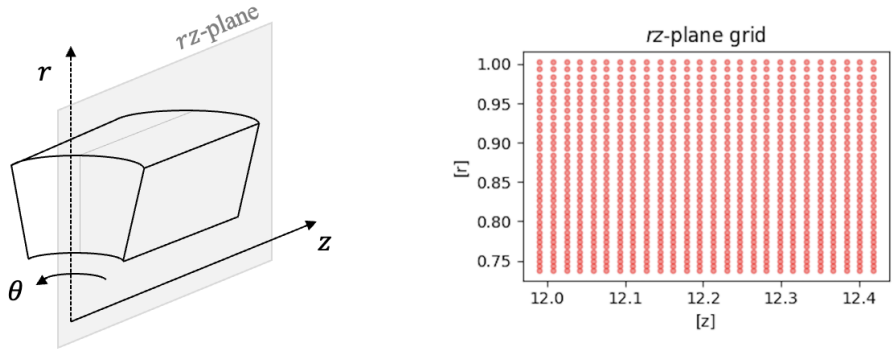

Figure 5.2: 3D visualisation of studied domain (left) and rz-plane grid points (right)

5.2 Model & Parameters

5.2.1 Model

As a balance between expressivity and ability to train a deep neural network (Raghu et al. [38]), the size of the neural network plays a major role in the convergence of neural networks. This is especially true for PINNs which require a large number of weights to reach adequate convergence and accuracy (Raissi et al. [20]). For the purpose of this study, the neural network's width and depth have been fixed according to Raissi et al. [20] and Cai et al. [21], [22], for 3D time-dependent problems. As such, the neural network used in this study consists of 10 hidden layers and contains 100 neurons per layer, representing a total of 91′905 weights that have been updated during the training process.

Although *sin* activation functions may sometimes improve the stability of the convergence (Raissi et al. [20]), *tanh* activation functions have been chosen as they are the most popular activation functions used in PINNs (Jagtap et al. [39], Eivazi et al. [15], Jin et al. [16]), but not only. Thanks to their continuity and non-linearity, they allow for better function approximations (Cybenko [5]) compared to linear functions such as ReLU. Furthermore, according to Hayou et al. [40], smooth activation functions enable better information propagation throughout the training process, improving the overall efficiency of neural networks. Their combination with Xavier's initialisation method is very efficient and allow to reduce the vanishing gradient effect they originally suffer from, by keeping the variance of gradients constant right at the beginning of the optimisation and during training, thus enabling weights to evolve at the same rate (Glorot et al. [34]).

Moreover, the Adam optimizer and the MSE loss have been selected as they are the main optimizer and loss metric used in PINNs. On one hand, Adam optimizer improves the convergence of PINNs compared to standard SGD by updating the learning rates of the weights individually, based on statistics of previous steps to compensate for the anisotropy of gradients (D. P. Kingma et al. [37]). On the other hand, although the MSE loss might generate high errors during the initial phase of training, it avoids the presence of outliers, resulting in a smoother error distribution (M. Raissi et al. [13]). This last aspect is particularly important when boundary conditions have to be considered in order to drive the simulation towards a unique solution.

5.2.2 Simulation Parameters

In order to reduce training time and without compromising the accuracy of the results, every simulation has been run three times for $7 \cdot 10^5$ epochs by considering 3 batches per epoch. The mini-batch size has been adapted according to the corresponding method, refer to section 5.3. The learning rate has initially been set to $1e^{-3}$ and been reduced by a factor of $\frac{1}{3}$ on the following epochs [1′000, 5′000, 10′000, 50′000]; reduction of the learning rate during the optimisation process has shown to dampen the loss oscillations and accurately stabilise the training procedure (Raissi et al. [20]). Each simulation has been performed on a single GPU using the Pytorch library: either A100s or V100s NVIDIA GPUs were utilized, with a minimum configuration of 16GB GPU-RAM.

5.3 Optimisation Methods

During this project, three different optimisation strategies have been investigated. The first strategy consists in inferring the velocity and pressure components by only considering target temperature values on the entire subdomain.
The second method considers additional Dirichlet boundary conditions for velocities and pressure.
Finally, the last strategy applies Gaussian noise on target values for temperature as well as for velocity and pressure boundary conditions. For each specific method, the loss terms and the batch size have been modified accordingly. Additionally, every simulation has been run for two different values of the loss weighting pre-factor $\alpha = (0.9, 0.99)$. High values of α have been chosen according to Cai et al. [21] such that convergence of the solution would be primarily driven by the data values. A noise level of 0.3 has been chosen in the last method for better analogy with the work from Cai et al [22]. All simulations are summarized in table 5.3:

	α_1	α_2	n_{lvl}
Temperature Target Values Only	0.9	0.99	-
Additional Boundary Conditions	0.9	0.99	-
Noisy Target Values	0.9	0.99	0.3

Table 5.3: Simulation parameter study, with α_i, $i = 1, 2$, being loss weighting pre-factors and n_{lvl} corresponding to the noise level

5.3.1 Temperature Target Values Only

Following the total loss formulation in equation 8, if only temperature target values are considered, the normalised data loss for this method is expressed as:

$$L_{data} = \left\| \frac{T - T_{tgt}}{std(T_{tgt})} \right\|_{2\,\Omega} \tag{22}$$

with L_{data} corresponding to the MSE loss (or L_2 norm) of the ratio between the temperature error (PINN's estimated temperature T minus target temperature T_{tgt} provided by the DNS simulation) and the standard deviation of target temperatures $std(T_{tgt})$ evaluated on the studied subdomain Ω.

To satisfy the GPU-RAM capacity described in section 5.2.2, a mini-batch size of $33'000$ points has been chosen for this method, allowing the total $96'200$ data points to be divided into 3 batches.

5.3.2 Additional Boundary Conditions

In addition to target temperature values, boundary conditions have been considered for this second optimisation method. These conditions apply at each time-step on the outer surfaces of the 3D subdomain illustrated on figure 5.2, and have been used as additional constraints in the loss definition to provide more information on velocity and pressure components. It is assumed that the PINN would benefit from those conditions, as it would be able to correctly fit values on the domain boundaries $\partial\Omega$, and force the optimisation to retrieve a unique solution satisfying the governing equations, e_1 to e_5.

As a matter of fact, from an analytical perspective and due to the presence of first and second order derivatives in the 4D evolution equations, e_1 to e_5, constants should be required to retrieve a solution. Indeed, for every flow variable, at least one initial condition and two boundary conditions in each spatial dimension should be taken into account to formally obtain accurate predictions of the flow. However, only imposing conditions on the boundaries of the domain $\partial\Omega$ has shown to be sufficient to generate satisfying results, refer to section 6.

Hence, to account for velocity and pressure boundary conditions, only the data loss term in equation (8) has been modified, by the addition of a boundary loss L_{BC} defined below:

$$\begin{aligned} L_{data} &= \left\| \frac{T - T_{tgt}}{std(T_{tgt})} \right\|_{2\,\Omega} + L_{BC} \\ L_{BC} &= \left\| \frac{u_z - u_{z,tgt}}{std(u_{z,tgt})} \right\|_{2\,\partial\Omega} + \left\| \frac{u_r - u_{r,tgt}}{std(u_{r,tgt})} \right\|_{2\,\partial\Omega} + \left\| \frac{u_\theta - u_{\theta,tgt}}{std(u_{\theta,tgt})} \right\|_{2\,\partial\Omega} + \left\| \frac{p - p_{tgt}}{std(p_{tgt})} \right\|_{2\,\partial\Omega} \end{aligned} \tag{23}$$

In the above, T, u_z, u_r, u_θ and p refer to the PINN's estimated temperature, velocities and pressure terms, whereas T_{tgt}, $u_{z,tgt}$, $u_{r,tgt}$, $u_{\theta,tgt}$ and p_{tgt} are their respective target values. Once again, *std* stands for the standard deviation evaluated over the entire geometry.

Including boundary points in the data loss term has increased the total number of data points to $126'000$. Similar to the previous method, to satisfy the GPU-RAM capacity, the mini-batch size has therefore been set to $42'000$ points.

5.3.3 Noisy Target Values

This last method has been investigated to study the influence of noisy target values on the accuracy of the solution. As described in section 4.6, a Gaussian noise with a noise level of 0.3 has been added to temperature target values as well as to velocities and pressure boundary conditions. For this final method, the total loss formulation is identical to equation (23), except that Gaussian noise has been added to target values at the numerator, denoted by the upper-tilde $\tilde{.}$:

$$\begin{aligned} L_{data} &= \left\| \frac{T - \tilde{T_{tgt}}}{std(T_{tgt})} \right\|_{2\,\Omega} + L_{BC} \\ L_{BC} &= \left\| \frac{u_z - \tilde{u_{z,tgt}}}{std(u_{z,tgt})} \right\|_{2\,\partial\Omega} + \left\| \frac{u_r - \tilde{u_{r,tgt}}}{std(u_{r,tgt})} \right\|_{2\,\partial\Omega} + \left\| \frac{u_\theta - \tilde{u_{\theta,tgt}}}{std(u_{\theta,tgt})} \right\|_{2\,\partial\Omega} + \left\| \frac{p - \tilde{p_{tgt}}}{std(p_{tgt})} \right\|_{2\,\partial\Omega} \end{aligned} \tag{24}$$

Similar to the additional boundary condition method, a mini-batch size of 42′000 points has been selected to account for GPU-RAM capacity and training time.

5.4 Evaluation Metric

For each method described in section 5.3, models have been evaluated on the same grid that has been used for training. In order to compare their accuracy and performance, a quantitative error metric has been defined.

A custom relative error has been chosen as a metric to compare the errors between different simulations. It is defined below and has been evaluated for each output of the PINN on the entire subdomain Ω:

$$\varepsilon_V = \left\| \frac{V - V_{tgt}}{std(V_{tgt})} \right\|_{2\,\Omega} * 100\ [\%] \tag{25}$$

with *std* being the standard deviation over Ω.

Note that the division of the error by the standard deviation of each corresponding target component has been chosen instead of the target values themselves because of the presence of extremely low velocity magnitudes in the radial and azimutal directions (u_r, u_θ). Moreover, dividing the error by the standard deviation of the targets relates directly to the precision of each component originally present in the DNS data, and reflects the turbulent statistics of the flow.

6 Results and Discussion

This section gathers simulations' results for each different method described in chapter 5.3: considering target temperature values only, additional boundary conditions and noisy target values. Illustrations and graphs are provided in the Appendix, and codes are available on the Github repository (`https://github.com/gbl-bessette/Turbulent_Heated_Jet_PINN`).

6.1 Temperature Driven Optimisation

When only taking into account temperature targets in the data loss (refer to section 5.3.1), the PINN model is not capable of correctly inferring velocity and pressure components. Indeed, according to table 6.1, for two configurations of the loss pre-factor α defined in equation (8), none of the u_z, u_r, u_θ or p terms reach a low relative error at the end of the training; the model only achieves accurate estimations of the temperature.

α	ε_T [%]	ε_{u_z} [%]	ε_{u_r} [%]	ε_{u_θ} [%]	ε_p [%]
0.9	0.006	244.60	82.05	81.10	102.86
0.99	0.001	250.55	94.12	89.31	100.80

Table 6.1: Relative errors of each output variable T, u_z, u_r, u_θ and p, averaged over 3 different simulations considering target temperatures only. Errors are listed for α equal to 0.9 and 0.99, at the end of each simulation

As a matter of fact, since all target temperature values are given in the data loss term in equation (22), no additional equation is necessary to drive the PINN's output temperature towards the ground truth. This is illustrated on figure 8.4, where the relative error of temperature decreases during training for both $\alpha = 0.9$ and $\alpha = 0.99$. In this case, a higher value of α gives more weight to the data loss and improves the accuracy of the estimated temperature values. However, relative errors for all other outputs do not decrease during the optimisation. Instead, they reach a plateau close to their initial relative error. This suggests that the PINN is incapable of retrieving the correct velocity and pressure values from the residuals. Although all losses drop from at least three orders of magnitude on figure 8.3, the PINN converges to a solution of the evolution equations, different from the one of the DNS. A visualization of the flow on the rz-plane is provided on figure 8.2 in the Appendix, for $\alpha = 0.99$. On these 2D-plots, we can observe similarities in the estimated velocities that seem to complement themselves and to adapt only for the physics equations, without any direct correspondence to the ground truth. For example, they share the same local maximum and minimum on the rz-plane. For the pressure term, the estimated value converges towards a constant solution: this signifies that the PINN has preferred to set spatial derivatives close to zero in the formulation of the residuals, to facilitate the optimisation process. From a mathematical point of view, the neural network effectively converges towards a solution that is easy to obtain, by eventually compensating the terms inside the residuals such that they would sum up to zero, without providing any physical meaning. Obtaining a trivial solution for PDEs is frequent in PINNs, as explained in Leiteritz et al. [41], and generally underlines

the lack of propagation of information between the grid points. A finer mesh could eventually improve the results, however in our case, the mesh has been selected as the one of the DNS grid, which is assumed to be precise enough to capture accurate flow features. Therefore, the PINN suffers from another issue related to the absence of information on the velocity and pressure terms. Incorporating additional data could force the optimisation to find a suitable optimum, closer to the ground truth.

It appears that as a passive scalar, temperature alone does not provide sufficient information on the flow to drive equations of motion e_1 to e_5 towards the ground truth. Unlike results from the buoyant heated flow in Cai et al. [21], minimising the residual's loss L_{res} for this high Peclet number jet flow, without specifying any other target values, will only make the PINN find a local minimum that is far away from the DNS solution. In fact, the number of unknowns in equations e_1 to e_5 is too big, and the PINN has too much freedom to retrieve the solution provided by the DNS data. As opposed to the flow in Cai et al. [21], the turbulent jet is not driven by temperature nor does it benefit from the buoyancy terms in its momentum equations. These buoyancy components could be extremely helpful in the inference of the velocity and pressure terms, if their values are significant. Another relevant analogy with the buoyant flow [21] is the difference in Reynolds numbers: the jet's turbulent behaviour (high Re) introduces important variations in some flow features, which could locally increase the variance of temporal and spatial derivatives, hindering the residual's optimization and reducing the PINN's ability to find accurate predictions. It is also important to notice that the jet inside the studied domain does not contain any symmetries, as opposed to the 2D heated flow in Cai et al. [22]. The presence of symmetries could reduce the degrees of freedom of the problem and the number of unknowns, facilitating the inference of velocity and pressure components without the need for extra information. A last remark concerns the initialisation method, which could be the most important factor influencing the convergence of the solution towards the ground truth. As mentioned in the previous paragraph, the jet's PINN's initialisation generates outputs that are far from the DNS solution. However, in Cai et al. [22], since the domain is bigger relative to the flow's geometry and since the flow evolves at a slow rate in space and time, velocities are smoother and have smaller values, closer to the ones given by Xavier's initialisation. As such, right from the start of the training, the PINN's initial solution would already be close enough to the true solution, driving the optimisation towards the right optimum.

Thus, for the jet flow, more information on the velocities and pressure should be considered to constrain the optimisation, and eventually drive the PINN's outputs towards a more accurate solution. To this extent, additional boundary conditions have been considered for each term u_z, u_r, u_θ and p, for which the PINN's performance has been evaluated and is presented in the next section.

6.2 Consideration of Additional Boundary Conditions

Adding boundary conditions for u_z, ur, u_θ and p, greatly improves the simulations' results, which are listed in bold in table 6.2. With more information on the jet flow, the PINN is now capable of retrieving accurate solutions inside the studied domain, with lower relative error compared to the previous method, except for the temperature term.

α	Method	ε_T [%]	ε_{u_z} [%]	ε_{u_r} [%]	ε_{u_θ} [%]	ε_p [%]
0.9	Temp. Only	0.006	244.60	82.05	81.10	102.86
0.9	**Add. BC**	**0.06**	**0.04**	**0.06**	**0.10**	**0.21**
0.99	Temp. Only	0.001	250.55	94.12	89.31	100.80
0.99	**Add. BC**	**0.02**	**0.02**	**0.03**	**0.04**	**0.12**

Table 6.2: Relative errors of each output variable T, u_z, u_r, u_θ and p, averaged over 3 different simulations considering target temperatures and additional boundary conditions. Errors are listed for α equal to 0.9 and 0.99, at the end of the simulation

According to figures 8.8 and 8.9 in the Appendix, all losses as well as all relative errors converge. During training, although only boundary conditions are prescribed for velocities and pressure fields, their relative error decreases at the same rate as the one for temperature. The PINN therefore benefits from additional boundary conditions included in the data loss term, in equation (23), to lead the optimisation towards a better optimum. To provide further explanations, with additional boundary conditions, the PINN is now limited in the range of functions it can generate to satisfy Navier-Stokes equations. It does not have the freedom anymore to find any velocity or pressure terms whose values and derivatives satisfy equations e_1 to e_5, and is now obliged to fit a function whose boundary values correspond to the DNS ground truth. This function appears to be a better fit to NS's solution.

Since the simulation domain is small relative to the size of the jet flow, and since boundary values are provided at each time-step, the PINN has enough information to converge to an accurate model of the flow, without the need of specifying an initial condition. Thanks to the minimisation of residuals, it is able to propagate information from the boundaries to the interior of the simulation domain with good accuracy. From table 6.2, we observe relative errors in the orders of 10^{-2} % for the temperature T and for all velocities u_z, u_r and u_θ, whereas the pressure p reaches the maximum relative error in the order of 10^{-1} %, for each configuration of α. A slightly higher error for the azimutal velocity component could be explained by its small variation in distribution in the original DNS data, making the optimisation less sensitive to this variable. From table 6.3, the standard deviation of u_θ is nearly twice as small as the standard deviation of u_z and u_r, which could explain lower variations of u_θ resulting in less important spatial or temporal gradients. This could have an impact during the optimisation process, as r_{e_θ} would be less important in comparison with other residuals at each training epoch, refer to figure 8.11.

$std(T)$ [−]	$std(u_z)$ [−]	$std(u_r)$ [−]	$std(u_\theta)$ [−]	$std(p)$ [−]
0.054	0.028	0.026	0.014	0.010

Table 6.3: Standard deviation of each non-dimensional variable T, u_z, u_r, u_θ and p, originally present in the DNS data and evaluated over the studied domain

For the pressure term, however, the error is higher due to the NS equations themselves, where only its first order spatial derivative is apparent. Because it is only present once in equations e_2, e_3 and e_4, the pressure has less weight in the formulation of the residuals compared to all other velocity components. Moreover, the lack of first order temporal and second order spatial derivatives in the NS equations forces the PINN to propagate pressure information from the boundaries to the inside of the domain by only ensuring its first order spatial derivative is accurate, at each time-step and for every point inside the domain. As opposed to velocity and temperature estimations, the pressure values are thus less guided by the physics equations.

Compared to the previous method, the optimisation is now submitted to additional constraints. This has a cost in approximation and optimisation errors described in section 4.1.3, because it modifies the loss landscape, and affects the temperature's relative error negatively by an order of magnitude: on figure 8.11, the residual of the heat equation r_{e_1} is forced to evolve at the same rate as all other residuals. It is then a compromise between finding the solution of the equations, and being able to fit all of the PINN's output terms simultaneously. Furthermore, during training, the temperature has to adapt for updates in u_z, u_r and u_θ: these velocities directly impact the convective term in the heat transport equation e_1. Indeed, as opposed to the previous method, in this configuration the residual r_{e_1} is affected by the accuracy of the velocity terms. Although the temperature field seems to be driven by the corresponding data loss term, its accuracy is deteriorated by the residual of the heat equation r_{e_1}. Whether the latter actually helps improving the accuracy of velocities and the one of pressure has not been studied in our work, and remains an interesting subject to analyse.

Referring back to table table 6.2, it can be observed that a value of α close to 0.99 improves the accuracy of the results, which is in agreement with Cai et al. [21]. A visualisation is provided on figure 8.7, in the Appendix. This suggests that giving sufficient importance to data during the optimisation improves the model's performance. Indeed, on figure 8.8, the losses for $\alpha = 0.99$ decrease faster compared to the ones for $\alpha = 0.9$, generating lower relative errors at the end of the simulation. Similar to the previous method, setting a low value of α eventually gives more importance to residuals during training. However, residuals can only converge to an accurate solution if boundary and temperature data values are predominant: in other words, a small α increases the optimisation error. Therefore, a fine balance between data loss and residuals needs to be found, for accurate predictions of the flow to be made: this is generally performed using parameter optimisation, but has been avoided in this work due to its time consuming procedure.

Instead, the effect of noisy target values and its impact on the optimisation's results has been investigated, and is the object of the next section.

6.3 Noise Reduction

Following the method described in chapter 5.3, noise has been added on every target value to model experimental noisy data. The relative errors of the PINN's estimations are listed in table 6.4, for a noise level of $n_{lvl} = 0.3$, and results are being compared to the initial error distribution of the Gaussian noise defined in section 4.6.

α	Method	ε_T [%]	ε_{u_z} [%]	ε_{u_r} [%]	ε_{u_θ} [%]	ε_p [%]
–	**initial error**	**9.0**	**9.0**	**9.0**	**9.0**	**9.0**
0.9	Noisy BC	0.41	0.87	1.22	1.14	0.70
0.99	Noisy BC	0.81	2.17	2.88	2.19	1.03

Table 6.4: Relative errors of each output variable T, u_z, u_r, u_θ and p, averaged over 3 different simulations considering noisy target temperatures and noisy boundary conditions, with a noise level of $n_{lvl} = 0.3$. Errors are listed for α equal to 0.9 and 0.99, at iterations corresponding to minimum residual loss: epochs $695'351$ and $36'938$, respectively

These relative errors have been retrieved at a specific epoch, when the residual loss is minimum. As it can be observed on figure 8.14 in the Appendix, the relative errors decrease during the first training iterations. This emphasises the ability of the neural network to converge towards the ground truth right at the start of the optimisation, although noisy data have been provided as targets. The latter have a noise level that is low enough for boundary conditions to be satisfied to some certain extent. Once again, the PINN benefits from boundary conditions to retrieve a solution that is close to the DNS ground truth, although in this case, boundary conditions contain some noise. This is only valid up to a certain number of iterations, after which the relative errors start to increase and to adopt a U-shape: this is more apparent for $\alpha = 0.99$ and is believed to appear for $\alpha = 0.9$ when increasing the number of epochs. This typical shape is well-known in machine learning and illustrates the increase in generalisation error: when the neural network starts to over-fit the target data. Indeed, during training, the optimisation is progressing towards an optimum that is too closely related to the noisy targets, eventually neglecting the residuals. As a matter of fact, on figure 8.14, L_{data} continuously decreases whereas L_{res} starts to increase in the middle of the optimisation, leading to poorer relative errors. To provide further insights, when given noisy boundary conditions and noisy temperature targets, the evolution equations are no longer satisfied, which increases each of the residual loss terms in equation (11). This behaviour is illustrated on figure 8.16, and is noticeable for $\alpha = 0.99$. In these examples, the PINN makes a trade-off between fitting noisy target values and finding a solution satisfying the physics equations of the jet flow. Hence, after some iteration, the PINN's estimations start to degrade relative to the DNS ground truth.

Along with the results listed in table 6.4, relative errors on figure 8.14 also reflect the power of the loss pre-factor α and its influence on the solution. Although a lower value of α was reducing the PINN's accuracy when accounting for raw targets (see results in section 6.2), setting a lower value of α for noisy targets allows the PINN to find a solution closer to the

ground truth, with a relative error that is roughly halved for $\alpha = 0.9$ compared to $\alpha = 0.99$, for each of the flow variables. On figure 8.13, although the number of iterations to reach the minimum residual loss for $\alpha = 0.9$ is twenty times higher than for $\alpha = 0.99$, the physics loss reaches an optimum that is two orders of magnitude lower. A small α in this case would favour the physics over the noise, by finding a better compromise between over-fitting noisy data (high generalisation error) and retrieving a solution close to the NS equations (low optimisation error). A flow visualisation for $\alpha = 0.9$ is provided on figure 8.12 in the Appendix.

Surprisingly, according to table 6.4, the PINN's outputs even reach a lower relative error compared to the initial error present in the noisy data. This result is promising in the use of PINNs for noise reduction, even for turbulent flows, by improving the solution of experimental measurements. Further work should be led in this direction by considering lower values of α and a larger number of iterations for this jet flow.

Hence, it appears in this method that giving less importance to the data loss relative to the residuals in equation (8) leads the optimisation to a better solution, closer to the ground truth. A low value of α, could avoid the PINN to over-fit the noisy data, and allow to drive the model to a better solution.

7 Conclusion

Throughout this work, we have shown that the PINN was not able to infer the velocities and the pressure of a 3D turbulent heated jet, from the temperature field only, on a reduced domain and under the incompressibility assumption. Indeed, apart from temperature estimates, the neural network's outputs have converged to a solution of the evolution equations which differs from the DNS ground truth. The PINN's residuals have been minimised without any physical meaning, with the obtention of trivial solutions for the pressure and similar profiles between the velocities. This has underlined the difficulty of the optimisation to predict accurate results with very few information on the flow. Emphasis has been given on the importance of the PINN's initialisation for the proper inference of the model. In our study, initial velocity and pressure values were too different from the jet's ground truth, hindering the optimisation process, and leading the flow variables towards a non-physical solution.

We have highlighted the relevance of including velocity and pressure boundary conditions in the loss formulation to drive the outputs of the PINN towards a better approximation of the true solution. In this study, the addition of boundary conditions in the data loss term has been detrimental in improving the PINN's accuracy. In the best scenario, relative errors in the order of 10^{-2} % have been retrieved for the temperature as well as for the three velocity components. A relative error in the order of 10^{-1} % has been obtained for the pressure term, due to the only presence of pressure first order spatial derivatives in the evolution equations. We have shown that additional constraints could reduce the optimisation's degrees of freedom to better fit the DNS data, and to avoid trivial solutions. By enforcing boundary conditions during training, information could propagate from the boundaries of the domain (thanks to the data loss term) to the interior of the domain (with the residual loss), to retrieve accurate predictions of the flow. We have drawn the attention to the importance of appropriately balancing the data and the physics in the PINN's total loss formulation, by tuning the weight pre-factor α. A high value ($\alpha = 0.99$) would give more importance to the data relative to the physics equations, and would allow to improve the convergence rate during training.

We have further demonstrated that accurate predictions of the jet flow could also be achieved by feeding noisy targets. Indeed, the PINN has shown to be able to drive the optimisation towards a solution close to the ground truth, although a noise level of $n_{lvl} = 0.3$ had been applied on the target temperature and boundary values. In this configuration, the neural network has started to over-fit the noisy targets after a certain number of iterations. We have shown that this effect could be adjusted by reducing the value of the loss pre-factor α, at the cost of an increase in the number of epochs to reach an optimum. Finally, the PINN has shown to be efficient in terms of noise reduction capabilities, by dropping relative errors initially present in the noisy data to an order of magnitude.

8 Future Work

To further evaluate the methods proposed in this report, similar optimisations should be performed on a bigger domain of interest: both in space and time. This would test the neural network's ability to retrieve a larger range of flow features present in the 3D turbulent jet, by only feeding boundary conditions, and therefore evaluate how well information is transmitted between the outer-surface of the geometry and the interior region. It would also assess the PINN's accuracy in predicting quantities far away from the boundaries, to eventually recover the entire jet flow. This way, an analogy with other numerical methods could be conducted to evaluate the PINN's efficiency. Moreover, an increase in the domain's size would allow to see whether the model described in this work is still valid for different boundary conditions. Note that in this work, since the borders of the geometry have been chosen close to the jet axis, the boundary conditions already provide a lot of information on the flow. Generally, this is not the case when dealing with a larger geometry with velocity inlet boundary conditions and constant pressure boundary conditions far away from the axis. This would most probably increase the degrees of freedom of the optimisation and could affect the solution's accuracy. The PINN's performance could be deteriorated with the possible need to provide more data to reach convergence, such as initial conditions.

Other studies could be led in the prediction of variables in a 3D flow, by only knowing some of its characteristics on a 2D-plane. Indeed, in experimental fluid mechanics, the flow is generally retrieved along 2D-planes by either using the Particle Image Velocimetry technique to measure velocities (Ronald [42]) or using the Background-Oriented Schlieren method to determine the density or temperature field (Shoaib [2]). Hence, PINNs could play a determinant role by estimating the flow's unknowns on the entire 3D region, from the 2D measurements. Other types of optimisation techniques such as progressive methods (Krishnapriyan et al. [14]) could also be investigated in this case, to improve the model's convergence rate for larger domains, by progressively solving in the spatial and temporal dimensions, from known data along a plane to the interior/exterior of the domain.

Additionally, it is important to acknowledge that in our methodology, the outputs of the neural network have been subject to normalisation with respect to statistics of the DNS data values. However, this step cannot be undertaken when the solution is unknown. As such, examining other ways to perform a normalisation on the PINN's outputs would be required to improve the model's convergence. For example, statistics of the flow variables on the boundary conditions could be used as a first approximation, if those are available. Otherwise, another method should be studied.

Furthermore, different techniques could be used for improvements in the stability of the convergence, in its speed, and in the accuracy of the results. As Raissi et al. [20] suggest, sinusoidal activation functions could be used to improve the stability of the convergence and to provide more accurate results compared to *tanh* activation functions. Additionally, the L-BFGS optimiser could be used after a certain number of epochs to decrease the loss even further and to reach its local minima faster. An alternative described by Jagtap et al. [43], could eventually be applied to parallelize the work on several GPUs by splitting the domain

into several regions. Moreover, the loss pre-factor α used in the loss definition of this work could be altered throughout the optimisation by following Wang et. al [17] dynamic weight update's method, depending on gradients of the loss function.

Other fluid mechanics problems with lower Reynolds and Peclet numbers should be studied with the methods employed in this report, to better grasp the limit of their applicability. Indeed, although inferring velocity and pressure components only from temperature values provides satisfactory results for the convective heated flow in Cai et al. [21], other examples involving passive scalars appear to have more difficulties to reach convergence, Raissi et al. [20]. For these problems, further investigations could be led on the simulation setup used. For example, as observed by Raissi et al. [20], using boundary conditions with non-zero gradient with respect to the normal of these boundaries would facilitate convergence.

Finally, further work should be led in the direction of noise reduction with PINNs, as their capabilities are promising and could greatly improve experimental measurements of turbulent fluid flows.

References

[1] Dahm, Werner & Su, Lester & Southerland, Kenneth. (1992) Scalar imaging velocimetry technique for fully resolved four-dimensional vector velocity field measurements in turbulent flows. Physics of Fluids A: Fluid Dynamics. 4. 10.1063/1.858461.

[2] Amjab, Shoaib. (2022) Development of a tomographic background-oriented schlieren method for instantaneous three-dimensional density field measurements of heated jets. Monash University. Thesis. https://doi.org/10.26180/19083560.v1.

[3] S. Karami, P. C. Stegeman, Andrew Ooi, J. Soria. (2019) *High-order accurate large-eddy simulations of compressible viscous flow in cylindrical coordinates*. Computer and Fluids 191.

[4] Ken-Ichi Funahashi. (1989) On the approximate realization of continuous mappings by neural networks. Neural Networks, Volume 2, Issue 3, Pages 183-192, ISSN 0893-6080. https://doi.org/10.1016/0893-6080(89)90003-8.

[5] Cybenko, G. (1989) Approximation by superpositions of a sigmoidal function. Math. Control Signal Systems 2, 303–314. https://doi.org/10.1007/BF02551274.

[6] Kurt Hornik, Maxwell Stinchcombe, Halbert White. (1989) Multilayer feedforward networks are universal approximators. Neural Networks, Volume 2, Issue 5, Pages 359-366, ISSN 0893-6080. https://doi.org/10.1016/0893-6080(89)90020-8.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala. (2019) PyTorch: An Imperative Style, High-Performance Deep Learning Library. 33rd Conference on Neural Information Processing Systems.

[8] Mart'ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. (2016) TensorFlow: A system for large-scale machine learning. Google Brain.

[9] NVIDIA. (2007) CUDA Technology. http://www.nvidia.com/CUDA.

[10] A. U. Mazlan, N.A. Sahabudin, M.A. Remli, N. S. N. Ismail, M. S. Mohamad, H. W. Nies, N. B. Abd Warif. (2021) A Review on Recent Progress in Machine Learning and Deep Learning Methods for Cancer Classification on Gene Expression Data. Processes. 9(8):1466. https://doi.org/10.3390/pr9081466.

[11] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis. (2018) Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience, vol. 2018, Article ID 7068349, 13 pages. https://doi.org/10.1155/2018/7068349.

[12] Hassani, H., Silva, E.S. (2015) *Forecasting with Big Data: A Review*. Ann. Data. Sci. 2, 5–19. https://doi.org/10.1007/s40745-015-0029-9.

[13] M. Raissi, P. Perdikaris, G.E. Karniadakis. (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, Volume 378, Pages 686-707, ISSN 0021-9991. https://doi.org/10.1016/j.jcp.2018.10.045.

[14] Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, Michael W. Mahoney. (2021) Characterizing possible failure modes in physics-informed neural networks. Neural Information Processing Systems.

[15] Eivazi, Hamidreza & Tahani, Mojtaba & Schlatter, Philipp & Vinuesa, Ricardo. (2022) Physics-informed neural networks for solving Reynolds-averaged Navier-Stokes equations. Physics of Fluids. 34. 10.1063/5.0095270.

[16] Xiaowei Jin, Shengze Cai, Hui Li, George Em Karniadakis. (2021) NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. Journal of Computational Physics, Volume 426, 109951, ISSN 0021-9991. https://doi.org/10.1016/j.jcp.2020.109951.

[17] S. Wang, Y. Teng, P. Perdikaris. (2020) Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv preprint arXiv:2001.04536.

[18] Vignesh Gopakumar, Stanislas Pamela, Debasmita Samaddar. (2023) Loss landscape engineering via Data Regulation on PINNs. Machine Learning with Applications, Volume 12, 100464, ISSN 2666-8270. https://doi.org/10.1016/j.mlwa.2023.100464.

[19] Kag, Vijay & Seshasayanan, Kannabiran & Gopinath, Venkatesh. (2022) Physics-informed data based neural networks for two-dimensional turbulence. Physics of Fluids. 34. 10.1063/5.0090050.

[20] Raissi, Maziar & Yazdani, Alireza & Karniadakis, George. (2018) Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data.

[21] Shengze Cai, Zhicheng Wang, Frederik Fuest, Young Jin Jeon, Callum Gray and George Em Karniadakis. (2021) Flow over an espresso cup: inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks. Journal of Fluid Mechanics, Volume 915, A102.

[22] Shengze Cai, Zhicheng Wang, Frederik Fuest, Young Jin Jeon, Callum Gray and George Em Karniadakis. (2021) Supplementary materials. Flow over an espresso cup: inferring 3-D velocity and pressure fields from tomographic background oriented schlieren videos via physics-informed neural networks. Cambridge University Press.

[23] C. G. Ball, H. Fellouah, A. Pollard. (2012) *The flow field in turbulent round free jets*. Progress in Aerospace Sciences, Volume 50, Pages 1-26, ISSN 0376-0421, https://doi.org/10.1016/j.paerosci.2011.10.002.

[24] Abdel-Rahman, Adel. (2010) A Review of Effects of Initial and Boundary Conditions on Turbulent Jets. WSEAS Transactions on Fluid Mechanics. 5.

[25] P. Chaissaing, R. A. Antonia, F. Anselmet, L. Joly, and S. Sarkar. (2002) Variable density fluid turbulence. Fluid Mechanics and its Applications, Vol. 69.

[26] S. Ghosal, M. M. Rogers. (1997) A numerical study of self-similarity in a turbulent plane wake using large-eddy simulation. Phys. Fluids, Vol. 69, No. 6.

[27] I. Wygnanski and H. Fiedler (1969) Some measurements in the self-preserving jet. Journal of Fluid Mechanics, Vol. 38, Part 3.

[28] H. J. Hussein, S. P. Capps and W. K.George. (1994) Velocity measurements in a high-Reynolds-number, momentum-conserving, axisymmetric, turbulent jet. Journal of Fluid Mechanics, Vol. 258.

[29] J. Blazek. (2005) Turbulence Modelling. Computational Fluid Dynamics: Principles and Applications, Second Edition.

[30] Z. Warhaft. (2000) Passive Scalars In Turbulent Flows. Annual Review of Fluid Mechanics.

[31] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. (1986) *Learning representations by back-propagating errors.* Learning representations by back-propagating errors. Nature 323, 533–536.

[32] Lu Lu, Xuhui Meng, Zhiping Mao, and George E. Karniadakis. (2019) Deepxde: A deep learning library for solving differential equations. CoRR, abs/1907.04502. `http://arxiv.org/abs/1907.04502`.

[33] Wenqian Chen, Qian Wang, Jan S. Hesthaven, Chuhua Zhang. (2021) Physics-informed machine learning for reduced-order modeling of nonlinear problems. Journal of Computational Physics 446 110666.

[34] Xavier Glorot and Yoshua Bengio. (2010) *Understanding the difficulty of training deep feedforward neural networks.* Appearing in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), Chia La- guna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

[35] A. L. Cauchy. (1847). Methode generale pour la resolution des systemes d'Equations simultanees. Comptes rendus, Ac. Sci. Paris, 25, 536-538.

[36] H. Robbins and S. Monro. (1951) A Stochastic Approximation Method. Ann. Math. Statist. 22(3): 400-407.

[37] Diederik P. Kingma and Jimmy Lei Ba. (2015) Adam: A Method for Stochastic Optimization. 3rd International Conference for Learning Representations, San Diego.

[38] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli and Jascha Sohl-Dickstein. (2017) On the Expressive Power of Deep Neural Networks. Proceedings of the 34th International Conference on Machine Learning, PMLR 70:2847-2854.

[39] Ameya D. Jagtap, Ehsan Kharazmi, George Em Karniadakis. (2020) Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. Computer Methods in Applied Mechanics and Engineering, 365:113028.

[40] Soufiane Hayou, Arnaud Doucet and Judith Rousseau. (2019) *On the Impact of the Activation Function on Deep Neural Networks Training*. Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97.

[41] Leiteritz, R., and Pflüger, D. (2021) How to Avoid Trivial Solutions in Physics-Informed Neural Networks. ArXiv, abs/2112.05620.

[42] Adrian, Ronald. (2005) Twenty years of particle image velocimetry. Experiments in Fluids. 39. 159-169. 10.1007/s00348-005-0991-7.

[43] Ameya D. Jagtap, George Em Karniadakis. (2020) Extended physics-informed neural networks (XPINNs) : A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Commun. Comput. Phys., 28, pp. 2002-2041.

Appendix

Assuming incompressibility and accounting for the heat equation, the equations for the turbulent heated jet e_1 to e_5, that have been used in this study, are written below in cylindrical coordinates (z, r, θ): e_1 refers to the heat equation, e_2 to e_4 correspond to the momentum equations and e_5 is the continuity equation. Associated residuals, r_{e_1} to r_{e_5}, correspond to the left hand side of each of these equations.

$$
\begin{aligned}
e_1 &: \frac{\partial T}{\partial t} + u_r\frac{\partial T}{\partial r} + \frac{1}{r}u_\theta\frac{\partial T}{\partial \theta} + u_z\frac{\partial T}{\partial z} - \frac{1}{Pe}(\frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} + \frac{\partial^2 T}{\partial z^2}) = 0 \\
e_2 &: \frac{\partial u_z}{\partial t} + u_r\frac{\partial u_z}{\partial r} + \frac{1}{r}u_\theta\frac{\partial u_z}{\partial \theta} + u_z\frac{\partial u_z}{\partial z} + \frac{\partial p}{\partial z} - \frac{1}{Re}(\frac{1}{r}\frac{\partial u_z}{\partial r} + \frac{\partial^2 u_z}{\partial r^2} + \frac{1}{r^2}\frac{\partial^2 u_z}{\partial \theta^2} + \frac{\partial^2 u_z}{\partial z^2}) = 0 \\
e_3 &: \frac{\partial u_r}{\partial t} + u_r\frac{\partial u_r}{\partial r} + \frac{1}{r}u_\theta\frac{\partial u_r}{\partial \theta} + u_z\frac{\partial u_r}{\partial z} - \frac{u_\theta^2}{r} + \frac{\partial p}{\partial r} - \frac{1}{Re}(\frac{1}{r}\frac{\partial u_r}{\partial r} + \frac{\partial u_r}{\partial^2 r^2} + \frac{1}{r^2}(\frac{\partial^2 u_r}{\partial \theta^2} - u_r - 2\frac{\partial u_\theta}{\partial \theta}) + \frac{\partial^2 u_r}{\partial z^2}) = 0 \\
e_4 &: \frac{\partial u_\theta}{\partial t} + u_r\frac{\partial u_\theta}{\partial r} + \frac{1}{r}u_\theta\frac{\partial u_\theta}{\partial \theta} + u_z\frac{\partial u_\theta}{\partial z} + u_r\frac{u_\theta}{r} + \frac{1}{r}\frac{\partial p}{\partial \theta} - \frac{1}{Re}(\frac{1}{r}\frac{\partial u_\theta}{\partial r} + \frac{\partial^2 u_\theta}{\partial r^2} + \frac{1}{r^2}(\frac{\partial^2 u_\theta}{\partial \theta^2} - u_\theta + 2\frac{\partial u_r}{\partial \theta}) + \frac{\partial^2 u_\theta}{\partial z^2}) = 0 \\
e_5 &: \frac{\partial u_r}{\partial r} + \frac{u_r}{r} + \frac{1}{r}\frac{\partial u_\theta}{\partial \theta} + \frac{\partial u_z}{\partial z} = 0
\end{aligned}
\tag{26}
$$

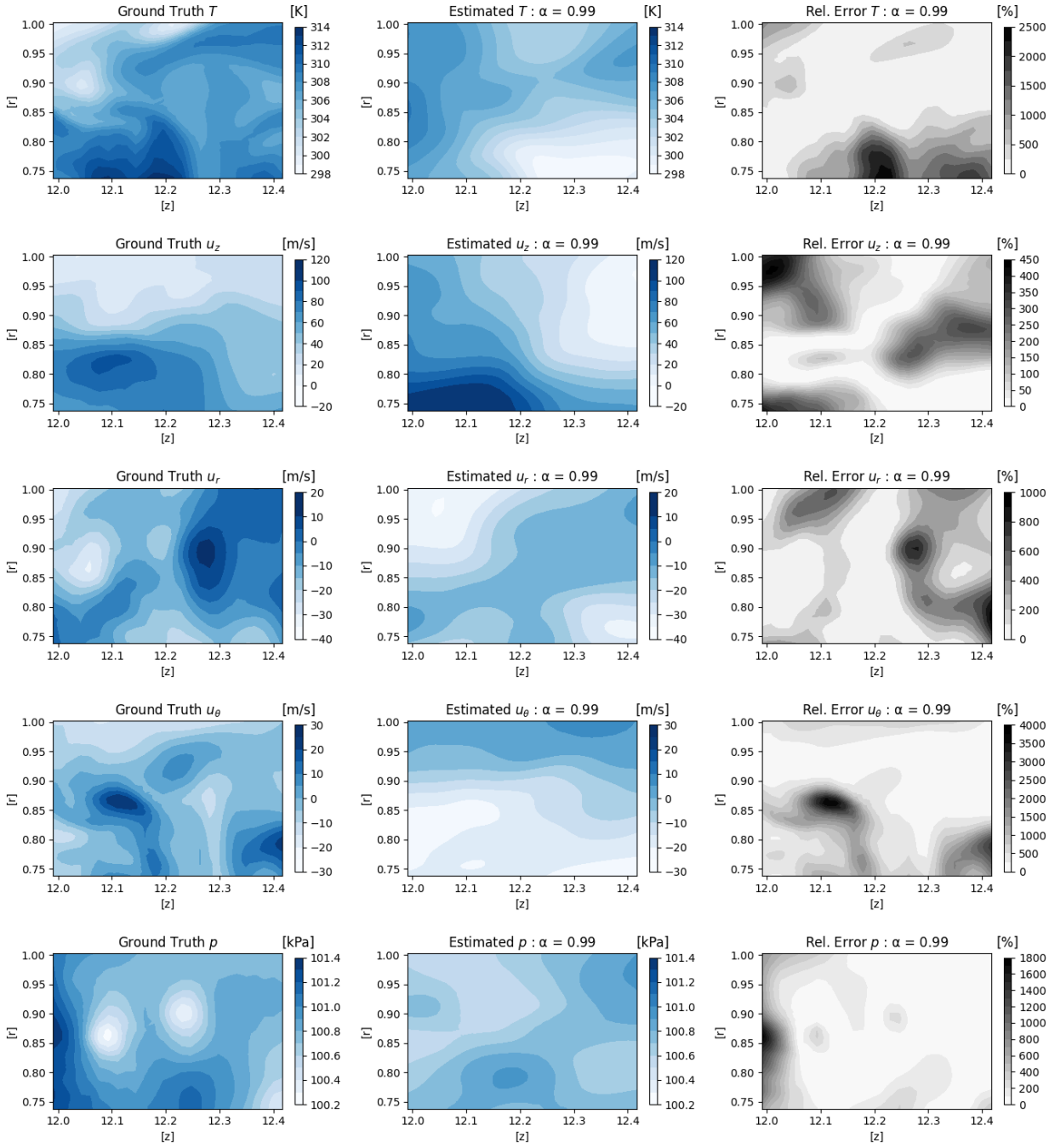

Figure 8.1: Temperature targets only: $\alpha = 0.99$. Ground truth (left), estimated value (center), and relative error (right) for each flow variable, T, u_z, u_r, u_θ, p, evaluated on the rz-plane at the last time-step, at the first iteration after initialisation

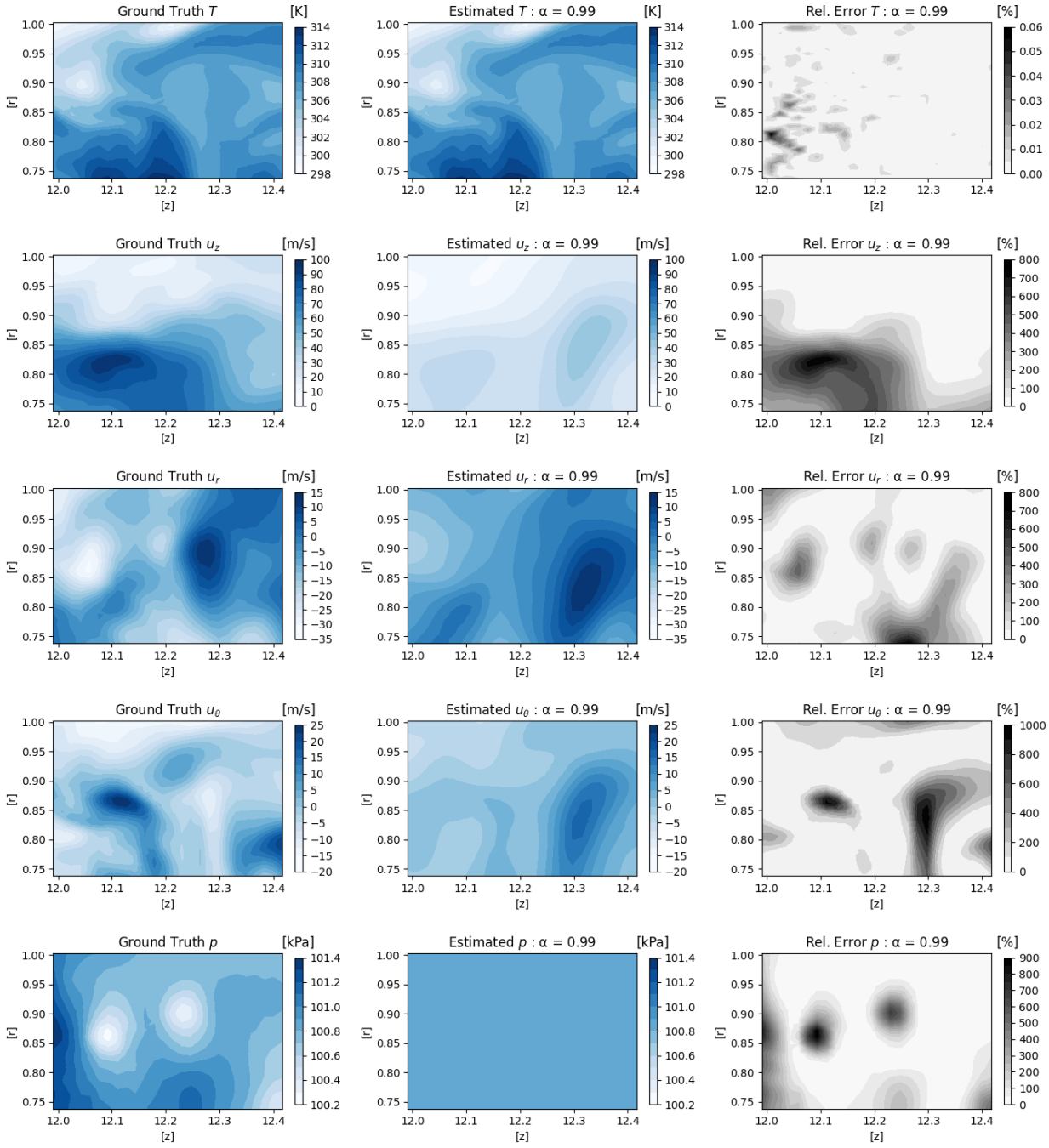

Figure 8.2: Temperature targets only: $\alpha = 0.99$. Ground truth (left), estimated value (center), and relative error (right) for each flow variable, T, u_z, u_r, u_θ, p, evaluated on the rz-plane at the last time-step, at the end of the simulation

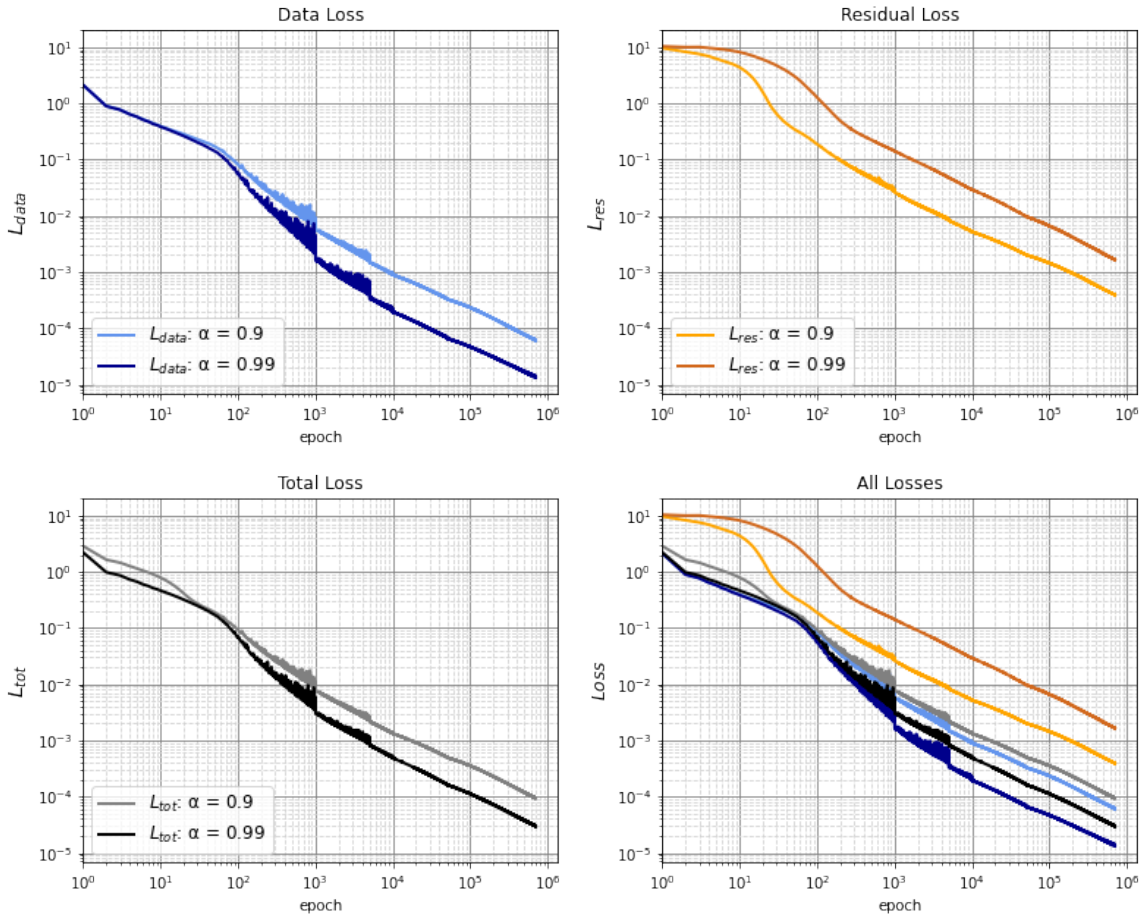

Figure 8.3: Temperature targets only. Average over three simulations of the data loss (top-left), residual loss (top-right), total loss (bottom-left) and summary of all losses (bottom-right) for $\alpha = (0.9,\ 0.99)$ during training

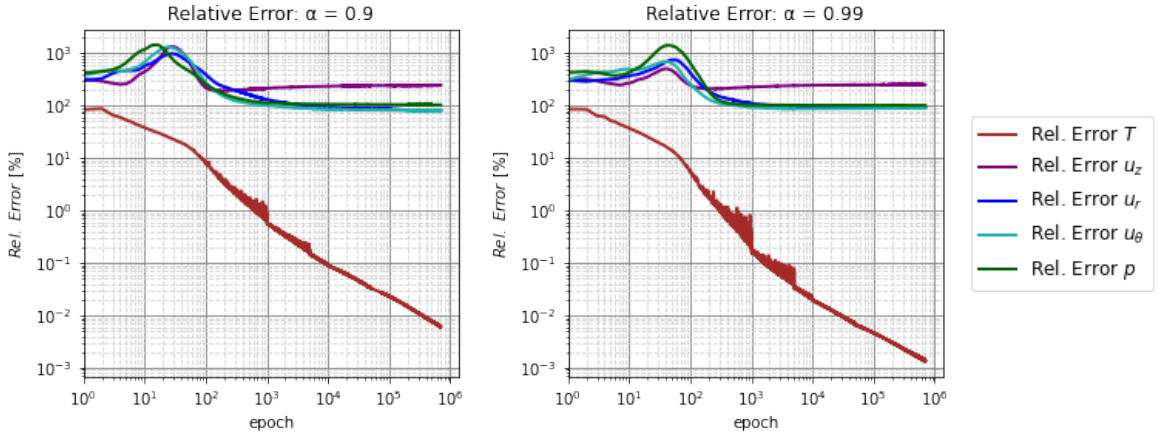

Figure 8.4: Temperature targets only. Average relative errors for each flow variable T, u_z, u_r, u_θ, p, at $\alpha = (0.9,\ 0.99)$, evaluated over the three simulations

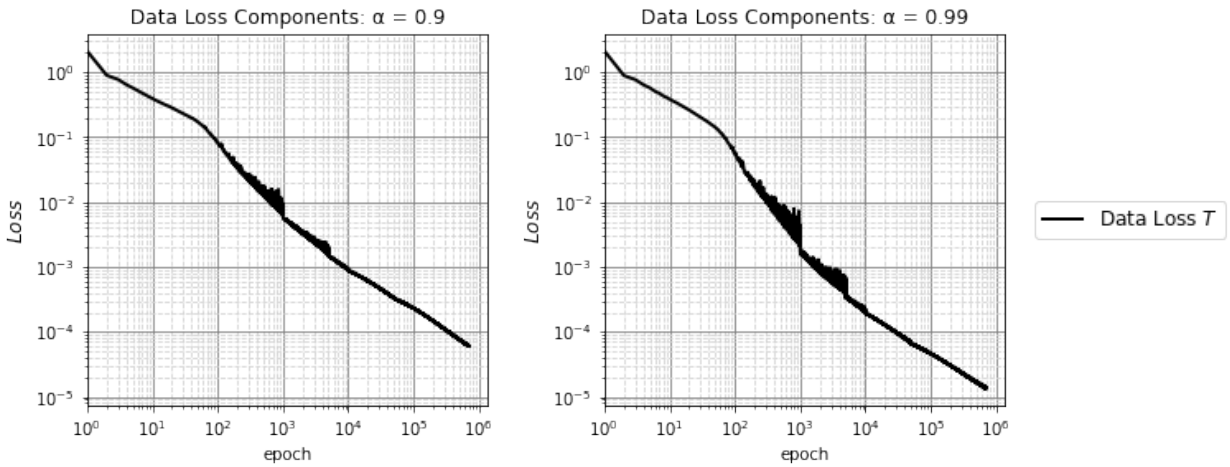

Figure 8.5: Temperature targets only. Average over three simulations of the data loss components for $\alpha = 0.9$ (left) and $\alpha = 0.99$ (right)

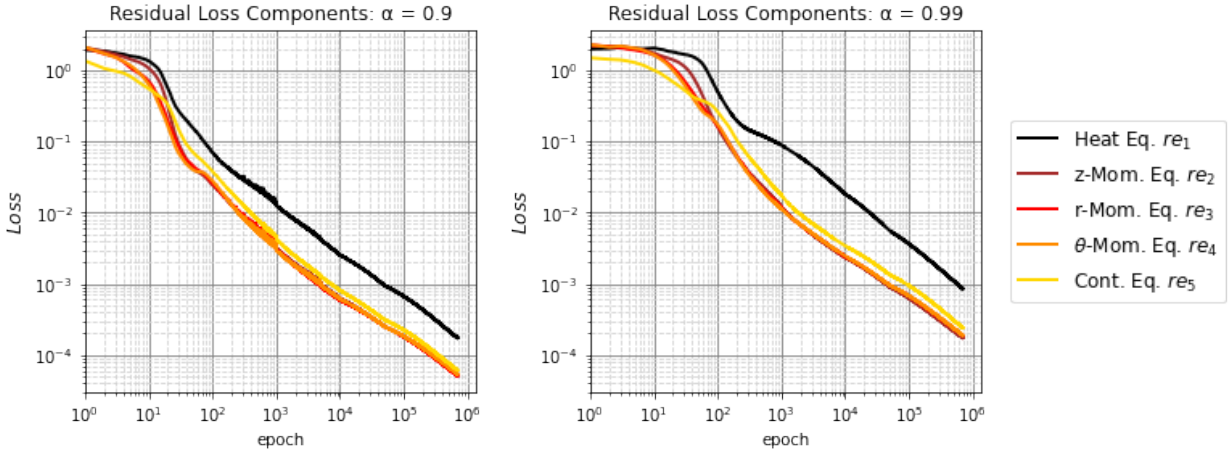

Figure 8.6: Temperature targets only. Average over three simulations of the residual loss components for $\alpha = 0.9$ (left) and $\alpha = 0.99$ (right)

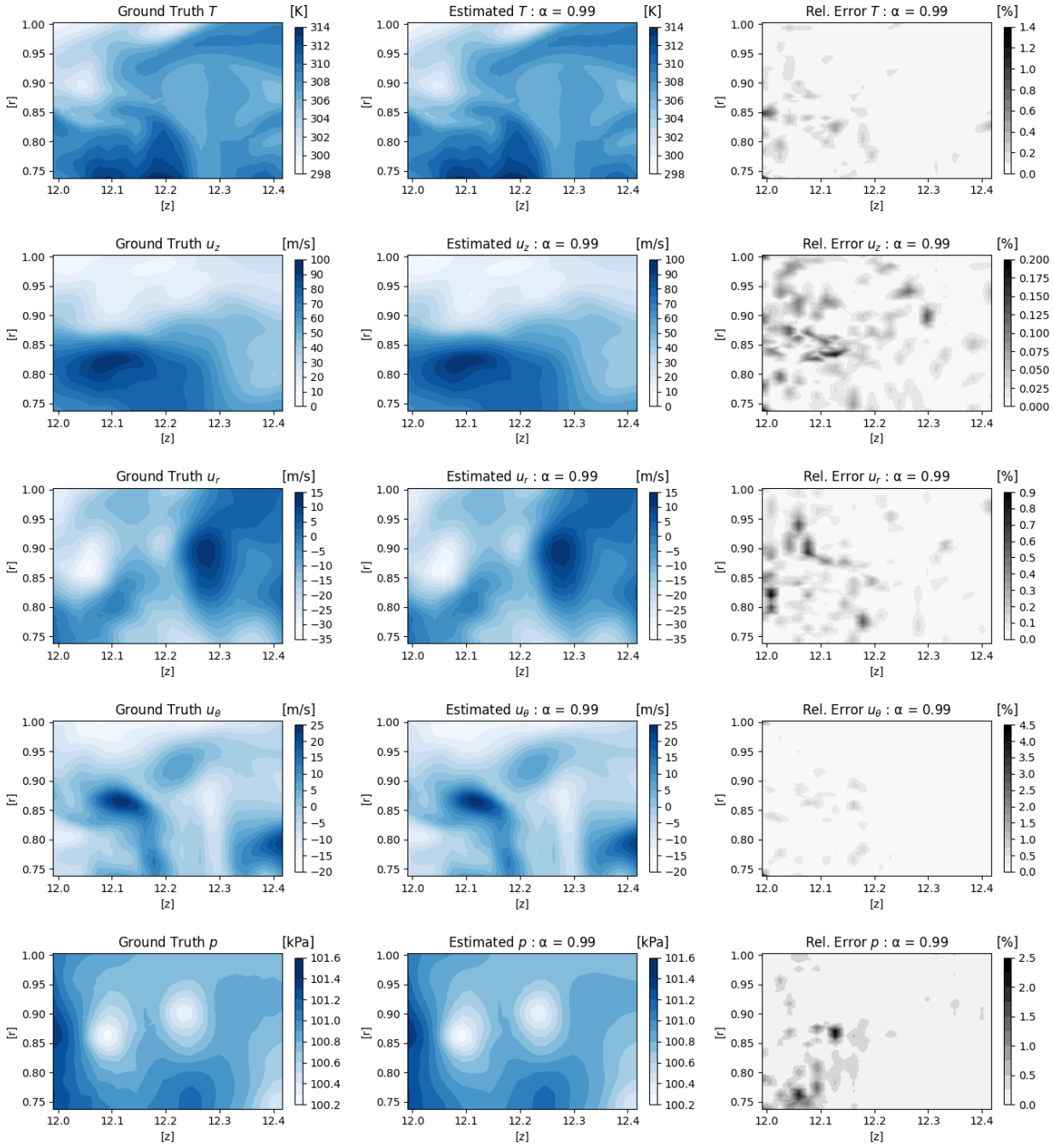

Figure 8.7: Additional boundary conditions: $\alpha = 0.99$. Ground truth (left), estimated value (center), and relative error (right) for each flow variable, T, u_z, u_r, u_θ, p, evaluated on the rz-plane at the last time-step, at the end of the simulation

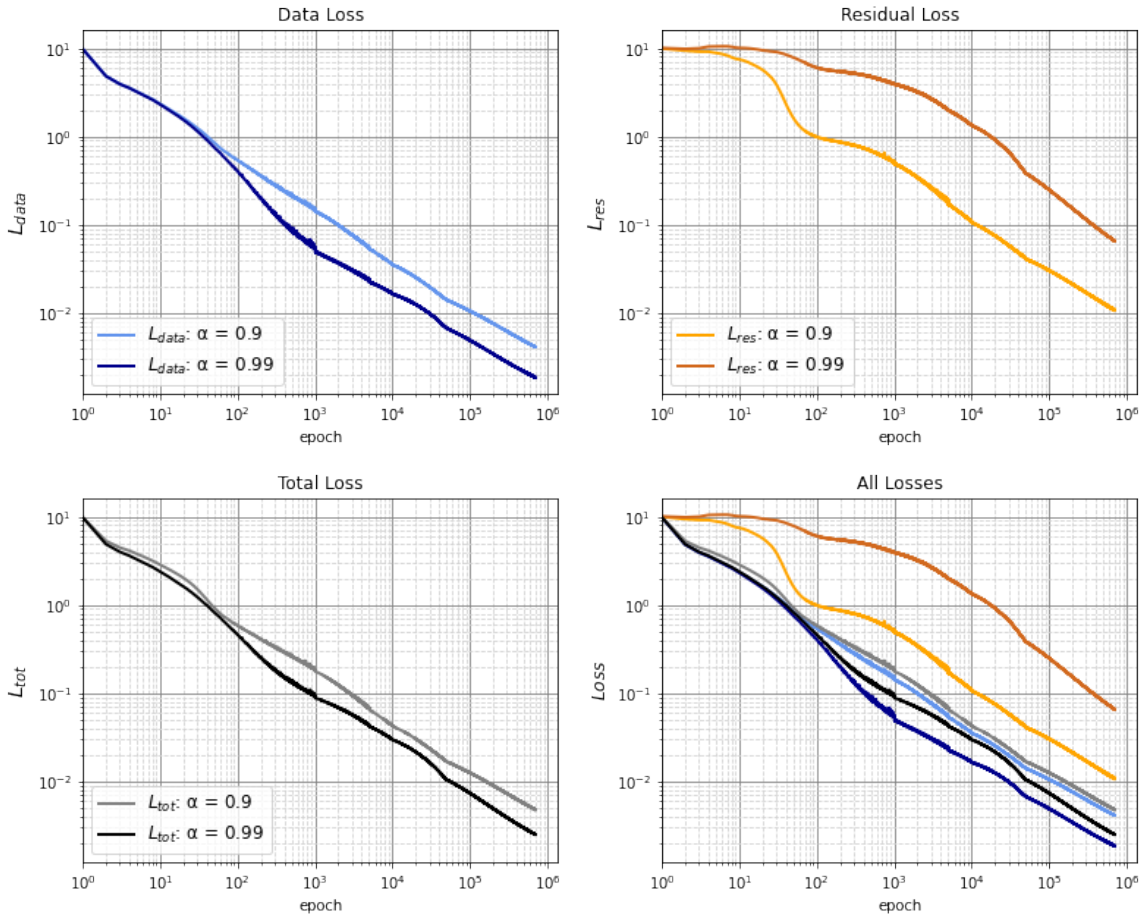

Figure 8.8: Additional boundary conditions. Average over three simulations of the data loss (top-left), residual loss (top-right), total loss (bottom-left) and summary of all losses (bottom-right) for $\alpha = (0.9,\ 0.99)$ during training

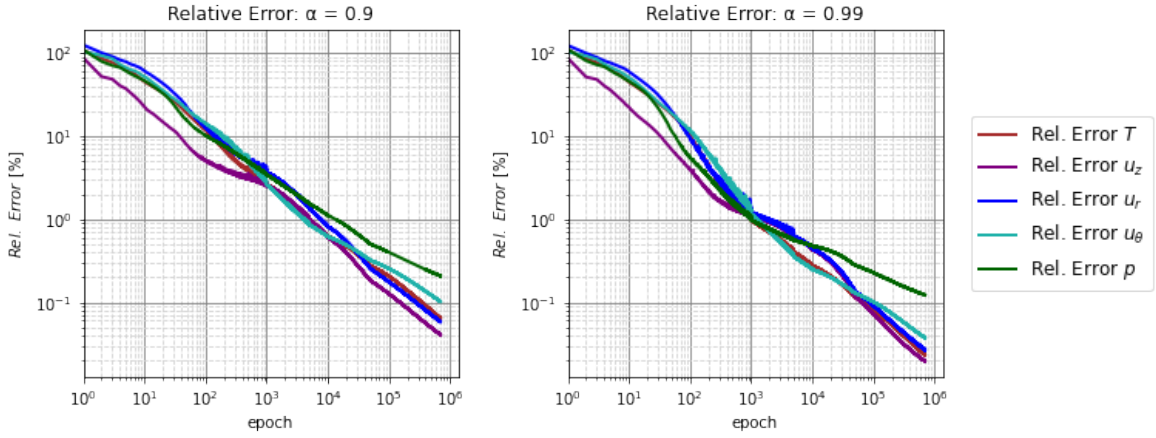

Figure 8.9: Additional boundary conditions. Average relative errors for each flow variable T, u_z, u_r, u_θ, p, at $\alpha = (0.9,\ 0.99)$, evaluated over the three simulations

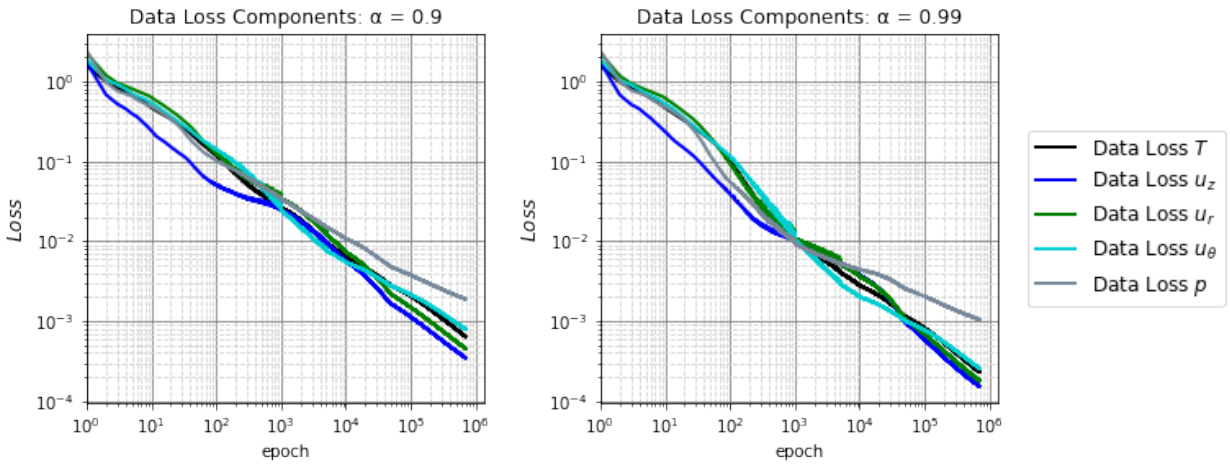

Figure 8.10: Additional boundary conditions. Average over three simulations of the data loss components for $\alpha = 0.9$ (left) and $\alpha = 0.99$ (right)

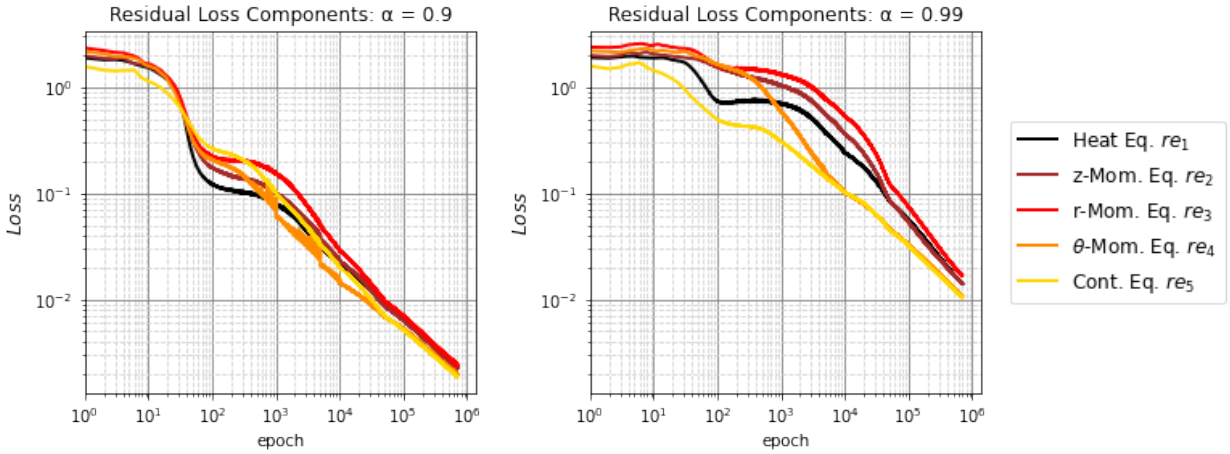

Figure 8.11: Additional boundary conditions. Average over three simulations of the residual loss components for $\alpha = 0.9$ (left) and $\alpha = 0.99$ (right)

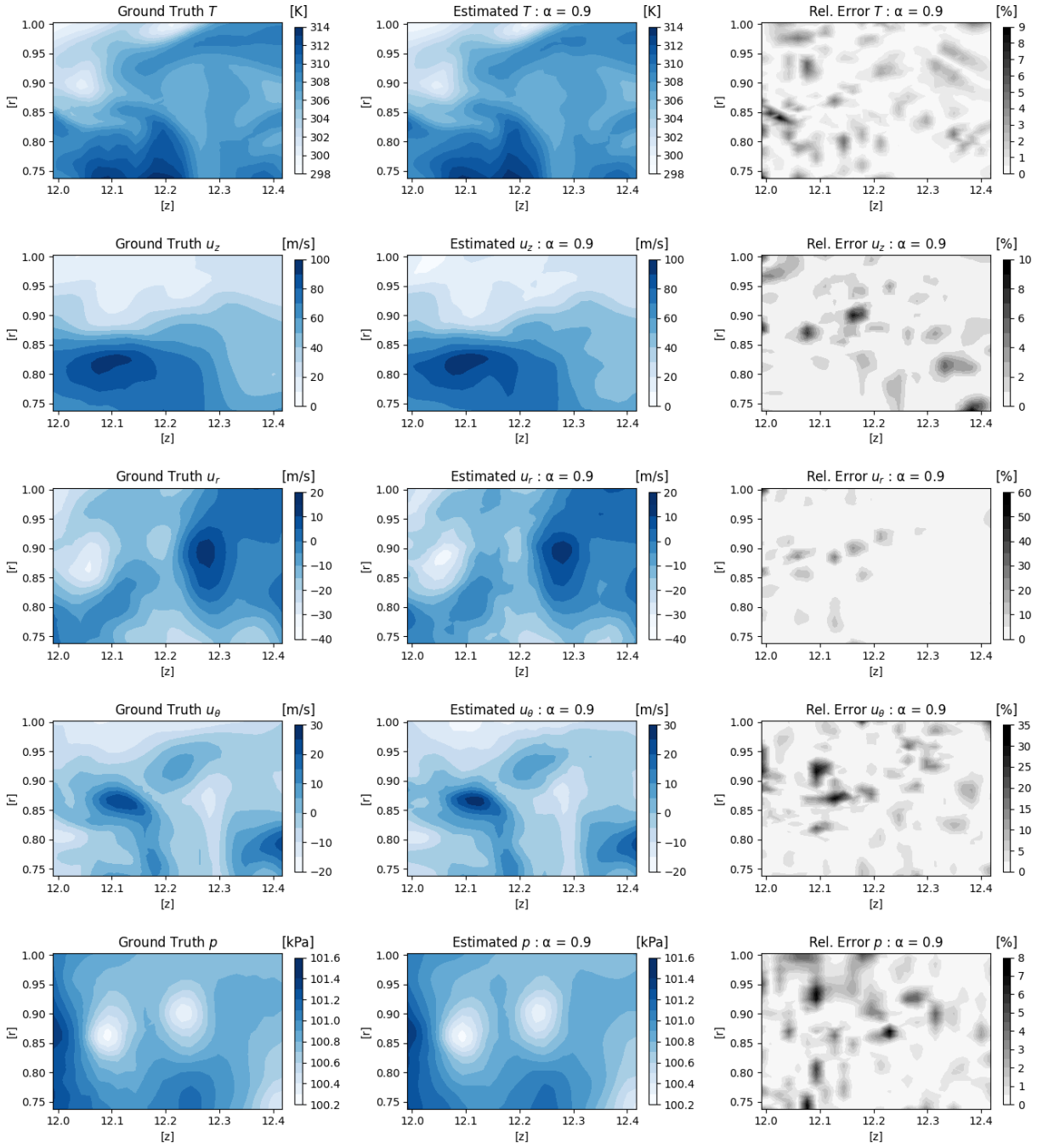

Figure 8.12: Noisy targets: $\alpha = 0.9$. Ground truth (left), estimated value (center), and relative error (right) for each flow variable, T, u_z, u_r, u_θ, p, evaluated on the rz-plane at the last time-step, at epoch corresponding to minimum residual loss

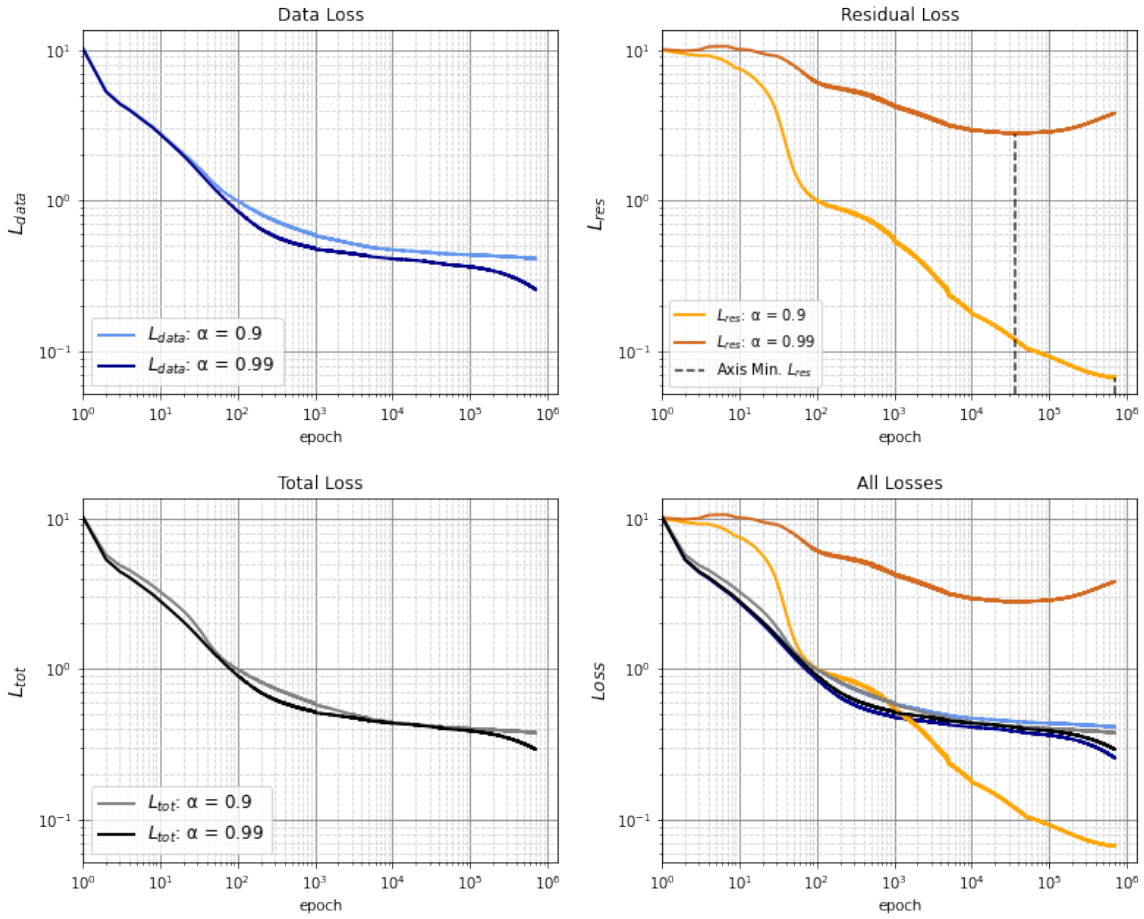

Figure 8.13: Noisy targets. Average over three simulations of the data loss (top-left), residual loss (top-right), total loss (bottom-left) and summary of all losses (bottom-right) for $\alpha = (0.9,\ 0.99)$ during training

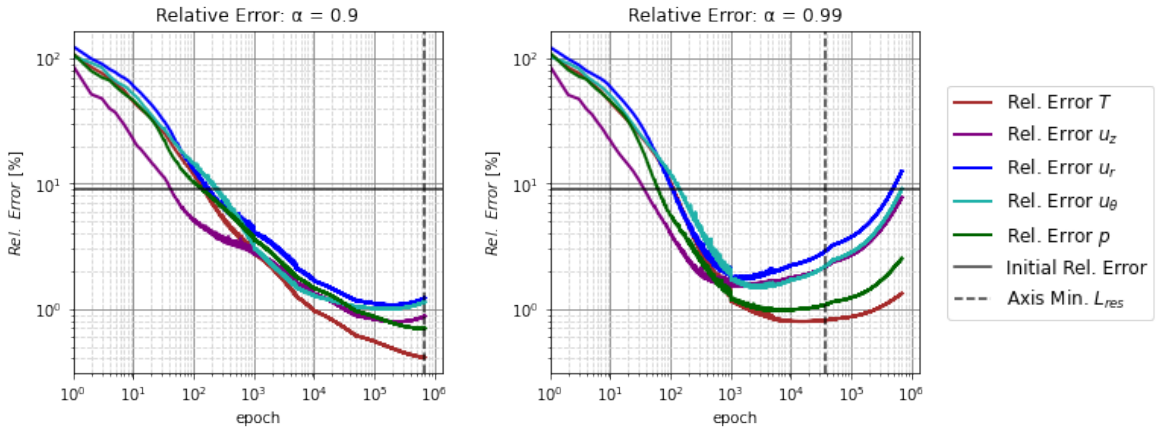

Figure 8.14: Noisy targets. Average relative errors for each flow variable T, u_z, u_r, u_θ, p, at $\alpha = (0.9,\ 0.99)$, evaluated over the three simulations

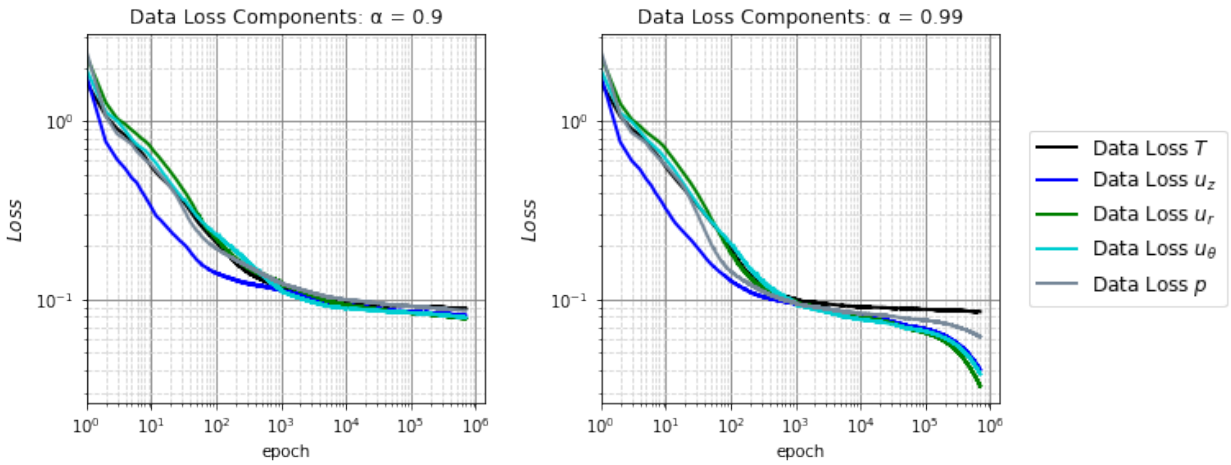

Figure 8.15: Noisy targets. Average over three simulations of the data loss components for $\alpha = 0.9$ (left) and $\alpha = 0.99$ (right)

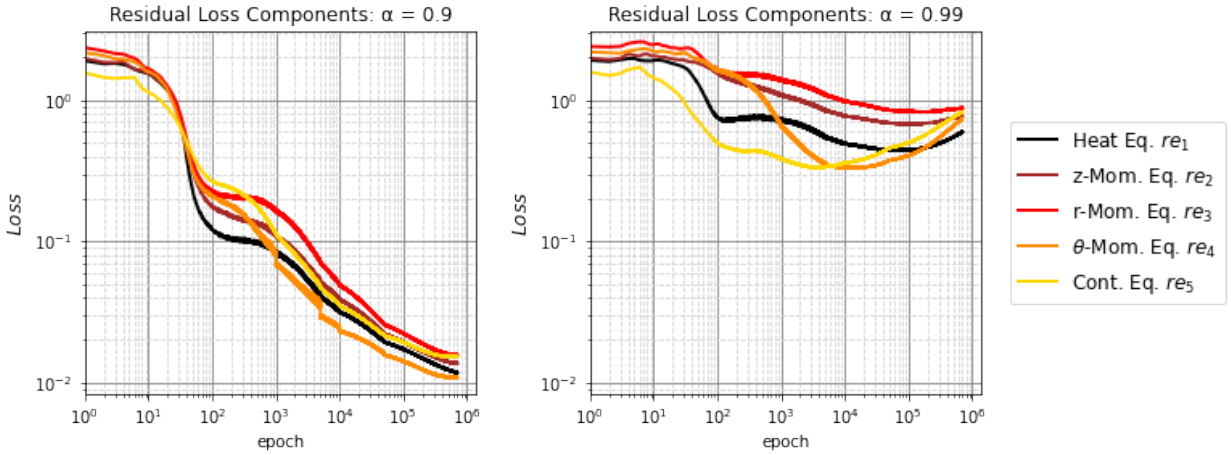

Figure 8.16: Noisy targets. Average over three simulations of the residual loss components for $\alpha = 0.9$ (left) and $\alpha = 0.99$ (right)