

Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection

Didier Lucor^{a,*}, Atul Agrawal^{b,a}, Anne Sergent^{a,c}

^a*Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, F91405 Orsay, France*

^b*Department of Mechanical Engineering, Technical University of Munich, Boltzmannstraße 15, 85748 Garching b. München, Germany*

^c*Sorbonne Université, Faculté des Sciences et Ingénierie, UFR Ingénierie, F-75005 Paris, France*

Abstract

The high expressivity and agility of physics-informed neural networks (PINNs) make them promising candidates for full fluid flow PDE modeling. An important question is whether this new paradigm, exempt from the traditional notion of discretization of the underlying operators very much connected to the flow scales resolution, is capable of sustaining high levels of turbulence. Another concern is whether it can be used as numerical substitutes to full DNS data storage; DNS remaining so far the standard tool for validation and inter-comparison with experimental results.

We explore the use of PINNs surrogate modeling for turbulent Rayleigh-Bénard convection flows, mainly relying on DNS temperature data from the fluid bulk and velocity data at some fluid boundaries. This technique depends on the minimization of a composite loss-function relying on labels and PDE residuals. We demonstrate the large computational requirements under which PINNs are capable of accurately recovering the flow hidden quantities. We then propose new techniques to mitigate the need for large training datasets.

*Corresponding author

Email addresses: `didier.lucor@lisn.upsaclay.fr` (Didier Lucor), `atul.agrawal@tum.de` (Atul Agrawal), `anne.sergent@sorbonne-universite.fr` (Anne Sergent)

First, we propose a padding technique to better distribute some of the scattered coordinates at which PDE residuals are minimized, in particular in zones where no labels are available. We show how it comes to play as a regularization close to the training boundaries and results in a noticeable global accuracy improvement at iso-budget. We then propose a relaxation of the incompressibility condition involved in the loss function contribution related to the PDE residuals. This development drastically benefits the optimization search and results in a much improved convergence. The results obtained at $Ra = 2 \cdot 10^9$ are particularly impressive: the predictive accuracy of the surrogate over the entire half a billion DNS coordinates yields errors for all flow variables ranging between $[0.3\% - 4\%]$ in the relative L_2 norm, with a training relying only on 0.32% of the full DNS dataset.

Keywords: deep learning, machine learning, PINNs, DNS, turbulence, convection

1. Introduction

Deep learning (DL) is investigated among data-driven methods as a surrogate for physics-driven computational fluid dynamics (CFD) methods solving expensive nonlinear coupled PDEs, such as the ones describing turbulent numerical simulations or experiments. It seems to be somehow capable of producing realistic instantaneous flow fields with reasonable physically accurate spatio-temporal coherence, without explicitly solving the actual partial differential equations (PDEs) governing the system [1, 2]. DL is also promising because of its proficiency in extracting low-dimensional information from large amount of high-dimensional turbulent data. This new paradigm is interesting for applications involving flow optimization and control, uncertainty quantification, gappy data reconstruction or multi-scale flow analysis, for which the turbulent prediction may be queried in real-time or many times. In practice, DL models are solved with streams of data as “black boxes” [3]. In their standard form, they lack knowledge of the underlying physics and when they achieve low prediction

errors their efficiency remains hard to interpret. In fact, they do not necessarily satisfy the physics of the systems they model. It is therefore crucial to inject some known physics and principles into their framework, not only to get more physically meaningful results but also in order to better guide the learning process [4]. Besides, the physical invariants may help in recovering hidden system quantities for which no data were available, which is very common in experiments. While machine learning methods make sense as data-driven closure models for Reynolds-Averaged Navier Stokes (RANS) [5] and Large Eddy Simulations (LES) techniques [6], they may also be used for full PDE modeling [7]. An interrogation remains in terms of their potential as actual proxy for *costly* PDE solution methods *at all scales*, such as direct numerical simulations (DNS). In this paper, we will test the efficiency of physics-aware DL for meta-modeling turbulent natural convection.

Turbulent natural convection is a spontaneous physical process present in many natural systems (oceans, atmospheres or mantles) as well as in engineering applications, such as passive cooling of braking systems, nuclear power plants or electronic devices or natural ventilation of buildings. A canonical system of such turbulent heat transport mechanisms is the Rayleigh-Bénard (RB) cell, where the temperature and velocity fields interact through the buoyancy force [8]. Efficiently modelling of this phenomenon is the first step towards heat transfer control for more sustainable energy systems, but it requires to track the heat carriers, namely the small-scale plumes. This remains a challenge due to the double kinetic and thermal nature of plumes, and the nonlinear interactions between various spatial and time scales from the large scale circulation to the small vortices and plumes [9]. The continued increase of supercomputing power in recent years has enabled the DNS of highly turbulent flows, resolving the entire array of scales at very high Rayleigh number. But it involves such a computational effort in terms of degree of parallelism, CPU resources and storage capacity that it will eventually entail a restriction on the spatial DNS resolution/storage and will therefore hamper its analysis. In practice, it is indeed very frustrating to not be able to store (part or most of) the computed data.

Few recent studies have been pioneering the use of DL in the framework of turbulent heat transfer with various aims. For instance, Kim et al. [10] used DL, in the form of a convolutional neural network (CNN), to predict the turbulent heat transfer – reconstructing the wall-normal heat flux at the wall – on the basis of other wall information obtained by DNS of channel flow with a passive temperature field. Fonda et al. [11] have tracked turbulent superstructures in RB convection in horizontally extended systems at $Ra = 10^{5,6,7}$ with U-shaped network (encoder-decoder CNN) allowing to reduce the dimensionality of the structures to slowly-evolving temporal 2D planar network of ridges; the idea being to propose an automated tool exploiting a large DNS simulation in order to explore heat transfer properties more easily. Pandey et al. were more interested by turbulent statistical prediction of 2D large scale structures. They relied on reservoir computing modeling, which may be seen as an hybridization between a proper orthogonal decomposition (POD) of DNS data and a recurrent neural network, to tackle RB cavity flow at $Ra = 10^7$ [12]. At the foundation of all of these works is the use of large DNS database from which partial information, in the form of wall data or time-windowed averaging, or more global information, in the form of POD, are extracted. We propose to leverage deep neural networks to *substitute* for the DNS *three-dimensional* solver by a much more agile data-driven and physics-aware surrogate. Most importantly, this surrogate will be trained with *partial* DNS data, but will incorporate known physics (such as symmetries, constraints and conservation laws in the training process), allowing the inference of hidden fluid quantities of interest [13, 14].

70

Raissi et al. [7] first introduced the concept of physics-informed neural networks (PINNs) to solve forward and inverse problems involving several different types of PDEs. This approach may be apprehended as a combination of data-driven supervised and physically-constrained unsupervised learning. For instance, they propose a way of approximating Navier-Stokes (NS) solutions that do not require mesh generation. Most of the flows considered in the aforementioned works are laminar at relatively low Reynolds numbers. A funda-

75

mental question related to whether PINNs could simulate three-dimensional turbulence directly, similarly to high-order DNS, was answered in [14]. In this study, the authors test different NS formulations for simulating turbulence. For their velocity-pressure formulation, that they found most effective, they provide their PINNs with velocity DNS data collected from the initial condition, boundaries and inside of a subdomain of their turbulent channel flow system. Closer to our application, they propose in [13] a similar approach for inferring pressure and fluid velocity from monitoring of *passively* advected scalar data with applications to laminar flows.

These previous studies [13, 14] have shown the high potentiality of the PINNs approach for recovering hidden quantities at any points inside the training domain. As a consequence, it can be viewed as a retrieval algorithm of the full flow (velocity, pressure, scalars) from a number of partial information while keeping time and space points as input. If the method were efficient enough, it could also reduce storage costs by limiting storage to an appropriate subsampling, both in time, space and variable type.

In this paper, we investigate the relevance of the approach to 3D turbulent heat transfer in the form of natural convection and modeled as Navier-Stokes equations under Boussinesq approximation. With this setup, the temperature acts as an active quantity with a feedback on the momentum equations. We will put particular emphasis on the efficiency of the surrogate modeling with respect to data and residual sampling strategy. This study shows that standard PINN version can be trained to learn a portion of the turbulent flow if large amount of data information is provided. In this framework, we propose two simple strategies to alleviate the need for large data even in highly turbulent regimes. The paper is organized as follows: we first recall the basics of the PINNs formulation and we detail our choices in terms of networks architecture, hyperparameters and databases in section 2. We then introduce the turbulent RB cavity problem setup and present standard and improved PINNs results for moderate turbulence level in section 3. In the next section, we propose a new numerical relaxation that improves the prediction for more turbulent flows. We

then discuss our findings and sketch some perspectives in section 5. We have
110 also referenced some additional results, which are important from our point of
view but not crucial to the main message of the paper in an Appendix section.

2. Standard PINNs for natural convection

2.1. Context and main ideas

Deep learning tools have recently seemed to start providing a different ap-
115 proach to computational mechanics. In particular, deep neural networks (DNN)
are now considered as an alternative way of approximating the solution of var-
ious *deterministic* PDEs types. Since some earlier studies [15, 16, 17, 18], and
thanks to significant computational advances in automatic differentiation, DNN
used as surrogates of PDEs solutions have generated a broad interest from the
120 community [19, 20, 21, 22]. Nevertheless, in the small data regime, their ef-
ficiency remains often limited and their prediction lacks robustness and inter-
pretability, motivating the idea of “adding” any form of prior knowledge to the
numerical surrogate, in order to provide some kind of “training guidance”.

One approach is to design a specialized network architecture embedding the
125 prior knowledge relevant to the task at hand. This is for instance the case of
the convolutional neural networks (CNN), thanks to their translation invari-
ant characteristics with impressive applications in image classification, medical
image analysis, natural language processing, etc. Another approach relies on
a softer enforcing of this knowledge. Raissi et al. [7] have proposed physics-
130 inspired neural networks (PINNs) for approximating solutions to general PDEs
and validated it with a series of benchmark test cases. The main feature is
the inclusion of some form of prior knowledge about the physics of the problem
in the learning algorithm, in addition to the data used to train the network.
This is done through an enlarged/enhanced loss/cost function. This way, the
135 outputs of the neural network are constrained to approximately satisfy a system
of PDEs by using a regularization functional \mathcal{L}_{PDE} that typically corresponds
to the residual (or the variational energy) of the set of PDEs under the neu-

ral network representation. The algorithm imposes a penalty for non-physical solutions and hopefully redirects it quicker towards the correct solution. As a
140 result, the algorithm has better generalization property even in the small data set regime. This approach recently drew a lot of attention, including recent development of dedicated computational packages [23].

Nevertheless, the plain version of PINNs numerically suffers from several drawbacks. They are notoriously hard to train for multi-scale and/or high-frequency
145 problems. In fact, a first difficulty resides in the discrepancy of the convergence rate between the different terms of the loss function depending on the change in the learning rate. This comes from an imbalanced magnitude of the back-propagated gradients during model training. It is therefore possible in practice to assign some weight coefficients within the loss function than can effectively
150 assign a different learning rate to each individual loss term. These weights may be user-specified or tuned automatically during network training [24]. Moreover, the required depth of the network increases with increasing order of the set of PDEs, leading to slow learning-rate due to the issue of vanishing gradients. It was also noticed that PINNs are not always robust in representing sharp local
155 gradients [25].

Another source of discredit of PINNs as described is its dependence to the data. Other teams have developed physics-constrained, *data-free* DNN for surrogate modeling of incompressible flows. The idea is to *enforce* the initial/boundary conditions instead of being penalized together during training, which is solely
160 driven by minimizing the residuals of the governing PDEs [26].

2.2. Notations and formulation

Here we introduce our notations and show how the DNN framework is coupled with a second network to form the PINN approach. The goal is to *approximate* the exact solution of a model noted \mathcal{M} , i.e. a set of unsteady PDEs,
165 written in a generic form inside and at the boundary of a physical domain Ω

evolving over a time interval $\mathcal{D} = [0, T_f]$ as:

$$\begin{aligned}\mathbf{u}(\mathbf{x}, t)_t + \mathcal{N}(\mathbf{u}(\mathbf{x}, t)) &= R(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \Omega \in \mathbb{R}^d \times \mathcal{D}, \\ \mathcal{B}(\mathbf{u}(\mathbf{x}, t)) &= B(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \partial\Omega \in \mathbb{R}^{d-1} \times \mathcal{D}.\end{aligned}\quad (1)$$

whose exact solution representing the system unknown variables is defined as $\mathbf{u} = \hat{f}(\mathbf{x}, t)$ and satisfying $\mathcal{M}(\mathbf{u}) = 0$, through the response of a neural network:

$$\mathbf{u} \approx \mathbf{u}_{\text{DNN}} = f_{\boldsymbol{\theta}}(\mathbf{x}, t), \quad \text{with} \quad \mathcal{M}(f_{\boldsymbol{\theta}}(\mathbf{x}, t)) = \mathbf{r}(\mathbf{x}, t), \quad (2)$$

with \mathbf{r} representing the residual fields of the set of equations. In Eq. (1), \mathcal{N} is
 170 a general spatial differential operator in the domain Ω , while \mathcal{B} is the boundary operator on $\partial\Omega$; R and B are potential source fields.

More specifically, the model describing our physical system of interest encompasses the 3D incompressible Navier-Stokes equations under the Boussinesq approximation, which may be written in non-dimensional form as:

$$T_t + \mathbf{v} \cdot \nabla T = \frac{1}{\text{Ra}^{1/2}} \Delta T, \quad (3)$$

$$\mathbf{v}_t + (\mathbf{v} \cdot \nabla) \mathbf{v} = -\nabla p + \frac{\text{Pr}}{\text{Ra}^{1/2}} \Delta \mathbf{v} + \text{Pr} T \mathbf{e}_z, \quad (4)$$

$$\nabla \cdot \mathbf{v} = 0, \quad (5)$$

175 with the Rayleigh number $\text{Ra} = g\beta \Delta T H^3 / \nu \kappa$, ν the kinematic viscosity, κ the thermal diffusivity, β the thermal expansion coefficient, ΔT the reference temperature difference and H the domain reference length, and the Prandtl number $\text{Pr} = \nu / \kappa$. The reference velocity is taken equal to the convective velocity $V_{ref} = \frac{\kappa}{H} \sqrt{\text{Ra}}$. The time t is therefore expressed in convective time units.
 180 Note that we have here omitted the initial and boundary conditions which are necessary for any numerical solution of the set of PDEs.

We denote the solution as: $\mathbf{u} \equiv (\mathbf{v}, p, T)$, where $\mathbf{v} \equiv (v_x, v_y, v_z)^t$, p and T are the dimensionless fluid velocity, pressure and temperature, respectively. Following the idea proposed in [13], an auxiliary variable $\bar{T} = 1 - T$ is added, that
 185 satisfies a similar transport equation as the temperature field. The DNN will therefore have to learn the nonlinear continuous mapping relating inputs and

outputs of the system.

The main portion of this network is a multi-layer perceptron (MLP) made of interconnected neurons assembled in $\ell \in \mathbb{N}$ hidden layers. The network dimensions are as follows: $n^0 = n_{\mathbf{x}} + 1 \in \mathbb{N}$ the input dimension (here $n_{\mathbf{x}} = 3$),
 $n^{\ell+1} = n_{\mathbf{u}} \in \mathbb{N}$ the output dimension and n^l the dimension of each hidden layer. The network architecture sequence \mathcal{A} may be summarized as $\mathcal{A} = (n_{\mathbf{x}} + 1, n^1, \dots, n^l, \dots, n^\ell, n_{\mathbf{u}})$. Looking at computational mechanisms of the DNN in more details, we define the following affine linear maps between adjacent layers:

$$\begin{aligned} g_{\boldsymbol{\theta}^l}^l &: \mathbb{R}^{n^{l-1}} \rightarrow \mathbb{R}^{n^l} \\ &: \mathbf{a}^{l-1} \mapsto \mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l, \quad \text{for } l = 2, \dots, \ell, \end{aligned} \quad (6)$$

where $\mathbf{a}^{l-1} \in \mathbb{R}^{n^{l-1}}$ is an array containing all the values taken by the neurons belonging to the $(l-1)$ -layer. The quantities $\boldsymbol{\theta}^{(l)} \equiv (\mathbf{W}^{(l)} \in \mathbb{R}^{n_{l-1} \times n_l}, \mathbf{b}^{(l)} \in \mathbb{R}^{n_l})$ represent the parameters containing the weights and biases to be calibrated.

Summarizing the telescoping approximation form of the DNN output, we may write it as follows:

$$\mathbf{u} \approx \mathbf{u}_{\text{DNN}} = f_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{t}) = g_{\boldsymbol{\theta}^\ell}^\ell \left(\rho \left(g_{\boldsymbol{\theta}^{\ell-1}}^{\ell-1} \left(\rho \left(\dots \rho(g_{\boldsymbol{\theta}^1}^1(\mathbf{x}, t)) \right) \right) \right) \right), \quad (7)$$

where $\rho: \mathbb{R} \rightarrow \mathbb{R}$ is a nonlinear activation function, kept the same in the entire network in this study.

Classically, for a chosen architecture, the neural network may be trained with a large, but (potentially noisy and) scattered, training set of data $\{((\mathbf{x}, t)^{(\text{train})}, \mathbf{u}_\star^{(\text{train})})\}$
 by optimizing its parameters $\boldsymbol{\theta}$. In order to minimize the error associated with the prediction of the DNN, an objective function is required by the optimization. It is referred as the loss (or cost) function and maps the set of parameter values for the network onto a scalar value. For regression problems, mean-squared error (MSE) loss functions also named L_2 -based loss functions are usually preferred:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N \|\mathbf{u}_{\text{DNN}}^{(i)} - \mathbf{u}_\star^{(i)}\|, \quad (8)$$

210 where N is the size of the data sample collected over the training domain of interest.

Finding the optimal value of $\boldsymbol{\theta}$ under this norm is equivalent to maximizing the conditional log-likelihood distribution $\sum_{i=1}^{N_L} \log \pi(\mathbf{u}^{(i)} | (\mathbf{x}, t)^{(i)}, \boldsymbol{\theta})$ [3].

215 Once the parameters have been tuned, thanks to the graph-based implementation of DNNs, it is straightforward to compute exactly derivatives of the surrogate network outputs \mathbf{u} with respect to its inputs. The spatial/temporal derivatives are obtained by applying the chain rule for differentiating compositions of functions using the automatic differentiation, conveniently integrated in many ML packages such as Tensorflow [27].

220 The PINN approach takes advantage of this functionality. Different PINN formulations exist which all share an evaluation of the PDE residuals. One formulation rely on the PDEs information alone, for which knowledge of the differential operator and initial and boundary data suffice, e.g. [28, 29]. Other formulations also use some additional solution data, for instance from the interior of the training domain. Our work follows this second approach where the loss function contains a mismatch in the given *partial* data on *some* state variables combined with the residual of the PDEs computed on a set of points in the time-space domain. It may be written as a combination of a loss term $\mathcal{L}_{\text{Label}}$ based on the data, and another loss term \mathcal{L}_{PDE} based on the PINN-predicted residuals of the PDEs, (cf. Equations (3-5)):

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \{(\mathbf{x}, t)^{(k)}\}_{k \in \mathcal{I} \cup \mathcal{J}}) &= \mathcal{L}_{\text{Label}}(\{(\mathbf{x}, t)^{(i)}\}_{i \in \mathcal{I}}) + \mathcal{L}_{\text{PDE}}(\{(\mathbf{x}, t)^{(j)}\}_{j \in \mathcal{J}}) \\ &= \frac{1}{N_L} \sum_{i=1}^{N_L} \|\mathbf{u}_{\text{PINN}}^{(i)} - \mathbf{u}_{\star}^{(i)}\| + \frac{1}{N_R} \sum_{j=1}^{N_R} \|\mathbf{r}_{\text{PINN}}^{(j)}\|, \quad (9) \end{aligned}$$

where $\mathcal{I} = \{1, \dots, N_L\}$ is a defined set with N_L the size of the input-output data sample $\{(\mathbf{x}, t)^{(i)}, \mathbf{u}_{\star}^{(i)}\}_{i \in \mathcal{I}}$ (collected inside and/or at the boundaries of the training domain), $\mathcal{J} = \{1, \dots, N_R\}$ another set with N_R the size of the sample at which PDEs residuals are computed. Note that the label and residuals sample sets are not necessarily the same, as their size and location may differ. 235 The terms \mathbf{u}_{\star} refer to sub-components of the full output (e.g. $\mathbf{u}_{\star}^{(i)} \equiv T_{\star, \Omega \times \mathcal{D}}^{(i)}$ or

$\mathbf{u}_\star^{(i)} \equiv (T_{\star, \Omega \times \mathcal{D}}^{(i)}, \mathbf{v}_{\star, \partial\Omega \times \mathcal{D}}^{(i)})$ if flow velocity information at the domain boundaries is also considered). For a given sample (e.g. fixed spatial and temporal coordinates), the residual is evaluated as the sum of an array of squared residuals of size equals to the *full* number of PDEs in the model. In the standard approach, the same ponderation is assigned to each residual of each equation of the system (cf. Equations (3-5)). The $\mathcal{L}_{\text{Label}}$ contribution can be decomposed in several terms corresponding to the contribution of various data sources: e.g. the initial condition, the boundaries or the inside of the domain. A very recent work proposed to dynamically assign some weights to each term in order to get a better error balance [24].

The standard PINN model is therefore a grid-free approach as no mesh is needed. All the complexities of solving the model are transferred into the optimization/training stage of the neural network. Updating the parameters requires the knowledge of the loss gradient $\partial\mathcal{L}(\boldsymbol{\theta}, (\mathbf{x}, t))/\partial\boldsymbol{\theta}$ that is computed thanks to the back-propagation algorithm [30]. A particular algorithm from the stochastic gradient descent (SGD) class with mini-batch (MB) updates based on an average of the gradients inside each block of MB examples:

$$\boldsymbol{\theta}^{k+1} \leftarrow \boldsymbol{\theta}^k - \epsilon_k \frac{1}{\text{MB}} \sum_{k'=k\text{MB}+1}^{(k+1)\text{MB}} \partial\mathcal{L}(\boldsymbol{\theta}, (\mathbf{x}, t)^{(k')})/\partial\boldsymbol{\theta}, \quad (10)$$

is considered and where ϵ_k is the learning rate of the k^{th} epoch. The great advantage of SGD update methods is that their convergence does not depend on the size of the training set, only on the number of updates and the richness of the training distribution [31]. To be more specific, an Adam (for Adaptive moment estimation) optimizer [32] is used, which combines the best properties of the AdaGrad and RMSProp algorithms. Moreover, the parameters of the neural networks are randomly initialized using the Xavier scheme [33]. The learning rate will take different values depending on the epochs cycle, while Adam beta values are $\beta_1 = 0.9$, and $\beta_2 = 0.999$, cf. [31].

Training cycles	MB	epochs/cycle	ϵ_k
1	2000	50	1e-03
2	2000	62	6.683e-4
3	2000	138	2.992e-4
4	2000	309	1.337e-4
5	2000	309	5.98e-5
6	2000	309	1e-5
7	2000	160	1e-6

Table 1: PINNs training hyper-parameters. MB is the size of the data sample contained in the mini-batch. ϵ_k is the learning rate. For each PINN model, the training is made of seven subsequent cycles.

2.3. PINNs architectures, hyperparameters choice and databases

265 In the following sections, the goal is to compare PINNs prediction with the DNS reference and understand how the PINN model can be made accurate, while as data-frugal as possible. The results are presented both in terms of training/validation for the chosen turbulent RB cavity with partial data information.

270 Unless mentioned otherwise, the following setup is used to construct the models. The standard PINN architecture contains $\ell = 10$ hidden layers of size $n^{l=1,\dots,\ell} = 300$ neurons each, so we have $\mathcal{A} = (n^0 = 4, n^1 = 300, \dots, n^\ell = 300, n_{\mathbf{u}} = 6)$, for a total of $\|\mathbf{W}\| = 813e3$ weights to be calibrated¹.

275 As for the training procedure, the results reported in the following are obtained after seven cycles, each of them being made of a certain number of consecutive epochs of the stochastic gradient Adam optimizer with a specific learning rate, cf. Table (1), each epoch corresponding to one pass through an entire dataset.

The total number of iterations of the Adam optimizer is therefore given by

¹Here is the computation of the weights goes according to: $\|\mathbf{W}\| = n^0 \times n^1 + \sum_{i=2}^{\ell} n^i * n^{i-1} + n^\ell \times n_{\mathbf{u}}$, computation including the biases quantities is: $\|\boldsymbol{\theta}\| = (n^0 + 1) \times n^1 + \sum_{i=2}^{\ell} n^i * (n^{i-1} + 1) + (n^\ell + 1) \times n_{\mathbf{u}}$.

280 the total number of epochs (e.g. 1500) times the size of the training data used,
divided by the mini-batch size (e.g. $MB = 2 \cdot 10^3$). The training is performed
on our laboratory Lab-IA cluster with a single NVIDIA Tesla V100 32GB GPU.

The details of the various databases that have been extracted from DNS data
285 and used to train different PINN models are written down in Tables (2,3,6,7).
The idea was for instance to test the impact of the database size and resolution
on the quality of the prediction. Tables (2, 6) describe the spatial and temporal
domains and their resolutions as well as the size of each database. Here the size
refers to the number of discrete points (\mathbf{x}, t) at which the solution of the system
290 of equations (3-5) is known. The spatial resolution is equal to (or half of) the
full DNS while temporal resolution is much coarser than the one of the full
DNS.

Tables (3, 7) summarize most of the proposed PINN models in terms of the
choice of their training and testing databases. The labels data is the following set
295 $\{((\mathbf{x}, t)^{(i)}, T_{\Omega \times \mathcal{D}}^{(i)}, \bar{T}_{\Omega \times \mathcal{D}}^{(i)}, \mathbf{v}_{\partial\Omega^\dagger \times \mathcal{D}}^{(i)})\}_{i=1}^{N_L}$, where $\mathbf{v}_{\partial\Omega^\dagger}^{(\cdot)}$ are fluid velocity components
collected at (some of) the boundaries of the training domain of interest. The
testing database is always disjoint from the training database. In the tables, N_T
refers to the size of the testing dataset (that is always disjoint from the training
database).

300 **3. Improved PINNs capability with penalty padding**

In this section, we tackle a moderately turbulent convective flow. We unwind
the PINNs methodology and we then propose a numerical technique to improve
the predictive accuracy of the approach. We first explain how we perform DNS
for the system we consider.

305 *3.1. DNS of turbulent Rayleigh-Bénard convection with heated blocks*

We consider a Rayleigh-Bénard-like configuration made of a bi-periodic wa-
ter layer heated from below ($Pr = 4.3$). The two horizontal plates are isothermal

($T_{\text{bottom}} = 1$; $T_{\text{top}} = 0$). Previous studies had shown that PINN reconstruction performs better when the training domain encompasses lively flow structures
 310 with non-zero gradients. Based on this experience, we propose a configuration producing an organized natural convection in order to easily position our training domain. To this end, two heated square-based blocks at T_{bottom} are closely placed on the bottom plate. They are aligned along one of the main diagonals. The resulting flow will be dominated by two main plumes developing over the
 315 blocks. They interact with each others in a complex pattern and swirl before impacting the ceiling (as seen on Figure 1).

The computational domain is a cube of width equal to $H = 1$, so the computational domain is defined as $\Omega = [0, H]^3$. The height of the square-based blocks is equal to $h = 0.05H$. Their base spans $(0.1H \times 0.1H)$, and their centres
 320 are located at $(x = 0.4, y = 0.4)$ and $(x = 0.6, y = 0.6)$, respectively.

The Rayleigh number of the studied case is $Ra = 2 \cdot 10^7$, leading to a moderate turbulent flow regime (Figure 1). The DNS database is obtained using the in-house numerical solver SUNFLUIDH. It is based on a finite volume approach on staggered grids. A semi-implicit scheme and a pressure-correction algorithm for the velocity–pressure coupling [34] are combined together to achieve
 325 a second-order time accuracy. The resulting Poisson’s equation is solved by a multi-grid method. The solid blocks are modeled through a loop truncation technique. A domain decomposition method with MPI communication protocol is applied for parallel computation. The code has been validated in the context
 330 of turbulent Rayleigh–Bénard convection with roughness [35]. DNS calculations are done using $(2 \times 2 \times 2)$ subdomains of 64^3 cells each with a constant convective time step equal to $2.5\text{e-}3$.

We retain a training domain about 55 times smaller in volume than the DNS computational domain (figure 1-a). It is a box-shaped volume placed over one
 335 of the two roughness blocks, of dimension: $\{\Omega_{\text{PINN}} = [0.5, 0.7] \times [0.5, 0.7] \times [0.055, 0.5]$, containing $(n_x \times n_y \times n_z = 26 \times 26 \times 38)$ grid points in the (x, y, z) frame of reference. The training domain therefore spans 20% along each x – and y –direction and 45% along the vertical z –direction. It is large enough to

cover the entirety of the plume formed above the block in its spatial ascent over
 340 half the height of the cavity, and the fluid it entrains in its near field.

The databases which are extracted from the DNS are made of a collection
 of fields spanning a maximum time length of 19.8 convective time units, cf.
 Table (2). Depending on the time sampling or the time span, four databases
 are constructed from a single DNS. Their content corresponds to four sets of
 345 time-space points that partly overlap, the most complete database being 2Db1.
 We will refer to $\Gamma_{\#Db\#}$ as the spatial-temporal discretization of size $(n_x \times n_y \times$
 $n_z \times n_t)$ of the retained database.

In addition to the turbulent character of the flow, the power spectrum of
 the vertical velocity shows the dominant shedding frequency f_{max} of the plume
 350 emission Figure (1-b). This indicates that the training domain time length
 typically includes about 10 plume rises through the studied domain.

Database	size	Δt	$(n_x \times n_y \times n_z \times n_t)$	time interval
1Db1	2 568 800	0.1	$26 \times 26 \times 38 \times 100$	ΔT_s
1Db2	1 284 400	0.2	$26 \times 26 \times 38 \times 50$	ΔT_s
2Db1	5 137 600	0.1	$26 \times 26 \times 38 \times 200$	ΔT_l
2Db2	2 568 800	0.2	$26 \times 26 \times 38 \times 100$	ΔT_l

Table 2: Specifics of the databases extracted from our DNS of the RB flow at $Ra = 2 \cdot 10^7$.
 The *size* refers to the number of discrete points (\mathbf{x}, t) of the database and is here affected by
 temporal sampling. The fourth column describes the spatial/temporal resolution inside the
 training domain $\Omega_{\text{PINN}} \times \mathcal{D}_{\text{PINN}}$ for a given database. The shorter (1Db#) and longer time
 intervals $\mathcal{D}_{\text{PINN}}$ (2Db#) are defined as: $\Delta T_s = [62, 71.9]$ and $\Delta T_l = [62, 81.8]$ while Δt refers
 to the time between two successive saved DNS snapshots.

3.2. Standard PINNs training and testing datasets

Table (3) shows the choice of training/testing datasets for the PINN models.
 In particular, for the 1Ref, 1C# and 2Ref models the training database is com-
 355 mon for the data labels and the data points at which the \mathcal{L}_{PDE} loss is evaluated.
 The data labels of total size N_L is the following set $\{((\mathbf{x}, t)^{(i)}, T_{\Omega \times \mathcal{D}}^{(i)}, \mathbf{v}_{\partial\Omega^\dagger \times \mathcal{D}}^{(i)})\}_{i=1}^{N_L}$,
 where $\mathbf{v}_{\partial\Omega^\dagger}^{(\cdot)}$ are fluid velocity components collected at the boundaries of the rect-

PINN model	Training		Testing
	labels	PDE loss	labels
	N_L , database	N_R , space/time grid	N_T , database
1Ref	2e6, 1Db1	2e6, Γ_{1Db1}	568.8e3, 1Db1
1C7	1e6, 1Db2	1e6, Γ_{1Db2}	284.4e3, 1Db2
1C7*	1e6, 1Db2	5e5, Γ_{1Db2}	284.4e3, 1Db2
2Ref	2e6, 2Db2	2e6, Γ_{2Db2}	1.131912e6, 2Db1
Padding type			
1P4	time	1e6, 1Db2	1e6, Γ_{2Db2}
1P5	space-vertical	1e6, 1Db2	1e6, Γ_{1Db2}^*
1P6	space-horizontal	1e6, 1Db2	1e6, Γ_{1Db2}^\ddagger

Table 3: Training and testing details of various PINN models for the test case at $Ra = 2 \cdot 10^7$. Mentioned DNS databases sources are detailed in Table 2. Note that for the padding cases (1P#), label and residual data points are issued from different databases. Γ_{1Db2}^* refers to the 1Db2 grid which has been vertically extended to cover the domain $\Omega_{PINN}^* = [0.5, 0.7] \times [0.5, 0.7] \times [0.05, 0.9]$ with resolution $(26 \times 26 \times 60)$ and Γ_{1Db2}^\ddagger refers to the 1Db2 grid which has been horizontally extended to cover the domain $\Omega_{PINN}^\ddagger = [0.5, 0.78] \times [0.5, 0.78] \times [0.05, 0.5]$ with resolution $(37 \times 37 \times 38)$. Reference models are highlighted in **bold**.

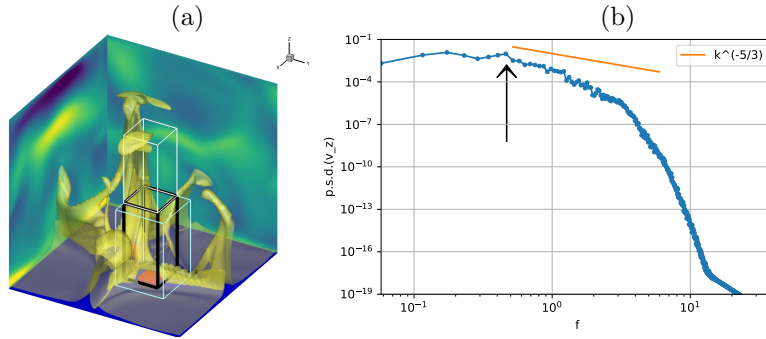


Figure 1: Rayleigh-Bénard cavity flow, with two square-based roughness blocks (red cubes attached to the bottom plate), at $Ra = 2 \cdot 10^7$. Temperature isocontours (a) ; Temporal power spectrum of v_z at $(x = 0.5, y = 0.6, z = 0.1)$. The maximum (arrow) is obtained at $f_{max} = 0.462$ (b). The training domain Ω_{PINN} is depicted as a box with thick black borders, located just above one of the roughness elements. Other contiguous boxes with lighter borders indicate the positioning of padding regions, used later in subsection 3.4.

angular blocks (excluding the top face).

360 Unless mentioned otherwise, for these models the training database is common for the N_L data labels (and N_R data points) at which $\mathcal{L}_{\text{Label}}$ (and \mathcal{L}_{PDE}) loss is evaluated, respectively. For instance, if we refer to $\mathcal{K}_{1\text{Db1}}$ as the set of data points indices from the 1Db1 database of cardinality $|\mathcal{K}_{1\text{Db1}}| = 2.5688\text{e6}$, then for the case 1C7 we consider $\mathcal{I} \subset \mathcal{K}_{1\text{Db1}}$, the subset with cardinality $|\mathcal{I}| = 1\text{e6}$
 365 and containing *any* elements from $\mathcal{K}_{1\text{Db1}}$. For the residuals, the subset is then taken the same, i.e. $\mathcal{J} \equiv \mathcal{I}$. Nevertheless, during training, mini-batches of data and points are *independently* randomly selected among these subsets \mathcal{I} and \mathcal{J} , meaning that the points at which labeled data are used and the points at which residuals are computed are not necessarily collocated within the training domain. This is illustrated in the grey region of Figure (4), showing an example
 370 of mini-batch sampling.

Testing data is not coming from a different DNS, but the database is partitioned at the beginning to separate testing and training datasets. In general testing datasets are smaller because we allocate a larger portion of available data to training. Nevertheless, an effort was made to have testing datasets keeping
 375 the same size when studying the impact of diminishing training datasets, cf. Appendix 6.

3.3. Standard PINNs results

We first present the best results obtained, i.e. for the 1Ref case trained with
 380 a *large* data sample over a short time period, i.e. 2e6 data points spanning the space/time domain (i.e. 2e4 points per DNS snapshot). The specifics of the accuracy of the PINN 1Ref model for each flow field are summarized in Tables (4,5). We see that the results are excellent with very small errors and high correlations between the PINN model and the DNS for each field. We note that the
 385 accuracy is a bit lower for the pressure field, and this finding will be consistent across all of our numerical experiments. We emphasize that no data was provided for the pressure, which was a hidden state and was obtained indirectly via

the incompressibility constraint without splitting the Navier-Stokes equations.

We also note that the accuracy on the temperature prediction is slightly
 390 lower than the one obtained from a plain DNN (noted 1Ref DNN in the table)
 with 10 layers and a loss function given by Eq. (8). A multi-output regression
 providing linear velocity and pressure predictions based on spatial/temporal
 coordinates and temperature field regressors show poor agreement especially
 for the first two components of velocity, which are known to be less correlated
 395 to the temperature field than the vertical one.

Model		Accuracy				
1Ref PINN	RMSE	MAE	μ error	σ error	R_{corr}	R^2
T	3.336e-03	2.05e-03	0.01	0.4	9.993e-01	9.986e-01
v_x	1.778e-03	1.260e-03	0.8	0.8	9.997e-01	9.993e-01
v_y	1.953e-03	1.386e-03	0.7	1.0	9.996e-01	9.991e-01
v_z	3.316e-03	2.064e-03	0.4	0.5	9.998e-01	9.996e-01
p	8.904e-04	6.341e-04	–	2.9	9.989e-01	9.971e-01
1Ref DNN	RMSE	MAE	μ error	σ error	R_{corr}	R^2
T	9.035e-04	6.362e-04	0.004	0.03	9.999e-01	9.999e-01
1Ref MOR	RMSE	MAE	μ error	σ error	R_{corr}	R^2
v_x	5.95e-02	4.73e-02	0.16	51.7	4.83e-01	2.33e-01
v_y	5.25e-02	4.11e-02	0.11	40.5	5.96e-01	3.55e-01
v_z	1.121-e01	9.02e-02	0.06	24	7.61e-01	5.8e-01
p	1.08-e02	8.2-e03	–	22.15	7.78e-01	6.05e-01

Table 4: 1Ref PINN model accuracy details (root mean squared error: RMSE, mean absolute error: MAE, the correlation coefficient: R_{corr} and the coefficient of determination : R^2), for each of the flow fields. Mean (μ) and standard deviation (σ) errors are expressed as percentage. Relative error of the mean pressure is not computable as the pressure signals are centered (zero-mean) prior to be compared. The 1Ref DNN temperature prediction and a multi-output linear regression (1Ref MOR) are also provided for comparison.

Under the large data regime, it is possible to keep an accurate predictive accuracy over a longer time period. In the following, the figure 2 shows some spatial comparisons of some flow quantities in the training domain at a specific instant representative of a single plume ascent. Despite strongly evolving three-

PINN model	Accuracy					
	aRMSE	aMAE	μ error	σ error	aR_{corr}	aR^2
1Ref	2.255e-03	1.479e-03	0.5	1.1	9.995e-01	9.988e-01
1C7	4.952e-03	2.841e-03	0.5	2.5	9.968e-01	9.928e-01
1C7*	9.858e-03	6.034e-03	0.9	7.6	9.783e-01	9.526e-01
2Ref	4.163e-03	2.469e-03	0.5	1.7	9.982e-01	9.961e-01
1P4	2.626e-03	1.736e-03	0.4	1.3	9.992e-01	9.982e-01
1P5	3.762e-03	2.354e-03	0.4	1.8	9.981e-01	9.958e-01
1P6	2.927e-03	1.982e-03	0.4	1.2	9.991e-01	9.980e-01

Table 5: Accuracy of PINN models compared to the DNS simulation at $Ra = 2 \cdot 10^7$. Statistics are computed from the available testing dataset and collected for each component of the flow fields $\mathbf{u} = (\mathbf{v}, p, T)$. They are then averaged for aRMSE, e.g. $\text{aRMSE} = \frac{1}{n_{\mathbf{u}}} \sum_{j=1}^{n_{\mathbf{u}}} \left(\frac{1}{N_T} \sum_{i=1}^{N_T} (\mathbf{u}_{\text{PINN},j}^{(i)} - \mathbf{u}_{\text{DNS},j}^{(i)})^2 \right)^{1/2}$, aMAE, aR_{corr} and aR^2 .

400 dimensional spatial structures, PINN predictions accuracy is remarkable and only subtle differences are noticed.

While current results were promising, it was obtained with a large amount of data. In the Appendix 6, a thorough study is gathered where more models are trained and discussed to understand the effect of - the architecture complexity, 405 - the size and sampling frequency of the training dataset and - the change in the time acquisition range. In particular, it was shown how the prediction accuracy deteriorates in the small training data regime. For instance, best and worse PINN model are compared in table 5 and their prediction quality is illustrated by scatter plots (including all testing data points in space/time) in Figure (3), 410 where DNS and predicted v_x flow velocity and fluid pressure are represented together with their regression fit.

An attempt to improve the results in the small-data regime is the topic of the next subsection.

3.4. Temporal- or spatial-penalty padding

415 Given a fixed training labelled dataset of size N_L , the idea is to increase the predictive accuracy of the PINN by adopting a smarter positioning/sampling

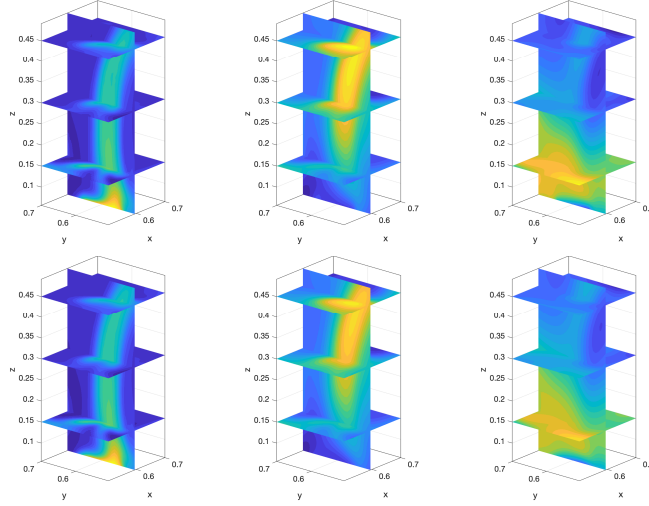


Figure 2: Comparison of some ground truth (DNS) (top) and 2Ref PINN-predicted (bottom) instantaneous flow fields. Left: temperature $T(x_0 = 0.595, \cdot, z_0 = [0.15, 0.3, 0.4], t = 70.7)$, middle: vertical flow velocity v_z and right column: pressure field p . Remark: Neither DNS data nor PDE residual points at the predicted time instant (issued from $\mathbf{\Gamma_{2Db1}}$) were used for the training.

strategy for the choice of the points at which the residual penalties are imposed. If we assume that the labeled data points are localized in a single temporal/spatial domain of interest, here referred as I for simplicity, cf. black dots
420 in grey area in Figure (4), the standard approach is to allocate all of the residual points within the same domain. Our idea is to use part of the residual samples to pad, either in space or/and in time, the surrounding regions of this domain, so that they belong to a larger domain J^p encompassing the data domain I .

Indeed, it was noted that predictive accuracy of the PINNS is often lower
425 close to spatial and temporal boundaries [24], so we hope to improve the accuracy there by extending the domain of regularization. For simplicity, in the following, we will assume that $(N_R = N_L)$. We may rewrite the loss function as:

$$\mathcal{L}(\theta, \{(\mathbf{x}, t)^{(k)}\}_{k \in \mathcal{I} \cup \mathcal{J}^p}) = \frac{1}{N_L} \left(\sum_{(i \in \mathcal{I}, j \in \mathcal{J}^p)=1}^{N_L} \|\mathbf{u}_{\text{PINN}}^{(i)} - \mathbf{u}_\star^{(i)}\| + \|\mathbf{r}_{\text{PINN}}^{(j)}\| \right), \quad (11)$$

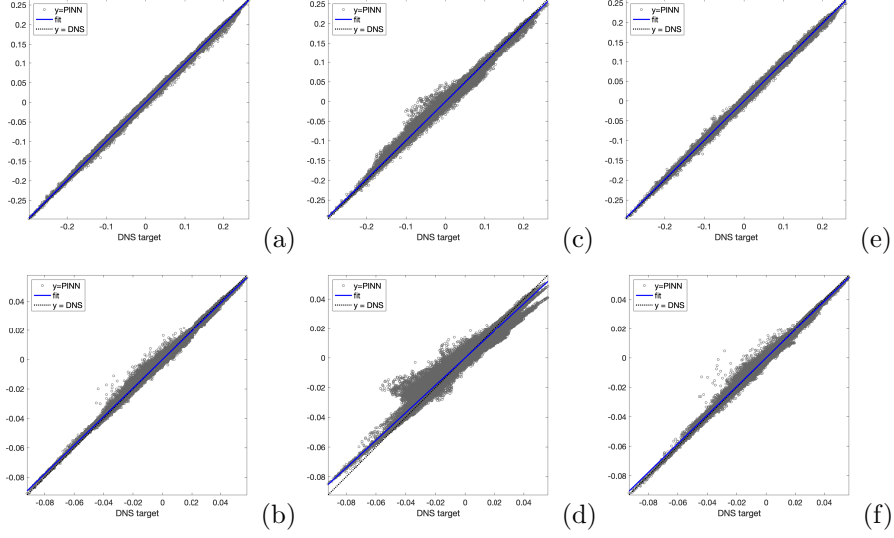


Figure 3: Examples of scatter plots comparing DNS and predicted values from most accurate 1Ref (a-b), least accurate 1C7 (c-d) and padded 1P6 (e-f) PINN models, with top row: in-plane x -axis flow velocity v_x , and bottom row: fluid pressure p .

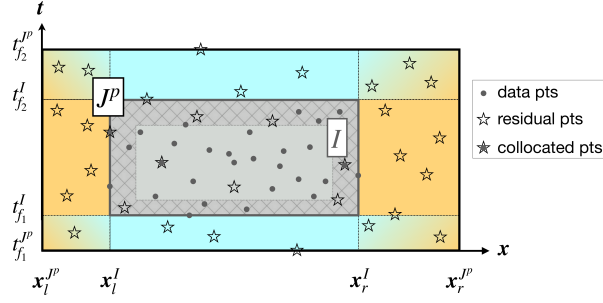


Figure 4: Various spatial/temporal-padding strategies: residual points are evenly distributed across the monitored (gray) region as well as in neighbor space/time (colored) regions where data points are lacking. The goal is to improve the approximation in close neighborhood (hatched layer) of the domain of interest boundaries. In space, only one dimension is represented for clarity. Points at which data are available (iron color circles) and points at which PDEs residual are minimized (black hollow stars) are not necessarily collocated during the optimization process due to the random selection of the minibatch SGD approach. Remark that regularization padding may be deployed in space, in time or both.

where \mathcal{J}^p is the set of indices of the points scattered across J^p . If we note
430 $\alpha_I \equiv N_L / (n_x n_y n_z n_t)$ the average density of data points in I relative to the 3D
DNS resolution (assuming this grid is regular), the average density of residual
points in J^p , $\alpha_{J^p} < \alpha_I$, will depend on the way the padding is extended around
 I (choice of direction(s), simple/complex extrusion, symmetry, etc.). In practice
once J^p is chosen, the positioning of the N_L residual points is really flexible and
435 for instance does not need to lay on a regular grid.

Table (3) shows the details of some numerical experiments used to investi-
gate this idea. Three new cases, 1P4, 1P5 and 1P6, adopt the same database
and data sampling as case 1C7, but this time the residual points span a longer
time period for 1P4, a wider spatial vertical (horizontal) range for 1P5 (1P6),
440 respectively. For all of these cases, J^p domains have been setup so as to lower
the residual points density by a factor two (either in time or space; padding in
time *and* space has not been tested).

Numerical testing over the 1Db2 database shows that the accuracy is much im-
proved for those cases compared to case 1C7. In particular, improvement of
445 case 1P6 for which the spatial domain has been horizontally padded is spectac-
ular, cf. subplots (e-f) from the Figure (3). Another numerical experiment has
been ran (i.e. 1C7*), in order to check that the improvement obtained for this
particular application was not simply due to the lower density of residual point
within the training domain, instead of the padding effect. This was not the case
450 as 1C7* produced very poor results, cf. Table (5).

The other padded cases, such as the temporal padding, are also interesting.
For instance, Figure (5) compares the temporal distribution of the L_2 spatial
errors integrated for case 1C7 (thin black line) and 1P4 (thick gray line). The
curves in each subplot are normalized by the maximum value of the error for
455 case 1P4 (i.e. 20% for temperature and 83% for velocity). The PINN models
are asked to predict the solution on the long time range and for a resolution that
is doubled (i.e. 200 snapshots for ΔT_l) compared to the one of their training
data. We clearly identify the first half of the time domain in which errors are
low. The 1C7 PINN solution is then predicted in the later time domain (no

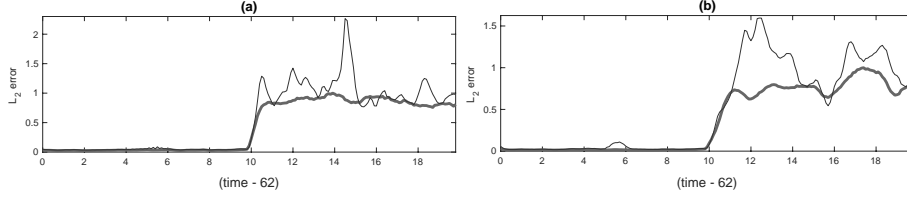


Figure 5: Comparison of the temporal distribution of L_2 spatial errors for case 1C7 (thin line) and 1P4 (thick line) for temperature (a) and v_z vertical flow velocity (b). The curves in each subplot are normalized by the maximum value of the error for case 1P4 (i.e. 20% for temperature and 83% for velocity). The padded approach 1P4 controls errors much better, especially in the second half of the time window.

information from this time range was used during training of this model), while the 1P4 prediction benefits from a training with residual points (but no data) in this time range, as explained previously. Not only, the 1P4 solution is improved in the first half but it is also much more robust in the second half, controlling better error spikes all along the time window.

In conclusion, despite diluting their concentration, redistributing in a larger domain the grid points at which low residuals are enforced, helps in improving the predictive accuracy of the PINN model within the domain of interest. In the following section, we will address the issue of modeling higher turbulence levels by proposing a new simple idea that considerably improves the prediction results.

4. Improving modeling capability for more turbulent scenario by relaxing surrogate constraints

The promising results obtained in the previous sections motivate an investigation of more challenging natural convection metamodeling at higher turbulence levels. The attempt might fail as it is known that conventional PINN models are not very successful at approximating complex dynamics leading to solutions with non-trivial behavior, such as directional anisotropy, multi-scale features or very sharp gradients. The specificity of the PINN approach is that the constraints alter the loss landscape of that type of deep neural networks. As

480 seen previously, different terms in the PINN composite loss function have different nature and magnitudes, sometimes leading to imbalanced gradients during back-propagation. Recent works have pointed to the problem of the stiffness of the PINN gradient flow dynamics. They have proposed a learning rate annealing algorithm that utilizes gradient statistics during training to adaptively
 485 weight the different terms in the *label part* of the loss function. They have also proposed a new fully-connected neural network architecture that is less sensitive to the stiffness of gradient flow.

In the following, we propose a minimal modification of the PINN computational framework to make it more efficient for our type of application. In particular,
 490 we do not want to modify the PINN architecture, the training computational budget or to upgrade too drastically the training dataset size. As we will describe in more details below, our idea is to relax some of the PDE residual losses in order to enhance the accuracy and robustness of our PINNs.

4.1. DNS of turbulent Rayleigh-Bénard convection in smooth cavity at high 495 Rayleigh number

In the following, we consider a much more challenging case of RB flow in a smooth cavity filled with water ($Pr = 4.4$) at higher Ra number, cf. Fig. 6. The geometry does not bear any roughness anymore and the flow turbulence is much more developed than previously, with a larger Rayleigh value by two
 500 order of magnitude, i.e. $Ra = 2 \cdot 10^9$. Our goal is to investigate the level of accuracy we can achieve with a computational budget equivalent to the previous simulations. We keep the same simple fully-connected PINN architecture ($\ell = 10$ layers) and the same total number of epochs = 1500 (which is very low compared to other works). A DNS is performed in a computational do-
 505 main of size $\Omega = [0, 1] \times [0, 0.5] \times [0, 1]$, cf. Fig. 6. The quite large domain $\Omega_{\text{PINN}} = [0.65, 0.85] \times [0.2, 0.3] \times [0.05, 0.3]$ over which the PINN models are trained, is depicted as a transparent box. The spatial flow scales are hard to capture with dispersed and disorganized small and thin turbulent plumes, which travelling direction and orientation are not well established relative to Ω_{PINN} .

510 They do not necessarily travel across the domain from bottom to top because the flow velocity components are less dominated by their vertical component.

At this Ra , the amount of data from our “true” DNS is colossal with a spa-

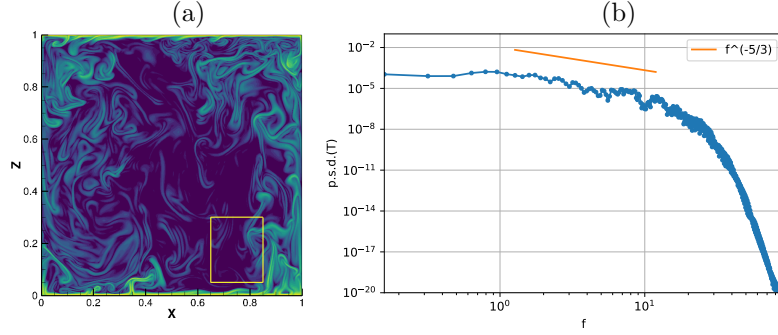


Figure 6: Example of instantaneous heat dissipation isocontours (2D (x, z) slice at $(y = 0.25, t = 290.83)$) from DNS of the RB cavity flow at $Ra = 2 \cdot 10^9$ (a). The domain Ω_{PINN} over which the PINN model is trained, is depicted as a transparent box. Temporal power spectrum of temperature at location $(x = 0.65, y = 0.25, z = 0.1)$ (b)

tial resolution for the PINN domain (Ω_{PINN}) *alone* of $(129 \times 65 \times 219)$ points updated every $\Delta t_{\text{DNS}} = 4.5 \cdot 10^{-4}$. The database being too large to be stored,

515 we have saved the solution with a coarser time resolution, i.e. we have collected 249 snapshots collected every $40 \times \Delta t_{\text{DNS}}$, totalizing almost half a billion $(4.57242435e8)$ points at which the flow data are saved. It is this latter down-sampled version that we will refer as our “full” DNS. Finally, a more tractable database 3Db1, referenced in Table (6), is constructed from the aforementioned

520 full DNS, with a lower resolution in space and time. The PINN training dataset retained from the 3Db1 database totalizes 90% of the (velocity-temperature) data points of the initial condition, 100% of the (velocity) boundary conditions (excluding the top of the training domain as before) and only 25% of the (temperature) bulk totalizing $7.334712e6$ data points, which is only 1.6% of the full

525 (i.e. 0.04% of the true) DNS spatial/temporal coordinates points and about 0.32% of the full DNS data (i.e. temperature, velocity and pressure data fields). The training dataset being now 3.7 times the size of the largest dataset used for the $Ra = 2 \cdot 10^7$ studies, the mini-batch size is increased to $MB = 18522$ (so as

to keep about the same cpu computational time).

530

Database	size	Δt	$(n_x \times n_y \times n_z \times n_t)$	time interval
3Db1	20 673 972	0.054	$66 \times 34 \times 111 \times 83$	ΔT_m

Table 6: Specifics of DNS-extracted database for RB flow at at $Ra = 2 \cdot 10^9$. Similar caption to table 2. The medium time interval is defined as: $\Delta T_m = [287.572, 292]$ while Δt refers to the time between two successive snapshots.

4.2. Methodology of relaxing surrogate constraints

The convergence of the various terms of the loss function are depicted in Figures (7). In subplot (a), standard PINN results show that the optimization is very sensitive to the incompressibility constraint (PDE₅: yellow line, cf. Eq. (5)), in agreement with the findings of other researchers [14]. Oscillations are very strong compared to the other components. Moreover PDE₅ convergence exhibits a piecewise behavior with error magnitude and fluctuations getting lower across changes in learning rate cycles, but a convergence that does not really progresses within each of the learning cycles. Despite being quite low, the convergence of the temperature loss (PDE₁, cf. cf. Eq. (3)) looks very flat and seems to stagnate.

With the PINN velocity-pressure formulation, the pressure equation is not obtained through an additional Poisson pressure equation as it is usually done with splitting methods. In fact, the pressure is a hidden state and is obtained via the incompressibility constraint. Nevertheless, the incompressibility condition is hard to impose through our SGD algorithm. Recent works have investigated other Navier-Stokes (NS) formulations [14] including a streamfunction-pressure formulation for which the incompressibility constraint is exactly satisfied [24]. They concluded that these alternative formulations were more efficient, especially for laminar flows.

Keeping our original velocity-pressure formulation, our novel idea is simply to relax the most sensitive constraints, i.e. the incompressibility constraint.

The new loss function reads:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{bulk}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{boundaries}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{initial}}(\boldsymbol{\theta}) + \sum_{l=1}^{M=5} \lambda_l \mathcal{L}_{\text{PDE}_l}(\boldsymbol{\theta}, \{(\mathbf{x}, t)^{(j)}\}_{j \in \mathcal{J}}), \quad (12)$$

with

$$\begin{aligned} \mathcal{L}_{\text{bulk}} &= \frac{1}{N_{\text{bu}}} \sum_{i=1}^{N_{\text{bu}}} \|T_{\text{PINN}}^{(i)} - T_{\star}^{(i)}\|, \\ \mathcal{L}_{\text{boundaries}} &= \frac{1}{N_{\text{b}}} \sum_{i=1}^{N_{\text{b}}} \|\mathbf{v}_{\text{PINN}}^{(i)} - \mathbf{v}_{\star}^{(i)}\|, \\ \mathcal{L}_{\text{initial}} &= \frac{1}{N_{\text{i}}} \sum_{i=1}^{N_{\text{i}}} \|(T_{\text{PINN}}^{(i)}, \mathbf{v}_{\text{PINN}}^{(i)}) - (T_{\star}^{(i)}, \mathbf{v}_{\star}^{(i)})\|, \\ \mathcal{L}_{\text{PDE}_5} &= \frac{1}{N_R} \sum_{j=1}^{N_R} \|\nabla \cdot \mathbf{v}^{(j)}\|, \end{aligned} \quad (13)$$

555 where $\mathcal{L}_{\text{bulk}}$, $\mathcal{L}_{\text{boundaries}}$ and $\mathcal{L}_{\text{initial}}$ are the bulk, boundaries and initial data-fit terms respectively, and $N_L = N_{\text{bu}} + N_{\text{b}} + N_{\text{i}}$ and the other PDEs residuals are obtained from system (cf. Equations (3-5)). Then a preliminary study followed by a careful search has shown us that only λ_5 needed adjustment to relax the constraint on the conservation of mass. We refer to this relaxed version of the
560 physics-informed neural network as PINN^r.

4.3. Results

Results obtained with this new approach are spectacular, and the total loss reaches a much lower value at the end of the training, cf. Figure (7-(b)). The
565 loss of the relaxed version of the divergence-free equation converges much faster and more regularly (cf. yellow curve). Its normalized version - not presented here and accounting for the λ_5 parameter rescaling - shows that the loss achieved is of the same order of magnitude as the original one without relaxation. But it impacts the temperature loss that follows a much more flattering convergence
570 slope as well. Very interestingly, there is also a retro-action of these lower PDE residual losses onto the convergence of the label losses (cyan curve). This coupling is extremely interesting as it demonstrates how an improved learning on

the PDE part of the losses directly benefits the learning of the scarce temperature data.

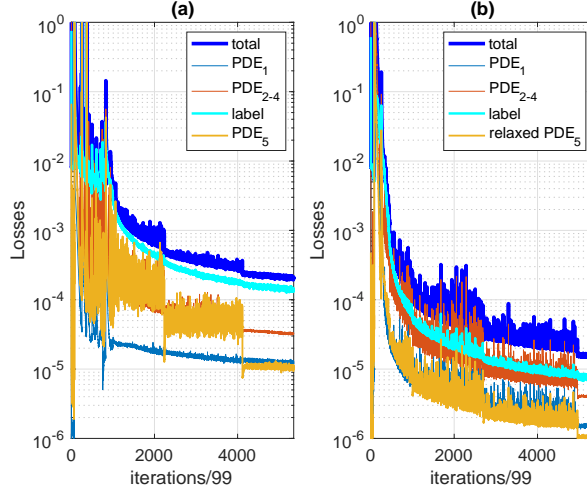


Figure 7: $Ra = 2 \cdot 10^9$ study: details of the convergence of loss functions for standard PINN 3C1 (a) and best relaxed PINN model ($3C1^r$, $\lambda_5 = 0.1$) (b) for which the mass conservation PDE constraint (i.e. PDE_5) is relaxed during training. PDE_1 : loss term associated with the temperature equation and PDE_{2-4} : sum of loss terms associated with the momentum equations, cf. system (3-5).

PINN models	Training		Testing
	labels	PDE loss	labels
	N_L , database	N_R , space/time grid	N_T , database
3C1; $3C1^r$	7.334712e6, 3Db1	7.334712e6, Γ_{3Db1}	6e6, 3Db1

Table 7: $Ra = 2 \cdot 10^9$ study: the caption is similar to the one of Table (3).

575 Tables (7,8) summarize the specifics and accuracy of various relaxed models
(3C1^r) compared to the reference one (3C1). The results are very clear and show
the undeniable superiority of the PINN^r approach with very good accuracy, for
 $\lambda_5 < 1$. In particular, best results are obtained for the relaxed model with
 $\lambda_5 = 0.1$. Moreover, an ambitious validation of this approach was also carried
580 out on the full DNS. That is to say that the best model was used to predict

PINN model		Accuracy					
	λ_5	aRMSE	aMAE	μ error	σ error	aR_{corr}	aR^2
3C1 ^r	10	1.591e-02	1.071e-02	30.6	20.2	8.218e-01	6.948e-01
3C1	1	5.354e-03	3.03e-03	8.1	6.8	9.704e-01	9.413e-01
3C1 ^r	0.1	1.262e-03	6.522e-04	1.4	1.0	9.983e-01	9.964e-01
3C1 ^r	0.01	2.156e-03	1.13e-03	1.4	2.1	9.948e-01	9.893e-01
3C1 ^r	0	2.827e-03	1.565e-03	4.1	2.2	9.949e-01	9.894e-01

Table 8: $Ra = 2 \cdot 10^9$ study: accuracy details for PINN models with various relaxed constraint; the caption is similar to the one of Table (5). Best relaxed PINN model is obtained for $\lambda_5 = 0.1$.

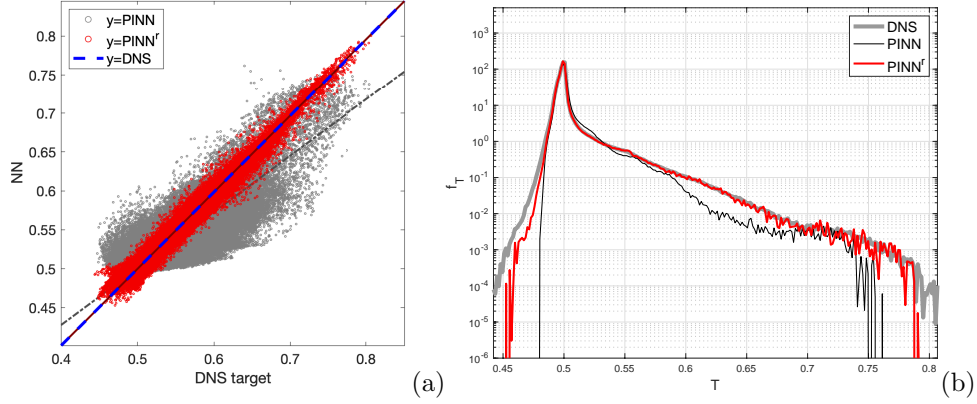


Figure 8: PINN predictive capabilities for RB flow at $Ra = 2 \cdot 10^9$: comparisons between standard (3C1) and relaxed (3C1^r, $\lambda_5 = 0.1$) approaches. Temperature scatter plots (a) and probability density function (b) compared to the reference DNS. The reference pdf is computed from the full DNS while the PINNs pdf are evaluated on the predicted test sample.

the solution at the coordinates of the full DNS referenced hereinbefore (i.e. on a sample of size $N_T = 129 \times 65 \times 219 \times 249$, corresponding to a doubling of the spatial resolution in each direction and a tripling of the temporal resolution compared to the training sample). Very impressively, the errors computed in

585 the relative L_2 norm were only: 0.3% for the temperature, 1.79% for v_x , 2.708% for v_y , 3.416% for v_z and 4.038% for the pressure, respectively.

Figures (8) present the scatter plot of the PINN temperature results of case 3C1 and 3C1^r ($\lambda_5 = 0.1$) (a), and the corresponding pdf compared to the reference

DNS. The difference is for instance very striking when looking at the way the
 590 approximation is now capable of sharply capturing the long tail of the skewed
 temperature probability density function (PDF). This asymmetric PDF shape
 is typical of the mixing layer [36], in which the training domain is placed. The
 long quasi-exponential tail for large temperature fluctuations is the signature of
 the travelling hot plumes, intermittently passing through the domain. Even if
 595 temperature is only scrutinized here, quantitative improvements of the PINN^r
 approach do occur for all flow variables. On the contrary, it can be seen that the
 original PINN approach miss a large part of the warm plumes, but especially
 the cold (descending) ones.

Figures (9) were chosen to show an example on how the surrogate is capable
 600 of accurately predicting a strong small plume with very anisotropic structure,
 despite being located close to the domain boundaries and occurring at a time
 instant never visited during training. It is remarkable how well the intricate
 temperature distribution within the plume is approached by PINN^r.

5. Discussion

605 5.1. Importance of boundary information: data vs. penalty padding

The previous discussion based on our results confronted to a literature re-
 view, clearly points to the importance of the boundary data information in the
 various PINN formulations. If no training data are used from the bulk, i.e.
 only the residuals are to be evaluated (for which knowledge of the differential
 610 operator and PDEs inputs suffice), initial and boundary data of the full solution
 state (which are implicitly included into the formulation of the PDEs) *must* be
 provided. With this approach, velocity, temperature and pressure data would
 therefore be needed initially and at the boundaries of the training domain. We
 will refer to this approach as the plain vanilla one.

615 Our motivation was different with a desire to rely mainly on temperature data
 from the bulk [13]. Unlike two-dimensional laminar flow problems, it was noticed
 in [14] that for more complex convective three-dimensional flows, temperature

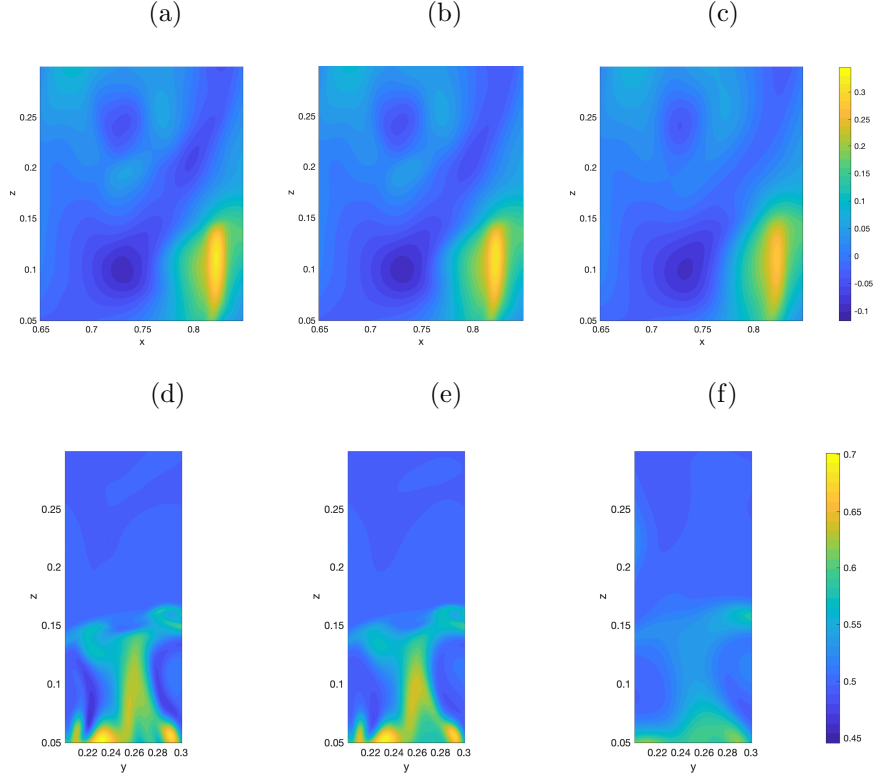


Figure 9: Comparison of some ground truth (DNS) (a,d), PINN^r-predicted (3C1^r, $\lambda_5 = 0.1$) (b,e) and standard PINN-predicted (3C1) (c,f) instantaneous sliced fields with a thin sheet-like plume located at the bottom of the domain. Top row: vertical flow velocity $v_z(\cdot, y_0 = 0.25, \cdot, t = 290.83)$ and bottom row: temperature $T(x_0 = 0.82, \cdot, \cdot, t = 290.83)$ fields. The time instant is chosen so as to correspond to a DNS snapshot that is *not* included in the training database. Predictions are requested on the fine DNS spatial grid.

data was not sufficient (problem of well-posedness) to satisfactorily train the model, and information relative to flow velocity boundary conditions were also
620 necessary. This is indeed something that we have confirmed in previous studies as well as the importance of the positioning of the training domain relative to the flow features [37]. For our numerical experiments, DNS fluid velocity from the domain boundaries (except the top one) is used to complement the temperature data. The dimensionality of this information being lower, for instance for case
625 1Ref: $|\mathbf{v}_{\partial\Omega^\dagger}^{\text{DNS}}| = 4628$ at each given time, the small mini-batch size that we use

at each training iteration ($MB = 2000$) collects (on average due to the random
 sampling) about $(MB/100_{\text{snapshots}})/5_{\text{faces}} = 4$ flow velocity data points per face
 which is a small number. Nevertheless, for each epoch based on the tempera-
 ture data, the algorithm cycles more than once (here about 4 times) through
 630 the fluid velocity boundary values, therefore using this information many times.
 It will be interesting to further quantify the impact of the boundaries informa-
 tion on the method efficiency. This could be achieved by playing with the ratio
 of training data points chosen at the boundaries vs. the inside of the domain.
 The penalty padding that we have proposed in the previous section can also be
 635 handy in this case. It comes to play as a regularization over the spatial and/or
 temporal zones surrounding the training boundaries which are often regions of
 poor accuracy of the PINN surrogate. It seems to complement the local bound-
 ary data and blends the solution nicely across the chosen boundaries, resulting
 in a noticeable global accuracy improvement. Future works need to be pursued
 640 in order to determine how to improve this technique, e.g. choice of the padding
 domains extent and shapes, distribution of the residual points density, choice of
 PDEs to enforce, etc.

It is important to check if the padding regularization may substitute (at least
 in part) the amount of boundary data required for learning. That is we wish to
 645 reduce the training data at the boundaries relative to the padding penalty. To
 this end, a test was carried out in order to infer on the importance of missing
 boundaries information: the best padded case (1P6) was rerun without includ-
 ing velocity data on certain domain faces covered by the padding. To make
 things clear, only velocity data at the bottom, back and left faces were pro-
 650 vided during training, while velocity at front (respectively right) face located at
 $x = 0.7$ (respectively $y = 0.7$) were not provided. The idea was to check if miss-
 ing local boundary information could be supplemented thanks to the padded
 neighbor region filled with low PDE residuals enforcement points. The results
 (not reported here) were deceptive with an averaged error close to 20%. This
 655 shows that the boundary information remains very important for this type of
 flows, especially when the amount of bulk labeled data is on the lower side. This

finding is consistent with recent works applied to incompressible internal flows where a structured DNN architecture was devised in order to automatically enforce (in a “hard” way) initial/boundary conditions [26]. In this particular case, 660 it was not necessary to include any bulk simulation data [29], the DNN being trained by solely minimizing the residuals of the Navier-Stokes equations.

Another question relates to the relevance of the padding technique in case the PINN domain overlaps the *entire* computational domain. If we retain the idea that the padding acts as a regularization on the surroundings of the studied 665 domain, a possible extension of our approach would be to add an external volume to weakly impose boundary conditions on its frontiers. An elegant formulation would then be to consider the initial domain and its extension as a whole over which an (enhanced) set of PDEs (with proper forcing terms accounting for the boundary conditions) would be enforced with the PINN formulation, in a 670 manner analogous to what is done for immersed boundaries [38].

5.2. *Impact of the composite loss regularity*

Physics-guided regularization of the loss function acts very differently than the more standard (L_1 or L_2) norm-based ML regularization techniques. Indeed the underlying differential operator makes the loss landscape very hard 675 to optimize, and from a PDE perspective, if the operator is ill-conditioned the PINN generalization will most likely contain large errors. PINNs seek to find a neural network that minimizes the loss in a particular class of neural networks. The minimizer then serves as an approximation to the solution of the PDEs. It is accepted that PINNs accuracy depends on an approximation error (dictated by the choice of the class of neural networks considered), an optimization 680 error (which depends on how well the global minimizer is found), and an estimation error (which comes from the use of finite data). While balancing these different sources of errors seems a reasonable goal, very recent papers started to build rigorous foundations of the approximations of PDEs by PINNs, and are particularly interested by the generalization error of plain vanilla PINNs for 685 various systems (e.g. linear elliptic/parabolic, nonlinear viscous scalar conser-

vation laws, etc.) [29, 28]. For a given choice of data sampling/quadrature, they obtain estimates for the generalization error in terms of the training error and number of training samples, showing that under the assumption of a small
690 training error, increasing the number of samples at which residuals are evaluated guarantees convergence of the approximation. While a fully turbulent convection regime is not yet covered by the theory, we thought that it was interesting to interpret our results in the light of these findings.

Figuring *a priori* the amount of data information necessary to the PINN training, is a complex matter because of the interplay of many different terms involved in the composite loss function. Indeed the lost function, that is just an evolving scalar, encapsulates various error terms related to the data and to the PDEs-based penalties. This apparently simple formulation hides its underlying complexity because $\mathcal{L}_{\text{Label}}$ gathers several data-fit terms, possibly including initial, boundary and internal data. It was for instance shown that this approach
700 might lead to an unstable imbalance in the magnitude of the back-propagated gradients during model training using gradient descent [24]. Some formulations have proposed a regularization parameter that acts as a weight in front of the penalty term.

For the relaxation method we have proposed, we have fixed the number of data and residual points and focused instead on a way of improving the optimization error by relaxing some of the underlying PDEs. In reference to the aforementioned works, we believe that our generalization error has therefore lowered thanks to a lower training error induced by a better conditioned loss function
710 to minimize. A better way to pursue this approach would probably be to do it dynamically as it is straightforward to monitor the distribution of the back-propagated gradients of each individual loss terms with respect to the neural network parameters in each hidden layer during training. Some preliminary work on adaptive dynamic weights strategy, shows that for turbulent flows (cf. section 4 in [14]), the perfect balance of the residuals gradients is not easily
715 reachable and asks the question of the proper normalization with careful tuning of those anisotropic weights.

5.3. Perspectives

The results obtained in this paper open the way for more involved *parametric* surrogate modeling (useful in the context of design optimization, model calibration, sensitivity analysis, etc), i.e. nonlinear problem parametrized by some (potentially not well-known) physical quantities, playing the role of additional inputs to our PINNs models. Despite the large body of literature on uncertainty quantification, including aleatoric and epistemic uncertainties in fluid mechanics [39], few works have attempted to propose DNN-based scalable algorithms for parametric surrogate CFD modeling, due to the lack of *a posteriori* error estimation and convergence theory. Moreover, training data is a severe bottleneck in most parametric fluid dynamics problems since each data point in the parameter space requires an expensive numerical simulation based on first principles.

Nevertheless, some numerical perspectives may be drafted by examining some of the limiting computational aspects of the PINN approach. The PINN algorithm infuses the system governing equations into the network by modifying the loss function with a contribution acting as a penalizing term to constrain the space of admissible solutions. The high-dimensional non-convex optimization problem of this composite loss function involves a large training cost related to the time-integration of the nonlinear PDEs and the depth of the neural network architectures. We have seen that the approach may be efficient despite a training based on a very sparse data sample, while other approaches investigate variant sampling strategy, e.g. [40, 41].

But in the more demanding case of parametric surrogate construction, an effort should be pursued on the front of efficient data sampling strategy. Indeed, optimal sampling would ensure a right balance and therefore good complementarity between the information provided by the PDEs and by the data. A potential breakthrough would be to propose a dynamic selection of relevant data for the PINN learning, operating synchronously with the physical simulation, and allowing sparser spatial-temporal sampling, responding in part to the storage problem of DNS simulations. Moreover, it could be beneficial to simultaneously

build a data index structure allowing to benefit from importance sampling, e.g.
750 importance sampling tree technique [42].

In the case of parametric surrogate modeling, another interesting approach
would be the one of transfer learning (TL). The TL domain seeks precisely
to transfer the knowledge acquired to a training dataset to better process a new
so-called “target dataset”. The transfer can therefore take the form of a parallel
755 relearning of the neural network taking into account the evolving parameters
(geometric or physical for instance).

More importantly, we have experienced the high sensitivity of the learning pro-
cess to the way we enforce (some of) the PDEs in the PINNs framework and
the impact it had on the global accuracy of the scheme. Inspired by the work
760 of Perdikaris and co-authors [24], we believe it would be worthwhile tracking
the gradients of each individual terms in the PDEs constraints with respect to
the weights in each hidden layer of the neural network, rather than tracking
the gradients of the aggregated loss. This will help monitoring the distribution
of those back-propagated gradients during training and propose a learning rate
765 annealing algorithm that utilizes gradient statistics to balance the interplay be-
tween the different terms in the regularization components of the composite loss
function. More specifically, due to the stochastic nature of the gradient descent
updates, updated learning rates should be computed as running averages of pre-
vious values and do not need to be updated at each iteration of the optimization
770 solver.

Other perspectives and current works involve – the decomposition of the
computational domain into several training sub-domains in order to better scale
locally-adapted PINN models, – handling of aleatoric uncertainty associated
with noisy training data by means of physics-informed Bayesian neural net-
775 works [43], – the mixing of various labeled data sources – hybrid regularization
techniques combining physics-informed regularization with more classical L_2 ,
 L_1 and/or dropout regularizations.

6. Appendix

In this Appendix, a preliminary study is carried out to determine reference
 780 PINN models that will be used in Section (3.3). In particular, several models
 were trained to understand the effect of - the architecture complexity, - the
 size and sampling frequency of the training dataset and - the change in the time
 acquisition range. Table (9) provides a summary of the accuracy of the different
 models considered. The model names – starting with a 1·, refer to the cases with
 785 training and testing over a short time window ΔT_s , while the names – starting
 with a 2· refer to the cases with a longer time window ΔT_l . Another difference
 resides in the way models are tested. The first models are always tested on an
 independent sample of points coming from the *same* database as the one used
 for training. The second models are always tested on an independent sample of
 790 points coming from a *different* database (e.g. denser in time) than the one used
 for training.

We see how the 10-layer PINN architecture with large amount of data and
 residual evaluations provide the best overall results for the first models. With
 less data sampled in space (i.e. conserving the same temporal resolution), cases
 795 1C5-6 show some reasonable decline of their accuracy. With less data sampled
 in time (i.e. conserving the same averaged spatial sampling frequency), cases
 1C7-8 exhibit a worse decay of their predictive capability.

When adequately tuned, PINN models demonstrate a fine expressivity. For
 instance, for the training of case 2C3, a half-a-million time-space scattered
 800 temperature data points over 50 snapshots and available boundary velocity
 data points (on $\partial\Omega^\dagger$) were used and provided a very good average accuracy
 of $aR^2 = 0.988$ over a testing sample encompassing 100 snapshots. This train-
 ing corresponds to a moderate sampling of the DNS data: i.e. temperature data
 were randomly collected with a temporal sampling of $\Delta t = 0.2$, that is every
 805 80 DNS time steps and a spatial sampling of about 20% of the $(26 \times 26 \times 38)$
 available DNS temperature data points at each time step.

An important matter to this study was the one of the PINNs robustness to lower

PINN model	ℓ	Training		Testing	
		size (N_L, N_R)	database	size (N_T)	database
1C3	6	(2e6, 2e6)	1Db1	5.688e5	1Db1
1C4	8	(2e6, 2e6)	1Db1	5.688e5	1Db1
1Ref	10	(2e6, 2e6)	1Db1	5.688e5	1Db1
1C5	10	(1e6, 1e6)	1Db1	5.688e5	1Db1
1C6	10	(5e5, 5e5)	1Db1	5.688e5	1Db1
1C7	10	(1e6, 1e6)	1Db2	2.844e5	1Db2
1C8	10	(5e5, 5e5)	1Db3	1.422e5	1Db3
2Ref	10	(2e6, 2e6)	2Db2	1.131912e6	2Db1
2C2	10	(1e6, 1e6)	2Db2	1.131912e6	2Db1
2C3	10	(5e5, 5e5)	2Db2	1.131912e6	2Db1

Table 9: Training and testing details of the ℓ -layer PINN models considered in this study. Each number of points spans space/time domain. When a single database is mentioned, it means that this database is used for the labeled data *and* for the residual points (i.e. residuals are evaluated at some DNS grid points). Reference models are highlighted in bold.

PINN model	Accuracy					
	aRMSE	aMAE	μ error	σ error	aR_{corr}	aR^2
1C3	3.701e-03	2.418e-03	0.4	1.6	9.986e-01	9.969e-01
1C4	2.468e-03	1.612e-03	0.5	1.2	9.994e-01	9.985e-01
1Ref	2.255e-03	1.479e-03	0.5	1.1	9.995e-01	9.988e-01
1C5	3.156e-03	2.125e-02	0.6	1.6	9.990e-01	9.976e-01
1C6	7.134e-03	4.802e-03	0.5	3.3	9.944e-01	9.880e-01
1C7	4.952e-03	2.841e-03	0.5	2.5	9.968e-01	9.928e-01
1C8	4.485e-03	2.999e-03	0.3	1.7	9.976e-01	9.950e-01
2Ref	4.163e-03	2.469e-03	0.5	1.7	9.982e-01	9.961e-01
2C2	6.313e-03	4.271e-03	0.5	2.4	9.959e-01	9.914e-01
2C3	7.571e-03	5.156e-03	0.5	2.8	9.943e-01	9.881e-01

Table 10: Accuracy of PINN models compared to the DNS simulation. Statistics are computed from the available testing dataset and collected for each component of the flow fields $\mathbf{u} = (\mathbf{v}, p, T)$. They are then averaged for aRMSE, e.g. $\text{aRMSE} = \frac{1}{n_{\mathbf{u}}} \sum_{j=1}^{n_{\mathbf{u}}} \left(\frac{1}{N_T} \sum_{i=1}^{N_T} (\mathbf{u}_{\text{PINN},j}^{(i)} - \mathbf{u}_{\text{DNS},j}^{(i)})^2 \right)^{1/2}$, aMAE, aR_{corr} and aR^2 .

resolution in the observation data. While it was shown that we were able to
 “replay” DNS simulations with the PINNs models, the PINNs accuracy depends
 on many factors including the amount of data and regularization used and the
 choice of the physical quantity under scrutiny. For this particular application,
 DNS temperature data inside the domain was preferentially provided, justifying
 the very good predictive accuracy obtained for the temperature and the vertical
 component of the flow velocity. Indeed, v_z is strongly correlated to the temper-
 ature gradient due to the vertical buoyant forces induced by the consideration
 of the gravity. In plane flow velocities which magnitude is lower were in general
 a bit less accurate. Finally, the pressure field was the least accurate predicted
 quantity. We emphasize that no data was provided for the pressure as boundary
 or initial conditions, which was a hidden state and was obtained indirectly via
 the incompressibility constraint without splitting the Navier-Stokes equations.
 The loss function is typically minimized using SGD algorithm and a large num-
 ber of training points can be randomized within each SGD iteration. Therefore,
 it is also the relative amount of points density which are sampled from those
 data and PDE residuals penalization sources, that are weighting either explic-
 itly or implicitly (depending on the formulation) the importance of the different
 terms. In this paper, we have decided to keep our approach simple and to fine
 tune the error balance by adjusting sampling density of data and residual points.

Acknowledgments. The DNS database has been built using granted access to
 the HPC resources of IDRIS under allocation 2a0326 made by GENCI. We
 thank Dr. Yann Fraigneau for his help and great expertise in the development
 of the DNS SUNFLUIDH solver.

References

- [1] J. N. Kutz, Deep learning in fluid dynamics, *Journal of Fluid Mechanics*
 814 (2017) 1–4.
- [2] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid
 835 mechanics, *Annual Review of Fluid Mechanics* 52 (2020) 477–508.

- [3] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016.
- [4] J. Ling, R. Jones, J. Templeton, Machine learning strategies for systems with invariance properties, *Journal of Computational Physics* 318 (2016) 22–35.
- [5] H. Xiao, P. Cinnella, Quantification of model uncertainty in RANS simulations: A review, *Progress in Aerospace Sciences* 108 (2019) 1–31.
- [6] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence modeling in the age of data, *Annual Review of Fluid Mechanics* 51 (2019) 357–377.
- [7] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics* 378 (2019) 686 – 707. doi:<https://doi.org/10.1016/j.jcp.2018.10.045>.
URL <http://www.sciencedirect.com/science/article/pii/S0021999118307125>
- [8] F. Chillà, J. Schumacher, New perspectives in turbulent Rayleigh–Bénard convection, *Eur. Phys. J. E* 35 (2012) 58.
- [9] A. Castillo-Castellanos, A. Sengupta, B. Podvin, M. Rossi, Cessation and reversals of large-scale structures in square Rayleigh–Bénard cells, *J Fluid Mech.* 877 (2019) 922 – 954.
- [10] J. Kim, C. Lee, Prediction of turbulent heat transfer using Convolutional Neural Networks, *Journal of Fluid Mechanics* 882 (2020).
- [11] E. Fonda, A. Pandey, J. Schumacher, K. R. Sreenivasan, Deep learning in turbulent convection networks, *Proceedings of the National Academy of Sciences* 116 (18) (2019) 8667–8672.
- [12] S. Pandey, J. Schumacher, Reservoir computing model of two-dimensional turbulent convection, arXiv preprint arXiv:2001.10280 (2020).

- [13] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, *Science* 367 (6481) (2020) 1026–1030.
- [14] X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, arXiv preprint arXiv:2003.06496 (2020).
- [15] A. Meade, A. Fernandez, The numerical solution of linear ordinary differential equations by feedforward neural networks, *Mathematical and Computer Modelling* 19 (12) (1994) 1 – 25. doi:[https://doi.org/10.1016/0895-7177\(94\)90095-7](https://doi.org/10.1016/0895-7177(94)90095-7). URL <http://www.sciencedirect.com/science/article/pii/S0895717794900957>
- [16] I. E. Lagaris, A. C. Likas, D. G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, *IEEE Transactions on Neural Networks* 11 (5) (2000) 1041–1049.
- [17] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE transactions on neural networks* 9 (5) (1998) 987–1000.
- [18] K. S. McFall, J. R. Mahan, Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, *IEEE Transactions on Neural Networks* 20 (8) (2009) 1221–1233.
- [19] M. Kumar, N. Yadav, Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey, *Computers & Mathematics with Applications* 62 (10) (2011) 3796 – 3811. doi:<https://doi.org/10.1016/j.camwa.2011.09.028>.
- [20] S. Mall, S. Chakraverty, Application of legendre neural network for solving

ordinary differential equations, *Applied Soft Computing* 43 (2016) 347 – 356. doi:<https://doi.org/10.1016/j.asoc.2015.10.069>.

- [21] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, *Neurocomputing* 317 (2018) 28–41.
- [22] C. Yang, X. Yang, X. Xiao, Data-driven projection method in fluid simulation, *Computer Animation and Virtual Worlds* 27 (3-4) (2016) 415–424.
- [23] E. Haghighat, R. Juanes, SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks, *Computer Methods in Applied Mechanics and Engineering* 373 (2021) 113552. doi:<https://doi.org/10.1016/j.cma.2020.113552>.
- [24] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, *SIAM Journal on Scientific Computing* 43 (5) (2021) A3055–A3081.
- [25] V. Dwivedi, N. Parashar, B. Srinivasan, Distributed physics informed neural network for data-efficient solution to partial differential equations, *arXiv preprint arXiv:1907.08967* (2019).
- [26] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, *Computer Methods in Applied Mechanics and Engineering* 361 (2020) 112732. doi:<https://doi.org/10.1016/j.cma.2019.112732>.
- [27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke,

- Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
 920 URL <https://www.tensorflow.org/>
- [28] Y. Shin, J. Darbon, G. E. Karniadakis, On the convergence and generalization of physics informed neural networks, arXiv e-prints (2020) arXiv-2004.
- [29] S. Mishra, R. Molinaro, Estimates on the generalization error of physics informed neural networks (PINNs) for approximating PDEs, arXiv preprint
 925 arXiv:2006.16144 (2020).
- [30] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536.
- [31] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, in: Neural networks: Tricks of the trade, Springer, 2012, pp.
 930 437–478.
- [32] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
- [33] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feed-forward neural networks, in: Proceedings of the thirteenth international
 935 conference on artificial intelligence and statistics, 2010, pp. 249–256.
- [34] J. Guermond, P. Mineev, J. Shen, An overview of projection methods for incompressible flows, Comp. Meth. Appl. Mech. Eng. 195 (44) (2006) 6011
 – 6045. doi:<https://doi.org/10.1016/j.cma.2005.10.010>.
- [35] M. Belkadi, A. Sergent, Y. Fraigneau, B. Podvin, On the role of roughness valleys in turbulent Rayleigh-Bénard convection, J. Fluid Mech. 923 (A6)
 940 (2021).
- [36] Y. Wang, X. He, P. Tong, Turbulent temperature fluctuations in a closed Rayleigh-Bénard convection cell, J. Fluid Mech. 874 (2019) 263 – 284.

- [37] A. Agrawal, D. Lucor, Y. Fraigneau, B. Podvin, A. Sergent, PDE-
 945 constrained neural network for turbulent Rayleigh-Bénard convection, in:
 Workshop on Frontiers of Uncertainty Quantification in Fluid Dynamics
 (FrontUQ19), 11–13 September 2019.
 URL [https://frontuq19.files.wordpress.com/2019/09/frontuq19_](https://frontuq19.files.wordpress.com/2019/09/frontuq19_book_of_abstracts-1.pdf)
[book_of_abstracts-1.pdf](https://frontuq19.files.wordpress.com/2019/09/frontuq19_book_of_abstracts-1.pdf)
- [38] T. Engels, D. Kolomenskiy, K. Schneider, J. Sesterhenn, Numerical simu-
 950 lation of fluid–structure interaction with the volume penalization method,
 Journal of Computational Physics 281 (2015) 96–115. doi:[https://doi.](https://doi.org/10.1016/j.jcp.2014.10.005)
[org/10.1016/j.jcp.2014.10.005](https://doi.org/10.1016/j.jcp.2014.10.005).
- [39] H. Bijl, D. Lucor, S. Mishra, C. Schwab (Eds.), Uncertainty Quantification
 955 in Computational Fluid Dynamics, Vol. 92 of Lecture Notes in Computa-
 tional Science and Engineering, Springer, Cham, 2013.
- [40] Z. Mao, A. D. Jagtap, G. E. Karniadakis, Physics-informed neu-
 ral networks for high-speed flows, Computer Methods in Ap-
 plied Mechanics and Engineering 360 (2020) 112789. doi:[https:](https://doi.org/10.1016/j.cma.2019.112789)
 960 [//doi.org/10.1016/j.cma.2019.112789](https://doi.org/10.1016/j.cma.2019.112789).
 URL [https://www.sciencedirect.com/science/article/pii/](https://www.sciencedirect.com/science/article/pii/S0045782519306814)
[S0045782519306814](https://www.sciencedirect.com/science/article/pii/S0045782519306814)
- [41] S. Mishra, T. K. Rusch, Enhancing accuracy of deep learning algorithms by
 training with low-discrepancy sequences, arXiv preprint arXiv:2005.12564
 965 (2020).
- [42] O. Canevet, C. Jose, F. Fleuret, Importance sampling tree for large-scale
 empirical expectation, in: M. F. Balcan, K. Q. Weinberger (Eds.), Proceed-
 ings of The 33rd International Conference on Machine Learning, Vol. 48 of
 Proceedings of Machine Learning Research, PMLR, New York, New York,
 970 USA, 2016, pp. 1454–1462.
 URL <http://proceedings.mlr.press/v48/canevet16.html>

- [43] L. Yang, X. Meng, G. E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, *Journal of Computational Physics* 425 (2021) 109913.