Hard constraint projection in a physics informed neural network

M J S Horne*[1], P K Jimack[1], A Khan[2], H Wang[3]

[1] School of Computer Science, University of Leeds, Leeds, LS2 9JT, UK
[2] School of Civil Engineering, University of Leeds, Leeds, LS2 9JT, UK
[3] UCL Centre for Artificial Intelligence, Department of Computer Science, University College London, London, WC1E 6BT, UK

Keywords: Physics Informed Machine Learning, Hard Constraints, Nonlinear PDEs

Abstract: Physics informed machine learning provides a framework to study and simulate fluid dynamics, potentially at a fraction of the computational cost of traditional numerical methods, with the goal of improving the explainability, accuracy and robustness of purely data driven machine learning methods. In this work, we extend the hard constraint projection (HCP) method developed by Chen *et al.* [Journal of Computational Physics, 445, 110624, 2021] from a linear partial differential equation (PDE) to a nonlinear PDE: the 1D Burgers' equation. To strictly impose a discrete approximation of the governing equation, the Burgers' equation is first locally discretised (using a finite difference stencil) then decomposed into a constraint matrix and a prediction matrix. Using tools from linear algebra, the constraint matrix is used to construct a projection matrix. A physics informed neural network (PINN) [Raissi *et al.* Journal of Computational Physics, 378, 686 – 707, 2019] is used to predict the velocity, and a final unlearnable layer projects the velocity prediction to the hyperplane of exact solutions to a discretised form of the Burgers' equation. Our initial results indicate that the HCP-PINN initially trains similarly to an equivalent PINN, and as the PINN loss plateaus, the HCP-PINN loss continues to decline. When tested on a previously unseen uniform grid of points covering the domain, the HCP-PINN exhibits higher accuracy, converging to a data error approximately an order of magnitude lower than the comparable PINN. We also observe that the inclusion of the HCP layer results in better observance of the governing PDE. Our findings suggest that the HCP-PINN can be more robust to the selection of physics collocation points, and less dependent on collocation point quantity.

1. Introduction

The dynamics of fluids are governed by nonlinear partial differential equations (PDEs), the solution of which remains an open field of research, even after centuries of study. Utilising recent developments in computational power, along with decades of simulated computational fluid dynamics data, machine learning is a promising framework to simulate fluid dynamics at a fraction of the computational cost of traditional numerical methods[1]. With the influx of available computationally simulated or experimentally gathered data, one machine learning approach is to feed this data into a neural network to learn the evolution of the system entirely from the data[2, 3]. The model can then make predictions for similar but unseen scenarios faster than the equivalent numerical simulations.

In the setting where we are seeking solutions to a known governing equation, the purely data-driven models neglect this physical knowledge. In 2019, Raissi *et al.* [4] laid the foundations for the field of physics informed machine learning with their novel architecture, the physics informed neural network (PINN). The innovation of the PINN lies in the formulation of the loss function, which is comprised of two weighted and summed elements, the data error (which may itself be a weighted sum of the boundary and internal data error) and the physics error (the residual of the governing equation, calculated using automatic differentiation on the model prediction, with respect to the input coordinates). It has been reported that the predicted solutions generated by a PINN produce more accurate results than the data driven models[4], while being more robust to noisy[5] and sparse[6] training data. The interpretability of the governing equation residual provides a more explainable network, an improvement over the black box learning of purely data driven networks.

However, the introduction of a physics based term in the loss function is not without its challenges, and in some settings the training of the PINN will fail to converge[7]. By examining the loss landscape, Krishnapriyan *et al.* [8] demonstrated that the inclusion of the physics term in the PINN loss causes the optimisation problem to become worse conditioned, with more local minima. Given that nonlinear PDEs have complex solution spaces, sensitive to initial conditions and PDE coefficients, Krishnapriyan *et al.* [8] also identified that the loss landscape became more complex with more difficult parameter

*Corresponding Author, M J S Horne scmho@leeds.ac.uk

regimes (for example, in fluids with high Reynolds numbers).

As the evolution of a fluid is partially determined by the initial and boundary conditions, efforts have been made to build PINNs which must obey these strictly. Mohan *et al.* [9] and Chen *et al.* [10] both employ ghost cells (virtual cells outside the domain), a method from the field of computational fluid dynamics, to discretely enforce the required initial and boundary conditions. Lu *et al.* [11] strictly enforce both Dirichlet and periodic boundary conditions by making an intermediate prediction which is then passed through a user defined function that satisfies the condition by construction.

Some authors have found that strictly enforcing a subset of the governing equations generates more favourable results than weakly enforcing the full set. Mohan *et al.* [9] and Raissi *et al.* [4] both enforced incompressibility in their fluid by making an initial prediction that could be transformed using methods from vector calculus to ensure a divergence free flow (the Helmholtz decomposition[9] and the stream function[4]).

Beucler *et al.* [12] transformed the system constraints into a system of linear equations, with nonlinear variables in the case of nonlinear constraints. Their network predicts a subset of the variables, and then uses the residual of their linear system to define the remaining variables. In this way the full set of variables fulfil the constraints exactly, however the authors report that the residual-determined variables had consistently higher data errors than the predicted variables. Ne´giar *et al.* [13] also transform the system constraints into a system of equations to be solved. They use a neural operator to predict a set of N 1D functions which are combined in a weighted sum to satisfy the physics on the n training coordinates. The system of n constraints and N variables can be solved to find the weights of the summation. If the constraints are nonlinear, then a nonlinear least-squares method is required to find the weights, and if the system is overdetermined ($N < n$) the user must choose between satisfying the constraints on a subset of the data or only satisfying the system approximately.

Chen *et al.* [10] strictly embedded their governing linear PDE by defining a projection that pulls an arbitrary prediction to the closest point on the hyperplane of exact solutions to the discretised form of their governing equation. When Chen *et al.* tested this method on ground water flow through a porous medium, they reported that during training all components of the loss function declined faster, converging in fewer epochs than a comparable PINN. In this paper, we extend the method of hard constraint projection used by Chen *et al.* [10] from a linear PDE to a nonlinear PDE, the 1D Burgers' equation, and start to investigate the effect of the projection on the training of a neural network.

2. Physics Informed Neural Network

We will consider a nonlinear time dependant PDE of the form $\partial \boldsymbol{u}/\partial t = \mathsf{N}(\boldsymbol{x}, \boldsymbol{u}, \nabla \boldsymbol{u}, \ldots)$ where the non-linear operator N is known. We are seeking the solution $\boldsymbol{u}$ at time and space coordinates $(t, \boldsymbol{x})$ subject to initial and boundary conditions. A feed forward neural network (FFNN) with learnable parameters $\boldsymbol{\theta}$ will be used to approximate our PDE solution $\text{FFNN}_{\boldsymbol{\theta}}(t_i, \boldsymbol{x}_i) = \hat{\boldsymbol{u}}_i(t_i, \boldsymbol{x}_i)$.

Given ground truth data of our PDE solution ($\boldsymbol{u}_i(t_i, \boldsymbol{x}_i)$, $i \in \{1, 2, .., N\}$), on the spatio-temporal boundaries and/or within the domain, the loss function can be defined as the distance between the ground truth data and the network's predictions at those coordinates. We will use the mean squared error as the distance measure. The data error based loss is defined on the ground truth data set, see equation (1).

$$\mathsf{L} = \text{DE} = \frac{1}{N}\sum_{i=1}^{N}(||\boldsymbol{u}_i(t_i, \boldsymbol{x}_i) - \hat{\boldsymbol{u}}_i(t_i, \boldsymbol{x}_i)||)^2 \tag{1}$$

Training the model only on the data error (DE) neglects the other information we have on the system, the governing PDE. A PINN (as proposed by Raissi *et al.* [4]) appends the data-driven loss function with a measure of the prediction's deviations from the governing equation, the physics error (PE).

$$\text{PE} = \frac{1}{M}\sum_{i=1}^{M}(||\boldsymbol{r}_i(t_i, \boldsymbol{x}_i)||)^2, \qquad \boldsymbol{r} = \frac{\partial \hat{\boldsymbol{u}}}{\partial t} - \mathsf{N}(\boldsymbol{x}, \hat{\boldsymbol{u}}, \nabla \hat{\boldsymbol{u}}, \ldots) \tag{2}$$

We define the governing equation residual as $\boldsymbol{r}$ in equation (2), which will be zero if the known physics is obeyed exactly. We choose M points in the domain to evaluate the residual at, and the physics error is defined as the mean square of these residuals (PE, (2)). The partial derivatives required to calculate the residual are found using automatic differentiation. Automatic differentiation exploits the differentiable chain of linear sums and nonlinear activation functions to construct exact partial derivatives with respect

*Corresponding Author, M J S Horne scmho@leeds.ac.uk

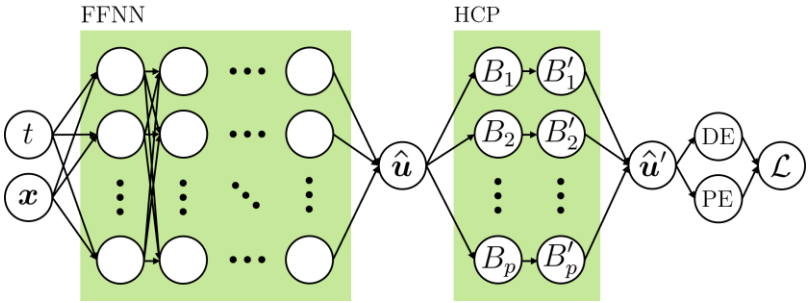

Figure 1: HCP-PINN architecture. The left box is the FFNN with learnable weights and biases, and the right box is the unlearnable HCP.

to the input variables.

$$\mathsf{L} = (\omega)\mathrm{DE} + (1 - \omega)\mathrm{PE} \tag{3}$$

The PINN loss function is defined by equation (3), where ω is a (user-defined) hyperparameter that controls the weighting of the two components.

3. Hard Constraint Projection

The HCP-PINN has the same learnable network and loss function as a PINN, but between the initial prediction of the solution and the loss evaluation there is an unlearnable hard constraint projection layer (see figure 1). The implementation of this method has three offline steps. First, the governing PDE (in this work, the Burgers' equation, (4)) is discretised with a finite difference approximation for the derivatives. Following the methodology used by Chen *et al.* [10], we choose a central finite difference scheme for the spatial derivatives, and a backwards finite difference scheme for the temporal derivative.

Second, the discretised form of the governing equation is decomposed into two matrices, the constraint matrix A (containing all the constant terms) and the prediction matrix B (containing all of the transient terms) such that multiplying AB recovers the discretised governing equation (see equation (5)).

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0. \tag{4}$$

$$AB = \begin{bmatrix} \frac{1}{\Delta t} & \cancel{\Delta t} & \cancel{\Delta x} & (\Delta x)^2 \end{bmatrix} \begin{bmatrix} u(t,x) \\ -u(t-\Delta t, x) \\ \frac{1}{2}u(t,x)[u(t,x+\Delta x) - u(t,x-\Delta x)] \\ -\nu[u(t,x+\Delta x) - 2u(t,x) + u(t,x-\Delta x)] \end{bmatrix} \tag{5}$$

The third step is to define a projection matrix to be used during model evaluation to enforce our governing equation. Using tools from linear algebra, we can define the projection matrix as $P = I - A^T(AA^T)^{-1}A$. When an arbitrary prediction matrix (B) is multiplied by the projection matrix, $PB = B'$, the outcome (B') is the closest point to B that lies on the hyperplane $AB = 0$. For a proof of this, please see the appendix of the paper by Chen *et al.* [10].

During training (or testing) the input to our network is sets of coordinates (t_i, x_i), $i \in \{1, 2, .., p\}$. The model predicts the solution at these points $\hat{u}(t_i, x_i)$. We also evaluate the model at the nearby points required to construct the chosen discretisation stencil (here: $\hat{u}(t_i - \Delta t, x_i)$, $\hat{u}(t_i, x_i + \Delta x)$, $\hat{u}(t_i, x_i - \Delta x)$). With these predictions, the prediction matrix $B(t_i, x_i)$ can be constructed, and consequently projected using the projection matrix we constructed offline, $B'(t_i, x_i) = PB(t_i, x_i)$. The HCP-PINN prediction $\hat{u}'(t_i, x_i)$ can be extracted from $B'(t_i, x_i)$ for a solution that satisfies the discretised governing equation exactly, locally. During training the loss is evaluated on our projected prediction $\hat{u}'(t_i, x_i)$.

4. Results

To test the hard constraint projection, we train a PINN with and without the HCP layer, with all other hyperparameters fixed. The networks both have 5 hidden layers of 50 neurons each, with the hyperbolic tangent (tanh) as the activation function. Adam optimiser[14] with default settings is used to train our

*Corresponding Author, M J S Horne scmho@leeds.ac.uk

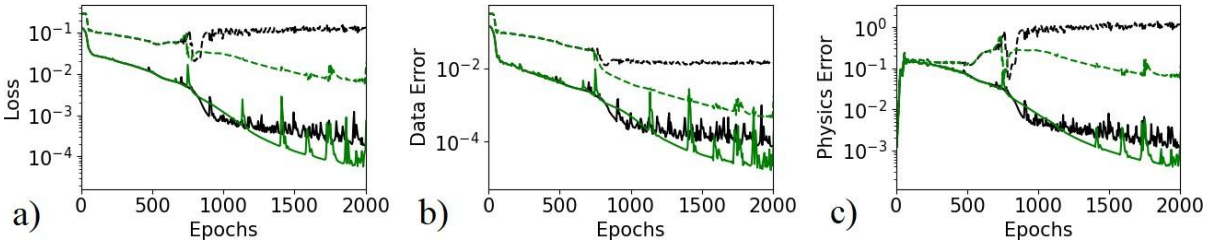

Figure 2: Graphs comparing the value of the (a) loss, (b) data error, and (c) physics error, on the test and training set, during the training of both models. The black lines relate to the PINN, and green lines the HCP-PINN. The solid line is the metric evaluated on the training data, and dashed line on the test data.

networks, and the weight in our loss function (ω in equation (3)) is set to 0.9. The discretisation used in the projection (Δt, Δx in equation (5)) is set to (10^{-9}, 10^{-9}).

The data used to train our models is a single 1D Burgers' equation (4), with domain $(t,\ x) \in (0,\ 1) \times (-1,\ 1)$, with a sine wave initial condition ($t = 0$), Dirichlet boundary conditions ($u = 0$; $x = -1,\ 1$), and viscosity $\nu = 0.01/\pi$. The training dataset is split into ground truth values (on which the DE is calculated) and physics collocation values (on which the PE is calculated). The ground truth values are provided only for the initial and boundary conditions, with 32 at $t = 0$ and 2×33 at $x = -1,\ 1$, a total of $N = 98$, which are represented by $\cdot$ in figure 3a. The physics collocation points are random uniformly sampled throughout the interior of the domain, with $M = 500$, represented by + in figure 3a. The test loss is composed of a DE and PE calculated on the same coordinate set ($N_{\text{test}} = M_{\text{test}} = 1300 = 26 \times 50$), a grid of points blanketing the entire domain. The grid of test coordinates can be observed by the 'pixels' in figure 3.

In figure 2 we have provided the value of the loss, data error, and physics error of the two models during training, evaluated on the training dataset (solid line) and the test dataset (dashed line). We can see that the PINN and HCP-PINN train similarly for the first $\sim$700 epochs, and that around the 1000th epoch the PINN seems to plateau, with the HCP PINN continuing to train, with both components of the loss declining. It appears that the PINN is overfitting after epoch 1000, as the training metrics continue to fall without improvement to the test metrics. The value of the test physics error is lower for the HCP-PINN than for the PINN (see the dashed lines in figure 2c), which we would expect from a model where the governing equations are strictly imposed. We would not expect the physics error of the HCP-PINN to be exactly zero, as the governing equation is only exactly obeyed in its discretised form, locally. The test data error measures how closely the predicted solution matches the ground truth inside the domain. We can see that at the end of training the test DE of the HCP-PINN is over an order of magnitude lower than the test data for the PINN, evidence that the inclusion of the hard physical constraint can improve the prediction accuracy along with enforcing the underlying physics.

For a qualitative analysis we can compare the predictions made by the HCP-PINN (figure 3b) to those made by the PINN (figure 3c). Neither model has access to solution data inside the domain, only information on the physics of the system. We can see that the HCP-PINN is in good agreement with the ground truth (figure 3a), which is reflected in the low test data error we saw in figure 2b, and the mostly white error plot in figure 3d. The PINN prediction however struggles to capture the location of the sharp gradient forming at $x = 0$ (figures 3c, 3e). It has been shown that the choice of sampling strategy for physics collocation points is an influential hyperparameter in the training of PINNs[15], and there is a sparse region around ($t = 0.55,\ x = 0$). Neither model has access to internal solution data, or physics information in the sparse regions. Despite this, the HCP-PINN generates a more realistic solution in this blind region, evidence that the HCP-PINN is more robust to the sampling choice of physics collocation points. We have also found that the HCP-PINN seems to require fewer physics collocation points than a vanilla PINN. In further investigations (not included here) we found that a PINN required $M \approx 1500$ for a comparable test PE to the HCP-PINN with $M = 500$, a promising result when the calculation of the residual is a computational bottle neck for both models.

The hard constraint projection method introduces two types of user defined hyperparameters to the hyperparameter tuning process. The first is the formulation of the two matrices A and B. Even with the order of the finite discretisation method fixed, the decomposition is non-unique (for example, a constant value and its reciprocal can be used to scale A and B). From our preliminary explorations, we have found

*Corresponding Author, M J S Horne scmho@leeds.ac.uk

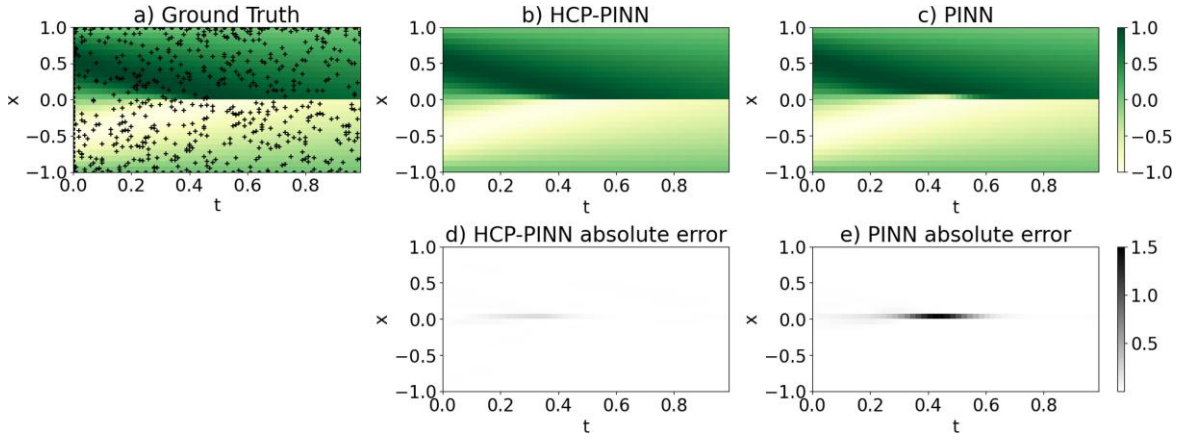

Figure 3: (a) Foreground: the sampling strategy of the training data, with • for the locations of the data error calculations, and + for the locations of the physics error calculations. Background: ground truth associated with the test data set. (b),(c) Predicted solution at the end of training, on the test set. (d),(e) Absolute error in solution predictions. Colour bars shared by plots in the same rows.

that this implementation choice has a non-trivial effect on the model's training and performance. Higher order finite-difference discretisations require more values for the construction of B_i (the discretisation stencil) and as the back-propagation must pass through all elements, the computational expense increases with discretisation order. We intend to investigate if higher order discretisations offer an improvement in performance to offset the computational expense, however this is left for future work.

The second HCP specific hyperparameter is the size of the discretisation, Δt, Δx in equation (5). Decreasing the values of Δt and Δx has no effect on the computational expense of the model, and analysis into this hyperparameter is ongoing. We have observed that for unrealistic values of Δt, $\Delta x \geq 1.0$ the model performs poorly, and that models where $\Delta t > \Delta x$ seem to exhibit higher accuracy (lower test data error) than for $\Delta t < \Delta x$.

When Chen *et al.* [10] applied the hard constraint projection method to a linear PDE they reported that all components of the loss declined faster than for the comparable PINN, but that with enough training the PINN converged to the same value. We speculate that this discrepancy is due to the nonlinear nature of the Burgers' PDE, but we cannot yet discount the influence of PDE parameters (the viscosity of the system) or the dimension of the PDE being solved. We have also started to implement the HCP method in a PINN that predicts solutions to the 2D incompressible Navier Stokes equations. The initial results from the Navier Stokes HCP-PINN have been less favourable than for the Burgers' HCP-PINN. There is evidence that when predicting solutions to the Navier Stokes equations, that the HCP-PINN predictions can be trapped in local minima that the vanilla PINN is able to escape from, and the results we have from our hyperparameter tests are so far inconclusive. We expect that upon the completion of our investigations into the 1D Burgers' HCP-PINN we will be better informed when revisiting the method implementation for higher dimensions and more complex PDEs.

5. Conclusion

We present preliminary results on the HCP-PINN, a PINN with an unlearnable hard constraint projection layer, tested on the 1D Burgers equation, a nonlinear convection-diffusion PDE. The training data comprised of solution data only at the spatial and temporal boundaries, and physics collocation points (on which the governing equation is weakly enforced) through the domain interior.

The HCP-PINN and equivalent PINN trained similarly for an initial training period, after which the PINN with no HCP appeared to plateau, and the HCP-PINN continued to decline until it converged to a lower value than for the PINN. When tested on a grid of points covering the entire space-time domain, the HCP-PINN exhibits higher accuracy than the comparable PINN, converging to a data error roughly one order of magnitude lower. We also observe that the inclusion of the HCP layer results in better observance of the governing PDE. Our findings suggest that the HCP-PINN is more robust to the selection of physics collocation points, and less dependent on collocation point quantity.

*Corresponding Author, M J S Horne scmho@leeds.ac.uk

The authors continue investigations into HCP specific hyperparameters, alongside planning to explore other PDE parameter regimes, higher dimensional Burgers' equations, and other nonlinear PDEs such as the incompressible Navier Stokes equations.

References

[1] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics," *Annual Review of Fluid Mechanics*, vol. 52, no. 1, pp. 477 – 508, 2020.

[2] Z. Tong and Y. Li, "Real-time reconstruction of contaminant dispersion from sparse sensor observations with gappy pod method," *Energies*, vol. 13, no. 8, 2020.

[3] T. Nakamura, K. Fukami, K. Hasegawa, Y. Nabae, and K. Fukagata, "Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow," *Physics of Fluids*, vol. 33, no. 2, p. 025116, 2021. Publisher: AIP Publishing.

[4] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686 – 707, 2019.

[5] F. Shone, N. Ravikumar, T. Lassila, M. MacRaild, Y. Wang, Z. A. Taylor, P. Jimack, E. Dall'Armellina, and A. F. Frangi, "Deep physics-informed super-resolution of cardiac 4d-flow mri," in *Information Processing in Medical Imaging* (A. Frangi, M. de Bruijne, D. Wassermann, and N. Navab, eds.), pp. 511–522, Springer Nature Switzerland, 2023.

[6] S. Cai, Z. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, "Flow over an espresso cup: inferring 3-d velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks," *Journal of Fluid Mechanics*, vol. 915, p. A102, 2021.

[7] C. Bajaj, L. McLennan, T. Andeen, and A. Roy, "Recipes for when physics fails: recovering robust learning of physics informed neural networks," *Machine Learning: Science and Technology*, vol. 4, no. 1, p. 015013, 2021.

[8] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," in *Advances in Neural Information Processing Systems* (M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds.), vol. 34, pp. 26548–26560, Curran Associates, Inc., 2021.

[9] A. Mohan, N. Lubbers, M. Chertkov, and D. Livescu, "Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence," *Physical Review Fluids*, vol. 8, p. 014604, 01 2023.

[10] Y. Chen, D. Huang, D. Zhang, J. Zeng, N. Wang, H. Zhang, and J. Yan, "Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method," *Journal of Computational Physics*, vol. 445, p. 110624, 2021.

[11] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM Journal on Scientific Computing*, vol. 43, no. 6, pp. B1105–B1132, 2021.

[12] T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, and P. Gentine, "Enforcing Analytic Constraints in Neural Networks Emulating Physical Systems," *Physical Review Letters*, vol. 126, no. 9, 2021. Publisher: American Physical Society (APS).

[13] G. Ne´giar, M. W. Mahoney, and A. S. Krishnapriyan, "Learning differentiable solvers for systems with hard constraints," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, OpenReview.net, 2023.

[14] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[15] J. Florido, H. Wang, A. Khan, and P. K. Jimack, "Investigating guiding information for adaptive collocation point sampling in pinns," in *Computational Science – ICCS 2024: 24th International Conference, Malaga, Spain, July 2–4, 2024, Proceedings, Part III*, (Berlin, Heidelberg), p. 323–337, Springer-Verlag, 2024.

*Corresponding Author, M J S Horne scmho@leeds.ac.uk