PHYSICS INFORMED NEURAL NETWORKS FOR MAGNETOHYDRODYNAMIC EQUATIONS

Eva Jaillon
ENSTA Paris - Institut Polytechnique de Paris
Palaiseau, 91120, France
eva.jaillon@ensta.fr

Pavlos Protopapas
Harvard University
Cambridge, MA, 02138, USA
pprotopapas@g.harvard.edu

ABSTRACT

We present a data-free Physics-Informed Neural Network (PINN) framework for solving a coupled system of magnetohydrodynamic (MHD) equations. Rather than introducing a new network architecture, this work focuses on a practical training strategy that enables stable optimization of large, strongly coupled PINN systems using only physical residuals. The proposed approach employs a stepwise initialization protocol, in which subsets of equations and output variables are progressively introduced while reusing previously trained weights. This structure naturally supports transfer learning across different initial conditions and optimization regimes. We demonstrate that the resulting model can solve the full eight-equation MHD system without observational data and can be efficiently adapted to new initial conditions using multi-head transfer learning and hybrid Adam/L-BFGS optimization, achieving substantial reductions in training time while preserving solution accuracy.

1 INTRODUCTION

Magnetohydrodynamics (MHD) couples Maxwell's equations with fluid mechanics (Alfvèn, 1942), yet strong nonlinearity and tight coupling challenge traditional solvers. While Physics-Informed Neural Networks (PINNs) (Karniadakis et al., 2021; Raissi et al., 2019; Desai et al., 2022) embed physical laws into loss functions (Rao et al., 2020), scaling them to large coupled systems often leads to optimization instabilities and training failure in data-free settings (Wang et al., 2020).

Building upon Bard & Dorelli (2021), this study achieves the data-free resolution of a full eight-equation coupled MHD system. First, we introduce the Stepwise Initialized Physics Network (SIPN): a training methodology that stabilizes optimization by progressively introducing equations while preserving learned weights. Second, while this incremental protocol broadly enables transfer learning (Goswami et al., 2020) across physical regimes, we specifically leverage it to facilitate systematic parameter reuse across distinct initial conditions. Third, we demonstrate that combining first-order Adam optimization with a final-stage L-BFGS refinement substantially accelerates convergence while preserving solution accuracy. This work emphasizes the practical stabilization of PINN training for strongly coupled systems in data-scarce regimes, as opposed to providing new theoretical convergence guarantees.

2 RESOLVING MHD EQUATIONS WITHOUT OBSERVATIONAL DATA

2.1 GENERALITY

The MHD framework, formalized by Alfvèn (1942), governs the interaction between fluid motion and magnetic fields. In our research, we utilize the specific set of eight coupled equations detailed in the work of Bard & Dorelli (2021). These equations, which include mass conservation, momentum, energy, and induction laws (see Appendix A.2 for the full system based on MHD equations of Appendix A.1), represent a significant challenge for standard PINN architectures due to their high dimensionality and non-linear coupling. By eliminating reliance on external data, we use physical residuals alone to drive the optimization process.

2.2 IMPLEMENTATION

Direct training of a single multi-output PINN on the full eight-equation system consistently failed to converge (see Appendix A.3), rendering the simultaneous resolution of the coupled MHD equations intractable despite extensive optimization. To address this fundamental challenge, we developed an incremental training methodology as a Stepwise Initialized Physics Network (SIPN). This approach replaces the conventional multi-output single-network architecture with multiple specialized networks, one per output variable. The SIPN method proceeds through two key phases:

1. Initial training on a reduced system containing only core equations (minimizing outputs and network complexity to ensure convergence without significant error).
2. Progressive system expansion through: (i) addition of new equations and corresponding networks, (ii) weight initialization using previously stored parameters, followed by (iii) full-network retraining.

This iterative procedure continues until complete system integration is achieved, enabling stable solutions to the full MHD equation set.

In order to solve the MHD equations, we use three-layer neural networks with 128 neurons in the hidden layers. Using the NeuroDiffEq library (Chen et al., 2020), we first establish stable solutions for five outputs (ρ, v_x, P, B_y, B_z) corresponding to three core equations. Forcing functions are incorporated via subtraction solely to enable quantitative validation against analytical solutions (see Appendix A.4 and A.5), without simplifying the optimization problem or reducing physical coupling. The NeuroDiffEq library (Chen et al., 2020) enables solving the complete eight-equation system through a two-stage approach; once the first stage is completed, the remaining equations can be incorporated simultaneously while maintaining solution stability.

2.3 RESULTS

Following the final training phase using the SIPN method, all eight equations were successfully solved with correct solutions obtained for each variable. As a representative case, we examine the evolution of the density variable ρ between the initial and final training phases.

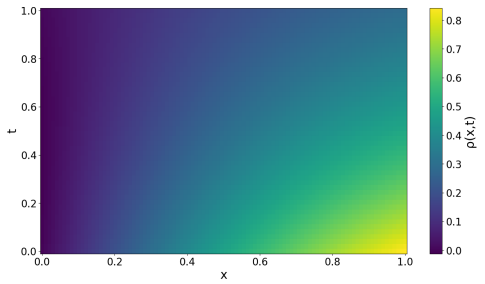

Figure 1: ρ after the first training

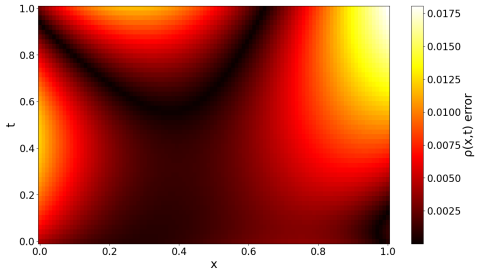

Figure 2: ρ error after the first training

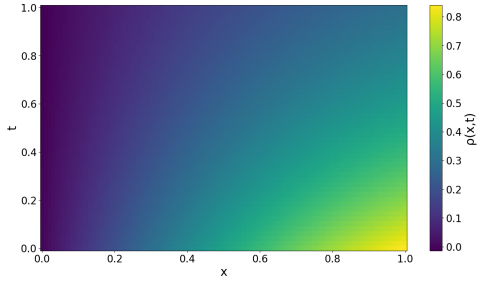

Figure 3: ρ after the second training

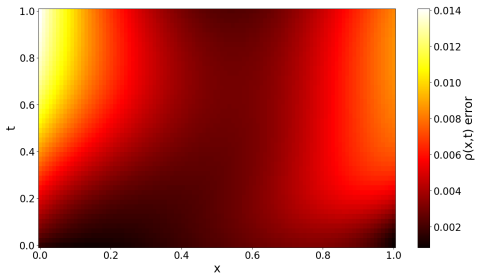

Figure 4: ρ error after the second training

3 TRANSFER LEARNING ON INITIAL CONDITIONS

3.1 IMPLEMENTATION

In this work, transfer learning (Goswami et al., 2020)is defined as the systematic reuse of parameters across different initial conditions (ICs) within the same governing PDE system, rather than transfer across unrelated physical domains. Following the SIPN methodology, we implement this approach (Goswami et al., 2020) based on a multi-head architecture (Zou & Karniadakis, 2023), where each dedicated head corresponds to a distinct set of ICs. To maintain optimization stability, we omit head-specific forcing functions—which otherwise introduce prohibitive complexity—and instead validate results against a dedicated solver (see Appendix A.6). A third head is initialized using the pre-trained weights of the second head to leverage learned representations of a similar physical regime. The ICs for all heads maintain $\rho = 2$ and $P = B_{x,y,z} = \exp(-x - t)$, while the velocity profiles are set to $v_{x,y,z} = x$ for head 1, $v_{x,y,z} = x^3$ for head 2, and $v_{x,y,z} = x^2$ for the transfer-learned head 3. This strategy facilitates efficient convergence by exploiting the proximity of the target solution to the previously optimized parameters.

3.2 RESULTS

Figure 5 displays the solution for the v_y component obtained using transfer learning. A comparison with Figures 9 and 10 in Appendix A.7, which depict the same output derived from classical training applied to head 1 and head 2 respectively, reveals that both methods yield results of comparable accuracy. All solutions exhibit low errors, indicating consistent solution quality across training strategies. This result shows that transfer learning achieves comparable precision while requiring substantially less computational time.

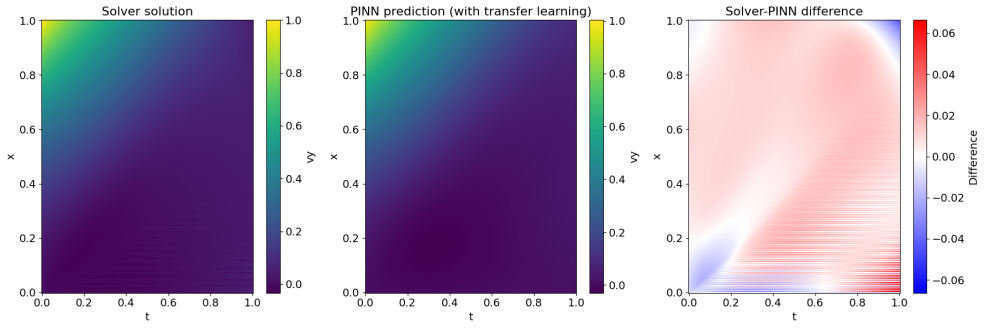

Figure 5: v_y after the transfer learning on IC

3.3 EFFICIENCY OF TRANSFER LEARNING METHOD COMPARED TO CLASSICAL TRAINING METHODS

As expected (Goswami et al., 2020), transfer learning demonstrates significantly improved computational efficiency. The current implementation requires approximately 5 minutes of runtime, compared to approximately 2 hours for full training of all eight neural networks (1 hour for the initial five outputs and an additional hour for all eight outputs using the SIPN method). The development was conducted in Visual Studio Code and all computations performed on a Dell Latitude 5530 laptop equipped with a Core i5 processor. The relative efficiency of transfer learning versus conventional training approaches can be quantitatively assessed through loss function evolution. The comparison is presented in the Appendix A.8. It should be noted that the conventional method's performance represents only the terminal phase of a multi-stage training process, requiring substantial prior optimization of individual network components. Our analysis reveals faster convergence when using transfer learning compared to conventional training. As evidenced in Figure 12, the transfer learning approach achieves a loss value of 10^{-5} within approximately 1,500 epochs, after which the solution stabilizes. By contrast, Figure 11 shows that traditional training does not reach this loss threshold even after 10,000 epochs of optimization.

4 TRANSFER LEARNING USING THE L-BFGS OPTIMIZER

4.1 ANOTHER OPTIMIZER

Until now, our models relied on the Adam optimizer, which is a first-order method combining momentum and adaptive learning rates for fast initial convergence (Kingma & Ba, 2015; Ruder, 2016). However, while Adam is efficient at navigating complex loss landscapes, it often lacks the precision required for final convergence in PINNs. To improve efficiency and accuracy, we transitioned to the L-BFGS optimizer—a second-order quasi-Newton method—specifically for the final training stages, as suggested by Naimi (2024) and Seo (2024). Unlike standard second-order methods, L-BFGS is memory-efficient as it approximates the Hessian via a limited gradient history (Nocedal, 1980), enabling high-precision minimization. In our implementation, Adam is used for initial training across all layers to ensure rapid convergence. Subsequently, L-BFGS is applied exclusively during the final optimization stage and only to the network's last layer, serving to refine the well-initialized solution with higher precision without replacing the first-order training.

4.2 COMPARED RESULTS AND EFFICIENCY

Consistent with our expectations, implementing the L-BFGS optimizer for the final layer yielded substantial improvements in training efficiency. As shown in Appendix A.9, the loss function stabilizes at its minimum value within just one epoch, demonstrating rapid convergence behavior. By comparison, the Adam-only optimizer required approximately 4,000 epochs to achieve comparable results for the same physical system.

Figure 6 displays the v_y output obtained through transfer learning with L-BFGS optimization. When compared with the results from the non-L-BFGS approach (Figure 5), we observe nearly identical solution accuracy. These results indicate that L-BFGS final-layer optimization preserves solution fidelity while substantially reducing the number of training epochs required.

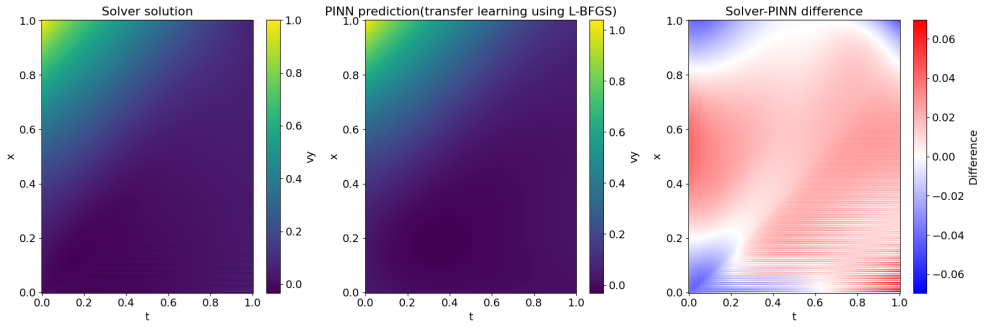

Figure 6: v_y evolution for transfer learning using the L-BFGS optimizer

5 CONCLUSION

This study shows that a data-free Physics-Informed Neural Network can be trained to solve a full system of eight coupled MHD equations using only physical residuals, provided that training is carefully structured to account for strong coupling. By introducing a stepwise training protocol together with a hybrid Adam/L-BFGS optimization strategy, we improve both training stability and convergence efficiency relative to standard PINN training. The proposed incremental and transfer learning approach highlights the practical potential of PINNs for large, strongly coupled physical systems in data-scarce settings.

Future work will explore the application of this framework to inverse problems (Kabanikhin, 2008), including the reconstruction of magnetic field components from partial measurements. In addition, extending the approach to more complex physical regimes, more challenging MHD benchmarks and higher-dimensional formulations (Han et al., 2018) represents a natural direction for further investigation.

REFERENCES

Hannes Alfvèn. Existence of electromagnetic-hydrodynamic waves. *Nature*, 1942.

C. Bard and J.C. Dorelli. Neural network reconstruction of plasma space-time. *Frontiers*, 2021.

J.C. Butcher. A history of runge-kutta methods. *Applied Numerical Mathematics*, 1996.

Feiyu Chen, David Sondak, Pavlos Protopapas, Marios Mattheakis, Shuheng Liu, Devansh Agarwal, and Marco Di Giovanni. Neurodiffeq: A python package for solving differential equations with neural networks. *Journal of Open Source Software*, 2020.

Shaan Desai, Marios Mattheakis, Hayden Joy, Pavlos Protopapas, and Stephen Roberts. One-shot transfer learning of physics-informed neural networks. 2022.

Somdatta Goswami, Cosmin Anitescu, Souvik Chakraborty, and Timon Rabczuk. Transfer learning en hanced physics informed neural network for phase-field modeling of fracture. *Theoretical and Applied Fracture Mechanics*, 2020.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 2018.

Sergey Kabanikhin. Definitions and examples of inverse and ill-posed problems. *Journal of Inverse and Ill-Posed Problems*, 2008.

George Em Karniadakis, Yannis Kevrekidis, Lu Lu, and Paris Perdikaris. Physics-informed machine learning. *Nature Reviews Physics*, 2021.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic opti mization. *ICLR*, 2015.

Hend Naimi. Optimizers for training physics-informed neural networks. *Freie Uni versität Berlin Master Thesis*, 2024.

Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 1980.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019.

Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 2020.

Sebastian Ruder. An overview of gradient descent optimization algorithms. 2016.

Jaemin Seo. Solving real-world optimization tasks using physics-informed neural comput ings. *Nature*, 2024.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Review*, 2020.

Zongren Zou and George Em Karniadakis. L-hydra: Multi-head physics-informed neural networks. *Arxiv*, 2023.

A APPENDIX

The implementation code for this study is publicly hosted on the following GitHub repository: https://github.com/EvaJaillon/ICLR2026-AI-PDE-Submission-Code

A.1 MHD EQUATIONS

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{1}$$

where ρ is the mass density and u is the fluid velocity.

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} \right) = -\nabla P + \frac{1}{\mu_0}(\nabla \times \mathbf{B}) \times \mathbf{B} + \rho \nu \nabla^2 \mathbf{u} \tag{2}$$

Here, P denotes the pressure, B the magnetic field, μ_0 the permeability of free space, and ν the kinematic viscosity.

$$\frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p + \gamma p (\nabla \cdot \mathbf{u}) = 0 \tag{3}$$

The adiabatic index γ ($= c_p/c_v$) appears for compressible flows, linking pressure changes to fluid expansion/compression.

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B}) + \eta \nabla^2 \mathbf{B} \tag{4}$$

The magnetic diffusivity η accounts for resistive effects when non-zero.

A.2 MHD EQUATIONS USED BY BARD AND DORELLI

$$\frac{\partial \rho}{\partial t} + v_x \frac{\partial \rho}{\partial x} + \rho \frac{\partial v_x}{\partial x} = 0 \tag{5}$$

$$\rho \frac{\partial v_x}{\partial t} + \rho v_x \frac{\partial v_x}{\partial x} + \frac{\partial P}{\partial x} + B_y \frac{\partial B_y}{\partial x} + B_z \frac{\partial B_z}{\partial x} - \rho \nu \frac{\partial^2 v_x}{\partial x^2} = 0 \quad \text{(for x)} \tag{6}$$

$$\rho \frac{\partial v_y}{\partial t} + \rho v_x \frac{\partial v_y}{\partial x} - B_x \frac{\partial B_y}{\partial x} = 0 \quad \text{(for y)} \tag{7}$$

$$\rho \frac{\partial v_z}{\partial t} + \rho v_x \frac{\partial v_z}{\partial x} - B_x \frac{\partial B_z}{\partial x} = 0 \quad \text{(for z)} \tag{8}$$

$$\frac{\partial P}{\partial t} + \gamma P \frac{\partial v_x}{\partial x} + v_x \frac{\partial P}{\partial x} = 0 \tag{9}$$

$$\frac{\partial B_x}{\partial t} = 0 \quad \text{(for x)} \tag{10}$$

$$\frac{\partial B_y}{\partial t} + v_x \frac{\partial B_y}{\partial x} + B_y \frac{\partial v_x}{\partial x} - B_x \frac{\partial v_y}{\partial x} = 0 \quad \text{(for y)} \tag{11}$$

$$\frac{\partial B_z}{\partial t} + v_x \frac{\partial B_z}{\partial x} + B_z \frac{\partial v_x}{\partial x} - B_x \frac{\partial v_z}{\partial x} = 0 \quad \text{(for z)} \tag{12}$$

A.3 FAILURE OF CONVERGENCE FOR B_y WITHOUT THE SIPN METHOD

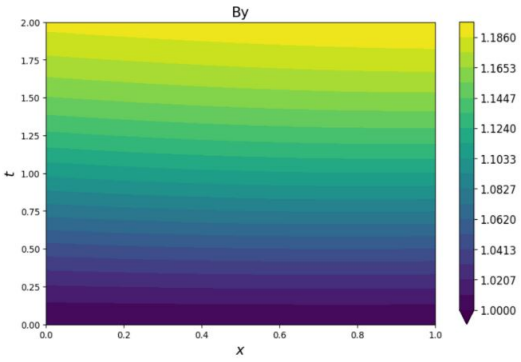

Figure 7: B_y obtained without the SIPN method

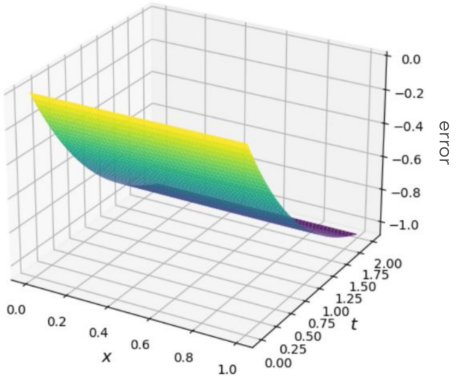

Figure 8: B_y error without the SIPN method

A.4 SOLUTIONS AND FORCING FUNCTIONS FOR THE FIRST STAGE

The solutions correspond to the three core equations in appendix A.2.5,6,9 : $\rho = sin(x)exp(-t)$, $v_x = exp(-t)$ and $P = B_y = B_z = exp(-x-t)$. Forcing functions :

$$f_1(x,t) = -sin(x)exp(-t) + cos(x)exp(-2t) \tag{13}$$

$$f_2(x,t) = -sin(x)exp(-2t) - exp(-x-t) - 2exp(2(-x-t)) \tag{14}$$

$$f_5(x,t) = -exp(-x-t) - exp(-x-2t) \tag{15}$$

A.5 SOLUTIONS AND FORCING FUNCTIONS FOR THE SECOND STAGE

The resulting solutions for the complete system are as follows: $\rho = sin(x)exp(-t)$,$v_x = v_y = v_z = exp(-t)$ and $P = B_x = B_y = B_z = exp(-x-t)$ With the forcing functions :

$$f_1(x,t) = -sin(x)exp(-t) + cos(x)exp(-2t) \tag{16}$$

$$f_2(x,t) = -sin(x)exp(-2t) - exp(-x-t) - 2exp(2(-x-t)) \tag{17}$$

$$f_3(x,t) = f_4(x,t) = -sin(x)exp(-2t) + exp(2(-x-t)) \tag{18}$$

$$f_5(x,t) = -exp(-x-t) - exp(-x-2t) \tag{19}$$

$$f_6(x,t) = -exp(-x-t) \tag{20}$$

$$f_7(x,t) = f_8(x,t) = -exp(-x-t) - exp(-x-2t) \tag{21}$$

A.6 SOLVER AND BOUNDARY CONDITIONS

In addition to initial conditions, boundary conditions must be incorporated when performing transfer learning on ICs (Initial Conditions). These are essential for the solver to obtain a well-posed problem with sufficient constraints. To enable direct comparison between the PINN predictions and solver results, we explicitly integrate these boundary conditions into the PINN architecture.

The implemented boundary conditions are as follows: $\rho(x=0) = 2$, $v_x(x=0) = v_y(x=0) = v_z(x=0) = 0$ and $P(x=0) = B_x(x=0) = B_y(x=0) = B_z(x=0) = exp(-t)$

We aim to solve a system of partial differential equations (PDEs) describing magnetized plasma dynamics. The numerical solver employs the Method of Lines, which transforms the original PDEs into a system of ordinary differential equations (ODEs). The spatial domain is discretized into a mesh of n_x points, with spatial derivatives ($\partial/\partial x$ approximated using central finite differences.)

The eight physical variables $(\rho, v_x, v_y, v_z, P, B_x, B_y, B_z)$ are concatenated into a single state vector y of dimension $8 \times n_x$. This formulation reduces the PDE system to a coupled ODE system:

$$\frac{d\mathbf{y}}{dt} = F(\mathbf{y}, t) \quad \text{where} \quad \mathbf{y} \in \mathbb{R}^{8 \times n_x} \tag{22}$$

The resulting ODE system is then integrated numerically using SciPy's implementation of the adaptive Runge-Kutta (Butcher, 1996) 4th/5th order method (RK45).

A.7 v_y OBTAINED BY CLASSICAL TRAINING FOR HEAD 1 AND HEAD 2

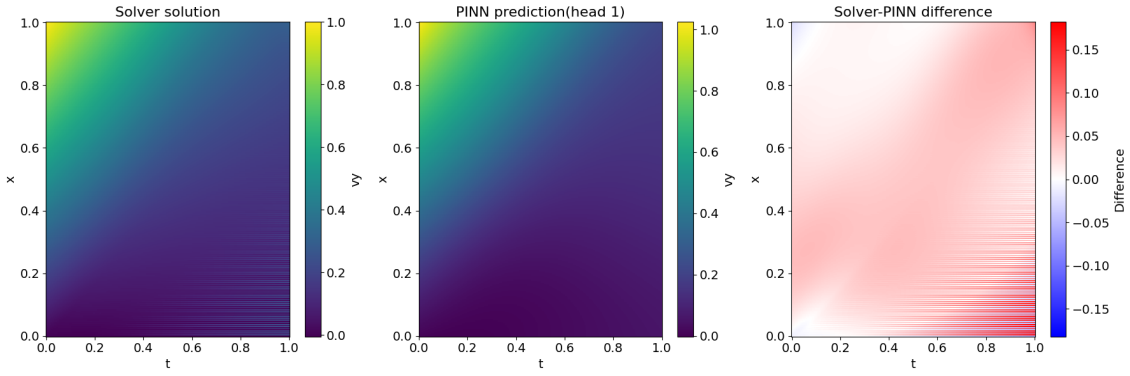

Figure 9: v_y after the second training on IC for head 1

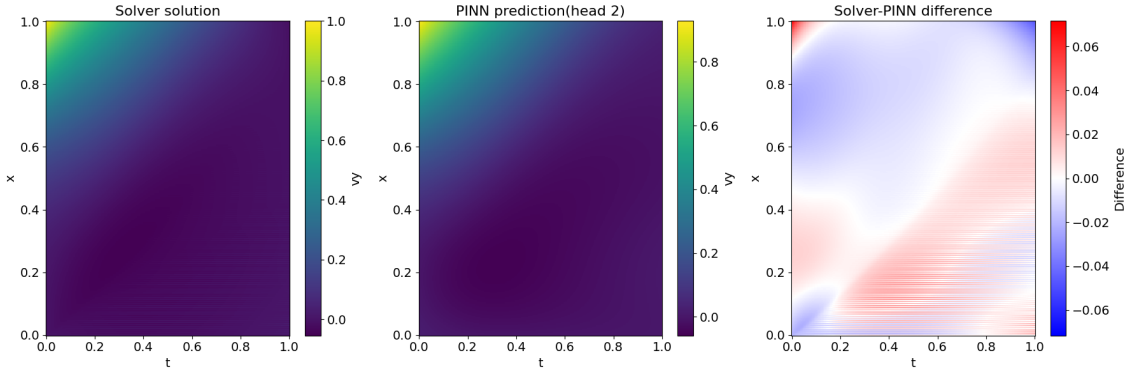

Figure 10: v_y after the second training on IC for head 2

A.8 LOSS FUNCTION COMPARISON USING TRANSFER LEARNING OR CLASSICAL METHOD

(comparison for the specific case of $\nu = 0.01$, where the neural network is trained on both $\nu_1 = 0.005$ and $\nu_2 = 0.05$ with initialization on ν_2.)

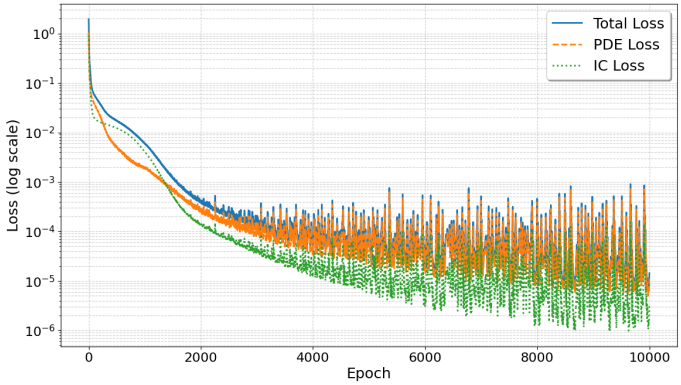

Figure 11: Loss function for classical training

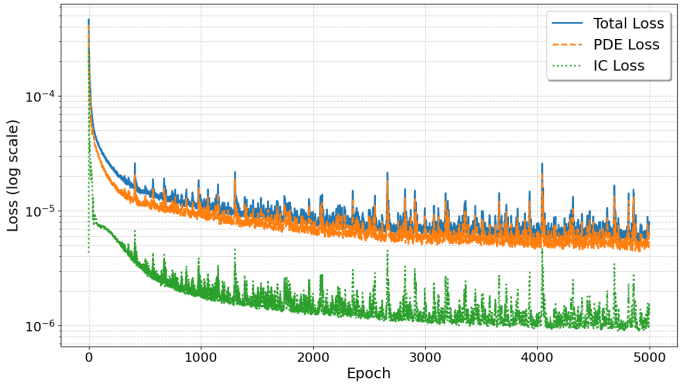

Figure 12: Loss function for transfer learning

A.9 LOSS FUNCTION FOR TRANSFER LEARNING USING THE L-BFGS OPTIMIZER

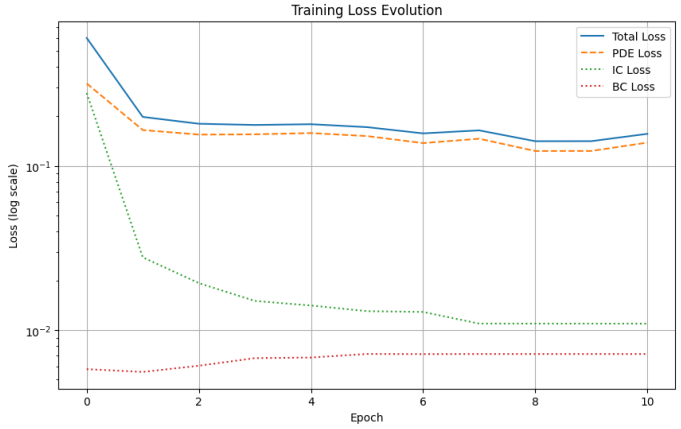

Figure 13: Loss function for transfer learning using the L-BFGS optimizer