NARLabs
National Applied Research Laboratories
National Center for High-performance Computing

# Neural Network for Magnetohydrodynamic Simulations

Y. C. Chen[1], T. C. Tsai[1], W. S. Chen[1], and K. H. Lee[2]

[1]National Center for High-Performance Computing, National Applied Research Laboratories, Hisinchu City, 30076, Taiwan
[2]Institute of Earth Science, Academia Sinica, Taipei City, 11529, Taiwan

## Abstract

We applied the Physics-informed Neural Networks (PINNs) to the magnetohydrodynamic (MHD) simulations by building a Neural Network (NN) to find solutions for the MHD equations. We chose the Brio-Wu shock tube problem as our test MHD sample. The inputs of the NN model are coordinates of space-time and the outputs of the model are the magnetic field, the plasma bulk velocity, the plasma mass density, and the plasma thermal pressure. We combined the MHD equations and initial conditions into the loss function for a physics constraint to the neural network model. We show our premilitary training results and discussion in the end.

## Introduction

Plasma is an ionized gas with positive and negative charge particles and the interactions between particles are complicated physics phenomena. One of the methods to study plasma is Magnetohydrodynamic (MHD) simulation. The MHD simulation could describe plasma properties successfully. However, simulation usually costs amounts of computational power and time. It also has a mesh problem for poor simulation results. With the advances in technology, neural networks can estimate complicated data. One of the popular neural networks was called physics-informed neural networks (PINNs) and was introduced by Raissi et al. (2019) for solving partial differential equation (PDE). The method includes equations to loss item for constraining the outputs of the neural network to approximate the solutions of the equations. Compared to traditional simulation methods, PINNs have the benefits of meshfree and are flexible to be applied to different problems. Wang et al. (2021) proposed a PINN for solving flow problems. Recently, Bard and Dorelli (2021) used PINNs to reconstruct the plasma properties in several shock problems successfully. In this work, we focus on the 1D Brio-Wu shock tube problem to investigate how to implement PINNs in plasma and if it works or not. We build a multi-layer neural network as an approximation of the solution to MHD equations by Pytorch.

## Neural Network

We derived the one-dimension (1D) MHD equations from Tsai et al. (2015) and showed the equations below:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0$$

$$\mathbf{U} = \begin{bmatrix} B_x \\ B_y \\ B_z \\ \rho \\ V_x \\ V_y \\ V_z \\ E \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} 0 \\ B_y V_x - B_x V_y \\ B_z V_x - B_x V_z \\ \rho V_x \\ \rho V_x^2 + (p + \frac{B^2}{2}) - B_x^2 \\ \rho V_x V_y - B_x B_y \\ \rho V_x V_z - B_x B_z \\ (\frac{1}{2}\,\rho V^2 + \frac{\gamma p}{\gamma - 1} + B^2) - B_x(B_x V_x + B_y V_y + B_z V_z) \end{bmatrix}$$

$$M_x = \rho V_x \qquad M_y = \rho V_y \qquad M_z = \rho V_z \qquad E = \frac{p}{\gamma-1} + \frac{1}{2}\rho V^2 + \frac{B^2}{2} = \frac{p}{\gamma-1} + \frac{1}{2}\rho\left(V_x^2 + V_y^2 + V_z^2\right) + \frac{(B_x^2 + B_y^2 + B_z^2)}{2}$$

We rearranged the equations into the forms of the outputs of the neural network with respect to the inputs and showed the equations below:

$$\frac{\partial B_x}{\partial x} = 0 \qquad \frac{\partial B_x}{\partial t} = 0 \qquad \frac{\partial B_y}{\partial t} + V_x\frac{\partial B_y}{\partial x} + B_y\frac{\partial V_x}{\partial x} - B_x\frac{\partial V_y}{\partial x} = 0 \qquad \frac{\partial B_z}{\partial t} + B_z\frac{\partial V_x}{\partial x} + V_x\frac{\partial B_z}{\partial x} - B_x\frac{\partial V_z}{\partial x} = 0 \qquad \frac{\partial \rho}{\partial t} + V_x\frac{\partial \rho}{\partial x} + \rho\frac{\partial V_x}{\partial x} = 0$$

$$\rho\frac{\partial V_x}{\partial t} + \rho V_x\frac{\partial V_x}{\partial x} + \frac{\partial p}{\partial x} + B_y\frac{\partial B_y}{\partial x} + B_z\frac{\partial B_z}{\partial x} = 0 \qquad \rho\frac{\partial V_y}{\partial t} + \rho V_x\frac{\partial V_y}{\partial x} - B_x\frac{\partial B_y}{\partial x} = 0 \qquad \rho\frac{\partial V_z}{\partial t} + \rho V_x\frac{\partial V_z}{\partial x} - B_x\frac{\partial B_z}{\partial x} = 0 \qquad \frac{\partial p}{\partial t} + V_x\frac{\partial p}{\partial x} + \gamma p\frac{\partial V_x}{\partial x} = 0$$

The initial condition is shown in the following:

$$(B_{x_1}, B_{y_1}, B_{z_1}, V_{x_1}, V_{y_1}, V_{z_1}, \rho_1, p_1) = (0.75, +1, 0, 0, 0, 0, 1, 1) \quad \text{for } x < 0$$
$$(B_{x_2}, B_{y_2}, B_{z_2}, V_{x_2}, V_{y_2}, V_{z_2}, \rho_2, p_2) = (0.75, -1, 0, 0, 0, 0, 0.125, 0.1) \quad \text{for } x > 0$$

The initial condition was connected by a hyperbolic tangent transition shown below:

$$A(x) = \frac{A_2 + A_1}{2} + \frac{A_2 - A_1}{2}\tanh\left(\frac{x}{W}\right)$$

A represents $B_x$, $B_y$, $B_z$, $V_x$, $V_y$, $V_z$, p, $\rho$ and W = 0.0085. We included the equations shown above and the initial condition in the loss function. The total loss is Loss = $\text{Loss}_{\text{equations}}$ + $\text{Loss}_{\text{initial condition}}$. We built a neural network with 6 layers which have an input layer, an output layer, and several hidden layers, and the structure of the model is shown in Table 1. The input of the model is space-time coordinate (x, t) and the output is $B_x$, $B_y$, $B_z$, $\rho$, $V_x$, $V_y$, $V_z$, p. The x ranges from -1 to 1 and t ranges from 0 to 1. The data points are 10000 and the learning rate = 0.0001. We used a stochastic gradient descent (SGD) optimizer with 500 epochs in our training.

| Layer | Input | Output | Activation function |
|---|---|---|---|
| **1** | 2 | 8 | Tanh |
| **2** | 8 | 64 | Tanh |
| **3** | 64 | 256 | Tanh |
| **4** | 256 | 256 | Tanh |
| **5** | 256 | 64 | Tanh |
| **6** | 64 | 8 | |

Table 1. Model structure

## Results and Discussion

A loss value is the difference between expected value and model output. The loss values with epochs are called a learning curve to evaluate whether a model learns well during its training. We showed our learning curve in Figure 1. and found our loss value decreases with epochs. This indicates the outputs from the trained model are near the expected value. We further checked equation and initial condition loss and showed the results in Figure 2. We found the equation loss decreases before the 100th epoch and increases after the 100th epoch. This indicates that the model output is deviated from the expected value and the model is overfitting. We also checked the individual loss inside the equation loss and found only the f0 ($\partial B_x/\partial x$=0) loss shows a steady decrease with epochs.

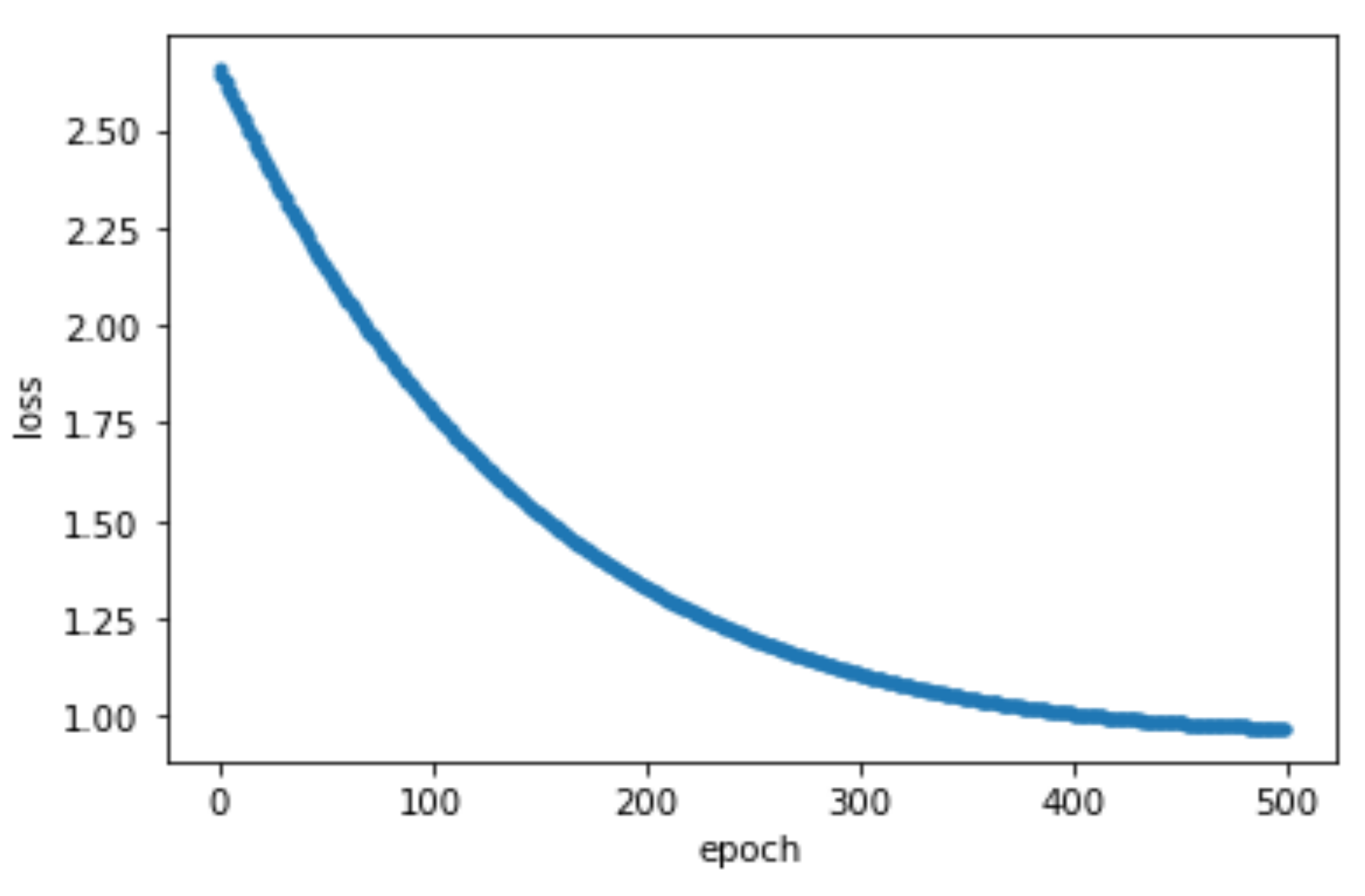

Figure 1. Total loss versus epochs

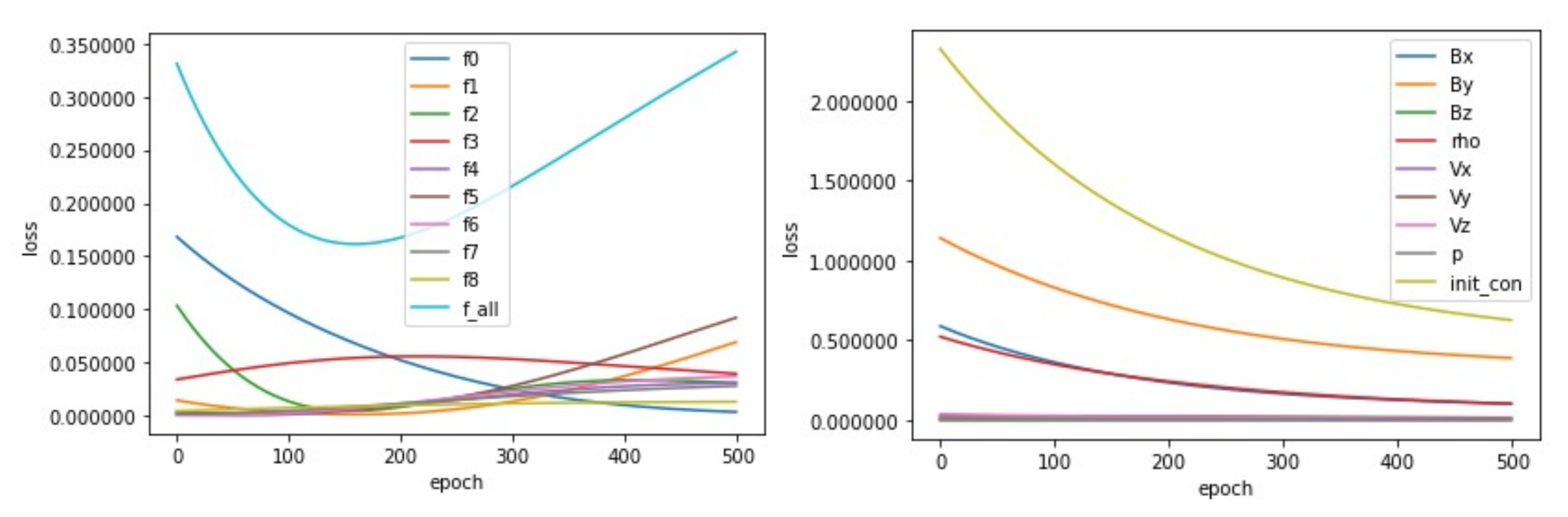

Figure 2. Individual loss term versus epoch. Left: equation loss. Right: initial condition loss.

These results show most terms in equation loss contribute to increasing loss with epochs. We checked the initial condition loss and its individual loss terms. We found the initial condition loss and its individual losses decrease with epochs. These results indicate that the model fits well with the initial condition loss terms. The possible reason for different behaviors in the equation and initial condition loss might be related to different scale values. We noticed that the values of equation loss and initial condition loss differ by one order of magnitude at the beginning epoch. This suggests that the two losses have different contributions to total loss during training and that leads to different magnitudes of the backpropagation gradient (Wang et al. 2021). Recently works of literature claim that the imbalance loss problem can be solved either by the learning rate annealing or the relative loss balancing with random lookback (Wang et al. 2021; Bischof & Kraus 2021). We showed the trained model output at t = 0.2 in Figure 3. and the results do not perfectly match the analysis results (Tsai et al. 2015) due to the poor training results.

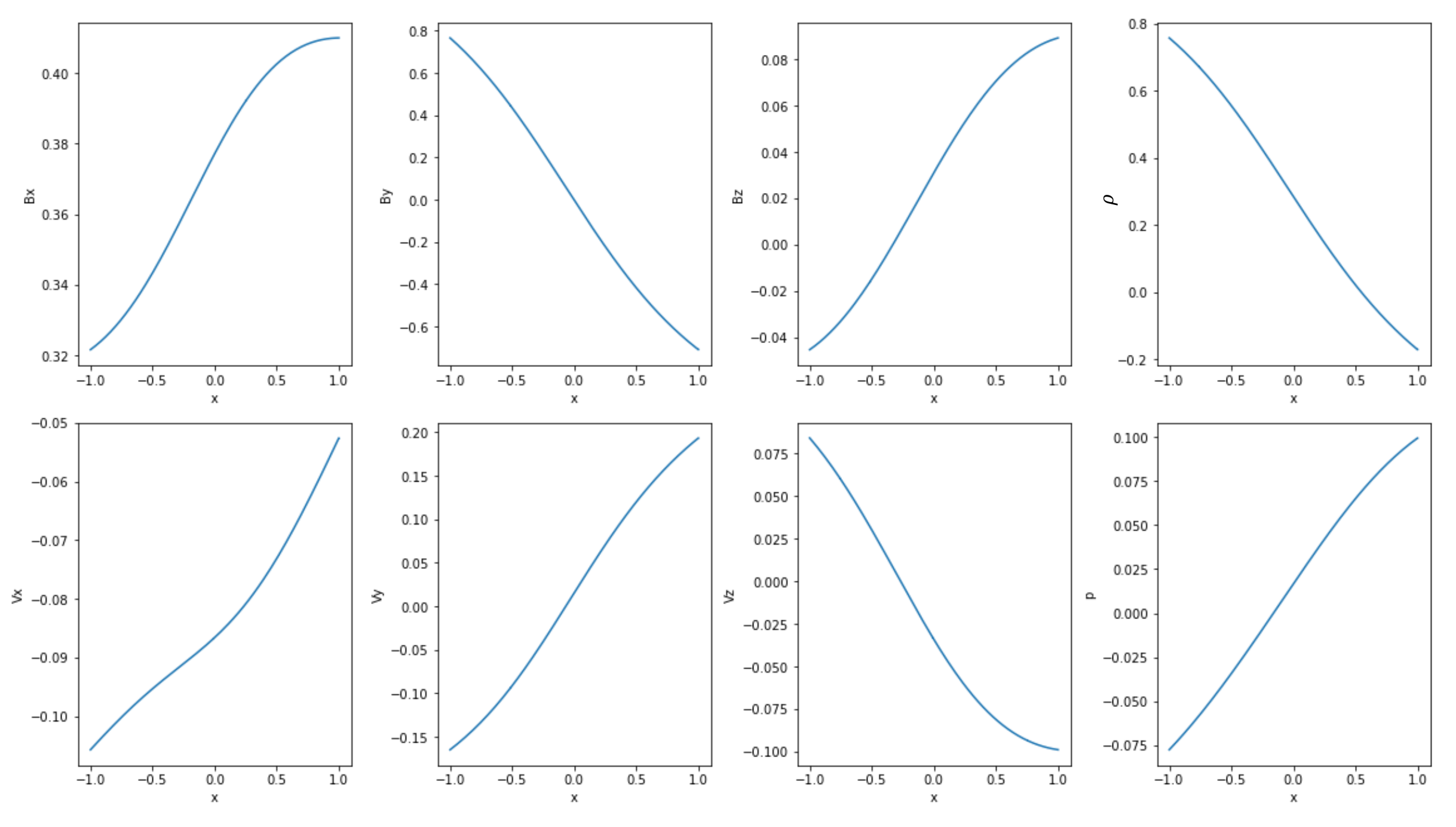

Figure 3. Trained model prediction for $B_x$, $B_y$, $B_z$, $\rho$, $V_x$, $V_y$, $V_z$, p at t = 0.2.

## Reference

- Raissi, M., Perdikaris, P., Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
- Tsai, T.C., Yu, H-S., Hsieh, M.-S., Lai, S.H., Yang, Y.-H. Implicit predictor–corrector central finite difference scheme for the equations of magnetohydrodynamic simulations. Computer Physics Communications. **196**, 1-12 (2015)
- Bard C. and Dorelli J. C. Neural Network Reconstruction of Plasma Space-Time. Front. Astron. Space Sci. **8**, 146-160 (2021)
- Wang, S., Teng , Y., Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. Siam J. Sci. Comput. **43**, A3055-A3081 (2021)
- R. Bischof and M. Kraus, Multi-objective loss balancing for physics-informed deep learning, arXiv preprint arXiv:21100.09813, (2021)