Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

AI in the Sciences and Engineering **Spring Semester 2024**

Student: Carla Judith López Zurita

Project 3

Due date: Wednesday 26 June 2024, 23:59

The main objective of the project is to apply the concepts learned in class related to Neural Differential Equations and Hybrid Workflows, as well as to a bonus quastion regarding Transfer Learning.

1. Inverted pendulum

The objective of this task is to control the inverted pendulum problem using a Neural Network. The inverted pendulum is a classic problem in control theory, where the goal is to balance a pendulum in an upright position by applying an external force F to the cart that holds the pendulum. The system is described by the following differential equation:

$$(M+m)\ddot{x} + ml\ddot{\theta}\cos(\theta) - ml\dot{\theta}^2\sin(\theta) = F \tag{1}$$

$$l\ddot{\theta} + g\sin(\theta) + \ddot{x}\cos(\theta) = 0 \tag{2}$$

where θ is the angle of the pendulum with respect to the vertical, $g = 9.81$ m/s^2 is the acceleration due to gravity, $l = 1.0$ m is the length of the pendulum, $m = 0.1$ kg is the mass of the pendulum, $M = 1.0$ kg is the mass of the cart and F is the force applied to the cart in Newtons.

Solving the coupled ODE system

First, we need to solve the coupled ODE system to simulate the dynamics of the inverted pendulum. The idea is to use an autodifferentiation library to code the solver in order to use it later to train a Neural Network to control the pendulum. In this case, I used torch due to its familiarity to me. I used Runge-Kutta 4th order method to solve the system. To test the implementations, I simulated the system with a sinusoidal force $F(t) = 10\sin(t)$. The results are shown in Figure 1.

Learning to balance the pendulum

Next, we will use a Neural Network to learn to balance the pendulum. The input to the network will be just the time t and the output will be the force F applied to the cart. We will train the network using the whole trajectory of the pendulum. The network will be trained using the Adam optimizer and a custom loss function that penalizes the deviation of the pendulum from the vertical position and large angular velocities. The network is a simple feedforward network with 1 hidden layer with 32 neurons and SiLU activation function. The output layer has a single neuron with linear activation function. The network is trained for 500 epochs with an initial learning rate of 10^{-2} and a scheduler that reduces the learning rate by a factor of 0.5 every 75 epochs. The plot of the loss function during training is shown in Figure 2.

The results of the simulation using the Neural Network to control the force applied to the cart are shown in Figure 3.

Attatched to the report you will find the code used to solve the ODE system and train the Neural Network, as well as the simulation results in a gif format.

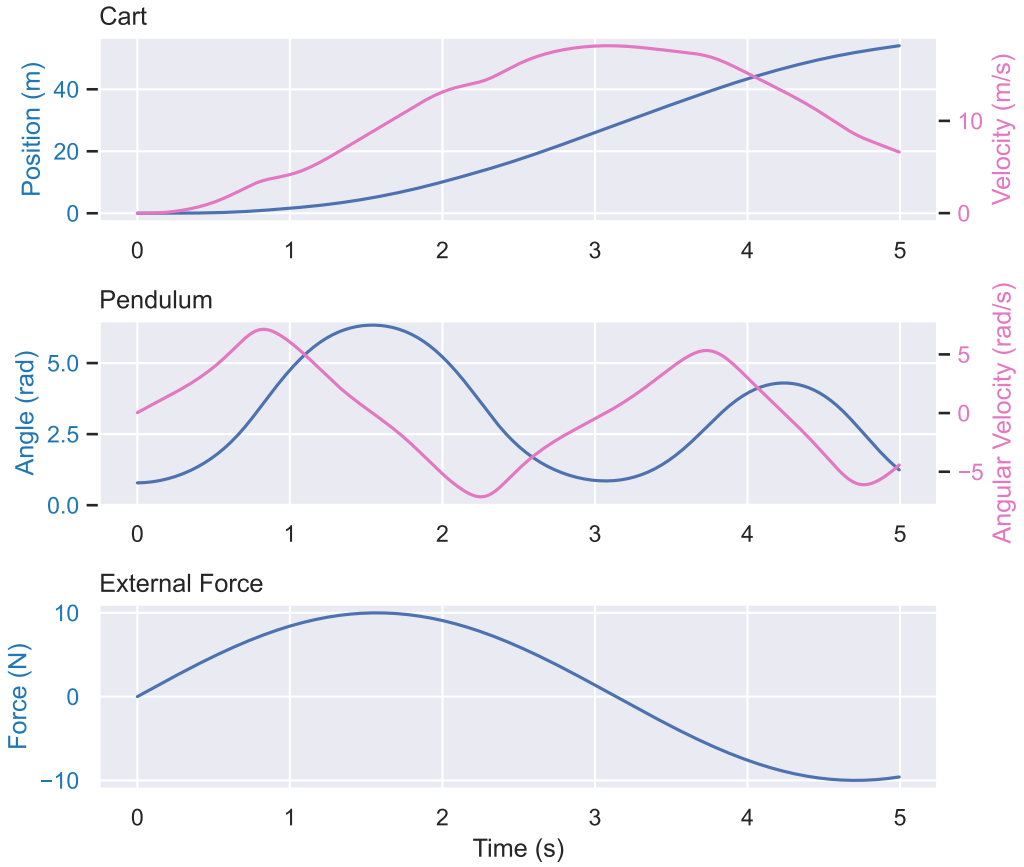

Figure 1: Simulation of the inverted pendulum with a sinusoidal force.

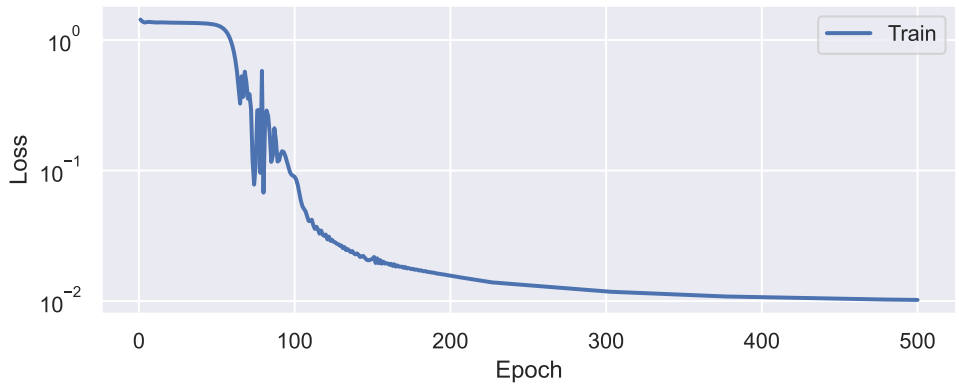

Figure 2: Loss function during training of the Neural Network.

2. Model discovery

The objective of this task is to write our own PDE-FIND [1] algorithm to discover the differential equation that governs the dynamics of the some unkown system using only provided data files.

Based on the assumption that the dynamics of the system can be described by a partial differential equation of the form,

$$u_t = F(u, u_x, u_{xx}, u_y, u_{yy}, u_{xy}, \ldots, x, y, \ldots, t), \tag{3}$$

we can create a library of possible functions that can appear in the right hand side of the equation,

$$\Theta(\mathbf{u}) = [\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \mathbf{u}_y, \mathbf{u}_{yy}, \mathbf{u}_{xy}, \mathbf{u}\mathbf{u}_x, \mathbf{u}\mathbf{u}_y, \ldots]. \tag{4}$$

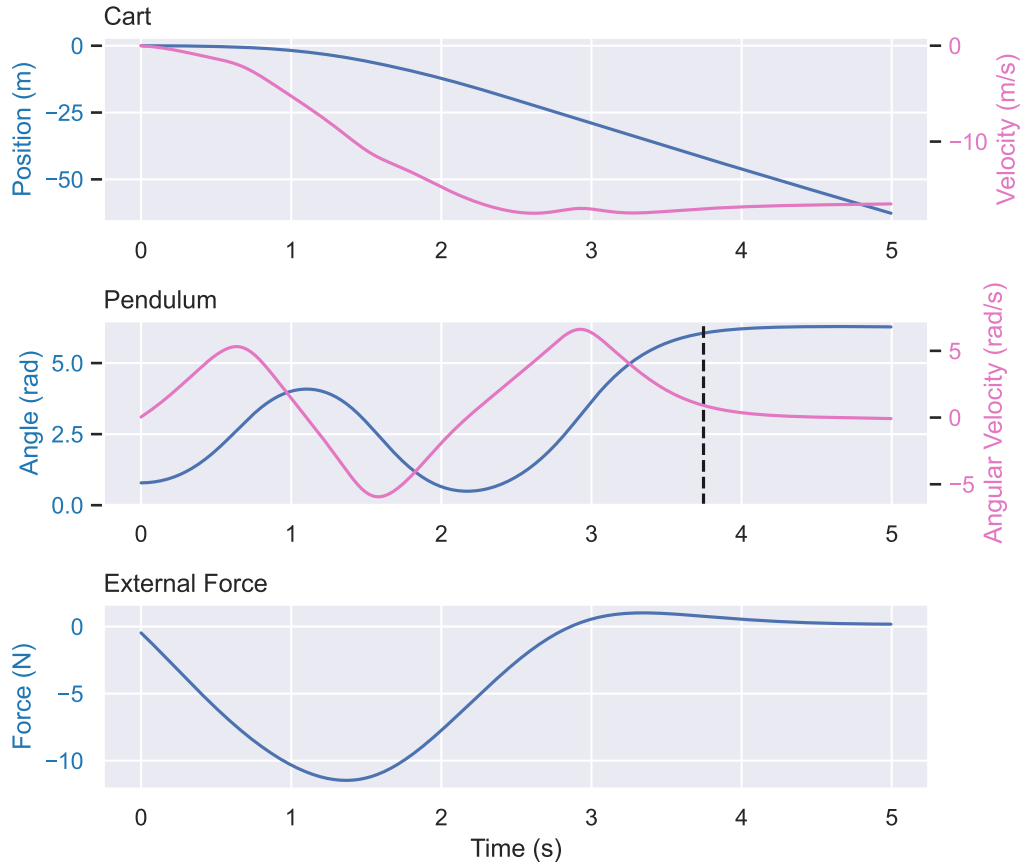

Figure 3: Simulation of the inverted pendulum using a Neural Network to control the force applied to the cart.

We then use the given data to find the coefficients of the functions that best describe the dynamics of the system,

$$\mathbf{u}_t = \Theta(\mathbf{u})\xi, \tag{5}$$

where ξ are the coefficients of the functions in the library. We are given three files with data of differents system for which we want to find the PDE that governs the dynamics.

Implementation of PDE-FIND

My implementation of the PDE-FIND consists on using finite difference scheme to compute partial derivatives of the data, building the library of possible functions and using Ridge regression to find the coefficients or weights.

The finite difference scheme is taken directly from the `pysindy` library [1]. It computes the centered differences of the data to approximate the partial derivatives, with consideration of the boundary conditions. I used it recursively to compute the second and third order mixed derivatives. Next, to build the library of possible functions, I included linear and non-linear combinations of the computed derivatives up to the third order, without including the time and space coordinates as possible functions. We also exclude mixed derivatives with that include time. I added a parameter to control the maximum number of terms in the library. Finally, I used Ridge regression to find the coefficients of the functions as in Equation (5). To ensure sparsity, I added a filter to remove coefficients that are below a certain threshold, which I set to 10^{-2}.

File 1

The first file is a 1+1D dimensional system u, with x and t coordinates. The algorithm requires the user to specify the dependent and independent variables and the maximum order of the derivatives to compute, in our case 3. In the file we find the dependent variable u and the independent variables x and t. Therefore, the derivatives we need are $\mathbf{u}_x$, $\mathbf{u}_{xx}$, $\mathbf{u}_{xxx}$. We can then build the library of

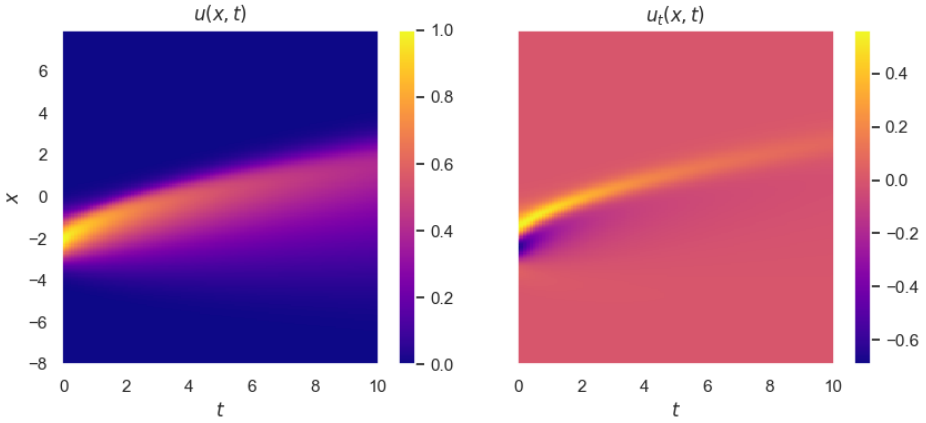

Figure 4: Data of the first file, with its temporal derivative.

possible functions as in Equation (4), which includes the following functions:

$$\Theta(\mathbf{u}) = [\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \mathbf{u}_{xxx}, \mathbf{uu}, \mathbf{uu}_x, \mathbf{uu}_{xx}, \mathbf{uu}_{xxx}, \mathbf{u}_x\mathbf{u}_x, \mathbf{u}_x\mathbf{u}_{xx}, \mathbf{u}_x\mathbf{u}_{xxx}, \mathbf{u}_{xx}\mathbf{u}_{xx}, \mathbf{u}_{xx}\mathbf{u}_{xxx}, \mathbf{u}_{xxx}\mathbf{u}_{xxx}, \mathbf{uuu}]. \tag{6}$$

The results of the PDE-FIND algorithm are

$$\mathbf{u}_t = -1.0\mathbf{u}\mathbf{u}_x + 0.1\mathbf{u}_{xx}. \tag{7}$$

From the results, we can discover that the system is governed by the Burgers equation.

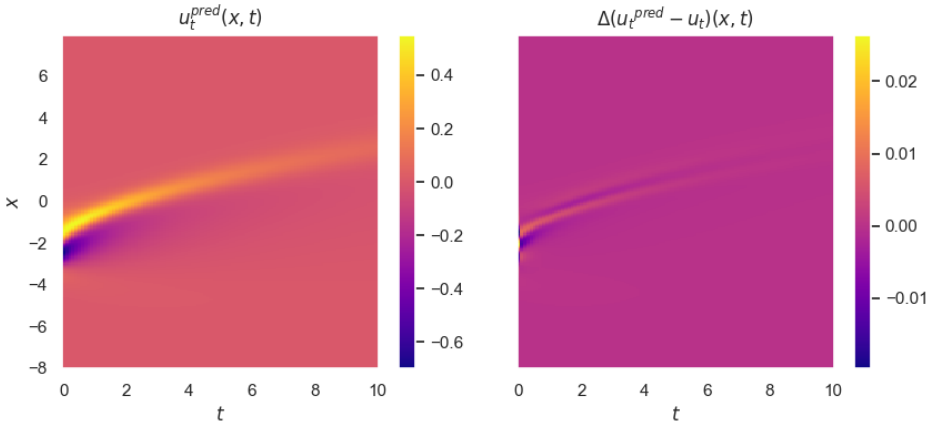

Figure 5: Prediction of the first file using the discovered PDE.

File 2

The second file is also a 1+1D dimensional system $u(t, x)$. The same derivatives and terms are computed as in the previous case, shown in Equation (6). The results of the PDE-FIND algorithm are

$$\mathbf{u}_t = -5.4\mathbf{u}\mathbf{u}_x - 0.9\mathbf{u}_{xxx} - 0.11\mathbf{u}_x\mathbf{u}_{xx} - 0.081\mathbf{u}_x. \tag{8}$$

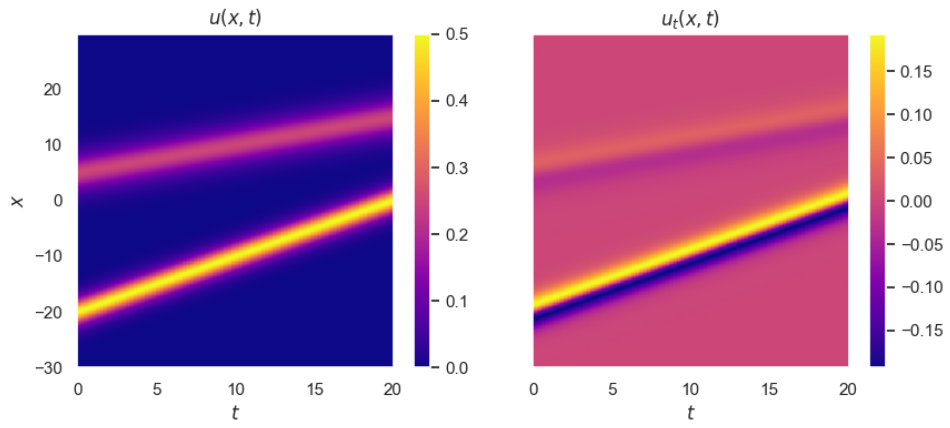

Figure 6: Data of the second file, with its temporal derivative.

From the results, we can discover that the system is governed by the Korteweg-de Vries equation if we ignore the terms $\mathbf{u}_x\mathbf{u}_{xx}$ and $\mathbf{u}_x$. Since this file is a bit more complex, the algorithm is not able to find the exact equation. Perhaps using a finer grid could help to improve the results due to the higher order derivatives.

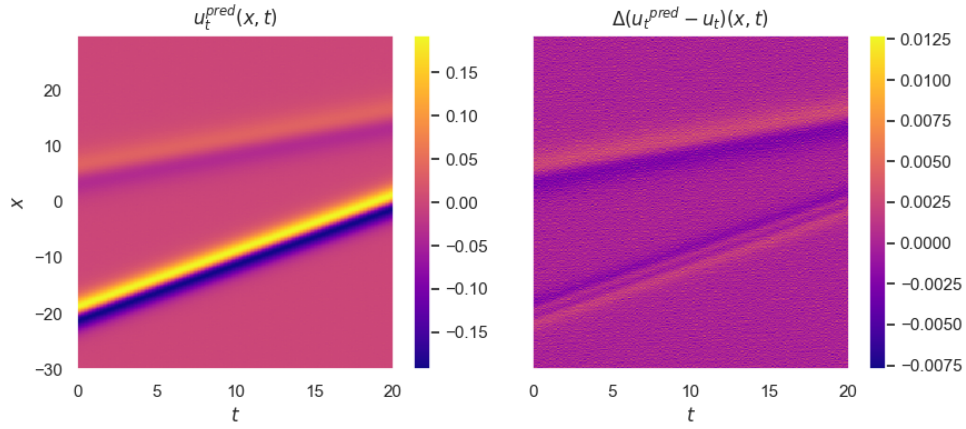

Figure 7: Prediction of the second file using the discovered PDE.

File 3

The third file is a 2+1D dimensional system, with x, y and t coordinates. Where the solution is a vector field with two components, i.e. $u(x,t)$ and $v(x,t)$. In this case the PDE is a set of two coupled equations of the form

$$\mathbf{u}_t = D_1(\mathbf{u}, \mathbf{u}_x, \mathbf{v}, \mathbf{v}_x, \mathbf{uv}, \ldots) \tag{9}$$
$$\mathbf{v}_t = D_2(\mathbf{u}, \mathbf{u}_x, \mathbf{v}, \mathbf{v}_x, \mathbf{uv}, \ldots). \tag{10}$$

This time, the library of possible functions includes the 52 terms, as shown in the Appendix A. The results of the PDE-FIND algorithm are

$$\mathbf{u}_t = 0.85\mathbf{v}^3 + 0.85\mathbf{u}^2\mathbf{v} + 0.62\mathbf{u} - 0.6\mathbf{u}^3 - 0.6\mathbf{u}\mathbf{v}^2 + 0.13\mathbf{v} + 0.085\mathbf{u}_{xx} + 0.081\mathbf{u}_{yy}, \tag{11}$$
$$\mathbf{v}_t = -0.85\mathbf{u}^3 - 0.85\mathbf{u}\mathbf{v}^2 + 0.62\mathbf{v} - 0.6\mathbf{v}^3 - 0.6\mathbf{u}^2\mathbf{v} - 0.13\mathbf{u} + 0.085\mathbf{v}_{yy} + 0.081\mathbf{v}_{xx}. \tag{12}$$

From the results, we can discover that the system is a reaction-diffusion system. We can see the terms match between the two equations. We can infer that the system is a reaction-diffusion system

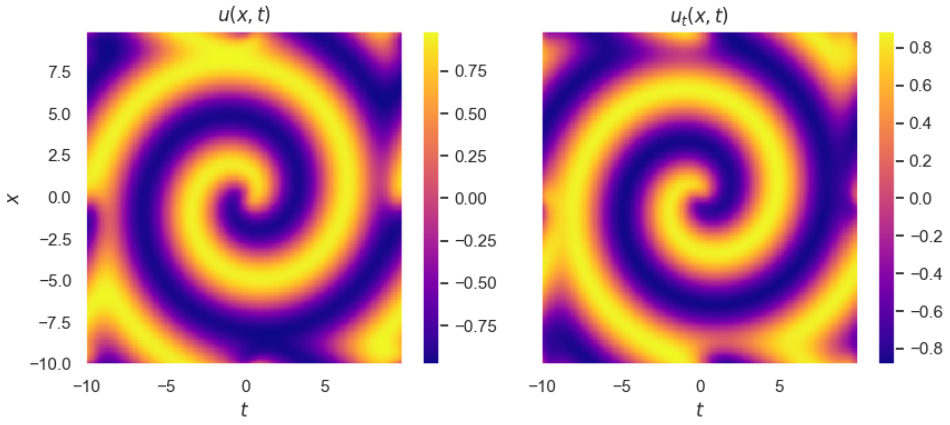

Figure 8: Data of the third file, with its temporal derivative for the first variable.

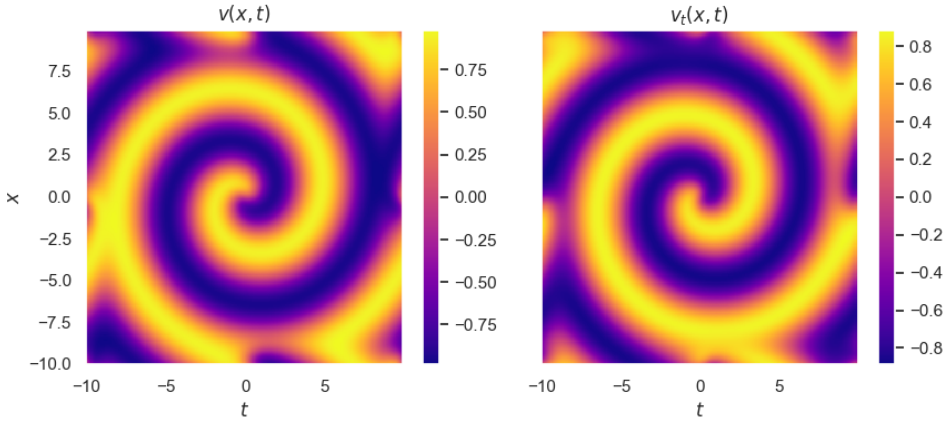

Figure 9: Data of the third file, with its temporal derivative for the second variable.

with a non-linear reaction term and a diffusion term:

$$\mathbf{u}_t = 0.1\nabla^2\mathbf{u} + (1 - A^2)\mathbf{u} + \beta A^2\mathbf{v}, \tag{13}$$

$$\mathbf{v}_t = 0.1\nabla^2\mathbf{v} - \beta A^2\mathbf{u} + (1 - A^2)\mathbf{v}, \tag{14}$$

$$A^2 = \mathbf{u}^2 + \mathbf{v}^2. \tag{15}$$

but the algorithm is not able to find the exact equation, with some extra terms (e.g. $0.13v$ and $0.62u$) that could be due to the noise in the data.

A. Library of possible functions for File 3

The library of possible functions for the third file is shown below.

- $\mathbf{u}$
- $\mathbf{v}$
- $\mathbf{u}_x$
- $\mathbf{u}_{xx}$
- $\mathbf{u}_{xxx}$
- $\mathbf{u}_{xxy}$
- $\mathbf{u}_{xy}$
- $\mathbf{u}_{xyx}$
- $\mathbf{u}_{xyy}$
- $\mathbf{u}_y$
- $\mathbf{u}_{yy}$
- $\mathbf{u}_{yyx}$
- $\mathbf{u}_{yyy}$
- $\mathbf{v}_x$
- $\mathbf{v}_{xx}$
- $\mathbf{v}_{xxx}$
- $\mathbf{v}_{xxy}$
- $\mathbf{v}_{xy}$
- $\mathbf{v}_{xyx}$
- $\mathbf{v}_{xyy}$
- $\mathbf{v}_y$
- $\mathbf{v}_{yy}$
- $\mathbf{v}_{yyx}$
- $\mathbf{v}_{yyy}$
- $\mathbf{uu}$
- $\mathbf{vv}$
- $\mathbf{u}_x\mathbf{u}_x$
- $\mathbf{u}_{xx}\mathbf{u}_{xx}$
- $\mathbf{u}_{xxx}\mathbf{u}_{xxx}$
- $\mathbf{u}_{xxy}\mathbf{u}_{xxy}$
- $\mathbf{u}_{xy}\mathbf{u}_{xy}$
- $\mathbf{u}_{xyx}\mathbf{u}_{xyx}$
- $\mathbf{u}_{xyy}\mathbf{u}_{xyy}$
- $\mathbf{u}_y\mathbf{u}_y$
- $\mathbf{u}_{yy}\mathbf{u}_{yy}$
- $\mathbf{u}_{yyx}\mathbf{u}_{yyx}$
- $\mathbf{u}_{yyy}\mathbf{u}_{yyy}$
- $\mathbf{v}_x\mathbf{v}_x$
- $\mathbf{v}_{xx}\mathbf{v}_{xx}$
- $\mathbf{v}_{xxx}\mathbf{v}_{xxx}$
- $\mathbf{v}_{xxy}\mathbf{v}_{xxy}$
- $\mathbf{v}_{xy}\mathbf{v}_{xy}$
- $\mathbf{v}_{xyx}\mathbf{v}_{xyx}$
- $\mathbf{v}_{xyy}\mathbf{v}_{xyy}$
- $\mathbf{v}_y\mathbf{v}_y$

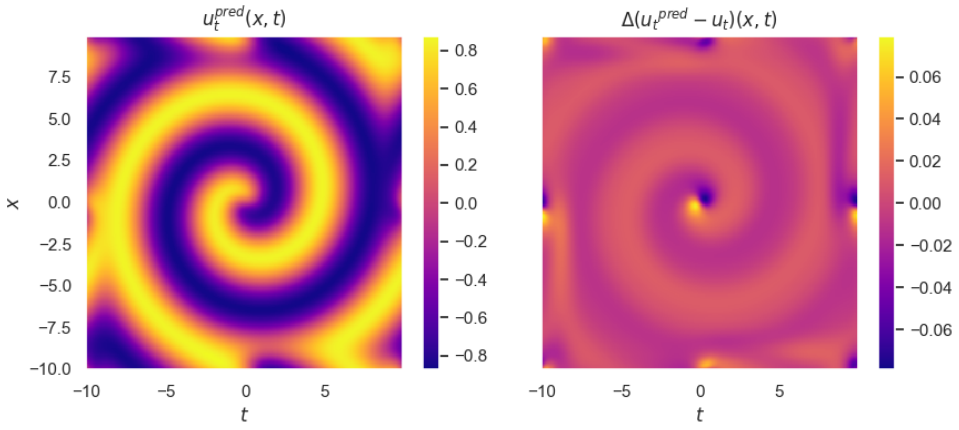

Figure 10: Prediction of the third file for the first variable using the discovered PDE.

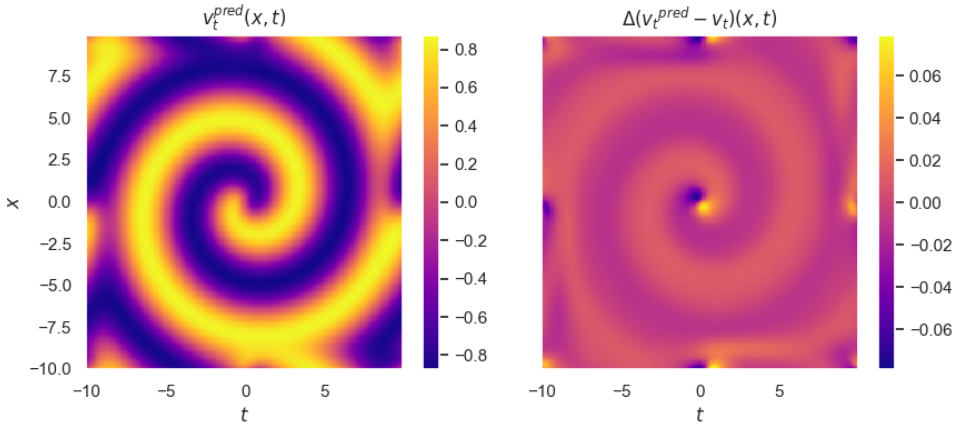

Figure 11: Prediction of the third file for the second variable using the discovered PDE.

- $\mathbf{v}_{yy}\mathbf{v}_{yy}$
- $\mathbf{v}_{yyx}\mathbf{v}_{yyx}$
- $\mathbf{v}_{yyy}\mathbf{v}_{yyy}$
- **uuu**
- **uuv**
- **uvv**
- **vvv**

References

[1] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.