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 Model Discovery. The project is divided into two tasks.

1. Inverted pendulum

The objective of the first task is to control the inverted pendulum problem using a Neural Network. This is to be achieved by applying an external force F to the cart that holds the pendulum, which is free to move horizontally. 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 and simulate the dynamics of the inverted pendulum. The idea is to use an autodifferentiation library to code the solver in order to be able to use it in the Neural Network training process. For my implementation, I chose `pytorch` due to its familiarity to me. As integration scheme, I used Runge-Kutta 4th order method. To test the code, I simulated the system with a sinusoidal force $F(t) = 10\sin(t)$. The results are shown in Figure 3 in the Appendix A.

Learning to balance the pendulum

Next, we will implement the Neural Network trainer. The input to the network is time t and the output is the force F applied to the cart. The network is trained using the whole trajectory of the pendulum, with 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 1.

The results of the simulation using the Neural Network to control the force applied to the cart are shown in Figure 2. 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.

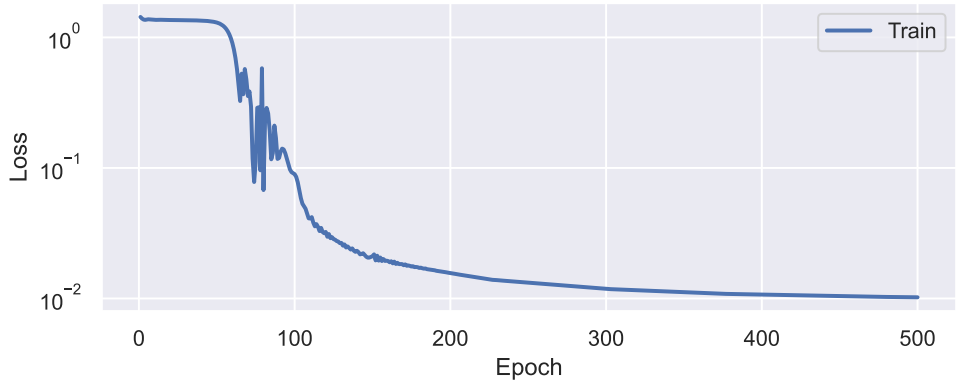

Figure 1: Loss function during training of the Neural Network.

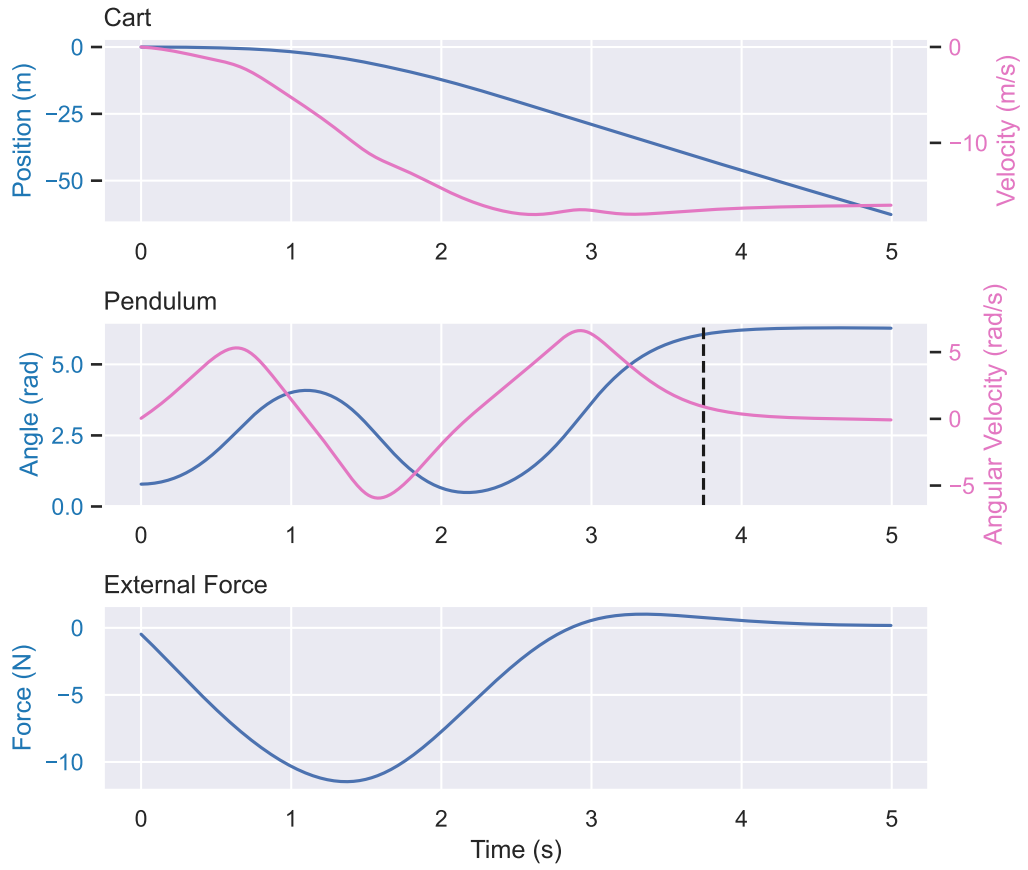

Figure 2: Simulation of the inverted pendulum using a Neural Network to control the force applied to the cart.

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,

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

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

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

For the task, we are given three files with data of differents system in order of increasing complexity.

Implementation of PDE-FIND

My implementation of the PDE-FIND consists on using finite difference scheme, taken directly from the `pysindy` library, to compute the partial derivatives. [1]. It uses centered differences, with consideration of the boundary conditions. I used it recursively to compute the second and third order mixed derivatives. To build the library of possible functions, I included linear and non-linear combinations of the computed derivatives, excluding the time and space coordinates as possible functions. I also excluded derivatives that include time. Finally, I used LassoCV from the `scipy` library to find the coefficients of the functions as in Equation (5). Finally, coefficients that are below a certain threshold (10^{-2}) are removed.

File 1

The first file is a 1+1D dimensional system with dependent variable u, and independent variables x and t as coordinates. We can then build the library of possible functions as in Equation (4), using the partial derivatives in x up to third order. The library of possible functions is shown in Equation (6), which includes 15 terms.

$$\begin{aligned}\Theta(\mathbf{u}) = [&\mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}, \mathbf{u}_{xxx}, \mathbf{u}\mathbf{u}, \mathbf{u}\mathbf{u}_x, \mathbf{u}\mathbf{u}_{xx}, \mathbf{u}\mathbf{u}_{xxx}, \mathbf{u}_x\mathbf{u}_x, \mathbf{u}_x \\ &\mathbf{u}_{xx}, \mathbf{u}_x\mathbf{u}_{xxx}, \mathbf{u}_{xx}\mathbf{u}_{xx}, \mathbf{u}_{xx}\mathbf{u}_{xxx}, \mathbf{u}_{xxx}\mathbf{u}_{xxx}, \mathbf{u}\mathbf{u}\mathbf{u}].\end{aligned} \tag{6}$$

The PDE-FIND algorithm was fitted using the `LassoCV` regression technique with following parameters

```
LassoCV(alphas=[1e-6, 1e-5], fit_intercept=False,
        cv=3, tol=1e-6, max_iter=int(1e8))
```

The results of the PDE-FIND algorithm are

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

which let us know that the system is governed by Burger's equation. Figure 4 in the Appendix B shows the prediction of the system using the discovered PDE and the difference between the prediction and the actual data. The prediction is very good, and has a mean squared error of 5.5729e-07. The discovered PDE is theoretically correct.

File 2

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

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

which imply that the system is governed by the Korteweg-de Vries equation, choosing to 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. Figure 5 in the Appendix B shows the numerical prediction of the system. The mean squared error is 1.4913e-06, which is higher than in the previous case. Perhaps using a finer grid could help to improve the results due to the higher order derivatives.

File 3

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

$$\begin{aligned} \mathbf{u}_t &= \mathcal{D}_1(\mathbf{u}, \mathbf{u}_x, \mathbf{v}, \mathbf{v}_x, \mathbf{uv}, \ldots), \\ \mathbf{v}_t &= \mathcal{D}_2(\mathbf{u}, \mathbf{u}_x, \mathbf{v}, \mathbf{v}_x, \mathbf{uv}, \ldots). \end{aligned} \tag{9}$$

This time, the library of possible functions includes the 52 terms, as shown in the Appendix C. To achieve fast computation, I decided to subsample the data along the temporal coordinate. I used similar parameters for the regression as in the previous cases, and solved for the two equations separately. The results of the PDE-FIND algorithm are

$$\mathbf{u}_t = 0.85\mathbf{v}^3 + 0.85\mathbf{u}^2\mathbf{v} + 0.62\mathbf{u} - 0.6\mathbf{u}^3 - 0.6\mathbf{u}\mathbf{v}^2 + 0.13\mathbf{v} + 0.085\mathbf{u}_{xx} + 0.081\mathbf{u}_{yy}, \tag{10}$$

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

We can infer that the system is a reaction-diffusion system,

$$\mathbf{u}_t = 0.1\nabla^2\mathbf{u} + (1 - A^2)\mathbf{u} + \beta A^2\mathbf{v}, \tag{12}$$

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

$$A^2 = \mathbf{u}^2 + \mathbf{v}^2. \tag{14}$$

The algorithm is not able to find the exact equation, and it gives coefficients for some extra terms ($0.13v$ and $0.62u$), and also not the exact coefficients for the rest of the terms. Figures 6 and 7 in the Appendix B show the prediction of the system. The prediction is not as good as in the previous cases, reaching a mean squared error of 7.1573e-05 for the first variable and 7.1707e-05 for the second, but it still captures the dynamics of the system and discovers the correct type of equation. A higher error could be due to the complexity of the system and the noise in the data. Higher convergence could be achieved by a lower subsampling rate, but at the cost of higher computational time.

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.

A. Inverted pendulum with a sinusoidal force

The results of the simulation of the inverted pendulum with a sinusoidal force are shown in Figure 3.

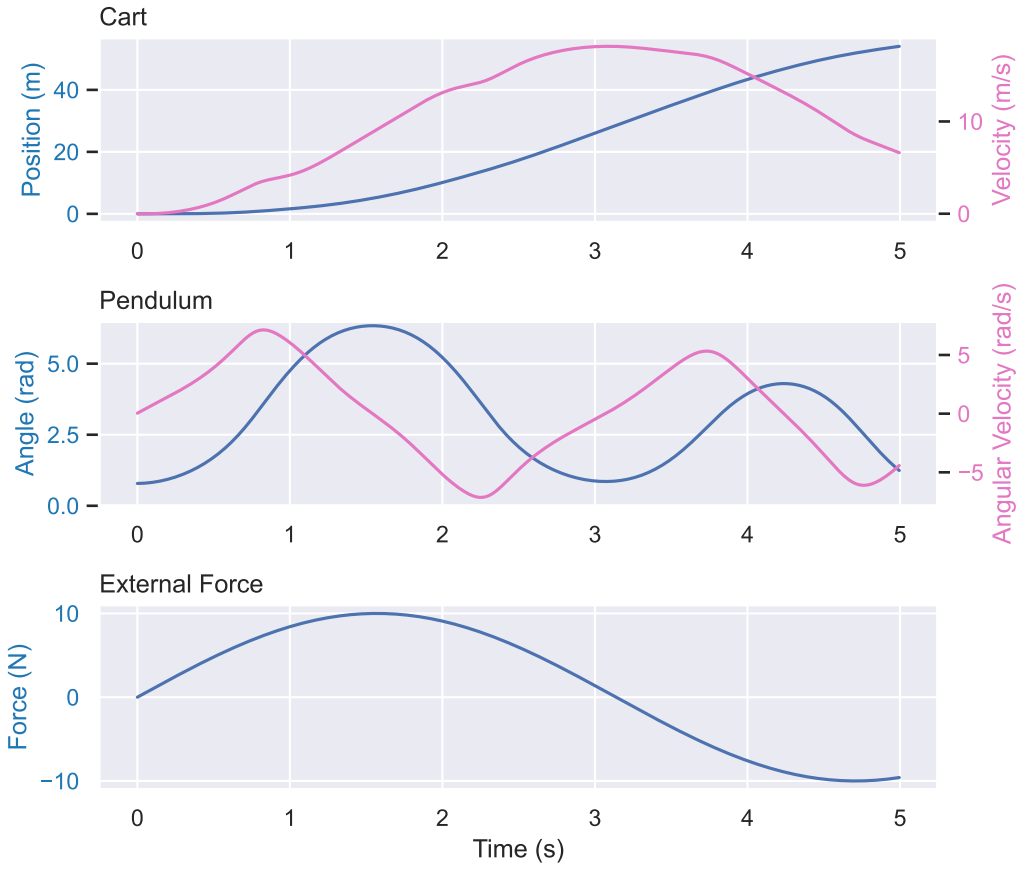

Figure 3: Simulation of the inverted pendulum with a sinusoidal force.

B. Numerical results of PDE-FIND

Attatched are the figures corresponding to the results of the PDE-FIND algorithm for all files.

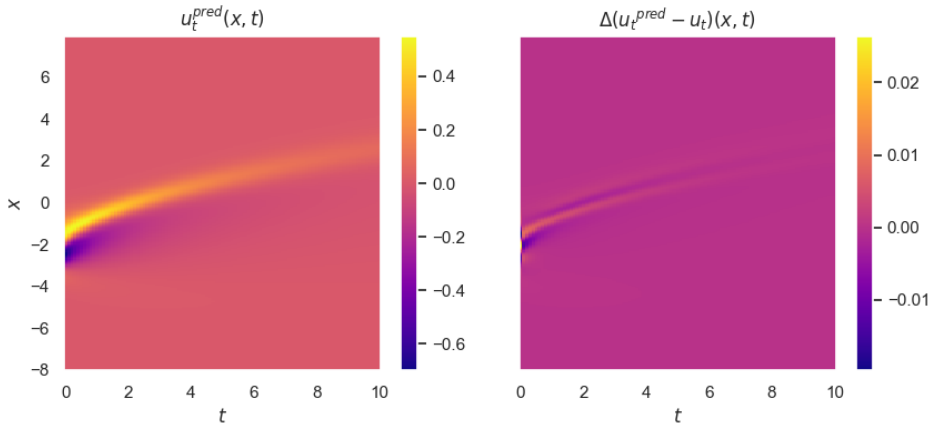

Figure 4: Prediction of the first file using the discovered PDE. MSE: 5.5729e-07.

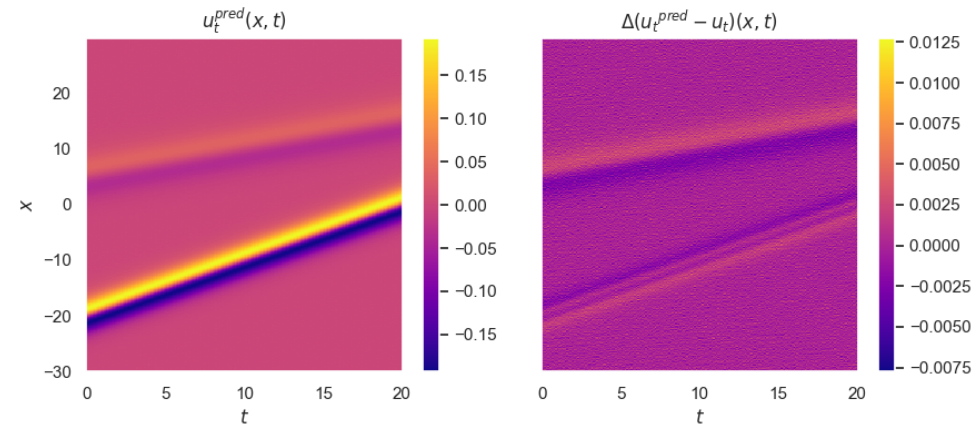

Figure 5: Prediction of the second file using the discovered PDE. MSE: 1.4913e-06

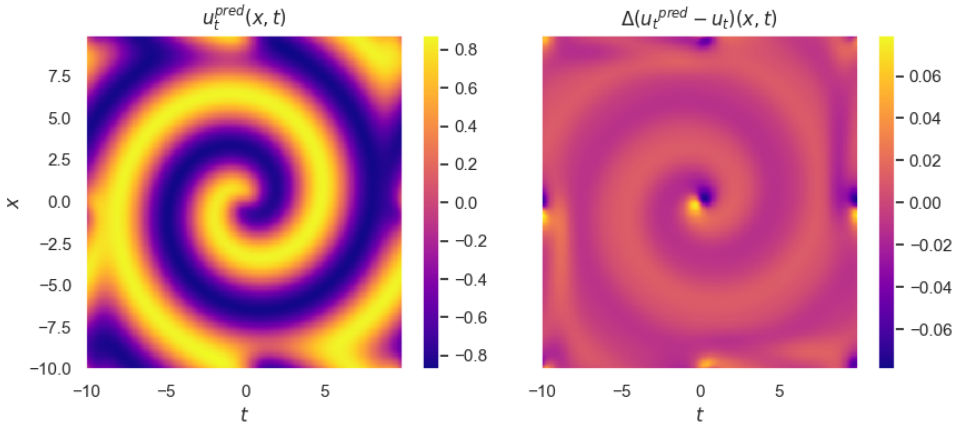

Figure 6: Prediction of the third file for u using the discovered PDE. MSE: 7.1573e-05.

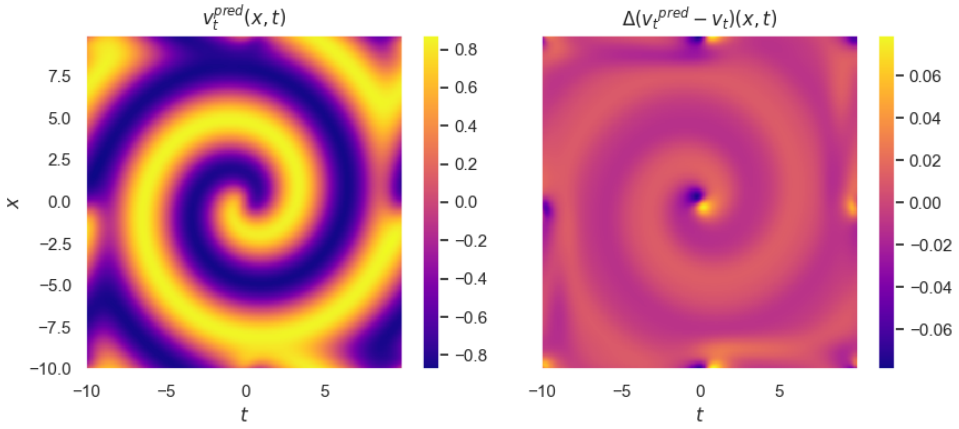

Figure 7: Prediction of the third file for the v using the discovered PDE. MSE: 7.1707e-05.

C. Library of possible functions for File 3

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

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

- $\mathbf{v}_{xxx}$
- $\mathbf{v}_{xxy}$
- $\mathbf{v}_{xy}$
- $\mathbf{v}_{xyx}$
- $\mathbf{v}_{xyy}$
- $\mathbf{v}_{y}$
- $\mathbf{v}_{yy}$
- $\mathbf{v}_{yyx}$
- $\mathbf{v}_{yyy}$
- $\mathbf{uu}$
- $\mathbf{vv}$
- $\mathbf{u}_{x}\mathbf{u}_{x}$
- $\mathbf{u}_{xx}\mathbf{u}_{xx}$
- $\mathbf{u}_{xxx}\mathbf{u}_{xxx}$
- $\mathbf{u}_{xxy}\mathbf{u}_{xxy}$
- $\mathbf{u}_{xy}\mathbf{u}_{xy}$
- $\mathbf{u}_{xyx}\mathbf{u}_{xyx}$
- $\mathbf{u}_{xyy}\mathbf{u}_{xyy}$
- $\mathbf{u}_{y}\mathbf{u}_{y}$
- $\mathbf{u}_{yy}\mathbf{u}_{yy}$
- $\mathbf{u}_{yyx}\mathbf{u}_{yyx}$
- $\mathbf{u}_{yyy}\mathbf{u}_{yyy}$
- $\mathbf{v}_{x}\mathbf{v}_{x}$
- $\mathbf{v}_{xx}\mathbf{v}_{xx}$
- $\mathbf{v}_{xxx}\mathbf{v}_{xxx}$
- $\mathbf{v}_{xxy}\mathbf{v}_{xxy}$
- $\mathbf{v}_{xy}\mathbf{v}_{xy}$
- $\mathbf{v}_{xyx}\mathbf{v}_{xyx}$
- $\mathbf{v}_{xyy}\mathbf{v}_{xyy}$
- $\mathbf{v}_{y}\mathbf{v}_{y}$
- $\mathbf{v}_{yy}\mathbf{v}_{yy}$
- $\mathbf{v}_{yyx}\mathbf{v}_{yyx}$
- $\mathbf{v}_{yyy}\mathbf{v}_{yyy}$
- $\mathbf{uuu}$
- $\mathbf{uuv}$
- $\mathbf{uvv}$
- $\mathbf{vvv}$