Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Paul Stefan Saegert

born in Heidelberg (Germany)

2022

On Data-Driven Discovery Of Symbolic Differential Equations From Unsuitable Coordinates Using SINDy-Autoencoders

This Bachelor Thesis has been carried out by Paul Stefan Saegert at the
Computer Vision and Learning Lab in Heidelberg
under the supervision of
Prof. Dr. rer. nat. Ullrich Köthe

Abstract

Machine-Learning-Methoden haben sich in vielen Bereichen der Wissenschaft, einschließlich der Physik, zu einem leistungsfähigen Werkzeug entwickelt. Die Methoden sind schnell und ausgefeilt genug, um eine Vielzahl von Aufgaben zu erlernen, doch oft auf Kosten der Interpretierbarkeit. Dieses Problem kann zum Teil durch den Einsatz symbolischer Regression überwunden werden, wie in [1] und [2] gezeigt, allerdings erfordert dies Daten in geeigneten Koordinaten. In meiner Arbeit bewerte und verbessere ich den von K. Champion et al. vorgeschlagenen *SINDy-Autoencoder* [3], der nicht nur symbolische Bewegungsgleichungen aus hochdimensionalen Daten lernen soll, sondern auch die Koordinaten, in denen die Gleichung am günstigsten formuliert würde. Ich führe eine Verifizierung und Replikation der vorgeschlagenen Methode durch, bevor ich Varianten entwerfe und evaluiere, die schließlich zu besseren und zuverlässigeren Ergebnissen führen.

Abstract

Machine Learning Methods have evolved to a powerful tool in many fields of science, including physics. While methods have become fast and sophisticated enough to learn a wide variety of tasks, they often come at the cost of interpretability. This problem can be partially overcome by making use of symbolic regression as shown in [1] and [2], but it requires data in suitable coordinates. In my work, I assess and improve the *SINDy-Autoencoder* proposed by K. Champion et al. [3], which is designed to learn not only symbolic equations of motion from high-dimensional data, but also the coordinates in which the equations would be most conveniently formulated. I conduct a verification and replication of the proposed method before creating and evaluating variants which eventually lead to better and more reliable results.

Contents

1 Introduction

The ability to distill complex physical behaviour into symbolic, algebraic equations has long been reserved for humans, and has lead to significant advancements in the history of our species and our understanding of our physical environment. As natural as this realization may seem to the physicist, it is not at all obvious how one would efficiently automate the process of symbolic model discovery. Especially since, apart from being able to formulate the relevant equations from a set of algebraic operations, describing a system often involves finding appropriate coordinates in which the equations become easily interpretable and generalizable.

Computer algorithms and machine learning methods dominate more and more increasingly complicated automation processes and thus, the question of what they can achieve in the field of physics becomes self-evident. Many research groups have used machine learning methods to improve physical simulations with success [4], [5]. However, the field of automated discovery of symbolic physical models currently includes a multitude of different approaches, many of which come with notable trade-offs.

My work focuses on finding, assessing and improving a promising approach and giving future researchers a point of reference in regards to the current state of the field and ideas for further improvements.

2 Related Work

Recent attempts at symbolic regression such as PySR [6] and AI-Feynman [7] represent equations as expression trees and learn through evolutionary algorithms or recursive application of a variety of analysis and regression methods. Despite the claimed successes of AI-Feynman, however, given its complexity, its source-code is poorly documented [8] and its application is slow and intransparent, rendering attempts of replication troublesome.

A different yet popular alternative to the use of expression trees is the regularized least-squares-based framework for sparse identification of nonlinear dynamics (SINDy) [1], which operates on an augmented dataset with candidate terms to find explicit ordinary differential equations. One attempt to use a SINDy derivative to find partial differential equations from noisy data in combination with Neural Networks found in DeepMoD [9] proved unreliable and highly dependent on the initialization of the algorithm.

However, one very interesting application of SINDy is found in SINDY-Autoencoders by K. Champion et al. [3] where the SINDy algorithm is combined with an Autoencoder Neural Network to simultaneously identify ordinary differential equations and suitable coordinates from data.

3 SINDy-Autoencoders

In this section, I give a brief summary of the SINDy-Autoencoder proposed by K. Champion et al. [3], the foundations of which will become important to understand in later sections.

The SINDy-Autoencoder (Figure 1) aims to symbolically find ordinary differential equations (ODEs) from data in unsuitable coordinates by simultaneously learning coefficients to predefined terms of the ODE, and the coordinates in which the equation is formulated. It consists of two major components: 1) The autoencoder, a Neural Network comprised of a symmetric pair of an encoder and a decoder using fully connected feed-forward layers and the sigmoid activation function, and 2) SINDy, which operates in the low-dimensional latent space of the autoencoder.

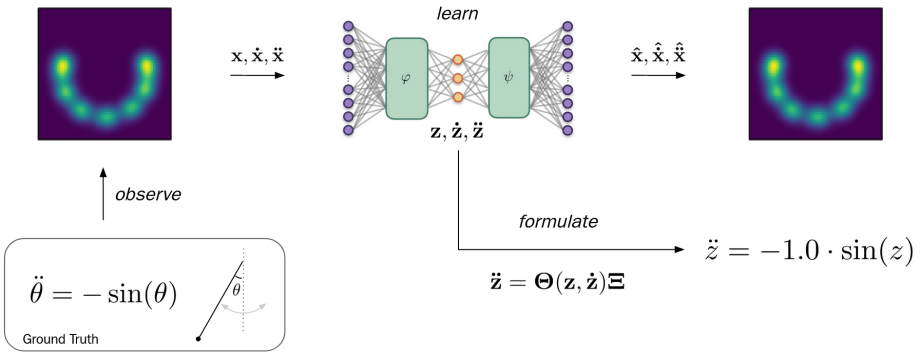


Figure 1: **The SINDy-Autoencoder.** The ground truth simulation is artificially embedded (*observed*) into a measurement space $\mathbf{x}$. The autoencoder learns a low-dimensional representation $\mathbf{z}$, in which the SINDy algorithm simultaneously formulates the governing equation. [3], edited.

In its most simple formulation, SINDy approximates an explicit ODE of the

form

$$\dot{\mathbf{z}}(t) = \mathbf{f}(\mathbf{z}(t)). \tag{1}$$

ODEs of order n may be discovered by making $\mathbf{f}$ dependent on lower order time derivatives up to order $n-1$. For the sake of clarity, I restrict this introduction to the first order case and only highlight the most important differences to the second order case.

The function $\mathbf{f}$ is modelled as a linear combination of nonlinear terms corresponding to polynomials of $\mathbf{z}$, lower-order time derivatives, and special, user-defined terms. This form of data augmentation reduces the problem to a regression of a coefficient-vector $\boldsymbol{\Xi} \in \mathbb{R}^{L \times D}$ to the library of predefined terms across all time steps $\boldsymbol{\Theta} \in \mathbb{R}^{T \times L}$ where L is the number of allowed terms, D is the dimensionality of $\mathbf{z}$ and T is the number of snapshots such that the equation of a snapshot t is given by the matrix multiplication

$$\dot{\mathbf{z}}_t = \boldsymbol{\Theta}(\mathbf{z})_t \boldsymbol{\Xi}. \tag{2}$$

A small but important part of SINDy is the *Sequential Thresholding* (ST) algorithm, which eliminates small and possibly interfering coefficients and thus their corresponding terms from the equation in regular, user-defined intervals during training. K. Champion et al. argue that ST promotes sparsity on the coefficients and claim the existence of convergence guarantees, referring to Zhang et al. [10].

The Autoencoder makes use of fully-connected feed-forward layers and the multiple differentiable Sigmoid activation function to facilitate easy propagation of the time derivatives through the layers of the networks (Section S1.3 in [3]).

At training time, the autoencoder is fed the high-dimensional data $\mathbf{x}$ and all precomputed time derivatives of $\mathbf{x}$ up to a user-defined order $n \in \{1, 2\}$, where $\mathbf{x}$ can be broadly interpreted as real-world measurements of a system. The encoder transforms the input into a low-dimensional representation of the measurement, $\mathbf{z} = \varphi(\mathbf{x})$, which should capture and condense the relevant features of the measurements. The time derivatives of $\mathbf{z}$ are transformed via the chain rule, where

$$\dot{\mathbf{z}} = (\nabla_{\mathbf{x}} \mathbf{z}) \dot{\mathbf{x}} \tag{3}$$

and

$$\ddot{z}_i = \sum_{k,l} \dot{x}_k \dot{x}_l (H_{\mathbf{z}_i})_{kl} + (\nabla_{\mathbf{x}} z_i)\ddot{\mathbf{x}} \tag{4}$$

where H is the hessian matrix defined as

$$(H_{\mathbf{z}_i})_{kl} = \frac{\partial^2 \mathbf{z}_i}{\partial \mathbf{x}_k \partial \mathbf{x}_l}. \tag{5}$$

In the latent space, the encoded measurements $\mathbf{z}$ and time derivatives of order $n-1$ are used to compute the values of all allowed terms at each time step, $\boldsymbol{\Theta}(\mathbf{z})$ (or $\boldsymbol{\Theta}(\mathbf{z}, \dot{\mathbf{z}})$ for $n = 2$). Then, the highest order time derivative in each snapshot is estimated with Equation 2.

Lastly, the encoded measurements $\mathbf{z}$ and the SINDy-estimate of the highest order time derivative, $\hat{\dot{\mathbf{z}}} = \boldsymbol{\Theta}(\mathbf{z})\boldsymbol{\Xi}$ (or $\hat{\ddot{\mathbf{z}}} = \boldsymbol{\Theta}(\mathbf{z}, \dot{\mathbf{z}})\boldsymbol{\Xi}$), are decoded again into their high-dimensional representation according to $\hat{\mathbf{x}} = \psi(\mathbf{z})$ and the chain rule of the decoder analogous to Equations 3 or 4.

The total loss is a weighted sum of four components, $\mathcal{L} = \mathcal{L}_{\mathbf{x}} + \lambda_{\dot{\mathbf{x}}}\mathcal{L}_{\dot{\mathbf{x}}} + \lambda_{\dot{\mathbf{z}}}\mathcal{L}_{\dot{\mathbf{z}}} + \lambda_1 ||\boldsymbol{\Xi}||_1$, where for the sake of simplicity in the case of a first order ODE

$$\mathcal{L}_{\mathbf{x}} = ||\mathbf{x} - \psi(\mathbf{z})||_2^2 \tag{6}$$

is the reconstruction term,

$$\mathcal{L}_{\dot{\mathbf{x}}} = ||\dot{\mathbf{x}} - (\nabla_{\mathbf{z}}\psi(\mathbf{z}))(\boldsymbol{\Theta}(\mathbf{z})\boldsymbol{\Xi})||_2^2 \tag{7}$$

represents the reconstruction term of the time-derivative in $\mathbf{x}$-space using the decoded SINDy prediction, and

$$\mathcal{L}_{\dot{\mathbf{z}}} = ||(\nabla_{\mathbf{x}}\mathbf{z})\dot{\mathbf{x}} - \boldsymbol{\Theta}(\mathbf{z})\boldsymbol{\Xi}||_2^2 \tag{8}$$

captures the loss of the SINDy-Dynamics in $\mathbf{z}$-space.

It is important to note again that this loss in combination with the Adam optimizer is used to *simultaneously* optimize the parameters of the encoder and decoder (i.e. the transformation function into suitable coordinates) *and* the coefficients of the nonlinear terms (i.e. the equation of motion in those coordinates).

K. Champion et al. argue that at the end of training, the coefficients are biased towards smaller absolute values due to the influence of the L_1 regulariza-

tion. Therefore, they extend training by an additional number of epochs without L_1 regularization (i.e. $\lambda_1 = 0$) which they call *Refinement*. After Refinement, the coefficients' bias would be eliminated.

For further details, I refer to the original work by K. Champion et al. [3].

4 Methodology

The use of a standardized Methodology for all experiments is crucial if one intends to draw substantiated conclusions from their results. In order to verify and replicate the original results by K. Champion et al. [3], I adopted the relevant parts of their methodology. For the proceeding research and development, I adjusted unconclusive parts and added new standards in the form of metrics for evaluation.

In the original work by K. Champion et al., the proposed method was benchmarked with with 3 simulated example systems [3], which I briefly recap in the following section. Detailed information about the training parameters for each system are given in Appendix A.

4.1 Example Systems

Non-Linear Pendulum The first system is a simulation of a non-linear 'physical' pendulum which is described by the second order differential equation

$$\ddot{z} = -\sin(z) \tag{9}$$

where z is the deflection of the pendulum. The pendulum is simulated over 10 s in time steps of 0.02 s using the `odeint` method of the python package `scipy` [11]. After the simulation, the state at each step is rendered into an image of size 51x51 by modelling the tip of the pendulum as a gaussian distribution (Figure 2).

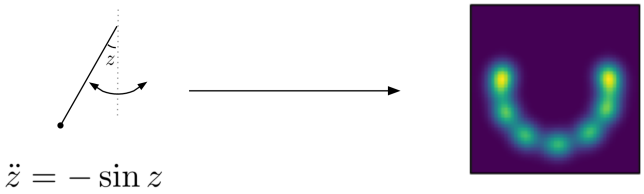


Figure 2: **Embedding of the Non-Linear Pendulum.** [3], edited.

A set of 100 fully rendered simulations with random initial conditions is combined to form one random training set. During training, a validation set generated from 10 random initial conditions is used.

The fully-connected autoencoder reduces the the 2601-dimensional input space to a 1-dimensional latent space. I test each model on one global, separate, precomputed test set of 100 initial conditions with high variance in $\mathbf{z}$.

Chaotic Lorenz System In the second system, the chaotic Lorenz System described by

$$\begin{aligned} \dot{z}_1 &= \sigma(z_2 - z_1) \\ \dot{z}_2 &= z_1(\rho - z_3) - z_2 \\ \dot{z}_3 &= z_1 z_2 - \beta z_3 \end{aligned} \tag{10}$$

with $\sigma = 10$, $\rho = 28$ and $\beta = 2.7$ is simulated over 5 s in time steps of 0.02 s (Figure 3).

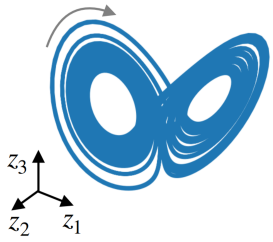


Figure 3: **Simulation of the Chaotic Lorenz System.** [3].

The simulation is artificially embedded into a 128-dimensional space by multiplying the coordinates with the values of the first 6 Legendre Polynomials at 128 uniform gridpoints $\mathbf{u}_1, ..., \mathbf{u}_6 \in \mathbb{R}^{128}$ between 0 and 1:

$$\mathbf{x}_t = \mathbf{u}_1 z_{t,1} + \mathbf{u}_2 z_{t,2} + \mathbf{u}_3 z_{t,3} + \mathbf{u}_4 z_{t,1}^3 + \mathbf{u}_5 z_{t,2}^3 + \mathbf{u}_6 z_{t,3}^3. \tag{11}$$

I followed the decision of K. Champion et al. to add normally distributed noise scaled by 10^{-6} to $\mathbf{x}$ [3]. The training and validation sets consist of simulations with 1024 and 20 random initial conditions. The autoencoder reduces the 128-dimensional input space to a 3-dimensional output space. Testing is performed on a fixed, precomputed test set of 100 initial conditions with high variance in $\mathbf{z}$ and the same additive noise as in the training set.

Reaction Diffusion System The third example system is described by a system of *partial* differential equations,

$$\begin{aligned} u_t &= (1-(u^2+v^2))u + \beta(u^2+v^2)v + d_1(u_{xx}+u_{yy}) \\ v_t &= -\beta(u^2+v^2)u + (1-(u^2+v^2))v + d_2(v_{xx}+v_{yy}), \end{aligned} \tag{12}$$

where $d_{1,2} = 0.1$ and $\beta = 1$. The system was only simulated once on a 100x100 grid with initial conditions corresponding to a spiral wave that rotates around its center (Figure 4).

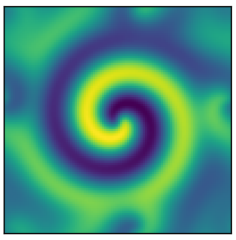


Figure 4: **Snapshot of the Reaction-Diffusion Simulation.** [3].

It ran over 500 s in steps of 0.05 s and was calculated by a Matlab script provided by K. Champion et al. over several hours. As input to the method, only u in form of a sequence of images is used and multiplied with a squared exponential mask positioned above at the center of the spiral to suppress boundary effects [12]. The training and validation sets capture 7999 and 1000 steps of the simulation. Normally distributed noise scaled by 10^{-6} is also added to the training set here.

The autoencoder reduces the 10 000-dimensional input space to a 2-dimensional latent space. Due to the design of their method, K. Champion et al. did not expect to find Equation 12.1 and therefore chose a 2-dimensional latent space in which SINDy is to learn the emergent behaviour of the spatial modes in the simulation [3].

Testing is performed on a separate subset of the simulation containing 1000 simulation steps.

4.2 Evaluation

Dynamics Error K. Champion et al. claim to use the fraction of variance unexplained (FVU) to assess the accuracy of the learned differential equation at each simulation step [3]. However, in the original codebase [13], they use the

following metric:

$$\mathrm{FVU}_y^0 = \frac{\frac{1}{N}\sum_i (y_i - \hat{y}_i)^2}{\frac{1}{N}\sum_i (y_i)^2}, \tag{13}$$

which differs from the more general definition of the FVU,

$$\mathrm{FVU}_y = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{14}$$

by the term $\bar{y}$ in the denominator. Although $\bar{y}$ is relatively small due to symmetries in the simulations, it is not guaranteed to be 0 or small enough to be neglected, expecially since the training sets are randomly generated. Therefore, I evaluate all tests and experiments using the more general Equation 14 to account for the possibility of uncentered data.

Furthermore, since the results are highly dispersed, the original method and variants thereof are tested by repeating the training 10 times with different random initializations of the model and different random training sets. Despite stating this in their work, K. Champion et al. report only on the two best of the resulting models in detail [3], whereas I compute the mean and standard deviation of the FVU in $\mathbf{x}$ $\dot{\mathbf{x}}$ and $\dot{\mathbf{z}}$, over all 10 repititions.

After training, all models are saved for evaluation and later use.

In the case of the Chaotic Lorenz System, K. Champion et al. distinguish between an in-distribution test set where the initial conditions are sampled from the same space as in training, and out-of-distribution, where the space of possible initial conditions in each dimension was increased by 50% over the space used for initial conditions during training [3]. The out-of-distribution error can be interpreted as a measure of how good the models generalize from the local dynamics.

Resimulation Error A further evaluation method - one that was not covered by K. Champion et al. - is the resimulation of the ground truth measurements. Resimulation is performed by simulating the system inside the latent space based on the encoded initial conditions and the learned differential equation. This simulation is then decoded into the high-dimensional measurement-space and compared to the original simulation. For comparison between the two, I use the cosine similarity,

$$D_C(\mathbf{x}_t, \hat{\mathbf{x}}_t) = \frac{\langle \mathbf{x}_t, \hat{\mathbf{x}}_t \rangle}{||\mathbf{x}_t|| \cdot ||\hat{\mathbf{x}}_t||}, \tag{15}$$

between the ground truth $\mathbf{x}_t$ and the resimulation $\hat{\mathbf{x}}_t$, which ignores relative scaling between the two. If $\mathbf{x}_t$ and $\hat{\mathbf{x}}_t$ show similar features, $D_C(\mathbf{x}_t, \hat{\mathbf{x}}_t)$ is approximately 1. If $\mathbf{x}_t$ and $\hat{\mathbf{x}}_t$ are orthogonal to each other, i.e. share no features, $D_C(\mathbf{x}_t, \hat{\mathbf{x}}_t)$ is 0. If $\mathbf{x}_t$ and $\hat{\mathbf{x}}_t$ show opposite features, $D_C(\mathbf{x}_t, \hat{\mathbf{x}}_t)$ is approximately -1, which, for example, could indicate that an oscillating resimulation is out of phase by 180° with respect to the original simulation. Thus, the *Resimulation Error* over time, $1 - D_C$, ranging from 0 to 2, gives an estimate of the forecast horizon of the method.

For the Non-Linear Pendulum and the Chaotic Lorenz System, I resimulate the dynamics over twice the duration of the simulations used for training and testing. Only for the Reaction-Diffusion System, the resimulation was limited to the length of the test simulation due to its enormously expensive computation.

This form of resimulation only computes the error in measurement space at fixed, absolute time intervals and does not take into account small shifts or distortions in time, as is intended. However, since two simulation frames shifted by only a small amount in time may look very different in measurement space especially in chaotic systems, the resulting errors have to be interpreted with care as this effect could make the resimulation look worse than it actually is, depending on where one puts their emphasis and the decision on how to evaluate this time-like error. Additionally, slight differences in the speed of the resimulation may accumulate, generally leading to larger errors the longer the resimulation runs.

Interpretability Measuring the quality and interpretability of the resulting differential equations is hard to quantify and may be subjective, varying from person to person. However, there are objectively desirable properties which allows the assessment of the discovered equations. In particular, I agree with the idea of K. Champion et al. that fewer terms are more desirable than a lot of terms [3]. Also, one may use similarity measures to compare the resulting equations to the ground truth equation. However, due to the freedom of choice of coordinates, the resulting equations may look very different from the ground truth while still describing the system to sufficient accuracy and one must take this into account during the evaluation of the discovered equations.

Phase Space Due to the low dimensionality of the Non-Linear Pendulum system, I was able to find an intuitive way of visualizing the resulting equations and two representative trajectories in the phase space in a 2D-image. Although

this visualization is only available for one of the three systems, it gives valuable insight into the differences between the tested variants.

Relevant choices to deviate from this methodology as well as further metrics specific to certain experiments will be explained just-in-time in the Experiments Section.

5 Experiments

5.1 Verfication

The verification was performed in two steps: First, I compared the output of the evaluation-notebooks in the source code [13] to the claimed results in the publication [3] and executed the evaluation-notebooks again to detect possible mismatches between the publication, the last state of the evaluation-notebooks and the actual models stored in the repository.

I then copied the original training-notebook of each scenario to train 10 verification models each (labelled 'V{1-10}') according to the methodology. Instead of manually running the notebook ten times, I automated the verification using a for-loop, which allowed me to do research in parallel. Since the python kernel would not reset after each training, I used `tensorflow.reset_default_graph()` before each training to ensure that the next model was re-initialized randomly and not using the parameters of the last model.

Incompatibility between new and old Tensorflow versions and poor documentation of used resources made the verification very cumbersome. However, using the `tensorflow.compat.v1` API and disabling TensorFlow 2.x behaviors, I was eventually able to perform the verification as intended.

5.2 Re-Implementation & Replication

Although working with the original source code was manageable, obsolete versions and the quality of the source code called for steps to improve its usability and maintainability. For this reason, I re-implemented the method from scratch in PyTorch [14].

The original implementation made use of Tensorflow's functional formulation and *feed-dicts*, which contained the input and the state of the model. Although this approach technically works fine, it is reasonable to prefer object

oriented programming for a two reasons: 1) Clarity. Splitting relevant parts of the method into classes and components helps creating an overview and encourages abstraction and generalization, thus leading to better understandable and usable code. 2) Flexibility. In the functional formulation, since everything is comprised of nothing but functions, adding, removing and replacing features leads to messy code, in which variants of a method are poorly defined and elusive, whereas in the object oriented approach, features can be enabled or disabled with behavioral patterns such as template methods, and added by just writing another class. Each variant is fully defined by the combination of used components and their parameters.

At the heart of my implementation lies the abstract `Discovery`-class[1]. It only implements two methods to transform the time-derivative through the encoder or decoder by the chain rule, inspired by the original transformation code. Thus, the equivalent of the original SINDy-Autoencoder in my implementation inherits the transformation rules from the `Discovery`-class. The autoencoder, which itself is a class with two attributes corresponding to its encoder and decoder, is passed as a parameter to the `Discovery`-class during initialization.

Following the idea of the well documented DeepMoD codebase [15], the collection and computation of the library terms is entirely handled by a separate `Library`-class, an instance of which is passed as a parameter to the `Discovery`-class just like the autoencoder. I also implemented a `VerboseLibrary`-class, which would print the symbolic terms of a given library. In the original implementation, the symbolic terms had to be figured out manually by iterating through uncommented nested for-loops in the source code. Although all terms and the order in which they appear the library were already known for the three scenarios, the verbose library greatly helped troubleshooting other parts of the method and accelerated the evaluation.

The most important design choice, however, was the separation of the Sequential Thresholding feature into a separate class and its generalization via inheritance. If the `Discovery`-class was passed any instance of a thresholder-class during initialization, it would ask the thresholder to apply its criteria based on the history of the coefficient values and return its decision in every epoch during training. This template pattern allowed for easy experimentation with different thresholding approaches such as the one described in Section 5.4 and marked a big step towards improving the method.

[1]The `Discovery`-class is called `TrajectoryDiscovery` in the source code for backwards-compatibility reasons.

Also, the simple addition of a progress bar to the output of the training process made monitoring clearer and helped scheduling multiple experiments in rapid succession or even in parallel.

Finally, saving instances of models is also designed to save their loss and coefficient history, which became especially handy during the failure analysis in Section 5.3.

During the re-implementation, I performed component tests on the most relevant classes and application tests with the pendulum system, since its low dimensionality meant easier troubleshooting. When it worked as intended, I validated the re-implementation on the Lorenz and Reaction-Diffusion systems and corrected errors that had remained unnoticed prior to the validation due to the singular dimensionality of the pendulum system.

Concluding the re-implementation and replication, I trained 10 models (labelled 'R{1-10}') for each system to obtain an up-to-date baseline result.

5.3 Analysis

As will become clear in Section 6, verification and replication of the claimed results only was a partial success: For one, there were slight discrepancies between the two. Furthermore, many identified equations contained unusual terms or were missing obviously plausible ones.

In the first steps of the analysis, I focus on the Non-Linear Pendulum which serves as an illustrative example through which the problem can be well clarified. In particular, the existence of only one desired term in the equation, $\sin(z)$, makes it easy to highlight deviations and unexpected behaviour.

Latent Space Analysis To find the origin of the observed phenomena, I started by examining the latent space of the verification models. I searched for unusually narrow distributions and particularly weak correspondences between the ground truth and the encoded coordinates in a scatter plot. My hope was that correlations between features in the latent distribution and the quality of the identified equations would give insights as to why sometimes some terms would be preferred over others. For example, if the latent distribution was very narrow, the term $\sin(z)$ would be nearly indistinguishable from z which could be a sign that the encoder had absorbed the non-linearity of the system. If no obvious correlations were to be found, there would have to be different explanation.

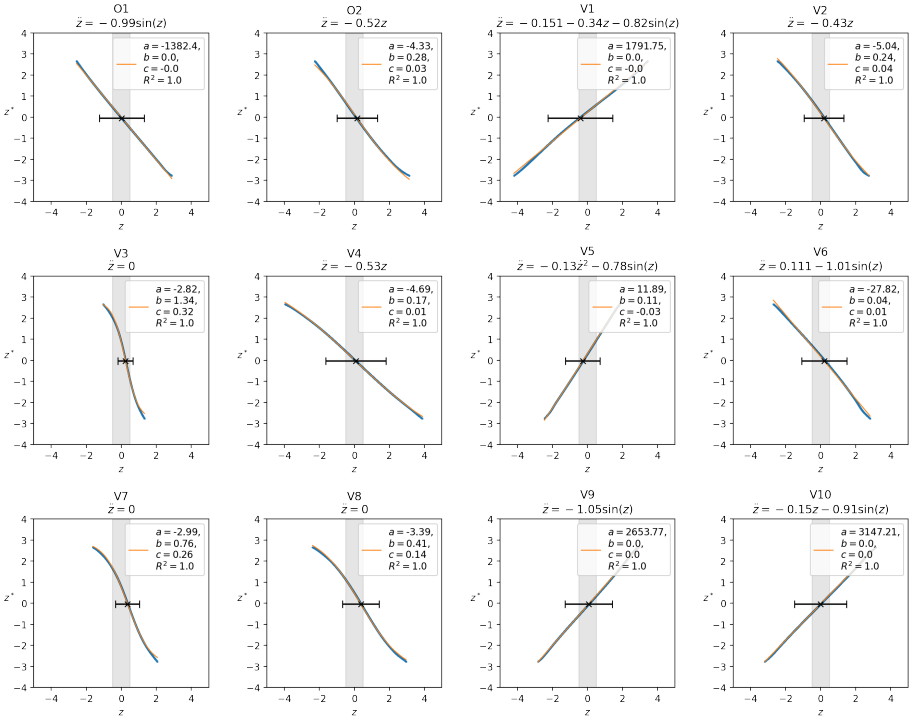


Figure 5: **Latent Space Analysis (Non-Linear Pendulum).** The ground truth coordinates of the test set z^* are compared with their representation $z = \varphi(\mathbf{x}^*(z^*))$ in the latent space (*blue scatter plot*) for the two provided original models (O) and the ten verification models (V). The function $z^* = a \cdot \tanh(bz - c)$ (*orange line*) is fit to the scattered data by means of least squares. R^2 indicates the adjusted coefficient of determination of the fit rounded to two decimal places. The black errorbar indicates the mean and standard deviation of the encoded distribution. The region $[-0.5, 0.5]$ where $\sin(z) \approx z$ is marked in gray. The identified equation is displayed directly below the model name.

Figure 5 hints that rather weak correspondences correlate with the resulting equation being $\ddot{z} = 0$, which corresponds to every coefficient being turned off at some point during training and can be considered a failure. Since the scattered correspondence of the ground truth coordinates z^* and the encoded coordinates $z = \varphi(\mathbf{x}^*(z^*))$ looked suspiciously similar to a hyperbolic tangent, I performed a least-squares fit of the function

$$z^* = a \cdot \tanh(bz - c) \tag{16}$$

to the data. To quantify the linearity of the correspondence, I used the the fit-parameter b: If b is small, the tanh-function is stretched and linearized around $z = c$. Thus, small values of b correspond to strong linearity and good correspondence of the encoded coordinates to the ground truth coordinates.

In Figure 6, the observations of the latent space are summarized in a single scatter plot. The data points are clearly divided into three regions: 1) Relatively wide latent distributions and, unsurprisingly, good correspondence to the ground truth coordinates result in the identification of the $\sin(z)$-term, while 2) equally wide latent distributions without ground truth correspondence result in the identification of other terms rather than the $\sin(z)$-term. 3) Narrow latent distributions and low correspondence seem to coincide with no terms being identified.

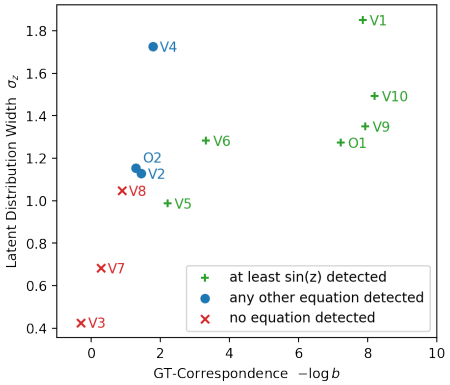


Figure 6: **Categorization of Qualities (Non-Linear Pendulum).** The standard deviation of the encoded test set σ_z and the correspondence of the encoded coordinates z with the ground truth coordinates z^* are scattered and colored according to qualities of the identified equation. The corresponcence is displayed logarithmically to emphasize differences for small b.

While it seems that good ground truth correspondence and a wide latent distribution are desirable features of the latent space for high quality equations, one has to be careful and consider the possibility that the lack of these features could merely be the effect of an earlier problem, namely the evolution of the equation during the beginning of the training to which the autoencoder would adapt.

This realization shows that the latent space analysis served as a good entry point into finding the root cause of the unusual or missing terms by understanding the link between the latent distributions and the quality of the equations,

however, it is limited in that it is only capable of showing the end result.

Loss Analysis The question whether the resulting encoding and latent distribution is the cause, or the effect of other failures was still left open by the latent space analysis and it seemed to require an analysis of the models' properties during training. Using the built-in functionality of the re-implementation, I first examined the loss components over time in the hope to find the responsible part of the method by detecting unusual evolutions of a component during training.

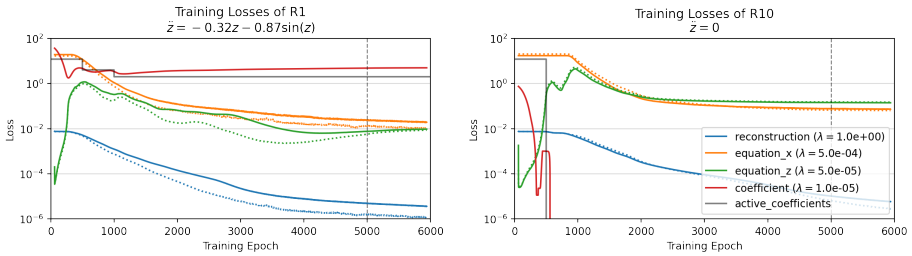


Figure 7: **Training Loss (Non-Linear Pendulum).** The training (*solid lines*) and validation (*dotted lines*) loss is plotted over the training epochs for two selected models with very different resulting equations. The loss histories are smoothed by convolution with a 1-normed, uniform vector of length 120. The refinement begins in epoch 5000 (*vertical dashed line*). λ depicts the loss weight of each component.

Indeed, in cases where no equation was detected (Figure 7, right), the coefficient loss (*red*) seems to be cut off in epoch 500 as an effect of all coefficients being turned off. Another notable difference between the two selected models is that the history of the failed one seems to be delayed compared to the successful one. Furthermore, the initial drop of the coefficient loss is more pronounced in the failed model. However it seems as though it begins to increase again before the cut-off.

At this point it became obvious that a detailed analysis of the coefficient histories especially around epoch 500 was necessary.

Coefficient Analysis The analysis of the coefficient history, too, benefitted from the early decision to record the coefficient history during training. For the analysis, I was particularly interested in the differences of coefficient histories that lead to three features in the resulting equation: 1) The equation contained

at least the $\sin(z)$ term, 2) the equation did not contain the $\sin(z)$ term but other terms instead, and 3) the equation contained no terms at all.

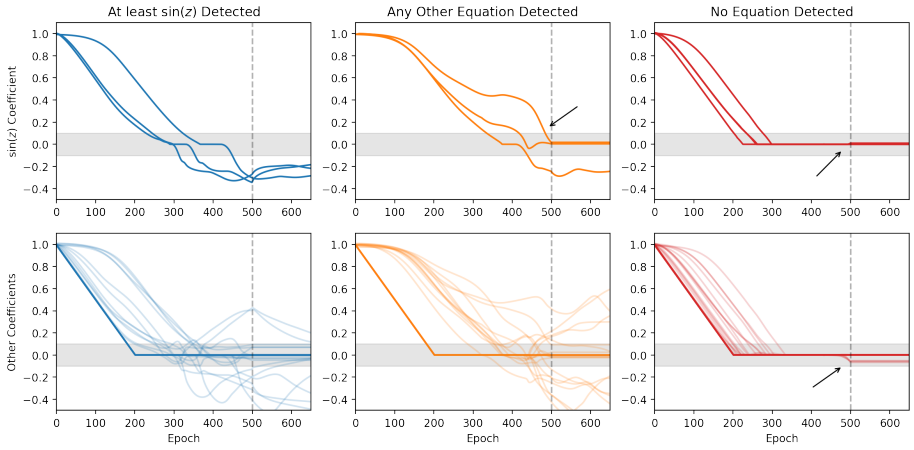


Figure 8: **Coefficient Histories with ST (Non-Linear Pendulum).** All 10 replication models R{1-10} are categorized according to the quality of their identified equation (*columns*). The coefficient histories of all models in one category are each superimposed in two figures, categorized into the desired $\sin(z)$ coefficient and other coefficients (*rows*). The x-axis is limited to 650 epochs to highlight the early evolution in detail. The gray horizontal bar depicts the thresholding region, within which coefficients are turned off in thresholding epochs for the rest of training. The first thresholding takes place in epoch 500 (*vertical dashed line*). The black arrows indicate where the Sequential Thresholding disables coefficients that still change or begin to recover after the initial pseudo-convergence.

Figure 8 compares the coefficient histories of all replication models R{1-10}. In the successful case (i.e. when the $\sin(z)$ term had been detected), the relevant coefficient adapted fast enough and reached negative values before the first thresholding. In the case where the $\sin(z)$ term was not detected, but another one was, the $\sin(z)$ coefficient adapted slower and was switched off in 2 of 3 models by the ST algorithm, although it was still changing (!). In the case of total failure, two features are striking. On the one hand, although it seemed to just begin to change again, the $\sin(z)$ coefficient was also switched off in this case. On the other hand, the evolution of the $\sin(z)$ coefficient was faster than in the success cases and it remained stuck at the zero line for a while. The latter observation suggests an unsuitable random initialization of the autoencoder, particularly the encoder. If the measurements are transformed

into messy coordinates, no sensible equations with the allowed terms in these coordinates can be found and thus all terms quickly reduce to 0 under the influence of the applied L_1 loss. Following this interpretation, the recovery of the encoder is hinted by the coefficient histories detaching from 0 just before being turned off.

To estimate the frequency of these suspiciously obvious premature thresholding events and their effect on the training, I determined the rate of change of the coefficients when they were being turned off by the ST. Since the coefficient histories were affected by noise, computing the stepwise difference of consecutive coefficient values was unreliable. Instead, I approximated the trend of a coefficient ξ turned off in epoch t by the difference of the mean coefficient values of two consecutive intervals,

$$T = \mathrm{mean}(\xi_{[\mathrm{t-L:t}]}) - \mathrm{mean}(\xi_{[\mathrm{t-2L:t-L}]}) \tag{17}$$

with length scale $L = 5$ and the significance of the trend,

$$S_T[\sigma] = \frac{T}{\sigma_T} = \frac{T}{\sqrt{\mathrm{var}(\xi_{[\mathrm{t-L:t}]}) + \mathrm{var}(\xi_{[\mathrm{t-2L:t-L}]})}}. \tag{18}$$

The significance served as a criterion of how much of the apparent trend T can be attributed to noise. If the variance of the recent history were large relative to the change, the change would not be significant and vice versa.

To get a sense of the speed of the trend, I multiplied the trend values whose two mean epochs $t - \frac{3}{2}L$ and $t - \frac{1}{2}L$ are exactly $L = 5$ epochs apart by 200 to project the local change in the coefficient onto time units of 1000 epochs. This conversion is not to be taken as an actual change over 1000 epochs, but as an illustration of the momentary rate of change. If we consider a change in coefficients of at most two times the thresholding limit (from -0.1 to +0.1) over the entire training (5000 epochs with L_1 loss, followed by 1000 without) converged, we can safely justify thresholding below $\Delta\xi = \frac{0.2}{5000\ \mathrm{epochs}}$. Furthermore, I estimate the upper limit of the allowed rate of change in thresholding at $\Delta\xi = \frac{0.2}{100\,\mathrm{epochs}}$. Above this rate of change it is not possible to speak of convergence and it has to be assumed with high certainty that the coefficient was switched off by mistake.

While the naive probability of turning off the coefficient at such a rate of change drops to $\frac{100}{500} = 20\%$ (the coefficient spends 100 epochs in thresholding and every 500 epochs, ST is applied), this is still too much. Regardless of whether

the unjustified thresholding leads to success or not, the ST leaves it more or less to chance which coefficients are switched off in the given intervals and can therefore be assessed as unreliable.

Figure 9 quantifies the rate of change and signficance of the trend at the time of disabling for all coefficients ever disabled in R{1-10}. While some coefficients were turned off while they could be assumed to be converged according to the above reasoning (*green area*), a comparable number were turned off when their convergence was still at least controversial (*orange, red area*). With the exception of the Chaotic Lorenz System, the significance of the rate of change shows that these values were rarely the result of noise and predominantly actual changes.

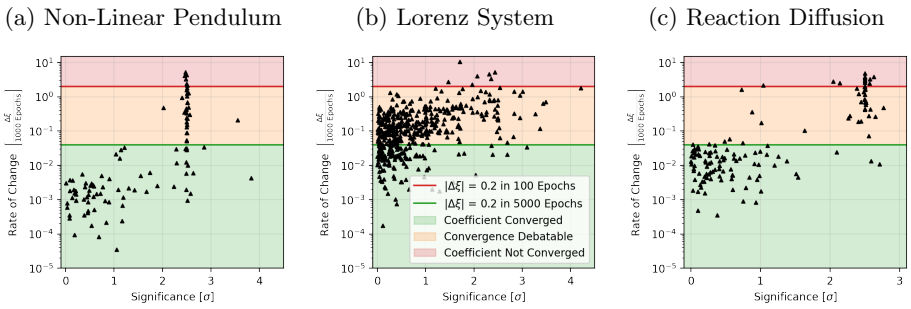


Figure 9: **Coefficient Trends with ST at the Time of Thresholding.** The scatter plots show the projected rate of change and significance of the rate of change of coefficients ξ when they were turned off by the Sequential Thresholding algorithm. The colored areas indicate an estimate of the plausibility that the coefficient was turned off justifiably. The trend was calculated using Equation 17 and $L = 5$.

In this argument, I did not distinguish between success cases but made a rather general observation. In fact, for many of the failed models, the rate of change when the coefficients were turned off was negligible (Figure 8, right) since their coefficients were still stuck at $\xi = 0$. In theory, the recovery could also have occured later, for example at epoch 520 or perhaps even 700, except that the coefficients did not get a chance to evolve, since they had already been turned off in epoch 500. Accordingly, the discussion about the rate of change at the time of thresholding can be considered biased towards less extreme cases as it does not account for the suspected delays in the adaptation of the encoder

and the resulting cases and misinterpretation of pseudo-converged coefficients. However, even with this consideration in mind, the number of coefficients that were arbitrarily turned off by the Sequential Thresholding despite not being remotely converged raises suspicion.

I also did not distinguish between $\sin(z)$ coefficients and the other coefficients, because at the early stages of the training every coefficient may play equally important roles for the stability of the algorithm until the autoencoder has adapted to the simple coordinates.

Coefficient Analysis without Thresholding To address and test the hypothesis that the pseudo-converged coefficients would actually start to change again if one were to let them, I conducted the same experiment (i.e. replication) again but without the ST component. Thus, no coefficient would be turned off during training.

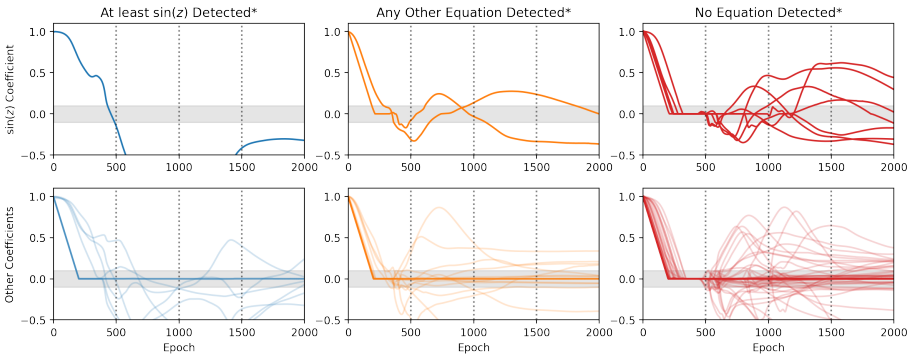


Figure 10: **Coefficient Histories without ST (Non-Linear Pendulum).** All 10 modified replication models are categorized according to the quality of their identified equation (*columns*). The coefficient histories of all models in one category are each superimposed in two figures, categorized into the desired $\sin(z)$ coefficient and other coefficients (*rows*). The x-axis is limited to 2000 epochs to highlight the first half of the training. The gray horizontal bar depicts the thresholding region, within which coefficients would have been turned off in thresholding epochs by the ST algorithm for the rest of the training. *For a term to be treated as 'detected', its coefficient had to be outside the thresholding range in every thresholding epoch (*vertical dashed lines*).

The free coefficient histories in Figure 10 show that right around epoch 500, the pseudo-converged coefficients in fact do start to change again (!). For one, this strengthens the hypothesis that the initialization of the encoder often corresponds to very unsuitable coordinates from which the algorithm first has to

recover. Most importantly, however, it shows very clearly that the application of the Sequential Thresholding algorithm in the SINDy-Autoencoder is inappropriate because it arbitrarily and prematurely turns off coefficients, leading to many false negatives in the detected terms of the resulting equation.

5.4 Improvements

There are two problems with using Sequential Thresholding (ST):

1. The ST epochs are fixed and their interval is arbitrarily chosen.
   - → Pseudo-converged coefficients that are in the ST zone in an ST epoch, although they would have changed shortly after, are turned off.
   - → Coefficients that otherwise that happen to be in the ST zone by chance during an ST epoch are turned off.
2. In ST epochs, coefficients are turned off regardless of their momentary rate of change.
   - → Coefficients that are in the process of changing are turned off.

I therefore propose a conceivably better thresholding method, the *Patient Trend-Aware Thresholding* (PTAT): In each epoch $t = 1, 2, ..., T$, a coefficient is turned off only if

1. the absolute value of the coefficient, $|\xi_{t'}|$, was smaller than a threshold a in all L past epochs, and,
2. the absolute value of the momentary change of the coefficient, $|\delta\xi_{t'}| = |\xi_{t'} - \xi_{t'-1}|$, was smaller than a second threshold b in all L past epochs.

With suitable thresholds a, b and a suitable patience L, I hypothesize that this solves the problems of the ST algorithm.

An important consequence to mention though is the possibility that, at the end of training with the PTAT algorithm, some active coefficients in the resulting equations may end up having absolute values less than the threshold a. This happens primarily when coefficients did not satisfy both thresholding criteria for long enough before the training ended. Since they would most probably be turned off eventually when exposed to slightly longer training, I do not treat them the same as the other *genuine* coefficients with $|\xi| \geq a$ at the end of training. Hence, I label coefficients with $|\xi| < a$ at the end of training *non-genuine* coefficients and mark them in gray in the Results Section and the Appendix.

PTAT Estimate To quickly estimate the prospects of success by the proposed thresholding algorithm, I applied its criteria with the adequate choice $a = 0.1$, $b = \frac{0.2}{100\,\text{epochs}} = \frac{0.002}{\text{epoch}}$ and $L = 1000$ epochs on the coefficient histories that had been recorded previously without thresholding. For each example system, I created a confusion matrix comparing the qualities of the identified equations (Table 1). In the Chaotic Lorenz System, I observed more noise in the coefficient histories than for the other systems and raised the trend threshold to $b = \frac{0.02}{\text{epoch}}$.

(a) Non-Linear Pendulum

		ST*		
		At least sin(z)	Other	None
PTAT*	**At least sin(z)**	1	2	7
	Other	0	0	0
	None	0	0	0

(b) Reacation Diffusion System

		ST*		
		Linear	Other	None
PTAT*	**Linear**	9	6	3
	Other	0	0	2
	None	0	0	0

(c) Chaotic Lorenz System

		ST*	
		up to $\mathcal{O}(2)$	$\mathcal{O}(3)$
PTAT*	**up to $\mathcal{O}(2)$**	22	0
	$\mathcal{O}(3)$	1	7

Table 1: **Estimated Comparison of the ST and PTAT.** Confusion matrices of the estimated number of features detected with imagined ST (ST^*) and imagined PTAT ($PTAT^*$) when applied to the pre-recorded coefficient histories without any thresholding. Desirable features of the resulting equation are marked in bold.

The estimates indicate that the patient thresholding algorithm should work significantly better for the Non-Linear Pendulum and the Reaction Diffusion System. For the Lorenz-System, no significant improvement should be expected.

However, there is one important bias to note: Due to the fact that the coefficient history for this experiment had been pre-recorded, the coefficients that would have been turned off by each algorithm, were not, and influenced all

other coefficients, distorting the subsequent coefficient history such that it no longer corresponded to the imagined one including the thresholding.

Thus, the estimate should not be taken too seriously but rather as a screening for the proposed algorithm.

PTAT Experiment After the rather successful screening, I finally conducted an experiment in which I replaced the ST-component with the PTAT-component. 10 Models for each example system were trained with the parameters listed in Table 2. I also extended the training of the Reaction-Diffusion models from 3000 + 1000 to 9000 + 3000 epochs, since it was apparent from the coefficient histories (Figure 14 in Apendix B) that none of the replication models had properly converged within the original time.

System	a	b	L	Reasoning
Non-Linear Pendulum	0.1	0.002	1000	
Chaotic Lorenz	0.1	0.02^*	1000	*stronger noise
Reaction-Diffusion	0.1	0.002	3000^*	*stretched training

Table 2: **PTAT Training Parameters.** Threshold a, trend threshold b and patience L. Deviations from the original proposition are marked with * and explained in the *Reasoning* column.

5.5 Sanity Check

To clarify the importance of the constraint that SINDy provides during training, I conducted an experiment in which I trained 10 models identical to the original models for each of the three scenarios, but removed SINDy, so that only the autoencoder remains (labelled pAE {1-10}). The only special feature remained that not only the coordinates $\mathbf{x}$ are transformed to $\mathbf{z}$, but also by chain rule the time derivatives $\dot{\mathbf{x}} \mapsto \dot{\mathbf{z}}$ or $\ddot{\mathbf{x}} \mapsto \ddot{\mathbf{z}}$. From the original loss only the reconstruction terms $\mathcal{L}_{\mathbf{x}}$ and $\mathcal{L}_{\dot{\mathbf{x}}}$ (or $\mathcal{L}_{\ddot{\mathbf{x}}}$) remained, which I weighted both equally.

Since the autoencoder does not get any information regarding the shape of the ODE, and the dynamics in the latent coordinates could be very complex, I did not expect SINDy to perform well in the coordinates transformed by the pre-trained encoder. Instead, I decided to use PySR [6] because is more generally formulated and would, should the dynamics be simple, discover equations comparable to SINDy. If the dynamics would turn out complicated, SINDy would fail and provide no useful information, while pySR would certainly de-

tect an equation. One drawback, however, is PySR's runtime. For a time limit of a few hours per equation, I had to partially reduce the number of training instances to 1000.

Also, the resimulation of the Non-Linear Pendulum with the pAE was performed using Euler's method. While simple second order differential equations can be reduced to a system of two coupled first order differential equations, the transformation is not as trivial with complicated equations such as the ones discovered with PySR in the pAE variant. The flexibility offered by Euler's method comes at the possible cost of numerical stability which I countered using sufficiently small integration steps, namely 100 steps per original simulation step, resulting in 5000 steps per second. Still, numerical stability for the whole variety of equations cannot be guaranteed.

6 Results

6.1 Dynamics Error

The dynamics errors capture the difference between the actual, transformed left hand side (LHS) of the equation to its right hand side (RHS) as approximated by the symbolic regression method in the latent space. In Tables 3-5, they are listed for each system respectively. The *SR-Method* column indicates which symbolic regression method was used in the latent space of the autoencoder. The identifier of each experiment is listed in the *Variant* column. The label *'O'* corresponds to the models in the original GitHub-repository [13] of which there were only two. The last three columns contain the mean and standard deviation of the reconstruction and dynamics errors in terms of the fraction of variance unexplained (FVU), as calculated by Equation 14 for the variables $\mathbf{x}$, $\dot{\mathbf{x}}$ or $\ddot{\mathbf{x}}$, and $\dot{\mathbf{z}}$ or $\ddot{\mathbf{z}}$. The values are scaled by a fixed order of magnitude displayed in the header of each column for the purpose of readability. The bracketed values are the FVU under the questionable assumption that the data was centered, as calculated by K. Champion et al. according to Equation 13 [13] (Conversion from the FVU_y^0 to FVU_y is impossible because the original mean of y is unknown. Hence, the original values are displayed and marked as such). All FVUs are subject to minimization (↓).

SR-Method	Variant	Source	$\text{FVU}_{\mathbf{x}}\,[10^{-4}]\downarrow$	$\text{FVU}_{\dot{\mathbf{x}}}\,[10^{-4}]\downarrow$	$\text{FVU}_{\ddot{\mathbf{x}}}\,[10^{-2}]\downarrow$
SINDy	N/A	K. Champ. et al. [3]	(8)	(3)	(2)
SINDy	O1	Output [13]	(3)	(4)	(2.1)
SINDy	O1	Verified	1.1	2.0	0.4
SINDy	O2	Output [13]	(9)	(10)	(21)
SINDy	O2	Verified	3	8	13
SINDy	V {1-10}	Verified	7 ± 6	27 ± 22	30 ± 40
SINDy	R {1-10}	Replicated	7 ± 6	26 ± 21	50 ± 40
SINDy	PTAT {1-10}	Modified	9 ± 13	8 ± 6	2.1 ± 0.9
pySR	pAE {1-10}	Modified	500 ± 260	$16\text{k} \pm 14\text{k}$	40 ± 80

Table 3: **Dynamics Errors (Non-Linear Pendulum).**

(a) In-Distribution

SR-Method	Variant	Source	$\text{FVU}_{\mathbf{x}}\,[10^{-5}]\downarrow$	$\text{FVU}_{\dot{\mathbf{x}}}\,[10^{-4}]\downarrow$	$\text{FVU}_{\ddot{\mathbf{x}}}\,[10^{-4}]\downarrow$
SINDy	O1	K. Champ. et al. [3]	(3)	(2)	(7)
SINDy	O1	Output [13]	(2.7)	(5)	(7)
SINDy	O1	Verified	2.7	1.8	10
SINDy	O2	K. Champ. et al. [3]	(0.2)	(0.6)	(3)
SINDy	O2	Output[13]	(0.2)	(0.9)	(5)
SINDy	O2	Verified	0.2	1.1	6
SINDy	V {1-10}	Verified	5 ± 4	11 ± 8	45 ± 20
SINDy	R {1-10}	Replicated	5 ± 4	10 ± 6	39 ± 15
SINDy	PTAT {1-10}	Modified	2.7 ± 1.5	7.2 ± 2.0	36 ± 9
pySR	pAE {1-10}	Modified	3 ± 3	$5.0\text{k} \pm 1.9\text{k}$	$4.0\text{k} \pm 1.4\text{k}$

(b) Out-Of-Distribution

SR-Method	Variant	Source	$\text{FVU}_{\mathbf{x}}\,[10^{-2}]\downarrow$	$\text{FVU}_{\dot{\mathbf{x}}}\,[10^{-2}]\downarrow$	$\text{FVU}_{\ddot{\mathbf{x}}}\,[10^{-2}]\downarrow$
SINDy	O1	K. Champ. et al. [3]	-	-	-
SINDy	O1	Output [13]	(1.3)	(11)	(8)
SINDy	O1	Verified	1	9	8
SINDy	O2	K. Champ. et al. [3]	-	-	-
SINDy	O2	Output[13]	(1.5)	(10)	(6)
SINDy	O2	Verified	2.1	14	8
SINDy	V {1-10}	Verified	1.6 ± 0.4	16 ± 3	24 ± 5
SINDy	R {1-10}	Replicated	1.6 ± 0.5	13.1 ± 2.5	18 ± 5
SINDy	PTAT {1-10}	Modified	1.9 ± 0.7	15 ± 5	22 ± 6
pySR	pAE {1-10}	Modified	510 ± 120	$6.7\text{k} \pm 2.1\text{k}$	$9\text{k} \pm 3\text{k}$

Table 4: **Dynamics Errors (Chaotic Lorenz System).**

SR-Method	Variant	Source	$\text{FVU}_{\mathbf{x}}\,[10^{-2}]\downarrow$	$\text{FVU}_{\dot{\mathbf{x}}}\,[10^{-2}]\downarrow$	$\text{FVU}_{\dot{\mathbf{z}}}\,[10^{-2}]\downarrow$
SINDy	O1	K. Champ. et al. [3]	(1.6)	(1.6)	(0.2)
SINDy	O1	Output [13]	(1.6)	(1.6)	(0.2)
SINDy	O1	Verified	1.6	1.6	0.21
SINDy	O2	K. Champ. et al. [3]	-	-	-
SINDy	O2	Output[13]	(1.6)	(1.6)	(0.8)
SINDy	O2	Verified	1.6	1.6	0.8
SINDy	V {1-10}	Verified	1.69 ± 0.24	2.1 ± 1.2	14 ± 29
SINDy	R {1-10}	Replicated	1.64 ± 0.1	1.9 ± 0.6	14 ± 22
SINDy	PTAT {1-10}	Modified	0.16 ± 0.05	0.18 ± 0.04	0.103 ± 0.023
pySR	pAE {1-10}	Modified	0.067 ± 0.010	0.28 ± 0.12	0.26 ± 0.12

Table 5: **Dynamics Errors (Reaction-Diffusion System).**

Overall, the FVUs are very small and in the order of 10^{-3} on average which means that most of the variance in the dynamics can be explained by the discovered equations. Still, there are notable differences between the tested variants.

Verification First of all, the claimed errors by K. Champion et al., if present, roughly correspond to at least one of the Output cells in the original notebooks. The verification of the original models, which only included re-running the original analysis notebook with the corrected formula for the FVU, resulted in slightly different values, hinting that the data was not centered and the use of Equation 13 is inappropriate, as it does not properly account for the variance.

The 10 verification models V{1-10} performed mostly similarly to the original models in reconstruction ($\text{FVU}_{\mathbf{x}}$), however their dynamics errors are up to one order of magnitude higher than the ones resulting from the original models. This increase also came with a large model-to-model variance, often of the same magnitude as the value itself, indicating that the performance is very scattered and unreliable and furthermore hinting the existence of outlier models with extremely high FVU, since $\text{FVU} \geq 0$.

Re-Implementation & Replication The replication models suffer from the same characteristics as the verification models which reveals to things: For one, the re-implementation and replication succeeded, since their resulting FVUs can be considered the same as the verification ones. Secondly, since both the original and re-implemented code resulted in worse performance on average than claimed by K. Champion et al. in [3], the suspicion of cherrypicking the original models is reasonable. In their work, K. Champion et al. state that O{1-2} had been the best performing models [3] but the absence of detailed performance results for the other 8 models is intransparent.

PTAT-Improvement The PTAT variant scores better in most cases, especially in the $\text{FVU}_{\dot{\mathbf{x}}}$ and $\text{FVU}_{\dot{\mathbf{z}}}$ for the Non-Linear Pendulum and all FVUs for the Reaction-Diffusion System, where they decreased by up to an order of magnitude. However, the improvements for the Reaction-Diffusion System could also be due to the extended training time justified in Section 5.4. The $\text{FVU}_{\mathbf{x}}$ for the Non-Linear Pendulum was largely unaffected and can be considered slightly worse. However, similarly to the PTAT FVUs for the Chaotic Lorenz System, this apparent worsening with respect to the verification and replication models is insignificant due to the overlapping uncertainties.

Sanity Check The results of the experiment in which pySR was used to find the equations inside the latent space of a plain, pretrained autoencoder (pAE{1-10}) clearly shows the necessity of the SINDY-constraint during training. With the exception of the Reaction-Diffusion System, all FVUs are multiple orders of magnitude larger than any other variant. Interestingly, for the the Reaction-Diffusion System, the pAE variant only performs slightly worse than the PTAT variant. This indicates that the conditions in which the autoencoder was trained (i.e. two latent dimensions, the loss weights and encoder architecture) already promoted the transformation into coordinates in which the dynamics follow an oscillation that could be described by a symbolic equation without problems.

6.2 Equations

Showcase Tables 6-8 present a showcase of the best and worst resulting equations of each variant. The labels *Best RHS* and *Worst RHS* refer to the equations' interpretability which takes into consideration their number of terms, order of polynomial terms, how nested they are and their correspondence to the ground truth equation. The selection is subjective and exemplary. All equations discovered can be found in Appendix D. In the case of a system of equations, the n-th sub-row within the row of a particular variant represents the RHS of the n-th equation in that system of equations. Non-genuine coefficients as defined in Section 5.4 are marked in gray.

While the equations of the selected original models by K. Champion et al. correspond to the ground truth equations [3], the same cannot be said in general for the other variants.

The verification, replication and PTAT equations share similar features. However, the worst PTAT equation for the Non-Linear Pendulum is more com-

Variant	Best RHS	Worst RHS
GT	$-\sin(z)$	
O {1, 2}	$-0.99\sin(z)$	$-0.52z$
V {9, 5}	$-1.05\sin(z)$	$-0.13\dot{z}^2 - 0.78\sin(z)$
R {6, 2}	$-0.13z - 0.88\sin(z)$	$0.27z^2 - 0.72\dot{z}^2 - 1.12z\dot{z}^2$
PTAT {3, 8}	$0.08 - 0.08z - 1.02\sin(z)$	$0.24 - 0.35\dot{z}^2 + 0.34z\dot{z}^2 - 0.79\sin(z)$
pAE {2, 4}	$z\left(5\sin\left(\sin\left(\sin\left(0.12z^2\right)\right)\right) - 2.4\right)$	$\frac{z^2\sin(z+\sin(z)+50)}{\sin(\sin(\dot{z})-400)}$

Table 6: **Selection of Identified Equations (Non-Linear Pendulum).** For each variant, the RHS of the best and worst detected equations are listed. The LHS of all equations is $\ddot{z}$.

Variant	Best RHS	Worst RHS
GT	$-10z_1 + 10z_2$	
	$28z_1 - z_2 - z_1z_3$	
	$-2.7z_3 + z_1z_2$	
O {1, 2}	$-10.04z_1 - 10.9z_2$	$5.68z_3 - 3.49z_1z_2 + 2.12z_2z_3$
	$-0.94z_2 - 9.55z_1z_3$	$-10.27 - 2.72z_2 + 2.63z_1^2 - 1.01z_3^2$
	$7.13 - 2.68 - 3.12z_1z_2$	$-10.94z_3 - 5.59z_1z_2 + 3.44z_2z_3$
V {2, 8}	$7.61 - 2.91z_1 + 3.13z_2z_3$	$-6.55 - 7.94z_3 + 5.42z_2^2 + 7.13z_2z_3 - 0.33z_2^2z_3 - 0.25z_2z_3^3$
	$-10.6z_2 - 10.59z_3$	$18.86 + 9.87z_1 - 8.93z_2 + 0.23z_2z_3 - 0.31z_1z_2^2$
	$-8.74z_1z_2 + 0.18z_1z_3$	$-0.64z_1 - 5z_1z_2 + 0.9z_1z_3 + 0.29z_3^2 - 0.2z_2^2z_3$
R {4, 1}	$-9.98z_1 - 10.47z_2$	$1.43 - 5.26z_1z_2 + 0.67z_2^2 - 0.85z_2z_3^2$
	$-z_2 - 9.72z_1z_3$	$-9.18z_3 - 1.35z_1^2 - 0.96z_1z_3 + 1.6z_2z_3 - 0.16z_1z_2^2$
	$7.28 - 267z_3 + 2.96z_1z_2$	$0.54z_1 + 4.48z_2z_3 + 0.26z_3^2 + 0.93z_1^2z_2 + 0.22z_1z_2z_3 - 0.44z_2^3$
PTAT {2, 9}	$-7.43 - 2.7z_1 + 3.35z_2z_3$	$-8.98 - 1.1z_1 - 0.19z_1z_3 + 8.18z_2^2 - 2.89z_3^2$
	$-0.96z_2 - 9.6z_1z_3$	$2.47 + 5.6z_1 - 5.87z_3 - 1.91z_1z_2 - 1.3z_1z_3 + 0.15z_2^2 - 0.08z_2^3$
	$10.43z_2 - 9.98z_3$	$-5.21 + 5.19z_1 - 11.59z_3 + 2.75z_1z_2 + 1.83z_1z_3$
pAE {2, 9}	$-6z_1 - 3z_2 - z_3 - 6z_2^2$	$-5 + z_1 + 8z_2 - 15z_3 + 40z_1z_3 + 40z_2z_3$
	$0.8 - 7z_1 + 40z_1^2 + 4z_1z_2$	$-4 - 22z_1 - 32z_3 - 23z_1^2 - 60z_1z_3$
	$9z_1 + 10z_2 - 9z_3 + 10z_1^2 - 19z_1z_2 + 10z_2^2$	$-0.18 - 1.9z_1 + 1.9z_2 - 1.9z_3 - 1.8z_1^2 - 1.8z_1z_2 \swarrow$ $-15z_1z_3 - 17z_2z_3 + 17z_3^2 + 15z_1z_2^2 - 15z_1z_2z_3 + 15z_2^3$

Table 7: **Selection of Identified Equations (Chaotic Lorenz System).** For each variant, the RHS of the best and worst detected equations are listed. The LHS of each system of equations correspond to $\dot{z}_1$, $\dot{z}_2$, $\dot{z}_3$ for each sub-row.

plicated than any of the other two. In contrast, the worst PTAT equation for the Reaction-Diffusion System can be considered as good as the best ones of all variants.

The most extreme and complicated equations were discovered by PySR in the intricate latent space of the plain pretrained autoencoder. In the case of

Variant	Best RHS	Worst RHS
GT	z_2	
	$-z_1$	
O {1, 2}	$-0.85z_2$	$-1.1\sin(z_2)$
	$0.97z_1$	$0.82z_1$
V {10, 1}	$1.03z_2$	$-0.25\sin(z_2)$
	$-0.8\sin(z_1)$	$0.49z_2 + 0.96z_1^2$
R {9, 10}	$0.86z_2$	$0.64z_1^2 z_2$
	$-0.96z_1$	-0.17
PTAT {7, 10}	$0.93z_2$	$-1.14z_2$
	$-0.89z_1$	$0.73z_1$
pAE {5, 8}	z_2	$z_2 - \sin(0.13z_1 + 0.19z_2)$
	$-0.9z_1$	$-0.8z_1$

Table 8: **Selection of Identified Equations (Reaction-Diffusion System).** For each variant, the RHS of the best and worst detected equations are listed. The LHS of each system of equations correspond to $\dot{z}_1$, $\dot{z}_2$ for each sub-row.

the Non-Linear Pendulum, even the best pAE equation shows no resemblance to the ground truth equation or an obvious approximation thereof. In the other two systems, the pAE produces more sensible equations. However, there are still unexpected terms, such as the $-\sin(0.13z_1 + 0.19z_2)$ term in the pAE8 for the Reaction-Diffusion System.

It is important to note that this rather superficial assessment completely ignores the aforementioned freedom of choice of coordinates. One could argue that for a thorough analysis of the equations, attempts to apply a set of very simple transformations should be made as in [3] to test whether the resulting equations correspond to the ground truth not exactly, but *up to* an affine transformation or permutation, for example.

For verification purposes, this may be sensible. However, the decision of where to draw the line can be considered subjective, or vary from one application to the other. Therefore, the only unbiased way of presenting the results is through the unembellished equations resulting from the method which, moreover, one would be confronted with in new, undiscovered systems.

Statistics In addition to showing typical examples of equations resulting from particularly successful and unsuccessful models, one can make observations about the distribution of all equations and collect statistics about the presence of features across the tested variants. The most important statistics are presented in Table 9 in which I distinguish between statistics with and without non-genuine coefficients (bracketed values) if any are present.

Looking at the number of terms per model (*N Terms*), the original, verified and replicated models achieve similar numbers, although the selected original models have the least terms in all systems. With the exception of the Non-Linear Pendulum, the PTAT variant has roughly the same amount or even slightly less terms on average than the other variants. The biggest reduction in the number of terms can be considered the one resulting from the PTAT equations in the Chaotic Lorenz System without non-genuine coefficients. This shows that in this case, the application of ST is counterproductive, as prematurely and overzealously turning off the 'wrong' coefficient significantly affects other coefficients which have to compensate the missing one.

On the other hand, for the Non-Linear Pendulum, the PTAT variant detected almost three times as many terms as the other variants, even when accounting for the non-genuine coefficients. In combination with the distribution of coefficients (Table 12 in Appendix D), the terms other than the $\sin(z)$ term could be interpreted as corrections to the dynamics captured by the dominant $\sin(z)$ term and attributed to overfitting. However, the presence of the z term could sometimes be just as valid as the $\sin(z)$ term if the dataset were to only contain simulations of small deflections of the pendulum or if the autoencoder absorbed the non-linearity, in which case the small-angle approximation would render the z term almost indistinguishable from the $\sin(z)$, leading to a mixture of both.

In regards to the PTAT estimate, the resulting statistics are remarkably similar to the actual PTAT statistics, which is probably due to the large patience parameter effectively corresponding to a considerable training time without any thresholding as was the case in the experiment whose data was used for the PTAT estimate.

As for the frequency of desirable features, the PTAT and pAE variants outperform the other variants and predominantly result in equations with desirable features, with the exception of the Non-Linear Pendulum.

(a) Non-Linear Pendulum

Variant	N Terms ↓	At least sin(z) [%] ↑	Other [%] ↓	None [%] ↓
GT	1	100	0	0
O {1-2}	1	50	50	0
V {1-10}	1.2	50	20	30
R {1-10}	1.3	30	30	40
PTAT {1-10}	3.7 (2.9)	100	0	0
pAE {1-10}	-	0	100	0
PTAT est. {1-10}	3.7 (3.2)	100	0	0

(b) Chaotic Lorenz System

Variant	N Terms ↓	up to $\mathcal{O}(2)$ [%] ↑	$\mathcal{O}(3)$ [%] ↓
GT	7	1	0
O {1-2}	8.5	100	0
V {1-10}	12 (11.7)	67 (73)	33 (27)
R {1-10}	12.2	73	27
PTAT {1-10}	12.1 (10.8)	70 (93)	30 (7)
pAE {1-10}	18.2	97	3
PTAT est. {1-10}	12.3 (11.7)	73	27

(c) Reaction-Diffusion System

Variant	N Terms ↓	Linear [%] ↑	Other [%] ↓	None [%] ↓
GT	2	100	0	0
O {1-2}	2	75	25	0
V {1-10}	2.1	55	45	0
R {1-10}	2.2	55	45	0
PTAT {1-10}	2	100	0	0
pAE {1-10}	~ 2.5	90	10	0
PTAT est. {1-10}	5.1 (1.4)	90	10 (0)	0 (10)

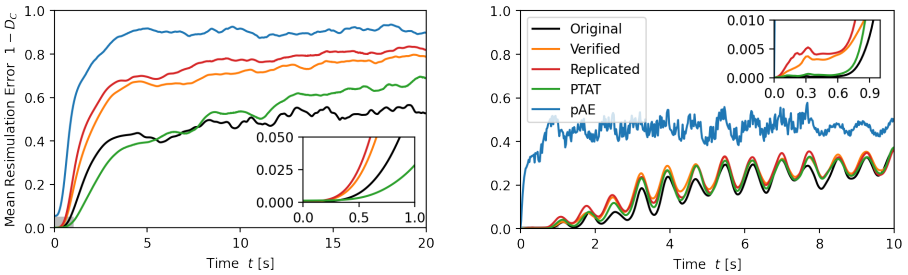
Table 9: **Equation Features and Statistics.** The *N Terms* column shows number of terms discovered by each model of a variant in all dimensions averaged over the number of models and is subject to minimization (↓), although values less than 1 would imply that some models had resulted in equations with no terms at all. The other columns show how often a feature in the equation could be identified, averaged over the number of models *and* the dimensions of the problem. Bracketed values correspond to the values excluding non-genuine coefficients. The *PTAT est.* row depicts the estimate conducted in Section 5.4.

6.3 Resimulation

The resimulation procedure tests for stability of the learned dynamics and robustness of the autoencoder. Figure 11 compares resimulation errors of each variant in the three example systems.

(a) Non-Linear Pendulum

(b) Lorenz System



(c) Reaction Diffusion

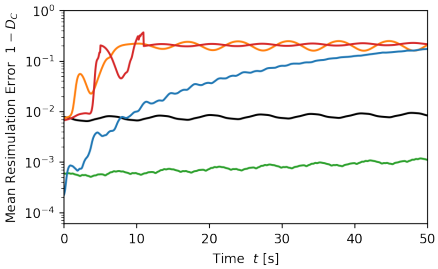


Figure 11: **Resimulation Errors.** Each figure shows the resimulation error $1 - D_C$ of the ground truth simulation and the resimulation over time, where D_C is the cosine similarity from Equation 15. The error is averaged over all models of a variant and all simulations in the test set, each with different random initial conditions.

In general, the verification (*orange*) and replication (*red*) models perform very similar and both worse on average than the two original (*black*) models selected by K. Champion et al. in [3] with the exception of the Chaotic Lorenz System. Here, most of the models perform very similar on big timescales and only the pAE (*blue*) models perform considerably worse than the rest. Also, the mean errors over time of all variants seem to oscillate in phase with each other, with the amplitude and shift towards higher errors increasing with time as expected. In the first second, the original and PTAT models perform slightly

better on average than the validation and replication models.

This can also be observed in the first resimulated second of the Non-Linear Pendulum where the PTAT models even have slightly smaller errors than the original models until about 5s, after which they switch positions. In the Reaction-Diffusion System, this trend continues and the average resimulation errors of the PTAT models are not only one order of magnitude smaller than the two selected original models on average but for the entire length of the resimulation (!).

Considering all three systems, the resimulation shows that the PTAT improvement results in models that perform as good or better *on average* as the two best selected original models which can be assumed to be picked from a distribution with its mean close to the mean verification and replication errors. This correlates well with the disappearence of exceptionally unusual equations in the PTAT models with respect to the verification and replication models, since in the PTAT variant, no coefficients are turned off by mistake, thus presumably allowing for the optimal equations to be found every time and eliminating outliers which would negatively affect the average error, specifically the average resimulation error over time.

The overall resimulation performance of the method still depends strongly on the system on which it is applied. While the Reaction-Diffusion System can be resimulated accurately over a very long time, the resimulations of the Non-Linear Pendulum and the Chaotic Lorenz System only predict the system accurately in the first second, after which they significantly diverge from the ground truth simulation.

6.4 PTAT Validation

In order to validate that the PTAT algorithm worked as intended, I compare the coefficient trends at the time of thresholding and their significance (Equations 17, 18) of the R models using ST (Figure 12a-12c, from Section 5.3) to the PTAT models (Figure 12d-12f).

Indeed, no coefficients with a trend corresponding to a projected rate of change of $|\Delta\xi| >= \frac{0.2}{100\,\text{epochs}}$ were turned off using the PTAT algorithm. For the Non-Linear Pendulum and the Reaction-Diffusion System, the majority of thresholds even occured below a trend corresponding to a rate of change of $|\Delta\xi| <= \frac{0.2}{5000\,\text{epochs}}$ (*green line*) which is well below the trend-threshold parameter $b = \frac{0.002}{\text{epoch}} = \frac{0.2}{100\,\text{epochs}}$ (*red line*), indicating that most coefficients had properly converged in the patience period and not been turned off by mistake. Even in

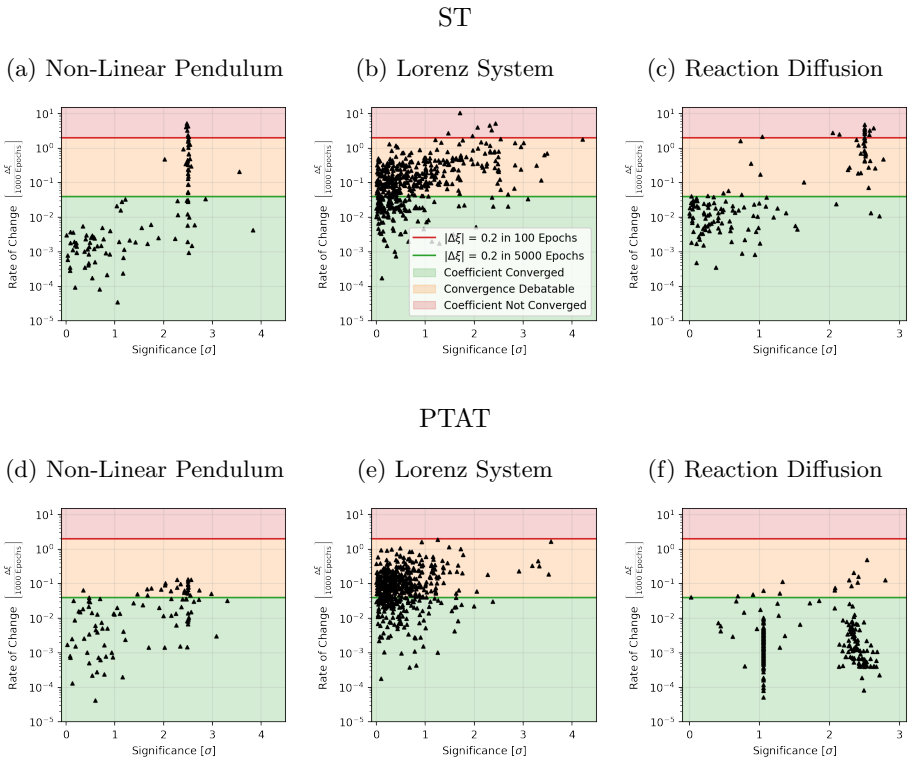


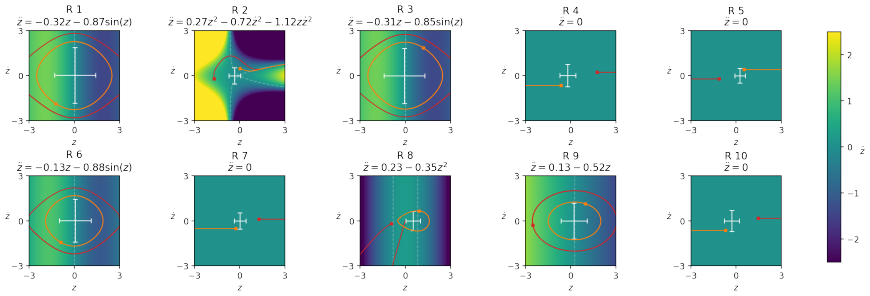
Figure 12: **Coefficient Trends with ST and PTAT.** The scatter plots show the projected rate of change and significance of the rate of change of coefficients ξ when they were turned off by the Sequential Thresholding (ST) or Patient Trend Aware Thresholding (PTAT) algorithm. The colored areas indicate an estimate of the plausibility that the coefficient was justifiably turned off. The trend was calculated using Equation 17 and $L = 5$.

the Chaotic Lorenz System where $b' = \frac{0.02}{\text{epoch}} = \frac{2}{100\,\text{epochs}}$, no coefficients above $\frac{b'}{10}$ had been turned off.

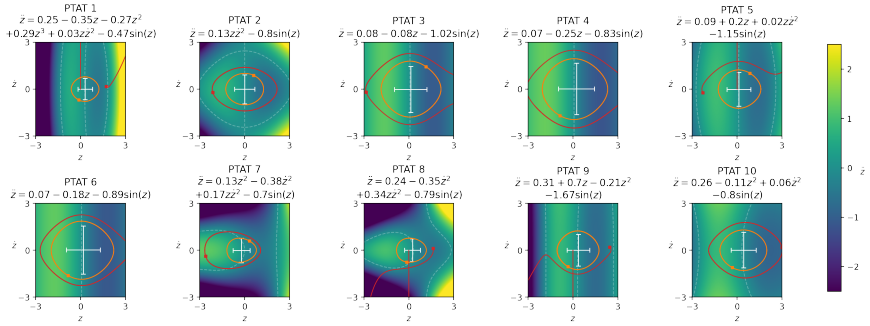
It is important to note again that the trend in Figure 12 does not correspond to the actual difference of subsequent coefficient values but the difference in the means of two subsequent segments of the coefficient history defined in Equation 17. Thus, the trend may differ from the momentary rate of change $|\delta\xi_{t'}| = |\xi_{t'} - \xi_{t'-1}|$ as seen by the PTAT algorithm and is hence not to be confused with it. Rather, the comparison of trends is intended to be an independent, intuitive measure of the quality of the two thresholding algorithms over a slightly wider timeframe.

6.5 Phase Space

(a) R Variant



(b) PTAT Variant



(c) pAE Variant

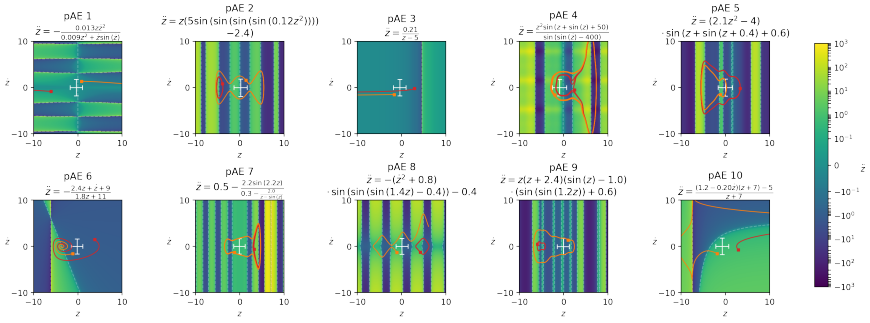


Figure 13: **Phase Space (Non-Linear Pendulum).** The equations of all models of the R, PTAT and pAE variants are shown visually as a function of z and $\dot{z}$. The white dashed lines correspond to the contours $\ddot{z} = 0$. The white error bars represent the standard deviations σ_z and $\sigma_{\dot{z}}$ of the encoded test set. The colored lines show the trajectories for two representative initial conditions (#0: orange, #14: red). The two initial conditions are each marked with a small square.

Lastly, I analyze the phase space of the pendulum models in detail for the R, PTAT and pAE variant. Figure 13 shows that the PTAT variant results in more models maintaining a stable oscillation than the replicated models, while four of the ten replicated R models diverge due to the equation being $\ddot{z} = 0$. Although all trajectories for the smaller initial condition (*orange*) are stable for the PTAT variant, only six of the ten trajectories are for the larger initial conditions (*red*). This instability often seems to coincide with $\ddot{z}$ flipping signs slightly too late or too early as indicated by the contours (*white dashed lines*).

As for the pAE variant, it becomes clear immediately that the transformation of the coordinates and time derivative is so complicated that the dynamics can only be described poorly even with the more complicated PySR equations. In some cases, the values of $\ddot{z}$ are multiple orders of magnitude higher than for the R and PTAT variant. Astonishingly, some of the representative trajectories seem to mark stable oscillations despite this complexity.

7 Discussion

In this thesis, I verified and replicated the original results of the SINDy-Autoencoder [3] by K. Champion et al. and expose the Sequential Thresholding algorithm used to promote extra sparsity in equations as a major cause of failure. I proposed a patient, trend aware thresholding algorithm and showed that it improves the mean dynamics and resimulation error and the consistency and reliability of the errors and identified equations.

However, the improved SINDy-Autoencoder still suffers from unexpected terms in some cases. Whether these exist by mistake or are the result of the freedom of choice of the coordinate system in which the equation is learned and formulated, is yet to be researched.

Further limitations include the requirement of good quality, noise-free data, precomputed derivatives as already discussed by K. Champion et al. in [3], and the explicit formulation of SINDy addressed in [16].

I recognize three additional limitations, the biggest one being that the transformation of the time derivative by chain rule is hardcoded. This approach is simple and fast but it immensely limits the topology of the autoencoder. Currently, only fully-connected feed-forward Neural Networks with a handful of activation functions are supported while the presented example systems corresponding to a series of images would probably benefit from more sophisticated

architectures such as Convolutional Neural Networks. Small tests on my part indicated that automatic differentiation as a method of circumventing the hard-coded transformation is in theory possible but not viable on the required scales due to memory limitations or slowness of the computation.

Second, there are currently many parameters that have to be known prior to the discovery of an equation of an unknown system. In particular, one has to specify the intrinsic dimensionality of the system and a set of suspected candidate terms. Also, the method only supports first and second order differential equations limited to a linear combination of non-linear terms. There are certainly a number of more complicated equations in physics which could not be properly identified with the current formulation. A rather simple example is given by Newton's law of gravitation containing the term $\frac{1}{r^2}$ [17], which one would have to specifically define as an extra term and possibly manually adapt the combination algorithm such that no duplicate terms would arise. A more general approach would rather require a set of basic operations which are combined to form an equation as in PySR [6].

Third, one still has to associate the discovered equation to the observations. Assuming the resulting equations are easily interpretable but given especially complicated measurement data, the task suddenly becomes interpreting the transformation carried out by the autoencoder and as such, the problem of interpretability only shifts to arbitrarily complicated transformations embodied in the autoencoder. This also relates to the discovery of emergent behaviour which brings uncertainty as to what the discovered equation actually describe and mean given their corresponding coordinates.

8 Outlook

Finally, I would like to address possible improvements for future work.

First of all, the pAE experiment leaves space for one improvement: Since after the the discovery of the equation in the latent coordinates of the pretrained autoencoder the equation only approximates the dynamics in these coordinates, a finetuning process may be appropriate, in which the coordinates would adapt under the constraint of the discovered equation. This could result in better dynamics errors but may not affect the stability of simulations as much.

It may also be advantageous to introduce further example systems and possibly apply the method to a real-world scenario.

In regards to generalizations, one may consider replacing the SINDy framework with a more proficient and flexible regression method or a non-symbolic physics-informed proxy used only for regularization of the latent coordinates during training of the Auroencoder, in which a symbolic regression method would later find the governing differential equations.

9 Additional Resources

The code for this thesis is publically available at GitLab: `https://gitlab.com/psaegert/sindy-autoencoders-improvements`.

Acknowledgements

I would like to thank Nathalie Soybelman for introducing me to PySR and sharing her experiences with me, which spared me time and effort getting familiar with its concepts and multitude of parameters, and greatly helped in the pAE experiment.

References

[1] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15 (Mar. 2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113. URL: https://doi.org/10.1073%5C%2Fpnas.1517384113.

[2] Miles Cranmer et al. "Discovering Symbolic Models from Deep Learning with Inductive Biases". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 17429–17442. URL: https://proceedings.neurips.cc/paper/2020/file/c9f2f917078bd2db12f23c3b413d9cba-Paper.pdf.

[3] Kathleen Champion et al. *Data-driven discovery of coordinates and governing equations*. 2019. DOI: 10.48550/ARXIV.1904.02107. URL: https://arxiv.org/abs/1904.02107.

[4] Tobias Pfaff et al. "Learning Mesh-Based Simulation with Graph Networks". In: (2020). DOI: 10.48550/ARXIV.2010.03409. URL: https://arxiv.org/abs/2010.03409.

[5] Siyuan Shen et al. *High-order Differentiable Autoencoder for Nonlinear Model Reduction*. 2021. DOI: 10.48550/ARXIV.2102.11026. URL: https://arxiv.org/abs/2102.11026.

[6] Miles Cranmer. *PySR: High-Performance Symbolic Regression in Python*. 2022. URL: https://github.com/MilesCranmer/pysr.

[7] Silviu-Marian Udrescu and Max Tegmark. "AI Feynman: a Physics-Inspired Method for Symbolic Regression". In: (2019). DOI: 10.48550/ARXIV.1905.11481. URL: https://arxiv.org/abs/1905.11481.

[8] Silviu-Marian Udrescu and Andrew Tan. *AI-Feynman*. 2021. URL: https://github.com/SJ001/AI-Feynman.

[9] Gert-Jan Both et al. "DeepMoD: Deep learning for model discovery in noisy data". In: *Journal of Computational Physics* 428 (Mar. 2021), p. 109985. DOI: 10.1016/j.jcp.2020.109985. URL: https://doi.org/10.1016%5C%2Fj.jcp.2020.109985.

[10] Linan Zhang and Hayden Schaeffer. *On the Convergence of the SINDy Algorithm*. 2018. DOI: 10.48550/ARXIV.1805.06445. URL: https://arxiv.org/abs/1805.06445.

[11] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[12] K. Champion. *Reaction Diffusion Simulation for Sindy Autoencoders*. 2019. URL: https://github.com/kpchamp/SindyAutoencoders/blob/master/rd%5C_solver/reaction_diffusion.m.

[13] K. Champion. *Sindy Autoencoders*. 2019. URL: https://github.com/kpchamp/SindyAutoencoders.

[14] Facebook, Inc. *PyTorch*. 2022. URL: https://pytorch.org/.

[15] G.J. Both R. Kusters. *Pytorch implementation of the DeepMoD algorithm*. 2020. URL: https://github.com/PhIMaL/DeePyMoD_torch.

[16] Kadierdan Kaheman, J. Nathan Kutz, and Steven L. Brunton. “SINDy-PI: a robust algorithm for parallel implicit sparse identification of nonlinear dynamics”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476.2242 (Oct. 2020), p. 20200279. DOI: 10.1098/rspa.2020.0279. URL: https://doi.org/10.1098%2Frspa.2020.0279.

[17] Douglas C. Giancoli. *Physik Lehr und Übungsbuch*. PEARSON DEUTSCHLAND GMBH, 2010. ISBN: 978-3-86894-023-7.

[18] Python Software Foundation. *Python*. 2021. URL: https://www.python.org/.

[19] Microsoft. *Jupyter Notebooks in VS Code*. 2021. URL: https://code.visualstudio.com/docs/datascience/jupyter-notebooks.

[20] Microsoft. *VS Code*. 2022. URL: https://code.visualstudio.com/.

[21] DeepL GmbH. *DeepL*. 2022. URL: https://www.deepl.com.

[22] Overleaf. *Overleaf*. 2022. URL: https://www.overleaf.com.

[23] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

A Training Parameters

For most of the experiments, I used the same training parameters as the original SINDy-Autoencoders [3]. All parameters, including deviations from the original parameters, are listed in Table 10.

As proposed by K. Champion et al., all autoencoders are initialized with random Xavier-distributed values [3]. The reconstruction loss weight was set to $\lambda_{\mathbf{x}} = 1$ for all experiments.

(a) Non-Linear Pendulum

Variant	Layers	`lr`	$\lambda_{\dot{\mathbf{x}}}$	$\lambda_{\dot{\mathbf{z}}}$	λ_1	E (+ref.)	B	N (train/val)
O	128,64,32	10^{-4}	$5 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	10^{-5}	5001+1001	1000	$5 \cdot 10^4 / 5 \cdot 10^3$
V	128,64,32	10^{-4}	$5 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	10^{-5}	5001+1001	1000	$5 \cdot 10^4 / 5 \cdot 10^3$
R	128,64,32	10^{-4}	$5 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	10^{-5}	5001+1001	1000	$5 \cdot 10^4 / 5 \cdot 10^3$
PTAT	128,64,32	10^{-4}	$5 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	10^{-5}	5001+1001	1000	$5 \cdot 10^4 / 5 \cdot 10^3$
pAE	128,64,32	10^{-4}	1	-	-	1500	1000	$5 \cdot 10^4 / 5 \cdot 10^3$

(b) Chaotic Lorenz System

Variant	Layers	`lr`	$\lambda_{\dot{\mathbf{x}}}$	$\lambda_{\dot{\mathbf{z}}}$	λ_1	E (+ref.)	B	N (train/val)
O	64,32	10^{-3}	10^{-4}	0	10^{-5}	5001+1001	1024	$256 \cdot 10^3 / 5 \cdot 10^3$
V	64,32	10^{-3}	10^{-4}	0	10^{-5}	5001+1001	1024	$256 \cdot 10^3 / 5 \cdot 10^3$
R	64,32	10^{-3}	10^{-4}	0	10^{-5}	5001+1001	1024	$256 \cdot 10^3 / 5 \cdot 10^3$
PTAT	64,32	10^{-3}	10^{-4}	0	10^{-5}	5001+1001	1024	$256 \cdot 10^3 / 5 \cdot 10^3$
pAE	64,32	10^{-4}	1	-	-	1500	1024	$256 \cdot 10^3 / 5 \cdot 10^3$

(c) Reaction-Diffusion System

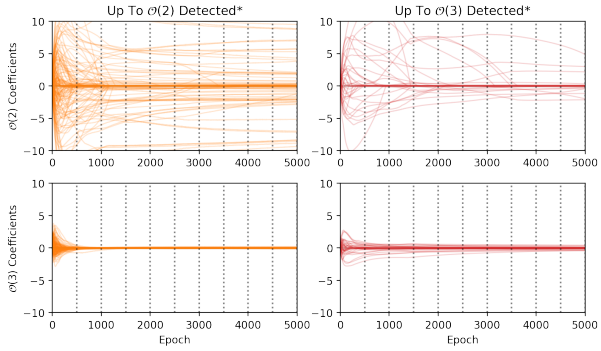
Variant	Layers	`lr`	$\lambda_{\dot{\mathbf{x}}}$	$\lambda_{\dot{\mathbf{z}}}$	λ_1	E (+ref.)	B	N (train/val)
O	256	10^{-3}	0.5	0.01	0.1	3001+1001	1000	7999/1000
V	256	10^{-3}	0.5	0.01	0.1	3001+1001	1000	7999/1000
R	256	10^{-3}	0.5	0.01	0.1	3001+1001	1000	7999/1000
PTAT	256	10^{-3}	0.5	0.01	0.1	9000+3000	1000	7999/1000
pAE	256	10^{-3}	1	-	-	1500	1000	7999/1000

Table 10: **Training Parameters** including the learning Rate `lr`, the loss weights $\lambda_{\dot{\mathbf{x}}}$, $\lambda_{\dot{\mathbf{z}}}$, λ_1 as in Section 3, the number of training epochs split into training with L_1 loss and without (Refinement, *+ref.*), the batch size B and the number of training and validation instances N.

B Additional Coefficient Histories

Similar to the coefficient histories of the Non-Linear Pendulum in Section 5.3, Figure 10, I present the ones for the Chaotic Lorenz System and the Reaction-Diffusion System in Figure 14 for the sake of completeness.

(a) Chaotic Lorenz System



(b) Reaction-Diffusion System

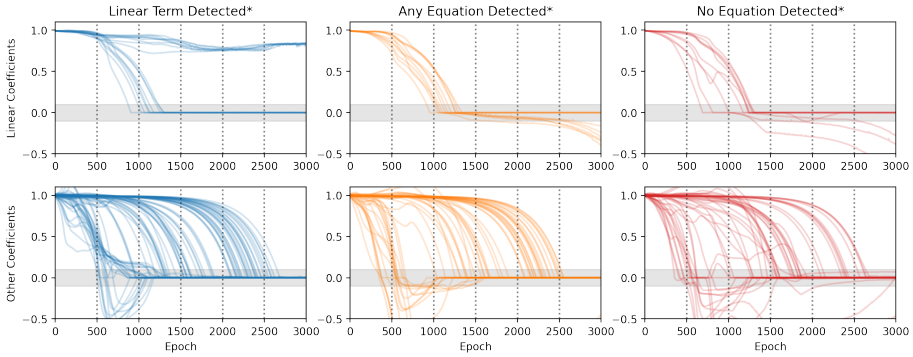


Figure 14: **Coefficient Histories without ST (Chaotic Lorenz System, Reaction Diffusion System).** All 10 modified replication models are categorized according to the quality of their identified equation (*columns*). The coefficient histories of all models in one category are each superimposed in two figures (*rows*), categorized into the desired coefficients and other coefficients. The x-axis is limited to training without Refinement epochs. The gray horizontal bar depicts the thresholding region, within which coefficients would have been turned off in thresholding epochs by the ST algorithm for the rest of the training. *For a term to be treated as 'detected', its coefficient had to be outside the thresholding range in every thresholding epoch (*vertical dashed lines*).

C Technical Details

C.1 Resources

All experiments and analysis were done on my own, local computer whose specifications are listed in Table 11.

Component/Module	Specification/Version
PyTorch	1.9.0+cu111
Tensorflow	2.5.0
Python	3.8.10
PySR	0.6.0
CUDA	11.4
cuDNN	8.1.0
NVIDIA Driver	470.103.01
OS	Ubuntu 20.04 LTS 64bit
Memory	32GB DDR4 2133MT/s
CPU	Intel® Core™ i7-6700K @ 4.60GHz (OC) × 8
GPU	NVIDIA GeForce GTX 980 Ti 6GB, GM200

Table 11: **List of Relevant Components and Specifications.**

C.2 Tools

I used Python [18] scripts and Jupyter Notebooks in VSCode [19] [20] to perform training and automated calculations for analysis. No AI-assisted code completion software was used. Occasionally, I used DeepL [21] to translate single words from the German language into English. This thesis was written with Overleaf [22]. All plots have been generated with Matplitlib [23] v3.4.2.

C.3 Cost

Excluding inefficiencies related to debugging and testing, I estimate the total training time for all models at

$$10 \cdot \Big[\underbrace{(65+40+220)\cdot 3}_{\text{V, R, PTAT}} + \underbrace{(15+5+20)+(15+30+5)}_{\text{pAE (AE + PySR)}}\Big]' \cdot \frac{1\,\text{h}}{60'} = \mathbf{177.5\,h}. \quad (19)$$

Assuming an average system load of 250W during training with the GPU and 120W during training of the PySR models without the GPU, I estimate the total energy consumption at **43.3 kWh**.

D List Of all identified Equations

For best transparency, I list all identified equations in this section. The coefficient values from the SINDy-Autoencoder are rounded to two decimal places. The parameters in the PySR equations are rounded to their significant digits.

Judging by the entirety of all equations for all systems (Tables 12-19) and accounting for the fact that the equations may be permuted in their dimensions, one can state that the verification and replication models produce very similar distributions of equations, further confirming the correctness of the reimplementation.

The PTAT variant produces more consistent equations in general, but has 'non-genuine' coefficients (*gray*) whose absolute values are smaller than the threshold (0.1) as a result of the training ending before the PTAT algorithm would have turned them off as explained in Section 5.4. In the case of the Reaction-Diffusion System, *all* equations discoverd by the PTAT variants describe a fully linear oscillation.

The equations identified by pAE variant are more complicated overall and share similarities with the ground truth equation only in the Reaction-Diffusion System.

D.1 Non-Linear Pendulum

Model	1	z	$\dot{z}$	z^2	$z\dot{z}$	$\dot{z}^2$	z^3	$z^2\dot{z}$	$z\dot{z}^2$	$\dot{z}^3$	$\sin(z)$	$\sin(\dot{z})$
GT	-	-	-	-	-	-	-	-	-	-	-1	-
O1	-	-	-	-	-	-	-	-	-	-	-0.99	-
O2	-	-0.52	-	-	-	-	-	-	-	-	-	-
V1	-0.15	-0.34	-	-	-	-	-	-	-	-	-0.82	-
V2	-	-0.44	-	-	-	-	-	-	-	-	-	-
V3	-	-	-	-	-	-	-	-	-	-	-	-
V4	-	-0.53	-	-	-	-	-	-	-	-	-	-
V5	-	-	-	-	-	-0.13	-	-	-	-	-0.78	-
V6	0.11	-	-	-	-	-	-	-	-	-	-1.01	-
V7	-	-	-	-	-	-	-	-	-	-	-	-
V8	-	-	-	-	-	-	-	-	-	-	-	-
V9	-	-	-	-	-	-	-	-	-	-	-1.05	-
V10	-	-0.15	-	-	-	-	-	-	-	-	-0.91	-
R1	-	-0.32	-	-	-	-	-	-	-	-	-0.87	-
R2	-	-	-	0.27	-	-0.72	-	-	-1.12	-	-	-
R3	-	-0.31	-	-	-	-	-	-	-	-	-0.85	-
R4	-	-	-	-	-	-	-	-	-	-	-	-
R5	-	-	-	-	-	-	-	-	-	-	-	-
R6	-	-0.13	-	-	-	-	-	-	-	-	-0.88	-
R7	-	-	-	-	-	-	-	-	-	-	-	-
R8	0.23	-	-	-0.35	-	-	-	-	-	-	-	-
R9	0.13	-0.52	-	-	-	-	-	-	-	-	-	-
R10	-	-	-	-	-	-	-	-	-	-	-	-
PTAT1	0.25	-0.35	-	-0.27	-	-	0.29	-	0.03	-	-0.47	-
PTAT2	-	-	-	-	-	-	-	-	0.13	-	-0.8	-
PTAT3	0.08	-0.08	-	-	-	-	-	-	-	-	-1.02	-
PTAT4	0.07	-0.25	-	-	-	-	-	-	-	-	-0.83	-
PTAT5	0.09	0.2	-	-	-	-	-	-	0.02	-	-1.15	-
PTAT6	0.07	-0.18	-	-	-	-	-	-	-	-	-0.89	-
PTAT7	-	-	-	0.13	-	-0.38	-	-	0.17	-	-0.7	-
PTAT8	0.24	-	-	-	-	-0.35	-	-	0.34	-	-0.79	-
PTAT9	0.31	0.7	-	-0.21	-	-	-	-	-	-	-1.67	-
PTAT10	0.26	-	-	-0.11	-	0.06	-	-	-	-	-0.8	-

Table 12: **All Identified Equations for the Non-Linear Pendulum, 1/2.**

Model	Best PySR-Equations
GT	$\ddot{z} = -\sin(z)$
pAE1	$\ddot{z} = -\frac{0.013 z \dot{z}^2}{0.009 z^2 + \dot{z} \sin(\dot{z})}$
pAE2	$\ddot{z} = z\left(5 \sin\left(\sin\left(\sin\left(0.12 z^2\right)\right)\right) - 2.4\right)$
pAE3	$\ddot{z} = \frac{0.21}{z - 5}$
pAE4	$\ddot{z} = \frac{z^2 \sin(z + \sin(z) + 50)}{\sin(\sin(\dot{z}) - 400)}$
pAE5	$\ddot{z} = \left(2.1 z^2 - 4\right) \sin(z + \sin(z + 0.4) + 0.6)$
pAE6	$\ddot{z} = -\frac{2.4 z + \dot{z} + 9}{1.8 z + 11}$
pAE7	$\ddot{z} = 0.5 - \frac{2.2 \sin(2.2 z)}{0.3 - \frac{2.0}{z - \sin(z)}}$
pAE8	$\ddot{z} = -\left(\dot{z}^2 + 0.8\right) \sin(\sin(\sin(1.4 z) - 0.4)) - 0.4$
pAE9	$\ddot{z} = z(z + 2.4)(\sin(z) - 1.0)(\sin(\sin(1.2 z)) + 0.6)$
pAE10	$\ddot{z} = \frac{(1.2 - 0.20\dot{z})(z + 7) - 5}{z + 7}$

Table 13: **All Identified Equations for the Non-Linear Pendulum, 2/2.**

D.2 Chaotic Lorenz System

Model	1	z_1	z_2	z_3	z_1^2	z_1z_2	z_1z_3	z_2^2	z_2z_3	z_3^2	z_1^3	$z_1^2z_2$	$z_1^2z_3$	$z_1z_2^2$	$z_1z_2z_3$	$z_1z_3^2$	z_2^3	$z_2^2z_3$	$z_2z_3^2$	z_3^3
GT	-	-10	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
O1	-	-10.04	-10.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
O2	-	-	-	5.68	-	-3.49	-	-	2.12	-	-	-	-	-	-	-	-	-	-	-
V1	31.5	-	-	-	-	2.94	-	-2.68	-	-	-0.28	-	-	-	-	-	-	-	-	-
V2	7.61	-2.91	-	-	-	-	-	-	3.13	-	-	-	-	-	-	-	-	-	-	-
V3	8.08	-4.16	4.69	5.65	-	-	-1.33	-	0.83	-	-0.06	-	-	-	-	-	-	-	-	-
V4	-	-1.11	-	-	-	-	-0.16	-	9.68	-	-	-	-	-	-	-	-	-	-	-
V5	-9.61	-7.63	-5.83	3.89	-	-1.2	-	-	1.04	-	-	-	-	-	-	-	-	-	-	-
V6	-	-10.77	-10.52	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V7	-	-	-13.72	-	-	-	-	-	-5.92	-	-	-	-	-	-	-	-	-	-0.04	-
V8	-6.55	-	-	-7.94	-	-	-	5.42	7.13	-	-	-	-	-	-	-	-	-0.33	-0.25	-
V9	8.87	-7.02	-	-6.69	-	0.6	0.84	-0.27	-0.82	-	-	-	-	-	-	-	-	-	-	-
V10	-	-2.94	-	-	-	-	-	-6.47	-	5.92	-	-	-	-	-	-	-	-	-	-
R1	1.43	-	-	-	-	-5.26	-	0.67	-	-	-	-	-	-	-	-	-	-	-0.85	-
R2	-	-	-	-	0.35	-	-	-1.32	-0.98	1.28	-	-	-	-	-	-	-	-	-	-
R3	7.11	-2.89	-	-	-	-	-	-	-3.08	-	-	-	-	-	-	-	-	-	-	-
R4	-	-9.98	-10.47	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R5	-	10.11	-	-	-0.26	-1.48	-	1.55	-	-	-	-	-	-	-	-0.42	-	-	-	0.38
R6	-	0.56	-6.23	-	-	-	3.37	-	2.19	-	-	-	-	-	-	-	-	-	-	-
R7	-	-15.79	-1.84	-	-	-2.22	1.14	0.39	1.7	-1.0	-	-	-	-	-	-	-	-	-	-
R8	-	-8.99	-	-5.52	-	-1.63	-1.95	2.33	-	-3.5	-	-	-	-	-	-	-	-	-	-
R9	-	-	-	-	-	0.14	-	-	-9.44	-	-	-	-	-	-	-	-	-	-	-
R10	-7.0	-2.68	-	-	-	-	-	-0.25	2.98	-	-	-	-	-	-	-	-	-	-	-
PTAT1	-	-8.02	-	-7.73	-	-	1.2	-	-1.71	-	-	-	-	-0.07	-	-	-	-	-	-
PTAT2	-7.43	-2.7	-	-	-	-	-	-	3.35	-	-	-	-	-	-	-	-	-	-	-
PTAT3	-	-10.04	-10.83	-	-	-	-0.04	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT4	-	-0.99	-	-	-	0.08	-	-	9.05	-	-	-	-	-	-	-	-	-	-	-
PTAT5	-8.32	-11.22	-	6.04	-	-	1.41	-	1.46	-	-	-	-	-0.09	-	-	-	-	-	-
PTAT6	-	-9.92	-10.56	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT7	-	-10.09	10.51	-	-	-	-0.06	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT8	13.25	-11.74	-	7.4	-	-	-1.31	-	0.44	-	-0.04	-	-	-	-	-	-	-	-	-
PTAT9	-6.98	-1.1	-	-	-	-	-0.19	6.18	-	-2.89	-	-	-	-	-	-	-	-	-	-
PTAT10	7.47	-3.67	-5.58	4.76	0.28	1.21	-	-	-0.8	-	-	-	-	-	-	-0.08	-	-	-	-
pAE1	-	-	-	-	-	-24	30	-16	11	10	-	-	-	-	-	-	-	-	-	-
pAE2	-	-6	-3	-1	-	-	-	-6	-	-	-	-	-	-	-	-	-	-	-	-
pAE3	-3	-	12	-10	-	-	-	-	-28	40	-	-	-	-	-	-	-	-	-	-
pAE4	-0.29	-18	-	18	-	-13	-	-	13	-	-	-	-	-	-	-	-	-	-	-
pAE5	2.1	-8	1.3	2.0	-	-13	-	13	-20	-	-	-	-	-	-	-	-	-	-	-
pAE6	2.0	-1.3	-5	-6	-	5	5	-9	-14	-5	-	-	-	-	-	-	-	-	-	-
pAE7	-	-8	-	8	12	-40	-12	-	43	-	-	-	-	-	-	-	-	-	-	-
pAE8	6	7	9	-7	-	11	8	-	-22	-17	-	-	-	-	-	-	-	-	-	-
pAE9	-5	1	8	-15	-	-	40	-	40	-	-	-	-	-	-	-	-	-	-	-
pAE10	0.6	-15	15	15	-	-	-40	-	40	40	-	-	-	-	-	-	-	-	-	-

Table 14: **All Identified Equations for the Chaotic Lorenz System, 1/3.** Each row corresponds to the equation $\dot{z}_1 = f(\mathbf{z})$.

Model	1	z_1	z_2	z_3	z_1^2	z_1z_2	z_1z_3	z_2^2	z_2z_3	z_3^2	z_1^3	$z_1^2z_2$	$z_1^2z_3$	$z_1z_2^2$	$z_1z_2z_3$	$z_1z_3^2$	z_2^3	$z_2^2z_3$	$z_2z_3^2$	z_3^3
C	-	28	-1	-	-	-	-1	-	-	-	-	-	-	-	-	-	-	-	-	-
O1	-	-	-0.94	-	-	-	9.55	-	-	-	-	-	-	-	-	-	-	-	-	-
O2	-10.27	-	-2.72	-	2.63	-	-	-	-	-1.01	-	-	-	-	-	-	-	-	-	-
V1	-	-	-	-	-	-	-11.06	-	-	-	-	-	0.43	-	-	-0.55	-	-	-	-
V2	-	-	-10.6	-10.59	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V3	-	-	-8.03	-8.08	-	-	-2.02	-	1.37	-	-	-	-	-	-	-	-	-	-	-
V4	-	-10.2	-9.8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V5	-	-	-	-	6.24	-0.13	-	-	-	-4.12	-	-	-	-	-	-	-	-	-	-
V6	-	-	-	-	-	-	-9.35	-	-	-	-	-	-	-	-	-	-	-	-	-
V7	-	-16.23	-9.9	-	-	-	-	-	-	-	-	-	-	0.13	-	-	-	-	-	-
V8	18.86	9.87	-8.93	-	-	-	-	-	0.23	-	-	-	-	-0.31	-	-	-	-	-	-
V9	-	3.47	-1.87	7.1	-0.36	0.64	0.98	-	-0.91	-	-	-	-	-	-	-	-	-	-	-
V10	-	7.55	-2.22	4.87	-	0.91	-0.74	-	-	-	-	-	-	-	-	-	-	-	-	-
R1	-	-	-	-9.18	-1.35	-	-0.96	-	1.6	-	-	-	-	-0.16	-	-	-	-	-	-
R2	-	-	12.39	-	-	-	-	-	-	-	-	-0.51	-0.63	-	-	-	-	-	-	-
R3	-	-	-	-	-	-	9.09	-	-	-	-	-	-	-	-	-	-	-	-	-
R4	-	-	-1.0	-	-	-	-9.72	-	-	-	-	-	-	-	-	-	-	-	-	-
R5	-	-	-12.64	-9.5	1.42	-1.46	-	-	-	-	-	-	-	-	-	-	-	-	0.29	0.35
R6	-	-	-11.43	-	-	-	-4.99	-	-3.29	-	-	-	-	-	-	-	-	-	-	-
R7	8.32	-	-	-	1.05	-0.88	2.0	-	-	-2.59	-	-	-	-	-	-	-	-	-	-
R8	-	3.82	2.91	-	0.83	-1.17	-	-	2.21	-2.59	-	-	-	-	-	-	-	-	-	-
R9	6.58	-	-2.73	-	-	-	2.87	-	-	0.19	-	-	-	-	-	-	-	-	-	-
R10	-	-	-11.02	11.12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT1	8.08	4.73	-3.84	5.51	-	-	0.76	-	-1.2	-	-	-	-	-	-	-	-0.09	-	-	-
PTAT2	-	-	-0.96	-	-	-	-9.6	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT3	-	-	-0.79	-	-	-	8.73	-	-0.09	-	-	-	-	-	-	-	-	-	-	-
PTAT4	7.61	-	-2.67	-	-	-	-3.2	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT5	-	-6.39	-	6.4	-	-	-1.36	-	-1.48	-	-	-0.06	-	-	-	-	-	-	-	-
PTAT6	-	-	-1.04	-	-	-	-9.23	-	0.06	-	-	-	-	-	-	-	-	-	-	-
PTAT7	-	-	-0.7	-	-	-	-9.28	-	-0.06	-	-	-	-	-	-	-	-	-	-	-
PTAT8	-	-	-	-5.66	-	-	-3.41	-	1.33	-	-	-	-	-	-	-	-	-	-	-
PTAT9	2.47	5.6	-	-5.87	-	-1.91	-1.3	0.15	-	-	-	-	-	-	-	-	-0.08	-	-	-
PTAT10	3.89	-	-	-	-7.61	-	-	-	0.16	3.21	-	-0.11	-	-	-	-	-	-	-	-
pAE1	4	11	-3	11	-14	-14	-28	-	-14	-18	-	-	-	-	-	-	-	-	-	-
pAE2	0.8	-7	-	-	40	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pAE3	1.5	-2.2	-	-4	-	12	-14	-	25	-28	-	-	-	-	-	-	-	-	-	-
pAE4	-	-	-	0.5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pAE5	0.9	-	-24	-16	-	-	-	-	-40	-24	-	-	-	-	-	-	-	-	-	-
pAE6	-2.6	-	-7	8	-	-	-21	-	-	24	-	-	-	-	-	-	-	-	-	-
pAE7	-1.2	-	-	-5	12	-12	-	-	12	-12	-	-	-	-	-	-	-	-	-	-
pAE8	-1.5	-2.9	-	-2.9	16	-	-16	-	-	-	-	-	-	-	-	-	-	-	-	-
pAE9	-4	-22	-	-32	-23	-	-60	-	-	-	-	-	-	-	-	-	-	-	-	-
pAE10	3	-17	4	-	-40	-55	-	-14	-	-	-	-	-	-	-	-	-	-	-	-

Table 15: **All Identified Equations for the Chaotic Lorenz System, 2/3.** Each row corresponds to the equation $\dot{z}_2 = f(\mathbf{z})$.

Model	1	z_1	z_2	z_3	z_1^2	z_1z_2	z_1z_3	z_2^2	z_2z_3	z_3^2	z_1^3	$z_1^2z_2$	$z_1^2z_3$	$z_1z_2^2$	$z_1z_2z_3$	$z_1z_3^2$	z_2^3	$z_2^2z_3$	$z_2z_3^2$	z_3^3
C	-	-	-	-2.7	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
O1	-7.13	-	-	-2.68	-	-3.12	-	-	-	-	-	-	-	-	-	-	-	-	-	-
O2	-	-	-	-10.94	-	-5.59	-	-	3.44	-	-	-	-	-	-	-	-	-	-	-
V1	-3.44	15.71	-	-2.74	-	-	-	-	-	-	-	-	-	-1.35	-	-	-	-	-	-
V2	-	-	-	-	-	-8.74	0.18	-	-	-	-	-	-	-	-	-	-	-	-	-
V3	-4.63	-	-	-1.05	7.92	-	-	-3.26	0.02	-	-	-	-	-	-	-	-	-	-	-
V4	-7.34	-	-	-2.65	-	-3.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V5	-	3.41	7.1	-5.59	-	-1.61	-	-	1.28	-0.18	-	-	-	-	-	-	-	-	-	-
V6	7.35	-	-	-2.88	-	2.81	-	-	-	-	-	-	-	-	-	-	-	-	-	-
V7	-	-	-	-3.24	-	2.53	-	-	-	-	-	-	-	-	-0.19	-	-	-	-	-
V8	-	-0.64	-	-	-	-5.0	0.9	-	-	0.29	-	-	-	-	-	-	-	-0.2	-	-
V9	-	-	-	-4.34	-5.99	-	-	5.33	-	-	-	-	-	-	-	-	-	-	-	-
V10	-7.88	-8.09	-	-8.27	-	0.52	-0.73	-	-0.54	-	-	-	-	-	-	-	-	-	-	-
R1	-	0.54	-	-	-	-	-	-	4.48	0.26	-	0.93	-	-	0.22	-	-0.44	-	-	-
R2	-	-	-	-20.35	-	-1.74	-	-	-	-	-	-	0.41	-	-	-	-	-	-0.12	-
R3	-	-	11.17	-10.75	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R4	7.28	-	-	-2.67	-	2.96	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R5	-	-	-6.55	-8.36	-1.1	-	1.44	1.01	1.15	-	-	-	-	-	-	-	-	-	-	-
R6	9.69	-	-	-2.71	-2.45	-	-	1.1	-	-	-	-	-	-	-	-	-	-	-	-
R7	-	-	6.19	2.32	-0.22	-2.54	0.71	0.43	2.18	-0.62	-	-	-	-	-	-	-	-	-	-
R8	10.86	-4.63	-	-6.92	0.64	-	1.05	-1.38	1.69	-	-	-	-	-	-	-	-	-	-	-
R9	-	-11.25	-	-10.82	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R10	-	-	-	-	-	-9.19	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT1	-4.74	-	-	-0.92	-3.38	-	-	7.92	-	-	-	-	-	-	-	-	-	-	-	-
PTAT2	-	-	10.43	-9.98	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT3	-7.64	-	-	-2.7	-	-3.05	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT4	-	10.6	-	-9.94	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT5	-5.11	-	-	-1.61	-4.7	-	-	5.46	-	-	-	-	-	-	-	-	-	-	-	-
PTAT6	7.55	-	-	-2.66	-	2.69	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT7	-7.39	-	-	-2.73	-	3.03	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT8	1.79	-	-	-1.32	4.74	5.16	-	-2.7	-	-	-	-	-	-	-	-	-	-	-	-
PTAT9	-5.21	5.19	-	-11.59	-	2.75	1.83	-	-	-	-	-	-	-	-	-	-	-	-	-
PTAT10	-	-	7.87	-8.89	-	1.63	-	-	-1.25	-	-	-	-	-	-	-	-	-	-	-
pAE1	1.9	-30	-18	1.5	-	50	-24	24	-12	-	-	-	-	-	-	-	-	-	-	-
pAE2	-	9	10	-9	10	-19	-	10	-	-10	-	-	-	-	-	-	-	-	-	-
pAE3	-	-	9	-6	-	14	-7	-	14	-7	-	-	-	-	-	-	-	-	-	-
pAE4	1.5	-5	-	-1	-19	-	-19	-	-	-	-	-	-	-	-	-	-	-	-	-
pAE5	-0.4	-6	6	-	-	-11	-11	11	-11	-	-	-	-	-	-	-	-	-	-	-
pAE6	-2.0	10	-5	-11	-	24	-	-	-	28	-	-	-	-	-	-	-	-	-	-
pAE7	-2.7	-5	-	-5	20	-40	-20	-	40	-	-	-	-	-	-	-	-	-	-	-
pAE8	-	4	-4	-6	-16	16	30	-	-	-	-	-	-	-	-	-	-	-	-	-
pAE9	-0.18	-1.9	1.9	-1.9	-1.8	-1.8	-15	-	-17	17	-	-	-	15	-15	-	15	-	-	-
pAE10	-	-5	-	-	12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 16: **All Identified Equations for the Chaotic Lorenz System, 3/3.** Each row corresponds to the equation $\dot{z}_3 = f(\mathbf{z})$.

D.3 Reaction-Diffusion-System

Model	1	z_1	z_2	z_1^2	$z_1 z_2$	z_2^2	z_1^3	$z_1^2 z_2$	$z_1 z_2^2$	z_2^3	$\sin(z_1)$	$\sin(z_2)$
GT	-	-	-1	-	-	-	-	-	-	-	-	-
O1	-	-	-0.85	-	-	-	-	-	-	-	-	-
O2	-	-	-	-	-	-	-	-	-	-	-	-1.1
V1	-	-	-	-	-	-	-	-	-	-	-0.25	-
V2	-	-	-	-	-	-	-	-	-	-	-	-0.94
V3	-	-	0.96	-	-	-	-	-	-	-	-	-
V4	-	-	0.93	-	-	-	-	-	-	-	-	-
V5	-	-	0.92	-	-	-	-	-	-	-	-	-
V6	-	-	-0.88	-	-	-	-	-	-	-	-	-
V7	-	-	0.88	-	-	-	-	-	-	-	-	-
V8	-	-	0.97	-	-	-	-	-	-	-	-	-
V9	-	-	-	-	-	-	-	-	-	-	-	-0.99
V10	-	-	1.03	-	-	-	-	-	-	-	-	-
R1	-	-	-	-	-	-	-	-	-	-	-	-1.05
R2	-	-	0.92	-	-	-	-	-	-	-	-	-
R3	-	-	0.87	-	-	-	-	-	-	-	-	-
R4	-	-	-	-	-	-	-	-	-	-	-	-1.13
R5	-	-	-	-	-	0.23	-	-	-	-	-	-
R6	-0.44	-	-	-	-	0.9	-	-	-	-	-	-
R7	-	-	-	-	-	-	-	-	-	-	-	-0.97
R8	-	-	-	-	-	-	-	-	-	-	-	-1.05
R9	-	-	0.86	-	-	-	-	-	-	-	-	-
R10	-	-	-	-	-	-	-	0.64	-	-	-	-
PTAT1	-	-	-0.93	-	-	-	-	-	-	-	-	-
PTAT2	-	-	-0.91	-	-	-	-	-	-	-	-	-
PTAT3	-	-	0.91	-	-	-	-	-	-	-	-	-
PTAT4	-	-	-0.84	-	-	-	-	-	-	-	-	-
PTAT5	-	-	0.93	-	-	-	-	-	-	-	-	-
PTAT6	-	-	0.88	-	-	-	-	-	-	-	-	-
PTAT7	-	-	0.93	-	-	-	-	-	-	-	-	-
PTAT8	-	-	-0.93	-	-	-	-	-	-	-	-	-
PTAT9	-	-	-1.07	-	-	-	-	-	-	-	-	-
PTAT10	-	-	-1.14	-	-	-	-	-	-	-	-	-

Table 17: **All Identified Equations for the Reaction-Diffusion System, 1/3.** Each row corresponds to the equation $\dot{z}_1 = f(\mathbf{z})$.

Model	1	z_1	z_2	z_1^2	$z_1 z_2$	z_2^2	z_1^3	$z_1^2 z_2$	$z_1 z_2^2$	z_2^3	$\sin(z_1)$	$\sin(z_2)$
GT	-	1	-	-	-	-	-	-	-	-	-	-
O1	-	0.97	-	-	-	-	-	-	-	-	-	-
O2	-	0.82	-	-	-	-	-	-	-	-	-	-
V1	-	-	0.49	0.96	-	-	-	-	-	-	-	-
V2	-	0.92	-	-	-	-	-	-	-	-	-	-
V3	-	-	-	-	-	-	-	-	-	-	-0.93	-
V4	-	-	-	-	-	-	-	-	-	-	-0.98	-
V5	-0.11	-	-	-	-	-	-	-	-	-	-	-
V6	-	0.94	-	-	-	-	-	-	-	-	-	-
V7	-	-	-	-	-	-	-	-	-	-	-0.99	-
V8	-	-	-	-	-	-	-	-	-	-	-0.94	-
V9	-	0.92	-	-	-	-	-	-	-	-	-	-
V10	-	-0.8	-	-	-	-	-	-	-	-	-	-
R1	-	0.91	-	-	-	-	-	-	-	-	-	-
R2	-	-0.9	-	-	-	-	-	-	-	-	-	-
R3	-	-	-	-	-	-	-	-	-	-	-1.04	-
R4	-	0.77	-	-	-	-	-	-	-	-	-	-
R5	-	1.45	-	-	-	-	-	-	-	-	-	-
R6	-	0.66	-	-	-	-	-	-	-	-	-	-
R7	-	0.98	-	-	-	-	-	-	-	-	-	-
R8	-	0.95	-	-	-	-	-	-	-	-	-	-
R9	-	-0.96	-	-	-	-	-	-	-	-	-	-
R10	-0.17	-	-	-	-	-	-	-	-	-	-	-
PTAT1	-	0.89	-	-	-	-	-	-	-	-	-	-
PTAT2	-	0.91	-	-	-	-	-	-	-	-	-	-
PTAT3	-	-0.91	-	-	-	-	-	-	-	-	-	-
PTAT4	-	0.99	-	-	-	-	-	-	-	-	-	-
PTAT5	-	-0.89	-	-	-	-	-	-	-	-	-	-
PTAT6	-	-0.94	-	-	-	-	-	-	-	-	-	-
PTAT7	-	-0.89	-	-	-	-	-	-	-	-	-	-
PTAT8	-	0.89	-	-	-	-	-	-	-	-	-	-
PTAT9	-	0.77	-	-	-	-	-	-	-	-	-	-
PTAT10	-	0.73	-	-	-	-	-	-	-	-	-	-

Table 18: **All Identified Equations for the Reaction-Diffusion System, 2/3.** Each row corresponds to the equation $\dot{z}_2 = f(\mathbf{z})$.

Model	Best PySR-Equations
GT	$\dot{z}_1 = z_2$ $\dot{z}_2 = -z_1$
pAE1	$\dot{z}_1 = 0.8z_2$ $\dot{z}_2 = -1.0z_1$
pAE2	$\dot{z}_1 = 0.9z_2$ $\dot{z}_2 = -0.9z_1$
pAE3	$\dot{z}_1 = -0.9z_2$ $\dot{z}_2 = 0.9z_1$
pAE4	$\dot{z}_1 = -0.9z_2$ $\dot{z}_2 = 0.9z_1$
pAE5	$\dot{z}_1 = 1.0z_2$ $\dot{z}_2 = -0.9z_1$
pAE6	$\dot{z}_1 = z_2 + 0.3$ $\dot{z}_2 = -0.9z_1$
pAE7	$\dot{z}_1 = 0.9z_2$ $\dot{z}_2 = -0.9z_1$
pAE8	$\dot{z}_1 = z_2 - \sin(0.13z_1 + 0.19z_2)$ $\dot{z}_2 = -0.8z_1$
pAE9	$\dot{z}_1 = 0.9z_2$ $\dot{z}_2 = -0.9z_1$
pAE10	$\dot{z}_1 = -0.9z_2$ $\dot{z}_2 = 0.9z_1$

Table 19: **All Identified Equations for the Reaction-Diffusion System, 3/3.**

Declaration

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ort, Datum	Paul Stefan Saegert