Professur für
Thermofluiddynamik

MASTER THESIS

Prediction of Chaotic Systems with Physics - Enhanced Machine Learning Models

Autor:
Mathias Lesjak

Matrikel-No:
03660998

Betreuer:
Nguyen Anh Khoa Doan, Ph.D.
Prof. Wolfgang Polifke, Ph. D.

July 29, 2020

Technische Universität München
Fakultät für Maschinenwesen
Professur für Thermofluiddynamik
Prof. Wolfgang Polifke, Ph. D.

Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst zu haben. Ich habe keine anderen Quellen und Hilfsmittel als die angegebenen verwendet.

München, 29.07.2020	
Ort, Datum	Mathias Lesjak

Acknowledgement

I would like to thank my thesis advisor Nguyen Anh Khoa Doan, Ph.D. of the thermo-fluid dynamics group at the technical university of Munich. His door was always open whenever I had a question about my research. I would particularly like to thank him for his support during the difficult Corona period. The regular meetings motivated me and helped me stay on topic. He consistently allowed this thesis to be my own work, but steered me in the right the direction whenever he thought I needed it.

Of course, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Author:
Mathias Lesjak

Abstract

Various processes in nature can be modelled as dynamical systems. These systems range from climate and ocean circulation to cells and organisms, from non-linear optics to turbulent and reactive flows. Many systems involve high intrinsic dimensionality and complex dynamics extending over multiple spatio-temporal scales. Furthermore, extreme events may appear. Extreme events are characterized by rare transitions that are several standard deviations away from the mean value of the state variables. Regarding the before mentioned dynamical systems, examples of extreme events are tsunamis, tornadoes, volcanoes, floodings and droughts, earthquakes etc. A lot of effort is put into understanding these systems and different forecasting approaches are used. The three main methods are large scale simulations, dimension reduction techniques and multiple data driven forecasting techniques. Successful methods involve the highly nonlinear energy transfer between modes.
Large scale simulations are based on the complete description of the system through governing equations. These equations may be computationally difficult to solve and make these methods more expensive. In order to reduce complexity, models can be used, but they themselves can have a certain model error.
Classical dimension reduction methods are based on projection. If the underlying set of equations possesses high intrinsic dimensionality, problems often arise as the truncated degrees of freedom are essential for an effective description of the system.
In this context, machine learning methods can help. Some studies have proven that recurrent neural networks are able to learn the dynamics of chaotic systems and are therefore universal approximators of dynamical systems. However, due to the intrinsic property of chaos, that small deviations get amplified exponentially, prediction time is limited.
The idea of this work is to combine a projection method with a data driven method to improve accuracy and to extend the prediction time. We use a Proper Orthogonal Decomposition as projection method and combine it with an Echo State (ESN). The key concept is that the recurrent neuronal network assists the reduced order model which lost some information due to the reduction. We introduce three hybrid methods, assess their performance of the Lorenz system, the Platt system, the Charney-DeVore system and the Kuramoto Sivashinsky (KS) equation and compare the results with the pure machine learning method. We start with the Lorenz system, which is a well known low-dimensional chaotic system and continue to increase complexity and dimensionality and end with the KS equation which is a high-dimensional chaotic system in the discretized version. The first hybrid method uses the ESN to predict the truncated dynamics and reconstructs the solution in the reduced space. The

second hybrid method uses the ESN to predict the truncated dynamics as well, however, the solution is reconstructed in full solution space. In the third hybrid method, the ESN not only predicts the next time step of the solution but is also able to let in information from the ROM. We apply a parameter search routine to find good parameters for each method, test the prediction performance for different reservoir sizes as well as investigate the behavior of the ROM and the methods for different levels of information loss.
The results show that hybrid method 1 and 2 can only be used successfully in certain areas. Hybrid method 1 demands accurate ROMs and the reconstruction of the solution in reduced space is too restrictive with low-dimensional systems. Hybrid method 2 tries to correct this and reconstructs the solution in full space. However, this method does not produce consistent results. Hybrid method 3 is superior to the pure data-driven method in all cases. The ability to decide between ROM and ESN information is the key to success.

Contents

1 Introduction

1.1 Motivation and State of the Art

Dynamical systems and especially chaotic ones appear virtually everywhere in nature and science. The fields range from climate and ocean circulation to biology with cell and organism evolution, or even fluid mechanics with turbulent and reactive flows and epidemiology. Due to these various applications, chaotic systems have long been the subject of scientific research and understanding them and being able to predict them is important. However, this is difficult as they may show complex dynamical features like intermittency and/or extreme events. Intermittency is the random switching amongst qualitatively random kinds of oscillations. Extreme events are characterized by rare transitions that are several standard deviations away from the mean value of the state variables. In addition, the high dimensionality of the systems and their chaotic nature make their forecast considerably more challenging. This chaotic behavior is often known as the "Butterfly effect" which means that small deviations at the beginning may have strong impact on the latter behavior of the system. In a mathematical sense, this means that deviations grow exponentially with time in chaotic systems.
Due to the complex nature of these systems, many attempts to simplify them and to forecast their behaviors have been made. Order reduction methods like the proper orthogonal decomposition (POD) or the strongly connected singular value decomposition (SVD) have been used for a long time to try to educe their most important dynamics. POD was originally introduced by Lumley [15] to identify coherent structures in turbulent flows. Since then, the number of citations and publications using POD has increased exponentially [22]. In parallel, several similar methods have also appeared in other scientific disciplines like the principal component analysis method, Hotelling transform, Karhunen-Loève decomposition or empirical orthogonal functions in meteorology. POD can either be used to postprocess data or to generate low dimensional models by projecting the governing equations (Galerkin projection) onto subspaces spanned by the POD base functions. Not all base functions have to be used to represent the dynamical evolution of the system due to the optimal convergence in terms of kinetic energy, therefore allowing for a model order reduction. Also, all previously mentioned methods are based on linear decomposition and hence are approximations. In this context, machine learning is a promising new approach as they represent a form of nonlinear identification and decomposition. Indeed, some studies have shown that Recurrent neural networks (RNN) are universal approximations of dynamical systems [18], [19] , [20], [25] and [1].
RNNs have been known for a long time. However, only recently, thanks to the increase in computational power and the availability of data, they have become increasingly important and used. The first recurrent neural network was based on David Rumelhart's article "Learn-

ing representations by back-propagating errors" [23]. Classical recurrent neural networks are known to be difficult to train especially for problems with long-term dependencies. The iterative nature can cause problems while calculating the gradient of the loss function, such as a gradient explosion or vanishing due to the nonlinear behavior of the system. To remedy this, Hochreiter et al. [9] introduced the Long short-term memory RNN (LSTM). These neural networks compensate for the training problem by introducing memory cells. Memory cells are self-recurrent neurons with additional input, forget and output gates [9]. LSTM became especially successful in speech recognition and pattern recognition.

More related to the focus of this thesis, Wan et al. [25] used LSTM in combination with a low order model to forecast the evolution of chaotic systems and predicted the occurrence of extreme events. They showed the advantages of this hybrid method compared to purely data-driven methods or those relying solely on reduced order models. They used the LSTM to estimate the complementary dynamics of the nonlinear Galerkin projection of the equation system. The complementary dynamics describes the out-of-plane portion that is neglected in the flat Galerkin method. First, they considered a chaotic intermittent low-order atmospheric model, the Charney-DeVore (CDV) equations. They chose a fixed set of parameters which seemed best to show chaotic intermittent transitions and used the POD method to reduce the system of governing equations with the method of snapshots to calculate a set of base functions. Only the first 5 modes were used to construct this basis. They observed that the long-term predictions of the data-assisted model were significantly better than the purely data-driven and the reduced order model alone as measured by the anomaly correlation coefficient (ACC). The ACC measures the correlation between anomalies of forecasts and those of the truth with respect to a reference level [6]. Secondly, they investigated intermittent bursts of dissipation in Kolmogorov flow. The data-assisted approach yields the most accurate results as well. In a similar study, Vlachas et al. [24] proposed a LSTM-based method for the prediction in the reduced order space of dynamical systems. They coupled their LSTM network with a mean stochastic model (MSM) and created a hybrid LSTM-MSM method. The LSTM is trained with time series data and predicts the high dimensional dynamics using the short-term history of the reduced order variables. They showed the ability of their method to accurately predict short terms and capture the long term behavior of the Lorenz 96 system, the Kuramoto -Sivashinsky system and at a prototypical climate model. In addition, they compared their method against state-of-the art methods like: MSM, GPR and mixed GPR-MSM models. In all chaotic systems, the LSTM method without MSM extension performed better in short-term predictions than the state-of-the-art methods. However, with an increasing number of predictions, the error accumulated. The LSTM-MSM method, which should alleviate this effect, performed worse for short terms but did not diverge for long terms.

Finally, a last approach combining neural networks with knowledge-based models was proposed by Pathak et al. [19] to build a hybrid forecasting scheme for chaotic processes. They tested their scheme on a low-dimensional and high-dimensional chaotic system with promising results. They focused on the "reservoir computing" machine learning technique, which has been previously used to predict low-dimensional systems and more recently to predict

large spatiotemporal systems. They tested their approach at the low dimensional Lorenz system and the high dimensional, spatiotemporal chaotic Kuramoto-Sivashinsky system. They showed the superior performance of the hybrid model against the pure knowledge based or pure machine learning approach for both systems. Their hybrid predictor showed accurate results although the two component systems are flawed so that they do not show good results on their own.

1.2 Objective of the Thesis

Given the state of the art provided above, it can be seen that various methods to combine knowledge-based reduced order models (ROM) with machine learning algorithms have been proposed with growing levels of success. Therefore, in this thesis, we will test and assess these methods. More specifically, we focus on the promising results for ESN and try to develop hybrid models for the prediction of chaotic systems. We start with the development of hybrid methods which combine projection based reduced order models and machine learning models. The different hybrid approaches either try to use the ESN to compensate for the loss of information in the ROM or to find the best possible balance between information from the ML model and the ROM. In this thesis, POD will be used to develop these ROMs.

The ultimate objective is to assess the hybrid methods performance on more and more complex chaotic systems and provide a comparison with pure data-driven methods such as an ESN alone. We aim for a maximal validity time, the validity time being defined as the time where the norm of the difference between the reference solution and the predicted solution exceeds a defined threshold. This study of different methods applied to different systems will contribute to a deeper understanding of the combined use of ML with knowledge-based models for the forecasting of chaotic systems.

1.3 Structure of the Thesis

In order to successfully work through the stated objective, we study different chaotic systems, starting with low-dimensional ones and working towards the high-dimensional complex ones. These are presented in Chapter 2. Next, in Chapter 3, we discuss the necessary fundamentals to generate solutions of the several chaotic systems. These fundamentals include finite differences, Runge Kutta methods and exponential time differencing. The generated data is considered the reference data and all machine learning methods will be trained using this data.
Chapter 4 deals with the fundamentals of reduced order modeling. POD is introduced to create an optimal basis set and Galerkin projection is presented next, where the governing equations of the various systems are projected onto the new basis set. The procedure is demonstrated for illustrating purposes on the low-dimensional Lorenz system and the same procedure using matrix vector notation for high-dimensional systems is shown for the Kuramoto-

Sivashinsky equation.
Chapter 5 introduces the ML basics, the ESN and the parameter search algorithm. We describe how to create, train, use and optimize an ESN. Chapter 5 thus provides the ML portion for the creation of the hybrid methods.
After all the basics have been introduced, we start with the development of the hybrid methods in Chapter 6. Three different hybrid methods with increasing level of complexity are presented.
Chapter 7 presents the results of the parameter search and a reservoir size study for each method with the best parameters. The results are presented in the following order: data-only for all systems, for each system the ROM alone and the results from all hybrid methods, at the end plots combining all results for direct comparison are presented.
Finally, the results are discussed in Chapter 8, the work is summarized in Chapter 9 and an outlook on future topics is given.

2 Chaotic Systems

2.1 Ordinary Differential Equation Systems

Ordinary differential equations (ODEs) contain functions dependent on one independent variable and their derivatives. In the context of dynamical systems, the time t is the independent variable. In this thesis, three ODE systems will be studied: the Lorenz, the Platt and the Charney-DeVore systems. They are presented next.

2.1.1 Lorenz System

The Lorenz system was proposed by Edward Norton Lorenz in 1962 [13]. The equations were invented in order to predict the states in the earth's atmosphere. They are idealized fluid mechanical equations describing mainly convection problems. The system reads as follows:

$$\dot{x_1} = -\sigma x_1 + \sigma x_2 \tag{2.1a}$$

$$\dot{x_2} = \rho x_1 - x_2 - x_1 x_3 \tag{2.1b}$$

$$\dot{x_3} = -\beta x_3 + x_1 x_2 \tag{2.1c}$$

where σ is the Prandtl number, $\beta = \frac{4}{1+a^2}$ is a measure for the cell geometry where a is the ratio of the convection cell's height and width and the parameter ρ is the relative Rayleigh number. This number is the ratio between the Rayleigh number and the critical Rayleigh number and compares lifting and breaking forces (viscous forces and/or gravitation).

The Lorenz system has for a certain choice of parameters a strange attractor that is also known as the Lorenz attractor. This attractor is a set of points which the solution approaches when proceeding in time. The standard parameters are: $\rho = 28$, $\sigma = 10$ and $\beta = \frac{8}{3}$ [13]. With these parameters, the solution oscillates irregularly and approaches the Lorenz attractor as depicted in Fig. 2.1 and Fig. 2.2.

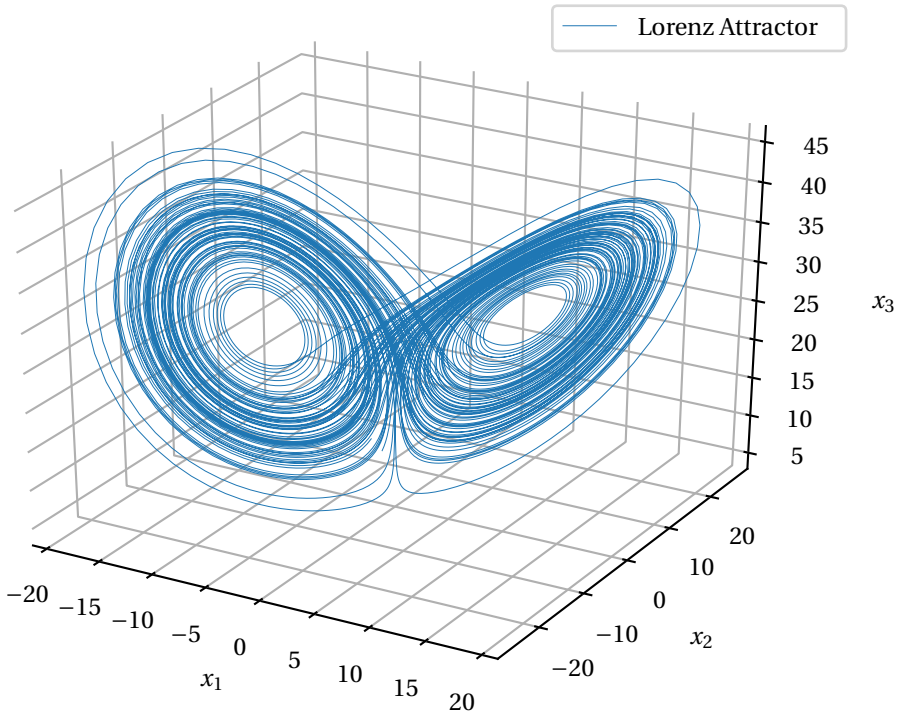

Figure 2.1: Time evolution of the Lorenz system in phase space.

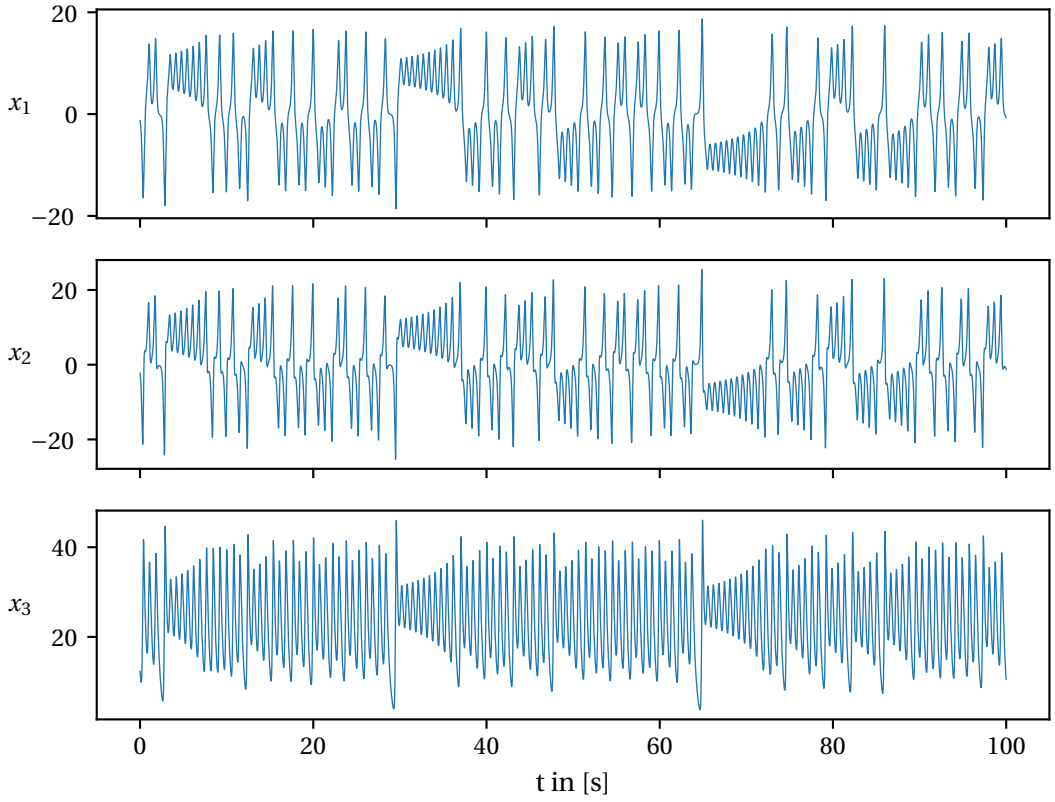

Figure 2.2: Time evolution of all three variables of the Lorenz system for 100 seconds.

2.1.2 Platt System

Platt et al. introduced the Platt system [21] which exhibits the so called "on-off intermittency". Initially observed in the field of fluid mechanics as transition between a laminar and turbulent flow, intermittency is the random switching amongst qualitatively random kinds of oscillations and more specifically on-off intermittency is the sudden change of a variable from long periods of stasis to bursts of large variations [21].
The Platt system reads:

$$\dot{x}_1 = x_2 \tag{2.2a}$$
$$\dot{x}_2 = -x_1^3 - 2x_1x_3 + x_1x_5 - \mu_{01}x_2 \tag{2.2b}$$
$$\dot{x}_3 = x_4 \tag{2.2c}$$
$$\dot{x}_4 = -x_3^3 - \nu_{01}x_1^2 + x_5x_3 - \nu_{02}x_4 \tag{2.2d}$$
$$\dot{x}_5 = \nu_{03}x_5 - \nu_{04}x_1^2 - \nu_{05}\left(x_3^2 - 1\right) \tag{2.2e}$$

For a deeper understanding of the on-off intermittency mechanism, the Platt system can be separated into two parts. A K-dimensional hyperplane $X = (x_1, x_2, ..., x_k)$ and the remainder $Y = (x_{k+1}, ..., x_N)$. The system's Eq. (2.2) can therefore be rewritten with $K = 2$ and $N = 5$:

$$\dot{X} = F\left(X, \mu(t)\right) \tag{2.3}$$
$$\dot{Y}(t) = G(X, Y, \nu_0) \underbrace{=}_{\text{if skew prod.}} \hat{G}(Y, \nu_0) \tag{2.4}$$

ν_0 is an additional parameter and $\mu(t)$ is:

$$\mu(t) = M\left(\mu_0, Y(t)\right) \tag{2.5}$$

where μ_0 is another parameter. An important characteristic of Eq. (2.4) is that the hyperplane $X = 0$ is invariant under the described evolution. In addition, for some parameter combinations (e.g.: $\nu_{01} = \nu_{04} = 0$) in Eq. (2.2), there is no X dependence in the evolution described by $\dot{Y}$. This is called a skew product structure.
The key is to consider Y as a modifier of the parameters that control the stability of the rest state X [21]. In our case, the rest state is $X = 0$ and $\mu(t)$ controls the rest state's stability. X becomes unstable for values larger than the critical value ν_c and stable for smaller values. We obtain the desired on-off intermittency behavior if μ spends suitable time intervals in those regions [21].
We choose the parameters in Eq. (2.2) according to [21] and obtain a system without skew product structure. However, these parameters yield the best on-off intermittency behavior. They are:

$$\begin{bmatrix} \mu_{01} \\ \nu_{01} \\ \nu_{02} \\ \nu_{03} \\ \nu_{04} \\ \nu_{05} \end{bmatrix} = \begin{bmatrix} 1.815 \\ 1.0 \\ 1.815 \\ 0.44 \\ 2.86 \\ 2.86 \end{bmatrix} \tag{2.6}$$

The Platt system simplifies for $x_1 = x_2 = 0$ to the Lorenz system. Due to the extremely different dynamics, this system represents a particular challenge for neural networks and serves as a limiting test case.
Using the Runge Kutta 4 scheme with a step size of $0.01s$, we obtain the solution as depicted in Fig. 2.3 without initial transients. The intermittent bursts can be seen in the evolution of variables x_1 and x_2.

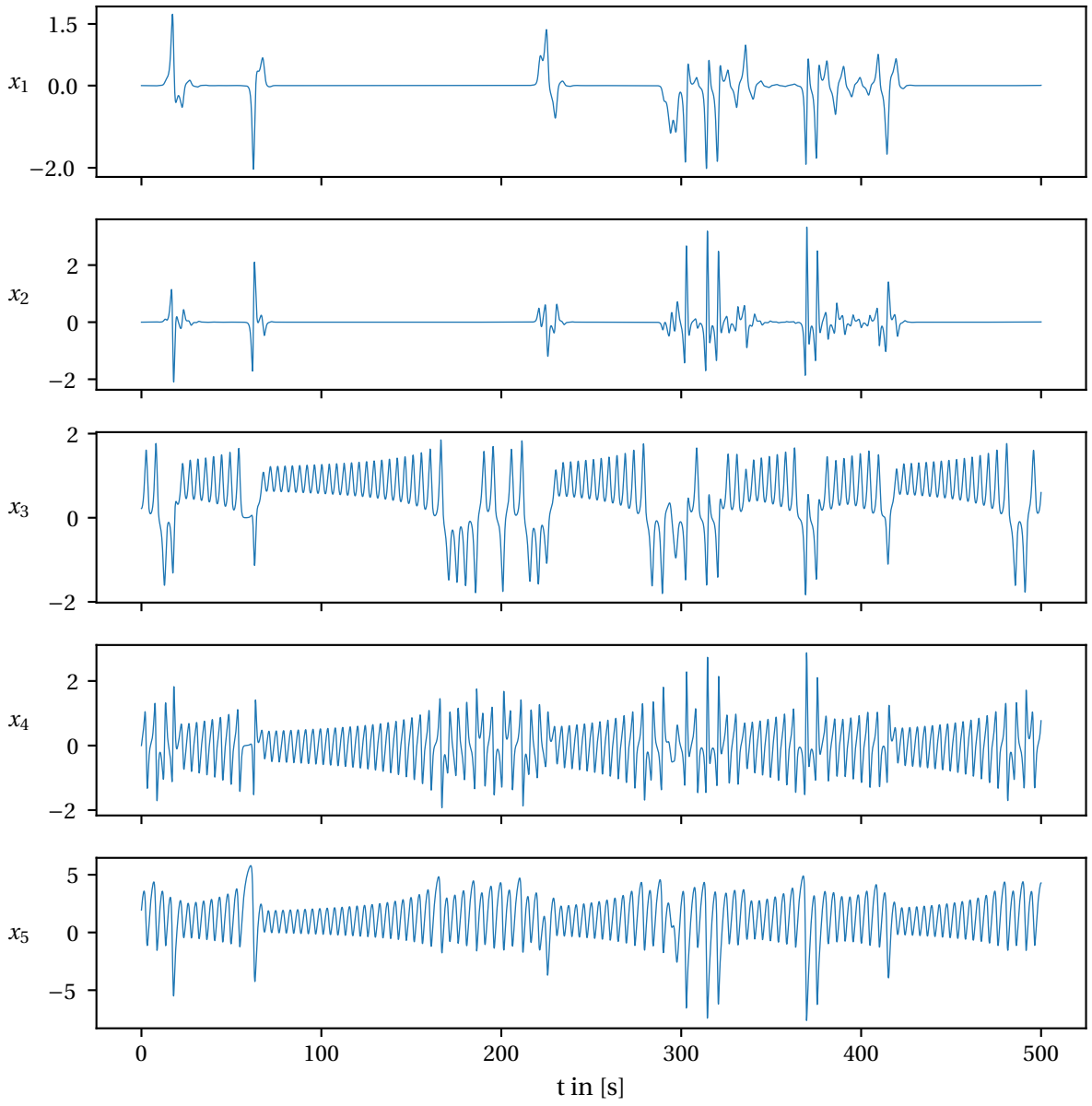

Figure 2.3: Time evolution of all variables of the Platt system for 500 seconds.

2.1.3 Charney-DeVore System

We consider a chaotic low-order atmospheric model inspired by [3], [4]. This model represents the truncated Charney-DeVore (CDV) equations which were originally developed to model barotropic flow in a β-plane with orography. The β plane approach incorporates Coriolis forces that vary linearly in space and orography that accounts for the surface's height profile. The governing equations as introduced by [25] are scaled slightly differently and read:

$$\dot{x}_1 = \gamma_1^* x_3 - C\left(x_1 - x_1^*\right) \tag{2.7a}$$
$$\dot{x}_2 = -\left(\alpha_1 x_1 - \beta_1\right) x_3 - C x_2 - \delta_1 x_4 x_6 \tag{2.7b}$$
$$\dot{x}_3 = \left(\alpha_1 x_1 - \beta_1\right) x_2 - \gamma_1 x_1 - C x_3 + \delta_1 x_1 x_5 \tag{2.7c}$$
$$\dot{x}_4 = \gamma_2^* x_6 - C\left(x_4 - x_4^*\right) + \epsilon\left(x_2 x_6 - x_3 x_5\right) \tag{2.7d}$$
$$\dot{x}_5 = -\left(\alpha_2 x_1 - \beta_2\right) x_6 - C x_5 - \delta_2 x_4 x_3 \tag{2.7e}$$
$$\dot{x}_6 = \left(\alpha_2 x_1 - \beta_2\right) x_5 - \gamma_2 x_4 - C x_6 + \delta_2 x_4 x_2 \tag{2.7f}$$

where the coefficients are defined as:

$$\alpha_m = \frac{8\sqrt{2}m^2\left(b^2+m^2-1\right)}{\pi\left(4m^2-1\right)\left(b^2+m^2\right)} \qquad \beta_m = \frac{\beta b^2}{b^2+m^2} \tag{2.8a}$$
$$\delta_m = \frac{64\sqrt{2}\left(b^2-m^2+1\right)}{15\pi\left(b^2+m^2\right)} \qquad \gamma_m^* = \gamma\frac{4\sqrt{2}mb}{\pi\left(4m^2-1\right)} \tag{2.8b}$$
$$\epsilon = \frac{16\sqrt{2}}{5\pi} \qquad \gamma_m = \gamma\frac{4\sqrt{2}m^3 b}{\pi\left(4m^2-1\right)\left(b^2+m^2\right)} \tag{2.8c}$$

for $m \in 1,2$. The remaining constants are:

$$\begin{bmatrix} x_1^* \\ x_4^* \\ C \\ \beta \\ \gamma \\ b \end{bmatrix} = \begin{bmatrix} 0.95 \\ -9.76095 \\ 0.1 \\ 1.25 \\ 0.2 \\ 0.5 \end{bmatrix} \tag{2.9}$$

Wan et al. [25] found that these parameters values are particularly suitable for the evaluation of new modeling and prediction methods as the system shows chaotic intermittent transitions as can be seen in Fig. 2.4. The solution depicted in Fig. 2.4 is calculated using the Runge Kutta 4 scheme with a time step size of 0.1. The initial transients are skipped and an interval of $2000s$ is plotted where transitions between two states can be observed approximately every $700s$.

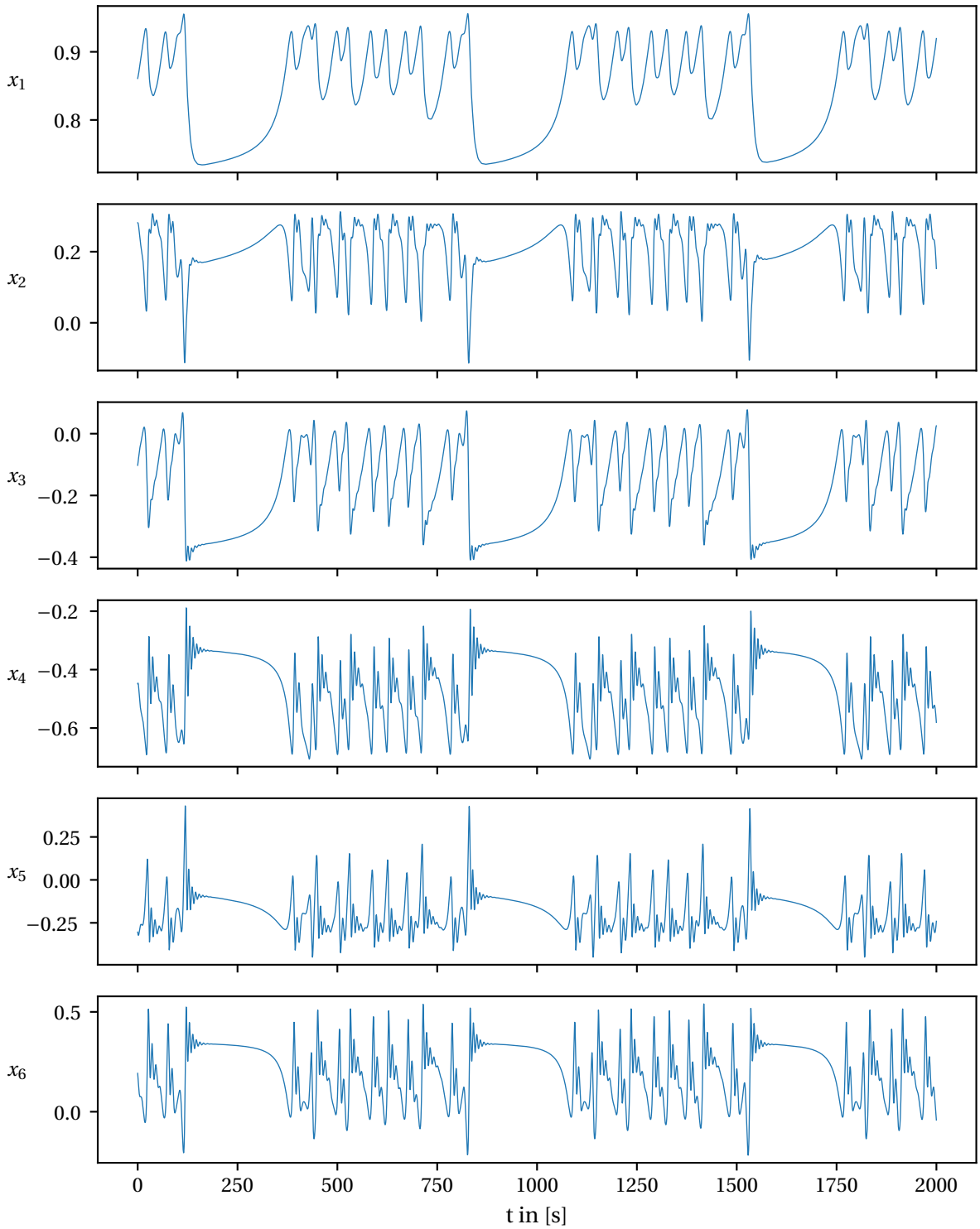

Figure 2.4: Time evolution of all variables of the CDV system for 2000 seconds.

2.1.4 Lyapunov Exponent Calculation

Many nonlinear mathematical or physical systems exhibit chaotic behavior. One of the most important quantitative measures for chaos is the Lyapunov exponent [7]. The Lyapunov exponents describe the average rate of separation of nearby trajectories [7]. Multidimensional problems show a Lyapunov spectrum with one Lyapunov exponent for each direction. We explain here how to compute the Lyapunov spectrum using the so-called "Standard Method" with periodic Gram-Schmidt normalization.
We consider an autonomous dynamical system:

$$\frac{\mathrm{d}\underline{x}}{\mathrm{d}t} = \underline{f}\left(\underline{x}(t)\right) \tag{2.10}$$

In order to find an equation for the evolution of small deviations $\delta\underline{x}$, we develop the Taylor series along a solution of Eq. (2.10).

$$\underline{f}\left(\underline{x}+\delta\underline{x}\right) = \underline{f}\left(\underline{x}\right) + \underline{\underline{\mathbf{Df}}}\Big|_{\underline{x}} \cdot \delta\underline{x} + \mathcal{O}\left(\left\|\delta\underline{x}\right\|^2\right) \tag{2.11}$$

$\underline{\underline{\mathbf{Df}}}\Big|_{\underline{x}}$ is the Jacobian matrix evaluated along a solution $\underline{x}$ of system Eq. (2.10). We combine Eq. (2.10) with Eq. (2.11) and obtain:

$$\underline{f}\left(\underline{x}+\delta\underline{x}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\underline{x}+\delta\underline{x}\right) = \frac{\mathrm{d}\underline{x}}{\mathrm{d}t} + \delta\dot{\underline{x}} = \underline{f}\left(\underline{x}\right) + \delta\dot{\underline{x}} \tag{2.12}$$

Combining Eq. (2.11) with Eq. (2.12) and discarding higher order terms yields the desired equation for the evolution of a deviation $\delta\underline{x}$ along the solution $\underline{x}$:

$$\frac{\mathrm{d}\delta\underline{x}}{\mathrm{d}t} = \underline{\underline{\mathbf{Df}}}\Big|_{\underline{x}} \cdot \delta\underline{x} \tag{2.13}$$

This equation holds exactly only for infinitesimal small deviations, however, by considering the limit $\delta \to 0$, one can identify $\delta\underline{x}$ as an element $\underline{\xi}$ of the tangent space at $\underline{x}$ and get an exact equation for $\underline{\xi}$ [7]:

$$\frac{\mathrm{d}\underline{\xi}}{\mathrm{d}t} = \underline{\underline{\mathbf{Df}}}\Big|_{\underline{x}} \cdot \underline{\xi} \tag{2.14}$$

We may choose arbitrary $\underline{\xi}_0$ but it is convenient to set $\left\|\underline{\xi}_0\right\| = 1$ to get the growth at a later time t as $\left\|\underline{\xi}(t)\right\|$ [7]. For chaotic systems, the average growth rate is expected to be exponential with the Lyapunov exponent λ.

$$\left\|\underline{\xi}(t)\right\| \approx e^{\lambda t} \tag{2.15}$$

Solving for the unknown exponent λ yields an expression for the exponent:

$$\lambda_i = \frac{log\left\|\underline{\xi}_i(t)\right\|}{t} \tag{2.16}$$

λ_i is the Lyapunov exponent in the i^{th} dimension corresponding to $\underline{\xi}_i$.
We explain the complete algorithm to simultaneously compute the Lyapunov spectrum according to Fig. 2.5 and Fig. 2.6 in the following:

1. Create n-dimensional unit sphere $\underline{\mathbf{U}}$ (for instance $n = 3$ for Lorenz System):

$$\underline{\mathbf{U}} = \left[\underline{\xi}_1, \quad \underline{\xi}_2, \quad \ldots, \quad \underline{\xi}_n\right] \tag{2.17}$$

2. Evolve the unit sphere in time according to:

$$\dot{\underline{\mathbf{U}}} = \underline{\mathbf{Df}}\Big|_{\underline{x}} \cdot \underline{\mathbf{U}} \tag{2.18}$$

   and stop after a few iterations. The unit sphere will have deformed to a n-dimensional ellipsoid. The deformation process occurs fast because the direction with the biggest Lyapunov exponent gets stretched rapidly.

3. Before the matrix $\underline{\mathbf{U}}$ gets ill-conditioned due to the fast deformation (some values becoming very large compared to the others), apply Gram-Schmidt orthonormalization, store the lengths of the deformed vectors $\left\|\underline{\xi}_i(t)\right\|$ for Lyapunov exponent calculation (Eq. (2.16)) and return to step 1. More details about Gram Schmidt and why it is legitimate to use it can be found in [7].

It is important to note that the Lyapunov exponents are the (time-dependent) eigenvalues of the Jacobian matrix. The trace of a matrix is invariant to a base transformation and therefore the trace of the Jacobian can be used as a check for the correctness of the calculation method.

$$\sum_i \lambda_i \stackrel{!}{=} \mathrm{tr}\left(\underline{\mathbf{Df}}\right) \tag{2.19}$$

For all ODE systems considered here, the trace is constant and reads for the various systems:

1. Lorenz System:

$$\mathrm{tr}\left(\underline{\mathbf{Df}}\right) = \mathrm{tr}\begin{bmatrix} -\sigma & \sigma & 0 \\ r - x_3 & -1 & x_1 \\ x_2 & x_1 & -b \end{bmatrix} = -\sigma - 1 - b \tag{2.20}$$

2. CDV System:

$$\mathrm{tr}\left(\underline{\mathbf{Df}}\right) = \mathrm{tr}\begin{bmatrix} -C & 0 & \gamma_1^* & 0 & 0 & 0 \\ \alpha_1 x_3 & -C & -\alpha_1 x_1 + \beta_1 & -\delta_1 x_6 & 0 & -\delta_1 x_4 \\ \alpha_1 x_2 - \gamma_1 & \alpha_1 x_1 - \beta_1 & -C & \delta_1 x_5 & \delta_1 x_4 & 0 \\ 0 & \epsilon x_6 & -\epsilon x_5 & -C & -\epsilon x_3 & \gamma_2^* + \epsilon x_2 \\ -\alpha_2 x_6 & 0 & -\delta_2 x_4 & -\delta_2 x_3 & -C & \beta_2 - \alpha_2 x_1 \\ \alpha_2 x_5 & \delta_2 x_4 & 0 & \gamma_2 + \delta_2 x_2 & \alpha_2 x_1 - \beta_2 & -C \end{bmatrix} = -6C \tag{2.21}$$

3. Platt System:

$$\mathrm{tr}\left(\underline{\mathbf{Df}}\right) = \mathrm{tr}\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -3x_1^2 - 2x_3 + x_5 & -\mu_{01} & -2x_1 & 0 & x_1 \\ 0 & 0 & 0 & 1 & 0 \\ -2\nu_{01}x_1 & 0 & -3x_3^2 + x_5 & -\nu_{02} & x_3 \\ -2\nu_{04}x_1 & 0 & -2\nu_{05}x_3 & 0 & -\nu_{03} \end{bmatrix} = -\mu_{01} - \nu_{02} - \nu_{03} \quad (2.22)$$

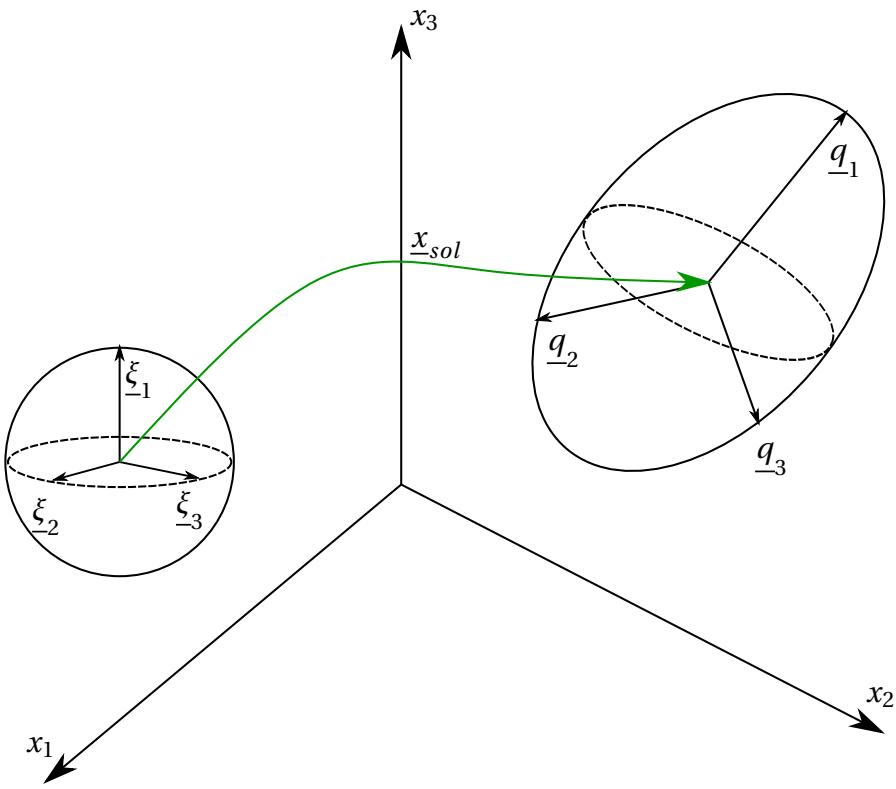

Figure 2.5: Deformation of a unit sphere U along a solution $\underline{x}_{sol}$.

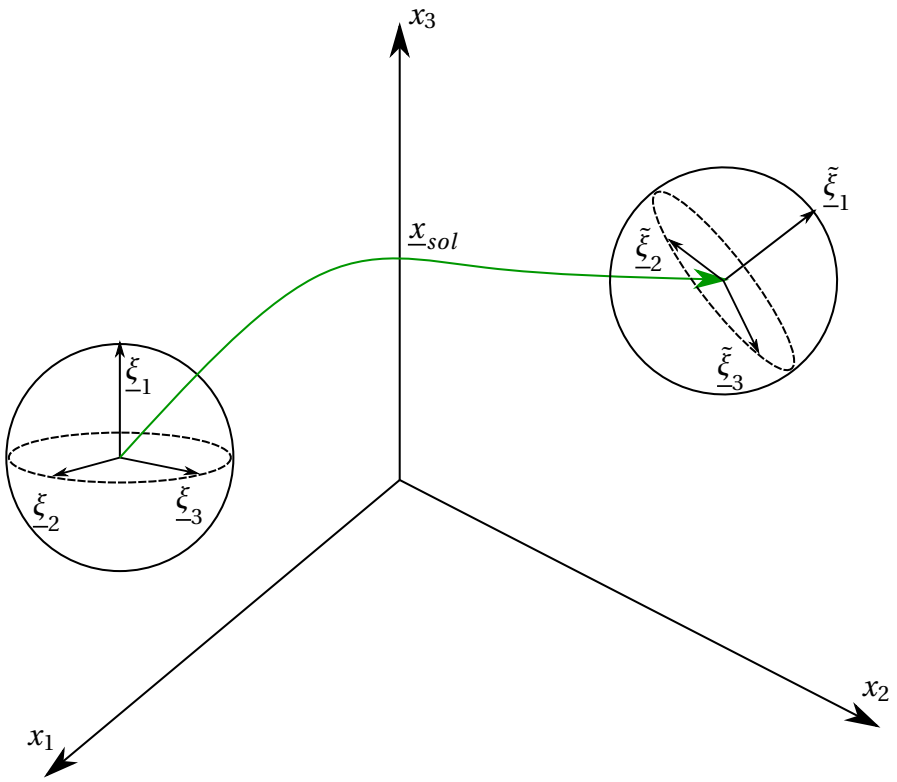

Figure 2.6: Orthogonalization of ellipsoid.

2.2 Partial Differential Equation Systems

2.2.1 Kuramoto-Sivashinsky Equation

The Kuramoto-Sivashinsky (KS) equation is a fourth-order nonlinear partial differential equation that is known for its chaotic behavior. We consider the one-dimensional KS equation on the spatial domain $0 \leq x \leq L$ and periodic boundary conditions:

$$\frac{\partial u}{\partial t} + \frac{\partial^4 u}{\partial x^4} + \frac{\partial^2 u}{\partial x^2} + u\frac{\partial u}{\partial x} = 0 \tag{2.23a}$$

$$u(t,0) = u(t,L) \tag{2.23b}$$

The initial condition reads:

$$u(t,x) = cos\left(2\pi\frac{x}{L}\right)\left[1 + sin\left(2\pi\frac{x}{L}\right)\right] \tag{2.24}$$

The domain length L plays a major role for spatio-temporal chaos since the bifurcation parameter, or the Lyapunov exponent which is the more meaningful parameter for us, strongly depends on the domain length. For small domain lengths, traveling waves emerge. With increasing domain length, intermittent bursts start to occur and disrupt the ordered structure. For sufficiently large domain sizes, the solution shows chaotic behavior with spatio-temporal

complex patterns [5].
We transform Eq. (2.23) into the Fourier domain to exclusively allow periodic solutions. We obtain the following set of ordinary differential equations for the time dependent Fourier coefficients $a_k(t)$:

$$\dot{a}_k(t) = \left(k^2 - k^4\right) a_k - \frac{ik}{2} \mathscr{F}\left[u(t)^2\right] \quad k \in n\frac{2\pi}{L},\, n \in \mathcal{Z} \tag{2.25}$$

where $\mathscr{F}$ is the Fourier transform, z is the imaginary unit and k is the wave number. Generally, the nonlinear term is computed directly. To do that, we used the following relationships:

$$u\frac{\partial u}{\partial x} = \frac{1}{2}\frac{\partial u^2}{\partial x} \tag{2.26a}$$

$$\mathscr{F}\left[\frac{\partial u^2}{\partial x}\right] = ik \cdot \mathscr{F}\left[u^2\right] \tag{2.26b}$$

How different terms in Eq. (2.23) act on periodic solutions with different wave numbers can be seen in Eq. (2.25). The fourth order term acts as an energy sink and the second order term as energy source. The nonlinear term acts as energy transport amongst different modes. Depending on the wave number, modes get damped or amplified. More specifically, long waves corresponding to small wave numbers are unstable and short waves with big wave numbers are stable. The special case $k = 1$ is the limit where modes get neither amplified nor damped.

The time evolution of the KS equation using exponential time differencing with Runge Kutta 4 as explained in section 3.3, with a step size $h = 0.25$, a domain length $l = 35$ and 64 grid points, is depicted in Fig. 2.7.

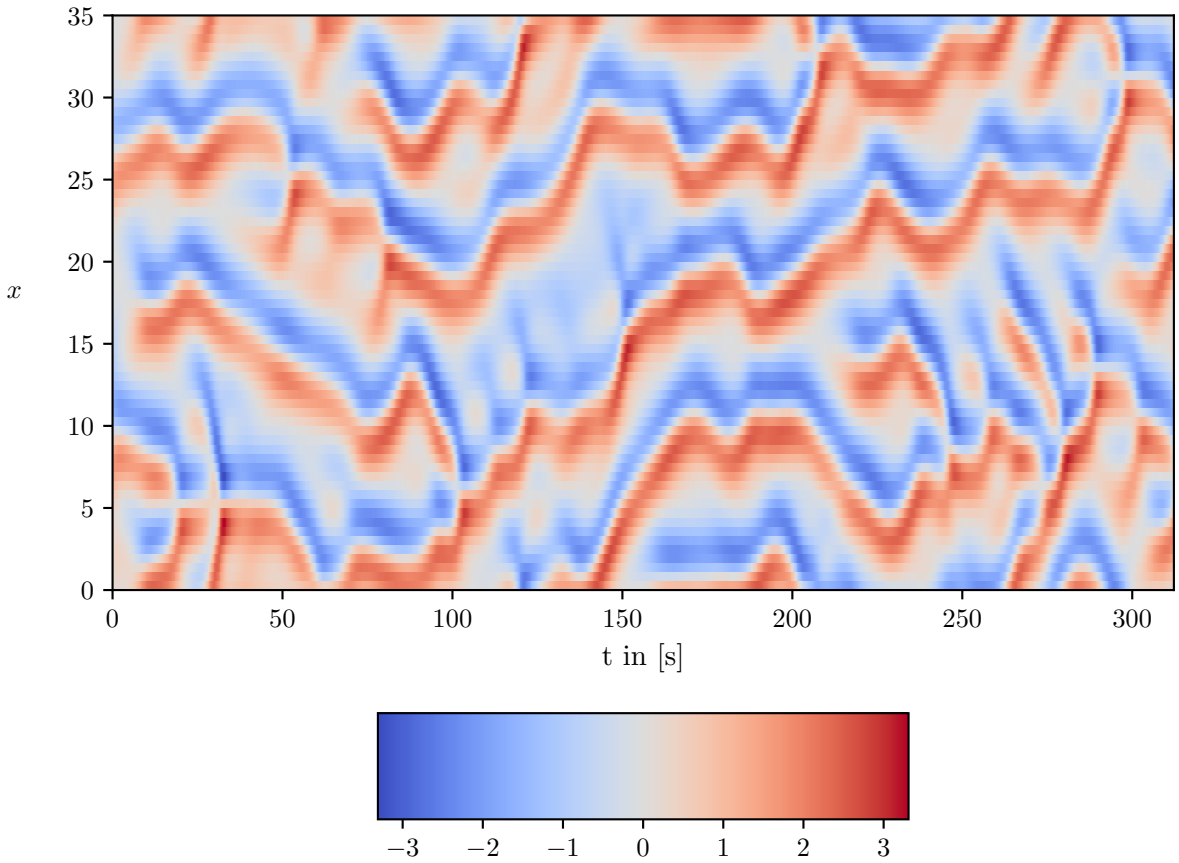

Figure 2.7: Time evolution of the KS equation.

3 Numerical Methods

This chapter deals with the time integration methods necessary to obtain the solution of a dynamical system. We consider a dynamical system described by an autonomous ordinary differential equation. The absence of explicit occurrence of the independent variable (here t) in the function $f = f(x) \neq f(x, t)$ of Eq. (3.1) characterizes an autonomous ordinary differential equation.

$$\frac{\mathrm{d}x}{\mathrm{d}t} \equiv \dot{x} = f(x(t)), \quad x(t_0) = x_0 \tag{3.1}$$

Eq. (3.1) is an initial value problem with given initial condition. In order to find find a solution $x(t)$, we integrate Eq. (3.1) in time:

$$x(t) = \int_{t_0}^{t} f(x(t))\,\mathrm{d}t + x_0 \tag{3.2}$$

In most cases, it is not possible to find an analytical solution of Eq. (3.2) and we seek an approximate solution. Numerical methods like finite difference methods applied on time derivatives help us to accomplish this task. They are described next.

3.1 Finite Difference

Finite difference methods aim to approximate the differential quotient with a suited difference quotient. Instead of computing the derivative exactly with the limit, we are satisfied with a sufficient small step size h that yields an acceptable truncation error and thus accurate enough results. There are theoretically infinite ways of creating finite difference schemes but in the following, we present some of the basic ones in one dimension. The first finite difference we consider is the right sided finite difference, where we take the value right of the evaluation location to construct the difference quotient:

$$\left.\frac{\mathrm{d}u(x)}{\mathrm{d}x}\right|_{x_n} \approx \frac{u(x_i + h) - u(x_i)}{h} + \mathcal{O}(h) \tag{3.3}$$

u is an arbitrary continuously differentiable function. The leading term in the local truncation error is of order of magnitude h as denoted by $\mathcal{O}(h)$. We call a difference scheme of n^{th} order if the leading term in the local truncation error is of $\mathcal{O}(h^n)$. In the same manner, we can construct a left sided finite difference by taking the left instead of the right value:

$$\left.\frac{\mathrm{d}u(x)}{\mathrm{d}x}\right|_{x_n} \approx \frac{u(x_i) - u(x_i - h)}{h} + \mathcal{O}(h) \tag{3.4}$$

Again, we obtain a first order difference scheme, which can be shown with a Taylor series. Taking the left and the right value next to the evaluation point yields a second order difference scheme and is called central difference.

$$\frac{\mathrm{d}u(x)}{\mathrm{d}x}\Big|_{x_n} \approx \frac{u(x_i+h)-u(x_i-h)}{2h} + \mathcal{O}\left(h^2\right) \tag{3.5}$$

Applying the right sided finite difference on ordinary differential equations is called the explicit Euler method. It is explicit because the next time step can be directly calculated with all known values. We replace the variables in Eq. (3.3) and insert it in Eq. (3.1).

$$\frac{x_{n+1}-x_n}{h} = f(x_n) \tag{3.6}$$

In addition, we introduce the abbreviation $x(t_n) = x_n$. Solving Eq. (3.6) for x_{n+1} yields the explicit Euler scheme:

$$x_{n+1} = x_n + h \cdot f(x_n) \tag{3.7}$$

In comparison, if we use the left sided finite difference to approximate the time derivative we get the implicit backward Euler method. It is implicit because the right hand side of Eq. (3.8) is evaluated with values at the next timestep.

$$\frac{x_n - x_{n-1}}{h} = f(x_n) \tag{3.8}$$

Shifting the time step by one and solving Eq. (3.8) for x_{n+1} yields the final implicit Euler scheme, however further calculations have to be made at every time step to find x_{n+1}.

3.2 Runge Kutta 4

In the previous section, we discussed solely one step methods applied on autonomous ordinary differential equations. The Runge Kutta 4 is the classical Runge Kutta method and is a multi step scheme (more precise, 4 steps)[6]. It is a weighted average of the slopes $f(x,t)$ evaluated at four different locations. To keep it more general, we consider a general ODE and allow the explicit dependence of the independent variable t in the right hand side of Eq. (3.9).

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(x(t),\, t) \tag{3.9}$$

The method is as follows:

$$k_1 = f(x_n,\, t_n) \tag{3.10a}$$

$$k_2 = f\left(x_n + h\frac{k_1}{2},\, t_n + \frac{h}{2}\right) \tag{3.10b}$$

$$k_3 = f\left(x_n + h\frac{k_2}{2},\, t_n + \frac{h}{2}\right) \tag{3.10c}$$

$$k_4 = f(x_n + h\cdot k_3,\, t_n + h) \tag{3.10d}$$

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.10e}$$

$$t_{n+1} = t_n + h \tag{3.10f}$$

With a suitable step size $h > 0$, the local truncation error of Runge Kutta 4 is of 5^{th} order and therefore the method is of 4^{th} order in the context of integration of ODEs.

3.3 Exponential Time Differencing with Runge Kutta 4

Exponential time differencing methods are particularly suitable for stiff ordinary differential equations which may appear when transforming PDEs in Fourier space. This is due to the fact that the linear part, which contains strongly varying eigenvalues in stiff systems, is exactly integrated.
We consider again a dynamical system, Eq. (3.1), and decompose it into its linear part $\underline{\mathbf{L}}x$ and its nonlinear part N:

$$\dot{x} = \underline{\mathbf{L}}x(t) + N(x(t)), \quad x(t_0) = x_0 \tag{3.11}$$

Exact integration of Eq. (3.11) yields the corresponding Volterra integral equation [8]:

$$x(t) = e^{\underline{\mathbf{L}}t}x(0) + \int_0^t e^{\underline{\mathbf{L}}(t-\tau)}N(x(\tau))\,\mathrm{d}\tau \tag{3.12}$$

Cox and Matthews [2] were the first scientists that applied Runge Kutte 4 method to integrate the nonlinear part of Eq. (3.12). To do this, we write Eq. (3.12) in time discrete form with time step size h:

$$x(t_{n+1}) = e^{\underline{\mathbf{L}}h}x(t_n) + e^{\underline{\mathbf{L}}h}\int_0^h e^{-\underline{\mathbf{L}}\tau}N(x(t_n+\tau))\,\mathrm{d}\tau \tag{3.13}$$

We apply Runge Kutta 4, combine everything and obtain the explicit update formula for one time step [17]:

$$a_n = \varphi_0\left(\frac{h\underline{\mathbf{L}}}{2}\right)x_n + \frac{h}{2}\varphi_1\left(\frac{h\underline{\mathbf{L}}}{2}\right)N(x_n) \tag{3.14a}$$

$$b_n = \varphi_0\left(\frac{h\underline{\mathbf{L}}}{2}\right)x_n + \frac{h}{2}\varphi_1\left(\frac{h\underline{\mathbf{L}}}{2}\right)N(a_n) \tag{3.14b}$$

$$c_n = \varphi_0\left(\frac{h\underline{\mathbf{L}}}{2}\right)x_n + \frac{h}{2}\varphi_1\left(\frac{h\underline{\mathbf{L}}}{2}\right)[2N(b_n) - N(x_n)] \tag{3.14c}$$

$$x_{n+1} = \varphi_0\left(h\underline{\mathbf{L}}\right) + hf_1\left(h\underline{\mathbf{L}}\right)N(x_n) + hf_2\left(h\underline{\mathbf{L}}\right)[N(a_n) + N(b_n)] + hf_3\left(h\underline{\mathbf{L}}\right)N(c_n) \tag{3.14d}$$

where f_1, f_2 and f_3 are linear combinations of the functions φ.

$$f_1\left(h\underline{\mathbf{L}}\right) = \varphi_1\left(h\underline{\mathbf{L}}\right) - 3\varphi_2\left(h\underline{\mathbf{L}}\right) + 4\varphi_3\left(h\underline{\mathbf{L}}\right) \tag{3.15a}$$

$$f_2\left(h\underline{\mathbf{L}}\right) = 2\varphi_2\left(h\underline{\mathbf{L}}\right) - 4\varphi_3\left(h\underline{\mathbf{L}}\right) \tag{3.15b}$$

$$f_3\left(h\underline{\mathbf{L}}\right) = -\varphi_2\left(h\underline{\mathbf{L}}\right) + 4\varphi_3\left(h\underline{\mathbf{L}}\right) \tag{3.15c}$$

$$\varphi_0\left(h\underline{\mathbf{L}}\right) = e^{h\underline{\mathbf{L}}} \tag{3.16a}$$

$$\varphi_1\left(h\underline{\mathbf{L}}\right) = h^{-1}\underline{\mathbf{L}}^{-1}\left(e^{h\underline{\mathbf{L}}} - \underline{\mathbf{I}}\right) \tag{3.16b}$$

$$\varphi_2\left(h\underline{\mathbf{L}}\right) = h^{-2}\underline{\mathbf{L}}^{-2}\left(e^{h\underline{\mathbf{L}}} - h\underline{\mathbf{L}} - \underline{\mathbf{I}}\right) \tag{3.16c}$$

$$\varphi_3\left(h\underline{\mathbf{L}}\right) = h^{-3}\underline{\mathbf{L}}^{-3}\left(e^{h\underline{\mathbf{L}}} - h^2\underline{\mathbf{L}}^2/2 - h\underline{\mathbf{L}} - \underline{\mathbf{I}}\right) \tag{3.16d}$$

The presented method is called exponential time differencing with Runge Kutta 4 (ETDRK4). For simplicity, we consider here exclusively the case where $\underline{\mathbf{L}}$ is diagonal and give some details about the computation of the functions φ_i with $i \in 1,2,3$. However, non diagonal $\underline{\mathbf{L}}$ can be diagonalized and all operations are applied only on the diagonal terms. Let λ be one diagonal element of $\underline{\mathbf{L}}$ and $z = h\lambda$. Each of the functions φ_i contains the function:

$$g(z) = \frac{e^z - 1}{z} \tag{3.17}$$

The accurate computation of this function due to round-off errors is a well known problem in numerical analysis [12]. Furthermore, $g(z)$ has a removable singularity in $z = 0$. Kassam and Trefethen [12] avoided both problems by making use of complex analysis's residue theorem. They evaluated the complex line integral around a closed curve Γ containing the desired argument $h\lambda$ where φ_i is to be evaluated.

$$\varphi_i(h\lambda) = \frac{1}{2\pi i}\oint_\Gamma \frac{\varphi_i(z)}{z - h\lambda}\mathrm{d}z,\ i \in 1,2,3 \tag{3.18}$$

We use a circle that is centered at $h\lambda$ as integration path Γ, approximate the integral with a mean over K equally spaced points and simplify the computation by evaluating only the upper half of the circle and taking the real part [12]. As a result, Eq. (3.18) simplifies to [17]:

$$\varphi_i(h\lambda) = \frac{1}{2\pi i}\oint_\Gamma \frac{\varphi_i(z)}{z - h\lambda}\mathrm{d}z \approx \frac{1}{K}\sum_{k=1}^{K}\varphi_i\left(h\lambda + e^{2\pi i(k-0.5)/K}\right),\ i \in 1,2,3 \tag{3.19}$$

3.4 Examples of applications

3.4.1 Lorenz System

We apply the RK4 method to the Lorenz System. Therefore, we rewrite the Lorenz system Eq. (2.1):

$$\dot{\underline{x}} = \underline{f}\left(\underline{x}\right) \tag{3.20a}$$

$$\underline{f} = \begin{bmatrix} -\sigma x_1 + \sigma x_2 \\ \rho x_1 - x_2 - x_1 x_3 \\ -\beta x_3 + x_1 x_2 \end{bmatrix} \tag{3.20b}$$

Since no explicit time dependence occurs in function $\underline{f}$, Eq. (3.10) simplifies to:

$$k_1 = f(x_n) \tag{3.21a}$$
$$k_2 = f\left(x_n + h\frac{k_1}{2}\right) \tag{3.21b}$$
$$k_3 = f\left(x_n + h\frac{k_2}{2}\right) \tag{3.21c}$$
$$k_4 = f(x_n + h \cdot k_3) \tag{3.21d}$$
$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{3.21e}$$

Iterative execution of the update rule above integrates the governing equation forward in time.

3.4.2 KS System

We recall the in Fourier space transformed KS Eq. (2.25):

$$\dot{a}_k(t) = \left(k^2 - k^4\right) a_k - \frac{ik}{2}\mathrm{FT}\left[u(t)^2\right] \quad k \in n\frac{2\pi}{L},\, n \in \mathcal{Z}$$

For this case, the Fourier transform is particularly suitable since it transforms the KS equation into a system of ODEs and restricts the solution to periodic solutions at the same time.
We rewrite the Fourier-transformed KS equation in matrix-vector notation and identify the linear and nonlinear term as needed for ETDRK4:

$$\underline{\dot{a}} = \mathbf{L}\,\underline{a} + \underline{N} \tag{3.22a}$$

$$\mathbf{L} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \left(\frac{2\pi}{L}\right)^2 - \left(\frac{2\pi}{L}\right)^4 & & \\ \vdots & & \ddots & \\ 0 & & & \left(n\frac{2\pi}{L}\right)^2 - \left(n\frac{2\pi}{L}\right)^4 \end{bmatrix} \tag{3.22b}$$

$$\underline{N} = -\frac{i}{2}\,\underline{k}\,\mathrm{FT}\left[u(t)^2\right] \tag{3.22c}$$

The next step is to precompute all utility functions. We start with the two easiest ones that do not suffer from singularity problem.

$$e^{\frac{1}{2}h\underline{\mathbf{L}}} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & e^{\frac{1}{2}h\left[\left(\frac{2\pi}{L}\right)^2-\left(\frac{2\pi}{L}\right)^4\right]} & & \\ \vdots & & \ddots & \\ 0 & & & e^{\frac{1}{2}h\left[\left(n\frac{2\pi}{L}\right)^2-\left(n\frac{2\pi}{L}\right)^4\right]} \end{bmatrix} \tag{3.23a}$$

$$e^{h\underline{\mathbf{L}}} = \varphi_0\left(h\underline{\mathbf{L}}\right) \tag{3.23b}$$

The remaining ones φ_1, φ_2 and φ_3 are solved using Eq. (3.19) as the constant portion ($k = 0$) would cause divisions by zero. We replace λ with the diagonal entries of $\underline{\mathbf{L}}$ and solve for each λ. Afterwards, we compute f_1, f_2 and f_3 that are linear combinations of the functions calculated before and are ready to start the iteration process, Eq. (3.14).

4 Fundamentals of Reduced Order Modelling

4.1 Proper Orthogonal Decomposition

The Proper Orthogonal Decomposition (POD) method was originally proposed by Lumley et al. [15] in the field of fluid mechanics. It is a modal decomposition designed to identify coherent structures in turbulent flows. The idea is to decompose a given field $\underline{u}\left(\underline{x},\, t\right)$ into a sum of time dependent coefficients $a_i\,(t)$ and spatial modes $\underline{\Phi}_i\left(\underline{x}\right)$ that are orthonormal to each other.

$$\underline{u}\left(\underline{x},\, t\right) = \sum_{i=1}^{M} a_i\,(t)\,\underline{\Phi}_i\left(\underline{x}\right) \tag{4.1}$$

where M is the number of modes to be used. At most, we use all the modes available to us, which corresponds to the number of input variables and includes all the information, however we aim here to reduce the problem's complexity and, thus, the least energy/information carrying modes are removed. How to find these modes is explained next.
The "proper" in POD tries to describe the most important characteristic of the POD, meaning that the decomposition is optimal regarding the mean squared error ϵ^2

$$\min_{\underline{\phi}_i} \epsilon^2\,(l) = \left\langle \left| \underline{u} - \underline{u}\,(l) \right|^2 \right\rangle \tag{4.2}$$

Let $\underline{u}$ be a random vector in $\mathbb{R}^n$ and $\underline{u}\,(l)$ be the reconstruction using the l first modes of the yet undetermined basis. No other decomposition yields a smaller mean squared error with the same number of modes. Using a given number of modes retains, in this case, the most possible kinetic energy in the system with that number of modes. The eigenvalues of the covariance matrix $\underline{\mathbf{C}}$ are an indicator for the portion of kinetic energy contained in each mode. If the mean velocity is subtracted and velocity fluctuations are considered, the mean squared error is actually a measure for the turbulent kinetic energy.
In the following, we explain the POD method step-by-step for the n-dimensional case as described in [26]. We start and create the snapshot matrix $\underline{\mathbf{U}}$. This matrix is the horizontal concatenation of different variables that are determined for the decomposition. Each variable is a vector, element of $\mathbb{R}^m$, where m is the number of time steps where the variable is evaluated. The number of variables n determines the dimension of the problem and is composed as follows

$$n = n_u \cdot n_x \cdot n_y \tag{4.3}$$

We consider results of a numerical two-dimensional flow simulation to elaborate on Eq. (4.3). In this case, n_u is for instance the number of velocity components $n_u = 2$. If we want to conduct a purely velocity based POD, n_x is the number of grid points in x direction and n_y is the number of grid points in y direction.
In contrast to the location dependent solutions of partial differential equations, the problem simplifies drastically if we consider POD for ODEs like the Lorenz system (section 2.1.1). The grid size is 1 in this case $n_x \cdot n_y = 1$ and the number of variables is 3 for the Lorenz system. Therefore, the POD problem is 3 dimensional.

As a result, the $m \times n$ snapshot matrix $\underline{\mathbf{U}}$ reads:

$$\underline{\mathbf{U}} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ u_{21} & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & & \vdots \\ u_{m1} & u_{m2} & \dots & u_{mn} \end{bmatrix} \tag{4.4}$$

The next step is the covariance matrix $\underline{\mathbf{C}}$ computation:

$$\underline{\mathbf{C}} = \frac{1}{m-1}\underline{\mathbf{U}}^{\mathrm{T}}\underline{\mathbf{U}} = \frac{1}{m-1}\begin{bmatrix} \sum_{i=1}^{m} u_{i1}u_{i1} & \sum_{i=1}^{m} u_{i1}u_{i2} & \dots & \sum_{i=1}^{m} u_{i1}u_{in} \\ \sum_{i=1}^{m} u_{i2}u_{i1} & \sum_{i=1}^{m} u_{i2}u_{i2} & \dots & \sum_{i=1}^{m} u_{i2}u_{in} \\ \vdots & \vdots & & \vdots \\ \sum_{i=1}^{m} u_{in}u_{i1} & \sum_{i=1}^{m} u_{in}u_{i2} & \dots & \sum_{i=1}^{m} u_{in}u_{in} \end{bmatrix} \tag{4.5}$$

The diagonal entries of Eq. (4.5) are the variances of the n variables and the non-diagonal entries the respective covariance. It should be noted that, in the context of fluid mechanics, the turbulent kinetic energy (TKE) is half of the trace of $\underline{\mathbf{C}}$:

$$\mathrm{TKE} = \frac{1}{2}\mathrm{tr}\left(\underline{\mathbf{C}}\right) = \frac{1}{2}\sum_{i=1}^{n} C_{ii} \tag{4.6}$$

The trace and, intuitively, the TKE as well is invariant to coordinate transformation. Therefore, also the eigenvalues of $\underline{\mathbf{C}}$ are a measure for the TKE. We want to find the direction of maximum variance to create our POD basis which results in the eigenvalue problem:

$$\underline{\mathbf{C}}\,\underline{\Phi}_i = \lambda_i \underline{\Phi}_i \tag{4.7}$$

We form a diagonal matrix containing the eigenvalues λ_i and the POD modes or principal axis $\underline{\Phi}_i$

$$\underline{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & 0 & \lambda_{n-1} & 0 \\ 0 & \dots & 0 & 0 & \lambda_n \end{bmatrix} \tag{4.8}$$

$$\underline{\Phi} = \begin{bmatrix} \underline{\Phi}_1, & \underline{\Phi}_2, & \dots, & \underline{\Phi}_n \end{bmatrix} \tag{4.9}$$

and can diagonalize $\underline{\mathbf{C}}$

$$\underline{\mathbf{C}} = \underline{\Phi}\,\underline{\Lambda}\,\underline{\Phi}^{-1} = \underline{\Phi}\,\underline{\Lambda}\,\underline{\Phi}^{\mathrm{T}} \tag{4.10}$$

Since the covariance matrix is symmetric, the eigenvectors form an orthonormal basis. As noted before, the trace is invariant and $\mathrm{TKE} = \frac{1}{2}\mathrm{tr}\left(\underline{\mathbf{C}}\right) = \frac{1}{2}\mathrm{tr}\left(\underline{\Lambda}\right)$ applies. This property is useful to determine which modes contribute the most to the TKE. In section 4.2, we will neglect the modes with the smallest energy contribution $\frac{\lambda_i}{\mathrm{TKE}}$.
We get the time dependent coefficients a_i evaluated at the m points in time by simply projecting the snapshot matrix onto the new POD basis:

$$\underline{a}_1 = \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \end{bmatrix} = \underline{\mathbf{U}}\,\underline{\Phi}_1 \tag{4.11}$$

Or in matrix notation:

$$\underline{\mathbf{A}} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = \underline{\mathbf{U}}\,\underline{\Phi} \tag{4.12}$$

The inverse mapping to get, from POD coefficients and POD modes, to the original snapshot matrix reads:

$$\underline{\mathbf{U}} = \underline{\mathbf{A}}\,\underline{\Phi}^{\mathrm{T}} \tag{4.13}$$

4.2 Galerkin Projection

Reduced order models aim to reduce computation time and complexity of dynamical systems while preserving their main characteristics and behavior. Once a set of POD base functions has been found, we can project the original equations onto them and simultaneously skip the modes carrying the least energy. The result is a new ODE that describes the POD coefficients evolution in time. This methodology is illustrated next for an ODE system, the Lorenz System and the KS equation.

4.2.1 ROM Lorenz System

The basic procedure will be demonstrated using the Lorenz system. The state vector $\underline{x}$ can be rewritten using Eq. (4.1):

$$\underline{x}(t) = \sum_{i=1}^{R} a_i(t)\,\underline{\Phi}_i \tag{4.14}$$

$R \leq n$ is the number of retained modes and the coefficients a are the projections of $\underline{x}$ onto $\underline{\phi}$.

$$a_i(t) = \langle \underline{x}(t), \underline{\Phi}_i \rangle \tag{4.15}$$

$\langle \bullet, \bullet \rangle$ in Eq. (4.15) denotes the standard inner product in Euclidian space. We project the Lorenz system, Eq. (2.1), onto the new basis Φ_i for $i \in 1,2,..,R$ and get:

$$\langle \underline{\dot{x}}, \underline{\Phi}_i \rangle = \dot{a}_i(t) = \left\langle \begin{bmatrix} -\sigma x_1 + \sigma x_2 \\ \rho x_1 - x_2 - x_1 x_2 \\ -\beta x_3 + x_1 x_2 \end{bmatrix}, \underline{\Phi}_i \right\rangle \tag{4.16}$$

The left side of Eq. (4.16) is obtained as:

$$\dot{a}_i = \frac{\mathrm{d}}{\mathrm{d}t}\langle \underline{x}, \underline{\Phi}_i \rangle = \frac{\mathrm{d}}{\mathrm{d}t} \sum_{j=1}^{n} x_j \Phi_{ji} = \sum_{j=1}^{n} \frac{\mathrm{d}x_j}{\mathrm{d}t} \Phi_{ji} = \langle \underline{\dot{x}}, \underline{\Phi}_i \rangle \tag{4.17}$$

We use Eq. (4.14) to rewrite the right hand side of Eq. (4.16) in terms of coefficients and modes and rename it:

$$\begin{bmatrix} -\sigma x_1 + \sigma x_2 \\ \rho x_1 - x_2 - x_1 x_2 \\ -\beta x_3 + x_1 x_2 \end{bmatrix} = \begin{bmatrix} \sigma \sum_{j=1}^{R} a_j \left(\Phi_{2j} - \Phi_{1j} \right) \\ \sum_{j=1}^{R} a_j \left(\rho \Phi_{1j} - \Phi_{2j} - \Phi_{1j} \sum_{k=1}^{R} a_k \Phi_{2k} \right) \\ -\sum_{j=1}^{R} a_j \left(-\beta \Phi_{3j} + \Phi_{1j} \sum_{k=1}^{R} a_k \Phi_{2k} \right) \end{bmatrix} = \begin{bmatrix} \mathscr{F}_1 \\ \mathscr{F}_2 \\ \mathscr{F}_3 \end{bmatrix} \tag{4.18}$$

Summing up all intermediate results yields the final system:

$$a_i(t) = \langle \underline{\mathscr{F}}, \underline{\Phi}_i \rangle = \sum_{j=1}^{3} \mathscr{F}_j \Phi_{ji} \qquad i \in 1,2,..,R \tag{4.19}$$

with Φ_{ji} the j^{th} component of the i^{th} mode.

4.2.2 ROM Kuramoto - Sivashinsky Equation

We recall the KS equation:

$$u_{,t} + u_{,xxxx} + u_{,xx} + u u_{,x} = 0$$

In a first step, we transform the infinite dimensional problem into a finite dimensional system of differential equations. We achieve this by either replacing the spatial derivatives with finite differences or by simply evaluating the equation at each of the n grid points x_k over domain length L. We abbreviate $u(t, x_k) = u_k$. The set of coupled differential equations reads:

$$u_{k,t} + u_{k,xxxx} + u_{k,xx} + u_k u_{k,x} = 0 \quad k \in 1,2,..., \frac{L}{n} + 1 \tag{4.20}$$

The coupling is hidden in the spatial derivatives since they depend on the values at the neighboring grid points. It should be noted that a transformation in the Fourier domain also reduces the dimensionality if the wave number is restricted to the resolvable ones on a computational grid (Nyquist frequency).
As described in section 4.1, for a n-dimensional problem, with n equal to the number of grid points here, we obtain n POD modes. The number of grid points is usually large, which is why

we use the matrix vector notation in the case of PDE. The modal decomposition in matrix vector notation reads:

$$\underline{u}_k = \underline{\Phi}(x)\,\underline{a}(t) = \begin{bmatrix} \underline{\phi}_1, & \cdots & , \underline{\phi}_n \end{bmatrix} \begin{bmatrix} a_1(t) \\ \vdots \\ a_n(t) \end{bmatrix} \tag{4.21}$$

We insert the POD approach, Eq. (4.21), in Eq. (4.20):

$$\underline{\Phi}\,\dot{\underline{a}} = -\underline{\Phi}_{,xxxx}\underline{a} - \underline{\Phi}_{,xx}\underline{a} - \left(\underline{\Phi}\,\underline{a}\right)\left(\underline{\Phi}_{,x}\underline{a}\right) \tag{4.22}$$

Next, we use the orthonormal property:

$$\underline{\Phi}^{\mathrm{T}}\underline{\Phi} = \underline{\mathbf{I}} \tag{4.23}$$

and multiply Eq. (4.22) with $\underline{\Phi}^{\mathrm{T}}$ from the left side:

$$\dot{\underline{a}} = -\underline{\Phi}^{\mathrm{T}}\left(\underline{\Phi}_{,xxxx} + \underline{\Phi}_{,xx}\right)\underline{a} - \underline{\Phi}^{\mathrm{T}}\left[\left(\underline{\Phi}\,\underline{a}\right)\left(\underline{\Phi}_{,x}\underline{a}\right)\right] \tag{4.24}$$

Eq. (4.24) is the KS equation expressed in a different, or more precisely an optimal, basis set. The dimension and thus also the information content is reduced with a successive decrease in the modes used. At this point, the KS ROM is ready for use, however, we want to give some detailed information on how to solve it efficiently.
We use ETDRK4 as introduced in section 3.3 to integrate the ROM in time and identify the linear term $\underline{\mathbf{L}}$ and nonlinear term N by comparing Eq. (4.24) with the standard formulation of a dynamical system Eq. (3.11):

$$\underline{\mathbf{L}} = -\underline{\Phi}^{\mathrm{T}}\left(\underline{\Phi}_{,xxxx} + \underline{\Phi}_{,xx}\right) \tag{4.25a}$$

$$N = -\underline{\Phi}^{\mathrm{T}}\left[\left(\underline{\Phi}\,\underline{a}\right)\left(\underline{\Phi}_{,x}\underline{a}\right)\right] \tag{4.25b}$$

The linear term Eq. (4.25a) is constant and it is efficient to compute it once at the beginning. To do this, we make use of the Fourier transformation's property that allows for exact differentiation:

$$\frac{\partial^n}{\partial x^n}\underline{\Phi} = \mathscr{F}^{-1}\left[(ik)^n\,\mathscr{F}\left[\underline{\Phi}\right]\right] \quad n \in \mathcal{N}_0 \tag{4.26}$$

There are several other terms in the ETDRK4 scheme that depend on $\underline{\mathbf{L}}$ and can be precomputed. Nearly all of them contain the matrix exponential function. Since here and in general it is not the case that $\underline{\mathbf{L}}$ is diagonal, we can solve an eigenvalue problem first to calculate the matrix exponential function.

$$\exp\left(\underline{\mathbf{L}}\right) = \exp\left(\underline{\mathbf{T}}\,\underline{\Lambda}\,\underline{\mathbf{T}}^{-1}\right) = \underline{\mathbf{T}}\,\exp\left(\underline{\Lambda}\right)\,\underline{\mathbf{T}}^{-1} \tag{4.27}$$

The same applies for the functions for which the complex curve integral has to be solved. After the eigenvalue decomposition, the functions can be directly applied to the eigenvalues of $\underline{\Lambda}$.

5 Fundamentals of Neural Networks

5.1 The Artificial Neuron

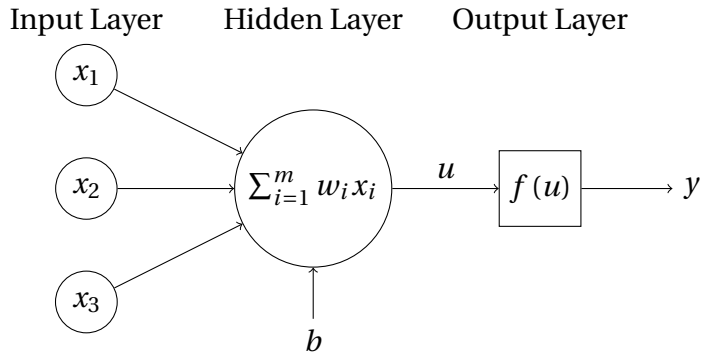

Figure 5.1: One artificial neuron with bias and activation function f.

The artificial neuron is the basic building block of neural networks. It consists of a layer containing the inputs $\underline{x}$, the hidden layer that aggregates the weighted inputs and adds a bias b and an output layer that applies the activation function f. In a first step, the inputs get weighted with the weights w_i, summed up and a bias b is added.

$$u = \sum_{i=1}^{m} w_i x_i + b = \underline{w} \cdot \underline{x} + b \tag{5.1}$$

In the case of layers with multiple neurons in the hidden layer, the weight vector $\underline{w}$ becomes the weight matrix $\underline{\mathbf{W}}$ and the bias becomes a bias vector $\underline{b}$ containing the bias for each neuron.

$$\underline{u} = \underline{\mathbf{W}}\,\underline{x} + \underline{b} \tag{5.2}$$

The output layer applies the activation function to the hidden layer's output u and returns the output variable y:

$$y = f(u) = f\left(\underline{w} \cdot \underline{x} + b\right) \tag{5.3}$$

The choice of the activation function is crucial to the neural network's performance. There exist various possibilities for f and for each application, different functions are used in those areas.

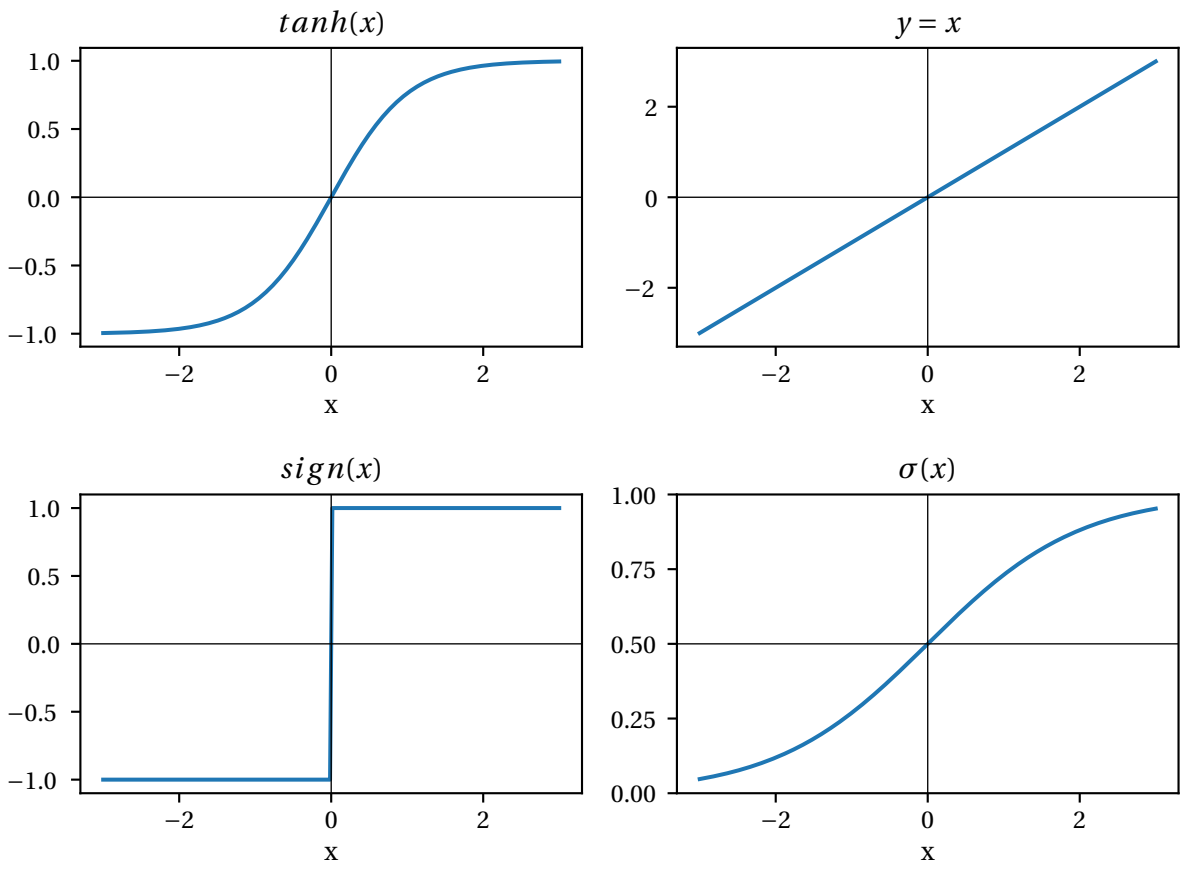

Figure 5.2: Different, often used activation functions.

There exist zero centered activation functions, such as the linear, the sign $sign(x)$ and the hyperbolic tangent $tanh(x)$ function but also non-zero centered ones like the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. These functions are plotted in Fig. 5.2. It also shows the importance of normalizing the data since bigger values may easily lead to saturation of neurons. Saturation means that the activation function acts far away from the linear region which is, in most cases, not desired.

5.2 Echo State Networks

Recurrent neural networks (RNN) have shown to be powerful tools to predict time series [18], [1]. Classical artificial neural networks (NN) are trained by error back propagation (BB). In the context of RNN, the BB algorithm is adapted to time series data and is called back propagation through time (BBTT). RNN training suffers from the vanishing or exploding gradient problem especially when using long time series. Repeating matrix multiplications cause either vanishing or very large gradients that prevents the updating of the weights and thus finding the

global minimum.
Circumventing this issue, it was actually shown that even without training all weights of an RNN, the prediction capability is often sufficient [14]. Using this consideration, Herbert Jäger [10], [11] introduced the Echo State Network (ESN) which is a form of reservoir computing. An ESN consists of three parts: the input layer, the reservoir and the output layer. The reservoir represents the actual RNN and is created randomly. The output layer contains adjustable weights and is trained. Together with teacher forcing, the training process simplifies to a simple linear regression problem. Teacher forcing removes the recurrent connection while training and is explained later in section (5.2).

Basic Model

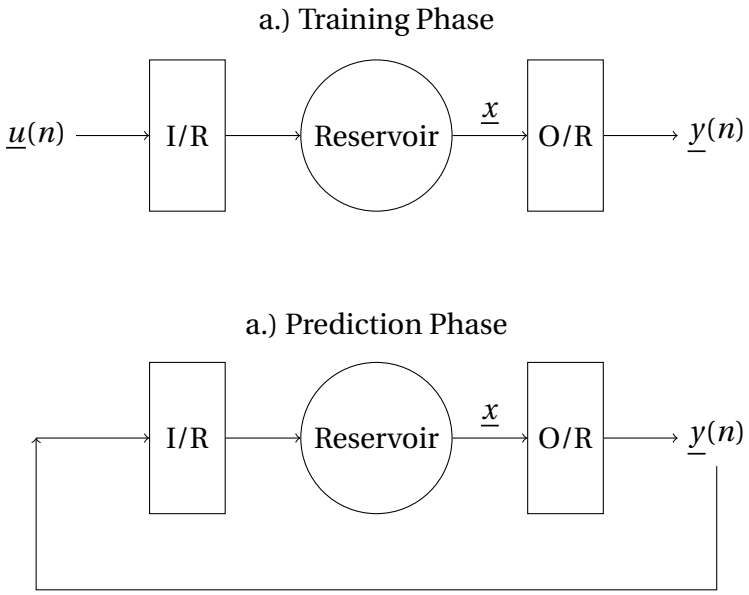

Figure 5.3: Basic ESN in (a) training phase and (b) prediction phase.

This sections explains our specific ESN implementation inspired by [14], [18], [19] and [1]. We consider a supervised temporal machine learning task where for a given input $\underline{u}(n) \in \mathbb{R}^{n_u}$ the output $\underline{y}(n) \in \mathbb{R}^{n_y}$ is known. $n = 1, 2, ..., T$ is the discrete time with T the number of samples in our data set. n_u is the number of input variables (for instance the state vector of the dynamical system at time t) and is, in all cases in this work, equal to the number of output variables n_y (for instance the state vector at $t+1$). Our goal is to create and train a model with output $\underline{y}(n)$ that reproduces the mapping $\underline{u}(n) \rightarrow \underline{y}^{target}(n)$ as good as possible with respect to an error measure $E(\underline{y}, \underline{y}^{target})$, typically the mean squared error (MSE).

The ESN consists of three main parts as shown in Fig. 5.3: The input to reservoir coupler, the reservoir and the reservoir output coupler. The input to reservoir coupler maps the input signal to the higher dimensional reservoir state $\underline{x}(n) \in \mathbb{R}^{n_x}$. n_x is the number of sparsely

connected neurons. Both, the input to reservoir and the reservoir to output coupler are linear operations and can therefore be realized with a matrix multiplication. However, a quadratic transformation is applied to the state vector $\underline{x}$ before applying the reservoir to output coupler. A typical update loop looks as follows:

$$\underline{\tilde{x}}(n) = \tanh\left(\underline{\mathbf{W}}_{in}\,\underline{u}(n) + \underline{\mathbf{W}}\,\underline{x}(n-1)\right) \tag{5.4}$$

$$\underline{x}(n) = (1-\alpha)\,\underline{x}(n-1) + \alpha\underline{\tilde{x}}(n) \tag{5.5}$$

$\underline{\mathbf{W}}_{in} \in \mathbb{R}^{n_x \times n_u}$ is the input matrix and $\underline{\mathbf{W}} \in \mathbb{R}^{n_x \times n_x}$ is the recurrent weight matrix. We use leaky integration in Eq. (5.5) with the leaking rate $\alpha \in (0,1]$. If we want to use the model without leaky integration, we simply set $\alpha = 1$. $\underline{\tilde{x}}(n) \in \mathbb{R}^{n_x}$ updates the internal state of the reservoir $\underline{x}$. The hyperbolic tangent function is applied element wise in Eq. (5.4).
Finally, the output is generated using a quadratic transformation [19] and the linear output layer $\underline{\mathbf{W}}_{out} \in \mathbb{R}^{n_y \times n_x}$. The quadratic transformation reads as follows:

$$\hat{x}_i = \begin{cases} x_i & \text{if } i = \text{odd} \\ x_i^2 & \text{if } i = \text{even} \end{cases} \qquad \text{for } i \in 1,2,....,n_x \tag{5.6}$$

The final output is:

$$\underline{y}(n) = \underline{\mathbf{W}}_{out}\,\underline{\hat{x}}(n) \tag{5.7}$$

Now that all basic equations have been determined, we can address the question of how to choose the matrices $\underline{\mathbf{W}}_{in}$ and $\underline{\mathbf{W}}$.

Reservoir

According to [14] the reservoir has two main functions: it is simultaneously a nonlinear higher dimensional expansion of the input signal $\underline{u}$ and a storage of previous inputs. Incorporating previous time steps is the essence of RNN. Using a high dimensional reservoir state ($\underline{x}$) should ensure that the desired output $\underline{y}_{target}$ can be recreated by a linear combination of the reservoir states.
The reservoir is specified by the input matrix $\underline{\mathbf{W}}_{in}$, the echo matrix $\underline{\mathbf{W}}$ and the leaking rate α. We focus in this section on a reasonable choice of the echo matrix while we discuss the input matrix in section 5.2 Input Scaling and the leaking rate in section 5.2 Parameter Search.

We follow the guidelines in [14] and begin with the size of the reservoir n_x. In general, the following rule applies: Choose a reservoir as big as you can afford computationally, choose the other hyperparameters with this reservoir size and scale to bigger ones afterwards. Indeed, the bigger the reservoir, the easier it is to find a solution by linear combination of the reservoir state. [14] provides an estimation for the lower boundary of the reservoir size:

$$n_{x,min} \geq n_u \times T \tag{5.8}$$

The reservoir size must be at least the number of independent variables n_u times the number of time steps T which is in total the number of independent real values the reservoir has to remember. However, in practice, existing temporal and inter-channel correlations in the input

reduce $n_{x,min}$.

The next parameter we consider is the sparsity of the echo matrix. The sparsity is the average number of connections one node in the reservoir has to other nodes. [16] suggests to connect each node to a fixed small number (for instance 5). In their practical experience, optimization leads only to slightly better results and therefore sparsity parameter has lower priority.

The remaining non-zero elements in $\underline{\mathbf{W}}$ are distributed uniformly in our case but also normal distribution or Gaussian distribution are popular among other authors. However, different methods show nearly the same performance using other parameters [14].

The spectral radius $\rho\left(\underline{\mathbf{W}}\right)$ is one of the most important global parameters of an ESN [14]. The spectral radius is defined as the largest absolute eigenvalue of the echo matrix. A reservoir should satisfy the so-called echo state property. This property ensures that with repeated application of the update equations (Eq. (5.4-5.7)), the influence of the initial condition fades with time. To satisfy the echo state property, the spectral radius is generally set to a value smaller than one.
While it is possible to violate the echo states property even with $\rho\left(\underline{\mathbf{W}}\right) < 1$, this is unlikely to happen in practice. Moreover, it is possible to satisfy the echo state property with $\rho\left(\underline{\mathbf{W}}\right) \geq 1$ for nonzero inputs [14]. [14] recommends to start with $\rho\left(\underline{\mathbf{W}}\right) = 1$ as initial point since optimal values can be bigger than one. In addition, tasks requiring longer history tend to work better with a bigger spectral radius and tasks requiring only short term history tend to work better with a smaller spectral radius.
We set the spectral radius of the echo matrix by first dividing $\underline{\mathbf{W}}$ with its own spectral radius and afterwards multiplying it with the desired one.

Input Scaling

Input scaling is another key parameter since it determines how nonlinear the reservoir responses are. Outliers in the input data should be avoided by normalizing the data or applying $tanh()$ before feeding it to the ESN. Otherwise these outliers cause the activation function to act far away from the linear region (saturation of the neuron) and may throw the reservoir in unfamiliar regions. Another functionality $\underline{\mathbf{W}}_{in}$ takes over is weighting the effects between $\underline{x}(n)$ and the history.
The input matrix $\underline{\mathbf{W}}_{in}$ is chosen randomly with a uniform distribution. We define the input scaling $[-a, a]$ as the range of the uniformly sampled values. Similar to the echo matrix, other types of distributions are possible. If different variables of the input $\underline{u}$ contribute differently to the task, it is also possible to scale the columns of $\underline{\mathbf{W}}_{in}$ separately [14].

Training

The biggest advantage of ESN is that the training problem is reduced to a simple linear regression problem. In addition, we use regularization to avoid overfitting and feedback instability.

For the training phase, we remove the recurrent connection (teacher forcing) and feed the ESN the input from the training set (see Fig. 5.3).
We rewrite Eq. (5.7) in matrix notation because we take into account all T time steps:

$$\underline{\mathbf{Y}} = \underline{\mathbf{W}}_{out}\,\hat{\underline{\mathbf{X}}} \tag{5.9}$$

$\underline{\mathbf{Y}} \in \mathbb{R}^{n_y \times T}$ is the collection of all $\underline{y}(n)$ and $\hat{\underline{\mathbf{X}}} \in \mathbb{R}^{n_x \times T}$ is the collection of all $\hat{\underline{x}}(n)$ for $n = 1, 2, ..., T$. The desired output is $\underline{\mathbf{Y}}_{target} \in \mathbb{R}^{n_y \times T}$. Finding the optimal weights in $\underline{\mathbf{W}}_{out}$ with respect to the mean squared error between $\underline{y}(n)$ and $\underline{y}_{target}(n)$ results in the following over determined system of linear equations:

$$\underline{\mathbf{Y}}_{target} = \underline{\mathbf{W}}_{out}\,\hat{\underline{\mathbf{X}}} \tag{5.10}$$

One of the most commonly used methods to solve Eq. (5.10) is ridge regression or also known as Tikhonov regularization [14]:

$$\underline{\mathbf{W}}_{out} = \underline{\mathbf{Y}}_{target}\,\hat{\underline{\mathbf{X}}}^{\mathrm{T}}\left(\hat{\underline{\mathbf{X}}}\hat{\underline{\mathbf{X}}}^{\mathrm{T}} + \beta\underline{\mathbf{I}}\right)^{-1} \tag{5.11}$$

β is the regularization parameter and can be chosen individually for a concrete ESN using validation without rerunning through the entire training data. Regular linear regression is a special case of ridge regression with $\beta = 0$.
For further details about large data sets and computing $\underline{\mathbf{W}}_{out}$ incrementally, we refer to [14]. Furthermore, weighting techniques may be applied to change the importance of different time steps in order to further improve performance.

Parameter Search

We follow the recommendations in [14] and determine the order of parameters we change one by one successively. With a fixed reservoir size n_x, the order is:

1. input scaling $[-a, a]$
2. spectral radius $\rho\left(\underline{\mathbf{W}}\right)$
3. degree of connection
4. leaking rate α
5. ridge regression β

As depicted in Fig. 5.4 and Fig. 5.5, the parameter search consists of two steps. Step one is to find the best choice of parameters in a range via a line search starting from a first guess of parameters. We decide which parameters are the best solely based on the valid prediction time. The valid prediction time is the time until the error E between the predicted solution

and the reference solution reaches a previously defined threshold ϵ. Typical values for ϵ are 0.2, 0.3 or 0.4. The error E is defined as:

$$E(t) = \frac{\left| \underline{y}_{target}(t) - \underline{y}(t) \right|}{\left\langle \left| \underline{y}_{target}(t) \right|^2 \right\rangle^{1/2}} \tag{5.12}$$

We predict in 50 regions not included in the training data and consider the mean for decision.

In the second phase of the parameter search, starting from the best guess from phase 1, we go in both directions with a certain step size h. After each step we take a look if the performance has improved. If so, we take a step with step size h. If not, we shorten the step size by a factor f_{ref} and take a step with the new step size $h_{new} = h_{old} \cdot f_{ref}$. The refinement process can take place n_{ref} times. After n_{max} steps or n_{ref} refinement steps, the algorithm moves on to the next parameter to be optimized and the better value from both directions gets chosen.

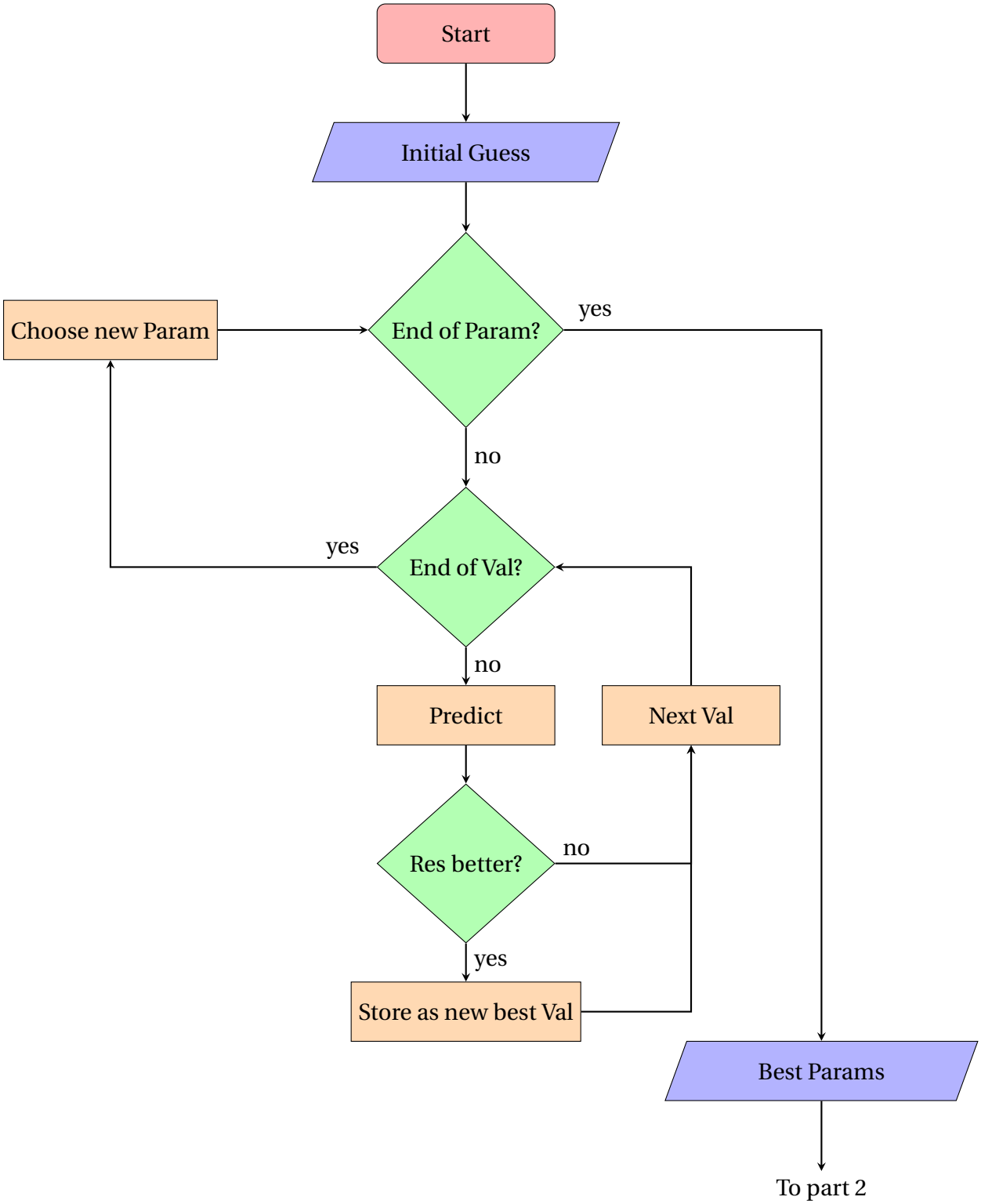

Figure 5.4: First step in the parameter search algorithm.

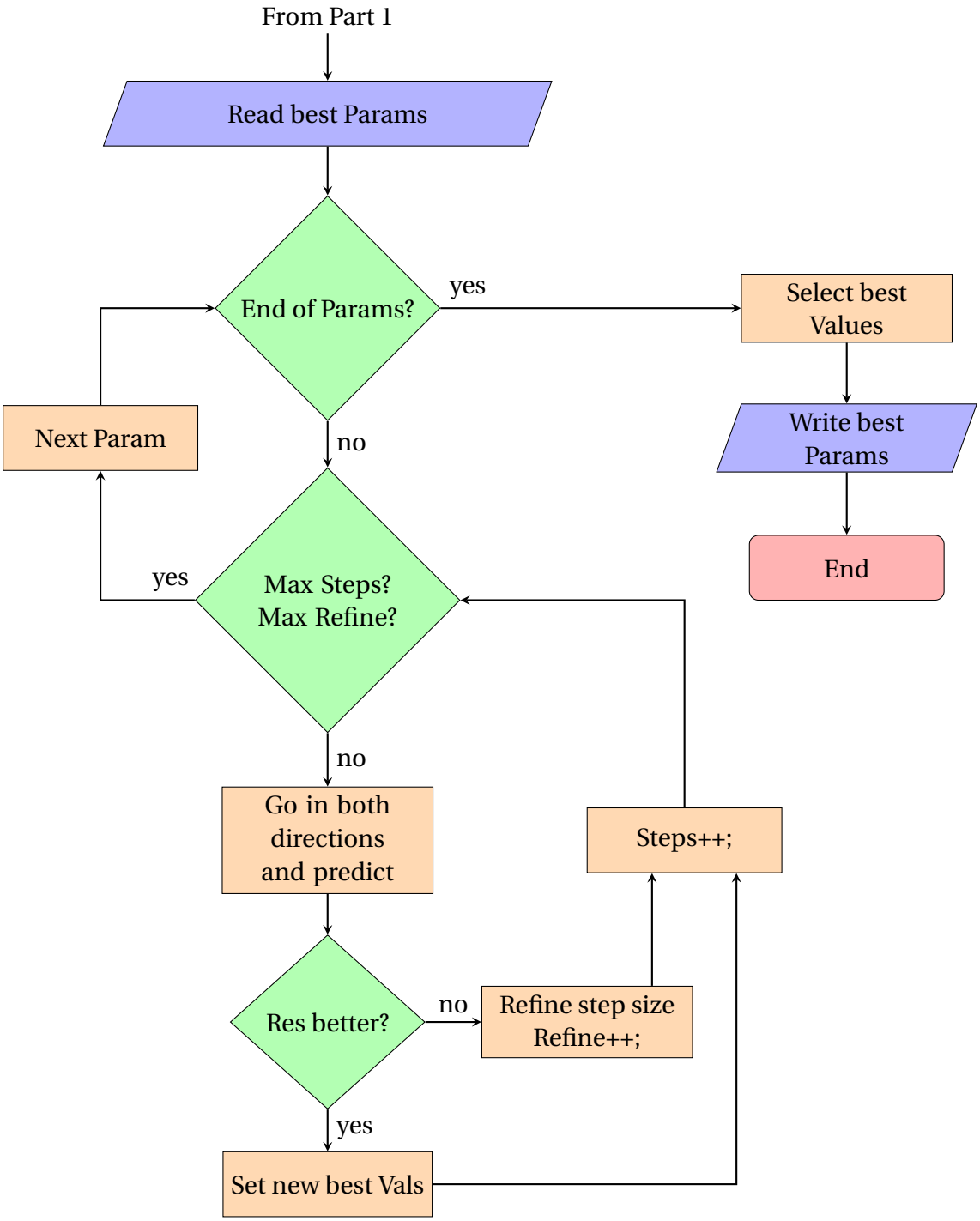

Figure 5.5: Second step in the parameter search algorithm.

6 Prediction Methods

On one hand, analytical relations are extremely powerful but are generally hard or impossible to solve due to the complexity of the problem. Furthermore, only in simplified cases, they include all the physical phenomena of the problem. On the other hand, recent development in information technology delivers more and more available data which is the core of ML techniques. Hybrid methods, explored here, try to combine the knowledge about physics (knowledge-based model) with pure data from past observations (data-driven model). In this thesis, we assess different approaches inspired by [20] and [19].

6.1 Method 1

The first hybrid architecture is inspired by [20]. Fig. 6.1 shows the schematic diagram of the prediction and training setup with a switch that defines whether the model runs in training or prediction mode. In the first step, the reference solution and therefore our training data is used to determine the POD basis with the respective eigenvalues. We decompose the state input vector and retain the $R \leq n$ most energy containing modes, with n being the total number of modes.

$$\underline{u}\left(\underline{x},\, t\right) = \sum_{i=1}^{R} a_i\left(t\right) \underline{\Phi}_i\left(\underline{x}\right) \tag{6.1}$$

Using the modes, we derive the ROM as described in section 4.2 and get a system of ODEs describing the time evolution of the retained time-dependent POD coefficients $a_k^{(n)}, \quad k = 1, 2, ..., R$. At each time step, we can derive an error term $C_k^{(n+1)}$ between the one time step integrated coefficient from the ROM $G\left(a_k^{(n)}\right)$ and the real coefficient at the next time step derived from the reference solution $a_k^{(n+1)}$:

$$C_k^{(n+1)} = a_k^{(n+1)} - G\left(a_k^{(n)}\right) \tag{6.2}$$

We then train the NN to learn the mapping from true coefficients $a_k^{(n)}$ to the correction term $C_k^{(n+1)}$ as used by [20]:

$$\left\{a_1^{(n)}, a_2^{(n)}, ..., a_R^{(n)}\right\} \in \mathbb{R}^R \rightarrow \left\{C_1^{(n+1)}, C_2^{(n+1)}, ..., C_R^{(n+1)}\right\} \in \mathbb{R}^R \tag{6.3}$$

Finally, we reconstruct the state vector using the predicted coefficients and the POD basis (Eq. 4.13). There are only R coefficients available for reconstruction and thus, the state vector is restricted to a reduced order space. Reconstruction of the solution only in reduced space

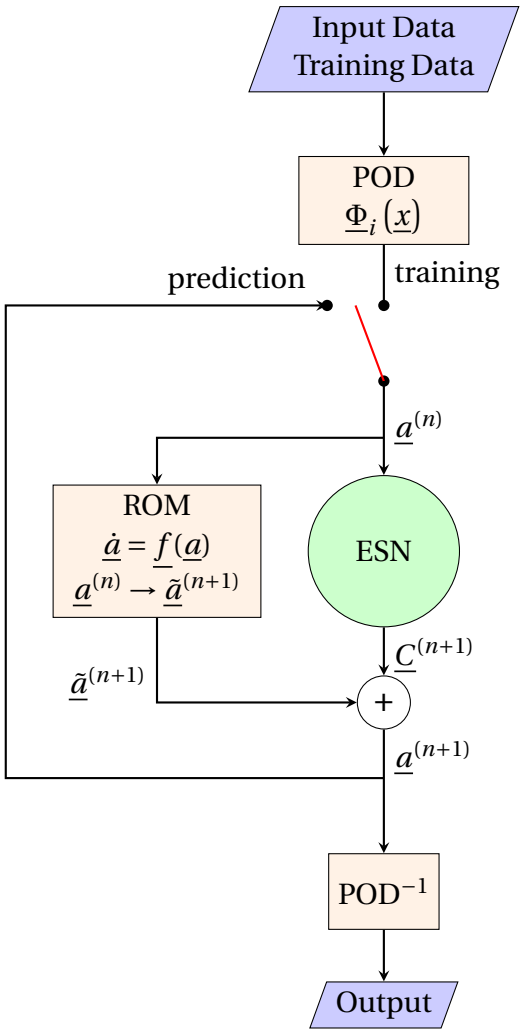

Figure 6.1: First hybrid architecture.

may be too restrictive for ODEs since the dimension is often too small to further reduce it. For example, the dimension of the Lorenz system is 3 only. However, the dimension of POD with solutions of PDE is the number of grid points times the number of variables which is a huge number and the reconstruction in reduced space may be accurate enough.

6.2 Method 2

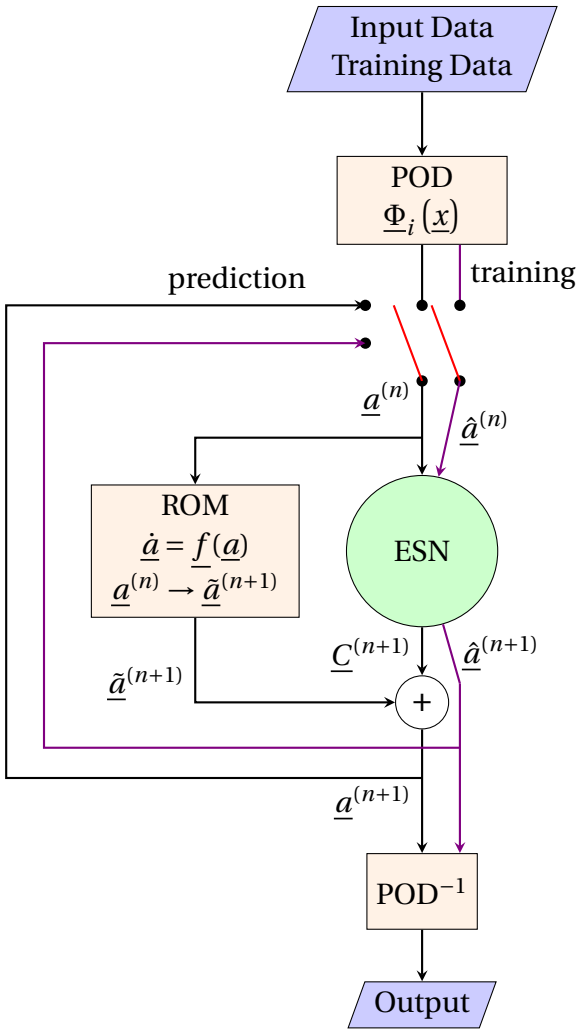

Figure 6.2: Second hybrid architecture.

Method 2 tries to compensate for the dimension problem of method 1 with ODE. We augment the mapping Eq. (6.3) and incorporate prediction of the skipped coefficients $a_{R+1}^{(n+1)}, ..., a_n^{(n+1)}$. The additional parameters can be seen in Fig. 6.2 at the violet lines. However, this method is

only useful for small dimensional problems like ODE. The new mapping reads:

$$\begin{bmatrix} a_1^{(n)} \\ \vdots \\ a_R^{(n)} \\ \hline a_{R+1}^{(n)} \\ \vdots \\ a_n^{(n)} \end{bmatrix} \in \mathbb{R}^n \rightarrow \begin{bmatrix} C_1^{(n+1)} \\ \vdots \\ C_R^{(n+1)} \\ \hline a_{R+1}^{(n+1)} \\ \vdots \\ a_n^{(n+1)} \end{bmatrix} \in \mathbb{R}^n \tag{6.4}$$

The upper part of the mapping Eq. (6.4) is equal to the mapping of hybrid method 1, however, we also concatenate the skipped coefficients. With this, we are able to reconstruct the solution at the end in full solution space since all coefficients are available.

6.3 Method 3

We use the same hybrid architecture as [19], however our knowledge based model is a ROM. Some adaptions to the ESN has to be made since the new state from the ROM is routed directly to the output layer. The exact architecture can be seen in Fig. 6.3. The three changes are summarized below:

- The new input vector is:

$$\underline{u}_{new}(n) = \begin{bmatrix} \underline{u}_{ROM}(n+1) \\ \underline{u}(n) \end{bmatrix} \in \mathbb{R}^{2n_u} \tag{6.5}$$

  Therefore, also the input matrix $\underline{\mathbf{W}}_{in} \in \mathbb{R}^{n_x \times 2n_u}$ changes its dimension. The reservoir size n_x stays the same.

- The new state from the ROM is concatenated with the state vector $\underline{\hat{x}}(x)$ in Eq. (5.7):

$$\underline{y}(n) = \underline{\mathbf{W}}_{out} \begin{bmatrix} \underline{u}_{ROM}(n+1) \\ \underline{\hat{x}}(n) \end{bmatrix} \tag{6.6}$$

  Therefore, also the input matrix $\underline{\mathbf{W}}_{out} \in \mathbb{R}^{n_y \times n_u + n_x}$ changes its dimension.

- In the training process in Eq. (5.9) $\underline{\hat{\mathbf{X}}}$ has to be replaced by the concatenated vector in Eq. (6.6).

Because of the direct connection between the input and the output layer, the ESN is able to learn a weighting between ROM prediction and ESN prediction of the next time step.

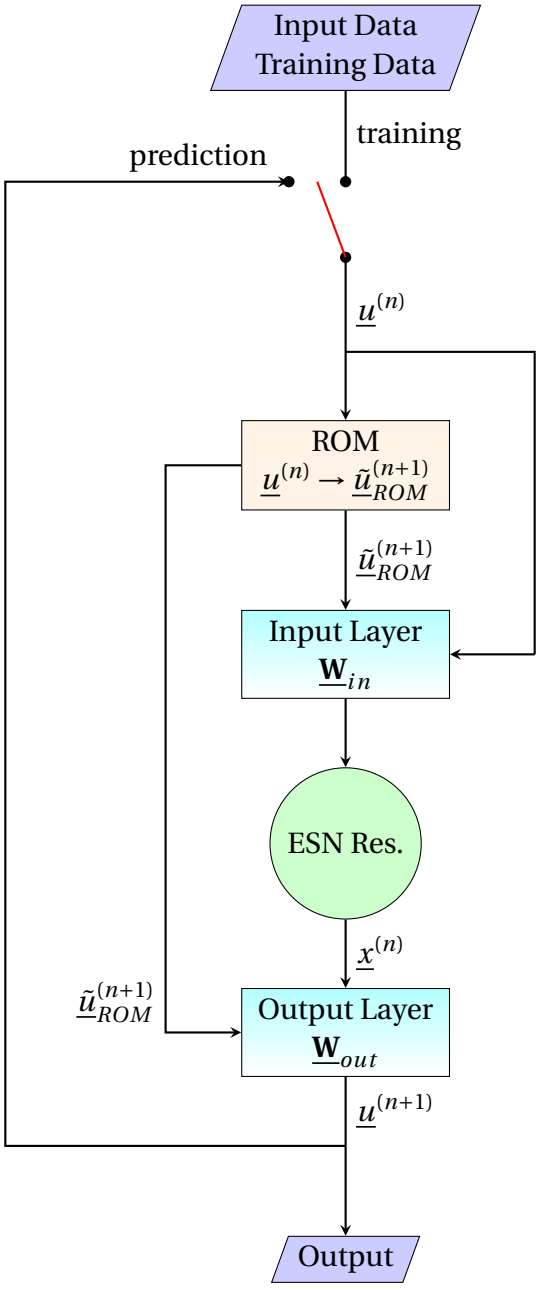

Figure 6.3: Third hybrid architecture.

7 Results

7.1 Pure Data Driven

First, we test the data-only approach to provide a benchmark against which the performance of the hybrid methods is assessed. The results for a single prediction are presented once at the beginning for the Lorenz system. Although a single prediction is not meaningful, it helps to understand the evolution of the error $E(t)$ and to see what is calculated next. Afterwards, a parameter search on a statistical basis is performed for all systems and a reservoir size study is conducted with the obtained parameters. Afterwards, in section 7.2, hybrid methods are going to be tested.

7.1.1 Lorenz System

We asses here the performance of the ESN with the Lorenz system. We used Runge Kutta 4 (sec. (3.2)) with a time step $0.01s$ to create a reference solution and use the obtained data to train the ESN and to asses the prediction performance. Starting from the initial conditions,

$$\underline{x}_0 = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix} \tag{7.1}$$

the initial transients, the first 20 seconds, were removed from our data. Table 7.1 summarizes the parameters for one prediction interval of 10 seconds. The data was normalized with the standard deviation and the mean was subtracted. 12500 samples were used to train the ESN, which is equal to a training length of 112.5 Lyapunov times with a step size of 0.01 seconds and a Lyapunov exponent of $\lambda = 0.9$. Table 7.1 summarizes the ESN parameters for the non-statistical single prediction.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	4.11	1.0	0.6	0.58
β	training samples	normalization	resync steps	washout
0.00001	12500	Stddev	100	100

Table 7.1: Parameters for single ESN prediction of the Lorenz System.

Fig. 7.1 shows the prediction of the ESN compared with the reference solution for 10 seconds (= 9 Lyapunov times) and the time dependent error $E(t)$. The red line in both pictures depicts

the valid time of 8.04 Lyapunov times. It is defined as the time where the error is smaller than 0.2.

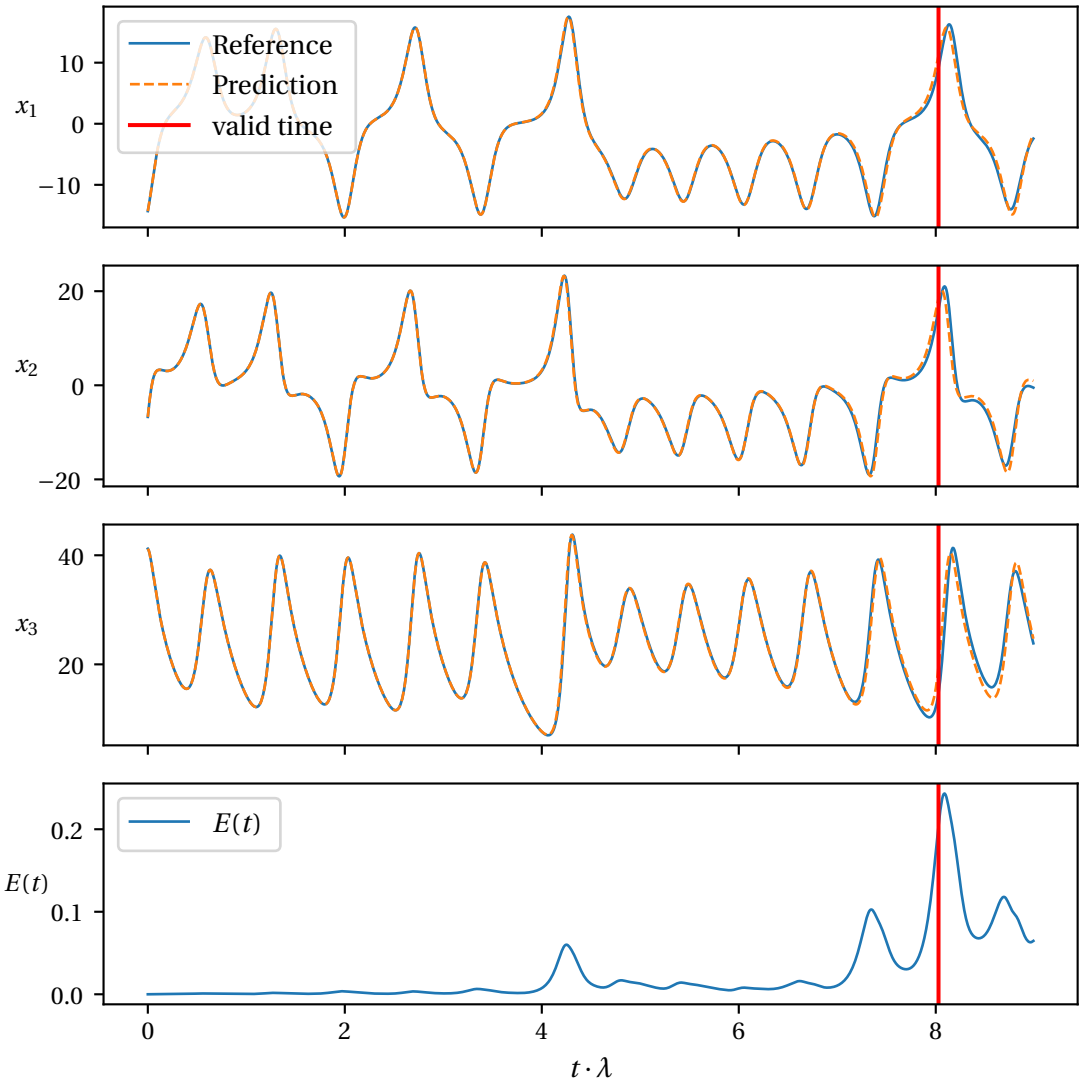

Figure 7.1: Comparison between reference solution and ESN prediction for the Lorenz system.

To estimate the overall statistical performance, we chose 50 intervals with a length of 10 seconds and computed the average valid time. For a reservoir with 500 neurons, the parameter search algorithm described in section 5.2 found that the parameters summarized in Table 7.2 yielded the best performance with a mean of 5.42 Lyapunov times and a standard deviation of 1.07 Lyapunov times.
With the parameters from the optimization for a reservoir of 500 neurons, the reservoir size was varied from 100 to 1000 neurons with 100 neuron steps. Fig. 7.2 shows the average prediction time in Lyapunov times ($\lambda \cdot t$)and the corresponding standard deviation for different

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	3.3	1.022	0.6	0.5499
β	training samples	normalization	resync steps	washout
0.00001	12500	stddev	100	100

Table 7.2: Parameters for ESN prediction of the Lorenz system.

reservoir sizes. We observe a slight increase in prediction time with increasing reservoir size. However, since the parameters are optimized for a reservoir with 500 neurons, the ESN's performance there was sometimes better than with larger reservoir sizes.

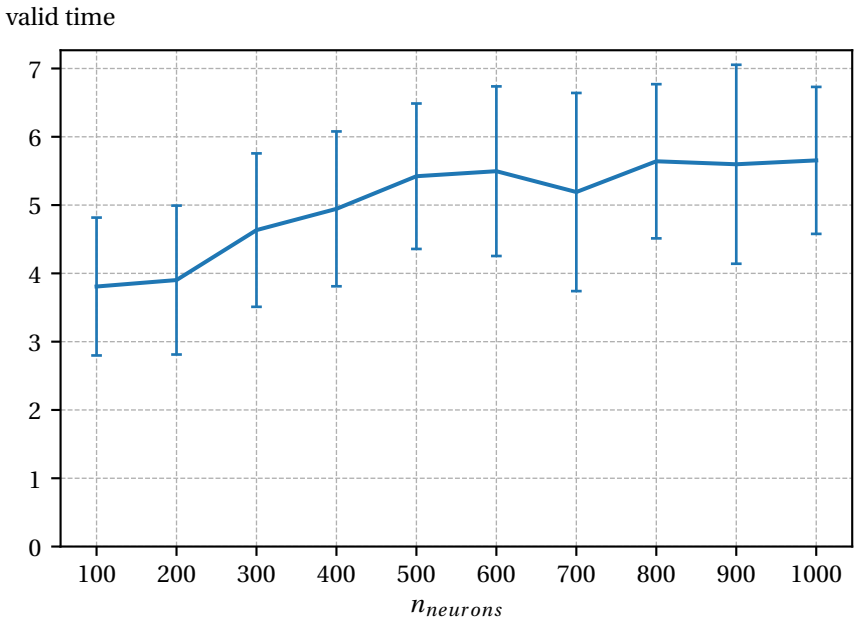

Figure 7.2: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$.

The reservoir initialization being random, an additional reservoir size study on a statistical basis was performed. Fig. 7.3 shows the results for the ESN prediction with 20 random initializations that are tested on 25 different regions in the reference solution. Since the Lorenz system is a low dimensional system, we observe a relatively small increase in prediction time. The ESN is able to learn the dynamics of the system already with small reservoir sizes. However, in the following, the ESN applied to the more complex systems showed a strong increase in prediction time as it requires a higher dimensional internal state of the ESN to reconstruct the solution.

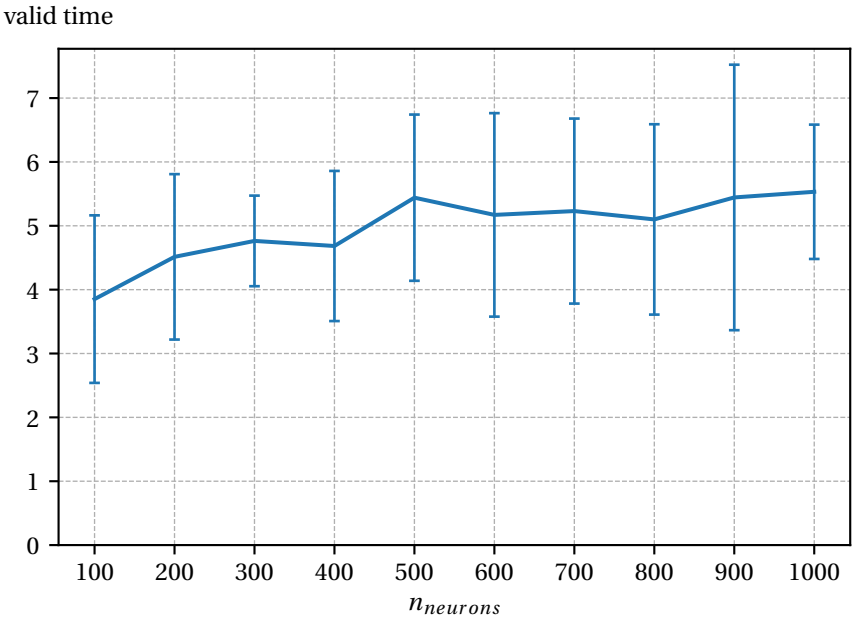

Figure 7.3: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ for 20 random reservoir initializations.

7.1.2 Platt System

The Platt system was integrated using RK4 with a time step of $0.01s$. The obtained reference solution can be seen partially in section 2.1.2. We removed the first $500s$ from our training data to remove the initial transients. 100 Lyapunov times were used to train the ESN. With a Lyapunov exponent of 0.11, we have 113636 samples to train the ESN. The first 100 steps were removed while training the ESN and use 99 samples(=1 Lyapunov time) to initialize the reservoir before a prediction starts. We conducted a parameter search on an ESN with 500 neurons and an error of 0.4. The obtained parameters are summarized in Table 7.3.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	3.7	0.9278	0.35	0.5
β	training samples	normalization	resync steps	washout
0.00001	113636	Stddev	99	100

Table 7.3: Parameters for ESN prediction of the Platt system.

Using the parameters from Table 7.3, the prediction performance with varying reservoir size is depicted in Fig. 7.4. The valid prediction time in Lyapunov times ($\lambda \cdot t$) for an error of 0.4 was increasing with increasing number of neurons and the best performance was at $n_{neurons} =$ 900, near the maximum reservoir size in this study. The ESN requires more neurons to accurately predict the next time step due to the difficult dynamics of the Platt system. Especially

the smaller reservoirs were not able to learn the dynamics of the system. A particularly strong increase can be observed from 100-500 neurons as the ESN approaches reservoir sizes for the first time that are able to reflect the system's dynamics. This is because the ESN requires a reservoir that can reproduce both the fast and slow intermittent dynamics.

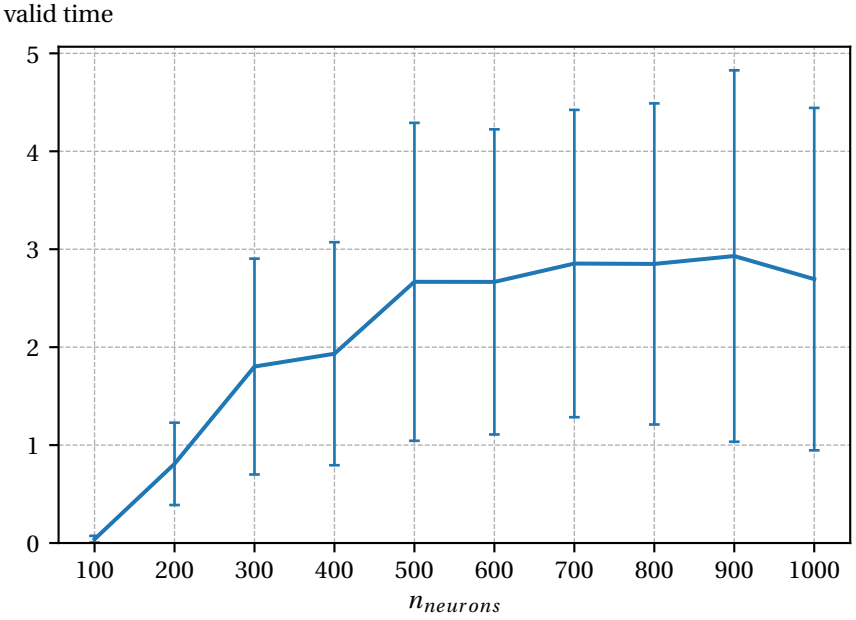

Figure 7.4: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$.

7.1.3 Charney-DeVore System

The CDV system was integrated using RK4 and a reference solution was obtained as shown in section 2.1.3. The first 500s, the initial transients, were removed from the training data and the ESN was trained with 100 Lyapunov times. With a Lyapunov exponent of 0.02 and a time step of 0.1s, the training set contains of 62500 samples. The first 100 samples are removed during the training process and the reservoir is initialized with 500 samples for every new prediction section. Table 7.4 summarizes the results of the parameter search algorithm for an ESN with a reservoir size of 500 and a prediction error of 0.4.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	3.5	0.8778	0.64999	0.7
β	training samples	normalization	resync steps	washout
0.00001	62500	Max	500	100

Table 7.4: Parameters for ESN prediction of the CDV system.

Fig. 7.5 shows the prediction time and standard deviation in Lyapunov times ($\lambda \cdot t$) for ESN only prediction with increasing reservoir size. The average prediction time and the respective standard deviation are calculated from 50 predictions in different regions of the reference solution. An increase in prediction time with increasing reservoir size could be seen. The maximum prediction time occurs for a reservoir with 500 neurons since the ESN is optimized to this reservoir size. We see similarities in the prediction performance with increasing reservoir size between Platt system and Charney-DeVore system. Both systems show small prediction performance with small reservoirs due to the higher dimensionality than Lorenz system. With increasing number of neurons, the ESN has learned the high-dimensional dynamics more accurately and therefore a strong increase in prediction time could be observed.

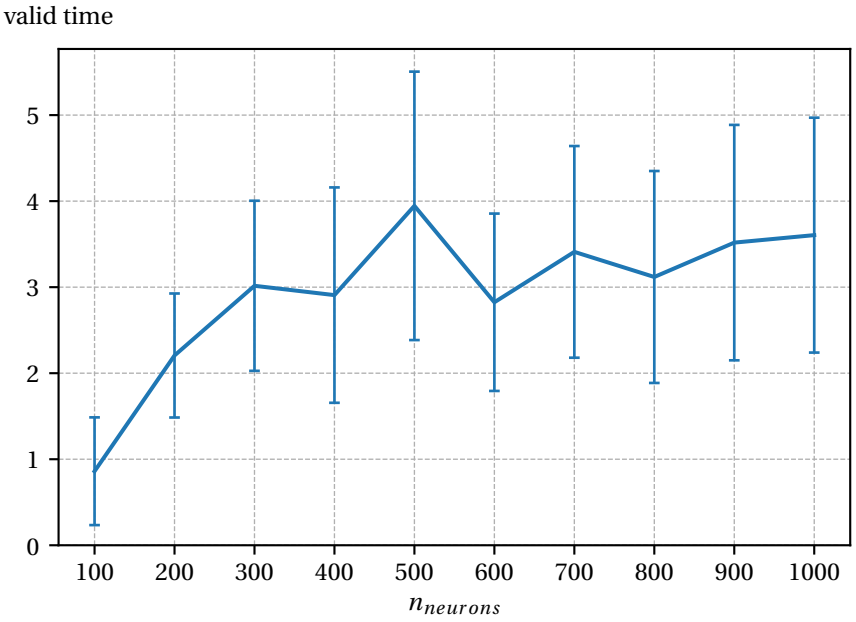

Figure 7.5: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$.

7.1.4 Kuramoto-Sivanshisky Equation

The parameters for the pure data-driven method are summarized in Table 7.5. We used 100 Lyapunov times to train the neural network which results in 5714 samples with a time step of 0.25 and the largest Lyapunov exponent is 0.07. We optimized the parameters for a reservoir with 1000 neurons and found that normalizing the data with the standard deviation showed the best prediction time with the smallest standard deviation.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
1000	3.5	1.05	0.1	0.3
β	training samples	normalization	resync steps	washout
0.00001	5714	Stddev	57	100

Table 7.5: Parameters for ESN prediction for KS equation.

Fig. 7.6 shows the prediction time and standard deviation in Lyapunov times ($\lambda \cdot t$) for ESN only prediction with increasing reservoir size for the KS equation. The average prediction time and the respective standard deviation are calculated from 50 predictions in different regions of the reference solution. The prediction time increased with increasing reservoir size. This is similar to the results from CDV system or Platt system. However, the system is much more complex. Compared to the systems before with 3, 5 or 6 degrees of freedom (DoF), the KS system has 64 DoF since this system is obtained by discretizing a PDE. The overall prediction time is smaller but that was to be expected as it is more complex.

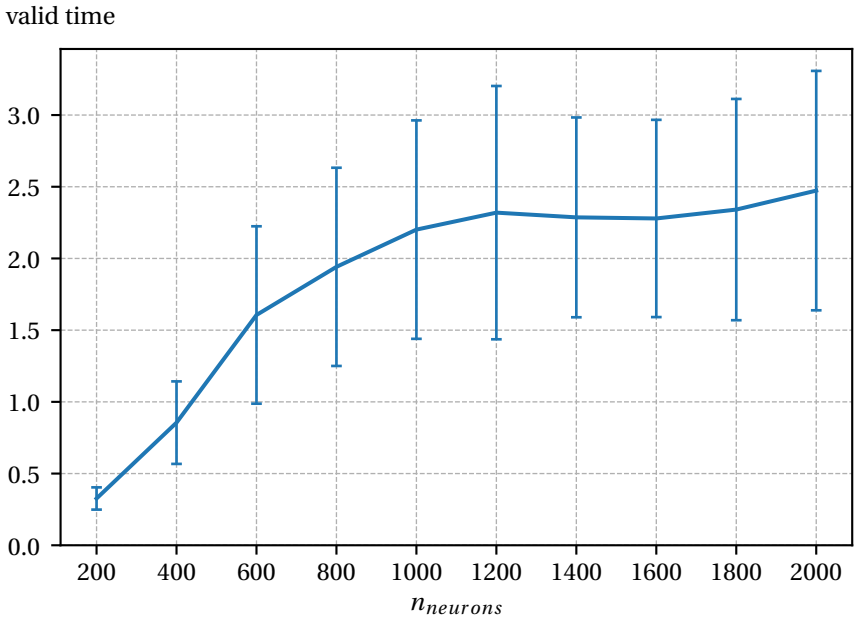

Figure 7.6: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$.

7.2 Hybrid Methods

In the previous section, we investigated the pure ML approach. We saw the ESN's ability to learn the dynamics and its increasingly better performance with the use of larger reservoirs. Now, we try to improve that by using physics-based ROMs together with an ESN. The three hybrid methods are tested, their performances are compared to the pure ML approach and with each other. Reservoir size studies are also conducted and the accuracy of the ROM for the KS equation is varied. The same training data and the same parameter search algorithm is used as for the pure ML approach for all systems. The results are presented in the following sections.

7.2.1 Lorenz System

This section investigates the different hybrid architectures and the ROM of the Lorenz system. We are interested in the prediction time where the error between the predicted and reference solution is smaller than 0.4. The average time and standard deviation of 50 predictions of $10s$ length in different regions of the reference solution is monitored. We used $n_{neurons} = 500$ as a reservoir size for each parameter search. We skipped the third mode with the smallest eigenvalue and use the first two modes for the ROM.
After good parameters have been found, we investigated the prediction performance for different reservoir sizes. The reservoir size was increased from 100 to 1000 in 100 neurons intervals for fixed parameters. For the Lorenz system, due to computational simplicity, additional analysis with and without different random initializations of the reservoir was conducted.

ROM only

First, we assess the performance of POD-only based ROM. After POD decomposition, we order the eigenvalues and corresponding POD modes in descending order and divide each eigenvalue λ_i by the sum of all eigenvalues to see the relative contribution of each mode to the total energy. Table 7.6 shows the contribution of each mode.

	$\frac{\lambda_i}{\sum \lambda_i}$
λ_1	81.35%
λ_2	17.53%
λ_3	1.12%

Table 7.6: Relative contribution of modes.

While the ROM with all modes was able to operate stably and resembled the exact dynamics of the Lorenz system as can be seen in Fig. 7.8, skipping one mode already leads to a complete prediction failure of the ROM. Figure 7.7 shows the time evolution of the ROM using only the first two modes. The ROM was not able to capture the Lorenz system's dynamics and converged to a fixed point.

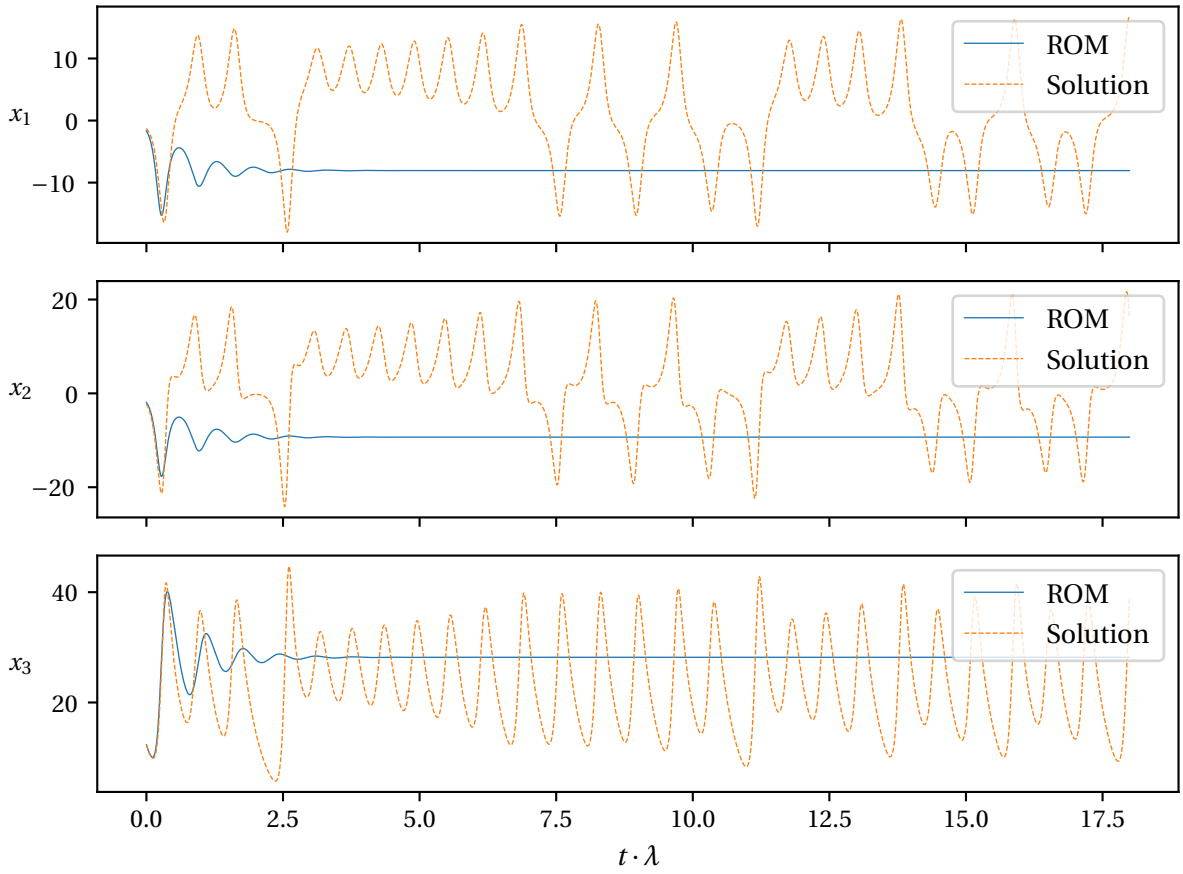

Figure 7.7: ROM only prediction with one skipped mode.

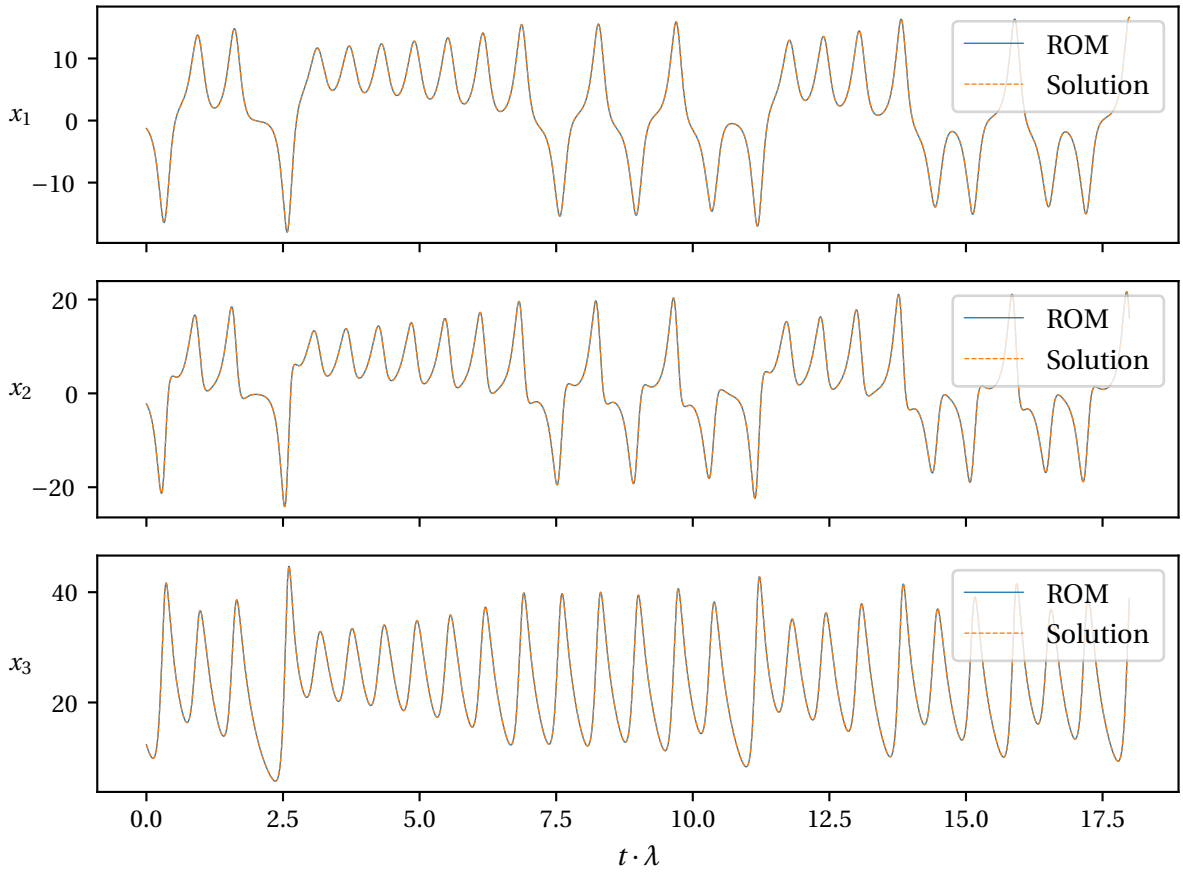

Figure 7.8: ROM only prediction with all modes incorporated.

Hybrid1

Table 7.7 summarizes the hyper-parameters found by the parameter study for the first hybrid method applied to the Lorenz system.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	3.7	0.86	0.42	0.805
β	training samples	normalization	resync steps	washout
0.00001	12500	Max	100	100

Table 7.7: Parameters for hybrid1 method of the Lorenz system.

Fig. 7.9 shows the average prediction time with respective standard deviation for increasing reservoir size in Lyapunov time units ($= \lambda \cdot t_{valid}$).

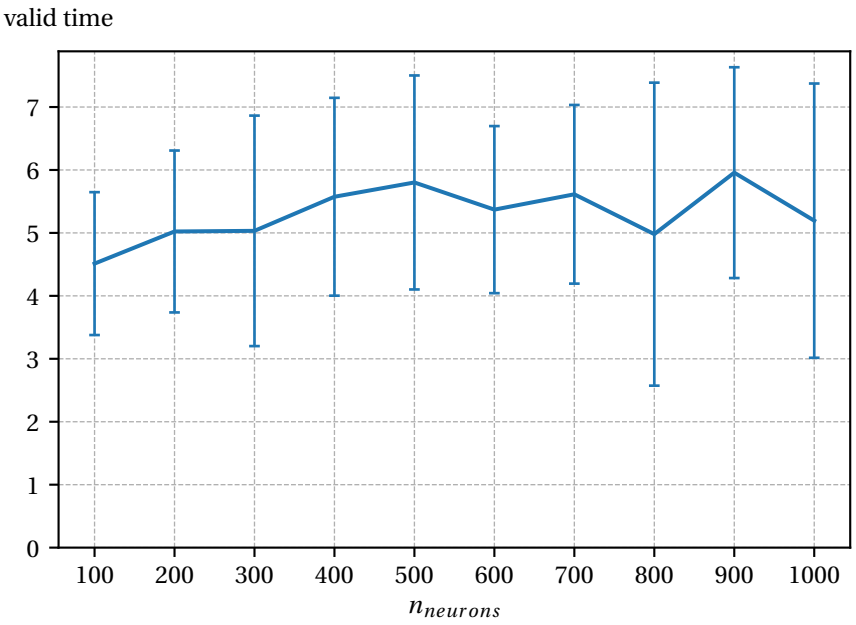

Figure 7.9: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 1.

Fig. 7.10 shows the average prediction time with respective standard deviation for increasing reservoir size in Lyapunov time units for the statistical study. The non-statistical and statistical study showed similar results.

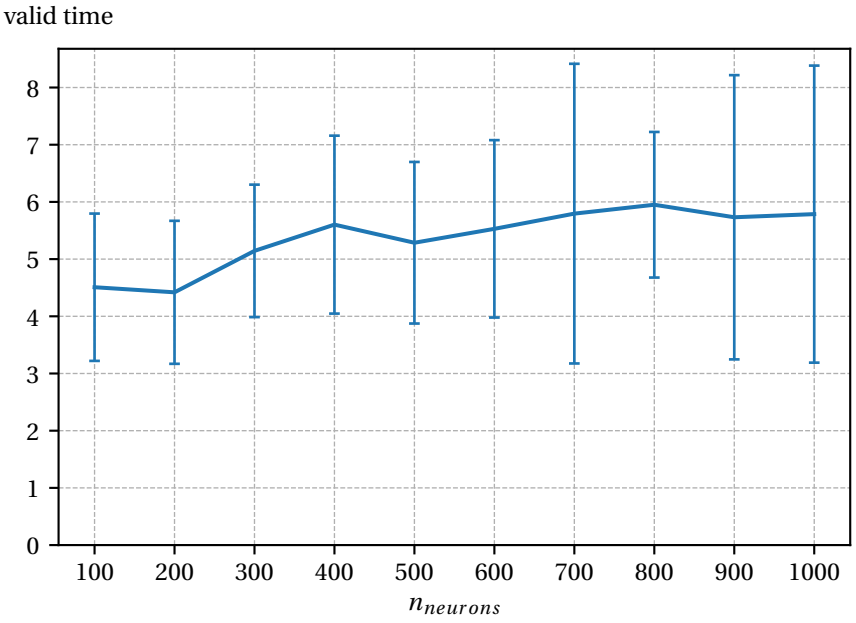

Figure 7.10: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ for 20 random reservoir initializations using hybrid 1.

Hybrid method 1 showed similar results as the pure data-driven method. A similar increase in prediction performance with increasing reservoir size could be observed as with the pure ML method and the results were negligibly better. The reconstruction of the solution in the reduced space seems to be too restrictive for low dimensional systems. The ESN's task to predict the skipped dynamics in the ROM could be as challenging as the prediction of the dynamics of whole the system.

Hybrid2

Table 7.8 summarizes the hyper-parameters found by the parameter study for the second hybrid method applied to the Lorenz system.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	3.3	0.9299	0.5	0.5449
β	training samples	normalization	resync steps	washout
0.00005	12500	Stddev	100	100

Table 7.8: Parameters for hybrid2 method of the Lorenz system.

Fig. 7.11 shows the average prediction time with respective standard deviation for increasing reservoir size. Fig. 7.12 shows the average prediction time with respective standard deviation for increasing reservoir size in Lyapunov time units for a statistical study with 20 random

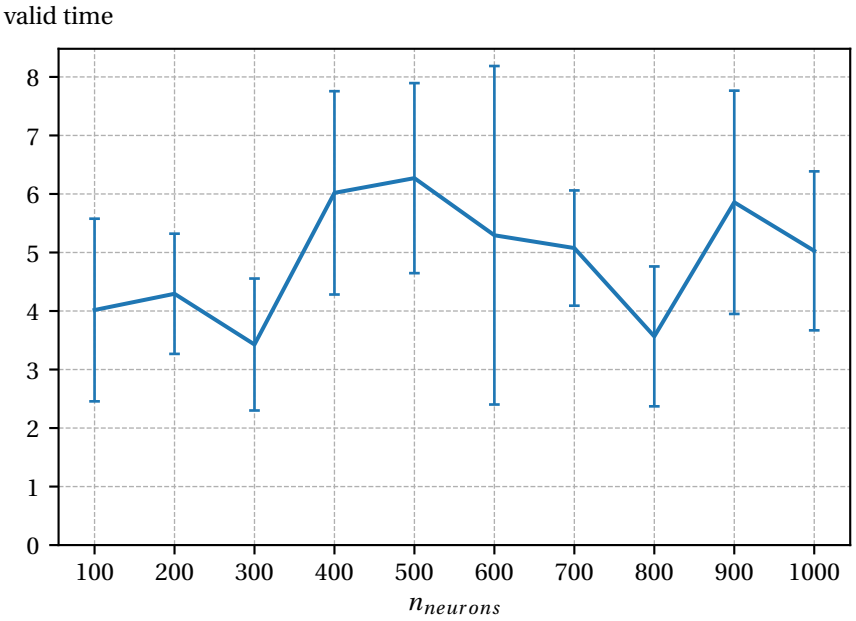

Figure 7.11: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 2.

reservoir initializations. Both, the statistical and non-statistical study showed strong variations in prediction time and standard deviation. The best result in the non-statistical case was at the design point with 500 neurons whereas reservoirs with 700 or 800 neurons delivered same valid times for the statistical case. Since the non-statistical and statistical results differ, it can be concluded that this method is strongly dependent on the hyper parameters and the random initialization. In addition, the ESN has to predict two different things, the correction term and the skipped dynamics, which could have different time scales which makes it more challenging for the ESN. Indeed, the ESN is only excited with the retained modes. Therefore, it may be too challenging for the ESN to learn the skipped-mode dynamics from this.

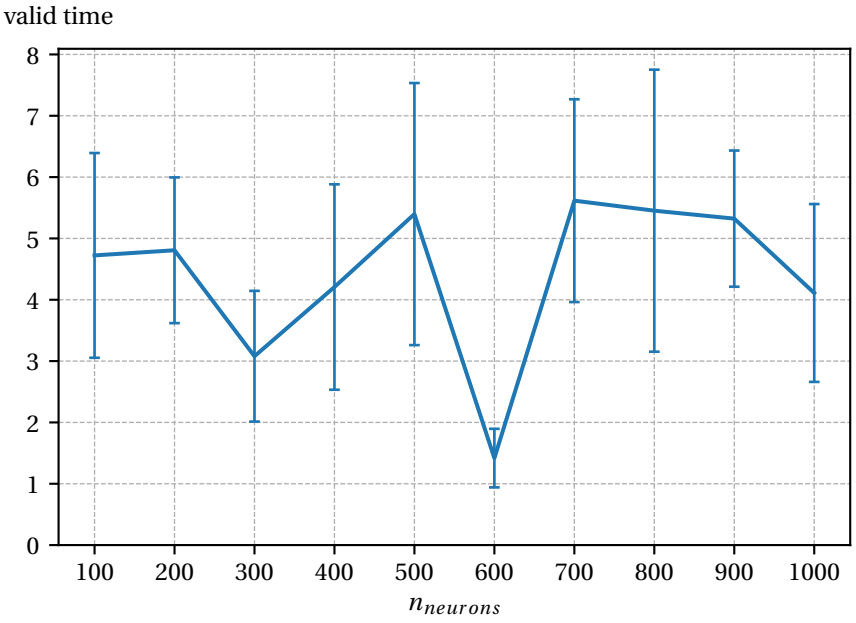

Figure 7.12: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ for 20 random reservoir initializations using hybrid 2.

Hybrid3

Table 7.9 summarizes the hyper-parameters found by the parameter study for the third hybrid method applied to the Lorenz system.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	2.8	0.878	0.2445	0.655
β	training samples	normalization	resync steps	washout
0.00005	12500	None	100	100

Table 7.9: Parameters for hybrid3 method of the Lorenz system.

Fig. 7.13 shows the average prediction time with respective standard deviation for increasing reservoir size. Fig. 7.14 shows the average prediction time with respective standard deviation for increasing reservoir size in Lyapunov time units for the statistical study. While the non-statistical study showed small fluctuations in prediction time with increasing reservoir size, the statistical study showed a flat but steady increase of prediction time with increasing reservoir size. This method outperformed the pure data-driven method.

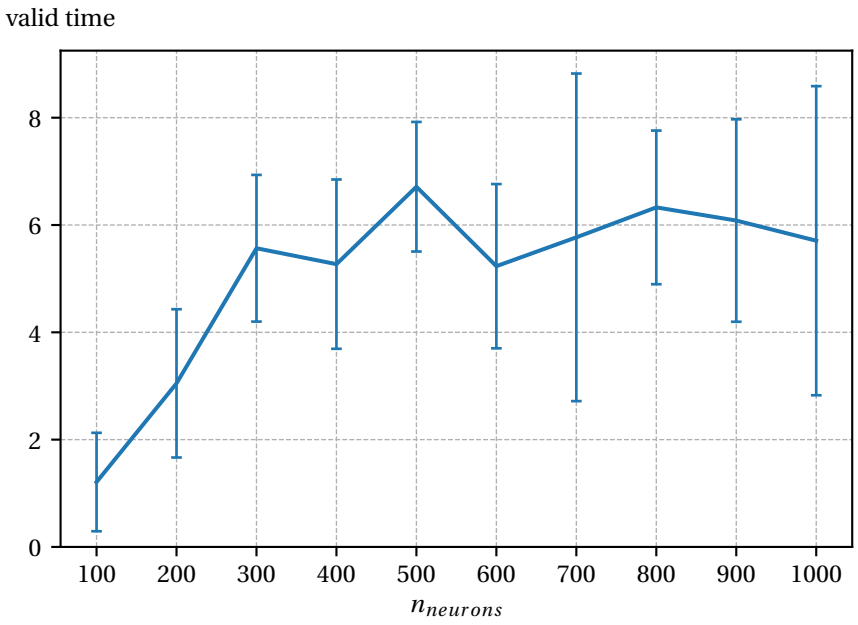

Figure 7.13: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 3.

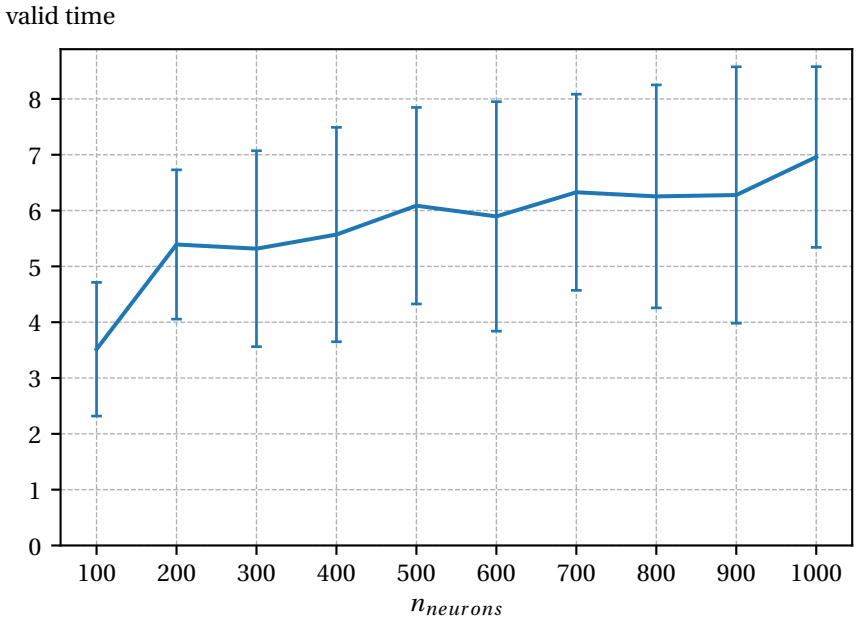

Figure 7.14: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ for 20 random reservoir initializations using hybrid 3.

7.2.2 Platt System

This section examines the prediction performance of the different hybrid methods applied to the Platt system which is particularly challenging due to the system's properties. The time when the error between predicted and reference solution gets bigger than 0.4 is measured for each method. Each parameter combination is tested on 50 different intervals of 10 Lyapunov times length and the average valid prediction time is monitored. We skipped the last mode with the smallest eigenvalue and use the remaining modes for the ROM. The parameters for ESN training are summarized in the tables below.
The prediction performance was investigated for different reservoir sizes with the listed parameters. The reservoir size was increased from 100 to 1000 in 100 neurons intervals for fixed parameters and each reservoir was tested at 50 intervals and the average valid time was plotted. The reservoir was only initialized once, thus, the results depend on the random creation of the reservoir.

ROM only

After a POD decomposition of the reference solution, we can derive the ROM and estimate the importance of each mode using the eigenvalues. The eigenvalues divided by the sum of all eigenvalues are summarized in Table 7.10.

	$\frac{\lambda_i}{\sum \lambda_i}$
λ_1	79.55%
λ_2	11.33%
λ_3	6.4%
λ_4	1.43%
λ_5	1.3%

Table 7.10: Relative contribution of modes.

We skip the last mode with eigenvalue λ_5 for the ROM and obtain a model with dynamics as depicted in Fig. 7.15. Compared to the other ODE systems, the ROM does not converge to a fixed point and is able to resemble some dynamical characteristics as can be seen for x_3. However, the on-off intermittency is not apparent anymore which was the systems special feature of interest for us.

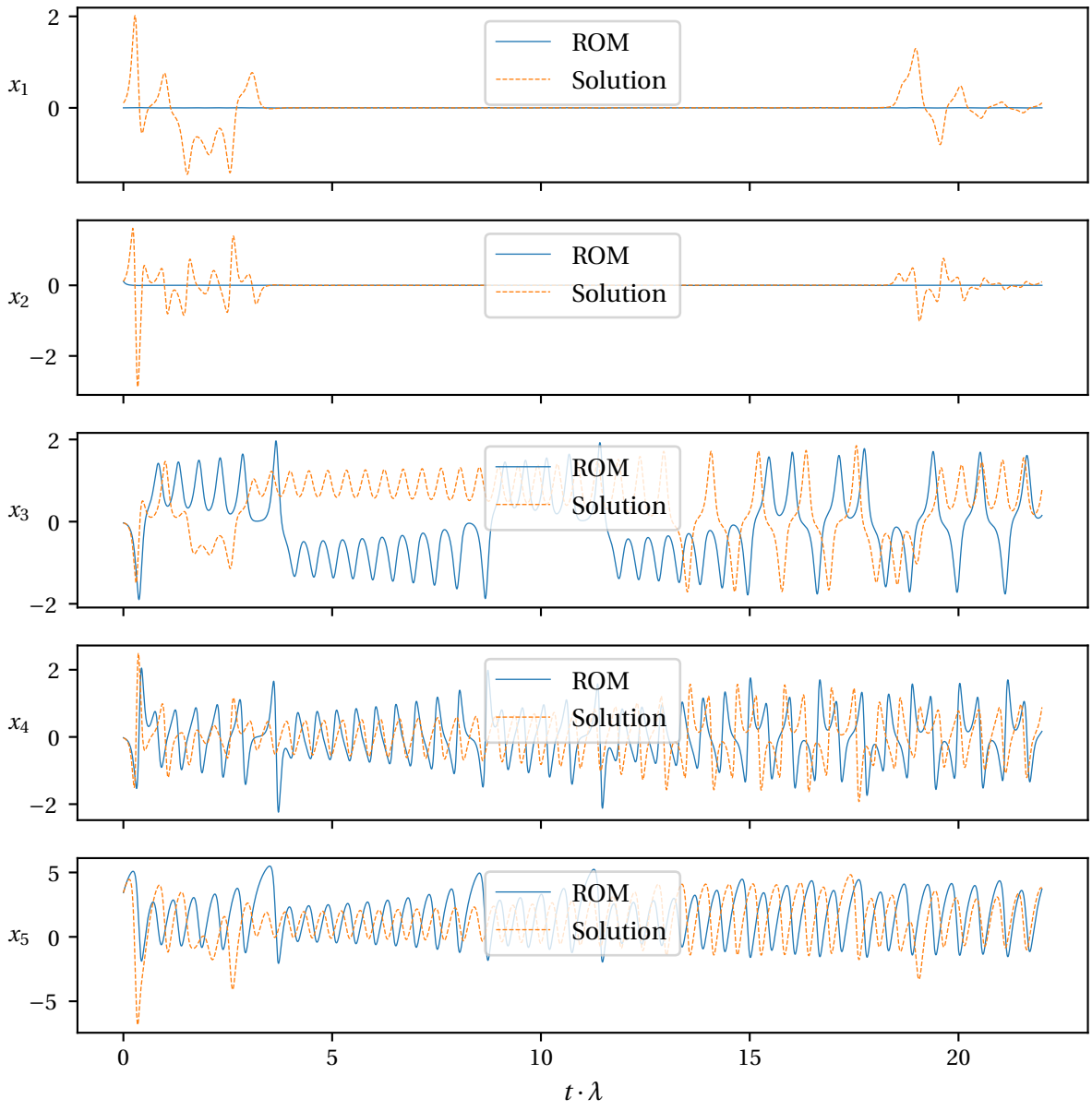

Figure 7.15: ROM only prediction with one skipped mode.

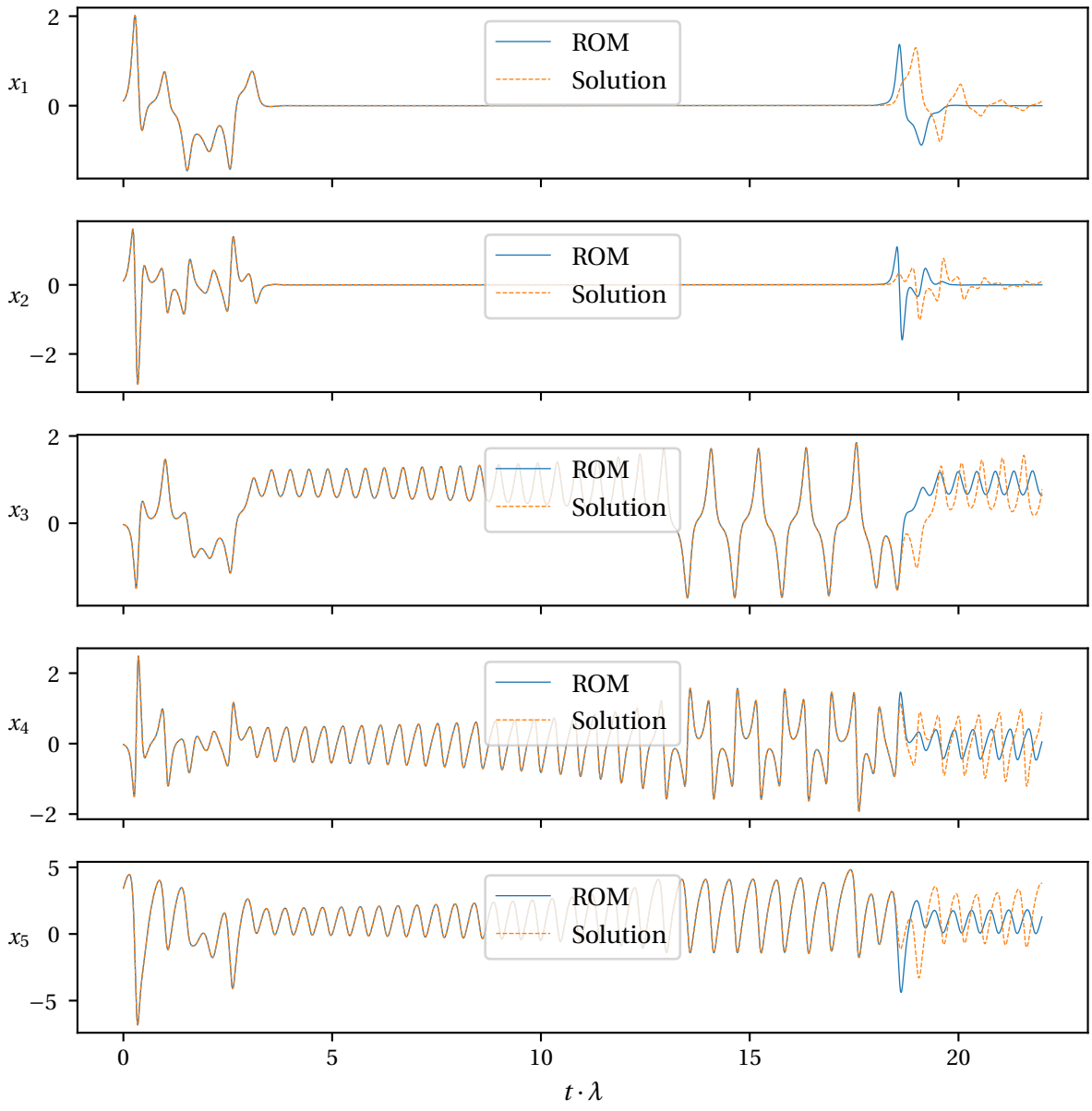

Figure 7.16: ROM only prediction with all modes incorporated.

Fig. 7.16 shows the ROM's dynamics using all modes. Even small changes in the numerical computation and round off errors can lead to discrepancies in the solution due to the sensitive response of the system in unstable regions. This can be seen in Fig. 7.16 at approximately 18 Lyapunov times, where the reference solution and the ROM solution diverge. This illustrates the extreme sensitivity of the Platt system which makes it extremely challenging to predict.

Hybrid1

Table 7.11 summarizes the hyper-parameters found by the parameter search for the first hybrid method applied to Platt system.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	3.5	0.7499	0.6399	0.7
β	training samples	normalization	resync steps	washout
0.00005	113636	Max	99	100

Table 7.11: Parameters for hybrid1 method of the Platt system.

Fig. 7.17 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size. The results did not show a consistent behavior. The valid prediction time in the design point $n_{neurons} = 500$ was maximal, however, the standard deviation was nearly as big. Compared to the Lorenz system, the reconstruction in the reduced space together with an inaccurate ROM seems to be too restrictive to capture any of the Platt system's complicated dynamics. The results for the CDV system and the Lorenz system are similar.

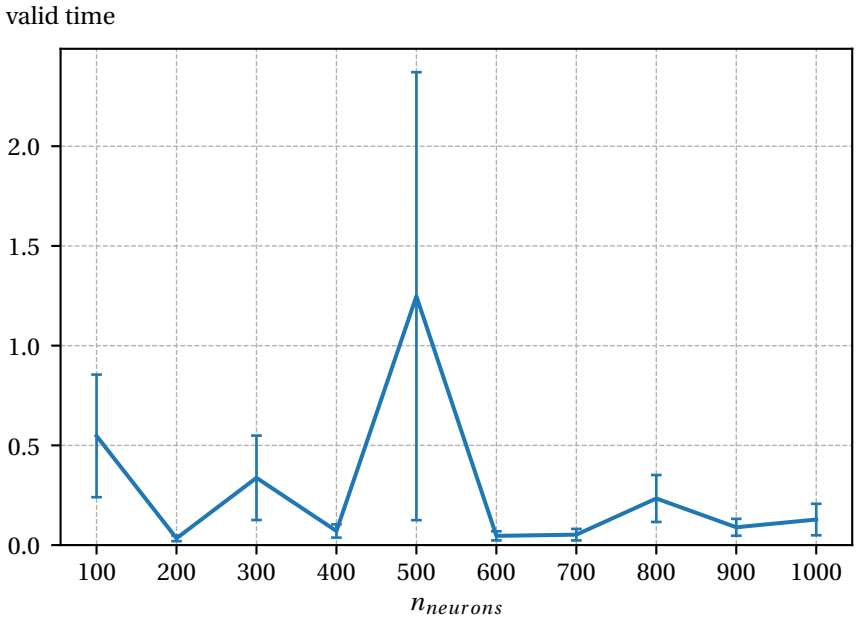

Figure 7.17: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 1.

Hybrid2

Table 7.12 summarizes the hyper-parameters found by the parameter search for the second hybrid method applied to Platt system.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	3.0	1.0	0.7	0.4
β	training samples	normalization	resync steps	washout
0.00005	113636	Max	99	100

Table 7.12: Parameters for hybrid2 method of the Platt system.

Fig. 7.17 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size. The results were constant for all reservoir sizes except for the smallest reservoir, 700 neurons and 800 neurons. Although, the prediction time stays constant, it is at higher values than the pure data-driven method. Reconstruction in the full solution space yielded in the case of the Platt system a significant increase in accuracy. Compared to the Lorenz system, this indicates that hybrid method 2 seems to be more accurate for larger systems.

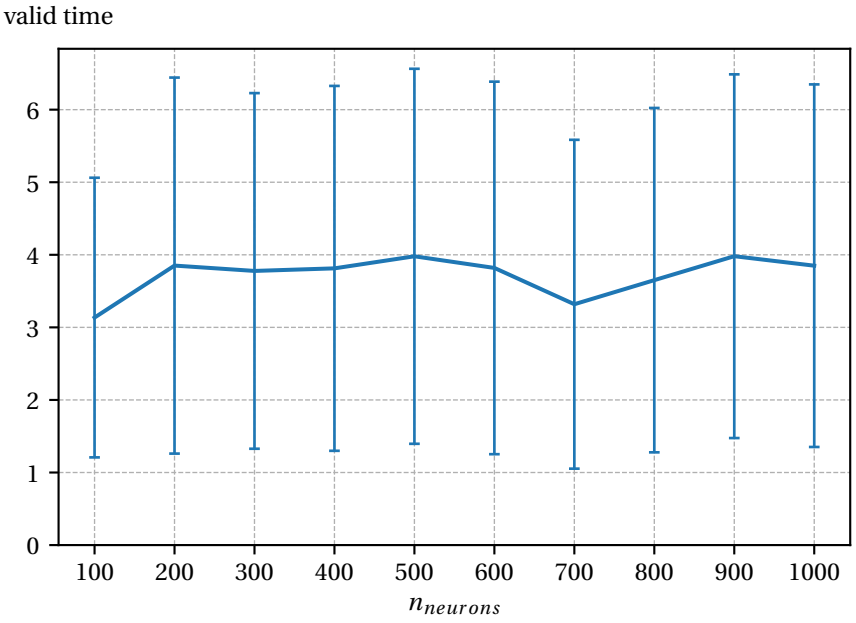

Figure 7.18: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 2.

Hybrid3

Table 7.13 summarizes the hyper-parameters found by the parameter search for the third hybrid method applied to Platt system.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	2.099	1.0	0.4	0.3
β	training samples	normalization	resync steps	washout
0.00003	113636	Max	99	100

Table 7.13: Parameters for hybrid3 method of the Platt system.

Fig. 7.19 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size. The valid prediction time slightly increased until it reached the design point with $n_{neurons} = 500$. Afterwards, the performance dropped and started to increase again after 700 neurons. The best performance was measured in the design point. Hybrid method 3 is able to decide whether more ROM or more ESN information contributes to the solution. The ROM of the Platt system retains, compared to the other ROMs derived from ODE systems, more dynamics. Therefore, the method decides to weight the ROM information more strongly and the proportion of the ESN scaling behavior is not strongly apparent in the results.

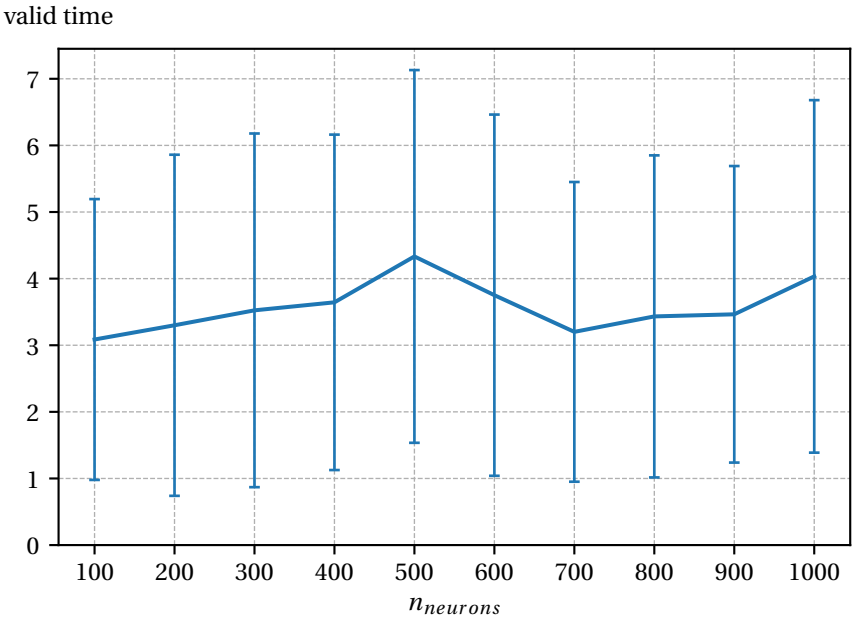

Figure 7.19: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 3.

7.2.3 Charney-DeVore System

This section investigates the different hybrid architectures and the ROM of the CDV system. The prediction time is again defined as the time where the error exceeds 0.4. Each parameter combination is tested on 50 different intervals of 10 Lyapunov times length and the average valid prediction time is monitored. We used $n_{neurons} = 500$ as a reservoir size for each parameter search and tested different normalization methods for ESN input data. We skipped the last mode with the smallest eigenvalue and use the remaining modes for the ROM.
The prediction performance for different reservoir sizes was tested and the reservoir size was varied from 100 to 1000 in 100 neurons intervals for fixed parameters. The reservoir was only initialized once, thus, the results depend on the random creation of the reservoir.

ROM only

The eigenvalues obtained by POD are normalized with the sum of all eigenvalues and summarized in Table 7.14.

	$\frac{\lambda_i}{\sum \lambda_i}$
λ_1	94.28%
λ_2	3.86%
λ_3	1.61%
λ_4	0.14%
λ_5	0.07%
λ_6	0.035%

Table 7.14: Relative contribution of modes.

Although the last eigenvalue does not contribute much to the total energy, the impact on the ROM's dynamics is severe. While a ROM with all modes incorporated is able to capture the CDV system's dynamics perfectly, Fig. 7.21, the ROM without the last mode already describes a completely different behavior that converges to a fixed point after 2 Lyapunov times.

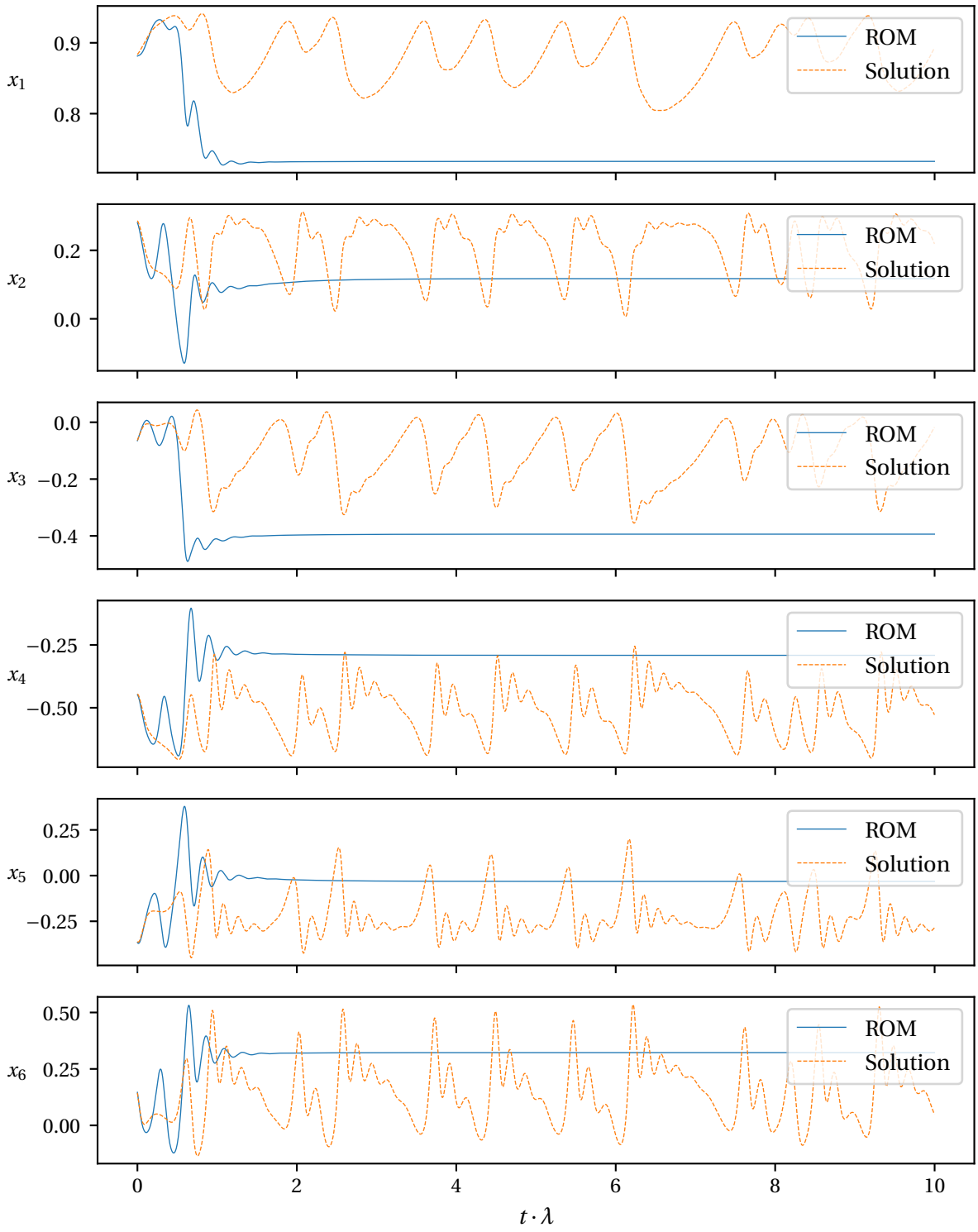

Figure 7.20: ROM only prediction with one skipped mode.

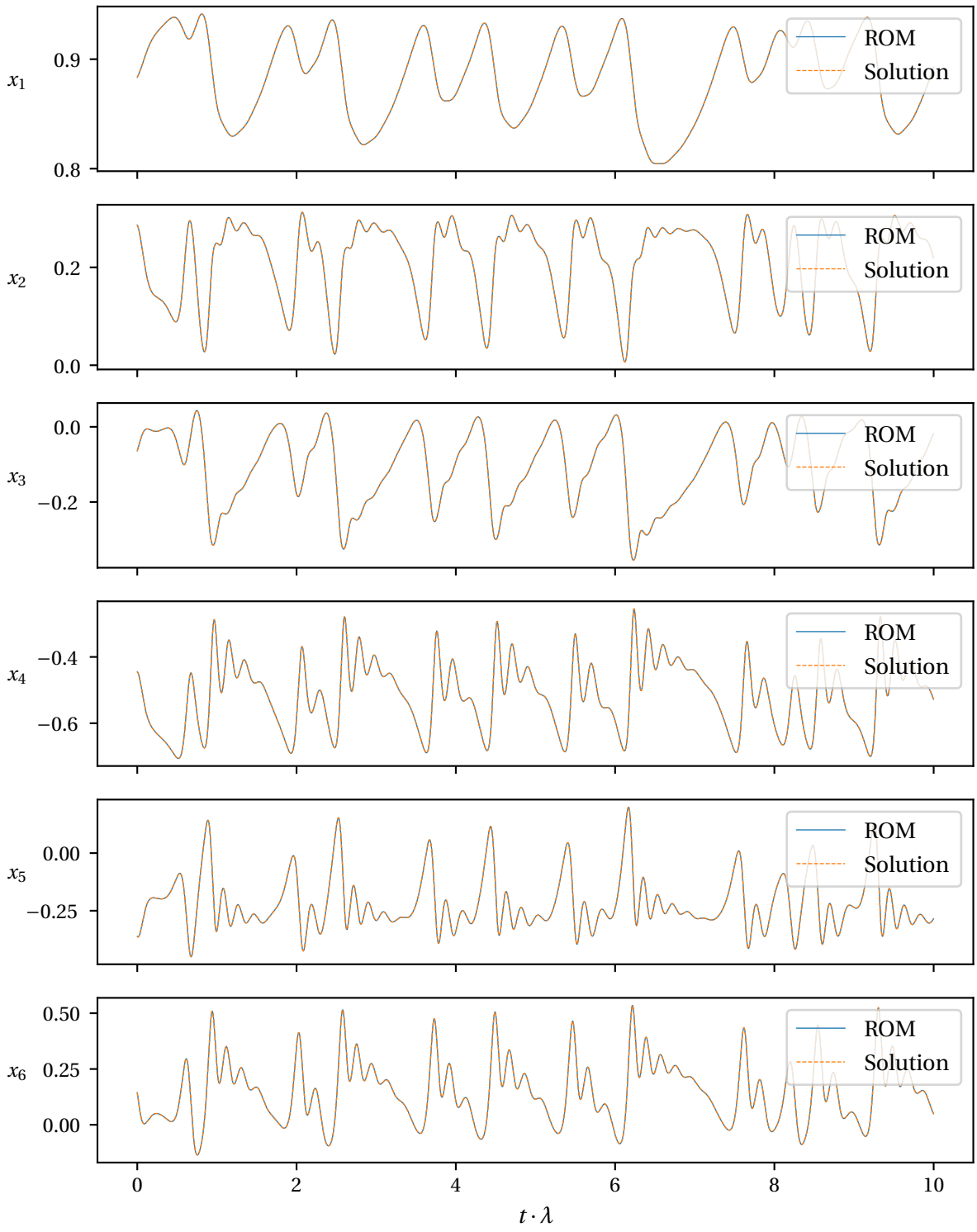

Figure 7.21: ROM only prediction with all modes incorporated.

Hybrid1

Table 7.15 summarizes the hyper-parameters found by the parameter search for the first hybrid method applied to CDV system.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	3.5	0.89	0.8	0.55
β	training samples	normalization	resync steps	washout
0.00005	62500	Max	500	100

Table 7.15: Parameters for hybrid1 method of the CDV system.

Fig. 7.22 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size. The valid time slightly increased at the beginning and kept constant with increasing reservoir size. The best prediction performance was found in the design point $n_{neurons} = 500$. Hybrid method 1 showed similar performance for the Lorenz system. The method is nearly as accurate as the pure data-driven method as the reconstruction in the reduced space is too restrictive. However, there was no total failure as in the case of the Platt system since the dynamics of the CDV system are not as complicated.

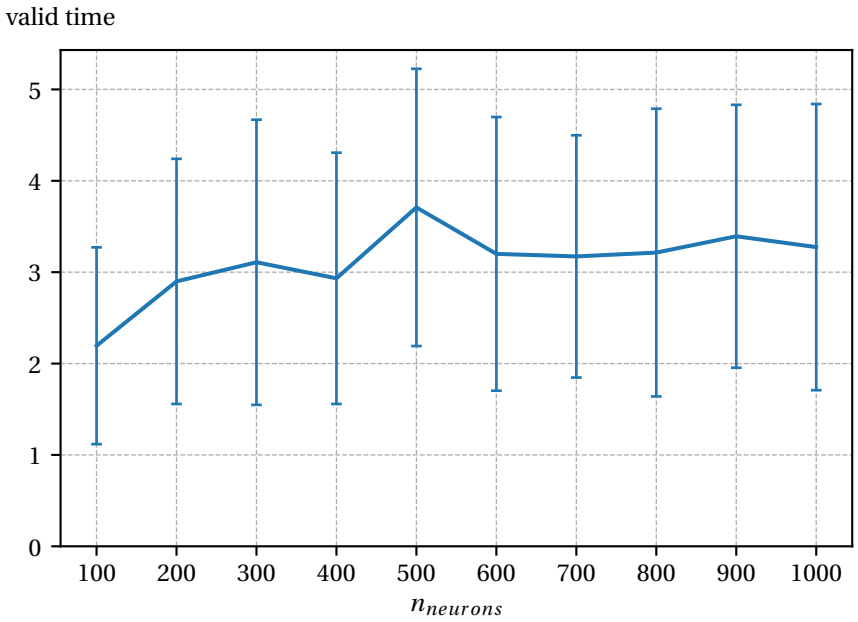

Figure 7.22: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 1.

Hybrid2

Table 7.16 summarizes the hyper-parameters found by the parameter search for the second hybrid method applied to CDV system.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	3.0	0.83	0.822	0.4
β	training samples	normalization	resync steps	washout
0.00001	62500	Stddev	500	100

Table 7.16: Parameters for hybrid2 method of the CDV system.

Fig. 7.23 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size. The valid time constantly increased with increasing reservoir size and had a maximum at the design point $n_{neurons} = 500$. Similar to the Platt system, hybrid method 2 provided significantly longer prediction times than the pure data-driven method confirming that is is only useful for systems of slightly larger dimension.

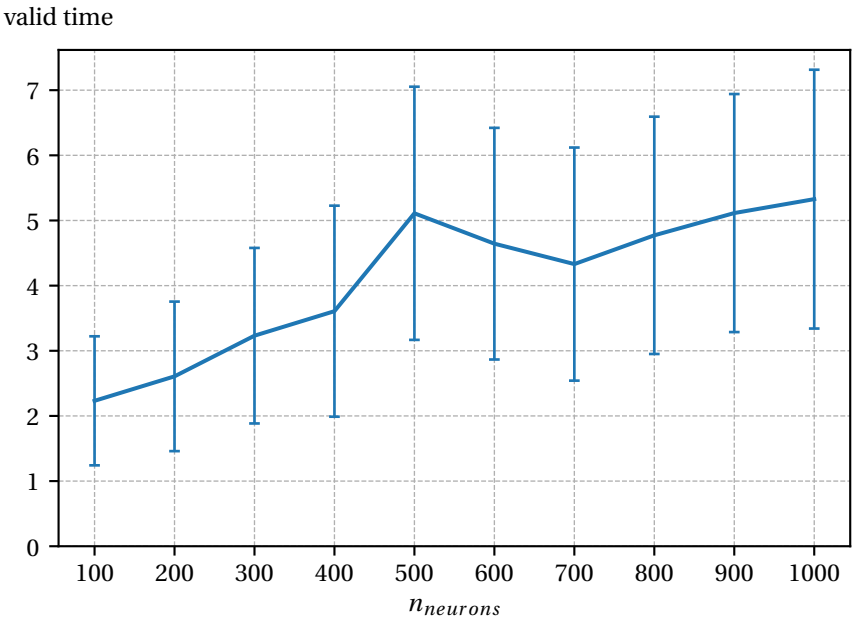

Figure 7.23: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 2.

Hybrid3

Table 7.17 summarizes the hyper-parameters found by the parameter search for the third hybrid method applied to CDV system.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	2.5	0.8099	0.40499	0.65
β	training samples	normalization	resync steps	washout
0.00001	62500	Max	500	100

Table 7.17: Parameters for hybrid3 method of the CDV system.

Fig. 7.24 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size. There was a slight but not steady increase in prediction time with increasing reservoir size. The valid time dropped after the design point $n_{neurons} = 500$ but increased again shortly afterwards. Similar to the Lorenz system, the method detected an inaccurate ROM and relied more on the ESN information which is why an increase in prediction time with increasing reservoir size could be observed.

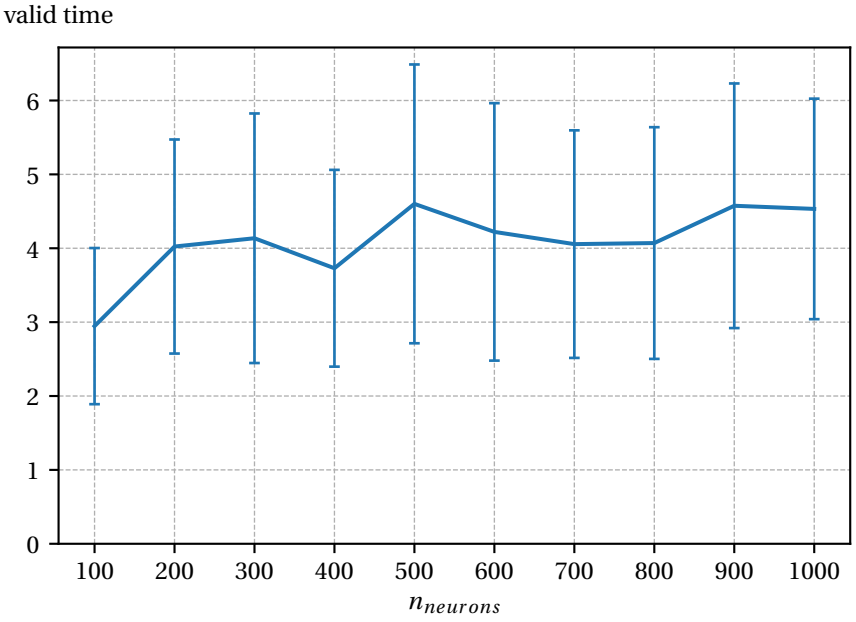

Figure 7.24: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ using hybrid 3.

7.2.4 Kuramoto-Sivanshisky Equation

This section deals with applying the proposed hybrid methods to a PDE, the KS equation. Each parameter combination is tested on 50 different intervals of 10 Lyapunov times length and the average valid prediction time, where the error exceeds 0.4, is monitored. We used $n_{neurons} = 500$ as a reservoir size for each parameter search and tested different normalization methods for the ESN input data. We performed the parameter search for different numbers of skipped modes. One with 35 skipped modes, with 40 skipped modes and with 45 skipped modes. Skipping more than 45 modes led to instabilities of the ROM.
The prediction performance was investigated for different reservoir sizes. The reservoir size was increased from 200 to 2000 in intervals of 200 neurons for fixed parameters. The reservoir was only initialized once, thus, the results depend on the random creation of the reservoir.
The prediction performance depending on the number of skipped modes was monitored in an additional study for each hybrid method. The parameters from the parameter search with 35 skipped modes in combination with an ESN with 500 neurons were used and the number of skipped modes was varied from 35 to 45. Again, each parameter combination was tested on 50 different intervals of 10 Lyapunov times length and the average prediction time was monitored.

ROM only

We investigated the behavior of the ROM depending on the number of used modes. Since we deal with an PDE, we have a spatial discretization that determines the number of grid points and thus the number of input variables. We used 64 grid points which yielded 64 eigenvalues in the POD decomposition. How much energy each mode contributes to the total energy can be seen in Fig. 7.25. The relative contribution of each eigenvalue rapidly decreases and reaches already values near 0% with eigenvalue 21. However, we found that it is necessary to use at least 17 modes to ensure the stable behavior of the ROM.

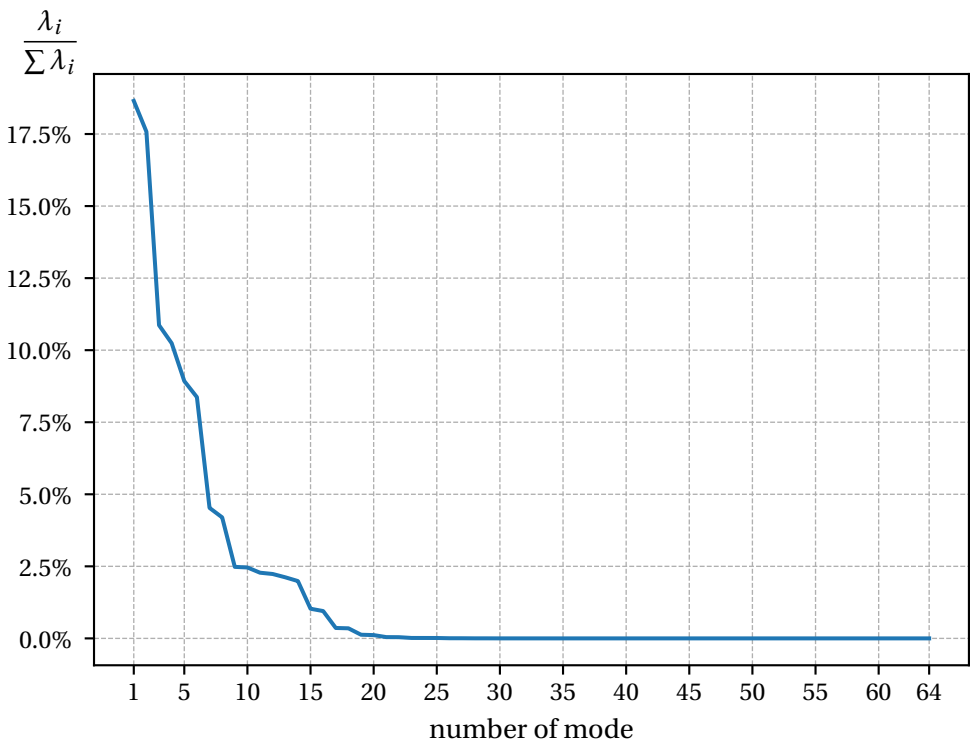

Figure 7.25: Relative energy content of modes.

In the following, we analyze the error between the ROM and the reference solution using different numbers of modes. Fig. 7.26 shows the error between the ROM with 40 modes skipped and the reference solution, Fig. 7.27 shows the error between the ROM with 30 modes skipped and the reference solution and Fig. 7.28 shows the error between the ROM with 20 modes skipped and the reference solution. With increasing number of used modes, the accuracy of the ROM and therefore the time where the error is small becomes larger.

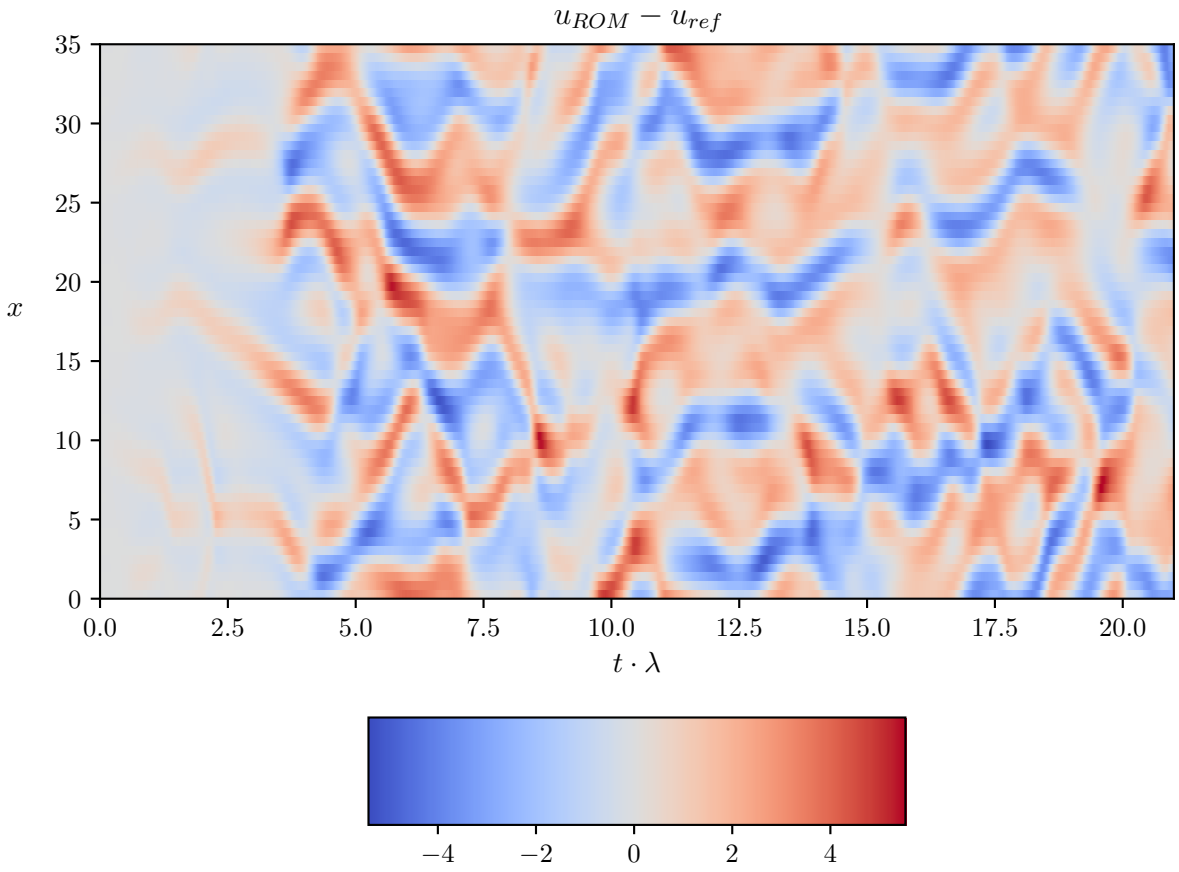

Figure 7.26: Error between ROM with 40 modes skipped and reference solution.

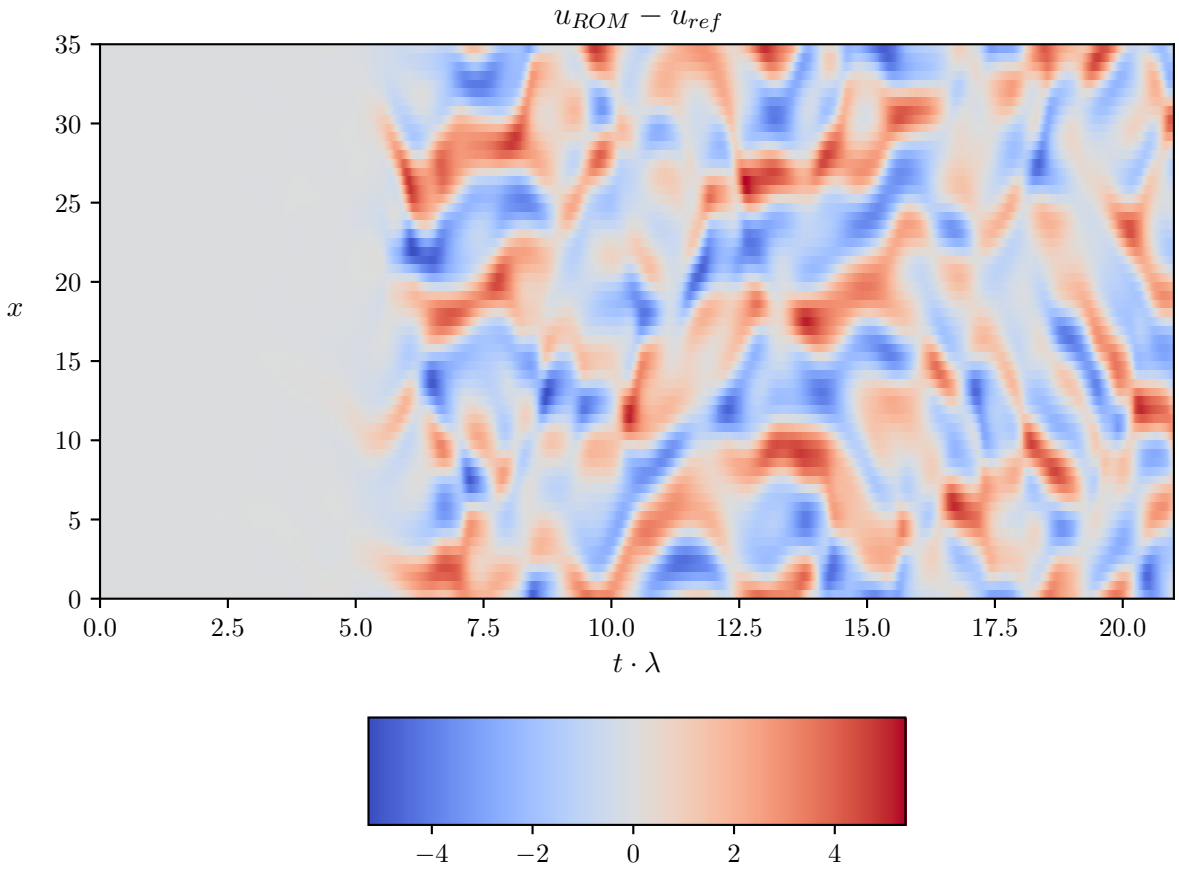

Figure 7.27: Error between ROM with 30 modes skipped and reference solution.

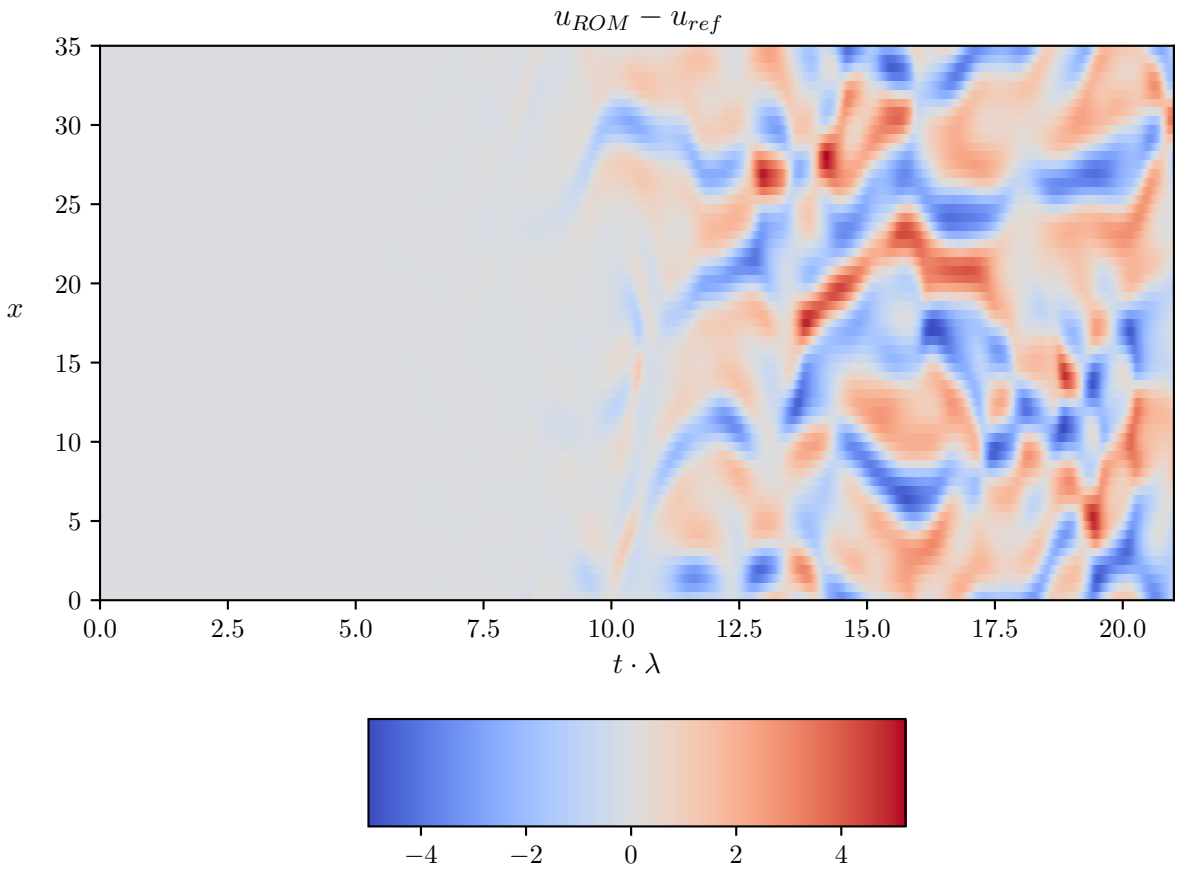

Figure 7.28: Error between ROM with 20 modes skipped and reference solution.

Hybrid1

Table 7.18 summarizes the hyper-parameters found by the parameter search for 35 skipped modes, Table 7.19 summarizes the hyper-parameters found by the parameter search for 40 skipped modes and Table 7.20 summarizes the hyper-parameters found by the parameter search for 45 skipped modes, for the first hybrid method applied to KS equation.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	2.5	0.8499	0.5	0.35
β	training samples	normalization	resync steps	washout
0.0001	5714	Max	57	100

Table 7.18: Parameters for hybrid1 method of the KS equation with 35 modes skipped.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	2.7279	0.9	0.94	0.6
β	training samples	normalization	resync steps	washout
0.00012	5714	Max	57	100

Table 7.19: Parameters for hybrid1 method of the KS equation with 40 modes skipped.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	3.0	1.05	0.378	0.4
β	training samples	normalization	resync steps	washout
0.00005	5714	Max	57	100

Table 7.20: Parameters for hybrid1 method of the KS equation with 45 modes skipped.

Fig. 7.29 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size for hybrid method 1 combined with different accurate ROMs. The method with 35 skipped modes showed the maximum valid time at the design point $n_{neurons} = 500$ and the values slightly decreased afterwards. The results were not consistent for 40 skipped modes, which showed a minimum at 1000 neurons and got bigger in both directions. For 45 skipped modes, the results were approximately constant for all reservoir sizes. Hybrid method 1 strongly depends on accurate ROMs and a high dimensional space for reconstruction. This is shown by the particularly large increase in prediction time between the ROM with 40 skipped modes and the ROM with 35 skipped modes. The accuracy slightly decreases with increasing number of neurons for the ROM with 35 skipped modes which could be due to the growing distance from the design point.

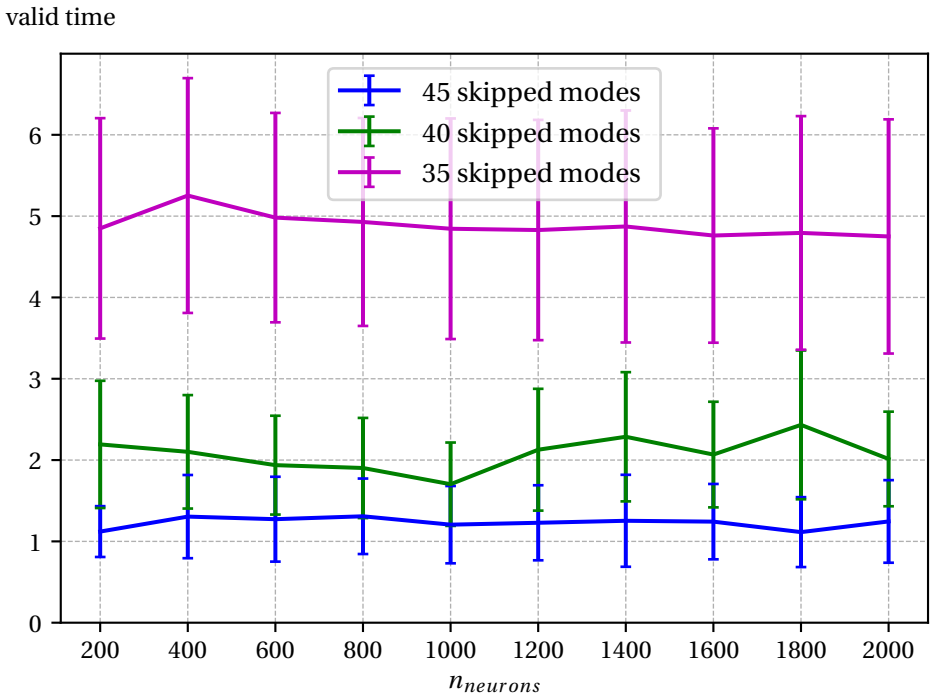

Figure 7.29: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ for three different numbers of skipped modes using hybrid 1.

Using the parameters from Table 7.20, we vary the number of skipped modes. The prediction performance depending on the number of skipped modes is depicted in Fig. 7.30. A decrease in accuracy is observed when skipping more modes which is expected. Surprisingly, for $n_{skippedmodes} = 38$ and $n_{skippedmodes} = 39$ no results were found. In rare cases, the coupling between ESN and ROM lead to numerical instabilities. At these two points (the only ones in this work) unstable behavior was observed.

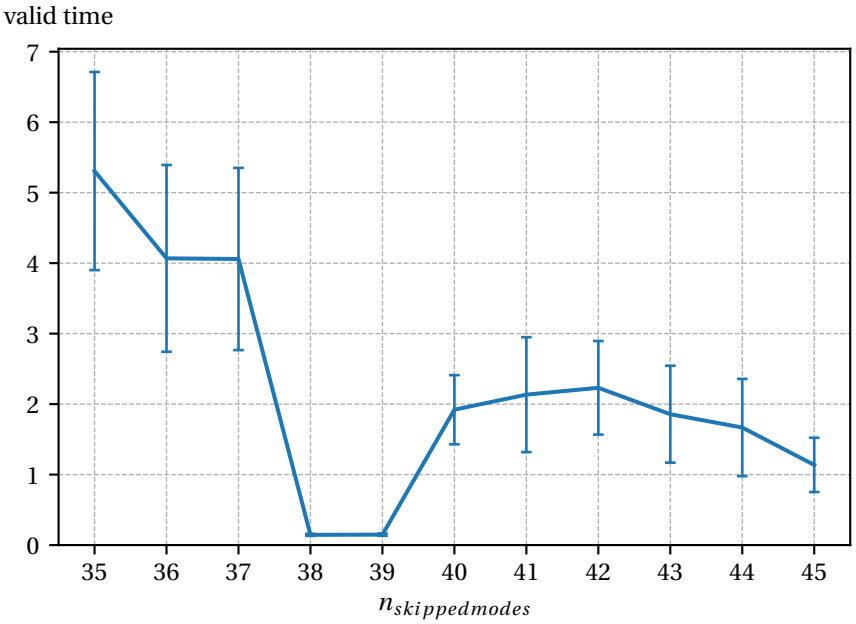

Figure 7.30: Valid prediction time with standard deviation for increasing number of skipped modes using hybrid 1 with a reservoir of 500 neurons.

Hybrid2

Table 7.21 summarizes the hyper-parameters found by the parameter search for 35 skipped modes, Table 7.22 summarizes the hyper-parameters found by the parameter search for 40 skipped modes and Table 7.23 summarizes the hyper-parameters found by the parameter search for 45 skipped modes, for the second hybrid method applied to KS equation.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	3.5	0.9	0.8	0.6
β	training samples	normalization	resync steps	washout
0.00005	5714	Stddev	57	100

Table 7.21: Parameters for hybrid2 method of the KS equation with 35 modes skipped.

$n_{neurons}$	sparsity	α	σ	$\rho(\underline{\mathbf{W}})$
500	3.5	0.9	0.8	0.65
β	training samples	normalization	resync steps	washout
0.000072	5714	Stddev	57	100

Table 7.22: Parameters for hybrid2 method of the KS equation with 40 modes skipped.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	3.0	0.9	0.2	0.2
β	training samples	normalization	resync steps	washout
0.00018	5714	Max	57	100

Table 7.23: Parameters for hybrid2 method of the KS equation with 45 modes skipped.

Fig. 7.31 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size for ROMs with different accuracy. We found the maximum valid time at the smallest reservoir and the minimum at 800 neurons for 35 skipped modes. The values constantly decreased from 200 to 800 neurons and started to grow slightly after the minimum until a reservoir size of 1200 neurons was reached, when the valid prediction time decreased again. For 40 skipped modes, the valid prediction time constantly decreased with increasing reservoir size except for a reservoir with 1200 neurons where the result was slightly better than the adjacent results. The prediction time for hybrid method 2 with 45 skipped modes constantly decreased with increasing reservoir size but had a minimum at 400 neurons. The results showed that hybrid method 2 works best with low dimensional systems as the Platt system or the CDV system. With increasing numbers of skipped modes, the ESN is increasingly unable to predict the truncated dynamics.

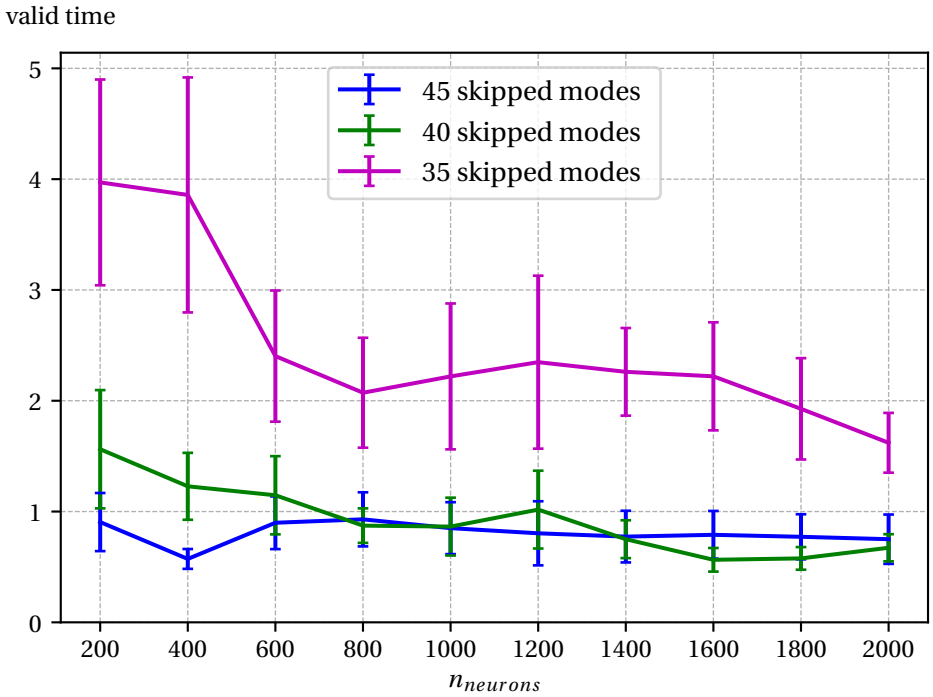

Figure 7.31: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ for three different numbers of skipped modes using hybrid 2.

Using the parameters from Table 7.23, we vary the number of skipped modes. The prediction performance depending on the number of skipped modes is depicted in Fig. 7.32. With increasing number of skipped modes, the prediction performance decreased until 39 modes. From then on, the valid time was better until it started to decrease again. The inconsistent drop in the curve could be due to the sensitivity of the method to the hyper parameters.

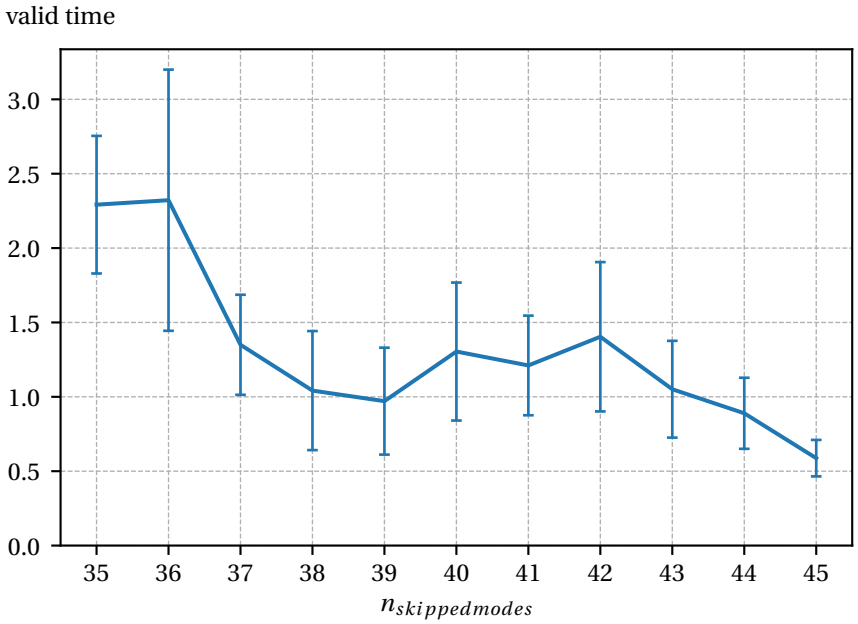

Figure 7.32: Valid prediction time with standard deviation for increasing number of skipped modes using hybrid 2 with a reservoir of 500 neurons.

Hybrid3

Table 7.24 summarizes the hyper-parameters found by the parameter search for 35 skipped modes, Table 7.25 summarizes the hyper-parameters found by the parameter search for 40 skipped modes and Table 7.26 summarizes the hyper-parameters found by the parameter search for 45 skipped modes, for the third hybrid method applied to KS equation.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	3.0	0.95	0.045	0.3
β	training samples	normalization	resync steps	washout
0.000072	5714	Max	57	100

Table 7.24: Parameters for hybrid3 method of the KS equation with 35 modes skipped.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	2.8	0.95	0.55	0.3
β	training samples	normalization	resync steps	washout
0.00005	5714	Max	57	100

Table 7.25: Parameters for hybrid3 method of the KS equation with 40 modes skipped.

$n_{neurons}$	sparsity	α	σ	$\rho\left(\underline{\mathbf{W}}\right)$
500	2.5	0.95	0.4	0.3
β	training samples	normalization	resync steps	washout
0.00001	5714	Max	57	100

Table 7.26: Parameters for hybrid3 method of the KS equation with 45 modes skipped.

Fig. 7.33 shows the average prediction time in Lyapunov times units ($= \lambda \cdot t_{valid}$) with standard deviation for increasing reservoir size for ROMs using different numbers of modes. The valid prediction time was nearly constant for all different reservoir sizes for a ROM with 35 skipped modes. As mentioned at the Platt system, hybrid method 3 prefers information from an accurate ROM if available. Therefore, the prediction performance is good and no increase in prediction time with increasing number of neurons could be observed since the ESN's information is inferior for the solution.
While the method with a ROM with 45 skipped modes is the least accurate at the beginning without the support of the ESN, it rapidly gains accuracy with increasingly large reservoirs. Towards the larger reservoirs, the ESN compensates for the inaccurate ROM and the prediction performance of the methods, with 45 skipped modes and 40 skipped modes, converged.

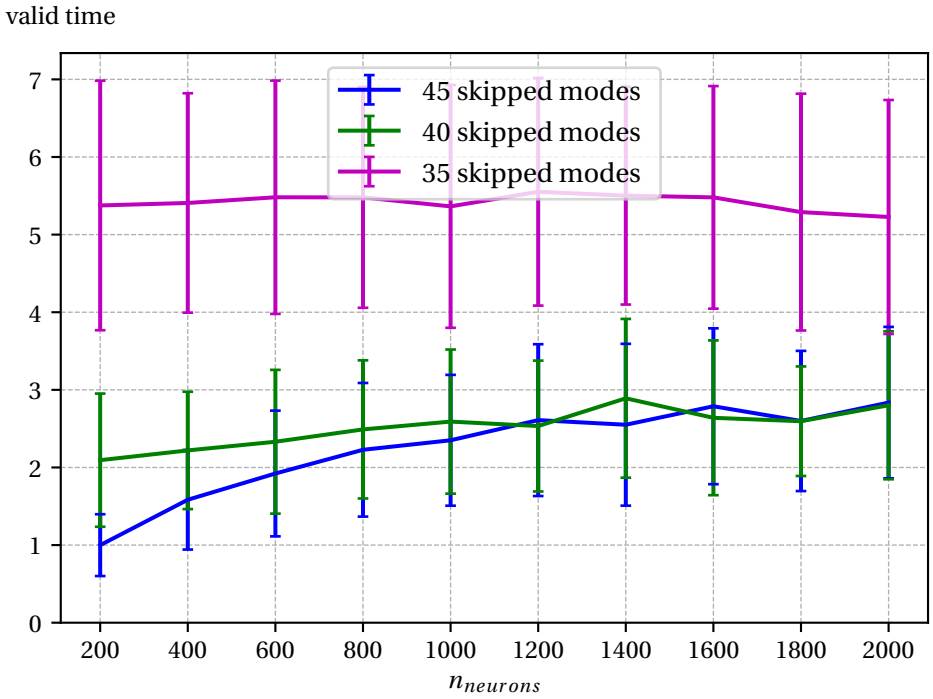

Figure 7.33: Valid prediction time with standard deviation for increasing reservoir size $n_{neurons}$ for three different numbers of skipped modes using hybrid 3.

Using the parameters from Table 7.26, we vary the number of skipped modes. The prediction performance depending on the number of skipped modes is depicted in Fig. 7.34. With increasing number of skipped modes, the prediction performance constantly decreased.

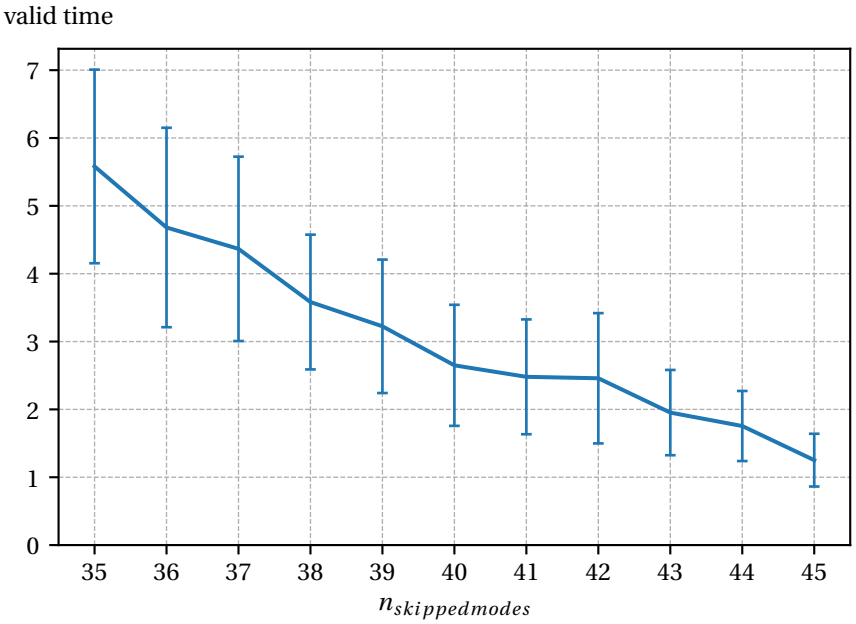

Figure 7.34: Valid prediction time with standard deviation for increasing number of skipped modes using hybrid 3 with a reservoir of 500 neurons.

In addition, we want to illustrate the weighting between ROM and ESN information using the trainable output matrix $\underline{\mathbf{W}}_{out}$. The ESN is trained using the parameters listed in Table 7.26. Since the KS system has 64 dimensions, the first 64 columns of $\underline{\mathbf{W}}_{out}$ correspond the ROM and the remaining 500 columns correspond to the units of the reservoir of the ESN. First, the output matrix is normalized by dividing each row with the sum of the row's squared entries. Second, the mean of each column is calculated and named $\langle\alpha(x)\rangle$. Fig. 7.35 shows the average weighting of each column in $\underline{\mathbf{W}}_{out}$. The entries in the output matrix mirror the observed behavior in the the other results. With decreasing accuracy, the method reduces the influence of the ROM information and the own ESN prediction becomes more important. To further illustrate the weighting between ESN and ROM information, the mean of $\langle\alpha(x)\rangle$ from columns 1-64 is calculated and named $\langle\alpha\rangle$ and the mean of the remaining columns (corresponding to the ESN) is calculated and named $\langle\beta\rangle$. Fig. 7.36 shows that with increasing number of skipped modes, the influence of the ROM $\langle\alpha\rangle$ over the ESN prediction $\langle\beta\rangle$ becomes less important.

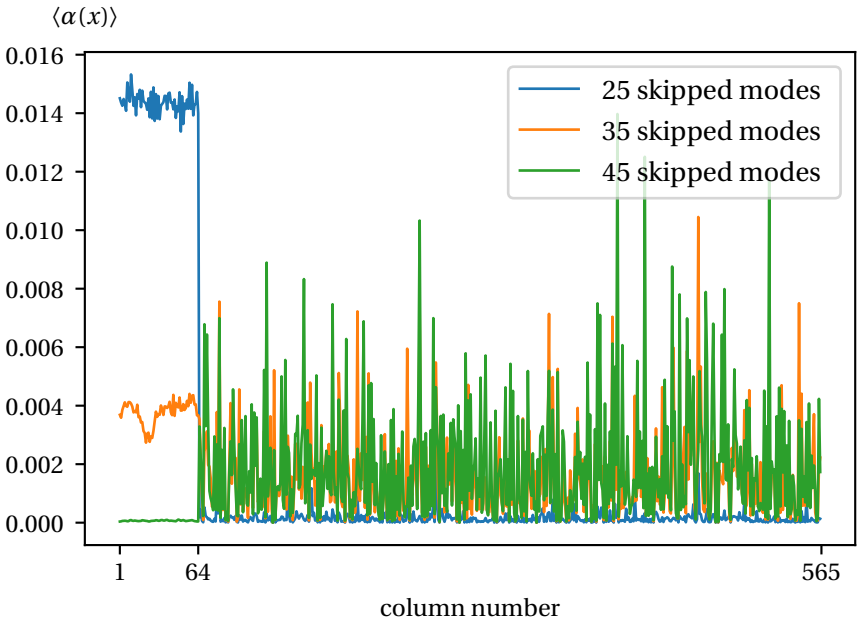

Figure 7.35: Average Weighting of each column in output matrix $\underline{\mathbf{W}}_{out}$ for different numbers of skipped modes.

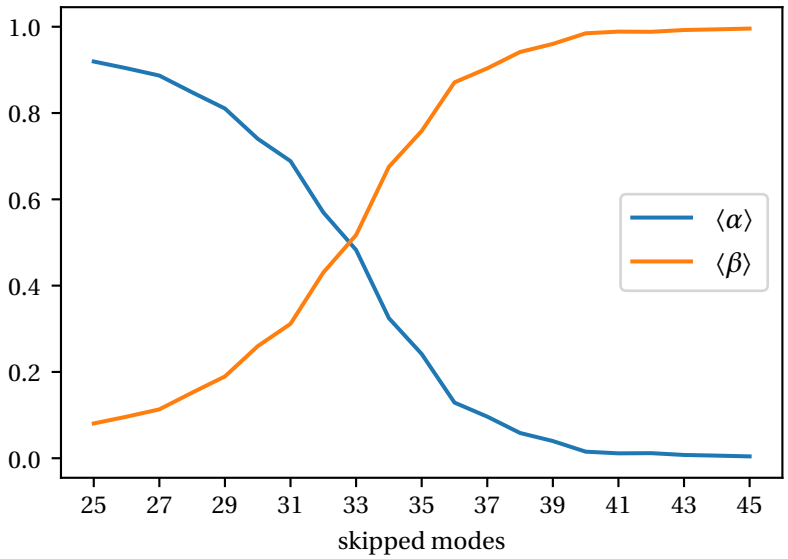

Figure 7.36: Weighting of ESN and ROM information in output matrix $\underline{\mathbf{W}}_{out}$ with increasing number of skipped modes.

7.3 Summary

We summarize all the results here to make the direct comparison easier. In the Figs. 7.37 - 7.42, the pure data-driven model and the three hybrid methods are plotted together to provide a direct comparison between the different methods.
Fig. 7.37 shows the results for the Lorenz system. While the second hybrid method shows strong variations and no consistent scaling behavior, the other hybrid methods outperform the pure data driven method. Especially hybrid method 3 is at each point better than the pure data-driven method whereas the first hybrid method is at two points slightly worse than the pure data-driven method. Furthermore, only hybrid method 3 provides a relatively consistent increase in prediction time with reservoir size.

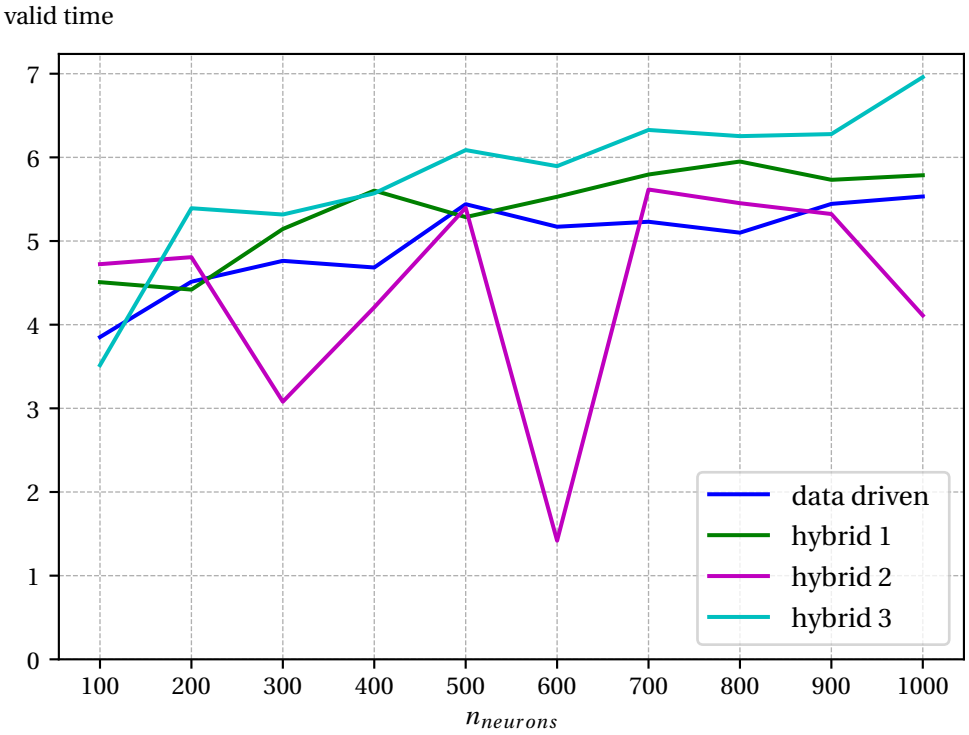

Figure 7.37: Comparison of all methods for Lorenz system.

Fig. 7.38 shows the results for the Platt system. The pure data-driven method shows improving performance with increasing reservoir size. However, both hybrid methods 2 and 3 provide, over the entire range of reservoirs better and nearly constant valid prediction times than the pure data-driven method. Hybrid method 1 is not able to capture the dynamics of the Platt system since the reconstruction in the reduced space is too restrictive for the complex dynamics of this system.

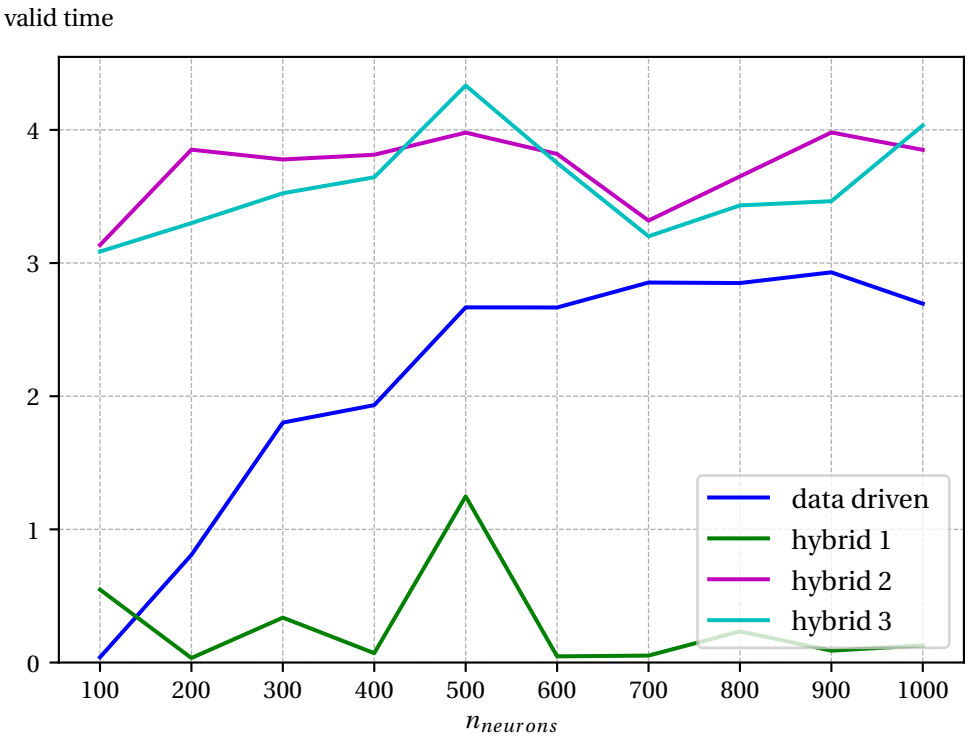

Figure 7.38: Comparison of all methods for Platt system.

Fig. 7.39 shows the results for the CDV system. In this system which has a larger dimension than the Lorenz system, hybrid method 1 has reduced performance and is no better than the pure data-driven method anymore. Hybrid method 3 shows throughout all reservoir sizes good, nearly constant, performance while hybrid method 2 starts with smaller values but outperforms hybrid method 3 towards the largest reservoirs. Both, hybrid method 2 and hybrid method 3 are better than pure data-driven over the entire range of reservoirs.

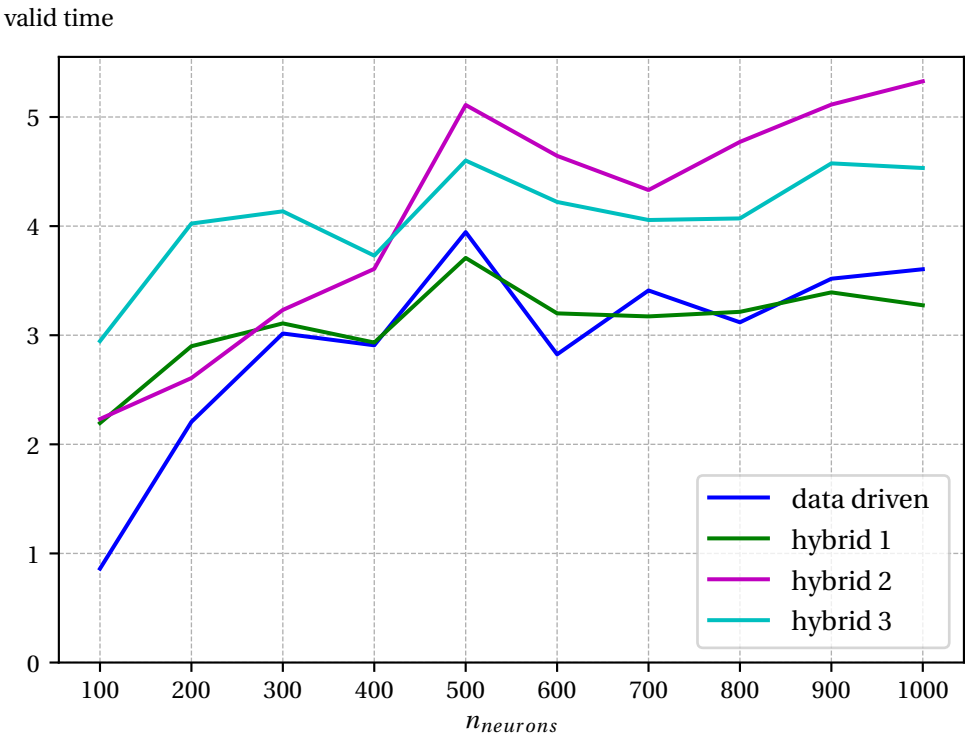

Figure 7.39: Comparison of all methods for CDV system.

Fig. 7.40 shows the results for the KS system and hybrid methods using a ROM with 35 skipped modes. Hybrid method 2 shows the worst performance. Although hybrid method 2 is superior to pure data-driven in the beginning, with increasing number of neurons, hybrid method 2 looses accuracy and the pure data-driven, improves its performance. Both, hybrid method 1 and hybrid method 3 outperform the pure data-driven method drastically where hybrid method 3 is even more accurate than hybrid method 1.

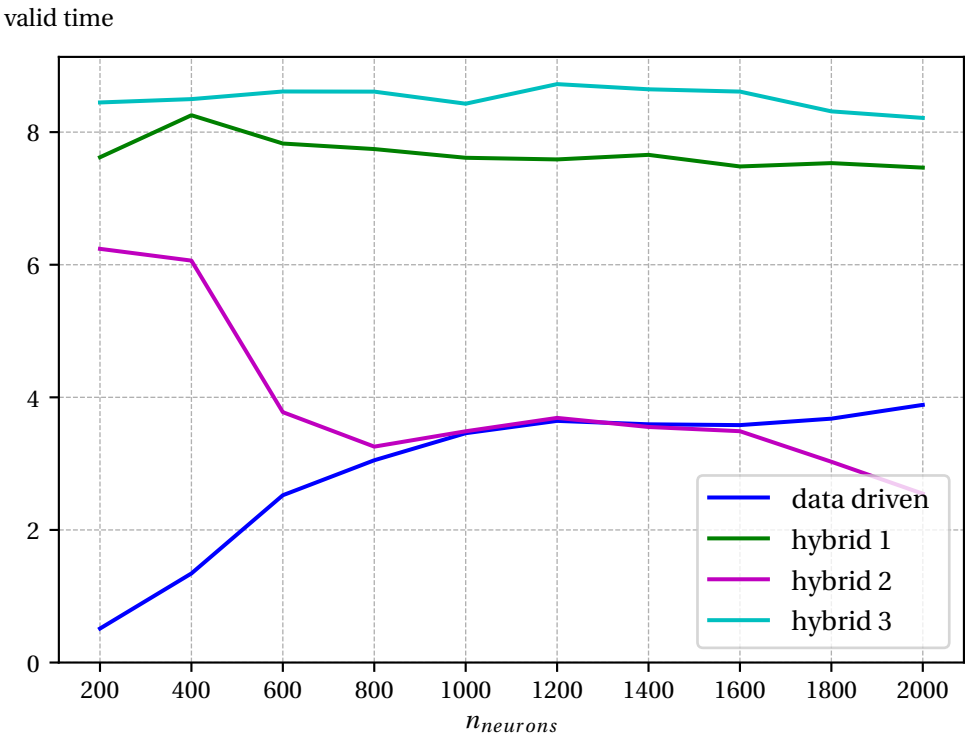

Figure 7.40: Comparison of all methods for KS equation with hybrid methods that skip 35 modes.

Fig. 7.41 shows the results for the KS system and hybrid methods using a ROM with 40 skipped modes. A similar trend as in Fig. 7.40 can be observed. With small reservoirs all hybrid methods are better than the pure data-driven method. Hybrid method 1 is approximately as accurate as the pure data-driven method for reservoirs larger than 800 neurons. Hybrid method 3 is the most precise method for all reservoir sizes with a slightly increasing performance with growing reservoirs.

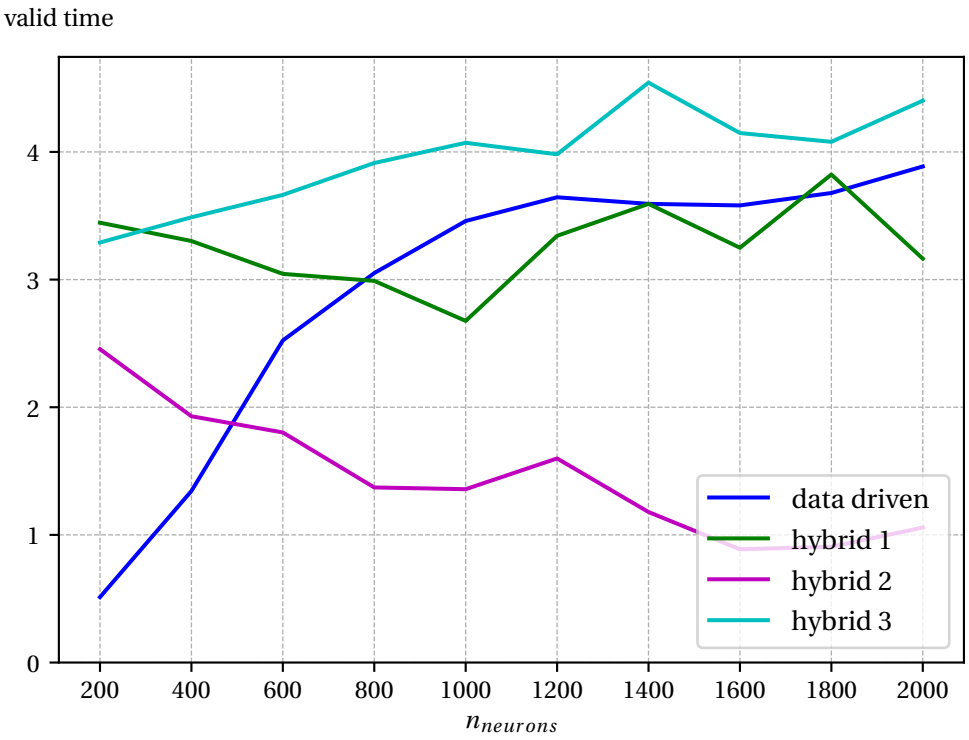

Figure 7.41: Comparison of all methods for KS equation with hybrid methods that skip 40 modes.

Fig. 7.42 shows the results for the KS system and hybrid methods using a ROM with 45 skipped modes. Hybrid method 1 and hybrid method 2 are not able to perform better than the pure data-driven method anymore. These methods are better than the pure data-driven method only for the very small reservoirs but their performance stays constant with increasing reservoir size. Hybrid method 3 is better than the pure data-driven method for all reservoir sizes.

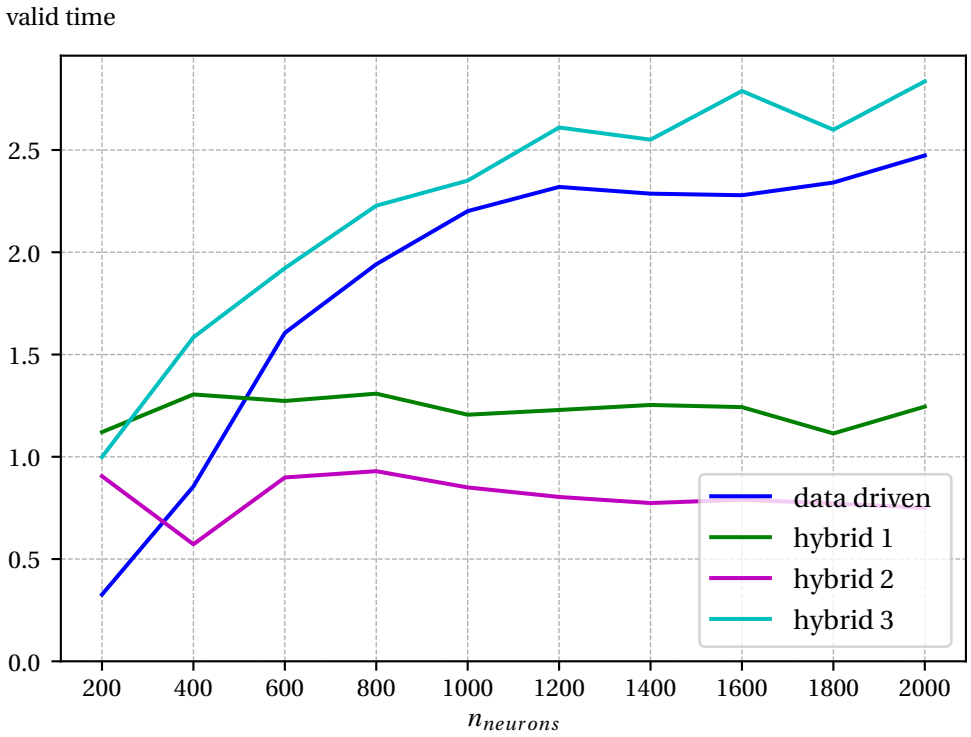

Figure 7.42: Comparison of all methods for KS equation with hybrid methods that skip 45 modes.

8 Discussion

From the results presented in Section 7, we can observe that the pure data-driven method shows good prediction performance. The ESN is generally able to learn the dynamics of the different systems. The prediction performance increases slightly with increasing reservoir size for low dimensional systems without too complicated dynamics like intermittency. However, the ESN improved its prediction performance more when learning the dynamics of high dimensional and intermittent systems. The Platt system is an example of the latter. Already small changes in the way the computation is done may lead to different behavior of the system in the sensitive regions as shown with the ROM for Platt system that uses all modes. Therefore, with a more accurate prediction, which is possible through larger reservoirs, the valid prediction time increases consistently. Additionally, high dimensional systems naturally require more neurons to recreate a higher dimensional state, such as the discretized KS equation where we see the strong increase of prediction performance with increasing reservoir size as well.

Hybrid method 1 reconstructs the solution only in a reduced space. This reconstruction of the solution in reduced space is difficult, especially for low-dimensional systems where reducing one dimension has drastic impact on the resulting dynamics. For the Lorenz and the CDV systems, reducing only one dimension leads to completely inaccurate ROMs which converged towards fixed points. The Platt system's ROM with one mode skipped was able to capture some dynamical features, however intermittency was not apparent anymore. The KS equation with 64 grid points and, thus, 64 dimensions, was the only ROM that was able to recover the systems dynamics. We showed in the ROM section that keeping more and more modes in the system increased the accuracy of the ROM. However, the contribution to the total energy was not a good measure as skipping only 5% of the total energy caused unstable behavior.
Hybrid method 1 did not work well for the CDV and Lorenz systems. The prediction performance is negligibly better than for the pure data-driven method. Even an increase in reservoir size did not improve the performance much. This could be due to the leading error caused by the reconstruction in reduced space and the governing dynamics of the ROM, which are completely different from the real systems dynamics. This is supported by the results for the Platt system, where reconstruction in reduced space seems to be too restrictive to capture any dynamics.
With the KS equation, the hybrid method 1 worked well with an accurate ROM that skipped fewer modes whereby many dimensions are available for reconstruction. In addition, prediction performance stayed nearly constant for all reservoir sizes. With increasing number of skipped modes, which is equivalent to a more and more inaccurate ROM, prediction perfor-

mance got worse. In the limiting case, $n_{skippedmodes} = 45$, near the stability threshold, the prediction performance was worse than the pure data-driven one. These observations support that hybrid method 1 is dominated by the ROM's behavior and that the error prediction of the ROM, which is in most cases also low-dimensional, therefore requiring no big reservoir for accurate results, plays a smaller role.

Hybrid method 2 tried to compensate for the dimensionality problem explained above. In addition to the prediction of the error term, the remaining coefficients are predicted and the reconstruction takes place in full space. The method was successful only for the Platt and CDV system and especially with the Platt system where hybrid method 1 failed. However, hybrid method 2 showed no consistent behavior which could be due to the following reasons:

1. Hybrid method 2 is sensitive to hyper parameters. As no statistical parameter search was performed, different random initializations of the reservoir could lead to strong variations in prediction performance and the parameter search could not find good values.

2. The way the correction term for the ESN is computed, it could contain already portions of the skipped modes that are now taken into account twice.

3. Different dynamics between the correction term and the skipped time dependent coefficients. ESN could have problems to predict variables with different typical time scales. In addition, the decrease in prediction performance with increasing reservoir sizes especially at high dimensional systems with accurate ROMs could be due to an increasing sensitivity of the ESN to variables with different dynamics.

Neither with Lorenz system nor with one ROM of KS equation was hybrid method 2 able to predict longer than pure data-driven.

Hybrid method 3 is the only method that constantly outperforms the pure data-driven method for all systems. Due to the special architecture, the ESN is able to "decide" whether to incorporate more ESN or ROM information. The method's ability to decide can be seen in the behavior of the method with increasing reservoir size. With ROMs that contain more information, the prediction performance stays nearly constant for all reservoir sizes, similar to hybrid method 1 where the ROM clearly dominates. The Platt system and KS equation with just 35 skipped modes are good examples for this. In all other cases, an increase of prediction performance can be seen with increasing reservoir size, similar to the pure data-driven method. Especially the steepening of the curve with increasing number of skipped modes and the more and more parallel course of hybrid method 3 to the pure data-driven method is striking.
Hybrid method 3 works best with all systems, as it seems that this method is able to combine information from a ROM and information from a ML model adequately.

9 Conclusion

9.1 Summary

This work dealt with the prediction of chaotic dynamical systems using machine learning approaches combined with reduced order models (ROM). We generated data sets for four different systems: the Lorenz system, the Platt system, the Charney-DeVore system and the Kuramoto Sivashinsky(KS) equation. We chose these systems due to their increasing complexity. The Platt system in particular serves us as a test case due to the intermittent dynamics. The KS equation is a PDE and the discretization transforms this equation into a high-dimensional system as the number of grid points is usually large.
We applied the proper orthogonal decomposition (POD) on the generated reference data and were able to extract modes and coefficients that are optimal in an energy sense. We created a new basis for the solution using these modes and projected the governing equations onto the new basis set. This gave us a mapping for the time evolution of the purely time dependent coefficients. Discarding the modes which contribute the least information to the solution yielded the ROM.
To compensate for the information loss in the ROM, we introduced the echo state network (ESN). This machine learning method is known for its ability to approximate dynamical systems and especially for the comparatively small training effort. ESN are not trained by back propagation as usual neural networks, their training simplifies to a linear regression problem. We combined these two key components to create three hybrid methods which differ in structure. The first hybrid method uses the ESN to predict the skipped dynamics with the reconstruction of the solution in the POD reduced space. Hybrid method 2 is the extension of hybrid method 1 and reconstructs the solution in full space. Hybrid method 3 is different compared with the other two methods and uses the ESN to predict the next timestep and additionally to decide how much information to take from the ROM.
We applied all methods to all systems, calculated optimal hyper parameters and performed reservoir size studies. We came to the following results.
As already shown in literature, the ESN is able to learn the dynamics of chaotic systems. For low-dimensional systems, ESN with small reservoirs are able to capture the dynamics and no large increase in prediction time could be observed for larger reservoirs. High-dimensional systems require larger ESNs, therefore we saw a rapid rise in prediction time. Hybrid method 1 seemed to be too restrictive for low-dimensional systems since reconstruction takes place solely in reduced space. In addition, with low-dimensional systems, the ROM with one mode skipped showed completely different dynamics. Hybrid method 1 provided good prediction time only with high-dimensional systems where a ROM could be generated that captured the

original dynamics and the order reduction provided enough dimensions for solution reconstruction.
Hybrid method 2 tried to predict the skipped dynamics. However, this method showed better results only for Platt and CDV systems. No consistent behavior with increasing reservoir size of the ESN could be observed. Hybrid method 3 was consistently better than pure data-driven method for all systems. This method proved its ability to decide whether to take information from the ESN or from the ROM. If an accurate ROM was present, nearly no increase in prediction performance was observable as the ESN decided to take more ROM information and the ESN was of secondary importance. In contrast, the prediction performance was getting better with increasing reservoir size if an inaccurate ROM was present and the ESN had to rely more on its own prediction capability.
This work showed the possibility to successfully combine pure machine learning models with incomplete physical information originating from reduced order models. The promising results motivate the in-depth discussion and continuing research on this topic. Further topics are presented in the next section.

9.2 Future Work

In this thesis we investigated different hybrid methods in conjunction with pure projection based ROMs. Different hybrid architectures were applied to different systems. However, there is still space for more research on various related topics.
Projection based reduced order models are known for suffering from numerical instabilities. We faced no instabilities, however, reducing the system's dimension instantly leads to completely different dynamics that hindered some hybrid architectures, which rely on accurate ROMs, from working well. Two hybrid architectures tried to compensate for the information loss with prediction of the error, that worked partially well. There exist other ROM methods, that incorporate some stabilizing dynamics hidden in the skipped modes inherently. To improve the ROM's accuracy is one of the further topics.

The parameter search algorithm probably leaves the most work open. First, a statistical parameter search that removes the dependence on the random initialization of the reservoir could be conducted. The computational resources during this work were not capable of doing that. Second, we performed a simple line search. A grid search or better methods based on backpropagation algorithms could help to find better hyper parameters. Last, instead of using not just the valid prediction time as an error measure, one could introduce accumulative errors that account for the error's path. Only small peaks in the error currently affect the search. These ideas could help to get closer to the error's global minimum instead of getting stuck in some local minima.

It is currently unclear why hybrid method 2 did not show a consistent behavior and lost accuracy with increasing reservoir size especially at high dimensional systems with accurate ROMs. An approach could be to investigate the dynamics of the different modes and their typ-

ical time scales. Since POD decomposes with respect to the kinetic energy, time scales are not directly taken into account and the different modes could have strongly varying time scales. In addition, the ESN has to predict the truncated modes together with the retained POD modes. This could be hard for a single ESN to predict. Models exist, where variables with similar typical time scales are collected and an own ESN is used for each group to predict their time evolution. The implementation of a model that uses small ESNs to predict the time evolution of groups of variables with similar time scales is one further topic.

Finally, there is always a possibility to reduce computational effort. The number of calculations can be reduced as well as a time step as big as possible can be chosen to reduce computation time and effort.

Bibliography

[1] Ashesh Chattopadhyay, Pedram Hassanzadeh, Krishna Palem, and Devika Subramanian. Data-driven prediction of a multi-scale lorenz 96 chaotic system using a hierarchy of deep learning methods: Reservoir computing, ann, and rnn-lstm. *arXiv preprint arXiv:1906.08829*, 2019.

[2] Steven M Cox and Paul C Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

[3] Daan T Crommelin, JD Opsteegh, and F Verhulst. A mechanism for atmospheric regime behavior. *Journal of the atmospheric sciences*, 61(12):1406–1419, 2004.

[4] DT Crommelin and AJ Majda. Strategies for model reduction: Comparing different optimal bases. *Journal of the Atmospheric Sciences*, 61(17):2206–2217, 2004.

[5] Russell A Edson, Judith E Bunder, Trent W Mattner, and Anthony J Roberts. Lyapunov exponents of the kuramoto–sivashinsky pde. *The ANZIAM Journal*, 61(3):270–285, 2019.

[6] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, volume 14. Springer, 2010.

[7] Michael D Hartl. Lyapunov exponents in constrained and unconstrained ordinary differential equations. *arXiv preprint physics/0303077*, 2003.

[8] Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[11] Herbert Jaeger. Echo state network. *scholarpedia*, 2(9):2330, 2007.

[12] Aly-Khan Kassam and Lloyd N Trefethen. Fourth-order time-stepping for stiff pdes. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

[13] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20(2):130–141, 1963.

[14] Mantas Lukoševičius. A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade*, pages 659–686. Springer, 2012.

[15] John Leask Lumley. The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, 1967.

[16] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[17] Hadrien Montanelli and Yuji Nakatsukasa. Fourth-order time-stepping for stiff pdes on the sphere. *SIAM Journal on Scientific Computing*, 40(1):A421–A451, 2018.

[18] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120(2):024102, 2018.

[19] Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R Hunt, Michelle Girvan, and Edward Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):041101, 2018.

[20] Suraj Pawar, Shady E Ahmed, Omer San, and Adil Rasheed. Data-driven recovery of hidden physics in reduced order modeling of fluid flows. *Physics of Fluids*, 32(3):036602, 2020.

[21] NSEA Platt, EA Spiegel, and C Tresser. On-off intermittency: A mechanism for bursting. *Physical Review Letters*, 70(3):279, 1993.

[22] Bérengère Podvin and Yann Fraigneau. A few thoughts on proper orthogonal decomposition in turbulence. *Physics of Fluids*, 29(2):020709, 2017.

[23] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[24] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[25] Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, and Themistoklis Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5), 2018.

[26] Julien Weiss. A tutorial on the proper orthogonal decomposition. page 3333, 2019.

[27] Hans Yu, Thomas Jaravel, Matthias Ihme, Matthew P Juniper, and Luca Magri. Data assimilation and optimal calibration in nonlinear models of flame dynamics. *Journal of Engineering for Gas Turbines and Power*, 141(12), 2019.