KU LEUVEN

FACULTY OF SCIENCE

# Hidden Information Behind the Time Series:

## Extracting Nullcline Structures using Artificial Neural Networks

**Jimmy BILLEN**

Supervisor: Prof. Dr. L. Gelens
KU Leuven

Mentor: Dr. N. Frolov
KU Leuven

Mentor: B. Prokop, MSc
KU Leuven

Thesis presented in
fulfillment of the requirements
for the degree of Master of Science
in Physics

Academic year 2023-2024

# Acknowledgments

First and foremost, I would like to express my gratitude towards my mentors Nikita and Bartosz, for their constant guidance, helpful suggestions, and for granting me the opportunity to get a taste of what it is like to be a researcher. I would also like to thank my supervisor Prof. Lendert Gelens, for introducing me to the fascinating topic of this thesis and his encouragement.

Also, I would like to thank all the members of the Laboratory of Dynamics in Biological Systems, and fellow students in the office for the great discussions and nice work environment.

Further, I am very grateful to my friends for their constant moral support. I am especially grateful towards my parents, they have supported me in every way they could, not only in the past year but for the entire five years of my studies in physics.
Thank you.

# Summary

Dynamical systems, from simple pendulums to complex biological oscillators, have been modeled mathematically for centuries, enabling detailed predictions and insights into their behavior. However, as the complexity of these systems increases, traditional first-principles approaches become insufficient, especially when dealing with large datasets or systems with intricate nonlinear dynamics. Recent advancements in computational power and machine learning pave the way for analyzing such systems. Especially machine learning algorithms such as neural networks are a powerful tool to uncover patterns in large structure datasets. Therefore they also could be used to uncover hidden structures within the time series data, such as nullclines and fixed points, which govern the dynamics of the system.

This thesis focuses on developing a robust machine learning algorithm to extract these fundamental structures from time series data of complex dynamical systems. We utilize the FitzHugh-Nagumo model, a well-known biological oscillator, to optimize the hyperparameters of neural networks for accurate nullcline recovery. Through systematic evaluation using metrics such as validation error, nullcline error, and Pearson correlation coefficient, we identify conditions under which the neural network reliably predicts nullclines. We investigate the algorithm's performance when challenges such as strong time-scale separation and low-data limits apply. We also test the algorithm's ability to generalize when applied to a more complex system of two bicubic nullclines.

# Summary for General Audience

Nature is full of dynamical systems, from the pendulum to the motion of celestial bodies. These systems have been modeled mathematically for centuries, allowing researchers to reveal more details, gain deeper insights, predict future behavior, and answer fundamental questions about the system. However, as the complexity of these systems increases, traditional first-principles approaches become insufficient, especially when dealing with nonlinear dynamics and large, intricate datasets. Examples can be found in climate science, where models are lacking.

Recent advancements in data storage, processing, and machine learning have offered new opportunities to analyze such complex systems more effectively. Machine learning, which has already significantly advanced fields such as computer vision, language understanding, and many others, has the ability to process vast amounts of data, uncovering hidden patterns and correlations that were previously inaccessible. These patterns, embedded within the time series data of complex systems, hold the key to understanding the underlying dynamics. In this study, we will focus particularly on biological oscillators. Examples of these oscillators are ubiquitous, from the periodic beating of your own heart to the cell cycle rhythms.

Despite their potential, the exact internal workings of machine learning algorithms remain somewhat vague, and these algorithms have been shown to exhibit unexpected behaviors in certain situations. For this reason, a careful and systematic approach is required to develop a trustworthy model that can reliably capture the dynamics of complex systems. To address these challenges, we propose an algorithm specifically designed to analyze and deduce structures within the time series data of complex dynamical systems. We develop the algorithm based on clear evaluation criteria. We will evaluate its performance for different dynamical behaviors. Additionally, we test its limitations and strengths for varying data resolution. Finally, we apply our method to more complex systems.

# Acronyms

**FHN** FitzHugh-Nagumo.
**NN** Neural Network.
**PCC** Pearson Correlation Coefficient.
**LR** Learning Rate.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

Nature is full of dynamical systems, from the pendulum to the motion of celestial bodies. Legendre and Gauss studied the latter in the beginning of the 19th century, developing mathematical models to describe the oscillatory motion of the planets around the sun [1]. Mathematical models enable us to derive the governing laws of a system from its equations, allowing us to reveal more detail, predict future behavior, and answer fundamental questions about the system [1, 2].

Modeling complex dynamical systems can be achieved by expressing the governing equations using differential equations [1]. Even though initially dynamics was a physics subject, today it is an interdisciplinary topic finding its applications in numerous fields like chemistry, biology, meteorology, astronomy and economy [3]. In recent years, analyzing nonlinear dynamical systems has become ever more difficult. The growing complexity of these systems makes it unfeasible to develop theories using first principles approaches. Additionally, identifying the mathematical model of a system can be increasingly difficult or impossible for large datasets. But at the same time, larger data sets are required for successful modeling of complex systems [4].

The recent advancements in data storage and processing, coupled with the exponential increase in computational power, made it possible to use data-driven methods in scientific research. The machine learning revolution, which has already significantly advanced the fields of vision, language understanding and many other fields, offers the ability to process large amounts of data for uncovering hidden patterns and correlations [5, 6]. Complex systems often contain a significant amount of hidden data within their time series or evolution, which, if extracted, could greatly enhance our understanding of these systems. Consequently, exact sciences have instead turned to data-driven methods and have evolved into big data sciences [4, 5].

However, machine learning algorithms, often regarded as black boxes due to their complexity and lack of interpretability, have been found to exhibit unexpected behaviors, challenging their trustworthiness despite their seemingly consistent performance [7].

A systematic approach is needed to acquire a machine learning algorithm that performs robustly, i.e. generalizes well to unseen data, is reproducible and reliable [8, 9, 10, 11]. Robust machine learning is particularly critical for capturing the dynamics of complex systems, as these systems often exhibit highly nonlinear and complex behaviors that can lead to unpredictable and inaccurate outcomes if the models are not reliable.
In this thesis we will create such a machine learning algorithm for the analysis of hidden structures in the time series of complex dynamical systems.

### Thesis Overview

Chapter 2 introduces the basics of dynamical systems. It serves to increase our understanding of the mathematical concepts, fundamental structures, and considered models. The basic operation of neural networks and the way they can serve as a tool for modeling are reported in Chapter 3. The final chapter examines the results and qualities of neural networks in model identification. Specifically, proposing a benchmark method and evaluating its robustness with low-resolution data and various other models.

### The Problem

In the study of dynamical systems, the data usually presents itself as a time series. For example, parameters such as ocean temperature, greenhouse gas concentrations in climate science, or the concentration of a biomolecule in biochemistry are measured at specific intervals and plotted over time, as illustrated on the left side of Figure 1.1. Although at first glance these time series appear to represent only the measured data, they inherently contain much more information, including the system's underlying dynamics for different initial conditions. Therefore, the challenge lies in extracting this hidden dynamical information from the time series. Data-driven methods, such as machine learning, offer ways to facilitate this extraction. A common step involves transforming the time series into the phase space representation, where the trajectory of the system can be analyzed more qualitatively. This reformulation highlights that the experimental data contains information on the potential landscape of the phase space. The most fundamental structures in the phase space are the nullclines and fixed points, which govern the system's behavior. However, despite being encoded in the time series, these structures are not immediately visible and must be uncovered through further analysis. We will demonstrate that these underlying structures can be recovered by applying machine learning algorithms. Specifically, in this thesis, we will employ neural networks. The schematic process is depicted in Figure 1.1, where we transition from the time series/phase space representation (with transparent nullclines) to a more refined view where these nullclines are explicitly revealed through the application of machine learning algorithms.

Extracting these fundamental structures is essential for the accurate identification and modeling of the dynamical systems underlying the time series data.

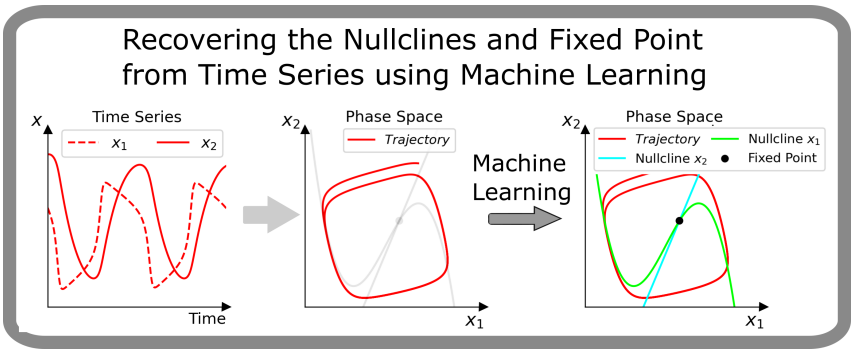


Figure 1.1: **Schematic: from time series data to nullcline and fixed point reconstruction**
The time series data is first transformed into the phase space representation. Subsequently, the hidden nullclines and fixed points can be extracted using machine learning.

# Chapter 2

# Basics to Dynamical Systems Theory

In this chapter, the mathematical formalism of dynamical systems is introduced. We start systematically, considering one-dimensional systems and their bifurcations. We illustrate methods to circumvent the challenges of unattainable analytic solutions. Subsequently, we generalize to two-dimensional systems, employing phase-plane analysis. This includes typical examples where we elaborate on the fundamental concepts of fixed points, null-clines, and limit cycles. The FitzHugh-Nagumo system contains all these structures and will be studied in detail in the final part of this chapter. The following section is based on the book of Strogatz, where the theory is explained in greater detail [12].

## 2.1 Theory of Dynamical Systems

A dynamical system is a system whose state is defined by a set of variables $x_1, x_2, \ldots, x_n$ and contains the evolution of that system in time [13]. The variables $x_1, x_2, \ldots, x_n$ can represent concentrations in a chemical reaction, populations in an ecosystem, or the position of celestial bodies. In this section, we will provide a quantitative way of studying dynamical systems by analyzing the evolution in a geometric way. The formal mathematical manner to describe the evolution of a system in time is using differential equations. In this thesis solely the temporal behavior is studied, thus we will focus on ordinary differential equations. This means that only ordinary derivatives are involved such as $\frac{dx}{dt}$, $\frac{d^2x}{dt^2}$, etc., with one independent variable, namely $t$. To gain a better understanding of how dynamical systems can be studied we will first consider a one-dimensional system, governed by the following differential equation

$$\dot{x} = F(x) \tag{2.1}$$

where $F$ is a function governing the dynamics of the variable $x$. The dots on the variables denote the differentiation with respect to $t$. It is defined as $\dot{x} = dx/dt$. If $F$ only contains terms that are linearly dependent on $x$ we name the system linear. Otherwise, the system is nonlinear, meaning that $F$ consists of products or functions of $x$, such as $x^2$ or $\cos(x)$, etc. As an example, let us first consider the following simple linear differential equation:

$$\dot{x} = rx \tag{2.2}$$

where $r > 0$ is called the growth rate. The change in $x$ increases with its value. Solving this system analytically we find for initial condition $x_0$

$$x = x_0 \exp(rt) \tag{2.3}$$

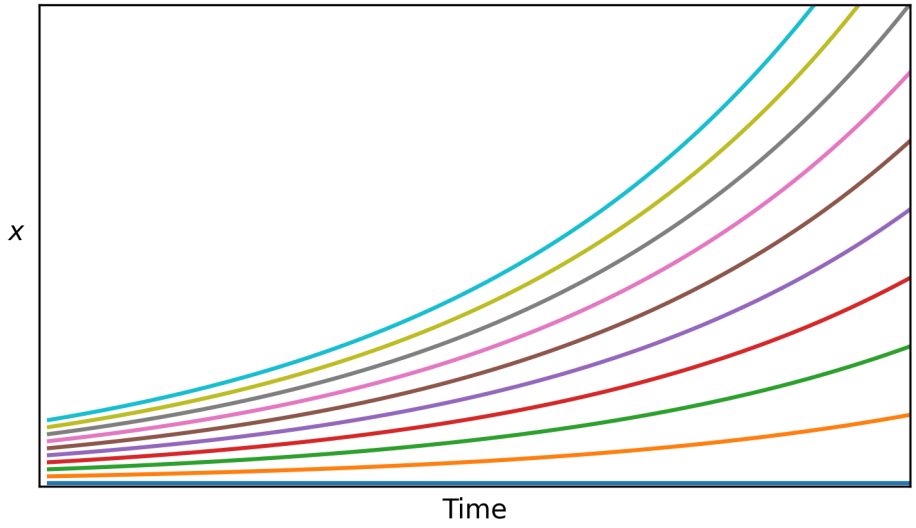


Figure 2.1: Exponential solutions for different initial conditions.

In Figure 2.1, the analytical solution is visualized for different initial conditions, showcasing exponential growth.

However, in practice, there is a great interest in the study of nonlinear systems. Nonlinearity is ubiquitous in nature, examples include lasers, turbulence in fluid, pendulums, celestial mechanics, etc. [3, 12]. Analytically solving these systems is rarely possible. Even when the time-dependent form $x(t)$ is known, its insights into the characteristics of the solution may be limited, particularly in understanding the system's rate of change and dynamics. Instead qualitative analysis can be performed by visualising the differential equations as a vector field. Consider the simple nonlinear differential equation

$$\dot{x} = x^2 + s \tag{2.4}$$

This equation can be visualized by plotting $\dot{x}$ as a function of $x$. The resulting curve is called the trajectory. For $x$ values where $\dot{x} > 0$, vectors point in increasing $x$ direction, and oppositely for $\dot{x} < 0$, as shown in Figure 2.2a for different values of $s$.

This figure, showing the qualitative behaviors of the trajectory, is called the phase portrait. This way of portraying the solutions is not quantitative but offers a geometric representation. It is not possible to read from the graph at which time the value of some $x$ is reached, instead the phase portrait provides the macroscopic structure of the system, revealing where solutions originate and their dynamical development. In this context, certain points where $\dot{x} = 0$ are of particular interest. These points, known as fixed points, indicate where the system remains in equilibrium.

Notice that for $s < 0$, there are two critical points where $x$ does not change, $\dot{x} = 0$, leading to $x^* = \pm\sqrt{-s}$. These are the fixed points of the system. The system exhibits a stable fixed point at $x^* = -\sqrt{-s}$ and an unstable fixed point at $x^* = \sqrt{-s}$, indicated

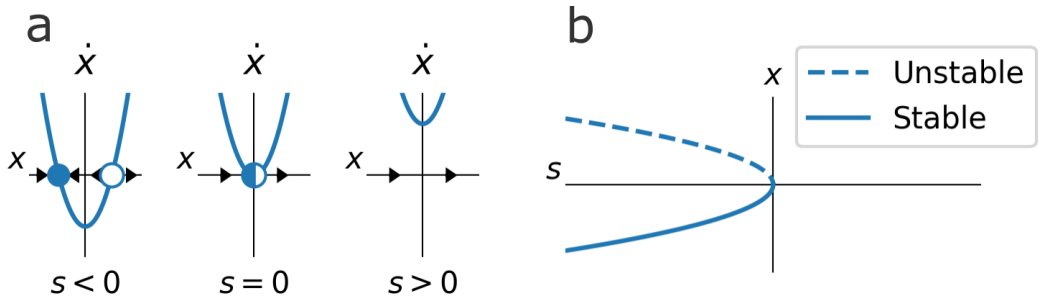


Figure 2.2: **Fixed point analysis**
**(a)** Phase portrait with trajectory and fixed points for three distinct values of $s$. Full dot corresponds to a stable fixed point, empty dot to an unstable fixed point and a half-filled dot to a half-stable fixed point. Figure inspired by Strogatz [12]. **(b)** Bifurcation Diagram. Dashed line corresponds to an unstable fixed point. A full line corresponds to a stable fixed point.

by arrows pointing towards the stable point and away from the unstable point. When $s = 0$, the two previous fixed points merge into a half-stable fixed point. For $s > 0$, no fixed points are present. The qualitative behavior of the system changes depending on whether $s < 0$ or $s > 0$, indicating the presence of a bifurcation at $s = 0$. A bifurcation diagram is presented in Figure 2.2b. The diagram depicts the $s$ dependence of the fixed points, distinguishing stable from unstable equilibria and visually demonstrating how the system transitions between different dynamical regimes as $s$ crosses zero.

The understanding of fixed points and their stability in one-dimensional systems is crucial for advancing to more complex, two-dimensional systems. Understanding the dynamics in one-dimensional systems helps to lay the groundwork for the analysis of two-dimensional systems. Here, complex phenomena like excitations and oscillations become possible and we turn to the study of trajectories $x_1, x_2$ in a two-dimensional phase space. The concept of nullclines is crucial to find the fixed points and it will prove its fundamental connection to the underlying structure of the system, as described in the following section.

Only a limited amount of dynamics is possible for the flow in one-dimensional phase spaces. Therefore the study of two-dimensional systems is of more interest, enabling excitations and oscillations, as will be demonstrated in the following section.

The following equations provide the general form of coupled two-dimensional ordinary differential equations

$$\begin{aligned} \dot{x}_1 &= F_1(x_1, x_2) \\ \dot{x}_2 &= F_2(x_1, x_2) \end{aligned} \tag{2.5}$$

where $F_i$ are functions governing the dynamics of the respective variables. Two-dimensional systems can be analyzed qualitatively using the two-dimensional phase space $(x_1, x_2)$, where the differential equations form a vector field on this plane: each point $(x_1, x_2)$ has an associated vector $(\dot{x}_1, \dot{x}_2)$. A trajectory corresponds to the curve $(x_1(t), x_2(t))$, starting at a specific initial condition. As before, we would like to study the fixed points $(x_1^*, x_2^*)$ of the system. A useful method for finding these points is the nullcline. The $x_i$-nullcline is

the set of points for which $\dot{x}_i = 0$, represented implicitly by $F_i(x_1, x_2) = 0$ in the $(x_1, x_2)$ phase space. This fundamental structure arises in systems with dimensions greater than two, with fixed points located at the intersections of the nullclines.
Nullclines provide useful information about the structure of the phase space. On the $x_1$-nullclines, the vectors have no $x_1$-component and point purely vertically, dividing the space into regions where vectors point only upwards or downwards. Similarly, on the $x_2$-nullclines the vectors point purely horizontally, separating the space into regions where vectors point only left or right. Thus, the nullclines fundamentally partition the phase space into regions with distinct behaviors [12, 14].

Conversely, it is possible to retrieve the differential equations from the nullclines. To give a simple example, if the nullclines exhibit a functional form such as, $x_2 = H(x_1)$ and $x_2 = K(x_1)$ one can trivially retrieve the corresponding differential equation as $\dot{x}_1 = H(x_1) - x_2$ and $\dot{x}_2 = K(x_1) - x_2$. More generally, this also works for multivariate polynomial terms like $xy$, $x^2y$, $x^2y^2$, etc. This demonstrates the profound relationship between nullclines and the structure of dynamical systems, proving their fundamental role in uncovering the underlying dynamics and explaining our interest in studying them.

Consider the following coupled nonlinear equations:

$$\begin{cases} \dot{x} &= \alpha x - \beta xy \\ \dot{y} &= \delta xy - \gamma y \end{cases} \tag{2.6}$$

This system is known as the Lotka-Volterra model [15], which describes the interaction between populations of prey $x$ and predator $y$ in an ecosystem. The fundamental characteristics of this system can be understood by examining the nullclines, resulting in $x = \gamma/\delta$ and $y = \alpha/\beta$. Notice how they divide the plane into regions where $\dot{x}$ and $\dot{y}$ have different signs, as illustrated in Figure 2.3. We can find the fixed point at the intersection of the nullclines, at the point $(\gamma/\delta, \alpha/\beta)$ both variables remain constant.

Figure 2.3 also illustrates the trajectories for different initial conditions. This system exhibits regular periodic oscillations of the populations, visualized as closed orbits in phase space. There are infinitely many closed orbits, each corresponding to a conserved quantity that remains constant along its trajectory. Such structures are known as conservative systems. If this system were temporarily perturbed (by actively increasing or decreasing the population) the trajectory would shift to another closed orbit and remain there.

However, in biology, one more frequently encounters examples of systems recovering from perturbations. Consider the heart, which is constantly subject to internal and external perturbations, such as hormonal changes and eletrical impulses. Applying the above model would correspond to the heartbeat being irregular in amplitude and uneven in duration. Similarly, cells encounter numerous internal and external disturbances, yet they must maintain their internal clock accurately for crucial processes such as cell division.

Consequently, there is major interest in more realistic dynamical systems that exhibit oscillations in the form of a limit cycle. This is an isolated closed orbit in phase space, meaning that neighboring trajectories will converge to the limit cycle, ensuring stability despite perturbations [12]. The FitzHugh-Nagumo model contains such a limit cycle and is highly relevant in biology and neurosciences. We will study this in the following section.

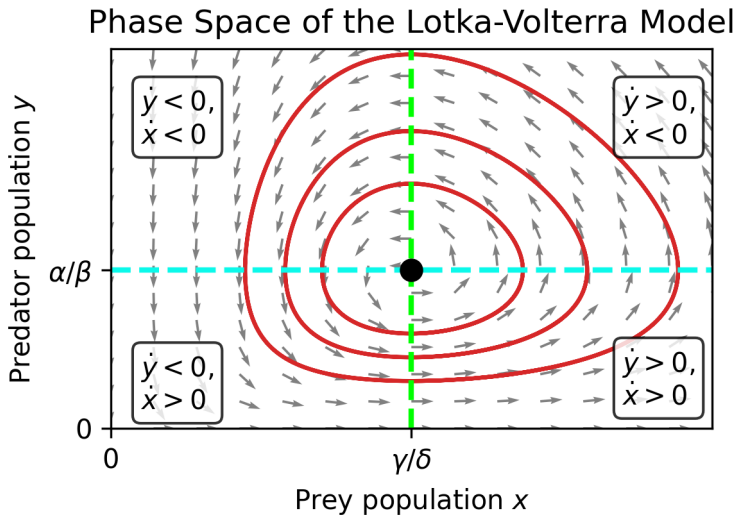


Figure 2.3: **Phase Space of Lotka-Volterra model.** Nullclines are shown as dashed lines, they partition the space into four regions with distinct $\dot{x}$ and $\dot{y}$ values. They intersect at the fixed point, shown in black. The red lines correspond to trajectories with different initial conditions.

## 2.2 FitzHugh-Nagumo

The study of action potentials in neurons has been substantially advanced by the pioneering work of Alan Hodgkin and Andrew Huxley. In their seminal 1952 paper, they developed a quantitative description of the ionic mechanisms underlying the propagation of action potentials in the squid giant axon [16]. For this groundbreaking work, Hodgkin and Huxley were awarded the Nobel Prize in Physiology or Medicine in 1963 [17]. The Hodgkin-Huxley model is a set of nonlinear differential equations that describe the dynamics of the membrane potential and nerve conduction, capturing the essential features of nerve cell excitability and action potential generation [18].

Despite its profound impact on neuroscience and biological research [18], the complexity of the Hodgkin-Huxley model, involving four coupled nonlinear differential equations, presents considerable analytical and computational challenges. To address these challenges, simplified models have been developed that preserve the essential dynamical features of the Hodgkin-Huxley model while being more tractable for analysis [19]. One such model is the FitzHugh-Nagumo (FHN) model, introduced by Richard FitzHugh in 1961 [20] and refined by Jin-ichi Nagumo in 1962 [21].

The FHN model reduces the complexity of the Hodgkin-Huxley model by focusing on two key variables: the membrane potential and a recovery variable. This reduction allows the FHN model to capture the characteristic excitability and oscillatory behavior observed in neurons. Its mathematical properties have made it a popular subject in the study of nonlinear dynamics. Consequently, the FHN model has been used to study numerous phenomena in neuroscience, cardiology, biology, mathematics, physics, electronics, optics, and other systems [19, 22].

We will use the FHN model of the following form:

$$\begin{cases} \dot{v} = v - \frac{v^3}{3} - w + z \\ \tau \dot{w} = v + a - bw \end{cases} \tag{2.7}$$

For neurons, the variable $v$ corresponds to a membrane potential and $w$ to the potassium and sodium channel reactivation. The parameter $z$ represents the membrane current density and $\tau$ is the time-scale separation. The choice of the parameters $a$, $b$, $z$ and $\tau$ determine which characteristic of the system will be present [22, 19].

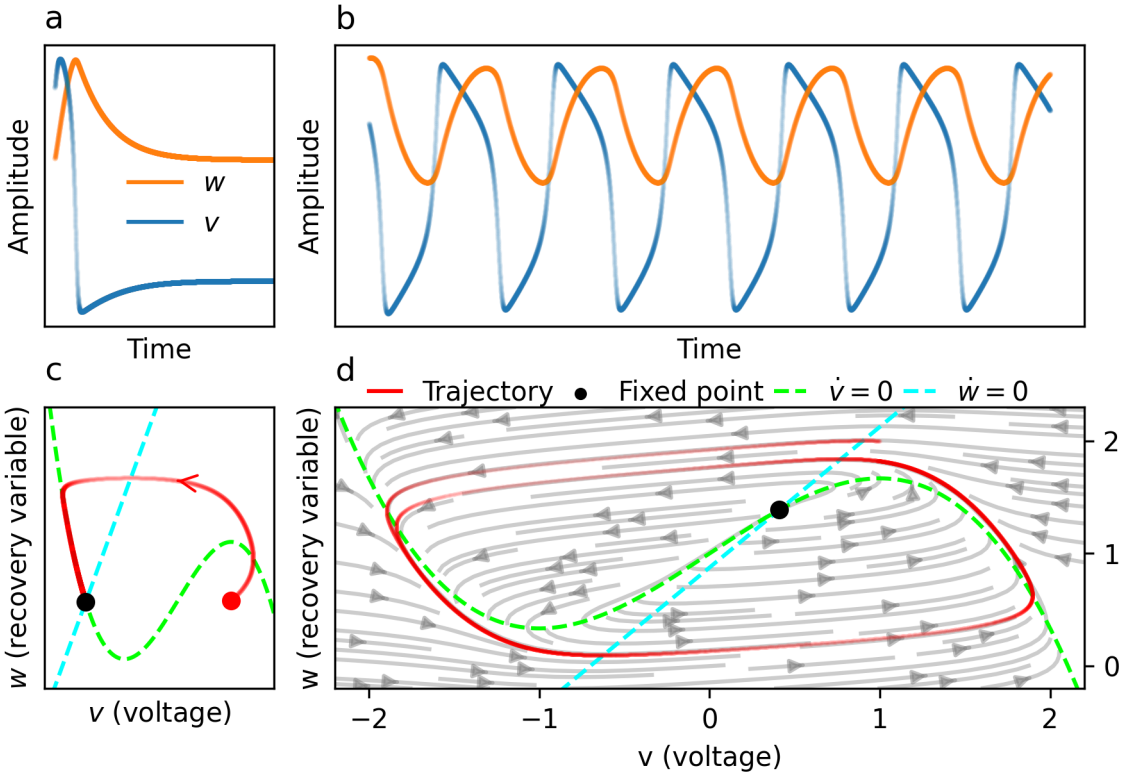


Figure 2.4: **Key features of FHN model.**
**(a)** Excitation: Time series of membrane potential $v$ and recovery variable $w$ with parameters $a = 1.717$, $b = 0.6$, $z = 0$, and $\tau = 7.5$. The time series is more transparent in regions with fewer local points, indicating rapid changes. **(b)** Oscillation: Time series of membrane potential $v$ and recovery variable $w$ with parameters $a = 0.7$, $b = 0.8$, $z = 1$, and $\tau = 7.5$. Similarly, transparency in the time series correlates with regions of accelerated dynamics due to fewer points. **(c)** Phase space of Excitation: Phase space representation with the same parameters as in $(a)$. The red line represents the system's trajectory, with transparency indicating areas of faster flow. **(d)** Phase Space of Oscillatory Behavior: Vector field illustrates limit cycle: Arrows indicate the direction of the vector field, changing direction upon crossing the nullcline. The same parameters as in $b$ are used. The trajectory illustrates transient behavior converging into a limit cycle, with transparency indicating regions of increased flow velocity.

Consider Figure 2.4a and 2.4b illustrating the time series of two different behaviors depending on the choice of constants $a$, $b$, and $z$. One can find more qualitative information by analyzing the system in phase space, as done in the previous section. Figure

2.4c illustrates the property of excitability. This means that a perturbation, for example, displaced from 0 to a point on the $v$-axis with $v > a$ results in a large transient trajectory before returning to the fixed point [22, 19]. The other key characteristic contained in the other time series is the oscillations, as shown in Figure 2.4d. The choice of constants thus affects the overall behavior of the system, since we are interested in the oscillatory state, we will continue with the following constants

$$\begin{aligned} a &= 0.7 \\ b &= 0.8 \\ z &= 1 \end{aligned} \tag{2.8}$$

The oscillatory state of the FHN model emerges as a limit cycle, introduced in the previous section. Figure 2.4d visualizes the convergence of neighboring trajectories to the limit cycle. This characteristic makes the FHN model a self-oscillatory sustained attractor, compared to the conservative nature of the Lotka-Volterra model in Figure 2.3.

The time-scale separation parameter $\tau$ differentiates the fast dynamics of the membrane potential $v$ from the slower dynamics of the recovery variable $w$. Its effect on the oscillations is illustrated in Figure 2.5. According to Eq. (2.7), as $\tau$ increases, the difference between $\dot{v}$ and $\dot{w}$ becomes more pronounced, indicating a greater discrepancy in their rates of change. Hence, stronger time-scale separation leads to relaxation oscillations, while weaker time-scale separation results in more sinusoidal oscillations. For a relaxation oscillator, the system is predominantly located in the slow-changing region, indicated in Figure 2.5c by denser lines [23, 22].

Let us now consider the fundamental structures of the phase space: the nullclines and fixed points. With the same procedure as in the previous section, the following nullclines are found

$$w = v - \frac{v^3}{3} + z \tag{2.9}$$

$$w = \frac{v + a}{b} \tag{2.10}$$

There is a linear and a cubic nullcline present, illustrated in Figure 2.4d. Notice in particular that the nullclines are independent of the time-scale separation parameter $\tau$.

The presence of the FHN model in various fields proves its relevance and broad range of applicability. Since this model is mathematically solved, with known differential equations and corresponding nullclines, and contains interesting biological structures such as limit cycles, it serves as an ideal candidate for testing our machine learning algorithm. This allows us to compare the algorithm's predictions against precisely known values. In the next chapter, we will address this identification of models from measured data.

### Other Dynamical Systems

In addition to the FitzHugh-Nagumo system, this thesis explores other dynamical systems that exhibit limit cycles, which are important for understanding oscillatory behavior.

One such system is the hysteresis in a biological oscillator studied by Novak and Tyson [24]. It is important in the context of biological rhythms and gene regulation. The

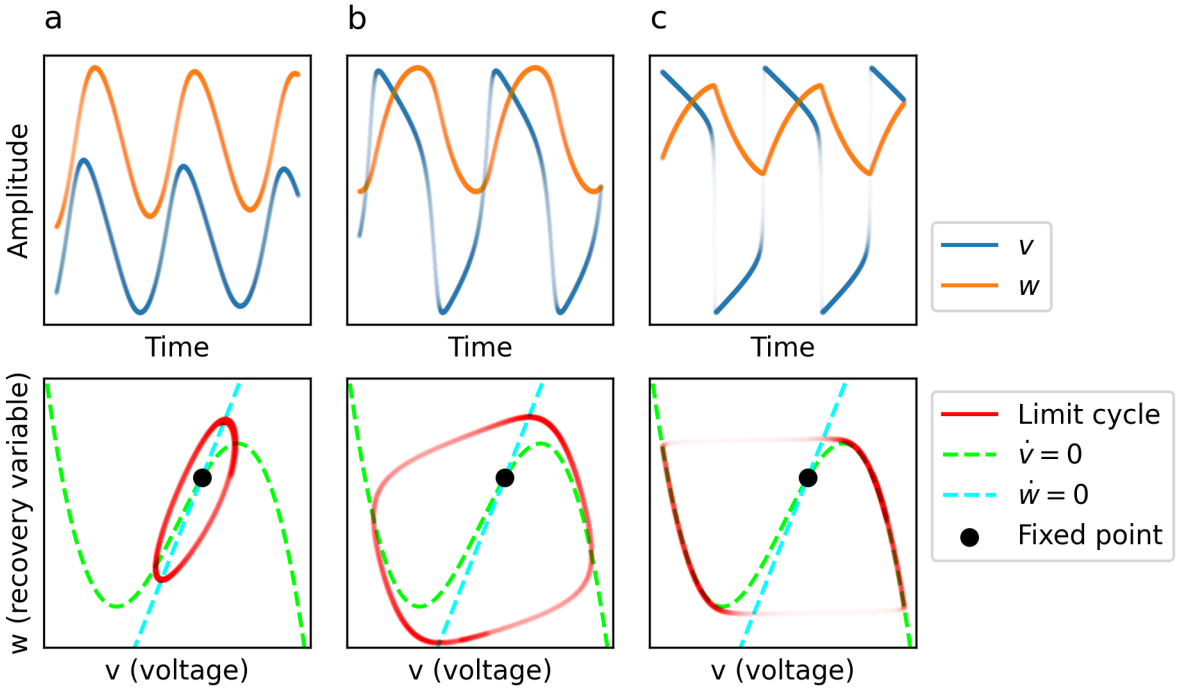


Figure 2.5: **FHN model for varying time-scale separations.**
**a)** Weak time-scale separation. For $\tau = 1$, sinusoidal oscillations are present in the time series. The limit cycle is narrow in the phase space. **b)** Moderate time-scale separation. For $\tau = 5$, relaxation oscillations appear. In the phase space, the limit cycle increases in size, and more of the nullcline lies in the interior. **c)** Strong time-scale separation. For $\tau = 100$, explicit relaxation oscillations show in the time series. In the phase space, the limit cycle follows the cubic nullcline more closely. Transparency indicates the density of points.

governing equations are given by:

$$\begin{cases} \dot{x} = \frac{0.05}{1+y^4} - 0.05x \\ \dot{y} = x - 0.05y - \frac{y}{0.1+y+2y^2} \end{cases} \tag{2.11}$$

Here $x$ represents mRNA concentration and $y$ represents protein concentration.

Another system of interest with a time series that visually resembles the FHN system is the bicubic model [25]. Its name refers to the two present cubic nullclines. The governing equations of the dynamics are:

$$\begin{cases} \dot{u} = -u^3 + u^2 + u - v \\ \dot{v} = -\frac{1}{2}v^3 + \frac{1}{2}v^2 - \frac{1}{3}v + u \end{cases} \tag{2.12}$$

Its nullclines will be examined in more detail in Section 4.7.

# Chapter 3

# Machine Learning

## 3.1 Introduction to Data-Driven Model Identification

### Challenges in Data-Driven Model Identification

In practice, when conducting experiments, the differential equation governing the system is often unknown; only the time series data of the studied parameters are measured. However, it is through these differential equations that we gain a deeper understanding of the underlying physical processes, enabling qualitative insights into the system's behavior [2].

To address this challenge, model identification aims to analyze the data to uncover the mathematical equations governing the dynamical evolution and interaction. The problem of reconstructing these equations from measured time series data is prominent in numerous fields of science and engineering [26]. Previously, this process was inaccessible due to the complexity of the systems and the computational limitations in analyzing vast datasets. However, with advancements in machine learning and data science, identifying these models is becoming increasingly feasible [26, 27]. Despite these advancements, the current data-driven methods still face significant limitations [1]. A possible approach involves regression-based techniques, which attempt to fit mathematical models to the data. For instance, Sparse Identification of Nonlinear Dynamical Systems (SINDy) is a method that aims to find a parsimonious model that describes the system's dynamics. However, methods like SINDy may encounter difficulties due to factors such as noise in the data, limited sampling rates, or complexities in the underlying model structure [26, 28].

Overall, while the advancements in machine learning and data science have made model identification more accessible, there is still a concerning need for developing robust methods able to effectively handle the complexities of real-world systems.

As demonstrated in Chapter 2, nullclines and fixed points are intricately connected to the governing mathematical equations, such that knowledge of one allows for the determination of the other. Even though these system's underlying structures are not known from the experiment, the use of data-driven methods can help extract information about them. Specifically, this thesis will focus on extracting these underpinning structures of

the dynamical system using neural networks.

### Nullclines in Model Identification

Consider again the two dimensional FHN differential Eq. (2.7). We can rewrite the expression for $w$ as follows:

$$\begin{cases} w = v - \frac{v^3}{3} - \dot{v} + z \\ w = \frac{v+a-\tau\dot{w}}{b} \end{cases} \tag{3.1}$$

We can define the right-hand sides of these equations as separate functions $H(v, \dot{v})$ and $K(v, \dot{w})$ respectively. We obtain:

$$\begin{aligned} H(v, \dot{v}) =& v - \frac{v^3}{3} - \dot{v} + z, \\ K(v, \dot{w}) =& \frac{v + a - \tau\dot{w}}{b} \end{aligned} \tag{3.2}$$

Consequently, the two-dimensional FHN differential Eq. (2.7) is equivalent to:

$$\begin{cases} w = H(v, \dot{v}), \\ w = K(v, \dot{w}) \end{cases} \tag{3.3}$$

The functional form of the nullclines is then simply $w = H(v, \dot{v} = 0)$ and $w = K(v, \dot{w} = 0)$ for the cubic and linear nullcline respectively.
The linear nullcline can also be rewritten by applying the inverse function, $v = K^{-1}(w, \dot{w} = 0)$. In contrast, the cubic nullcline lacks an inverse function, as there is no one-to-one correspondence between $v$ and $H(v, \dot{v} = 0)$, this is especially clear in Figure 2.4d.

In the following section, we examine the methods for extracting the nullcline structure $H$, $K$, and $K^{-1}$ from the time series data.

## 3.2 Heuristic Approaches to Identify Nullcline Structure from Data

The time series data is acquired by numerically solving the differential equations of the FHN system, Eq. (2.7), using the Euler method. The initial point $(v_0, w_0) = (1, 2)$ is chosen and six oscillations are studied using 15000 points. The derivatives $\dot{v}$ and $\dot{w}$ are calculated using the forward difference method. An illustration of this solution in time and phase space is shown in Figure 2.4b and 2.4d. Let us first consider the cubic nullcline, with the corresponding function $H(v, 0)$. In the following part, we will extend the phase space into three dimensions and examine two specific approximation methods for nullcline recovery.

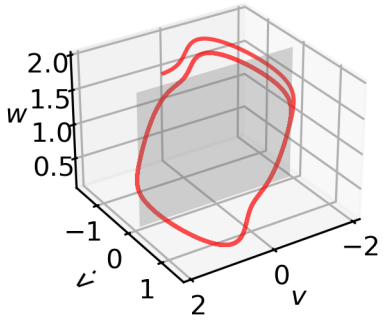


Figure 3.1: **Three-dimensional extension of the phase space for cubic nullcline. (a)** Time series data is plotted in red. In grey is the surface defined by the plane $\dot{v} = 0$. Notice that, for visual purposes, the values on the $v$-axis are reversed.

## Three-Dimensional Phase Space Analysis

The time series data can be visualized on a three-dimensional plot, as illustrated in Figure 3.1a. The inputs $v$ and $\dot{v}$ of the cubic nullcline functional, $H(v, \dot{v})$, are placed on the $x$- and $y$-axis. The output $w$ is put on the $z$-axis. This method gives an extra dimension to the phase space of Figure 2.4d. The nullcline is then located in the plane where $\dot{v} = 0$. This three-dimensional form increases insights for recovering the nullcline from the trajectory. In the following subsection, we will apply two interpolation methods that predict a surface with the trajectory as a boundary. The predicted nullcline forms at the intersection with the $\dot{v} = 0$ plane.

## Approximation Methods for Nullcline Prediction

Having acquired a three-dimensional structure wherein predictions can be made, we now turn to investigate two approximation methods aimed at uncovering the nullcline structure. In doing so, we will highlight the inherent limitations of these methods. The first approach involves fitting a two-dimensional surface using Laplace's equation, with the limit cycle serving as the boundary condition [29]. Laplace's equation is central to various fields of mathematical physics, including electrodynamics and hydrodynamics [29, 30]. The equation provides a method to fit the function $f(v, \dot{v})$ within the region enclosed by the limit cycle, where $f$ satisfies Laplace's equation

$$\frac{\partial^2 f}{\partial v^2} + \frac{\partial^2 f}{\partial \dot{v}^2} = 0 \tag{3.4}$$

Solutions to this equation are called harmonic functions and satisfy the property that the value at each point $f(v, \dot{v})$ is an average of the points around it. Consequently, the surface over the interior of the limit cycle will have local maxima only on the boundary. The nullcline is the intersection between this surface and the $\dot{v} = 0$ plane. Thus, the mentioned property is particularly advantageous because, within a limit cycle, the absolute maximum values $w$ of the nullcline $H(v, \dot{v} = 0)$ of Eq. (3.2) are less than or equal to the maximum values $w$ of the limit cycle. Applying this approximation method to the FitzHugh-Nagumo system can be done by dividing the $(v, \dot{v})$-plane into a grid with

gridsize 200 and numerically applying relaxation methods [31]. The resulting prediction of the nullcline is shown in Figure 3.2a, with the mean squared error between the real and predicted nullcline equal to $8.5 \times 10^{-3}$.

The second approximation method is using a piecewise linear interpolator. First, the data is triangulated, then linear barycentric interpolation is performed on each triangle [32, 33, 34]. The resulting nullcline prediction is shown in Figure 3.2b, and the mean squared error of the prediction is $4.6 \times 10^{-4}$.

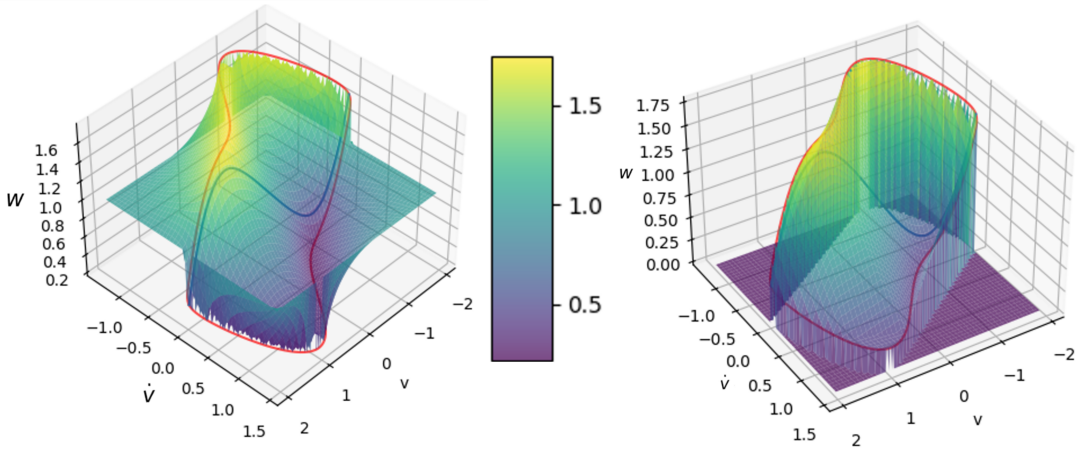


Figure 3.2: **Nullcline prediction using approximation methods.**
**(a)** Three-dimensional extension of the phase space. Fitting a harmonic surface to the boundaries of the limit cycle in red by solving the Laplace equation iteratively using a convergence tolerance of $5 \times 10^{-5}$. At the intersection with the plane defined by $\dot{v} = 0$ the nullcline, in blue, is predicted. Legend depicts the value of $w$. **(b)** Three-dimensional extension of the phase space. Fitting a surface to the boundaries of the limit cycle in red by using a piecewise linear interpolator. At the intersection of the predicted surface with the plane defined by $\dot{v} = 0$, the nullcline, in blue, is predicted. Legend depicts the value of $w$.

Both methods are capable of accurately extracting the cubic nature of the nullcline. The downside of fitting a surface using Laplace's equation and the triangular interpolator is that they are inflexible. The former will tend to perform better on really specific differential equations due to the second partial derivatives. The latter performs better in predicting surfaces where a linear correlation is present, like in the case of FHN, explaining its accurate performance. These approximation methods are thus not applicable to every system, and it is also not possible to know when to apply them because prior knowledge of the form of the differential equations is unknown when studying the time series of a dynamical system.

Thus it becomes apparent that we require more general approximations of the phase space landscape that can overcome these limitations. Such a method could be the application of neural networks which act as universal approximators. In the following chapter we will explain and explore how a neural network could be applied explicitly.

## 3.3 Neural Networks

In this section, we explain what a neural network is and how it is able to learn from data. We discuss the different parameters that affect the performance of a neural network and finish by examining the different methods to optimize and ensure a robust neural network model. This section is mainly based on the works of Gurney (1997) and Priddy and Keller (2005) [35, 36].

### Neural Network Structure

Neural networks are computational models inspired by the structure and function of the human brain. They are a fundamental concept in machine learning, a branch of artificial intelligence focused on creating systems that can learn complex patterns from data. This characteristic is desired in many fields and contains numerous applications such as image classification, large language models, etc, [37, 38].

Figure 3.3a illustrates a neural network. Similar to the neurons in our brains, neural networks consist of interconnected nodes organized into layers. These nodes are the processing units of the network. On the left, the input layer is portrayed, this is where the input enters the neural network. The output layer on the right returns the output. In between are the hidden layers, where most of the computations required by the network are performed. The neural network takes in data, trains itself to recognize the patterns in it, and predicts the outcomes for a new set of similar data, with each layer directly supplying data to the next layer. This type of network, where data flows through the layers, is called a feedforward network. Different architectures of the layers, nodes, and interconnections can be used depending on the task like graph neural networks for particle systems, convolutional neural networks for object detection, autoencoders for data compression, etc, [39, 40, 41]. In the strengths of the interconnections, known as weights, lies the prediction ability of the network. The process of modifying these weights based on training data to enhance the prediction capabilities of the network is called learning. The number of layers and nodes are hyperparameters of the system, meaning that they are tunable and affect the architecture and optimization of the network. Larger networks can handle more complexity but are also more prone to overfitting: they may perform well on training data but poorly on new, unseen data. Meanwhile, smaller networks might not be able to handle the complexity of the problem or require significantly more training time [36, 42, 35, 43, 44, 45].

### Node Operation

Let us consider a single node to better understand how a neural network learns. This is illustrated in Figure 3.3b. All the incoming interconnections have an associated weight $w_i$, and when data $x_i$ from a previous node flows into the current one, all the incoming values $x_i$ are multiplied with the corresponding weights $w_i$. The sum of all these products provides the node with its value.

$$\sum_i x_i w_i \tag{3.5}$$

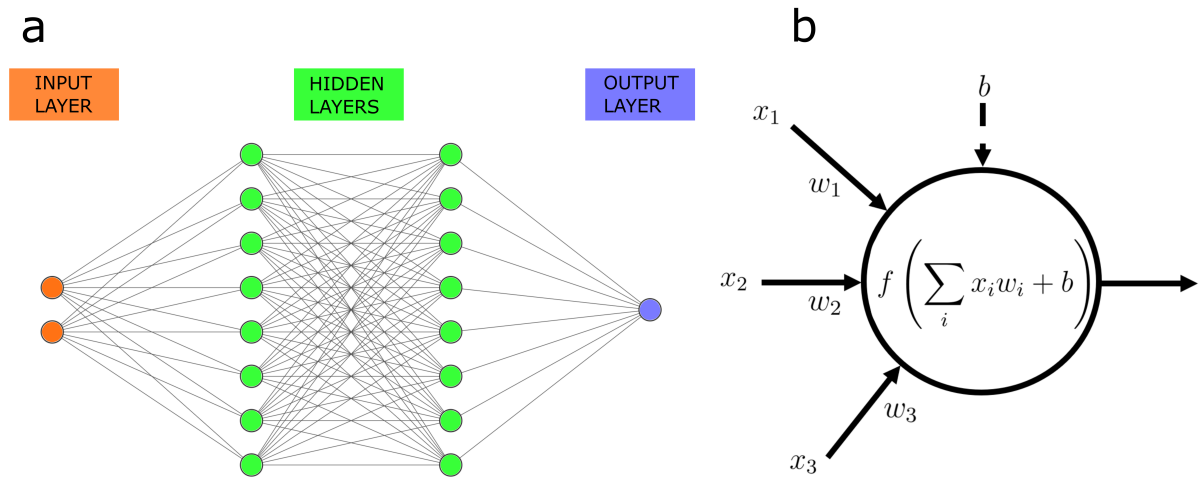


Figure 3.3: **Neural network architecture**
**(a)** A neural network with 2 nodes as input layer, followed by 2 hidden layers each with 8 nodes and one node as output layer.
**(b)** A single node visualized. The values $x_i$ flow through the arrows and are multiplied with the corresponding weight $w_i$. In the node, the activation function is applied to the sum, together with the bias $b$.

But a network in this form can only perform linear regression. Due to our interest in complex systems, nonlinearity is introduced by applying an activation function $f$ on the sum of the inputs. To further determine the activation of the node, an extra weight called the bias $b$ is added to the node. The resulting output of the node can then be calculated as:

$$f\left(\sum_i x_i w_i + b\right) \tag{3.6}$$

When a neural network is initially created all the weights are initialized randomly.

## Training Process

Consider the known data and rewrite it in the form $(x_1, \ldots, x_n, y)_i$ where $i \in [1, N]$, with $N$ the amount of samples. This is a convenient way to express the data for neural networks that are trained with $(x_1, \ldots, x_n)_i$ as input and learn to predict $y_i$. In the training process, the neural network is given sets of input data $(x_1, \ldots, x_n)_i$ with $n$ the number of input nodes, and predicts some output $y_{p_i}$. At first, the prediction $y_{p_i}$ will not be close to the real value $y_i$. The error of the prediction is measured using a loss function, the mean squared error (MSE) is suitable in most cases:

$$\text{MSE} = \frac{1}{N}\sum_i^N (y_i - y_{p_i})^2 \tag{3.7}$$

The neural network will change its weights to minimize the loss function. This adjustment of the weights is calculated using gradient descent, a mathematical optimization algorithm that determines the direction in which the weights should change to minimize

the MSE [46]. A crucial hyperparameter in this process is the learning rate, which dictates the magnitude of weight adjustments in response to the calculated error. Selecting an appropriate learning rate is essential for the efficient training of a neural network. If it is too small, the training process becomes slow, potentially causing the network to become trapped in a local minimum of the loss landscape, preventing the network from finding the most optimal solution. Conversely, suppose the learning rate is too large. In that case, the weight updates can be excessive, causing the network to oscillate around the error minimum without converging to the best possible solution [35, 36].
The derivatives used in the gradient descent are calculated through a process known as backpropagation. This process allows the neural network to learn and is applied to all training data within the time series. A complete pass through the entire set of training data, including the updating of weights, is referred to as an epoch. After each epoch, the loss function converges more to its minimal value [46, 47]. In 2014, a significant advancement in optimization techniques was made with the introduction of Adam (Adaptive Moment Estimation) [48]. This algorithm enhances the gradient descent algorithm. It has gained widespread adaptation and has for this reason been employed in this thesis.

## Activation Function

The complexity of the network should correspond to the complexity of the problem it is trying to solve. To adjust the balance between these two during training, activation functions are used. They are a hyperparameter of the neural network that adds the needed nonlinearity [49]. Having briefly introduced them in Eq. (3.6), we now discuss three examples, all employed in this thesis. First, consider the ReLU (rectified linear unit) activation function, it is defined as

$$f(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

and is illustrated in Figure 3.4. When the input is negative, the output is zero, and when the input is positive, the output equals the input. ReLU is the most common activation function due to its computational speed. However, it has limitations, such as under-utilizing negative values and providing limited nonlinearity compared to other activation functions. The second activation function to discuss is the popular logistic sigmoid activation function, defined as

$$f(x) = \text{sigm}(x) = \frac{1}{1 + e^{-x}} \tag{3.9}$$

which is illustrated in Figure 3.4. For very negative input values, the output is close to zero, and for large inputs, the output is close to 1. By returning values in the range of 0 and 1 the weights are more stable [50]. However, the nonzero-centered outputs can negatively affect convergence to optimal weights [51]. Additionally, sigmoid's exponential nature makes computation complex and inefficient. Notice that the left and right parts of the sigmoid become increasingly flat leading to gradients close to zero, which in larger networks results in minimal updates of the weights. This is known as the vanishing gradient problem and hinders effective backpropagation.

To overcome a part of the limitations of the sigmoid activation function, researchers developed the hyperbolic tangent activation function. It is defined as

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.10}$$

This function is illustrated in Figure 3.4, where one sees that the values are contained between $-1$ and 1. Its zero-centered nature helps speed up calculations [51]. However, similar to the sigmoid, the tanh function also suffers from the vanishing gradient problem because its output saturates for high and low inputs. It also shares the computational complexity due to its exponential components. More information about activation functions and how they compare is provided by [49].

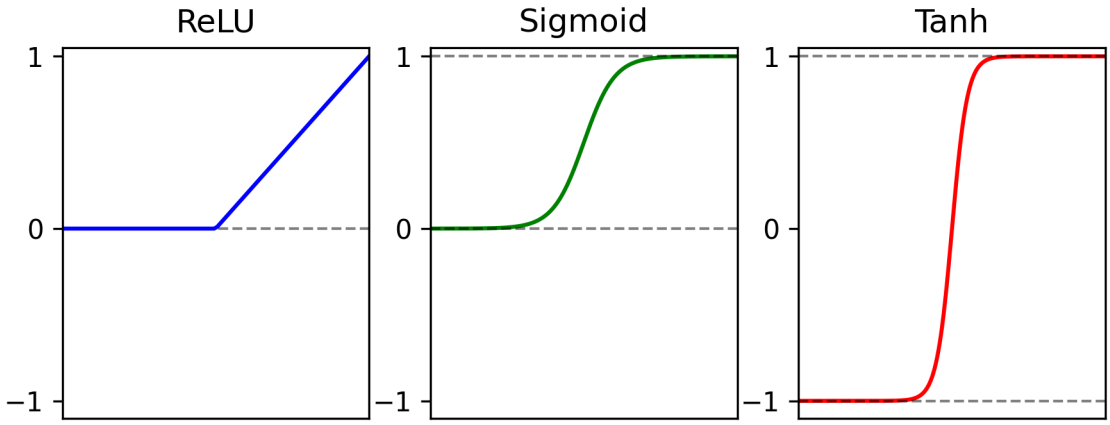


Figure 3.4: The activation functions ReLU, Sigmoid and Tanh.

## Normalization Method

A commonly used tool to facilitate the learning process is the preprocessing of input data. In particular, the most adopted data normalization techniques include z-score and min-max normalization. They are both employed in this thesis. Scaling the input data ensures that no input is dominant in the training process. It decreases the MSE of prediction and additionally, it has been shown that the amount of epochs needed for convergence is lower when the normalization methods shift the average of each input variable close to zero [51]. Let us first consider the z-score normalization. The data is transformed using the following equation

$$x_{i_{\text{norm}}} = \left[\frac{x_i - \mu}{\sigma}\right] \tag{3.11}$$

where $\mu$ and $\sigma$ represent the mean and standard deviation of the input. The z-score normalization technique reduces the effect of outliers in the data. The second normalization technique is the min-max normalization. This method rescales the values of the input data to lie within a range of $[0, 1]$, performed using the following equation

$$x_{i_{\text{norm}}} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{3.12}$$

The advantage of min-max normalization is that it preserves the same underlying distribution of the input data without introducing any bias [36, 52].

## Validation and Generalization

Training neural networks should be done carefully to avoid overfitting to the training data and ensure generalizability. That is why the performance of a neural network should be tested on a separate set of unseen data from the same distribution. This data is referred to as validation data and also originates from the measured data. It is important to balance the amount of training data and validation data. If one uses too much training data, the performance of the neural network on the validation data will not accurately represent the generalizability. Conversely, not enough training data might result in increasing inaccuracy of the weights. The mean squared error of the predictions on the validation data indicates the performance of the neural network on the validation data, known as the validation error. To ensure that all features of the data are included in the training data, the separation of data must be sampled randomly [36, 53]. Overfitting, underfitting and the correct fit are illustrated in Figure 3.5a
In this thesis, a ratio of 80% training data and 20% validation data is chosen, as this configuration achieved reliable results.

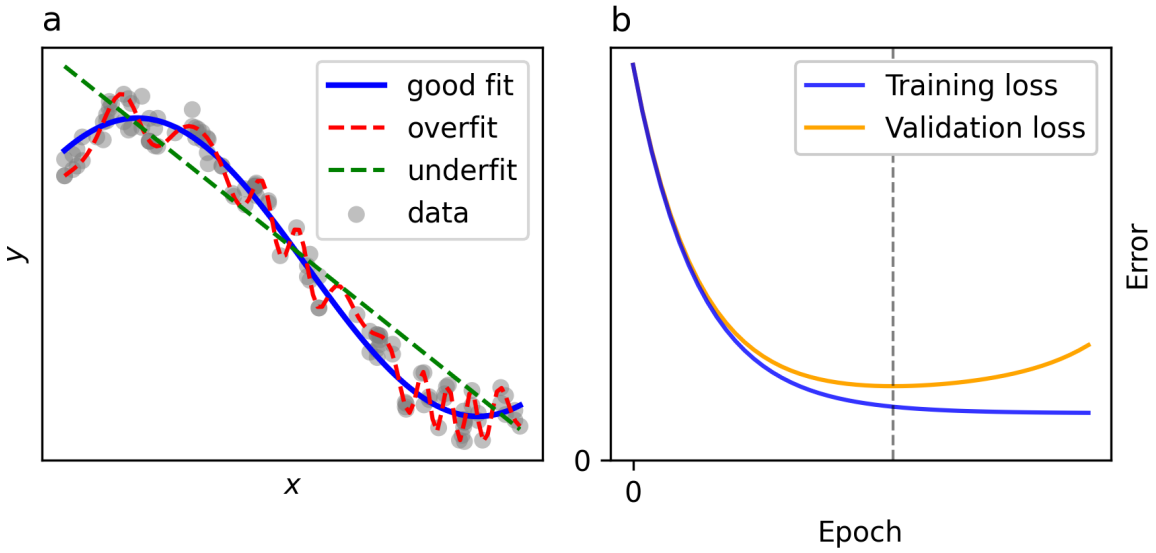


Figure 3.5: **Data fitting**
**(a)** Fitting $y$ in function of $x$. Indicating underfitting, overfitting, and good fit.
**(b)** Training loss and validation loss during training. Vertical stripes indicate the optimal performance of the NN to the validation error.

To qualitatively measure the generalizability and the performance of the network one must thus seek to minimize the loss function on the validation data. At first, the loss functions of the training and validation error will follow the same trend, but the validation curve will converge sooner, meaning it has reached its minimum. This minimum indicates

the optimal weight values of the neural network. If one would keep training the neural network, overfitting would occur, increasing the loss function for the validation data. This is illustrated in Figure 3.5b.

In this section we have seen that many hyperparameters influence the performance of a neural network, including the number of layers and nodes, learning rates, number of epochs, activation functions, and normalization methods. A priori, it is challenging to determine the best combinations due to their sheer number and the complex interactions between them. However, the performance of the network critically depends on the chosen hyperparameters. Selecting the appropriate hyperparameters is essential for developing a robust and effective algorithm. Therefore, optimizing them is a major focus of this thesis.

## Nullcline Prediction

The ability of NNs to learn complex patterns from data is particularly helpful for uncovering hidden structures from time series data. This justifies the application of NNs for nullcline reconstruction.
To address the problem of nullcline prediction using neural networks, already introduced in Section 3.2, we examine the already introduced Eq. (3.3). As an example, we will consider the cubic nullcline with function $H(v, \dot{v})$, but this method works analogously for the other functions. To find the nullcline structure $H(v, \dot{v} = 0)$ we would like a neural network that takes in values of $v$ and returns the corresponding value $w = H(v, \dot{v} = 0)$. To create such a network, we shall train it using $v$ and $\dot{v}$ from the time series as inputs, and test its performance on the predicted output $w$. The architecture of the neural network thus consists of two input nodes $(v, \dot{v})$ and one output node $w$. Then, to test its performance in recovering the nullcline, the mean squared error between the actual and predicted nullcline on $\dot{v} = 0$ is calculated. It will be referred to as the nullcline error. The best-performing neural network will have the lowest nullcline error.
It should be stressed that the nullcline error can only be calculated because the nullcline is known for this system. In practice, the only quality of prediction that is available is the neural network validation error of the training process. Even when training with the same time series data, the performance of the neural network can vary due to the inherent stochasticity of the training process [54], such as the random initialization of the weights, and the arbitrary separation of data into training and validation sets.
This thesis will aim to find the appropriate hyperparameters that result in a high correlation between the validation error, which represents the quality of predicting the time series trajectory, and the nullcline error, which represents the quality of predicting the nullcline. This approach will result in a benchmark for determining the nullcline structure from the time series without knowing the governing equations of the dynamical system.

To measure the correlation between the validation and nullcline error the Pearson correlation coefficient (PCC) is employed [55]. It is defined as

$$\mathrm{PCC} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.13}$$

It is applicable for given data of the form $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. The mean is represented as $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ where $n$ represents the sample size. When the PCC is equal

to 1 there is a perfect positive linear correlation. Meanwhile a value of $-1$ corresponds to a perfect negative linear correlation. The variables are linearly unrelated when the PCC equals 0. Its importance will be demonstrated in the beginning of Chapter 4.

By leveraging the analytically solved FHN model, we can use the known nullcline structure to develop a robust algorithm that strongly correlates the accuracy of the time series prediction and the accuracy of the nullcline predictions, as indicated by the PCC value.

## 3.4 Challenges in Trajectory-Based Model Identification

Model identification is a crucial task in dynamical systems analysis, where the goal is to determine the underlying differential equations that produce a given trajectory in phase space. This section explores the associated challenges, focusing on the derivation of nullclines and the implications for model identification.

### Deriving Nullclines from Trajectories

Given a continuously differentiable differential equation with an initial condition, the existence and uniqueness theorem states that there exists a unique trajectory in the phase space [14].
The principle of this thesis is the opposite: to use model identification and determine the differential equation belonging to a given trajectory. However, the reverse theorem does not hold. A concrete counter-example is illustrated in Figure 3.6. It illustrates how the same trajectory corresponds to three different differential equations, each characterized by distinct nullclines.

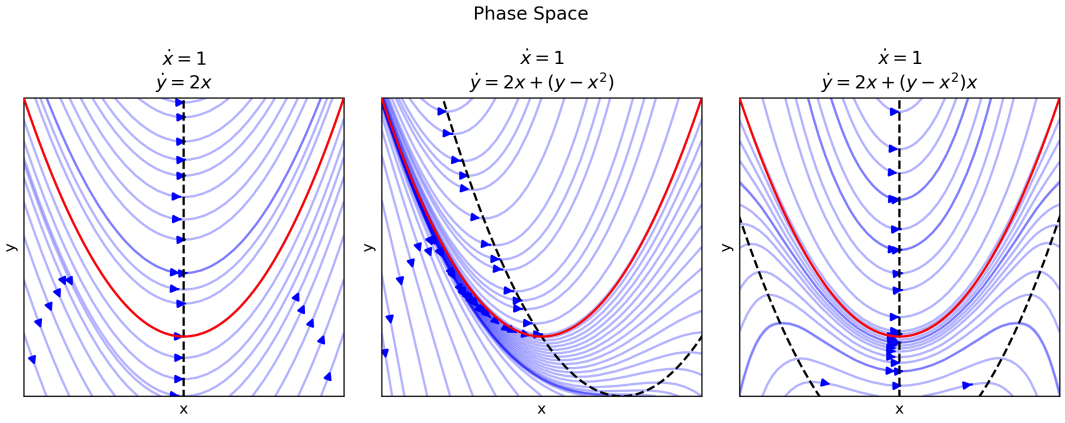


Figure 3.6: **Phase Space of same trajectory, different differential equation** Numerous trajectories are shown in blue. In red is the $y = x^2$ trajectory, produced by all three differential equations. The nullclines for each system are shown as a dashed black line. Model complexity increases from left to right, with higher-order terms entering the differential equation.

This would imply that attempting to identify a differential equation from a trajectory is misguided. However, using knowledge besides the trajectory allows the determination

of a biologically realistic model [56]. Notice that for the second and third differential equations, trajectories starting in the same neighborhood as the red trajectory increasingly deviate from it over time. Such phenomena are uncommon in our systems of study, in which we assumed that perturbations recover back into the oscillations of the limit cycle. This preference for simple models, like the first differential equation, is also a fundamental assumption for other model identification research, like SINDy, which aims to identify sparse models [26, 25, 28].

## Recovering Nullclines outside the Limit Cycle

Neural networks only allow the prediction of the nullclines within the limit cycle, this is illustrated with the following example. Figure 3.7a presents a high-quality prediction of the cubic nullcline. Within the boundaries of the limit cycle it achieves a nullcline error of $8.3 \times 10^{-5}$. This result represents on of the most accurate predictions in this thesis. However, outside the limit cycle's boundaries, the prediction error increases exponentially. This behavior highlights a fundamental limitation of neural networks, while they interpolate exceptionally well within known data ranges, their performance deteriorates when extrapolating beyond these boundaries [57].

## Inferring Differential Equations from Nullclines

This subsection is inspired by [58].

Accurately predicting the nullcline within its boundaries does not guarantee that the inferred differential equation is correct, as will be demonstrated by the following example. Consider two different systems and their differential equations. System 1 is defined as:

$$\begin{cases} \dot{x} = 1.13x - 0.9y - 1.17x^3 - 0.55x^2y^2 \\ \dot{y} = 0.69x - 0.32y - 0.13xy^2 \end{cases} \tag{3.14}$$

and system 2 as:

$$\begin{cases} \dot{x} = 1.13x - 0.93y - 1.17x^3 \\ \dot{y} = 0.69x - 0.32y \end{cases} \tag{3.15}$$

The nullclines $\dot{x} = 0$ and $\dot{y} = 0$ for both systems are illustrated in Figure 3.7b. Within the illustrated boundaries, they are nearly identical. However, once outside these limits they differ significantly. As neural networks are only able to predict nullclines within the limit cycle's boundary it seems unfeasible to infer which differential equation it belongs to. The result becomes evident when analyzing the phase space of each system in Figure 3.7c. For both systems a limit cycle is present, but their behavior near this attractor differs. For the second system, all trajectories in the vicinity flow into the attractor, whereas for the first system, a small variation in the initial position can result in trajectories diverging away from the attractor instead of going into it. The latter property is less appealing because biology contains many systems that recover after perturbations, as discussed in Chapter 2. For this reason, the second system is preferred, not only for its realism but also for its simplicity, reflected in the sparser (containing fewer terms) form of Eq. (3.15)

compared to Eq. (3.14).

This thesis will solely focus on nullcline reconstruction. The following step, to infer the differential equation from the nullcline, can be achieved using sparse regression methods like SINDy. They already incorporate the preference for simpler, more realistic physical systems by their sparsity-promoting techniques [26, 25].

## The Utility of Transients in Model Verification

The previous subsections highlighted the difficulties of extrapolating beyond the interior of the limit cycle, and in recovering the differential equations from the nullclines. These problems can be mitigated by the use of transients. The transient is the part of the dynamics where it is traveling to the attractor. It is the trajectory between the perturbed initial condition and the limit cycle.
Transients can provide extra information that serves as feedback to verify the proposed differential equation. When transients move far away from the attractor they uncover more of the phase space. This helps neural networks interpolate outside the boundaries of the limit cycle. Additionally, neural networks allow multiple transients to be studied simultaneously. If the phase space were filled with transients neural networks would be able to increase their operational domain significantly, Furthermore, transients can affirm identified models by comparing the simulated and experimental transients.

However, the applicability of transients is largely limited to systems studied in highly controlled experimental environments. This is required because transients are usually hard to distinguish from artifacts, unlike limit cycles which appear as oscillations. Creating these transients can also be challenging, particularly in systems that are difficult to perturb, such as the climate, where control is inherently complex.

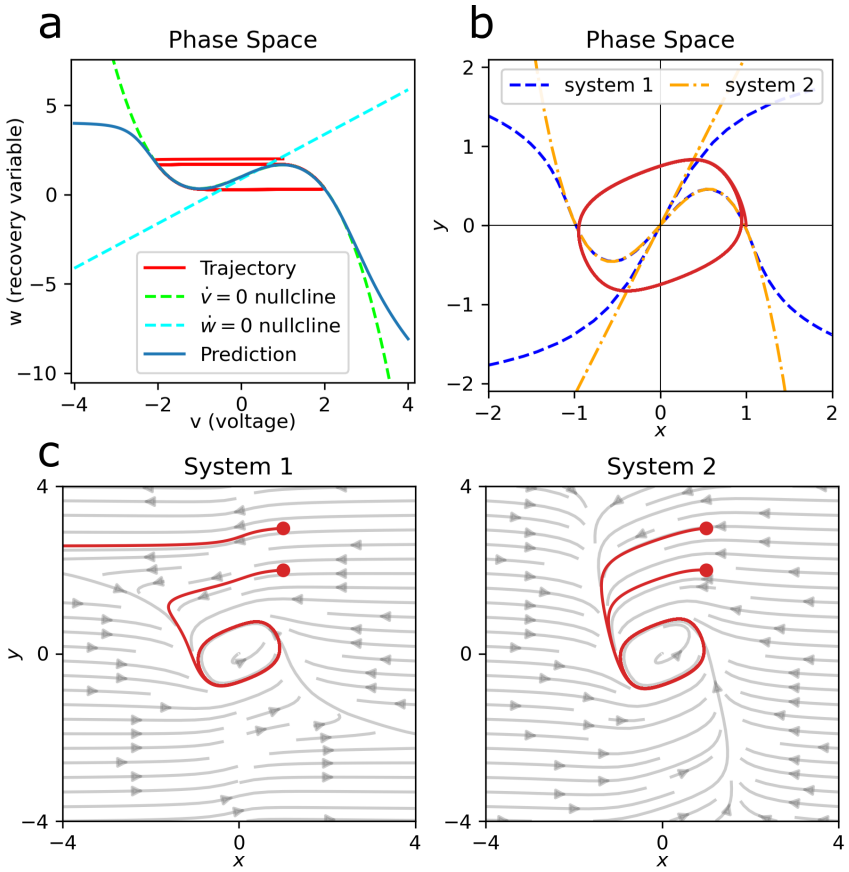


Figure 3.7: **Limitations of predicting within boundary of limit cycle (a)** A system with a trajectory and nullclines is shown. Blue is the visualization of an accurate prediction of the cubic nullcline within the limit cycle. Outside of the limit cycle, the prediction deviates significantly. **(b)** The phase space is shown, in red is the limit cycle. It depicts how the nullclines of two different dynamical systems are identical within the limit cycle but behave differently outside its boundaries. **(c)** The phase spaces of the systems of (b) are shown with two chosen trajectories in red. The left figure indicates that a small perturbation to the initial condition drastically affects the evolution of the system. Meanwhile, the systems in the right figure will both proceed into the limit cycle. This figure is inspired by [58].

# Chapter 4

# Results

The optimization of the neural network was a major part of this thesis. Accordingly, a large part has been devoted to explaining the reasoning and conclusions, demonstrated in Section 4.1. After the benchmark neural network has been designed, its robustness will be compared for different time-scale separations in Section 4.2. We will consider the linear nullcline in Section 4.3. We will employ this in Section 4.4 to predict the fixed point and analyze the flexibilities in nullcline recovery. We will provide a possible explanation for the noteworthy performance of the hyperparameter ReLU in Section 4.5. The limitations of the proposed benchmark are assessed in Section 4.6 pertaining to varying data resolution. Finally, we test the proposed method for another dynamical system in Section 4.4.

**Purpose of PCC**

To demonstrate the importance of the PCC, consider the following example. Two neural networks with different hyperparameters have each been trained 40 times. We consider two networks that have a similar distribution of nullcline error and validation error but with different PCC. The performance of each is quantified by examining the average prediction of the five best-trained networks, meaning they had the lowest validation error. Figure 4.1a portrays the network where the PCC equals 0.41, the nullcline error here is $1.43 \times 10^{-2}$ and it manages to recover the form of the nullcline, especially at the left side of the nullcline. Meanwhile, Figure 4.1b portrays the prediction of the other network with the other set of hyperparameters. The PCC is $-0.22$, resulting in a higher nullcline error of $6.98 \times 10^{-2}$. This neural network with this set of hyperparameters struggles to recover the nullcline structure. Especially at the right side of the cubic nullcline where there is also a high standard deviation.

This highlights that the PCC is an important metric to assess the reliability of the neural network. We will use the solved FHN model to develop such a reliable algorithm in Section 4.1. The subsequent sections will apply the developed algorithm and test its robustness on new dynamics (Section 4.2), temporal resolution (Section 4.6) and another system (Section 4.7).

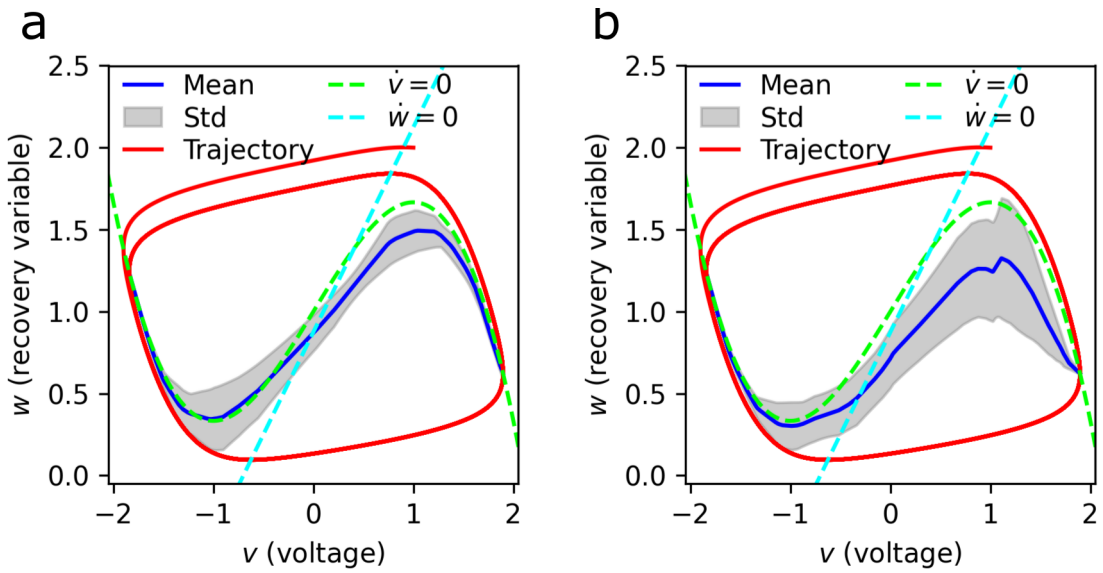


Figure 4.1: **PCC importance in predicting nullclines**
**(a)** PCC = 0.41. Mean and standard deviation (std) of nullcline prediction using hyperparameters: learning rate 0.01, 500 epochs, activation function ReLU, normalization method min-max, 2 hidden layers each 8 nodes. **(b)** PCC = −0.22. Mean and standard deviation of nullcline prediction using hyperparameters: lr 0.01, 500 epochs, activation function ReLU, no normalization method, 2 hidden layers each 8 nodes.

## 4.1 Optimizing Neural Network

At the end of Section 3.3, we presented the numerous hyperparameters that must be tuned carefully for optimal performance. Hyperparameters such as the learning rate (LR) and epoch, activation function and normalization method, as well as nodes and layers, are intricately related and are optimized together. It is unfeasible to test the performance of all possible combinations of hyperparameters. Therefore, we propose the following systematic optimization methodology: vary the values of the hyperparameters, if a particular value of a specific hyperparameter consistently performs best, its value is fixed. Continue this process until all hyperparameters have been optimized.

To start, the LR and epochs are optimized simultaneously with the normalization method and activation function. As a starting point, we choose the model with a moderate time-scale separation of $\tau = 7.5$. The neural network consists of two hidden layers, each with 8 nodes. The LRs and epochs that are compared are a LR of 0.01 with 500 epochs and a LR of 0.005 with 1000 epochs. Next, we will assess the model for strong time-scale separation $\tau = 100$. In the end, we will vary the neural network size and propose a benchmark.

### 4.1.1 Optimizing for $\tau = 7.5$

#### Validation Error

First, consider the configuration using a LR of 0.01 and 500 epochs. The comparison of the training and validation error during training is illustrated in Figure 4.2a. This figure depicts the mean error of 40 networks with the corresponding standard deviation. It can

be said that the validation error closely follows the training error, being slightly higher. There is no sign of overfitting present. Most configurations seem close to convergence, except for the activation function Tanh combined with min-max normalization. Also, the networks using the activation function Sigmoid do not seem close to convergence. Even though convergence has not yet been reached, the validation error is already low, especially compared to the other activation functions.
Figure 4.2b illustrates the boxplot of the final value of the validation error, representing the performance of the neural network in predicting the target time series values within the known time range. For all three normalization methods, ReLU has the highest validation error, followed by Tanh, with Sigmoid achieving the lowest validation error. The normalization using z-score performs the worst, meanwhile, the others perform equally well. Even though the medians vary, the relative performance of the activation functions stays in proportion over the different normalization methods. Let us first consider the performance of the normalization methods. The general trends are presented in Figure 4.2c. The medians of no-norm and min-max are both similar and lower than the z-score, but high variability is present for all. The performance of the activation functions is depicted in Figure 4.2d. ReLU has the highest validation error, Tanh has a lower error but Sigmoid has the lowest. This analysis has also been conducted for a LR of 0.005 with 1000 training epochs, with similar results as shown in Figure A.1. Only the total error was on average 0.8 orders of magnitude lower.
These results imply that the choice of normalization method and activation function significantly impacts the training performance of a neural network. Therefore, it is essential to explore various combinations to identify the most effective pairing.

## Nullcline Error

We continue our analysis with these neural networks since the validation errors of the neural networks converged into favorable values. The following step is to analyze the performance of the networks in predicting the nullclines. Similarly as for the validation error, Figure 4.3 illustrates the performance in nullcline prediction of the different activation functions and normalization methods using box plots. Consider the normalization method no-norm. Contrary to its performance in the validation error, ReLU's performance here is greater than Tanh and Sigmoid, which both have similar performance. However, the variability of the nullcline error of ReLU is greater, also for the other normalization methods. The z-score performance is slightly worse than the no-norm, but the relative performances stay similar. For min-max the performance of Tanh and Sigmoid has suddenly increased significantly. This evidently indicates an interaction effect between the normalization method min-max and the activation functions Tanh and Sigmoid. The same conclusions can be made for a LR of 0.005 with 1000 epochs as seen in Figure A.2, although the nullcline error, like the validation error, is slightly lower.

## PCC

As explained in Section 3.3, it is crucial that the validation error and nullcline error are correlated. The general distribution of all the trained networks are demonstrated in Figure 4.4a. Neural networks located in the bottom left corner achieved low nullcline and validation error. Especially the normalization methods min-max, and to a lesser extent

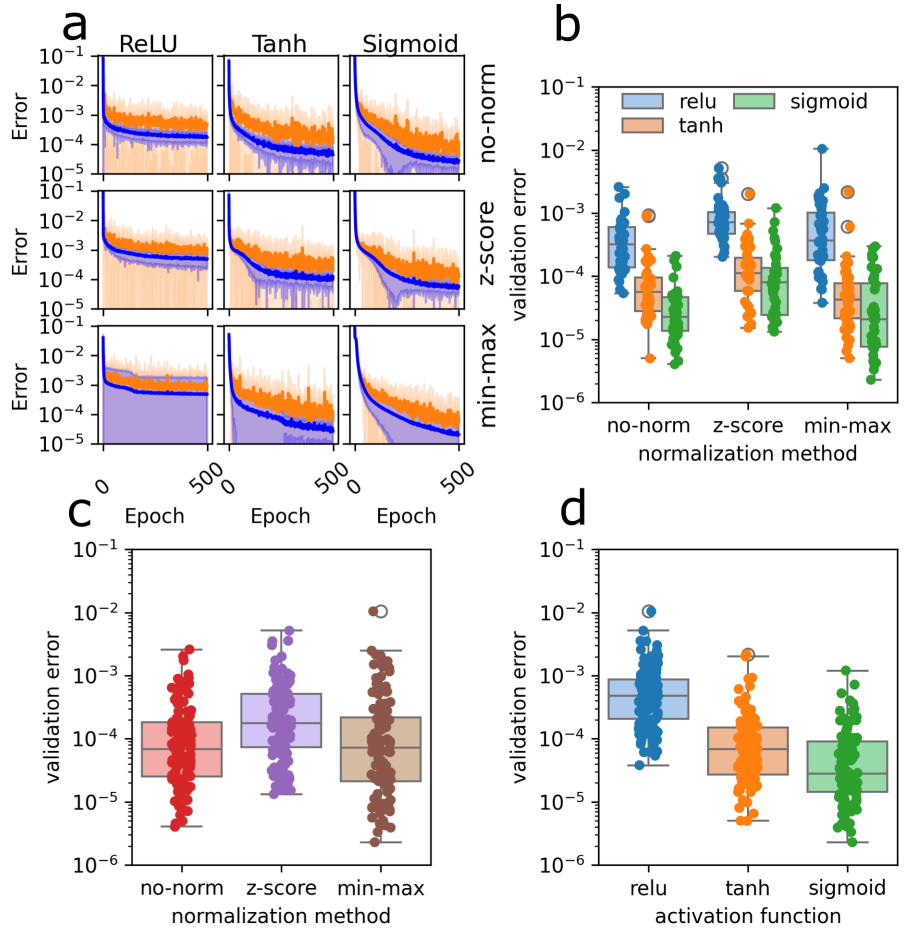


Figure 4.2: **Error while training for cubic nullcline with LR 0.01 and 500 epochs for** $\tau = 7.5$
**(a)** Training (blue) and validation (orange) error while training for different epochs, with standard deviation as colored region. **(b)** Validation error of the last epoch, separately for each normalization method and activation function. **(c)** Validation error of the last epoch, separately for each normalization method. **(d)** Validation error of the last epoch, separately for each activation function.

no-norm is present in this region. In the same way, we especially find the Sigmoid activation function, with some Tanh in this corner. A more qualitative understanding of each behavior is provided in Figure 4.4b. It illustrates the correlation for the combinations of normalization methods and activation functions, organized into a 3x3 grid. Each row represents a different normalization method, and each column represents a different activation function. When using z-score or no normalization, the produced neural networks exhibit negligible correlation (PCC $< 0.1$). However, for min-max normalization, the combination with the activation function ReLU yields the highest correlation (PCC $= 0.41$). The combination with Tanh results in (PCC $= 0.09$), which is the lowest in this row. A slightly higher value is achieved when combined with the Sigmoid activation function (PCC $= 0.13$). With a LR of 0.005 and 1000 epochs, depicted in Figure A.3, no norm still results in networks with uncorrelated errors. Z-score normalization combined with ReLU results in a moderate correlation (PCC $= 0.18$). When combined with Tanh the

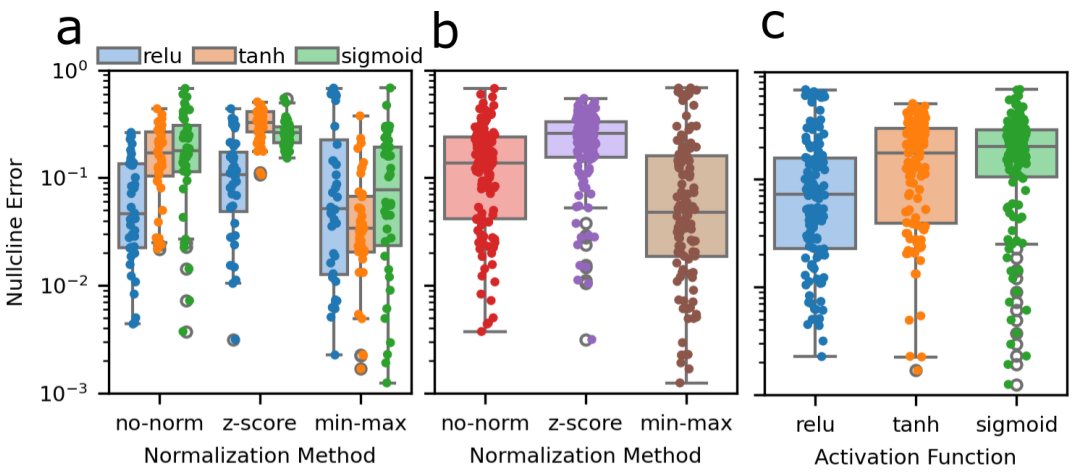


Figure 4.3: **Error of predicting the cubic nullcline with LR 0.01 and 500 epochs for** $\tau = 7.5$
**(a)** Nullcline error separately for each normalization and activation function. **(b)** Nullcline error of the last epoch, separately for each normalization method. **(c)** Nullcline error of the last epoch, separately for each activation function.

PCC is unrelated (PCC = 0.05). The combination with Sigmoid also shows a moderate correlation (PCC = 0.21). For the normalization method min-max, the combination with ReLU (PCC = 0.48) performs the best again, indicating excellent compatibility. The combination with Tanh (PCC = 0.23) and Sigmoid (PCC = 0.13) are both mediocre.

Overall, the combination of min-max normalization with the ReLU activation function demonstrated the highest PCC. The mediocre PCC values observed with the z-score normalization method will not be pursued due to the associated high nullcline errors. The lower nullcline error of min-max normalization allows us to consider configurations with Tanh or Sigmoid activation functions, despite their mediocre PCC values. However, the variability of PCC for Tanh across the two LRs renders it an unsuitable choice for further analysis.
In the above study, an LR of 0.01 with 500 epochs exhibited similar behavior to an LR of 0.005 with 1000 epochs. Both settings met the important convergence criteria for neural network training and demonstrated consistent patterns in validation and nullcline errors. Given the significantly faster computational time for an LR of 0.01 with 500 epochs, we will adopt this configuration for further experiments. Therefore, we will proceed with the min-max normalization method, combining it with the high-performing ReLU activation function and the moderately performing Sigmoid activation function.

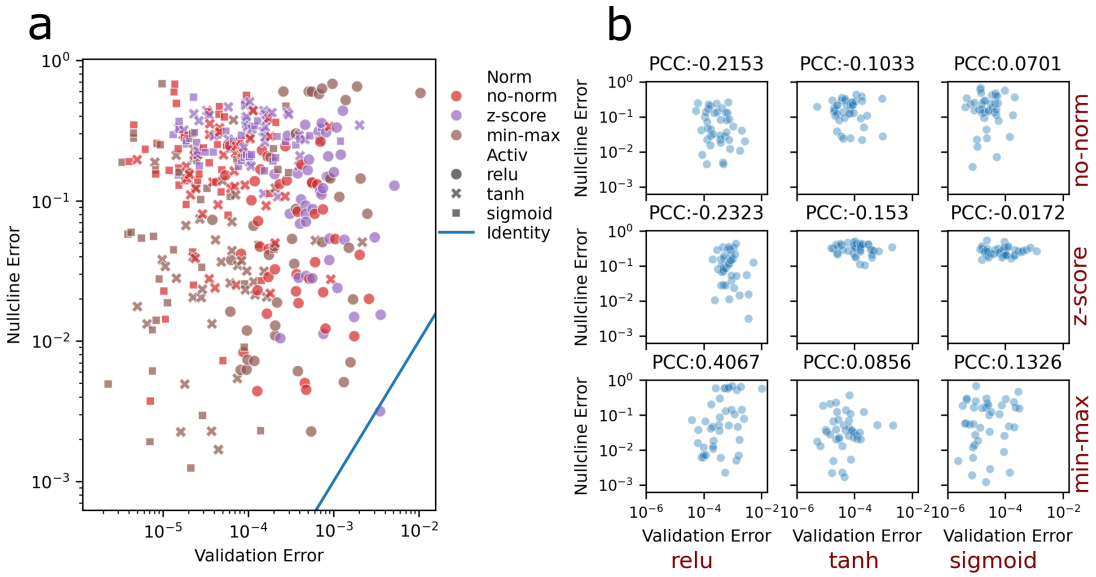


Figure 4.4: **Correlation between nullcline error and validation error for networks with LR 0.01 and 500 epochs for $\tau = 7.5$**
**(a)** Combined for all normalization methods (Norm) and activation functions (Activ), the identity line illustrates equal performance of nullcline prediction and time series prediction. Remarkably, a dot is below this line, this network's predictive performance is greater on the nullcline then on the time series. **(b)** Separately for each normalization method and activation function.

.

### 4.1.2 Strong Time-Scale Separation $\tau = 100$

We are interested in a robust neural network that performs well across all time-scales ($\tau$), if such a network exists. In this paragraph, we examine the validation error, nullcline error and PCC with a strong time-scale separation $\tau = 100$. Analysis is done similar to our previous analysis with $\tau = 7.5$.

#### Validation error

Figure A.4a provides the validation and training error during training. Consider first the neural networks using the ReLU activation function. The general training evolution of the mean is similar to the one with the same hyperparameters with $\tau = 7.5$ of Figure 4.2a. But on average, the error is slightly higher. The Tanh activation function converges earlier than for $\tau = 7.5$, illustrated by an early and deep decline of the error. Combined with the z-score normalization method this resulted in an earlier convergence with an 8 times greater validation error. The error of the Sigmoid activation function combined with no normalization method evolves similarly to $\tau = 7.5$. Although a higher variability is present. Combined with z-score normalization it results in an earlier convergence, resulting in a final error that is ten times higher than the first. Lastly, the combination of Sigmoid with min-max achieves an earlier convergence, but the final error is still equal to that of $\tau = 7.5$.
The final validation errors of training, depicted in Figure A.4b, illustrate other behavior than for $\tau = 7.5$ in Figure 4.2. The ranking of the activation functions stayed the same for no-norm, but for z-score normalization the error of Tanh and Sigmoid activation functions worsened to the level of the ReLU activation function. For min-max normalization the proportions stayed similar to that of $\tau = 7.5$. For the higher time-scale separation, z-score normalization has the highest error, and a smaller deviation as before, shown in Figure A.4c. Also, the error of the activation functions Tanh and Sigmoid increased by a factor of 1.5 and 3 respectively, demonstrated in part d of the figure.

#### Nullcline Error

Consider now the nullcline error of the normalization methods and activation functions depicted in Figure A.5a. First, consider the neural networks that are trained using no normalization. ReLU achieved a median nullcline error right below $10^{-2}$, Tanh achieved an error right below $10^{-1}$, and Sigmoid's performance increased exponentially, achieving an error of $2 \times 10^{-3}$. Contrarily, for the z-score normalization, the performance of Tanh and Sigmoid decreased significantly to the value of 1. ReLU achieved a lower nullcline error by a factor of 2.5. Lastly, for the min-max normalization method, all activation functions increased their predictive performance. ReLU and Tanh secure a nullcline error below $10^{-2}$, and Sigmoid records a remarkably low nullcline error of $3 \times 10^{-4}$, achieving the most minimal value yet. The general behavior of the normalization method is shown in Figure A.5b. The z-score normalization method performs the worst, with a mean around 1, corresponding to an inaccurate prediction. It is followed by no normalization method, and the min-max normalization method attains the lowest median of $3 \times 10^{-3}$. The activation functions have varying performance, the median of ReLU is $10^{-2}$, Tanh performs worse with a median of $9 \times 10^{-2}$, the lowest median corresponds to Sigmoid with

$2 \times 10^{-3}$. However, the variability is significant due to its improper performance for the z-score normalization.

#### PCC

The most quantitative measure of robustness of the neural networks, the PCC, is shown in Figure A.6. Figure A.6a illustrates the distribution for all the normalization methods and activation functions. The general behavior indicates that min-max normalization is on the side of lower validation and nullcline error, while z-score is located at higher error and no normalization is in between. Remarkably, the distribution of all these points together achieves a PCC of 0.64. Figure A.6b illustrates the PCCs for each combination of normalization method and activation function. Considering ReLU, the PCC increases from no-norm (PCC = 0.17), followed by z-score (PCC = 0.40), to min-max normalization (PCC = 0.52). The Tanh activation function combined with no normalization method achieves a PCC of 0.10. Combined with z-score it achieves a substandard PCC of $-0.07$ and with min-max a significant value of 0.35. Lastly, the Sigmoid activation function achieves an unrelated PCC for no normalization (PCC = $-0.02$). However, it attains moderate results combined with z-score (PCC = 0.26) and min-max (PCC = 0.22).

The validation errors indicate effective convergence for the different combinations of normalization methods and activation functions with no sign of overfitting. This time-scale separation $\tau = 100$ reaffirms the consistent robustness of min-max normalization with the ReLU (PCC = 0.52) and Sigmoid (PCC = 0.22) activation functions, achieving superior values. These results indicate that the configuration of hyperparameters significantly impacts the model's capabilities. Min-max normalization combined with the ReLU or Sigmoid activation function yields the best performance, suggesting a synergistic effect. Consequently, further investigation continues with configurations using these combinations, although the PCC of the Sigmoid activation function is not as large as that of the ReLU activation function, it compensates by having a notably low nullcline error.

### 4.1.3 Varying Network Size and Depth

As mentioned in Chapter 3, increasing the complexity by using larger neural networks does not necessarily lead to better performance, as it can result in overfitting. We now undertake a more comprehensive investigation into the performance across different neural network sizes. Employing the optimized normalization method and activation functions identified previously, namely min-max normalization combined with either the ReLU or Sigmoid activation function, we will assess networks with 2, 4, 8, or 16 hidden layers, each containing 4, 8, or 16 nodes. Each hidden layer will thus have the same amount of nodes.

#### Time-Scale Separation $\tau = 7.5$

#### Nullcline Error

Figure 4.5 demonstrates the box plots of the nullcline error for time-scale separation $\tau = 7.5$. There seems to be a general trend that neural networks with more layers have more difficulty predicting the nullcline structure. Let us first consider the network with

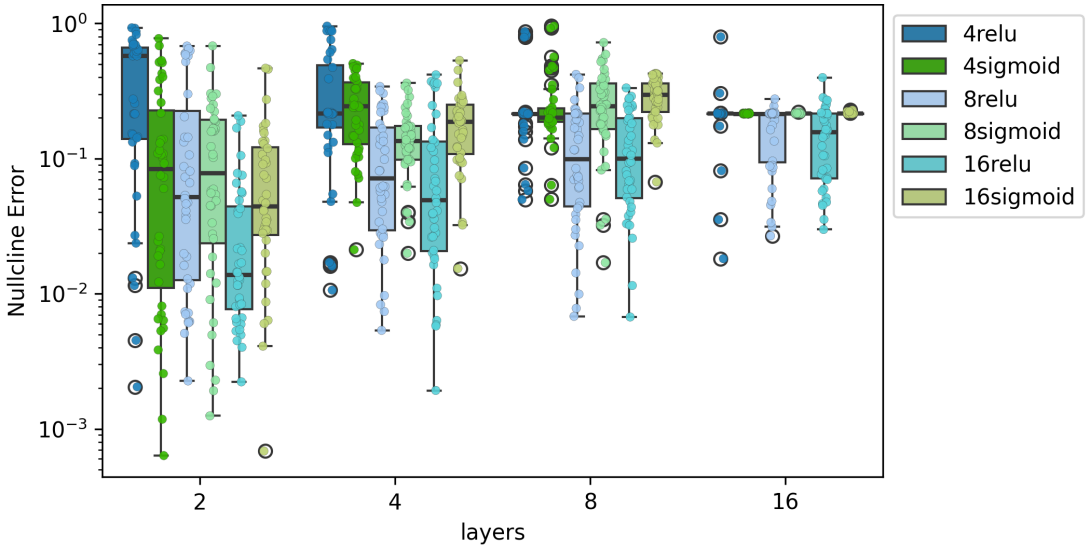


Figure 4.5: **Nullcline error for varying nodes and layers with LR 0.01 and 500 epochs for** $\tau = 7.5$
In the legend, the number before the activation function name indicates the corresponding amount of nodes. On the x-axis is the amount of layers, on the y-axis is the nullcline error. The ReLU activation function is shown in shades of blue, while the Sigmoid activation function is shown in shades of green. From left to right, each layer contains 4 nodes, 8 nodes, or 16 nodes.

16 layers. These networks do not manage to predict the nullcline, evidenced by their high nullcline errors. Notice how the activation function Sigmoid returns the same nullcline error, equal to 0.21 for every network, ReLU also predicts this nullcline often. This corresponds to the prediction of a horizontal nullcline, shown in Figure 4.6a. For networks with 8 layers, the combination with 4 nodes also results in frequent prediction of the horizontal nullcline. The Sigmoid activation function with 8 and 16 nodes does not manage to recover the precise form of the nullclines, endorsed by its high nullcline error. Meanwhile, activation function ReLU does manage a nullcline error of less than $10^{-2}$. The same holds for networks with 4 layers: for all the nodes, the activation function Sigmoid is not able to provide a network that finds the accurate form of the nullcline. Meanwhile, the activation ReLU achieves to find the accurate form for 8 and 16 nodes. The higher the number of nodes, the greater the decline in ReLU's nullcline error. The best-performing networks are those with 2 layers. Every network here manages a nullcline error of less than $10^{-2}$. Activation function ReLU combined with 4 nodes has the highest median. For 4 nodes, the boxplot of the activation function Sigmoid is similar to that of ReLU and Sigmoid with 8 nodes, and the Sigmoid with 16 nodes. The network that uses ReLU with 2 layers and 8 nodes does perform best, with a median around $10^{-2}$.

### PCC

The most robust neural network must consist of hyperparameters that also result in a reliable correlation between training error and nullcline error. In Table 4.1, the PCC values are presented for the different combinations of nodes, layers and activation functions. The

configurations that achieve a nullcline error below $10^{-2}$ are underlined and are examined more closely. For more detail, the distributions of each can be found in Figure A.8 in the appendix. In that figure, one also sees that the configurations that recurrently predict the horizontal nullcline, are also the configurations that did not perform well in training, with the identical validation error of 0.112. These networks might have been stuck in a local minima of the weights when training. This could explain their poor performance as these points also lie approximately at the same location. The PCC is not a representative measure of the quality of predictions for these plots, but their high nullcline error prevented their use anyway.

Table 4.1: $\tau = 7.5$: PCC for Different Network Architectures

| **Layers** | **Nodes** | **Activation Function** | |
|---|---|---|---|
| | | **ReLU** | **Sigmoid** |
| 2 | 4 | 0.24 | 0.28 |
| | 8 | 0.41 | 0.13 |
| | 16 | 0.46 | 0.22 |
| 4 | 4 | 0.26 | 0.54 |
| | 8 | 0.31 | -0.31 |
| | 16 | -0.03 | -0.13 |
| 8 | 4 | 0.16 | 0.23 |
| | 8 | 0.26 | 0.33 |
| | 16 | -0.16 | -0.22 |
| 16 | 4 | 0.56 | 0.16 |
| | 8 | 0.66 | 0.24 |
| | 16 | 0.27 | 0.38 |

Consider the neural network with 4 layers, the only values we proceed with are those with the activation function ReLU with 8 or 16 nodes. The latter has an uncorrelated PCC and will not be discussed further. Let us visualize the nullcline prediction of the former, which achieves a mediocre PCC of 0.31. The prediction and its error are shown in Figure 4.6c. Although it manages to recover the general behavior of the nullcline, the deviation is still significant. A similar observation applies to Figure 4.6d, the configuration corresponding to 2 hidden layers with 4 nodes each. For our purpose, we are interested in networks that produce more consistent predictions, ensuring greater reproducibility. This consistency can be observed for Figure 4.6e and 4.6f, corresponding to the configuration that uses ReLU, with 2 layers combined with 8 (PCC=0.41) and 16 (PCC=0.46) nodes respectively. These higher PCCs contribute to the smaller predictive deviation and reduced nullcline error. In contrast, the predictions for the Sigmoid activation function are all smaller, having PCC values 0.22, 0.13, and 0.28, in decreasing number of nodes.

### Time-Scale Separation $\tau = 100$

To gain a better understanding of the best-performing networks we will explore the effects of different amounts of nodes and layers using the same analysis, using a strong time-scale

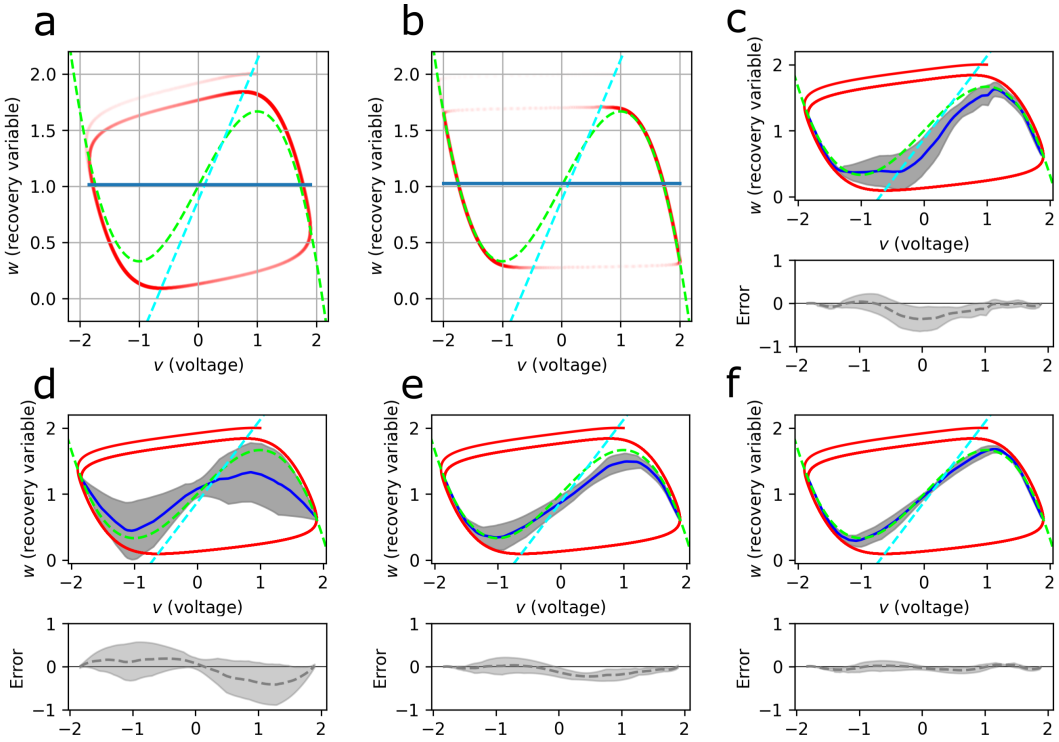


Figure 4.6: **Nullcline Predictions for varying nodes and layers**
In green the cubic nullcline $\dot{v} = 0$, in light blue the linear nullcline $\dot{w} = 0$, in red the trajectory. The mean predicted nullcline in dark blue and the standard deviation in grey. Each prediction depicts the average nullcline prediction of the five networks with the lowest validation error. **(a)** The constant cubic nullcline prediction for $\tau = 7.5$. **(b)** The constant cubic nullcline prediction for $\tau = 100$. **(c)** Configuration: $\tau = 7.5$, ReLU, min-max, 4 hidden layers each 8 nodes. Mean nullcline error is $3.16 \times 10^{-2}$. **(d)** Configuration: $\tau = 7.5$, ReLU, min-max, 2 hidden layers each 4 nodes. Mean nullcline error is $5.0 \times 10^{-2}$. **(e)** Configuration: $\tau = 7.5$, ReLU, min-max, 2 hidden layers each 8 nodes. Mean nullcline error is $1.4 \times 10^{-2}$. **(f)** Configuration: $\tau = 7.5$, ReLU, min-max, 4 hidden layers each 16 nodes. Mean nullcline error is $2 \times 10^{-3}$.

separation of $\tau = 100$. Figure A.7 illustrates the nullcline errors for different layers and nodes.

### Nullcline Error

Contrary to the case when $\tau = 7.5$, most models are able to recover the general structure of the nullcline, reaching a nullcline error below $10^{-2}$. The only networks unable to recover the nullcline are those with 16 layers using the Sigmoid activation function, and those with 8 layers and 16 nodes also using the Sigmoid activation function. For networks with 8 hidden layers, only the ReLU activation function with 8 nodes achieves a nullcline error of less than $10^{-3}$. Meanwhile, networks with 4 layers can only achieve such accuracy with 4 nodes. When using configurations with 2 layers, only the ReLU activation function combined with 4 nodes fails to achieve an accuracy of $10^{-3}$. The networks with 4 and 8 nodes using the Sigmoid activation function even attain accuracies up to $10^{-4}$, while

maintaining small variability.

### PCC

Consider the PCC values in Table 4.2 derived from Figure A.9. The underlined values

Table 4.2: $\tau = 100$: PCC for Different Network Architectures

| Layers | Nodes | Activation Function | |
|---|---|---|---|
| | | **ReLU** | **Sigmoid** |
| 2 | 4 | 0.85 | **0.22** |
| | 8 | 0.52 | **0.22** |
| | 16 | 0.14 | 0.33 |
| 4 | 4 | 0.75 | -0.03 |
| | 8 | 0.46 | 0.10 |
| | 16 | 0.04 | 0.18 |
| 8 | 4 | 0.60 | 0.56 |
| | 8 | 0.52 | 0.64 |
| | 16 | 0.07 | 0.47 |
| 16 | 4 | 0.56 | 0.23 |
| | 8 | 0.82 | 0.02 |
| | 16 | 0.87 | 0.29 |

achieve nullcline errors below $10^{-2}$, double underline below $10^{-3}$, and bold below $10^{-4}$.

Similar to $\tau = 7.5$, for larger neural networks the training is often unsuccessful, resulting in the same validation errors and identical nullcline predictions, as observed in Figure 4.6b. The PCCs of these models are not accurate. This issue also arises in the 4-layer network with 4 nodes using the ReLU activation function.

Analyzing the two-layer systems, which also performed the best for $\tau = 7.5$. We notice that again networks using the ReLU activation function achieve high PCCs, except for 16 nodes, where it performs moderately. Meanwhile, networks using 2 layers and the Sigmoid activation function always perform moderately but can predict the nullclines much more accurately.

The ReLU nullcline predictions are shown in Figures A.10a, A.10b, and A.10c, for respectively 4, 8 and 16 nodes. The prediction error of the configuration using ReLU, 2 layers each 4 nodes, shown in Figure A.10a achieves a notably low error ($1.42 \times 10^{-3}$) compared to same configuration with $\tau = 7.5$ ($5.00 \times 10^{-2}$). Figure A.10b illustrates the configuration for 8 nodes in each layer, where the nullcline structure is recovered very accurately, with an error of $7.99 \times 10^{-4}$. More than an order of magnitude smaller than for the same configuration for $\tau = 7.5$. Similar accuracies are achieved for the configuration with 16 nodes in each layer, its prediction shown in Figure A.10c, with error $2.06 \times 10^{-3}$. The deviation stayed consistent throughout the prediction range for the two smallest configurations. On the contrary, the deviation of the biggest network peaks in the middle.

Figures A.10d, A.10e, and A.10f illustrate the predictions of Sigmoid in an increasing

number of nodes. The accuracy increases for smaller nodes, reaching a minimal value of $2.35 \times 10^{-4}$.

There is a notable difference in performance for the medium time-scale separation ($\tau = 7.5$) and the strong time-scale separation ($\tau = 100$), with the nullcline errors being overall lower for the larger time-scale separation.
The analysis indicates that using neural network configurations with 2 hidden layers are the most robust. Although networks using the Sigmoid activation function achieve lower nullcline errors, their moderate PCC values make them less reliable for obtaining reproducible results. Using the reliability of the ReLU activation function we propose the following systematic methodology:

### 4.1.4 Benchmark

To establish a benchmark, we employ two-layered networks utilizing the min-max normalization and the ReLU activation function to recover the general structure of the nullcline. To enhance the reliability of the prediction, we select the five networks with the lowest validation error and average over their nullcline predictions. As demonstrated in Figure 4.7d, networks using the ReLU activation function tend to piecewise approximate the nullcline. Therefore, when more accuracy and detail are required, one can employ two-layered networks using the min-max normalization and the Sigmoid activation function. By fitting Sigmoid's predictions to the previously obtained average ReLU prediction, more precise nullclines can be recovered.

Let us first apply the defined methodology on $\tau = 7.5$. We have found that a 2-layered neural network with 16 nodes provides the most reliable results. Figure 4.7a portrays the real nullcline, the averaged nullcline, and the fitted nullcline. The error depicts clearly that the Sigmoid fit reduces the fluctuations present in the mean prediction. Furthermore, Figure 4.7c displays that by applying the mean, you reduce the error further than each of the five separate predictions. Indicating that the predictions slightly above and below the nullcline partially cancel out. The acquired nullcline error is $1.98 \times 10^{-3}$. Additionally, the Sigmoid fit reduces the nullcline error further to $6.85 \times 10^{-4}$. Remarkably, the network that fitted best to the mean is the one with the lowest nullcline error of all 40 configurations trained with Sigmoid. The same methodology can also be applied to the strong time-scale separation of $\tau = 100$, with similar results. The corresponding size of the neural network that is most reliable is 2 layers and 4 nodes. Analogously as with $\tau = 7.5$, Figure 4.7b illustrates the predictions in phase space. Also, here the Sigmoid fit reduces the fluctuations of the ReLU predictions. Figure 4.7c displays the benefit of averaging, the nullcline error is reduced significantly to a value of $1.42 \times 10^{-3}$. Fitting the network that uses the Sigmoid activation function achieves an outstanding nullcline error of $9.33 \times 10^{-5}$. The limitation of ReLU is that its predictions are a piecewise linear approximation, which does not ensure differentiability at endpoints. This property is important as nullclines are differentiable, additionally, it aids in inferring the differential equation [28]. The Sigmoid fit solves this problem as its predictions are smooth and thus differentiable. This improvement is visualized in Figure 4.7d.

The proposed systematic methodology proves an accurate and robust way of predicting the nullcline structure. Not only did the error decrease by approximately an order of magnitude, it also smoothened the prediction. This method performs better than other

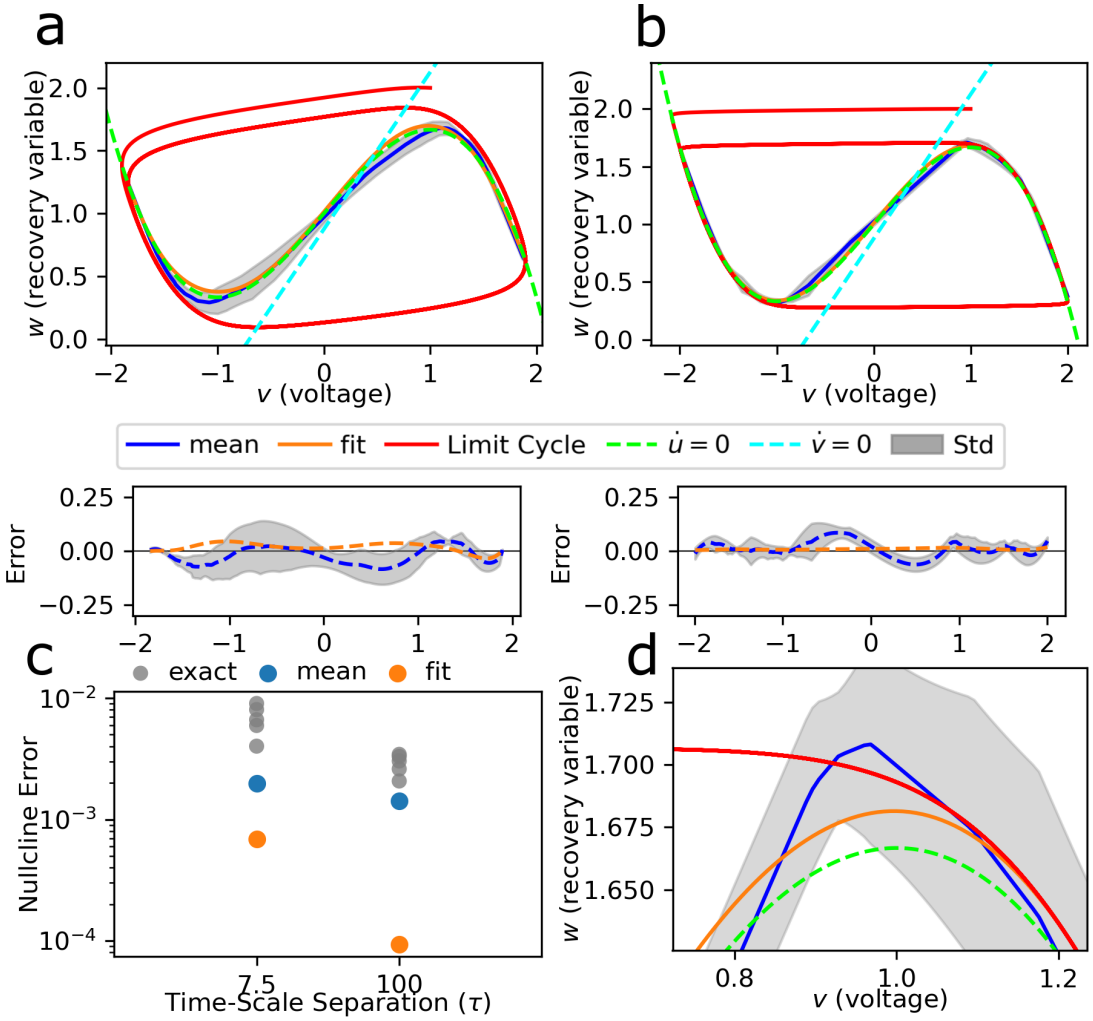


Figure 4.7: **Increasing accuracy by employing a benchmark.**
Cubic nullcline prediction: blue is the mean prediction of the five networks, trained using ReLU, with the lowest validation error. In grey is the standard deviation of these predictions. In orange is the Sigmoid prediction with the best fit to the mean prediction. **(a)** Phase space and error for system with $\tau = 7.5$. ReLU trained with 2 layers each 16 nodes. Mean nullcline error $1.98 \times 10^{-3}$. Fitted nullcline error $6.85 \times 10^{-4}$. **(b)** Phase space and error for system with $\tau = 100$. ReLU trained with 2 layers each 4 nodes. Mean nullcline error $1.42 \times 10^{-3}$. Fitted nullcline error $9.33 \times 10^{-5}$. **(c)** Spread of the errors of the five separate ReLU predictions, the mean of those predictions, and the Sigmoid fit to this mean. Each step of the proposed benchmark reduces the error significantly. **(d)** Zoomed version of Subfigure (b), ReLU tends to approximate the nullcline piecewise. By applying a Sigmoid fit the prediction is smoothened, ensuring differentiability.

less flexible methods mentioned in Section 3.2, where the Laplace method managed an error of $8.5 \times 10^{-3}$ for $\tau = 7.5$. The linear interpolator does achieve slightly better results: $4.6 \times 10^{-4}$, but this is only because the three-dimensional phase space is linearly correlated in $\dot{v}$, as explained earlier in Section 4.5.

The results still do not provide a consensus if we should use 4, 8 or 16 nodes. The

analysis of $\tau = 7.5$ suggests more nodes, since the PCC increases with higher nodes, the opposite is true for $\tau = 100$. The next section will serve to gain more insight into the performance of the networks when exploring different values of time-scale separations.

## 4.2 Exploring Time-Scale Separations

To explore the effects of time-scale separation, we will employ neural networks with 2 layers, varying the number of nodes between 4, 8, and 16. It has been established that min-max normalization is the most reliable preprocessing step when combined with the ReLU and Sigmoid activation functions. In the previous section, we defined a systematic methodology using ReLU to uncover the nullcline structure. We will further explore this benchmark by applying these networks to weak, medium, and strong time-scale separations. The following time-scale separation ($\tau$) values will be used: 1, 2, 5, 7.5, 10, 20, 40, 60, 80, and 100.

Figure 2.5 illustrates oscillations in the time series and phase space for different values of $\tau$. Notice the gradual transition from sinusoidal-like waves to more pronounced relaxation oscillations. The extent to which the cubic nullcline is contained within the limit cycle depends on the shape of the limit cycle. Specifically, lower values of $\tau$ result in less of the cubic nullcline being contained, as $\tau$ increases, progressively more parts of the cubic nullcline are contained within the limit cycle. Since it is only possible to predict the nullcline in the region contained in the limit cycle, variations in the shape of the limit cycle also influence the nullcline prediction.

An analysis of the PCC values and nullcline errors across different $\tau$ values is presented visually in Figure 4.8a. When using strong time-scale separations $\tau > 40$, neural networks with fewer nodes perform more reliably. Networks with 4 nodes achieve a PCC between 0.66 and 0.85, increasing for higher $\tau$. In contrast, neural networks with 8 nodes attain a PCC between 0.29 and 0.59, while larger networks with 16 nodes have a PCC ranging from 0.02 to 0.14. Analyzing the nullcline errors for these networks reveals that the configurations with higher PCCs result in lower nullcline error medians. A more detailed overview is provided in Figure 4.8b. In this regard, we conclude that the low PCC at 16 nodes might be because this configuration tends to yield neural networks of which the majority trains well. The resulting low spread in validation error but greater spread in nullcline error lowers the PCC. The networks with fewer nodes do not always achieve low training error, but when they do, the nullcline prediction is also more accurate. This might indicate that when the smaller networks train well, they predict simple nullclines. Meanwhile, the networks with larger nodes that train well, might overcomplicate the nullcline prediction, due to the more complex nature of larger networks. Larger standard deviations usually accompany this. For this reason, the smaller nodes are preferred at strong time-scale separations, as a higher PCC provides more reliable and reproducible results.

For time-scale separations $\tau$ smaller than 40, the system exhibits a significant change in behavior. Analysis of Figure 4.8a reveals that for $\tau = 10$ and $\tau = 20$ the networks with 8 nodes are most reliable, evidenced by the PCC. Conversely, more inconsistent behavior is observed for $\tau$ values of 2, 5 and 7.5 that interchangeably prefer 4 and 16 nodes. Notably, at $\tau = 5$, both the PCC and nullcline errors are the worst yet. Meanwhile, all the

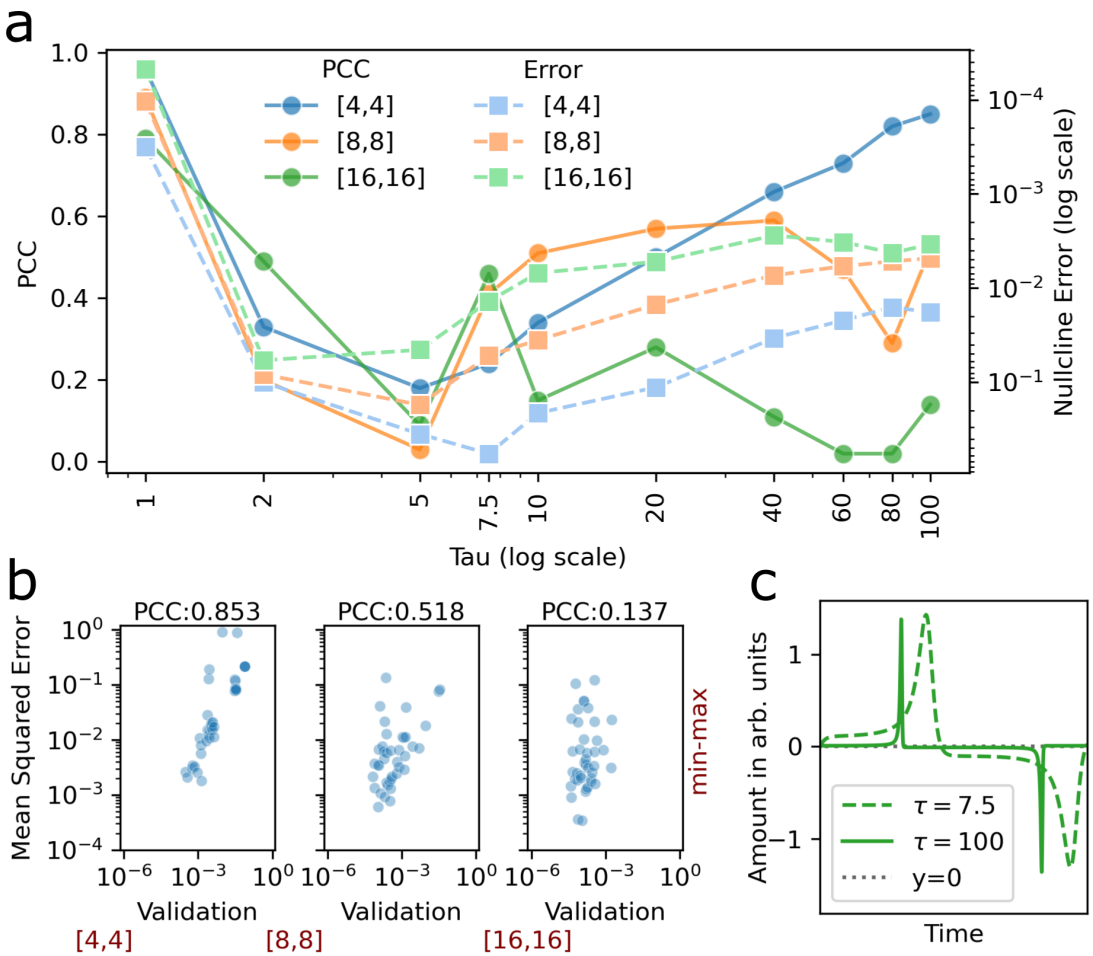


Figure 4.8: **Analysis of ReLU mean performance for different time-scale separation $\tau$.**
**(a)** PCC (left axis) and nullcline error median of the 40 neural networks (right axis) in function of the time-scale separation $\tau$. The $\tau$ axis is in log scale to highlight the differences for low $\tau$. **(b)** PCC plots for $\tau = 100$, networks are trained with ReLU. The [x,x] signifies 2 layers each with x nodes. **(c)** The derivative $\dot{v}$ for $\tau = 7.5$ and $\tau = 100$, zoomed on one oscillation.

networks perform exceptionally for the very weak time-scale separation of $\tau = 1$.

This phenomenon can be explained using Figure 2.5a, depicting the nullclines and limit cycle for $\tau = 1$. Here, the boundary of the limit cycle simplifies the cubic nullcline, reducing it to a quasi-linear form. The high PCC and low nullcline error suggest that neural networks have reduced difficulty predicting quasi-linear curves. This behavior will also emerge in Section 4.3, where we predict the linear nullcline.

Strong time-scale separation $\tau \geq 40$ performs well for another reason. When analyzing the derivative $\dot{v} = 0$ in time, we notice that it is close to zero, increasingly close for higher $\tau$ values, as depicted in Figure 4.8c. When predicting the nullcline, the neural networks evaluate the dynamics for $\dot{v} = 0$. Given that $\dot{v}$ of the time series is close to zero might facilitate networks' learning of the nullcline structure, particularly the outer edges

of the cubic nullcline. The alignment of the nullcline and limit cycle at the outer edges is especially clear in Figure 2.5c.

The $\tau$ values like 2, 5, and 7.5, do not benefit from the advantages associated with $\tau = 1$, nor do they benefit from those of the strong time-scale separation. Instead, the cubic form of the nullcline already emerged and $\dot{v}$ is usually not close to zero, as visualized in Figure 4.8c. The low performance of the neural networks indicates that this hinders extraction of the nullcline structure.
This suggests that neural networks that can handle more complexity may be necessary to accurately determine the nullcline structure, which would explain why the more intricate neural networks with 16 nodes achieve the highest PCCs and lowest nullcline error medians for $\tau = 2$ and $\tau = 7.5$. The low PCC and high nullcline error medians of the networks at $\tau = 5$ may indicate that the explored neural network configurations are not appropriate for this case. Instead, the use of larger networks capable of handling greater complexity may be more appropriate.

## 4.3 Linear Nullcline

So far we have examined the cubic nullcline, it is the most complex nullcline and requires precise optimization. In contrast, in this section we will discuss the linear nullcline. Its linear simplicity will allow for less constrained hyperparameter optimization and more accurate predictions. The linear nullcline emerges when $\dot{w} = 0$ in Eq. (2.7).
For the neural network hyperparameter optimization, we employ a configuration containing two hidden layers each containing eight nodes and an LR of 0.01. Given the rapid convergence of the validation error, 100 epochs proved sufficient for attaining reliable results.

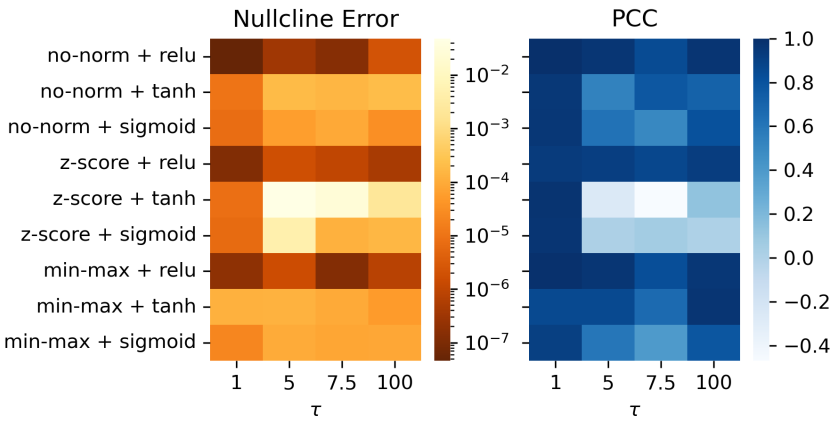


Figure 4.9: **Performance analysis hyperparameter optimization for linear nullcline** Evaluation of Median Nullcline Error of the 40 neural networks and PCC for neural networks trained with different combinations of normalization methods and activation functions, for systems with different time-scale separations $\tau$.

The results for different combinations of normalization methods, activation functions and time scale separations are shown in Figure 4.9. The results highlight that the ReLU

activation function performs best. It consistently achieves a nullcline error below $10^{-5}$, regardless of the normalization method used. Additionally, ReLU performs exceptionally in terms of the PCC, with values consistently above 0.8. Even though using no normalization yields very accurate results, it is still recommended to apply a normalization method, as discussed in Section 3.3. In this system, the values are already close to zero, which might explain the effectiveness of no normalization. However, this might not be the case for other systems.

A possible explanation for ReLU's strong performance might be its structure, despite being inherently non-linear, ReLU includes a linear component (for $x > 0$), which might allow it to capture the linear parts more effectively.

When ReLU is not employed the configurations only manage accurate predictions and high PCCs at weak time-scale separations ($\tau = 1$). Especially z-score normalization fails to recover the nullcline for medium to strong time-scale separations.

The linear nullcline demonstrates that the proposed benchmark of the previous section should be applied with care. The proposed method of fitting the neural network trained with Sigmoid over the ReLU predictions are employed to smoothen the linearlike piecewise fit of the ReLU predictions. Since the prediction here is one uninterrupted linear curve, no additional smoothing is needed.

## 4.4 Fixed Point Prediction and Symmetric Nullclines

### Fixed Point Prediction

Thus far, we have optimized the neural network to extract the nullcline structure from the time series. In Chapter 2, we introduced another fundamental structure of the dynamical systems, namely fixed points, which are located at the intersections of two nullclines. In this section, we will demonstrate the prediction of the fixed point for $\tau = 7.5$ using the proposed systematic methodology.

The phase space, including the limit cycle, nullclines, and both real and predicted fixed points, is depicted in Figure 4.10a. For the cubic nullcline we apply the proposed benchmark for this analysis. This consists of selecting five neural networks with the lowest validation error, with two hidden layers and 16 nodes, employing the ReLU activation function. Further, we analyzed the mean prediction of these five networks and a network using Sigmoid with the same architecture, selected for its optimal fit to the mean ReLU prediction. In Section 4.2, we deduced that these configurations performed best for this time-scale separation. The same neural network configuration was used to recover the linear nullcline.

Analysis of the figure reveals that all predicted fixed points align with the linear nullcline. This alignment is attributed to the high accuracy of the networks predicting the linear nullcline, evidenced by nullcline errors below $10^{-9}$. The only significant variability is that in the prediction of the cubic nullcline.

The five individual predictions exhibit considerable variability across the linear nullcline. The mean prediction is generally in the same region as the actual fixed point, but there remains a noticeable gap between them. However, the fixed point prediction is more precise when fitted with Sigmoid, shortening the distance to the fixed point by 60%. This supports the systematic methodology by demonstrating that additional detail can be

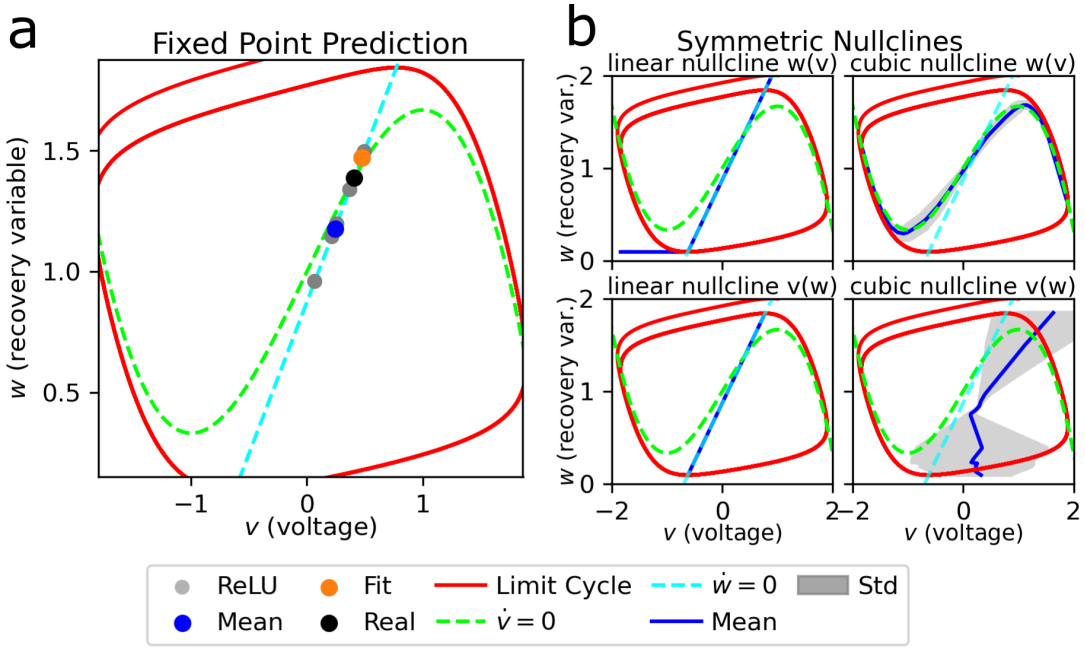


Figure 4.10: **Analysis on Prediction of Fixed Points and Symmetric Nullclines** **(a)** Fixed point prediction for $\tau = 7.5$ in phase space employing the proposed benchmark. The real fixed point (black) of the system is first predicted separately using five neural networks using ReLU, each with 2 layers, each 16 nodes (grey). The mean of these predictions is shown in blue and the Sigmoid fit to these predictions is shown in orange. **(b)** The results of predicting a nullcline for $\tau = 7.5$ in two symmetric directions. The $w(v)$ stands for the variable $w$ predicted from $v$, $v(w)$ is the opposite. The mean prediction (blue) uses the same configuration as (a). The standard deviation of these predictions is shown in grey.

achieved by fitting a neural network using the Sigmoid function to the benchmark network using ReLU. It is noteworthy that combining the linear nullclines with nullcline errors below $10^{-9}$ and the neural networks using the Sigmoid activation function, which has an error of $6.85 \times 10^{-4}$, still results in a relatively large distance of 0.11 between the predicted and real fixed point. This discrepancy arises because, locally around the fixed points, the slopes of the cubic nullcline and linear nullcline are close in value. Consequently, a slight displacement of either nullcline can cause the fixed point to shift excessively.

## Symmetric nullclines

In Section 3.1, we discussed that we have a choice of direction when training the neural network. However, the cubic nullcline can only be predicted as $w$ in function of $v$ because the inverse of the cubic nullcline is not well-defined. Conversely, the linear nullcline can be predicted in both directions due to its one-to-one correspondence.

In an experimental setup, we do not have prior knowledge of the form of the nullclines, and therefore we can not know if a functional form exists in the direction in which we train the neural network. Figure 4.10b compares these cases. The top left and bottom left of the figure illustrate that the linear nullcline was recovered in both directions. For predicting $w$ out of $v$, we do see that outside the limit cycle the prediction for low $v$

fails but stays right for high $v$. We expected it to fail, as neural networks have declining extrapolating properties. It might have remained accurate at the top due to the nearby transient, which might have provided enough local information about the phase space for the neural network. The top right picture has been analyzed intensively in the previous sections. The bottom right figure demonstrates the results when one attempts to predict the nullcline in a direction where no functional form exists. Predicting the inverse cubic nullcline leads to significant inaccuracies and high deviations. Thus, an unexpectedly high error in the prediction of the nullcline may indicate the absence of a functional form in that direction.

Although, it is generally recommended to predict the nullcline $\dot{x}$ in function of $x$. Here, this would mean to predict the cubic nullcline $\dot{v} = 0$ as $w$ in function of $v$, and the linear nullcline $\dot{w} = 0$ as $v$ in function of $w$, representing respectively the top right and bottom left cases in Figure 4.10b. For these cases, the domain of the nullcline within the limit cycle exactly spans the limit cycle. This occurs because the turning points of the limit cycle, going from decreasing in $x$ to increasing in $x$, define the points where $\dot{x} = 0$. This method avoids extrapolation as in the top left of Figure 4.10b. However, if no functional form exists, this approach becomes unfeasible. In such cases, a change of variables in the phase space may be necessary.

## 4.5 Insights into Effectiveness of ReLU Activation Function

In Section 4.1.4 we proposed to use the ReLU activation function as a benchmark for future work. When optimizing the hyperparameters for the cubic nullcline ReLU indicated high performance and robustness, and also for the linear nullcline it proved to be the most suitable. Although it is difficult to fully understand a neural network due to its black box nature, this section will attempt to provide a possible reason for ReLU's reliability.

In Section 3.2, we already introduced the idea of extending the two-dimensional phase space into an extra dimension. This structure enables us to understand what the neural network is learning, how it is able to predict the nullclines, and why ReLU performs so well. Let us consider the cubic nullcline, with the corresponding function $H(v, 0)$, as introduced in Section 3.1. By analyzing the differential equation, Eq. (2.7), our objective is to develop an intuitive understanding that will enable the extraction of the nullcline structure from the time series data.

### Three-Dimensional Phase Space Analysis

As mentioned earlier, in the three-dimensional phase space, the inputs $v$ and $\dot{v}$ of the cubic nullcline function, $H(v, \dot{v})$, are placed on the $x$- and $y$-axis. The output $w$ is put on the $z$-axis.

Consider the following algorithm for extracting the nullcline: Interested in the value of the nullcline, choose arbitrarily a value $v_c$ between the limits of the time series. There exist two points $(v_c, w_1, \dot{v}_1)$ and $(v_c, w_2, \dot{v}_2)$ in the time series for this value of $v$, with corresponding distinct $\dot{v}$ values, one being positive $\dot{v}_1$ and the other being negative $\dot{v}_2$.

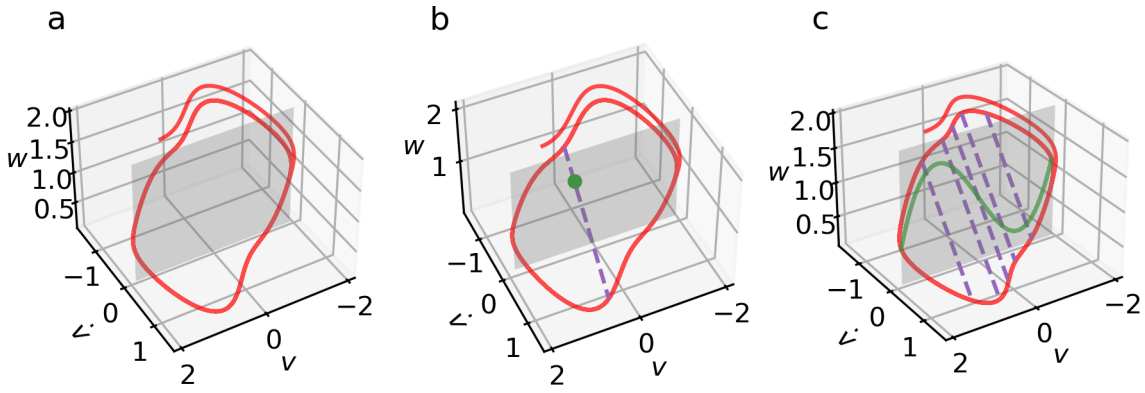


Figure 4.11: **Three-dimensional extension of phase space for cubic nullcline.** **(a)** Time series data is plotted in red. In grey is the surface defined by the plane $\dot{v} = 0$ **(b)** In addition to (a), a line connecting two points with identical $v$ values is shown in dashed purple. At the intersection with the prior mentioned plane is a green dot. **(c)** In addition to (b), more purple lines are shown for varying $v$ values. The cubic nullcline arising from the intersections is shown in green.

Using the differential Eq. (2.7), we fill in $v = v_c$, and find

$$w = -\dot{v} + v_c - \frac{v^3}{3} + z \tag{4.1}$$

a negative linear function with slope $-1$ between the variables $w$ and $\dot{v}$. The above equation connects the points $(v_c, w_1, \dot{v}_1)$ and $(v_c, w_2, \dot{v}_2)$ in a linear fashion. The intersection of this linear function with the plane $\dot{v} = 0$ is the corresponding nullcline value $H(v_c, 0)$, depicted in Figure 4.11b.
Subsequently, to determine the nullcline for every value of $v$ in the limits of the limit cycle, one must repeat the process for every value of $v$. One finds the full cubic form, illustrated in Figure 4.11c.

### Limitations

So an alternative and intuitive way to retrieve the nullcline from the data is to connect the two points in the limit cycle that have the same value of $v$ using a linear function. The corresponding intersection with the plane defined by $\dot{v} = 0$ forms the cubic nullcline. The same method can be applied to identify the linear nullcline, which is also established by a linear relation between the time series data.
This linearity is dependent on the dynamical system and is only known because we extracted it from the differential equation. Other dynamical systems can have other relations, and in an experimental setup, this is usually not known.
As an example, consider the study of hysteresis in a biological oscillator introduced in section 2.2. We apply the same algorithm as above in $x$, $\dot{x}$ and $y$ space. The nullcline lies in the plane defined by $\dot{x} = 0$. The analogous relation, as Eq (4.1), can be written as:

$$y = \sqrt[4]{\frac{1}{\dot{x} + x} - 1} \tag{4.2}$$

This function is nonlinear in $\dot{x}$.

## 4.6 Robustness of Predictive Model: Effect of Time Series Resolution

Up until now, the time series consisted of 15000 points, corresponding to 2500 points per period of oscillation. In many fields, more so in biology, the analysis using time series data presents challenges due to the limited amount of data points or low resolution. Although high-resolution data is valuable, they are less common [28]. In this part, we discuss the robustness of the proposed model identification for changing temporal resolution.

For this simulation, we continue with the optimized hyperparameters for $\tau = 100$ of Section 4.2, which provided the most accurate prediction yet. Those hyperparameters are min-max normalization, with ReLU, LR of 0.01, 500 epochs, with 2 layers each 4 nodes.

The performance of the model is assessed by evaluating the validation error during training, the mean nullcline error for the five networks with the lowest validation errors, and the PCC as shown in Figure 4.12. Analyzing the errors reveals a general tendency where both errors increase as the resolution decreases. There are two distinct trends present, one in the region with 100-1000 points and another in the region of 1000-15000 points, their difference is visualized by the linear interpolated curve. The slopes in the region of low-resolution are approximately twice as large as in the high-resolution region. This suggests that in the low-points regime, the error is more sensitive to reductions in the number of points. This sensitivity is particularly pronounced in the nullcline error, which exhibits a higher slope. The least amount of data points required for a nullcline error less than $10^{-2}$ is 900, corresponding to 150 data points per oscillation. On the contrary, the PCC remains stable up to 700 points, where it reaches the value of 0.8 before diminishing rapidly.

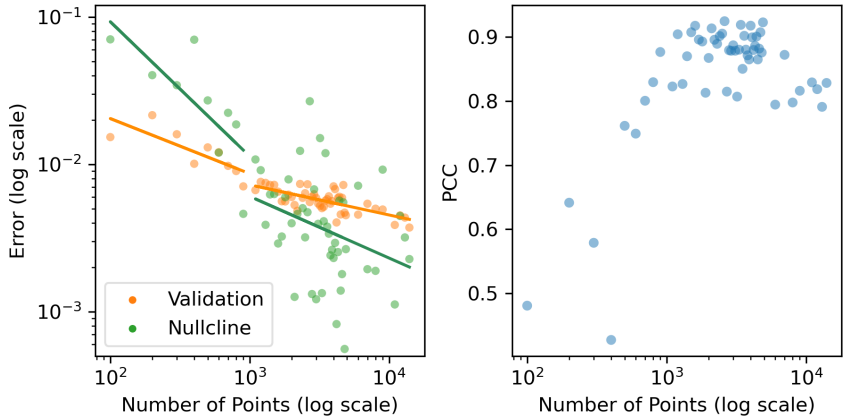


Figure 4.12: **Analysis temporal robustness**
Performance of neural network with varying resolution. The changing slope of the linear fit in the region between 100-1000 points and 1000-15000 points highlights the differing behavior. In the region with fewer points (100-1000 points), the linear fit indicates a more pronounced increase in the error as the number of points decreases.

To understand why model identification fails below 900 data points for six oscillations, consider the inputs $v$ and $\dot{v}$ and the output $w$ of the neural network. The neural network is trained using $v$ and $\dot{v}$ as input, with $w$ as output. The time series of $v$ and $w$ for a finite amount of points is a subset of the continuous $v$ and $w$ values. Meanwhile, the value of the

derivative $\dot{v}$ is dependent on the resolution of the time series of $v$, as depicted in Figure 4.13 for varying data points. The derivatives stay accurate in the region of slow-changing dynamics because the time series change is small and most data points are located there. On the contrary, in the region of fast-changing dynamics, the variables change quickly and there are not many data points describing the change, decreasing the accuracy of the derivative. For fewer points, the derivative moves away from the original peak.

In theoretical studies it might be more productive to use uniform sampling of the fast and slow-changing regions, although this is hard to achieve in experimental measurements where data is usually sampled uniformly in time. Another possible solution to acquire more data in the fast-dynamic region is to use long transients, which also uncover more phase space and provide more information about the potential landscape. Additionally, neural networks allow the study of multiple trajectories simultaneously. However, experimentally generating such transients is challenging.

The neural network's strong performance with 150 points per period is promising, as other methods like SINDy struggle to accurately identify models at this time-scale separation and resolution [28].

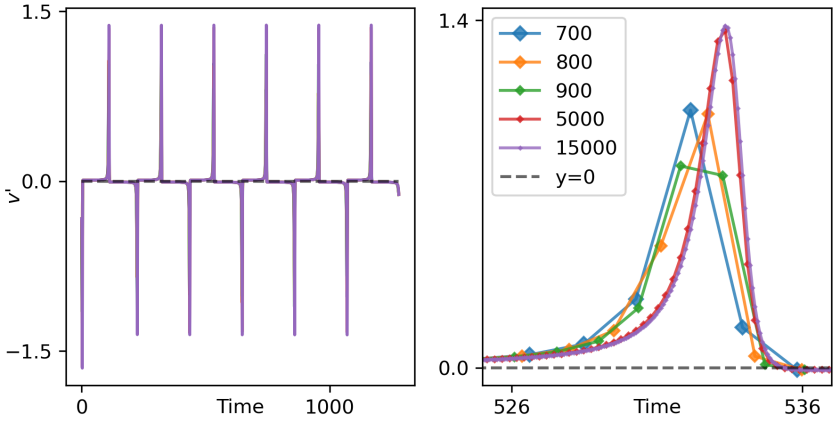


Figure 4.13: **Analysis of derivative** Left: Derivative of $v$ for six oscillations. Right: Zoomed in on the peak of the left figure, the curves are for different amounts of points. The curves for 5000 and 15000 points resemble each other, meanwhile for less than 1000 points the curves move away significantly from the high-resolution curve.

## 4.7 Other Systems

### Bicubic Model

Until now we have only tested and optimized the model identification algorithm on the FHN model. To broaden the applicability of this algorithm its performance on other models is examined. Let us consider the more complex bicubic model, introduced in Section 2.2 [25]. The governing equations of the dynamics are given by Eq. (2.12). This model produces a time series, illustrated in Figure 4.14a, that cannot be visually discerned from that of the FHN model. Notice that in three-dimensional phase space, the variables $\dot{u}$ and $v$, and $\dot{v}$ and $u$ are also linearly correlated, providing the same intuition as discussed in Section 4.5.

As the time series for the cubic model resembles the time series of the FHN model for medium time-scale separation, we choose the same optimized hyperparameters, namely two hidden layers with each 16 nodes. As model identification algorithm we utilize the introduced benchmark approach of Section 4.1.4.

Consider first the nullcline of $\dot{u} = 0$. There is a low linear dependence between the validation and nullcline error (PCC=0.04) for ReLU, but for Sigmoid it is higher (PCC=0.30). The mean average prediction of the five networks with the lowest validation error, the Sigmoid fit and the standard deviation, are shown in Figure 4.14b. The mean nullcline error is $2.68 \times 10^{-2}$, and is almost halved by the Sigmoid fit to $1.39 \times 10^{-2}$. The accuracy closely resembles the mean error ($2.35 \times 10^{-2}$) for the FHN model at $\tau = 2$. Both predict the general shape of the nullcline but fail to capture the finer details. A possible explanation, introduced in Section 4.2, is that the cubic nature of the nullcline is not fully apparent within the limit cycle. This subtlety could complicate the neural network's ability to recognize that the slight bend marks the start of a turning point. However, this set of hyperparameters does have the potential to perform more accurately, as one of the 40 networks achieved a nullcline error of $4.5 \times 10^{-3}$.

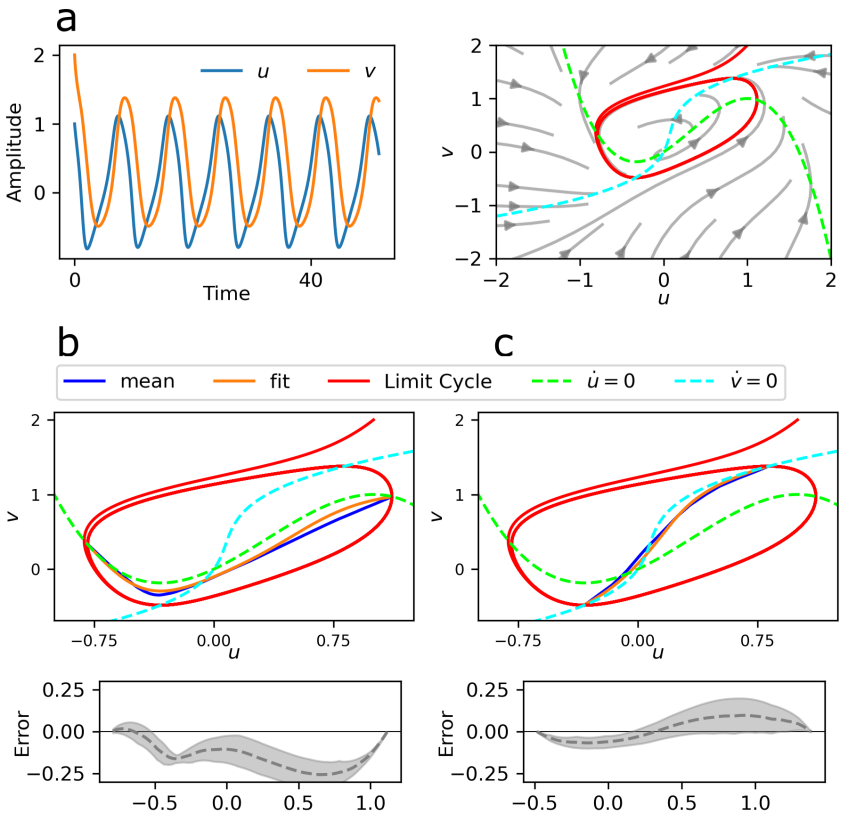


Figure 4.14: **Bicubic model**
**(a)** The time series and corresponding phase space. **(b)** The predicted $\dot{u} = 0$ nullcline. In blue: the mean prediction of the five neural networks with the lowest validation error, trained with ReLU. In orange: the prediction of Sigmoid that was fitted on the blue curve. In grey are the standard deviations of the ReLU predictions. **(c)** Similar to (b) but with $\dot{v} = 0$.

For the other nullcline $\dot{v} = 0$, the PCC is greater for ReLU (0.23) and less for Sigmoid (0.15). The mean prediction has a nullcline error of $4.00 \times 10^{-3}$ and is improved by Sig-

moid to $2.76 \times 10^{-3}$. The prediction is visualized in Figure 4.14c. The simpler shape of this nullcline could explain the increased accuracy compared to the $\dot{u}$ nullcline.

The introduced benchmark method has again shown to be advantageous by incorporating a Sigmoid fit, which reduces the error and increases the smoothness of the nullclines. Although the algorithm proves its ability to predict more complex nullclines, its robustness and applicability to other models remain an open issue.

# Chapter 5

# Conclusions

In this thesis, we addressed the problem of extracting hidden information from time series data of complex dynamical systems. Specifically, we focused on creating a robust machine-learning algorithm that recovers nullcline structures in the phase space. In order to do this, we selected the FitzHugh-Nagumo model, which has been used to study a wide range of physical, chemical and biological oscillatory systems. We focused on optimizing the neural network's hyperparameters for nullcline extraction, utilizing evaluation metrics such as validation error, nullcline error and the Pearson correlation coefficient. These metrics assess neural network generalizability, accuracy in nullcline recovery, and the correlation between them, which is important when this method is applied to systems with unknown governing equations. Based on these, we proposed a benchmark that uses min-max normalization, a two-layered neural network, with a 0.01 learning rate and 500 epochs using the ReLU activation function, alongside added precision from the Sigmoid activation function. Additionally, we identified a possible reason why ReLU performed most reliably. We demonstrated that the proposed algorithm was able to successfully uncover the form of both nullclines of the FHN model. Furthermore, we revealed that, if the nullcline's inverse exists, this could be performed regardless of the chosen variables.

We also investigated the performance of the method towards different time-scale separations $\tau$ in the dynamics. For strong time-scale separation, small neural networks could recover the nullclines with high accuracy. Meanwhile, larger neural networks managed an approximate form for weak time-scale separations; An exception is $\tau = 1$, the cubic nullcline was recovered accurately because it reduced to a quasi-linear form within the limit cycle.

In the case of strong time-scale separation, the algorithm was additionally tested for robustness in changing temporal resolution. We conclude that the algorithm struggles under 150 points per period, as the accuracy of the calculated derivatives degraded significantly. This stresses the need for high-resolution data, which is lacking in many fields, especially biology [28]. Instead, the method could be improved by correcting derivative errors, potentially by using a neural network for derivative prediction. Similarly, experimental data often contains noise, therefore we propose further investigation into its effects on the method's robustness [59].

We successfully applied our method to more complex systems such as the bicubic model. We were able to identify at least one of the nullclines successfully with high accuracy. This shows that the method can be applied to a wider range of dynamical systems which share the same features, e.g. comparable time-scale separation or same

scales in state variables. The next steps in improving the method is to apply it to other synthetic and experimental oscillatory systems, which exhibit different dynamical behaviors and characteristics (e.g. different scales in state variables, different nonlinearity, non-symmetric nullclines, multiple fixed points and more than two dimensions). An example is the biological oscillator of the Novak and Tyson model [24]. Preliminary results indicate a one-sided bias owing to the added nonlinearity. Additionally, to improve accuracy, especially under weak time-scale separation, exploring alternative architectures and hyperparameters could further enhance the algorithm's robustness.

## CODE AND DATA AVAILABILITY STATEMENT

The code, as well as the data that support the findings of this study, are available in the GitHub repository https://github.com/JimmyBillen/Master-Thesis/.

# Appendix A

# Extra Figures

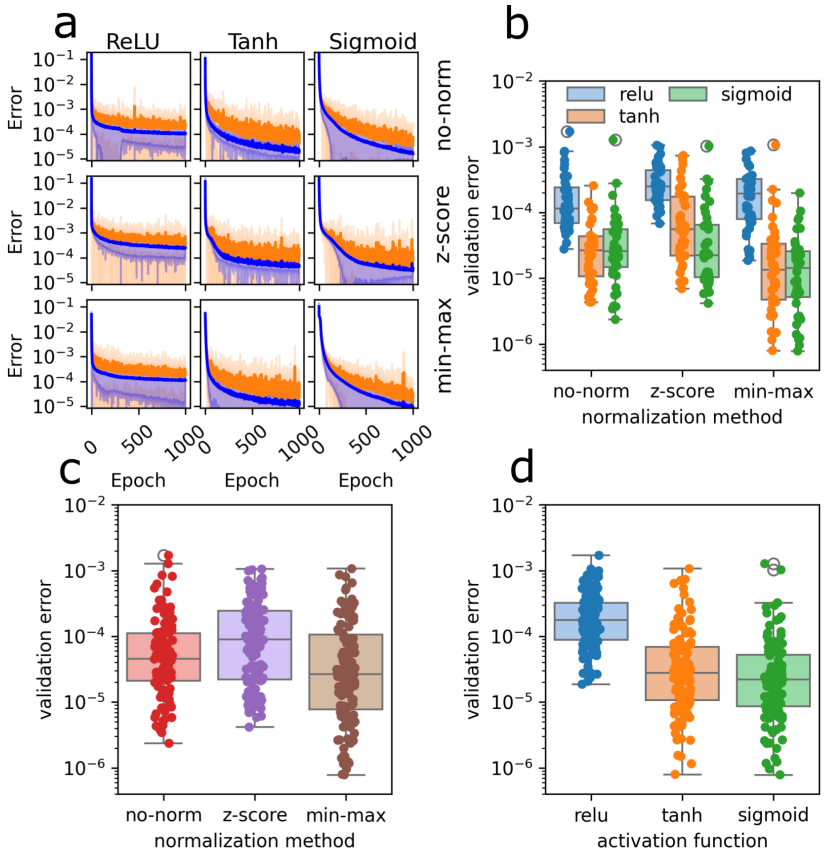


Figure A.1: **Error while training for cubic nullcline with LR 0.005 and 1000 epochs for** $\tau = 7.5$
**(a)** Training (blue) and validation (orange) error while training for different epochs, with standard deviation as colored region. **(b)** Validation error of the last epoch, separately for each normalization method and activation function. **(c)** Validation error of the last epoch, separately for each normalization method. **(d)** Validation error of the last epoch, separately for each activation function.

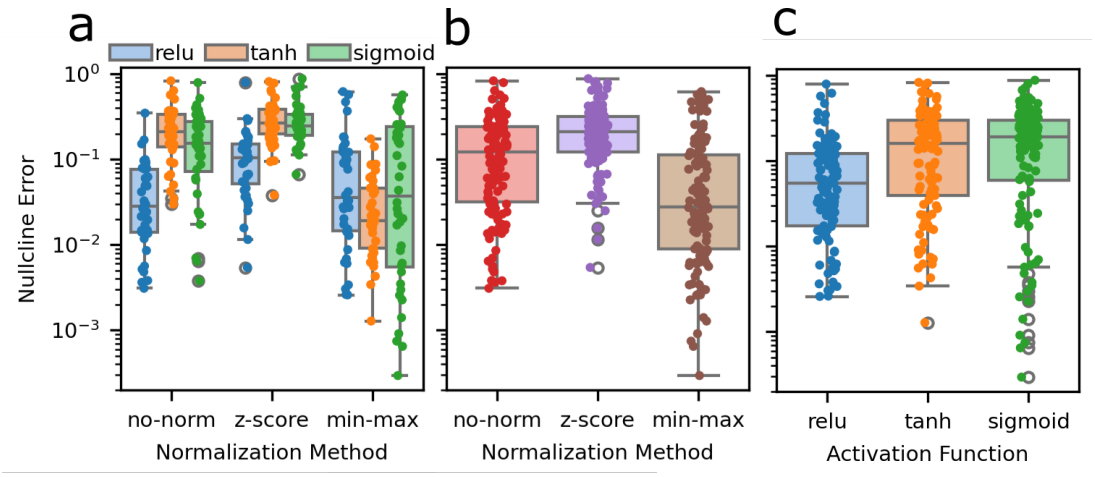


Figure A.2: **Error of predicting the cubic nullcline with LR 0.005 and 1000 epochs for $\tau = 7.5$**
**(a)** Nullcline error separately for each normalization and activation function. **(b)** Nullcline error of the last epoch, separately for each normalization method. **(c)** Nullcline error of the last epoch, separately for each activation function.

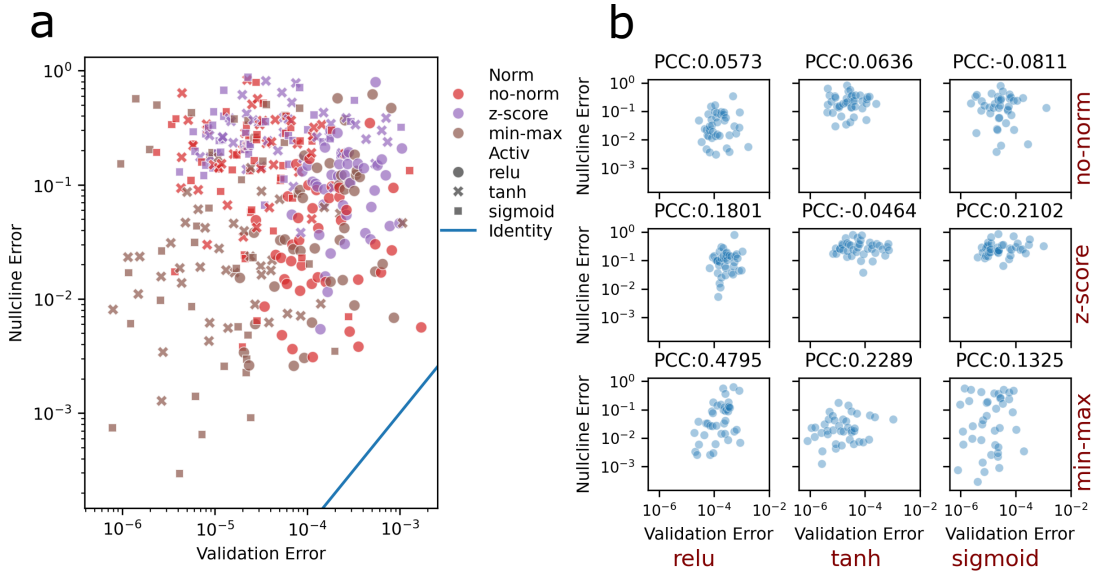


Figure A.3: **Correlation between nullcline error and validation error for networks with LR 0.005 and 1000 epochs**
**(a)** Combined for all normalization methods (Norm) and activation functions (Activ), the identity line illustrates equal performance of nullcline prediction and time series prediction. Remarkably, a dot is below this line, this network's predictive performance is greater on the nullcline then on the time series. **(b)** Separately for each normalization method and activation function.

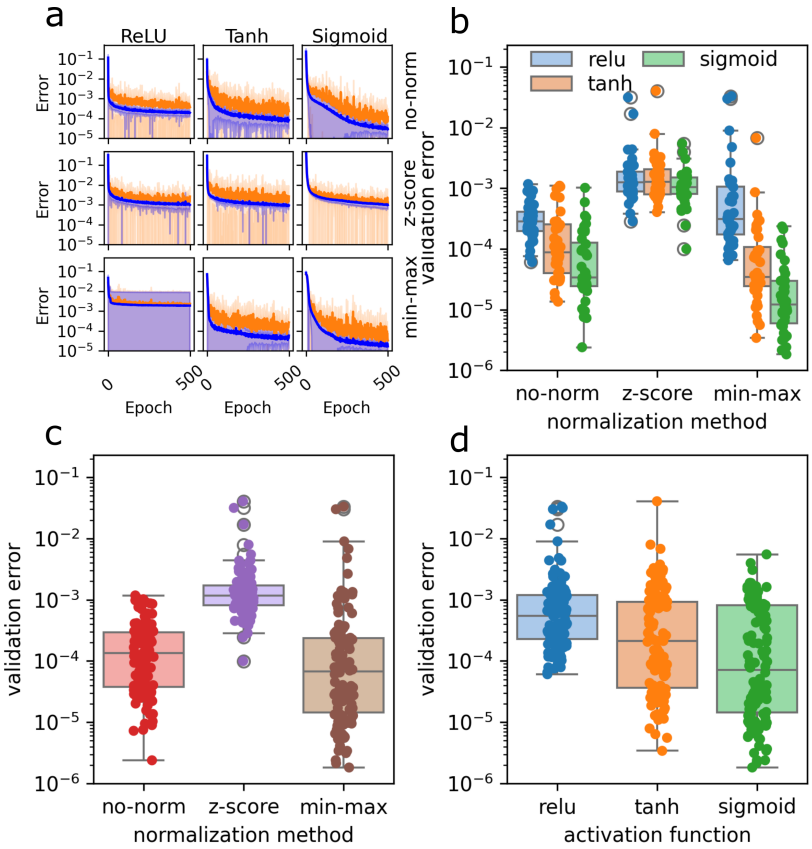


Figure A.4: **Error while training for cubic nullcline with LR 0.01 and 500 epochs for** $\tau = 100$
**(a)** Training (blue) and validation (orange) error while training for different epochs, with standard deviation as colored region. **(b)** Validation error of the last epoch, separately for each normalization method and activation function. **(c)** Validation error of the last epoch, separately for each normalization method. **(d)** Validation error of the last epoch, separately for each activation function.

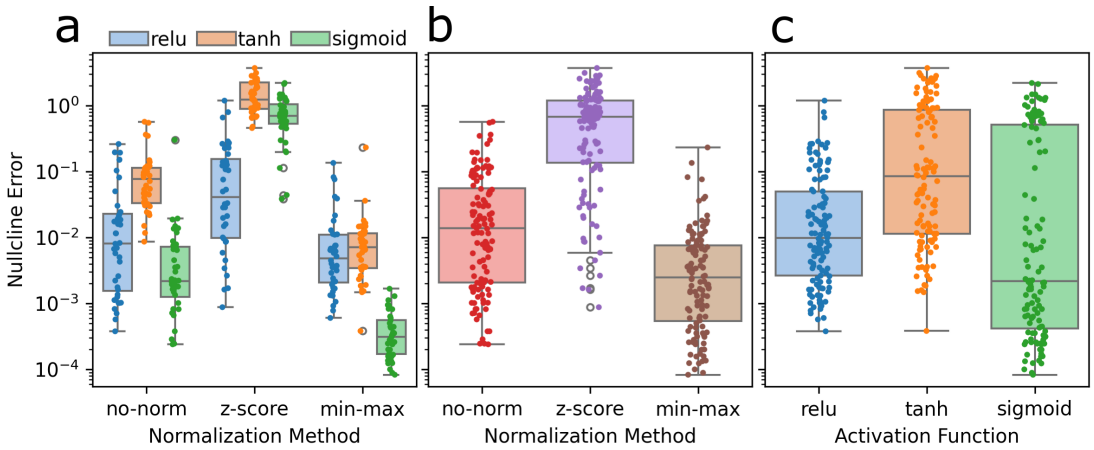


Figure A.5: **Error of predicting the cubic nullcline with lr 0.01 and 500 epochs for $\tau = 100$**
**(a)** Nullcline error separately for each normalization and activation function. **(b)** Nullcline error of the last epoch, separately for each normalization method. **(c)** Nullcline error of the last epoch, separately for each activation function.

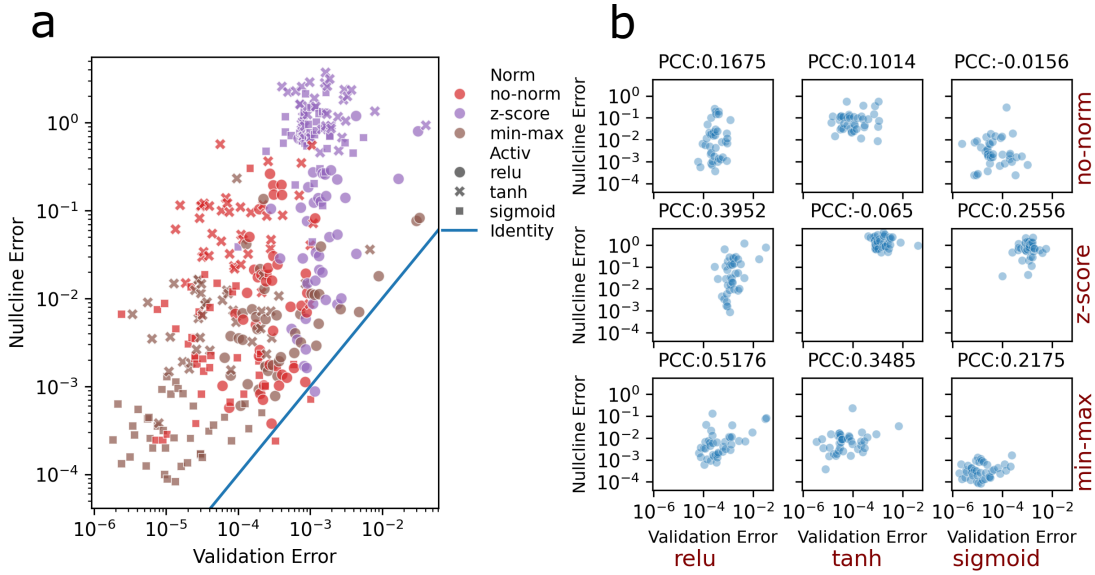


Figure A.6: **Correlation between nullcline error and validation error for networks with LR 0.01 and 500 epochs for $\tau = 100$**
**(a)** Combined for all normalization methods (Norm) and activation functions (Activ), the identity line illustrates equal performance of nullcline prediction and time series prediction. Remarkably, a few dots are below this line, this network's predictive performance is greater on the nullcline then on the time series. **(b)** Separately for each normalization method and activation function.

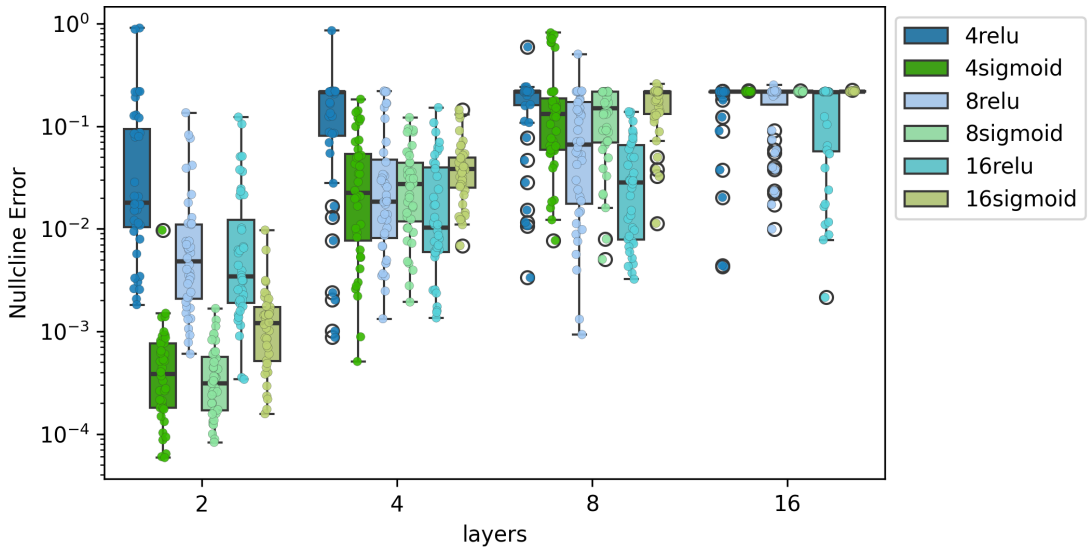


Figure A.7: **Nullcline error for varying nodes and layers with LR 0.01 and 500 epochs for $\tau = 100$**
In the legend, the number before the activation function name indicates the corresponding amount of nodes. On the x-axis is the amount of layers, on the y-axis is the nullcline error. The ReLU activation function is shown in shades of blue, while the Sigmoid activation function is shown in shades of green. For each layer, from left to right, 4 nodes, 8 nodes, and 16 nodes.

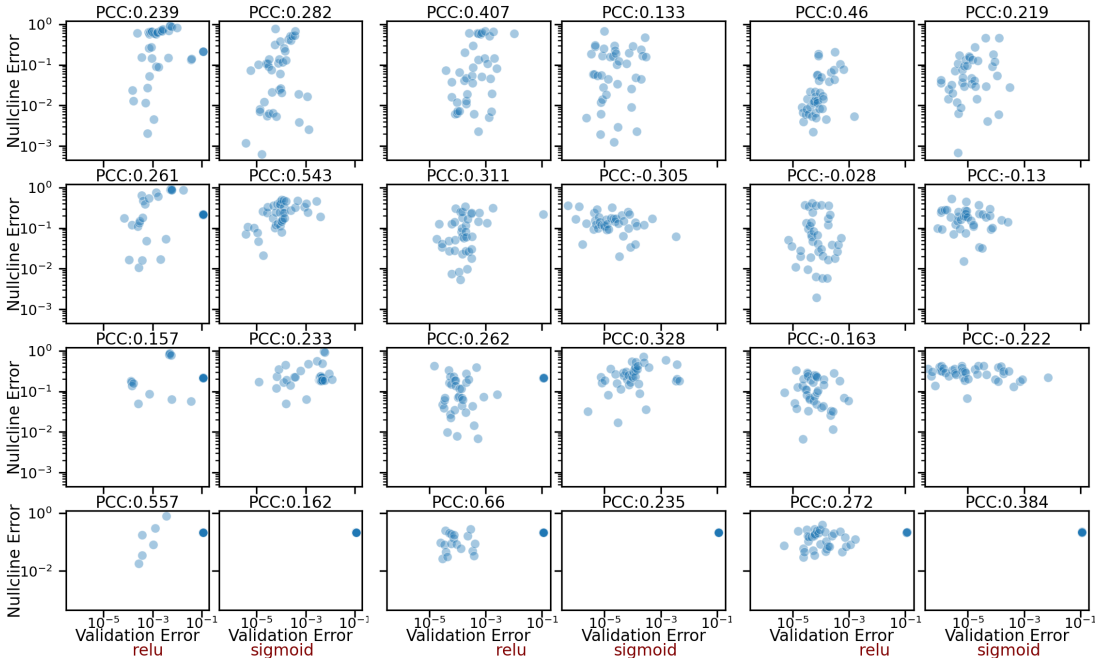


Figure A.8: **PCCs for varying nodes and layers with LR 0.01 and 500 epochs, min-max normalization combined with ReLU and Sigmoid activation function for $\tau = 7.5$**
The top row employs 2 layers, followed by 4 layers, 8 layers and 16 layers. The most left column employs 4 nodes, the middle column employs 8 nodes and the right column employs 16 nodes.

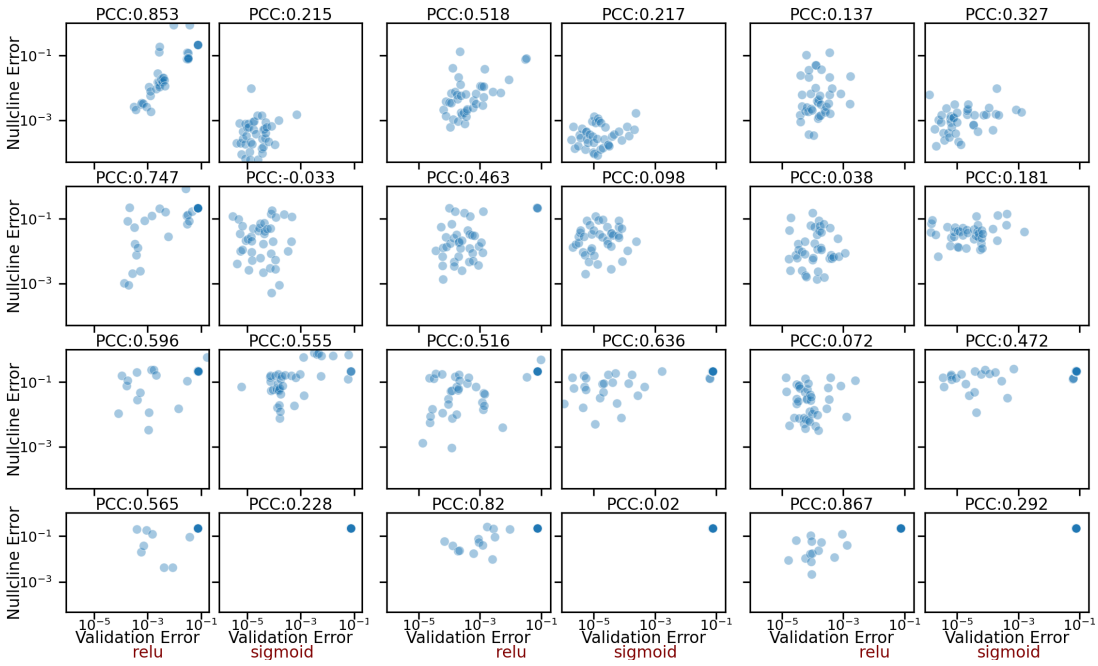


Figure A.9: **PCCs for varying nodes and layers with lr 0.01 and 500 epochs, min-max normalization combined with ReLU and Sigmoid activation function for** $\tau = 100$

The top row employs 2 layers, followed by 4 layers, 8 layers and 16 layers. The most left column employs 4 nodes, the middle column employs 8 nodes and the right column employs 16 nodes.

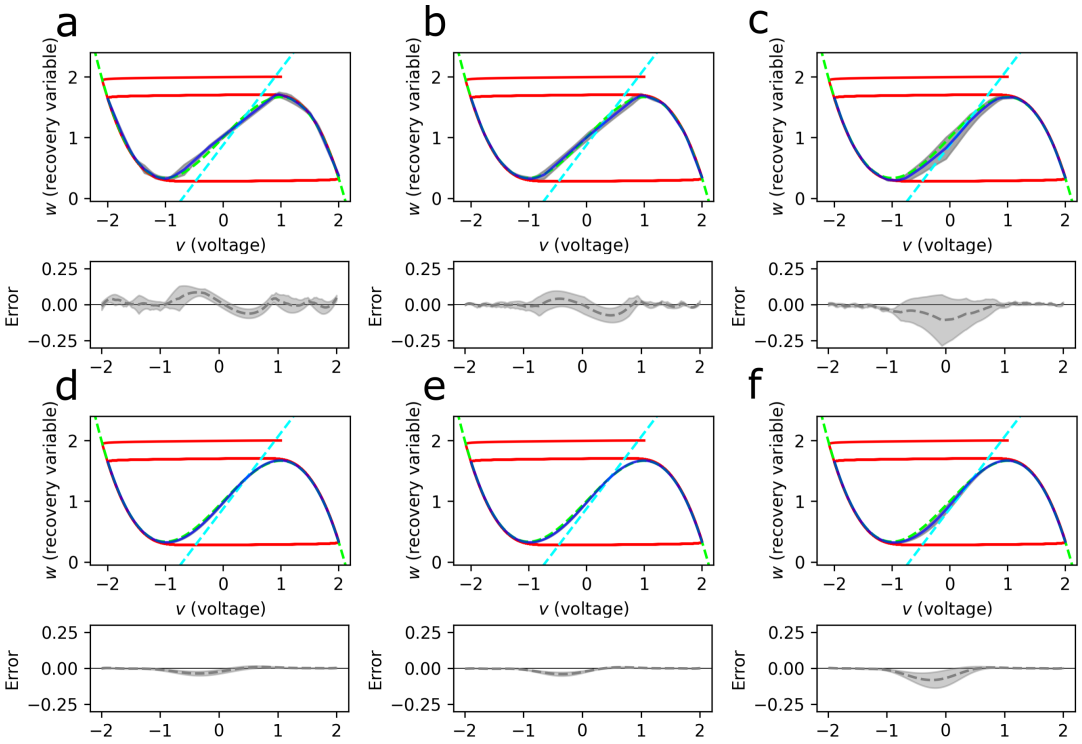


Figure A.10: **Cubic nullcline predictions for varying nodes combined with 2 layers for** $\tau = 7.5$

In green, the cubic nullcline $\dot{v} = 0$, in light blue, the linear nullcline $\dot{w} = 0$, in red, the trajectory. The mean predicted nullcline in dark blue and the standard deviation in grey. Each prediction depicts the average nullcline prediction of the five networks with the lowest validation error. **(a)** Configuration: ReLU, min-max, 2 hidden layers each 4 nodes. Mean nullcline error is $1.42 \times 10^{-3}$. **(b)** Configuration: ReLU, min-max, 2 hidden layers each 8 nodes. Mean nullcline error is $7.99 \times 10^{-4}$. **(c)** Configuration: ReLU, min-max, 2 hidden layers each 16 nodes. Mean nullcline error is $2.06 \times 10^{-3}$. **(d)** Configuration: Sigmoid, min-max, 2 hidden layers each 4 nodes. Mean nullcline error is $2.35 \times 10^{-4}$. **(e)** Configuration: Sigmoid, min-max, 2 hidden layers each 8 nodes. Mean nullcline error is $2.77 \times 10^{-4}$. **(f)** Configuration: Sigmoid, min-max, 2 hidden layers each 16 nodes. Mean nullcline error is $1.13 \times 10^{-3}$.

# Bibliography


[1] Joshua S. North, Christopher K. Wikle, and Erin M. Schliep. "A Review of Data-Driven Discovery for Dynamic Systems". en. In: *International Statistical Review* 91.3 (Dec. 2023), pp. 464–492. ISSN: 0306-7734, 1751-5823. DOI: 10.1111/insr.12554.

[2] Christof Eck, Harald Garcke, and Peter Knabner. *Mathematical Modeling*. en. Springer Undergraduate Mathematics Series. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-55160-9 978-3-319-55161-6. DOI: 10.1007/978-3-319-55161-6.

[3] Michael Brin and Garrett Stuck. *Introduction to Dynamical Systems*. 1st ed. Cambridge University Press, Oct. 2002. ISBN: 978-0-521-80841-5 978-0-511-75531-6 978-1-107-53894-8. DOI: 10.1017/CBO9780511755316.

[4] A. Murari et al. "Data driven theory for knowledge discovery in the exact sciences with applications to thermonuclear fusion". en. In: *Scientific Reports* 10.1 (Nov. 2020), p. 19858. ISSN: 2045-2322. DOI: 10.1038/s41598-020-76826-4.

[5] Seref Sagiroglu and Duygu Sinanc. "Big data: A review". en. In: *2013 International Conference on Collaboration Technologies and Systems (CTS)*. San Diego, CA, USA: IEEE, May 2013, pp. 42–47. ISBN: 978-1-4673-6404-1 978-1-4673-6403-4 978-1-4673-6402-7. DOI: 10.1109/CTS.2013.6567202.

[6] Jeff Dean, David Patterson, and Cliff Young. "A New Golden Age in Computer Architecture: Empowering the Machine-Learning Revolution". In: *IEEE Micro* 38.2 (Mar. 2018), pp. 21–29. ISSN: 0272-1732, 1937-4143. DOI: 10.1109/MM.2018.112130030.

[7] Yu Zhang et al. "A Survey on Neural Network Interpretability". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 5.5 (Oct. 2021), pp. 726–742. ISSN: 2471-285X. DOI: 10.1109/TETCI.2021.3100641.

[8] Timo Freiesleben and Thomas Grote. "Beyond generalization: a theory of robustness in machine learning". en. In: *Synthese* 202.4 (Sept. 2023), p. 109. ISSN: 1573-0964. DOI: 10.1007/s11229-023-04334-9.

[9] Huan Xu and Shie Mannor. "Robustness and generalization". en. In: *Machine Learning* 86.3 (Mar. 2012), pp. 391–423. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-011-5268-1.

[10] Luís A. Nunes Amaral. "Artificial intelligence needs a scientific method-driven reset". en. In: *Nature Physics* 20.4 (Apr. 2024), pp. 523–524. ISSN: 1745-2473, 1745-2481. DOI: 10.1038/s41567-024-02403-5.

[11] Nongnuch Artrith et al. "Best practices in machine learning for chemistry". en. In: *Nature Chemistry* 13.6 (June 2021), pp. 505–508. ISSN: 1755-4330, 1755-4349. DOI: 10.1038/s41557-021-00716-z.

[12] Steven H. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. en. Second edition. OCLC: ocn842877119. Boulder, CO: Westview Press, a member of the Perseus Books Group, 2015. ISBN: 978-0-8133-4910-7.

[13] Yuri A. Kuznetsov. "Introduction to Dynamical Systems". In: *Elements of Applied Bifurcation Theory*. Ed. by S. S. Antman, J. E. Marsden, and L. Sirovich. Vol. 112. Series Title: Applied Mathematical Sciences. New York, NY: Springer New York, 2004, pp. 1–37. ISBN: 978-1-4419-1951-9 978-1-4757-3978-7. DOI: 10.1007/978-1-4757-3978-7_1.

[14] Morris W. Hirsch, Stephen Smale, and Robert L. Devaney. *Differential equations, dynamical systems, and an introduction to chaos*. eng. 3. ed. Amsterdam Heidelberg: Elsevier, Academic Press, 2013. ISBN: 978-0-12-382010-5.

[15] Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. 1st ed. Cambridge University Press, May 1998. ISBN: 978-0-521-62365-0 978-0-521-62570-8 978-1-139-17317-9. DOI: 10.1017/CBO9781139173179.

[16] A. L. Hodgkin and A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". en. In: *The Journal of Physiology* 117.4 (Aug. 1952), pp. 500–544. ISSN: 0022-3751, 1469-7793. DOI: 10.1113/jphysiol.1952.sp004764.

[17] NobelPrize.org. *The Nobel Prize in Physiology or Medicine 1963*. Mon. 27 May 2024. Nobel Prize Outreach AB, 2024.

[18] Christof J. Schwiening. "A brief historical perspective: Hodgkin and Huxley". en. In: *The Journal of Physiology* 590.11 (June 2012), pp. 2571–2575. ISSN: 0022-3751, 1469-7793. DOI: 10.1113/jphysiol.2012.230458.

[19] J D Murray. "Mathematical Biology: I. An Introduction, Third Edition". en. In: *Interdisciplinary Applied Mathematics* ().

[20] Richard FitzHugh. "Impulses and Physiological States in Theoretical Models of Nerve Membrane". en. In: *Biophysical Journal* 1.6 (July 1961), pp. 445–466. ISSN: 00063495. DOI: 10.1016/S0006-3495(61)86902-6.

[21] J. Nagumo, S. Arimoto, and S. Yoshizawa. "An Active Pulse Transmission Line Simulating Nerve Axon". In: *Proceedings of the IRE* 50.10 (Oct. 1962), pp. 2061–2070. ISSN: 0096-8390. DOI: 10.1109/JRPROC.1962.288235.

[22] Daniel Cebrián-Lacasa et al. *Six decades of the FitzHugh-Nagumo model: A guide through its spatio-temporal dynamics and influence across disciplines*. en. arXiv:2404.11403 [math-ph, physics:nlin, physics:physics, q-bio]. Apr. 2024.

[23] DeLiang Wang. "Relaxation Oscillators and Networks". en. In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. Ed. by John G. Webster. 1st ed. Wiley, Dec. 1999. ISBN: 978-0-471-34608-1. DOI: 10.1002/047134608X.W2282.

[24] Béla Novák and John J. Tyson. "Design principles of biochemical oscillators". en. In: *Nature Reviews Molecular Cell Biology* 9.12 (Dec. 2008), pp. 981–991. ISSN: 1471-0072, 1471-0080. DOI: 10.1038/nrm2530.

[25] Bartosz Prokop, Nikita Frolov, and Lendert Gelens. "Enhancing model identification with SINDy via nullcline reconstruction". en. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 34.6 (June 2024), p. 063135. ISSN: 1054-1500, 1089-7682. DOI: 10.1063/5.0199311.

[26] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". en. In: *Proceedings of the National Academy of Sciences* 113.15 (Apr. 2016), pp. 3932–3937. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1517384113.

[27] Wen-Xu Wang, Ying-Cheng Lai, and Celso Grebogi. "Data based identification and prediction of nonlinear and complex dynamical systems". en. In: *Physics Reports* 644 (July 2016), pp. 1–76. ISSN: 03701573. DOI: 10.1016/j.physrep.2016.06.004.

[28] Bartosz Prokop and Lendert Gelens. "From biological data to oscillator models using SINDy". en. In: *iScience* 27.4 (Apr. 2024), p. 109316. ISSN: 25890042. DOI: 10.1016/j.isci.2024.109316.

[29] David J. Griffiths. *Introduction to electrodynamics*. eng. Fifth edition. Cambridge New York Port Melbourne New Delhi Singapore: Cambridge University Press, 2024. ISBN: 978-1-00-939775-9.

[30] Dagmar Medková. *The Laplace Equation*. en. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-74306-6 978-3-319-74307-3. DOI: 10.1007/978-3-319-74307-3.

[31] Per Brinch Hansen. "Numerical Solution of Laplace's Equation". en. In: ().

[32] *Qhull code for Convex Hull, Delaunay Triangulation, Voronoi Diagram, and Halfspace Intersection about a Point*.

[33] *scipy.interpolate.LinearNDInterpolator — SciPy v1.13.0 Manual*.

[34] Pauli Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". en. In: *Nature Methods* 17.3 (Mar. 2020), pp. 261–272. ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-019-0686-2.

[35] Kevin Gurney. *An Introduction to Neural Networks*. en. 0th ed. CRC Press, Oct. 2018. ISBN: 978-1-4822-8699-1. DOI: 10.1201/9781315273570.

[36] Kevin L. Priddy and Paul E. Keller. *Artificial neural networks: an introduction*. en. Tutorial texts series. Bellingham, Wash: SPIE Press, 2005. ISBN: 978-0-8194-5987-9.

[37] Xin Feng et al. "Computer vision algorithms and hardware implementations: A survey". en. In: *Integration* 69 (Nov. 2019), pp. 309–320. ISSN: 01679260. DOI: 10.1016/j.vlsi.2019.07.005.

[38] Zhen Tan et al. *Large Language Models for Data Annotation: A Survey*. arXiv:2402.13446 [cs]. June 2024.

[39] Guangsi Shi et al. "Towards complex dynamic physics system simulation with graph neural ordinary equations". en. In: *Neural Networks* 176 (Aug. 2024), p. 106341. ISSN: 08936080. DOI: 10.1016/j.neunet.2024.106341.

[40] Abolfazl Younesi et al. *A Comprehensive Survey of Convolutions in Deep Learning: Applications, Challenges, and Future Trends*. Version Number: 2. 2024. DOI: 10.48550/ARXIV.2402.15490.

[41] Kamal Berahmand et al. "Autoencoders and their applications in machine learning: a survey". en. In: *Artificial Intelligence Review* 57.2 (Feb. 2024), p. 28. ISSN: 1573-7462. DOI: 10.1007/s10462-023-10662-6.

[42] Hyun-il Lim. "A Study on Layers of Deep Neural Networks". en. In: *2020 3rd International Conference on Intelligent Autonomous Systems (ICoIAS)*. Singapore: IEEE, Feb. 2020, pp. 31–34. ISBN: 978-1-72816-078-8. DOI: 10.1109/ICoIAS49312.2020.9081834.

[43] Leo Verhagen Metman and Katie Kompoliti. *Encyclopedia of movement disorders*. eng. 1st ed. Amsterdam: Elsevier Academic Press, 2010. ISBN: 978-0-12-374105-9.

[44] John C. Lindon, Jeremy K. Nicholson, and Elaine Holmes. *The handbook of metabonomics and metabolomics*. eng. 1st ed. Amsterdam Boston: Elsevier, 2007. ISBN: 978-0-444-52841-4.

[45] John Howard. "Artificial intelligence: Implications for the future of work". en. In: *American Journal of Industrial Medicine* 62.11 (Nov. 2019), pp. 917–926. ISSN: 0271-3586, 1097-0274. DOI: 10.1002/ajim.23037.

[46] P. Baldi. "Gradient descent learning algorithm overview: a general dynamical systems perspective". en. In: *IEEE Transactions on Neural Networks* 6.1 (Jan. 1995), pp. 182–195. ISSN: 10459227. DOI: 10.1109/72.363438.

[47] A.T.C. Goh. "Back-propagation neural networks for modeling complex systems". en. In: *Artificial Intelligence in Engineering* 9.3 (Jan. 1995), pp. 143–151. ISSN: 09541810. DOI: 10.1016/0954-1810(94)00011-S.

[48] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. en. arXiv:1412.6980 [cs]. Jan. 2017.

[49] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. *Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark*. en. arXiv:2109.14545 [cs]. June 2022.

[50] Yunji Chen et al. "Fundamentals of neural networks". en. In: *AI Computing Systems*. Elsevier, 2024, pp. 17–51. ISBN: 978-0-323-95399-3. DOI: 10.1016/B978-0-32-395399-3.00008-1.

[51] Grégoire Montavon, Genevieve Orr, and Klaus-Robert Müller, eds. *Neural networks: tricks of the trade*. 2nd ed. Lecture notes in computer science 7700. OCLC: ocn828098376. Heidelberg: Springer, 2012. ISBN: 978-3-642-35288-1.

[52] J. Sola and J. Sevilla. "Importance of input data normalization for the application of neural networks to complex industrial problems". en. In: *IEEE Transactions on Nuclear Science* 44.3 (June 1997), pp. 1464–1468. ISSN: 0018-9499, 1558-1578. DOI: 10.1109/23.589532.

[53] Yun Xu and Royston Goodacre. "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning". en. In: *Journal of Analysis and Testing* 2.3 (July 2018), pp. 249–262. ISSN: 2096-241X, 2509-4696. DOI: 10.1007/s41664-018-0068-2.

[54] Boyuan Chen et al. "Towards training reproducible deep learning models". en. In: *Proceedings of the 44th International Conference on Software Engineering*. Pittsburgh Pennsylvania: ACM, May 2022, pp. 2202–2214. ISBN: 978-1-4503-9221-1. DOI: 10.1145/3510003.3510163.

[55] Viv Bewick, Liz Cheek, and Jonathan Ball. "Statistics review 7: Correlation and regression". In: *Critical Care* 7.6 (2003), p. 451. ISSN: 13648535. DOI: 10.1186/cc2401.

[56] Amir Ali Ahmadi and Bachir El Khadir. *Learning Dynamical Systems with Side Information*. Version Number: 2. 2020. DOI: 10.48550/ARXIV.2008.10135.

[57] Adrien Courtois, Jean-Michel Morel, and Pablo Arias. "Can neural networks extrapolate? Discussion of a theorem by Pedro Domingos". en. In: *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matemáticas* 117.2 (Apr. 2023), p. 79. ISSN: 1578-7303, 1579-1505. DOI: 10.1007/s13398-023-01411-z.

[58] Luis Tadeu Nascimento Pires. "Data-driven modelling of biological oscillators". unpublished. KU Leuven, 2022.

[59] Nils Eling, Michael D. Morgan, and John C. Marioni. "Challenges in measuring and understanding biological noise". en. In: *Nature Reviews Genetics* 20.9 (Sept. 2019), pp. 536–548. ISSN: 1471-0056, 1471-0064. DOI: 10.1038/s41576-019-0130-6.

**Faculty of Science**
Celestijnenlaan 200H bus 2100
3001 LEUVEN, BELGIË
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
www.kuleuven.be