KTH ROYAL INSTITUTE
OF TECHNOLOGY

Licentiate Thesis in Electrical Engineering

Learning flow functions: architectures, universal approximation and applications to spiking systems

MIGUEL AGUIAR

Stockholm, Sweden 2024

Learning flow functions: architectures, universal approximation and applications to spiking systems

MIGUEL AGUIAR

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Licentiate of Engineering on Friday the 15th of March 2024, at 10:00 a.m. in D3, Lindstedtsvägen 9, Kungliga Tekniska högskolan, Stockholm.

Licentiate Thesis in Electrical Engineering
KTH Royal Institute of Technology
Stockholm, Sweden 2024

© Miguel Aguiar

TRITA-EECS-AVL-2024:20
ISBN 978-91-8040-852-3

Printed by: Universitetsservice US-AB, Sweden 2024

Abstract

Learning flow functions of continuous-time control systems is considered in this thesis. The flow function is the operator mapping initial states and control inputs to the state trajectories, and the problem is to find a suitable neural network architecture to learn this infinite-dimensional operator from measurements of state trajectories. The main motivation is the construction of continuous-time simulation models for such systems. The contribution is threefold.

We first study the design of neural network architectures for this problem, when the control inputs have a certain discrete-time structure, inspired by the classes of control inputs commonly used in applications. We provide a mathematical formulation of the problem and show that, under the considered input class, the flow function can be represented exactly in discrete time. Based on this representation, we propose a discrete-time recurrent neural network architecture. We evaluate the architecture experimentally on data from models of two nonlinear oscillators, namely the Van der Pol oscillator and the FitzHugh-Nagumo oscillator. In both cases, we show that we can train models which closely reproduce the trajectories of the two systems.

Secondly, we consider an application to spiking systems. Conductance-based models of biological neurons are the prototypical examples of this type of system. Because of their multi-timescale dynamics and high-frequency response, continuous-time representations which are efficient to simulate are desirable. We formulate a framework for surrogate modelling of spiking systems from trajectory data, based on learning the flow function of the system. The framework is demonstrated on data from models of a single biological neuron and of the interconnection of two neurons. The results show that we are able to accurately replicate the spiking behaviour.

Finally, we prove an universal approximation theorem for the proposed recurrent neural network architecture. First, general conditions are given on the flow function and the control inputs which guarantee that the architecture is able to approximate the flow function of any control system with arbitrary accuracy. Then, we specialise to systems with dynamics given by a controlled ordinary differential equation, showing that the conditions are satisfied whenever the equation has a continuously differentiable right-hand side, for the control input classes of interest.

Sammanfattning

Denna avhandling studerar maskininlärningsmetoder för tidskontinuerliga reglersystem. Vi utgår från en abstrakt systemrepresentation med en lösningsoperator, som avbildar systemets initialtillstånd och insignal på motsvarande tillståndstrajektorian. Målet är att undersöka inlärning av tidskontinuerliga simuleringsmodeller utifrån tillståndsmätningar. Avhandlingen består av tre huvudbidrag.

Vi undersöker först arkitekturer baserade på neurala nätverk, för klasser av insignaler som är brukliga i tillämpningar och har en viss tidsdiskret struktur. Vi formulerar problemet matematiskt, och visar att lösningsoperatorn kan representeras exakt av ett tidsdiskret system. Detta leder till en arkitektur baserad på ett återkopplande neuralt nätverk (RNN), som vi utförligt beskriver, analyserar och validerar med hjälp av data från två modeller av icke-linjära oscillatorer, nämligen Van der Pol oscillatorn och FitzHugh-Nagumo oscillatorn. I båda fall visar vi att vi kan träna modeller som noggrant reproducerar systemens lösningsbanor.

Därefter studerar vi en tillämpning på system vars tillståndstrajektorier kännetecknas av förekomsten av snabba oscillationer i form av impulser, såsom modeller av biologiska neuroner. Denna klass av system karakteriseras av ett flerskaligt och högfrekvent tidssvar, vilket gör det önskvärt att ta fram tidskontinuerliga modeller som är lätta att simulera. Vi lägger fram ett ramverk för inlärning av surrogatmodeller av sådana system från data. Ramverket demonstreras med hjälp av data från en modell av en biologisk neuron och en modell av två kopplade biologiska neuroner, och resultaten visar att våra modeller noggrant reproducerar systemens beteende.

Slutligen tar vi fram ett bevis för ett approximationsteorem för inlärning av lösningsoperatorer av tidskontinuerliga system. Vi visar att den RNN-arkitektur som vi har tagit fram kan approximera godtyckliga reglersystem under vissa villkor som vi först formulerar abstrakt. Sedan bevisar att reglersystem som beskrivs av ordinära differentialekvationer uppfyller dessa villkor, vilket betyder att de kan approximeras av den studerade arkitekturen.

Acknowledgements

I wish first and foremost to express my deepest gratitude to my supervisor, Karl Henrik Johansson. His encouragement, guidance and enduring patience have been crucial for the completion of the work presented here. A similar word of thanks is due to my co-supervisor João Sousa, whose advice I have been fortunate to profit from since I first started doing research. It has been a pleasure to work on the material presented below in collaboration with Amritam Das, to whom I am grateful for his enthusiasm and availability through many hours of enriching discussions. Discussions with Matthieu Barreau and Ingvar Ziemann were also important in the development of this research direction. I would like to extend my sincere thanks to Maarten Schoukens for graciously accepting to review this thesis, and to Mikael Johansson for the readiness to serve as advance reviewer and examiner. I am moreover indebted to Jacob Lindbäck, Braghadeesh Lakshminarayanan and Daniel Selvaratnam for their availability to help in proofreading parts of this thesis, and to Robert Bereza for help with some last-minute logistics. This work was supported by the Swedish Research Council Distinguished Professor Grant 2017-01078 and a Knut and Alice Wallenberg Foundation Wallenberg Scholar Grant. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at C3SE, partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

Many thanks are owed to my past and present colleagues at the Division for Decision and Control Systems, in particular those who I've been fortunate to collaborate with on research and teaching, my office companions Mayank Sewlia and Pedro Roque, and all who regularly grace the kitchen with their presence at lunchtime.

I also sincerely thank my friends in Stockholm, Porto and elsewhere. A particular word of appreciation is due to Jesper Provoost for his constancy and generosity in friendship. Saint Eugenia's church in Stockholm has been an essential place of recollection and solace; I dearly thank all involved in the parish community, in particular Fr. Dominik Terstriep SJ.

I wish to thank my family for invariably welcoming me back home and ensuring that I do not stray too far from the Portuguese culinary tradition. In particular, my parents have often been an indispensable help and I am grateful for their listening ear and advice.

My fiancée Agnes has been a source of inspiration and peace in abundance. I thank her deeply for the encouragement, understanding and her patient love, as well as for making sure my sugar and caffeine levels were kept acceptably high during the writing of this document.

Finally, I give thanks to God, my consolation, for the grace of being able to do this work and for placing all those mentioned above in my path. *Send forth thy light and thy truth: they have conducted me, and brought me unto thy holy hill, and into thy tabernacles.*

Contents

1 Introduction **3**
1.1 Motivation 3
1.2 Problem formulation 7
1.3 Background 9
1.4 Contributions and outline 12

2 An architecture for flow function learning **15**
2.1 Introduction 15
2.2 Problem formulation 18
2.3 Methodology 23
2.4 Training procedure 27
2.5 Numerical evaluation: Van der Pol oscillator 28
2.6 Numerical evaluation: FitzHugh-Nagumo oscillator 36
2.7 Summary 39

3 Surrogate modelling of spiking systems **41**
3.1 Introduction 41
3.2 Problem formulation 44
3.3 Methodology 46
3.4 Numerical experiments 50
3.5 Summary 53

4 A universal approximation theorem **55**
4.1 Introduction 55
4.2 Preliminaries 57
4.3 Architecture definition 58
4.4 Universal approximation of flow functions 60
4.5 Flows of controlled ODEs 63
4.6 Summary 67
4.7 Appendix 67

5 Conclusions **71**
5.1 Summary 71
5.2 Future work 72

References **75**

Chapter 1

Introduction

In this chapter we introduce the topic addressed in this thesis. We begin by motivating the work and describing some application examples. Thereafter we formulate the problem studied in the thesis and the specific research questions addressed in each chapter. This is followed by a background section where we provide a literature review. Finally, we give an outline of the thesis and describe the contributions in each of the chapters.

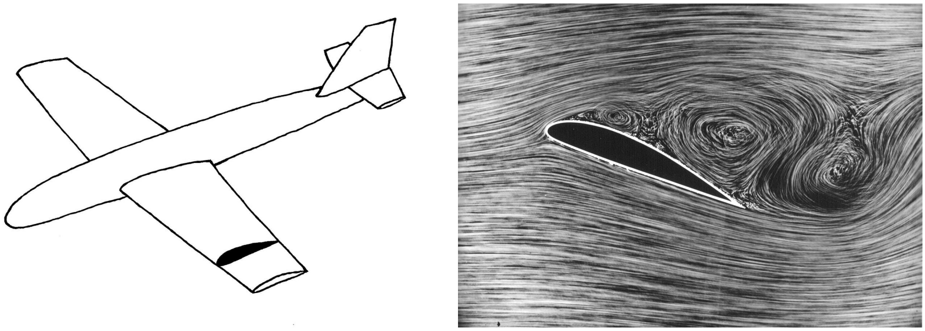
1.1 Motivation

Simulation of dynamical systems is an essential aspect of engineering practice in a large number of application domains. Across engineering disciplines, computer simulations are extensively used for validation, and increasingly integrated as part of the design process itself (Koziel and Leifsson, 2016, ch. 1). Predictive control, in which a simulation model of the system to be controlled is an essential component, has established itself as a standard method (Schwenzer et al., 2021). The drive for holistic digitalisation in systems design, manufacturing and operation through the use of digital twin technology (Semeraro et al., 2021) implies that simulation will play an ever more prominent role. Hence, there is a growing need for simulation models, which not only are computationally efficient and fast to evaluate, but also possible to integrate in optimisation algorithms.

As the complexity of engineering systems increases, modelling based on first principles becomes infeasible. Even in cases where first-principles models are available, they might be too complex to be useful for simulation, due to large state spaces, complex nonlinearities and geometries, or spatiotemporal dynamics. Increasing capabilities for data collection and storage, in tandem with greater availability of high-performance computing resources, have motivated a surge of research attention on data-driven modelling of dynamical systems (Ghadami and Epureanu, 2022). Machine learning algorithms, especially those based on neural network architectures, have been a point of particular interest. While this comprehends a large collection of methods, a common thread is that the resulting models are computationally efficient to simulate. Additionally, since automatic differentiation is the backbone of these algorithms, the outputs of the simulation can be easily and efficiently differentiated, allowing for their integration in optimisation algorithms. Furthermore, machine learning methods are supported by a large software ecosystem and mature toolchains, facilitating performant implementations (Azizzadenesheli et al., 2024).

In this thesis, we address the problem of constructing simulation models of dynamical systems, focusing in particular on continuous-time control systems. We adopt the flow function representation of such systems, and study the application of machine learning techniques for learning continuous-time models from data. In what follows, we illustrate the discussion thus far by three application examples.

Aircraft wing shape optimisation



(a) (b)

Figure 1.1: (a) Illustration of an aerofoil as a cross-section of an aeroplane wing. Image source: `https://commons.wikimedia.org/wiki/File:1915ca_abger_fluegel_(cropped_and_mirrored).jpg`, authored by DLR under a CC-BY 3.0 licence. (b) Flow around an aerofoil in a wind tunnel. Image source: `https://commons.wikimedia.org/wiki/File:Airfoil_cross_section.jpg`, authored by J Doug McLean at English Wikipedia.

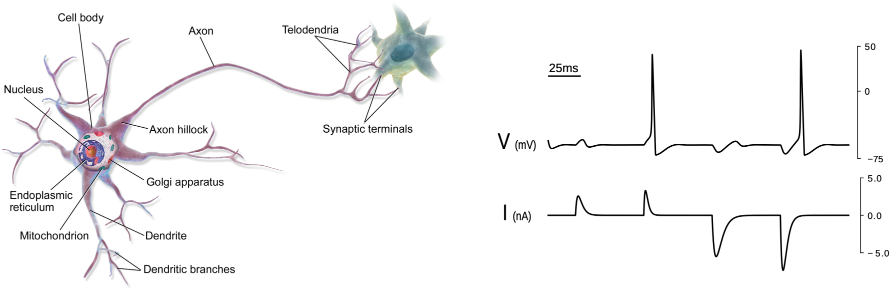
Consider the problem of designing the shape of an aerofoil section of a fixed-wing aircraft (Leifsson and Koziel, 2015, ch. 1, 2 and 10). A typical objective is to obtain an aerofoil shape that maximises the lift while satisfying drag and area constraints. The shape may be parameterised by a spline curve, and the design problem then has a natural formulation as a constrained minimisation problem. However, the computation of the lift and drag coefficients associated to a given shape calls for running a computational fluid dynamics (CFD) simulation, requiring computation times in the order of hours to days.

While simulation-based design has successfully replaced expensive experimental tests in the initial stages of the aircraft design process, design optimisation using

CFD simulations presents a challenge, as iterative optimisation solvers will typically require a large number of evaluations of the cost function and its gradient, both of which are prohibitively costly in terms of computation time.

One solution is the replacement of the fluid dynamics model with a so-called surrogate model that provides an approximation of the high-fidelity CFD model. The surrogate model is computationally less costly to evaluate and differentiate. Though there exists a variety of ways to construct surrogate models (Frangos et al., 2010), a class of methods with particularly interesting properties is that of neural operators, which have been shown to be able to accurately approximate solution operators of partial differential equations, including those arising in fluid dynamics (Azizzadenesheli et al., 2024). Furthermore, these models are fast to evaluate and, because they are based on layered neural network architectures, gradients can be efficiently computed using automatic differentiation.

Biophysical neural networks



(a) (b)

Figure 1.2: (a) Illustration of a neuron cell. Image source: `https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png`, authored by Bruce Blaus. (b) Current input (I) and corresponding voltage output (V) in a spiking system. Figure taken from `https://arxiv.org/pdf/1704.04989.pdf`.

Consider the problem of simulating the biophysical dynamics of networks of biological neurons, as well as that of designing electronic components which replicate behaviours encountered in these networks.

The principal means of communication between neuron cells are spikes or action potentials, a rapid change in the voltage across the neuron membrane which is propagated to other cells through an elongated portion of the neuron called the axon (Izhikevich, 2006, ch. 1). By means of a voltage clamp experiment performed on the squid giant axon, Hodgkin and Huxley (1952) showed that the fundamental properties of the action potential produced in the axon membrane can be explained by the dynamics of sodium and potassium ionic currents in the axon (Nelson, 2005). Furthermore, they identified an equivalent circuit model of the axon, where the mechanisms responsible for the ionic currents are represented by nonlinear conductances. Conductance-based models have since then become the main paradigm for modelling the dynamics of individual neurons; moreover, these can be interconnected to model networks of several neurons coupled through electrical and chemical synapses, see (Giannari and Astolfi, 2022) and references therein.

The differential equations corresponding to such circuit equivalents of neurons and their interconnections are highly nonlinear and stiff, with dynamics exhibiting multiple timescales, input-dependent stability properties and high-frequency signals, meaning that the simulation of the network dynamics through direct integration of the corresponding system of differential equations is computationally challenging. These issues, combined with the fact that the models do not possess fading memory with respect to the input currents, imply that parameter identification is nontrivial, even for models of a single neuron (Burghi, 2020; Almog and Korngreen, 2016; Stiefel and Brooks, 2019).

These circuit models make it possible to fabricate electronic devices which operate in the same way as biological neural networks (Indiveri et al., 2011; Ribar and Sepulchre, 2021), presenting an efficient alternative to traditional computing. The design of such systems thus poses the problem of tuning the network topology and the conductance parameters of individual neurons, such that the circuit exhibits the desired behaviours (Sepulchre, 2022).

In Chapter 3, we address the construction of surrogate models for this class of systems, focusing on the problem of simulating their dynamics.

Control of building HVAC systems

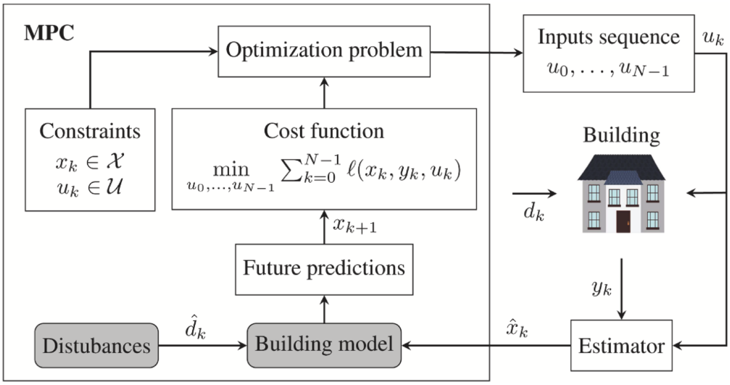

MPC
Optimization problem
Inputs sequence
$u_0, \ldots, u_{N-1}$
u_k
Constraints
$x_k \in \mathcal{X}$
$u_k \in \mathcal{U}$
Cost function
$\min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} \ell(x_k, y_k, u_k)$
Building
d_k
x_{k+1}
Future predictions
y_k
$\hat{x}_k$
$\hat{d}_k$
Distubances
Building model
Estimator

Figure 1.3: Block diagram of a model predictive control loop for a building HVAC system. Figure taken from (Drgoňa et al., 2020).

Building heat, ventilation and air conditioning (HVAC) systems, traditionally controlled by rule-based controllers, are responsible for a large portion of the energy use in buildings, and these in turn constitute a significant amount of global energy expenditure (Drgoňa et al., 2020). Predictive control has been proposed as a way to improve energy efficiency, and several studies have suggested that significant gains can be obtained by way of enhancing the control strategies used in these systems.

Drgoňa et al. (2020) name modelling as one of the main challenges in the use of predictive control in HVAC systems. Indeed, the goal in predictive control is to optimise, at each time instant, an objective which depends on the predicted state trajectory of the system over a given time horizon. Hence, this requires the availability of a model to simulate the state of the system forward in time from the current measured state. Typically, discrete-time dynamics models are incorporated into the optimisation problem as constraints. This presents a twofold challenge. Because of the highly nonlinear dynamics of HVAC systems, complex models are required to represent the process accurately, and these are hard to obtain from first principles. At the same time, the use of too complex a model will increase the difficulty of solving the optimisation problem, or even render it intractable.

The use of simulation models based on flow functions, such as those we propose in this thesis, thus provides an interesting alternative to traditional models in this application. These models can be obtained from data, and, as the trajectories of the system may be evaluated directly, one can do away with nonlinear constraints related to the dynamics of the system. Furthermore, the gradient of the state with respect to the control input can be computed by automatic differentiation, an advantage when using such models in optimisation formulations, as is the case in predictive control.

1.2 Problem formulation

We consider continuous-time time-invariant control systems described by a flow function φ. As illustrated in Figure 1.4, for a time instant $t \geq 0$, initial state x and input signal u, the flow φ maps the triple (t, x, u) into the value of the state of the system at time t. In other words, the function of time ξ given by

$$\xi(t) = \varphi(t, x, u),\ t \geq 0$$

is the state trajectory of the system with the initial state x, under the input u. The flow function is characterised by two fundamental properties, namely the identity property and the semigroup property. The identity property states that $\xi(0) = x$, that is, that the initial state of the trajectory $\varphi(\cdot, x, u)$ is indeed x. The semigroup property describes the behaviour of the system under concatenation of inputs, and is the expression of the fact that the state $\xi(t)$ at time t summarises all the past behaviour of the system.

We first consider how to learn an approximation of φ from measurements ξ_{ik} of the state taken at discrete time instants t_{ik} from different trajectories:

$$\xi_{ik} = \varphi(t_{ik}, x_i, u_i) + v_{ik}, \tag{1.1}$$

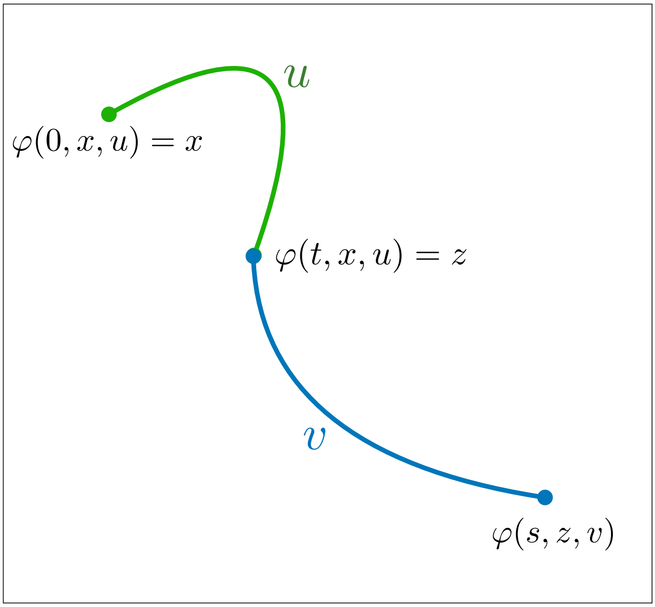


Figure 1.4: Illustration of the flow function of a control system. A control input u is applied for t units of time starting from the state x. The system state travels along the green trajectory curve from x to $z = \varphi(t, x, u)$. Afterwards, a control input v is applied for s units of time, and the system state is transferred along the blue curve to $\varphi(s, z, v)$.

where x_i, u_i are known initial conditions and inputs and v_{ik} is measurement noise. Given an approximation $\hat{\varphi}$ of φ, we can quantify how close it is to the true flow function using a mean squared error measure

$$\ell_T(\hat{\varphi}) = \mathbf{E}_{x,u} \frac{1}{T} \int_0^T \left\| \varphi(t, x, u) - \hat{\varphi}(t, x, u) \right\|^2 \mathrm{d}t,$$

for a given time horizon $T > 0$ and probability distributions of the initial state x and the control input u. The overarching problem considered in this thesis thus consists in finding such an approximation $\hat{\varphi}$ which minimises ℓ_T, given a collection of measurements as in (1.1).

Solving the learning problem requires the definition of an hypothesis space $\mathcal{H}$, that is, a space of models from which an approximation $\hat{\varphi} \in \mathcal{H}$ should be selected. The hypothesis space should contain functions close to the true flow function, so that a low value of ℓ_T can be attained. At the same time, it must be practical to optimise ℓ_T over $\mathcal{H}$. We thus state our first research question as follows.

Question 1. What is an appropriate hypothesis space for solving the flow function learning problem?

Question 1 is addressed in Chapter 2.

We consider in particular an application of flow function learning to spiking systems. As described in the previous section, simulating these systems is a nontrivial task, due to their multi-timescale and high-frequency response. Conductance-based models are the prototypical examples of such systems, and provide a state-space model of spiking behaviour. We thus focus on the following question.

Question 2. How can we learn a surrogate model of a spiking system from samples of state trajectories?

Chapter 3 addresses this question, and we propose a solution based on our flow function learning approach.

Our results regarding Question 1 indicate that a hypothesis space based on a discrete-time recurrent neural network architecture is able to solve the learning problem. This motivates a theoretical investigation of the approximation capabilities of the architecture.

Question 3. Under what conditions do discrete-time recurrent neural networks universally approximate flow functions of continuous-time control systems?

Our contribution towards this question is found in Chapter 4, where we provide conditions on the flow function and the input signal class for the architecture to approximate the flow function arbitrarily well, both in a general setting and in the special case of systems with dynamics given by controlled ordinary differential equations.

1.3 Background

In this section, we provide an overview of the literature related to the topic considered in this thesis. The problem of simulating dynamical systems has a long history and has been approached from many perspectives and in many application areas. Therefore, our goal is not to present an exhaustive review of the literature concerning this topic, and we focus on the fields closest to the work we present here, namely, inverse problems and optimal design, system identification, and machine learning. Moreover, we are primarily concerned with continuous-time methods and control applications.

Concerning the general role of simulation in the fields of dynamical systems and control, as differential equations are the original 'mother tongue' for modelling dynamical systems (Holmes, 1990), simulation is historically closely linked to the field of numerical integration (Cellier and Kofman, 2006). Naturally, then, the relevance of simulation in engineering has grown concurrently with the generalised availability of computers, which was the main cause for the development of numerical analysis methods in the 20th century (Brezinski and Wuytack, 2001). In control engineering, simulation has long played an important role, particularly in the design process. The most detailed model available of a system is typically too complex for control design, so a controller is designed for a simplified (e.g. linearised or lower-order) model and validated by simulation on the full system model (Atherton, 1987; Otter and Cellier, 2000).

Efficient simulation of dynamical systems is crucial in inverse problems and design optimisation, as both require a large number of simulation runs. For many systems, first-principles models can be too complex to simulate efficiently, and a surrogate model is used instead in these applications. A surrogate model is an approximated model intended to replace a high-fidelity model while being less expensive to simulate. Two broad classes of methods exist for constructing such models. Physics-based surrogate models are application-dependent and exploit simplifications or coarser discretisations of the equations which define the high-fidelity model. Data-fit models, on the other hand, use a general-purpose interpolation or regression method, such as a Gaussian process or a neural network, to fit data generated using the high-fidelity model. In surrogate-based design optimisation, an iterative process is used where a design is first optimised using the surrogate model and then validated on the high-fidelity model. The data from the validation process can then be used to update the surrogate model, and the optimisation may thus be repeated on the improved surrogate. We refer to (Frangos et al., 2010; Koziel, Ciaurri and Leifsson, 2011) for a survey of these methods and their use in inverse problems and optimisation, respectively.

System identification addresses the data-driven construction of models for control systems. An extensive exposition of classical and modern techniques is given in (Ljung, 1999), focusing on time-domain prediction-error methods. In this approach, the prediction error of the identified model with respect to the observed data is optimised. Pintelon and Schoukens (2012) address methods for identification in the frequency domain. Early work on system identification focused on linear models, but a variety of methods have been proposed for nonlinear systems. A description of the main issues in nonlinear identification as well as an extensive survey can be found in (Schoukens and Ljung, 2019). The above references focus for the most part on discrete-time models. Concerning identification of continuous-time models, two main classes of methods can be distinguished: indirect methods, based upon the identification of a discrete-time model from the data which is then converted into a continuous-time model; and direct methods, which provide a continuous-time model directly; see (Garnier, 2015; González, 2022) for literature surveys.

The field of machine learning comports a large number of methods for constructing prediction models from examples, an overview is given in (Hardt and Recht, 2022, ch. 1). Driven by the increase in computing power, especially the use of graphical processing units for general computing purposes, and the ability to store and process ever larger volumes of data, a number of these methods have become widely popular, in particular those based on deep learning techniques. These are broadly characterised by the use of layered neural network architectures to solve empirical risk minimisation problems by stochastic gradient methods. The structure of these architectures, resulting from the composition of general-purpose parametric nonlinearities, endows them with approximation capabilities obviating the need for engineering problem-specific features, formerly the common practice. Furthermore, the layered structure is amenable to the use of efficient automatic differentiation methods, considerably facilitating the process of optimising these models. We refer to (Hardt and Recht, 2022, ch. 7) for a recent detailed overview of the theory and practice of deep learning.

Deep learning methods have been used in prediction and analysis of dynamical systems (Ghadami and Epureanu, 2022) and in system identification (Pillonetto et al., 2023). A research direction which has received much attention, usually called physics-informed machine learning, addresses the integration of physical or system-theoretical constraints in these algorithms. A survey of such approaches with applications in control is given in (Nghiem et al., 2023). Regarding continuous-time methods, we can distinguish three main approaches. A first class of methods consist in parameterising the right-hand side of an ordinary differential equation by a neural network (Chen et al., 2018). Related methods have been proposed for continuous-time system identification (Forgione and Piga, 2021; Rahman et al., 2022; Beintema, Schoukens and Tóth, 2023). The second approach consists in using a neural network architecture to approximate solution operators of differential equations. These architectures, called neural operators, provide differentiable, discretisation-independent approximations of infinite-dimensional operators which are efficient to simulate, since simulation amounts to the evaluation of a nonlinear map, properties which are attractive in inverse and design optimisation problems. Additionally, these methods are purely data-driven in the sense that they do not require knowledge of the dynamics of the system being approximated. An overview of neural operator methods is given in (Azizzadenesheli et al., 2024). Bhan, Shi and Krstic (2023) describe applications of neural operators in nonlinear adaptive control design. The architecture for flow function learning we propose in Chapter 2 belongs to this class of methods. Finally, physics-informed neural network methods are typically applied in forward and inverse problems in partial differential equations. In both cases, a neural network is used to parameterise a single solution of the differential equation, as a function of time and space. The equation residual is used as a regulariser in a regression problem, which in a forward problem consists in fitting the network to data for the boundary values, while in an inverse problem the network is fit to measurements of a solution. We refer to (Karniadakis et al., 2021) for a survey of these methods.

1.4 Contributions and outline

The remainder of this thesis is organised as follows. Chapter 2 describes and analyses a recurrent neural network architecture for learning flow functions. Chapter 3 considers the application of such an architecture to the problem of surrogate modelling of spiking systems. Chapter 4 formulates and proves a universal approximation theorem for the proposed architecture. Finally, in Chapter 5 we summarise the thesis and describe future research directions.

Below, we give a summary of Chapters 2, 3 and 4 and the contributions therein.

Chapter 2

In Chapter 2 we propose a recurrent neural network (RNN) based architecture to learn the flow function of a causal, time-invariant and continuous-time control system from trajectory data. By exploiting the structure of the classes of inputs commonly used in practice, we show that learning the flow function is equivalent to learning the input-to-state map of a discrete-time dynamical system. This motivates the use of an RNN together with encoder and decoder networks which map the state of the system to the hidden state of the RNN and back. We experimentally validate the proposed method using models of the Van der Pol and FitzHugh-Nagumo oscillators. In both cases, the results demonstrate that the architecture is able to closely reproduce the trajectories of these two systems. For the Van der Pol oscillator, we further provide an extensive study of the capabilities of the trained models to simulate trajectories outside of the training distribution, as well as the sensitivity of the training algorithm to measurement noise and the amount of measurements.

The chapter is partly based on the following publication:

- M. Aguiar, A. Das and K. H. Johansson (2023a). 'Learning Flow Functions from Data with Applications to Nonlinear Oscillators'. In: *Proceedings of the 22nd IFAC World Congress*.

Chapter 3

In Chapter 3 we propose a framework for surrogate modelling of spiking systems. These systems are often described by stiff differential equations with high-amplitude oscillations and multi-timescale dynamics, making surrogate models an attractive tool for system design and simulation. We parameterise the flow function of a spiking system using a recurrent neural network architecture, allowing for a direct continuous-time representation of the state trajectories. The spiking nature of the signals makes for a data-heavy and computationally hard training process, and we describe two methods to mitigate these difficulties. We demonstrate the framework numerically on two conductance-based models of biological neurons, showing that we are able to train surrogate models which accurately replicate the spiking behaviour.

This chapter is based on the following publication:

- M. Aguiar, A. Das and K. H. Johansson (2023b). ‘Learning Flow Functions of Spiking Systems’. In: *arXiv e-prints* arXiv:2312.11913. Submitted to the 6th Annual Learning for Dynamics & Control Conference.

Chapter 4

In Chapter 4 we consider the problem of approximating flow functions of continuous-time dynamical systems with inputs. We prove that an architecture based on discrete-time recurrent neural networks universally approximates flows of such systems. The considered class of inputs, modelling signals commonly used in practice as control inputs, is shown to induce a discrete structure in the flow function, which we exploit. We first obtain the result in an abstract setting, where it is demonstrated to hold under certain assumptions on the flow function. We then specialise to systems whose dynamics are well-behaved controlled ordinary differential equations, showing by system-theoretic arguments that the required assumptions hold.

This chapter is based on the following publication:

- M. Aguiar, A. Das and K. H. Johansson (2023c). ‘Universal Approximation of Flows of Control Systems by Recurrent Neural Networks’. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*.

Chapter 2

An architecture for flow function learning

In this chapter we propose a recurrent neural network architecture for learning flow functions of time-invariant continuous-time control systems from measurements of state trajectories. By considering the classes of control inputs most commonly used in practical applications, we show that a discrete-time structure is induced in the flow function, which we exploit to derive our architecture. We demonstrate the architecture on data from the Van der Pol and FitzHugh-Nagumo oscillators, highlighting its generalisation capabilities.

2.1 Introduction

Motivation

Models play a vital role in control engineering. For instance, in predictive control, the model is used to predict the future evolution of the state variables and acts as a constraint in the formulation of the optimal control problem. With increasing complexity, the curse of dimensionality limits the usefulness of standard first-principle models. This limitation has motivated research on data-driven approximation of such physical models, in particular using methods from machine learning. Besides fast simulation for arbitrary initial conditions, many learning methods allow for efficient computation of the gradients of the model with respect to initial conditions, parameters or input signals, which is an attractive property when using these models in inverse problems and design optimisation, for instance.

The advantage of modelling continuous-time dynamical systems directly in continuous time has been pointed out in a number of recent works (Chen et al., 2018; De Brouwer et al., 2019; Beintema, Schoukens and Tóth, 2023). Such models naturally handle irregularly sampled or missing data, and are the natural model class for most physical systems.

This motivates the search for a corresponding learning scheme for continuous-time control system where inputs are present. However, this is a nontrivial problem since the domain of the solution operator of a continuous-time control system is infinite-dimensional, as the control inputs are functions of time.

Related work

Some approaches for continuous-time identification have been proposed (Garnier, 2015), but for nonlinear systems the majority of research concentrates on discrete-time models (Schoukens and Ljung, 2019). A number of modelling approaches have arisen using ideas and model classes from classical and deep machine learning.

Neural ordinary differential equations (Neural ODEs) (Chen et al., 2018) are a particular class of continuous-time models proposed to replace standard network layers appearing in models used for common learning tasks, and have been shown to be competitive with state-of-the-art models in system identification (Rahman et al., 2022). In (Forgione and Piga, 2021) some specific architectures and learning methods for identifying differential equation models of control systems using neural networks are presented, and Beintema, Schoukens and Tóth (2023) present an framework for continuous-time identification from noisy input-output data. As the dynamics correspond to the time derivative of the flow, a neural ODE must be integrated through an ODE solver to obtain the system trajectories, representing an extra computational burden both for prediction and for computing gradients. Furthermore, errors in the learned dynamics will accumulate over time when the dynamics are integrated, and the error in the simulated trajectory can become unbounded.

An assortment of related methods have been proposed for modelling autonomous systems with applications in the physical sciences (Geneva and Zabaras, 2022; Floryan and Graham, 2022; Brunton, Proctor and Kutz, 2016). For methods based on Koopman operator approximation in particular, extensions to certain classes of systems with inputs are possible (Bevanda, Sosnowski and Hirche, 2021). Physics-informed neural networks have also emerged as a paradigm for learning solutions of ordinary and partial differential equations from data (Raissi, Perdikaris and Karniadakis, 2019; Karniadakis et al., 2021). These methods incorporate a set of differential equations known to be satisfied by the data as a regulariser in the loss function used to train the network, and can also be used to identify parameters in the equations.

In contrast to the majority of these approaches, the class of methods known as neural operator methods attempt to directly learn the solution operator of a differential equation, that is, the operator mapping initial conditions, forcing terms and parameters to the corresponding solution, rather than identifying the governing equations (Kissas et al., 2022; Li, Kovachki et al., 2021; Lu et al., 2021). The focus is then on engineering architectures with the appropriate inductive biases for a particular class of problems. Biloš et al. (2021) propose a method in this vein for learning the flow function of an autonomous dynamical system. Related methods for learning the one-step-ahead map of a non-autonomous system have been considered in (Qin et al., 2021; Lin, Moya and Zhang, 2023).

Contribution

The contributions in this chapter are a recurrent neural network architecture for learning flow functions of continuous-time control systems and a detailed experimental demonstration of the performance and generalisation capabilities of the proposed architecture in predicting the response of nonlinear oscillators. We provide a mathematical formulation of the problem of learning the flow function of a control system, showing that it can be reduced to a tractable optimisation problem. The inputs are restricted to a class of signals which are parameterised by a sequence of parameters, corresponding to the type of inputs commonly used in practice. We show that, for these input signals, the continuous-time flow function can be efficiently approximated by a discrete-time recurrent neural network-based architecture. We demonstrate the capabilities of the proposed architecture in predicting the input-dependent response of the Van der Pol and FitzHugh-Nagumo oscillators.

This approach has a number of advantages in comparison with methods based on learning the right-hand side of a differential equation. Errors in the learned dynamics can be propagated and affect long-term prediction performance. When the flow is directly approximated, the need for integration is obviated. This has the additional advantage of reducing the computational burden at both training and prediction time. In effect, under our formulation, the problem of learning a flow function amounts to a standard regression problem, and thus enables the use of off-the-shelf learning frameworks for training the model. At prediction time, one can query the solution map at any time instant, and, since the model uses standard neural network components, gradients of the flow with respect to, e.g. initial conditions or control values can be computed in a straightforward manner through automatic differentiation. Furthermore, the approach is able to accommodate more general classes of systems than those with dynamics given by controlled ODEs.

Outline

The remainder of this chapter is organised as follows. In Section 2.2, we describe the notion of a flow function for the class of dynamical systems under study, specify the class of control inputs and give a mathematical formulation of the flow function learning problem. In Section 2.3 we motivate and present the proposed architecture, and in Section 2.4 we describe the training procedure which is used in the two following sections. Sections 2.5 and 2.6 present results from numerical experiments on data from models of the Van der Pol and FitzHugh-Nagumo oscillators, respectively. Finally, Section 2.7 summarises the chapter.

2.2 Problem formulation

In this section, we formulate the problem addressed in this chapter. We first give a definition of the flow function of a control system and describe its properties. Thereafter, we define the class of inputs which we consider in the remainder of the chapter. We then state the problem under consideration.

Flow functions

A time-invariant control system Σ is defined by a triple

$$\Sigma = (\mathcal{X}, \mathbb{U}, \varphi), \tag{2.1}$$

where the set $\mathcal{X}$ is the state space of the system and $\mathbb{U}$ is a set of input functions $u : \mathbb{R}_{\geq 0} \to \mathcal{U}$, where $\mathcal{U}$ is a set of input values. The flow function $\varphi : \mathbb{R}_{\geq 0} \times \mathcal{X} \times \mathbb{U} \to \mathcal{X}$ is a map dictating the temporal evolution of the system state under inputs from $\mathbb{U}$. The trajectory of Σ with initial state $x \in \mathcal{X}$ and input $u \in \mathbb{U}$ is the function $\xi : \mathbb{R}_{\geq 0} \to \mathcal{X}$ given by

$$\xi(t) = \varphi(t, x, u),\ t \geq 0.$$

The flow function φ satisfies the following properties:

1. *Identity*: For any state $x \in \mathcal{X}$ and input $u \in \mathbb{U}$, it holds that $\varphi(0, x, u) = x$.
2. *Semigroup*: Let $t \geq 0$ and $u, v \in \mathbb{U}$. Let the concatenation of u and v at time t, $u \underset{t}{\wedge} v$ be defined by

   $$\left[u \underset{t}{\wedge} v\right](s) = \begin{cases} u(s), & 0 \leq s < t \\ v(s-t), & s \geq t \end{cases}.$$

   Then $u \underset{t}{\wedge} v \in \mathbb{U}$ and

   $$\varphi(t+s, x, u \underset{t}{\wedge} v) = \varphi(s, \varphi(t, x, u), v) \tag{2.2}$$

   for all $s \geq 0$ and $x \in \mathcal{X}$.

The semigroup property implies a form of causality for the flow function: taking $s = 0$ in (2.2) we get

$$\varphi(t, x, u \underset{t}{\wedge} v) = \varphi(t, x, u),\ v \in \mathbb{U},$$

in other words, the state at time t depends only on the values of the input on $[0, t)$.

In this chapter we restrict our attention to finite-dimensional dynamical systems. Hence, in (2.1), $\mathcal{X} \subset \mathbb{R}^{d_x}$, $\mathcal{U} \subset \mathbb{R}^{d_u}$, and $\mathbb{U} = L^\infty(\mathbb{R}_{\geq 0}, \mathcal{U})$ is the set of measurable and essentially bounded functions $u : \mathbb{R}_{\geq 0} \to \mathcal{U}$. We will assume that trajectories of Σ are continuous, that is, for each $x \in \mathcal{X}$ and $u \in \mathbb{U}$, the function $t \mapsto \varphi(t, x, u)$ is continuous.

The flow function representation we make use of is well-known in the mathematical theory of autonomous dynamical systems (Arnol'd, 1991), and is not foreign to control theory, see (Sontag, 1998, ch. 2 and references therein). Our formulation is similar to that of Willems (1972), but we simplify the notation as we take initial conditions to always hold at $t = 0$. The more routine representation of a system in terms of a controlled ordinary differential equation $\dot{\xi} = f(\xi, u)$ is subsumed under the flow function representation, but the latter also includes more general classes of systems, such as systems with switched dynamics. However, the continuity condition excludes hybrid systems with jumps, for instance.

Class of inputs

The problem of approximating the flow function φ of a system naturally requires specifying the set of (t, x, u) over which such an approximation should hold. The choice of inputs u is particularly sensitive, as u is a function and hence the set of inputs $\mathbb{U}$ is typically a subset of an infinite-dimensional space. In practice, however, the inputs applied to a control system are often the output of a digital-to-analogue converter which generates a continuous-time input signal u from some discrete-time sequence of parameters through an interpolation scheme. The most common case is that of piecewise-constant input signals, where the interpolation is performed by a zero-order hold. In other cases, higher-order polynomial or spline representations may be used. We consider a general parameterisation of the control inputs which encompasses all of these cases. The main requirement is that each control input can be represented by a sequence of parameters in such a way that the causality of the flow φ is preserved when φ is seen as a function of this sequence.

More precisely, let $\Delta > 0$, let $\Omega_p = \big\{(\omega_k)_{k \in \mathbb{Z}_{\geq 0}} : \omega_k \in \mathbb{R}^p\big\}$ be the set of $\mathbb{R}^p$-valued sequences, and $\alpha : \mathbb{R}^p \times \mathbb{R}_{\geq 0} \to \mathcal{U}$ be periodic with period 1 in its second argument. We say that an input $u : \mathbb{R}_{\geq 0} \to \mathcal{U}$ is *causally parameterised by* α *with period* Δ if there is a sequence of parameters $(\omega_k)_{k \in \mathbb{Z}_{\geq 0}} \in \Omega_p$ such that

$$u(t) = \alpha\left(\omega_{k_t}, \frac{t}{\Delta}\right),$$

for all $t \geq 0$, where $k_t := \lfloor t/\Delta \rfloor$.

To give some examples, $\alpha(\omega, t) = \omega$ corresponds to the case of piecewise constant inputs with period Δ, and $\alpha((\omega^a, \omega^b), t) = (1 - t)\omega^a + t\omega^b$ corresponds to piecewise linear inputs, as illustrated in Figure 2.1.

Now, assume that a control period Δ and a function α as above have been fixed. Any u which is causally parameterised by α with period Δ can be written

$$u(t) = \mathfrak{C}_{\alpha,\Delta}[(\omega_k)](t) := \sum_{k=0}^{\infty} \alpha\left(\omega_k, \frac{t}{\Delta}\right) \mathbf{1}_{[k\Delta,(k+1)\Delta)}(t) \tag{2.3}$$

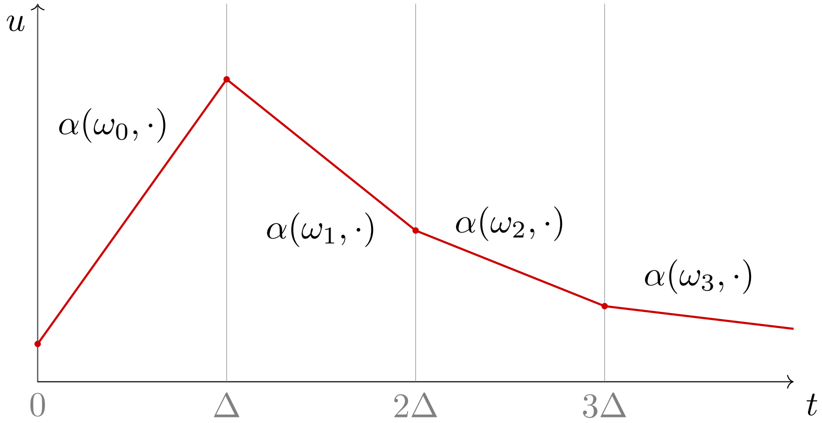


Figure 2.1: Illustration of the considered input parameterisation for the case of piecewise linear inputs.

for some sequence $(\omega_k) \in \Omega_p$, where $\mathfrak{C}_{\alpha,\Delta} : \Omega_p \to \mathcal{U}^{\mathbb{R}_{\geq 0}}$ maps the sequence (ω_k) to the input signal u. We thus restrict our attention to the set of inputs $\mathbb{U}_{\alpha,\Delta}$, defined as the image of $\mathfrak{C}_{\alpha,\Delta}$:

$$\mathbb{U}_{\alpha,\Delta} := \mathfrak{C}_{\alpha,\Delta}[\Omega_p] = \{\mathfrak{C}_{\alpha,\Delta}[(\omega_k)] : (\omega_k) \in \Omega_p\}\,. \tag{2.4}$$

We assume that α is bounded and measurable, so that $\mathbb{U}_{\alpha,\Delta} \subset \mathbb{U}$.

Problem statement

We consider the problem of approximating a given flow function φ from measurement data on the time interval $[0,T]$. The approximation is to be selected from a hypothesis class $\mathcal{H} \subset \{\hat{\varphi} : \mathbb{R}_{\geq 0} \times \mathcal{X} \times \mathbb{U}_{\alpha,\Delta} \to \mathcal{X}\}$, and we define for $\hat{\varphi} \in \mathcal{H}$ the loss function

$$\ell_T(\hat{\varphi}) := \mathbf{E}_{x,u}\left[\frac{1}{T}\int_0^T \|\hat{\varphi}(t,x,u) - \varphi(t,x,u)\|^2 \,\mathrm{d}t\right], \tag{2.5}$$

where the input signal u and initial condition x are independent and drawn from probability distributions P_u on $\mathbb{U}_{\alpha,\Delta}$ and P_x on $\mathcal{X}$, respectively. These distributions define the initial conditions and input signals of interest.

Remark 1. Through the mapping $\mathfrak{C}_{\alpha,\Delta}$ in (2.3), the distribution P_u induces a distribution on sequences of parameters (ω_k). In practice we do not sample a signal u on $\mathbb{R}_{\geq 0}$, but only the restriction $u|_{[0,T)}$ of u to a finite interval of the form $[0,T)$. Thus, we are interested in the probability $P_u(A)$ for sets $A \subset \mathbb{U}_{\alpha,\Delta}$ which satisfy

$$u \in A \text{ and } v|_{[0,T)} = u|_{[0,T)} \implies v \in A, \tag{2.6}$$

i.e. whether $u \in A$ depends only on the values of $u(t)$ for $t \in [0, T)$. Then we can write

$$\begin{aligned} P_u(A) &= P_u\Big(\Big\{u : u = \mathfrak{C}_{\alpha,\Delta}\Big[(\omega_k)_{k\in\mathbb{Z}_{\geq 0}}\Big] \in A, (\omega_k)_{k\in\mathbb{Z}_{\geq 0}} \in \Omega_p\Big\}\Big) \\ &= P_u\Big(\Big\{\mathfrak{C}_{\alpha,\Delta}\Big[(\omega_k)_{k\in\mathbb{Z}_{\geq 0}}\Big] : (\omega_k)_{k\in\mathbb{Z}_{\geq 0}} \in \mathfrak{C}_{\alpha,\Delta}^{-1}(A)\Big\}\Big). \end{aligned}$$

Note that (2.6) and (2.3) imply

$$(\omega_k)_{k\in\mathbb{Z}_{\geq 0}} \in \mathfrak{C}_{\alpha,\Delta}^{-1}(A) \text{ and } \omega_k' = \omega_k,\ k = 0, \ldots, k_T \implies (\omega_k')_{k\in\mathbb{Z}_{\geq 0}} \in \mathfrak{C}_{\alpha,\Delta}^{-1}(A)$$

(recall that $k_T = \lfloor T/\Delta \rfloor$). Thus, letting $\Pi_j : \Omega_p \to \mathbb{R}^{j+1}$ denote the projection on coordinates $0, \ldots, j$, that is,

$$\Pi_j\big((\omega_k)_{k\in\mathbb{Z}_{\geq 0}}\big) = (\omega_0, \ldots, \omega_j),$$

we may write

$$(\omega_k)_{k\in\mathbb{Z}_{\geq 0}} \in \mathfrak{C}_{\alpha,\Delta}^{-1}(A) \iff (\omega_0, \ldots, \omega_{k_T}) \in \Pi_{k_T}\Big(\mathfrak{C}_{\alpha,\Delta}^{-1}(A)\Big).$$

Hence, we can define a distribution on $(\omega_0, \ldots, \omega_{k_T})$ as

$$P_\omega^{(k_T)}(S) = P_u\Big(\Big\{\mathfrak{C}_{\alpha,\Delta}\Big[(\omega_k)_{k\in\mathbb{Z}_{\geq 0}}\Big] : (\omega_0, \ldots, \omega_{k_T}) \in S\Big\}\Big),$$

and sampling from $P_\omega^{(k_T)}$ is equivalent to sampling inputs on $[0, T)$ from P_u.

In practice, one can often directly define a distribution on sequences of parameters ω_k and let P_u be defined through $\mathfrak{C}_{\alpha,\Delta}$. As we have shown here this entails no loss of generality, but for simplicity of notation we take the abstract view and work with P_u directly.

Remark 2. If we do not require that x and u in (2.5) are independent, the approach accommodates the case where the inputs are given by a feedback law. To be precise, consider the case where the inputs are given by a sampled feedback law of the form $u(t) = \kappa(x((k-1)\Delta), t)$ for $t \in [k\Delta, (k+1)\Delta)$, where κ is Δ-periodic in t. Denote by $\varphi_\kappa : \mathcal{T} \times \mathcal{X} \to \mathcal{X}$ the (autonomous) flow function of the system under this feedback law. For each initial condition x, define the input signal

$$u_x(t) = \kappa(\varphi_\kappa(k\Delta, x), t),\ t \in [k\Delta, (k+1)\Delta).$$

Then $u_x \in \mathbb{U}_{\alpha,\Delta}$, where α in (2.3) is given by $\alpha(z, t) = \kappa(z, t\Delta)$. The control u_x is parameterised by the sequence $z_k := \varphi_\kappa(k\Delta, x)$, $k \in \mathbb{Z}_{\geq 0}$, and we have

$$\varphi_\kappa(t, x) = \varphi(t, x, u_x),\ t \in \mathcal{T},\ x \in \mathcal{X},$$

so that sampling (x, u) in (2.5) with $x \sim P_x$ and $u = u_x$ gives the distribution of controls corresponding to the feedback law κ, as desired. We will focus here on open loop inputs.

We propose to find an approximation of φ by minimising the loss function ℓ_T over $\mathcal{H}$. Let $L_T(\hat{\varphi}, x, u)$ denote the loss for a single trajectory:

$$L_T(\hat{\varphi}, x, u) := \frac{1}{T} \int_0^T \|\hat{\varphi}(t, x, u) - \varphi(t, x, u)\|^2 \,\mathrm{d}t, \tag{2.7}$$

so that $\ell_T(\hat{\varphi}) = \mathbf{E}\, L_T(\hat{\varphi}, x, u)$. If we measure N trajectories $\varphi(\cdot, x^i, u^i)$, $i = 1, \ldots, N$, where (x^i, u^i) are independent draws from $P_x \times P_u$, we can minimise the empirical mean estimate of ℓ_T:

$$\ell_T(\hat{\varphi}) \approx \frac{1}{N} \sum_{i=1}^{N} L_T(\hat{\varphi}, x^i, u^i).$$

In practice, the data consists of discrete-time samples of the trajectories:

$$\xi_k^i = \varphi(t_k^i, x^i, u^i) + v_k^i,\ k = 1, \ldots, K,\ i = 1, \ldots, N, \tag{2.8}$$

where K is the number of samples of each trajectory, $t_k^i \in [0, T]$ is an increasing sequence of time samples and v_k^i is measurement noise. As

$$\frac{1}{K} \sum_{k=1}^{K} \left\|\xi_k^i - \hat{\varphi}(t_k^i, x^i, u^i)\right\|^2 \approx L_T(\hat{\varphi}, x^i, u^i),$$

we define the empirical loss function $\hat{\ell}_T$ as

$$\hat{\ell}_T(\hat{\varphi}) := \frac{1}{N} \sum_{i=1}^{N} \frac{1}{K} \sum_{k=1}^{K} \left\|\xi_k^i - \hat{\varphi}(t_k^i, x^i, u^i)\right\|^2. \tag{2.9}$$

Remark 3. There is no assumption on the structure of the sequences $\left\{t_k^i\right\}_{k=1}^K$, but not all samples should coincide with the control sample instants, so that the inter-sample behaviour can be captured.

Our objective is to define a hypothesis space $\mathcal{H}$ which renders the problem of minimising $\hat{\ell}_T$ tractable. Furthermore, we would like to obtain a good approximation of the flow function while preserving at least some of its properties.

2.3 Methodology

In this section we propose a neural network architecture appropriate for solving the problem described above. We first show how the flow function of a system with inputs in $\mathbb{U}_{\alpha,\Delta}$ can be exactly represented by a discrete-time dynamical system. Thereafter, we exploit this representation to design the architecture, and discuss some of its system-theoretic properties.

A discrete-time representation of the flow function

Let $u \in \mathbb{U}_{\alpha,\Delta}$ be such that $u = \mathfrak{C}_{\alpha,\Delta}\Big[(\omega_k)_{k\in\mathbb{Z}_{\geq 0}}\Big]$. Due to the causality of φ and the structure of $\mathbb{U}_{\alpha,\Delta}$, the flow $\varphi(s,x,u)$ at a time instant $s \geq 0$ depends only on the finite sequence of parameters $(\omega_k)_{k=0}^{k_s}$, where as before $k_s = \lfloor s/\Delta \rfloor$. In particular, for $s \in [0,\Delta)$,

$$\varphi(s,x,u) = \varphi(s,x,u_{\omega_0}),$$

where $u_{\omega_0}(t) := \alpha(\omega_0, \frac{t}{\Delta})$. Let us thus define

$$\begin{aligned} \Phi &: [0,1] \times \mathcal{X} \times \mathbb{R}^p \to \mathcal{X} \\ \Phi(\tau,x,\omega) &:= \varphi(\tau\Delta, x, u_\omega). \end{aligned}$$

The function Φ allows us to compute the flow φ on a single control period, but unlike φ it has a finite dimensional domain. Define also the map $d_\Delta : s \mapsto (\tau_k)_{k=0}^{k_s}$, where

$$\tau_k = \begin{cases} 1, & k < k_s \\ \dfrac{s - k_s\Delta}{\Delta}, & k = k_s \end{cases},$$

that is, τ_i is the relative amount of time that the input parameter ω_i is in effect. As illustrated in Figure 2.2a, for an arbitrary time instant $s > 0$, we can compute the value of $\varphi(s,x,u)$ by iterating Φ as follows:

$$\begin{aligned} x_0 &= x, \\ x_{k+1} &= \Phi(\tau_k, x_k, \omega_k), \; k \leq k_s \end{aligned} \tag{2.10}$$

so that $x_{k_s+1} = \varphi(s,x,u)$, by the semigroup property. Let $\mathcal{F}$ map the initial state and input sequences to the final state of the dynamical system (2.10), i.e.

$$\mathcal{F}(x_0, (\tau_k)_{k=0}^n, (\omega_k)_{k=0}^n) := x_{n+1}. \tag{2.11}$$

We can then write

$$\varphi(s,x,\mathfrak{C}_{\alpha,\Delta}[(\omega_k)]) = \mathcal{F}\Big(x_0, d_\Delta(s), (\omega_k)_{k=0}^{k_s}\Big) \tag{2.12}$$

for all $s \geq 0$, $x \in \mathcal{X}$ and $(\omega_k)_{k\in\mathbb{Z}_{\geq 0}} \in \Omega_p$. That is, trajectories of φ can be equivalently represented by the trajectories of a discrete-time dynamical system with inputs (τ_k, ω_k).

Proposed architecture

The representation of φ as a discrete-time state-space dynamical system in (2.12) motivates the use of recurrent neural network (RNN) models to approximate $\mathcal{F}$, as they are universal approximators of this type of mapping (Schäfer and Zimmermann, 2006). An RNN model has the same structure as the mapping $\mathcal{F}$, where Φ is replaced by a parametric function defined by the RNN architecture and x_i is the hidden state sequence.

However, the map Φ is in general complex and likely difficult to approximate directly. Furthermore, directly approximating (2.10) by an RNN model would imply that the hidden state dimension is fixed and equal to the dimension of $\mathcal{X}$. To increase the flexibility of the model, we first map the initial state to a feature space $\mathcal{Z}$ using a deep neural network (DNN) called the encoder network. Denote by h_{enc} the input-output map of the network. Letting f_{RNN} be the one-step map of the RNN, we thus have

$$\begin{aligned} z_0 &= h_{\text{enc}}(x) \\ z_{k+1} &= f_{\text{RNN}}(z_k, \tau_k, \omega_k),\ \ k = 0, \dots, k_t, \end{aligned}$$

where $z_i \in \mathcal{Z}$ are the hidden states of the RNN. Define the corresponding sequence-to-sequence map

$$(z_0, z_1, \dots, z_{k_t+1}) = h_{\text{RNN}}\Big(z_0, (\tau_k)_{k=0}^{k_t}, (\omega_k)_{k=0}^{k_t}\Big).$$

In order to ensure that the flow approximation is continuous in time, we let the output $\tilde{z}$ of the RNN be given by a combination of the last two states as follows:

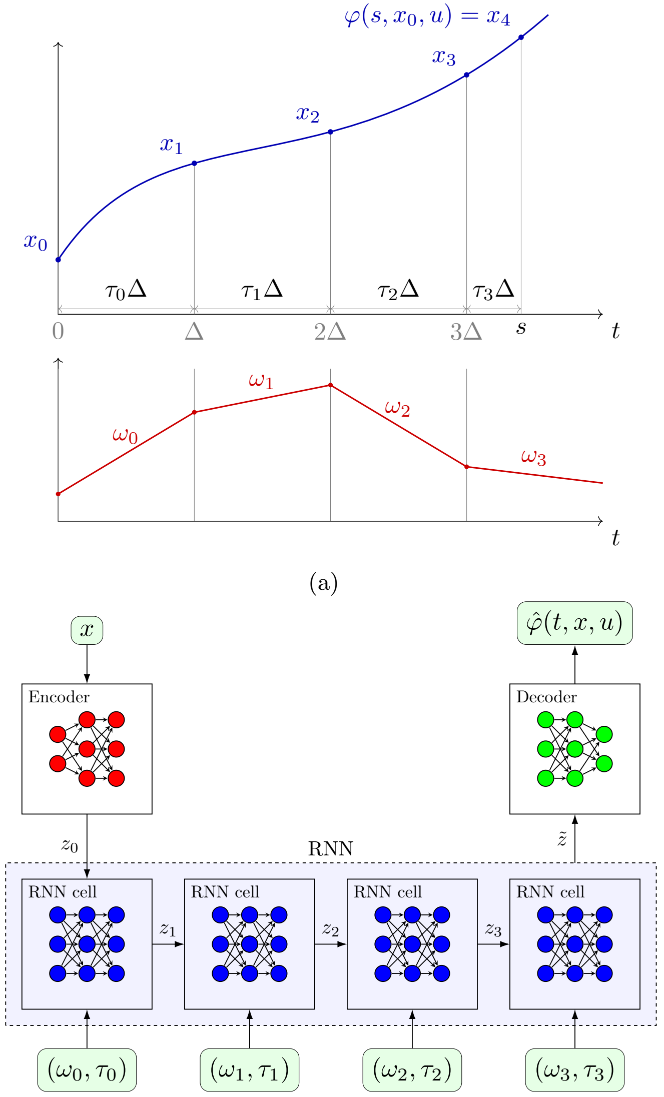
$$\tilde{z} = h_{\text{int}}\Big((\tau_k)_{k=0}^{k_t}, (z_k)_{k=0}^{k_t+1}\Big) := (1 - \tau_{k_t}) z_{k_t} + \tau_{k_t} z_{k_t+1} \tag{2.13}$$

Note that the mapping h_{int} does not amount to a linear interpolation of the states since z_{k_t+1} depends on τ_{k_t}. Because $\varphi(\cdot, x, u)$ is a continuous function, we want to enforce that if τ_{k_t} is small then the output $\tilde{z}$ is close to z_{k_t}, and this is achieved by (2.13).

To map the output $\tilde{z}$ back to a state vector in $\mathcal{X}$ we use another DNN, the decoder network, whose input-output map we denote by h_{dec}. As shown in Figure 2.2b, the application of h_{dec} to $\hat{z}$ yields the approximated flow $\hat{\varphi}$ at the time instant t, i.e.

$$\begin{aligned} \hat{\varphi}(t, x, \mathfrak{C}_{\alpha,\Delta}[(\omega_k)]) &= h_{\text{dec}}(\tilde{z}) \\ &= h_{\text{dec}}\Big(h_{\text{int}}\Big(d_\Delta(t), h_{\text{RNN}}\Big(h_{\text{enc}}(x), d_\Delta(t), (\omega_k)_{k=0}^{k_t}\Big)\Big)\Big). \end{aligned} \tag{2.14}$$

We thus take our hypothesis space to be the set of functions $\hat{\varphi}$ defined in this way for different values of the parameters of the RNN and the encoder and decoder networks.


$\varphi(s, x_0, u) = x_4$
x_0
x_1
x_2
x_3
$\tau_0\Delta$
$\tau_1\Delta$
$\tau_2\Delta$
$\tau_3\Delta$
0
Δ
2Δ
3Δ
s
t
ω_0
ω_1
ω_2
ω_3
t
x
Encoder
z_0
RNN
RNN cell
z_1
RNN cell
z_2
RNN cell
z_3
RNN cell
$\tilde{z}$
Decoder
$\hat{\varphi}(t, x, u)$
(ω_0, τ_0)
(ω_1, τ_1)
(ω_2, τ_2)
(ω_3, τ_3)

(a)

(b)

Figure 2.2: (a) Schematic illustration of the true flow function φ for the input plotted in red parameterised by $\{\omega_k, \tau_k\}_{k=0}^{3}$. (b) Corresponding model for the approximated flow $\hat{\varphi}$. In the approximated model, we first map the initial condition x to the encoded state $z_0 \in \mathcal{Z}$ through a feedforward encoder network. Then, the encoded state is propagated in time through an RNN. Each cell of the RNN sequentially takes (ω_k, τ_k) as inputs. The two last hidden states are interpolated and mapped back to $\mathcal{X}$ through another feedforward decoder network.

Another interpretation of the architecture may be given as follows. The RNN together with (2.13) can be seen as the flow of the continuous-time system

$$\dot{\zeta}(s) = f_{\mathrm{RNN}}(v(s), \tau(s), \omega(s)) - v(s)$$

under discrete-time feedback

$$v(s) = \zeta(k),\ s \in [k, k+1),\ k \geq 0,$$

and with the piecewise-constant inputs

$$\tau(s) = \tau_k,\ \omega(s) = \omega_k,\ s \in [k, k+1).$$

With $\zeta(0) = h_{\mathrm{enc}}(x)$, we have that the sequence of hidden states is given by $z_k = \zeta(k)$, $k \leq k_t$, and $z_{k_t+1} = \zeta(k_t + \tau_{k_t}) = \zeta(t/\Delta)$. Hence, letting ψ denote the flow of this system, we may write

$$\hat{\varphi}(t, x, \mathfrak{C}_{\alpha,\Delta}[(\omega_k)]) = h_{\mathrm{dec}}(\psi(t/\Delta, h_{\mathrm{enc}}(x), \tau, \omega)).$$

Properties

At this point, we would like to briefly discuss some properties of the architecture just described, starting with the properties of the flow function considered in Section 2.2. Concerning the identity property, we have

$$\hat{\varphi}(0, x, u) = h_{\mathrm{dec}}(h_{\mathrm{enc}}(x)),$$

so that the identity property of the flow function will hold if the decoder is a left inverse of the encoder. This is in general hard to enforce, but implies the necessary condition that the space $\mathcal{Z}$ be of greater dimension than the state space $\mathcal{X}$.

Although the semigroup property is is general not satisfied, causality as formulated in Section 2.2 is seen to hold. Indeed, by (2.14), the value of $\hat{\varphi}$ at time t depends only on the control parameters ω_k for $k = 0, \ldots, k_t$, or, equivalently, on the input $u(s)$ for $s \in [0, t)$.

From (2.13), we see that $\hat{\varphi}(\cdot, x, u)$ is continuous in time, but in general not differentiable at $t = k\Delta$, $k \in \mathbb{Z}_{\geq 0}$. This concurs with a case of practical interest: suppose that φ is the flow of a system defined by a controlled ordinary differential equation

$$\dot{\xi}(t) = f(\xi(t), u(t)),\ \xi(0) = x,$$

so that $\xi(t) = \varphi(t, x, u),\ t \geq 0$. Assuming for simplicity that f is smooth, the state trajectory ξ then possesses one more degree of smoothness than the input u (Sontag, 1998, Proposition C.3.11). For instance, if u is piecewise constant, ξ is in general not differentiable everywhere (only absolutely continuous). Thus, in this case it is appropriate to enforce only continuity in time of $\hat{\varphi}$, as opposed to a higher degree of smoothness. Note in addition that for this special case, if $\alpha(\cdot, \omega)$ is continuous on $[0, 1)$, then ξ is differentiable on each interval $(k\Delta, (k+1)\Delta)$ and differentiable from the right at $t = k\Delta$, $k \in \mathbb{Z}_{\geq 0}$, which also holds for $\hat{\varphi}$.

In Chapter 4 we formulate and prove an universal approximation property for this architecture.

2.4 Training procedure

In this section we discuss the training procedure for the proposed architecture. This procedure will be used in the Sections 2.5 and 2.6 for the experimental evaluation of the architecture.

Data generation

We consider piecewise-constant control inputs, i.e. we take $\alpha(\omega, t) = \omega$ for $t \in \mathbb{R}$, $\omega \in \mathbb{R}^p$. Recall that the data is given by a collection of measurements from trajectories of the system as in (2.8). In all experiments below, the data is generated by numerically integrating the dynamics of the system under consideration. The measurement time instants t_k^i are generated using Latin Hypercube sampling. The collected trajectories are divided into train, validation and test sets using a random 60-20-20% split. When training different models on the same data, the dataset is always split in the same way. When measurement noise is considered, it is added to the train and validation trajectory datasets only.

Network architecture

We use a long short-term memory (LSTM) network to realise the map h_{RNN}. The encoder and decoder networks are standard feedforward neural networks with tanh activations.

The state of the LSTM cells has two components, the cell state and the hidden state. In these experiments we set the initial cell state to zero, and only the initial hidden state is learned by the encoder network. Similarly, the decoder takes the final hidden state as an input.

Training process

All models are trained using the Adam optimisation algorithm with a batch size of 512 on a cluster node with an NVIDIA T4 GPU and an Intel® Xeon® Gold 6226R CPU @ 2.90GHz. We will refer to the empirical mean square loss (2.9) as the train, validation, and test loss according to the data set from which the measurements used to compute the loss are taken from. The learning rate is decreased by a factor of 10 whenever the validation loss does not decrease for 5 consecutive epochs, and the training is interrupted if the validation loss does not decrease more than 1×10^{-6} for 15 consecutive decades.

2.5 Numerical evaluation: Van der Pol oscillator

In this section we consider the Van der Pol oscillator with external input. While having low state dimension and being well-understood, making it an attractive candidate to explore the capabilities of the architecture, this system exhibits rich and complex behaviour (Guckenheimer, Hoffman and Weckesser, 2003; Bold et al., 2003). We begin by examining the selection of the architecture hyperparameters and their impact on the loss value. This is followed by a study of the simulation performance of the system under the training distribution as well as for long time horizons and a different distribution of the input signals. Finally, we investigate the sensitivity of the training algorithm with respect to the measurement noise, as well as the influence of the number of training samples on the simulation performance.

The system is described by the system of ordinary differential equations

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= -x_1(t) + \left(1 - x_1(t)^2\right) \mu x_2(t) + u(t). \end{aligned} \tag{2.15}$$

We take $\mu = 1$ and $x(0) \sim N(0, I)$, i.e. a standard normal distribution. The control input sampling time is $\Delta = 0.2$ and the inputs considered are square wave inputs with period 5Δ and amplitudes sampled i.i.d. from $N(0, \sigma = 5)$, i.e.

$$\begin{aligned} \omega_{1+5k} &\sim N(0, 5), \\ \omega_{j+5k} &= \omega_{1+5k}, \ j = 2, 3, 4, 5 \end{aligned}$$

holds for all $k \geq 0$.

We integrate (2.15) over the time interval $[0, 15]$ with a RK45 solver. A total of $N = 300$ trajectories are generated and, for each trajectory, $K = 200$ time points t_k^i are sampled. The measurement noise in (2.8) is zero-mean Gaussian noise with standard deviation of 0.1.

Hyperparameter selection

We begin by investigating the choice of some of the architecture hyperparameters. Figure 2.3 shows the validation loss as a function of the learning rate and the size of the LSTM's hidden state. For each of the shown values of the hyperparameters, 20 models were trained on the same data. In all cases, the encoder and decoder have 2 hidden layers with depth equal to the number of hidden states of the LSTM, and a single-layer LSTM is used.

The results indicate that an LSTM with 24 hidden states and a learning rate of 1×10^{-3} are optimal for this system, attaining rather low validation loss in the order of 1×10^{-2}. Note that, for all considered choices of the hyperparameter values, the validation loss values are of the same order of magnitude as the noise values, and two orders of magnitude smaller than the amplitude of the state trajectories. Thus, the proposed training procedure is able to minimise the empirical training loss robustly with respect to the initial choice of the network parameters, and to variations in the hyperparameters in the ranges here considered.

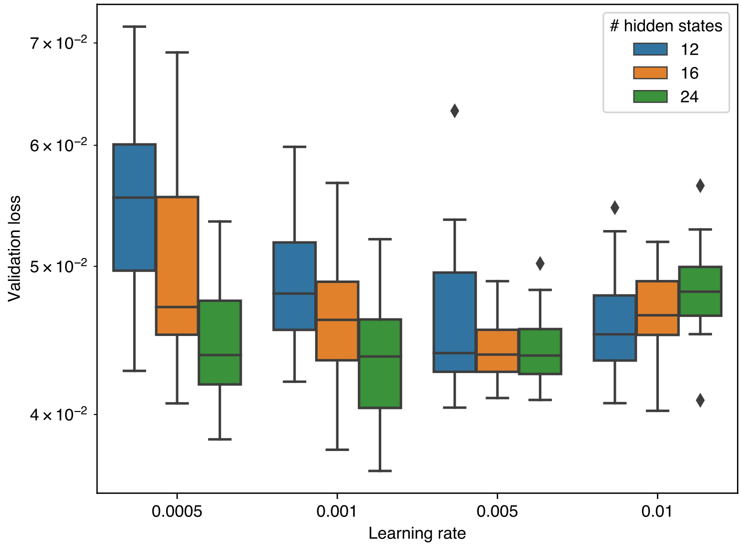


Figure 2.3: Validation loss as a function of learning rate and hidden state size for the Van der Pol system

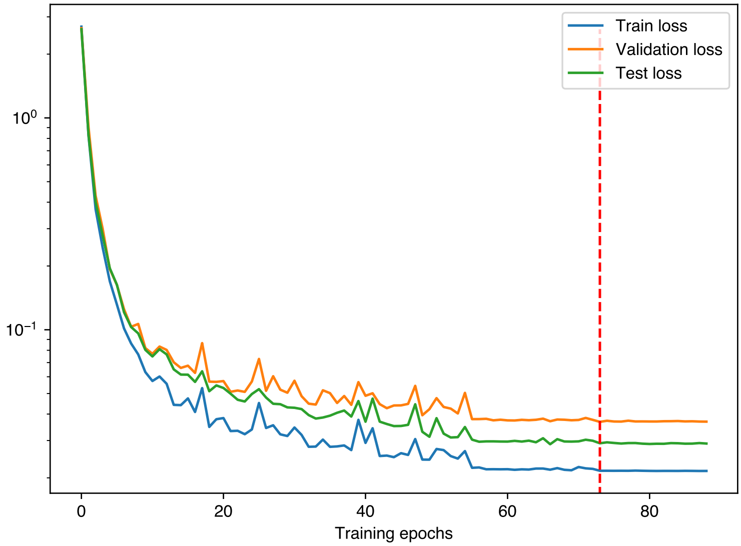


Figure 2.4: Evolution of the train, validation and test losses during training. The red line indicates the last epoch where there was an improvement in the validation loss.

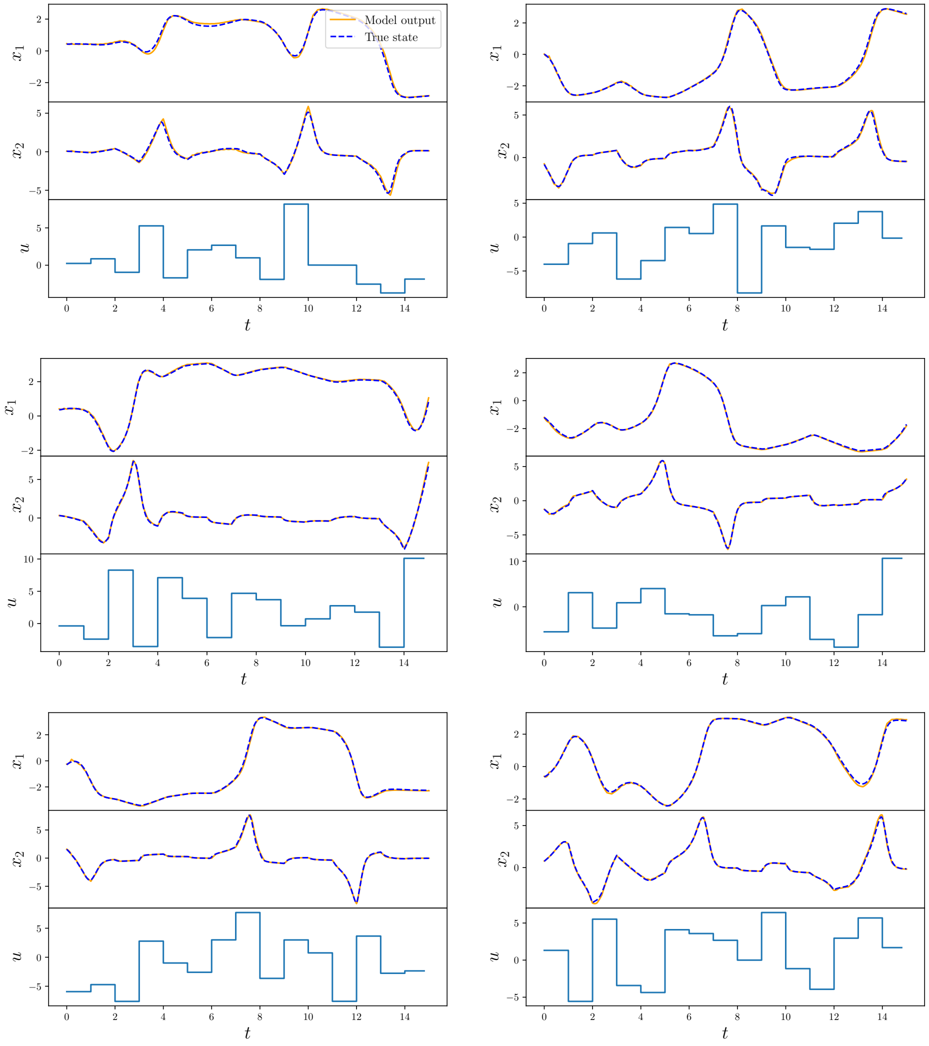


Figure 2.5: Actual (blue, dashed) and simulated (orange) trajectories of the Van der Pol model with initial conditions and inputs drawn from the corresponding distributions.

Simulation performance

Among the models considered above, we select that which exhibits the lowest value of the validation loss. The selected model thus has a validation loss of approximately 3.7×10^{-2} and test loss approximately equal to 2.9×10^{-2} (recall that the test dataset is noiseless). The corresponding training time is just under 182 seconds.

Figure 2.4 shows the evolution of the three loss functions during training, showing that convergence is obtained after about 75 epochs with a small gap between the test and train losses. The red line shows the last epoch after which no improvement was detected in the validation loss, so that the model obtained at this epoch is saved and regarded as the optimum.

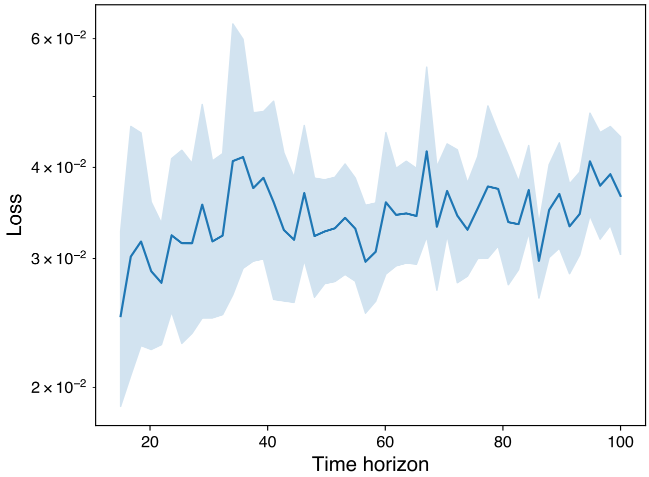
Figure 2.5 shows the output of the model for 6 new (i.e. not seen during training) inputs and initial conditions drawn from the same distributions P_x and P_u used to generate the data, where it is seen that the model can faithfully reproduce the true state trajectory of the system.

Long-time simulation performance

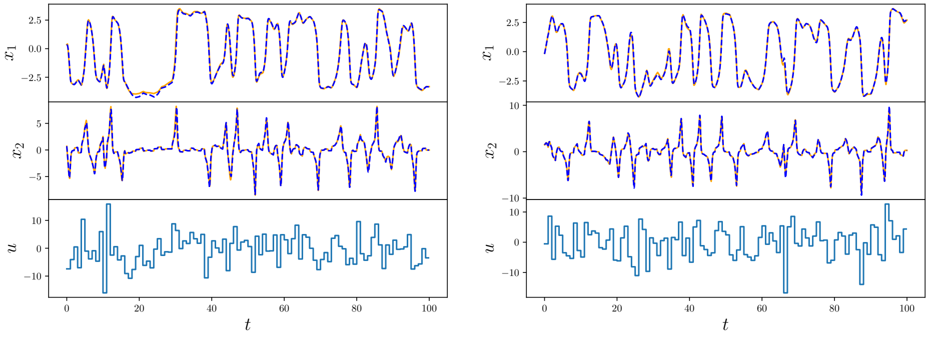
Because of the recurrent and time-invariant structure of the architecture and the stability of the Van der Pol oscillator, we expect that the model considered in the previous subsection exhibit good simulation performance even for times larger than the length T of the trajectories used for training the model.

Suppose we have trained a model $\hat{\varphi}_T$ with data from trajectories on $[0, T]$. In order to assess how the trained model generalises to $t > T$ we can investigate how the loss $\ell_t(\hat{\varphi}_T)$ grows as t grows larger. To do this, we generate new sets of trajectories on $[0, t]$ for different values of $t \geq T$ and estimate the mean and variance of the single-trajectory loss $L_t(\hat{\varphi}_T, x, u)$ as defined in (2.7), with $x \sim P_x$ and $u \sim P_u$, as before.

In Figure 2.6a the estimate of $\ell_t(\hat{\varphi}_T) = \mathbf{E}\ L_t(\hat{\varphi}_T)$ as a function of t for the model $\hat{\varphi}_T$ considered in the previous subsection (with $T = 15$) is shown, with the coloured area representing the 95% confidence interval approximated using the empirical variance of $L_t(\hat{\varphi}_T)$. We observe that ℓ_t remains approximately constant as t increases, indicating that the model gives reliable predictions for t much larger than the value of T used for the training trajectories, as expected. This is further confirmed in Figure 2.6b, where two simulated trajectories on the time interval $[0, 100]$ are shown, and we observe that they closely follow the true state trajectory.



(a)



(b)

Figure 2.6: (a) Estimate of ℓ_t as a function of t for the Van der Pol oscillator; (b) actual (blue, dashed) and simulated (orange) trajectories for the Van der Pol model on the time interval $[0, 100]$

Changing the input distribution

In order to evaluate the generalisation of the model with respect to the input distribution P_u, we choose another distribution Q_u and compute the empirical loss (2.9) with respect to this distribution. We consider here the distribution Q_u on sinusoidal inputs with random amplitude and frequency passed through a zero-order hold, so that

$$\omega_k = A\text{sin}(\Omega k),$$

where $A \sim \text{LogNormal}(0, 1)$ (i.e. $A = \exp \nu$, where ν is a standard normal random variable) and $\Omega \sim \text{Uniform}\big(0, \frac{2\pi}{10}\big)$. Equivalently,

$$u(t) = A\text{sin}\left(\Omega \left\lfloor \frac{t}{\Delta} \right\rfloor\right),$$

corresponding to signals with maximum frequency $\frac{1}{10\Delta}$ Hz, i.e. 20% of the Nyquist frequency. Figure 2.7 shows the distribution of the single-trajectory loss L_T when the input is distributed according to P_u and Q_u. As expected, the loss increases on average and is less concentrated but remains comparable with the loss values obtained from P_u. This is illustrated in Figure 2.8 which shows four predicted trajectories with inputs drawn from Q_u over the time interval $[0, 30]$.

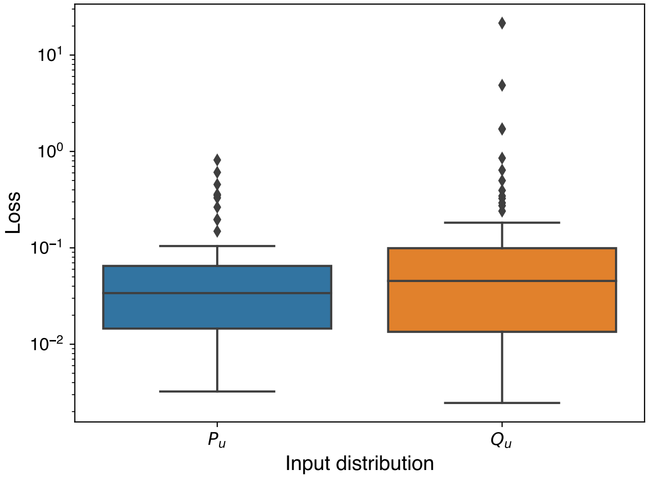


Figure 2.7: Distribution of L_T with the training input distribution (P_u) and a different input distribution (Q_u).

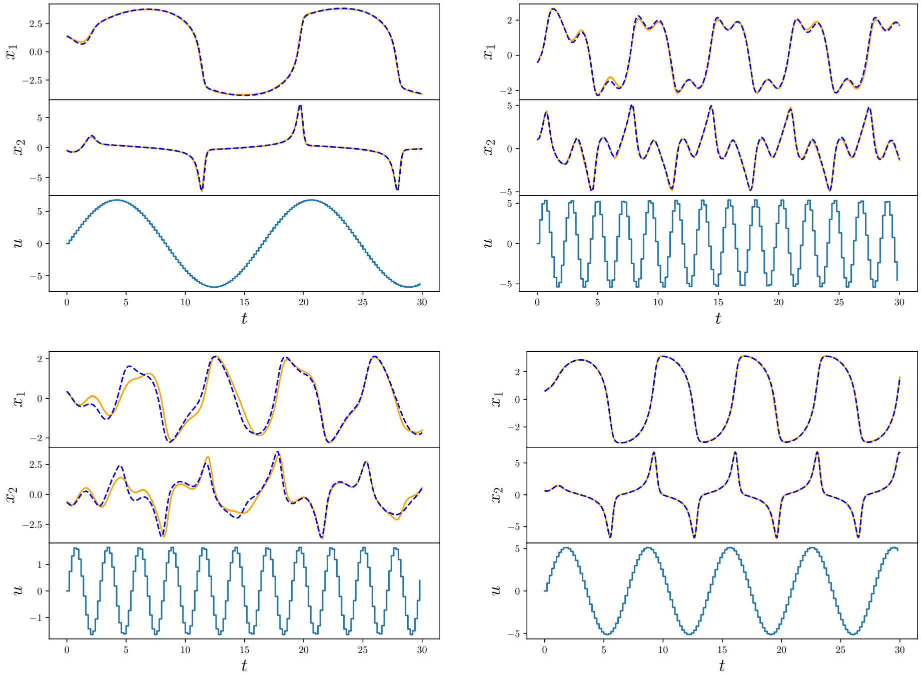


Figure 2.8: Real and predicted trajectories for the Van der Pol model with sinusoidal inputs.

Sensitivity with respect to measurement noise

We study the dependence of the test loss $\hat{\ell}_T$ on the measurement noise. Assume the v_k^i in (2.8) to be i.i.d. zero-mean random variables with covariance $\sigma^2 I$, and let $\hat{\varphi}_\sigma$ be a model trained with the data (2.8). We would like to understand how the performance of the model deteriorates as σ increases. Let $\delta_\sigma(t,x,u) = \varphi(t,x,u) - \hat{\varphi}_\sigma(t,x,u)$. Taking an expectation with respect to test data (x^i, u^i) and ξ_k^i,

$$\begin{aligned}
\mathbf{E}\ \hat{\ell}_T(\hat{\varphi}_\sigma) &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\mathbf{E}\left\|\delta_\sigma(t_k^i, x^i, u^i) + v_k^i\right\|^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\mathbf{E}\left[\left\|\delta_\sigma(t_k^i, x^i, u^i)\right\|^2 + 2\delta_\sigma(t_k^i, x^i, u^i)^T v_k^i + \left\|v_k^i\right\|^2\right] \\
&= \sigma^2 + \mathbf{E}_{x,u}\frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\left\|\delta_\sigma(t_k^i, x, u)\right\|^2 \\
&\approx \sigma^2 + \ell_T(\hat{\varphi}_\sigma).
\end{aligned}$$

The first term is independent of the hypothesis class and the learning algorithm, so the true model performance is captured by the second term.

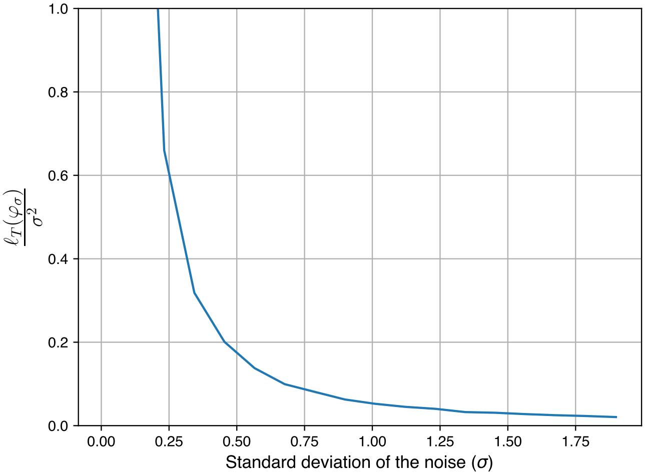


Figure 2.9: Sensitivity of the training algorithm with respect to noise.

In order to estimate the value of $\ell_T(\hat{\varphi}_\sigma)$ we can train models on the same data injected with different noise levels, and test these different models on a single noiseless dataset. In Figure 2.9, we plot an estimate of $\sigma^{-2}\ell_T(\hat{\varphi}_\sigma)$ as a function of σ. For each value of σ, 20 models $\hat{\varphi}_\sigma$ are trained on the dataset described in the beginning of this section, where the noise is resampled with $v_k^i \sim N(0, \sigma)$ (a different noise realisation is used for each model). The result shows that the performance on the test data is robust with respect to the measurement noise.

Impact of the amount of measurements

Finally, we study the dependence of the loss on the number of trajectories in the training data set and the number of samples per trajectory. To this end, we generate datasets as in (2.8), with $v_k^i = 0$, for different values of N and K, and train 20 models for each pair of (N, K) values. All models are tested on a single noiseless test dataset with 400 trajectories and 800 samples per trajectory, and the numbers shown in Figure 2.10 indicate the loss of of the best model in each group on this dataset. Note that the values in the vertical axis correspond to the total number N of collected trajectories, so that the number of trajectories in each training dataset is equal to 60% of the shown values.

As expected, we observe a trade-off between the number of trajectories and the number of samples, but we see also that the number of trajectories N has a significantly larger influence on the test loss than the number of samples. For instance, datasets corresponding to the pairs of values $(N, K) = (400, 100)$ and

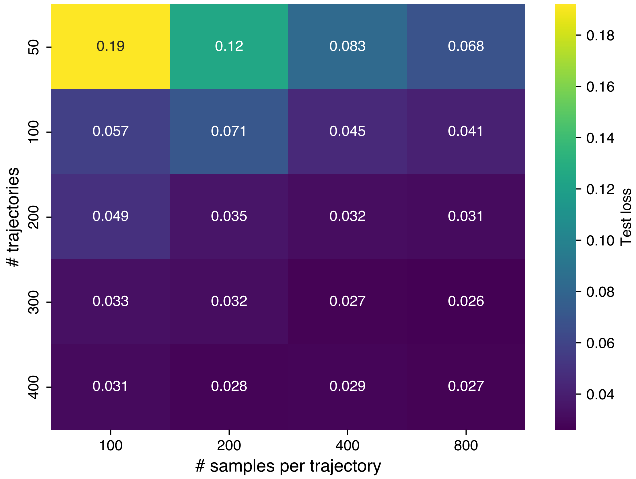


Figure 2.10: Test loss as a function of the number of trajectories N and the number of samples K.

$(50, 800)$ contain the same number of samples, but the best model trained on the latter dataset has a test loss of more than double that of the best model trained on the former. This is partly to be expected, since the initial condition is one of the inputs of the flow function.

2.6 Numerical evaluation: FitzHugh-Nagumo oscillator

The second system under study is the FitzHugh-Nagumo oscillator, whose trajectories either converge to a stable equilibrium or to a limit cycle, depending on the input amplitude. This phenomenon is termed excitability (FitzHugh, 1961). We investigate whether our architecture is capable of capturing this behaviour.

The FitzHugh-Nagumo oscillator is described by the following system of nonlinear differential equations:

$$\begin{aligned} \eta \dot{x}_1(t) &= x_1(t) - x_1(t)^3 - x_2(t) + u(t) \\ \eta \tau \dot{x}_2(t) &= x_1(t) + a - b x_2(t), \end{aligned} \tag{2.16}$$

where η, τ, a and b are positive constants, which we choose as $\eta = 1/50$, $\tau = 40$, $a = 0.3$, $b = 1.4$.

As before, we take $x(0) \sim N(0, I)$. The control period is $\Delta = 0.1$ and the input distribution P_u is given by

$$\begin{aligned} &\omega_{1+40k} \overset{\text{i.i.d.}}{\sim} \text{LogNormal}(\mu = \log(0.2), \sigma = 0.5), \\ &\omega_{j+40k} = \omega_{1+40k}, \ j = 2, \ldots, 40 \end{aligned}$$

for all $k \geq 0$. This is chosen so that the excitable behaviour of the oscillator is observed, as with the given parameters, the system enters into a limit cycle for a narrow band of input amplitudes around $u = 0.2$.

We generated $N = 300$ trajectories on $[0, 20]$ using a backward differentiation formula solver, sampling $K = 300$ time points from each trajectory using Latin hypercube sampling. The measurement noise in (2.8) is zero-mean Gaussian noise with standard deviation equal to 0.05.

Running a search over the architecture hyperparameters in the same way as described in the previous section, we arrive at the result shown in Figure 2.11. The optimal model has 24 hidden states and is trained with a learning rate of 1×10^{-2}, but we observe that the loss values are two orders of magnitude smaller than the amplitude of the state trajectories over the entire hyperparameter range. As before, for all models the encoder and decoder have 2 hidden layers with 24 nodes each. The optimal model took 598 seconds to train.

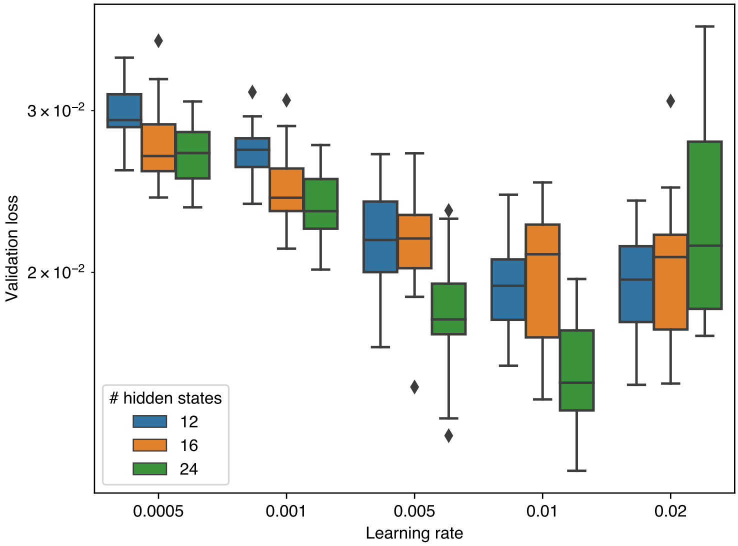
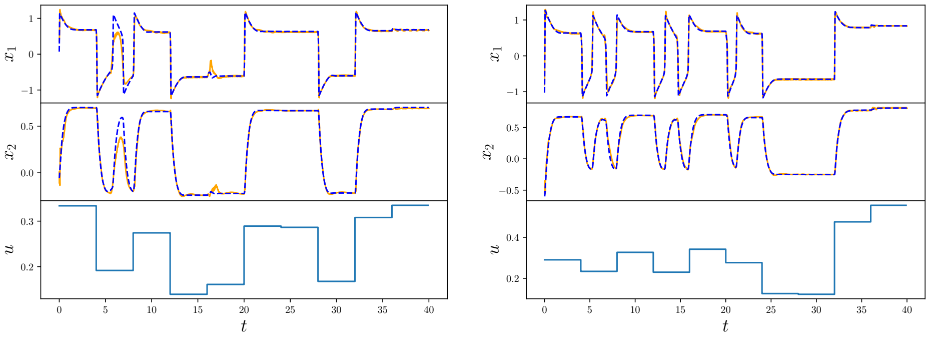


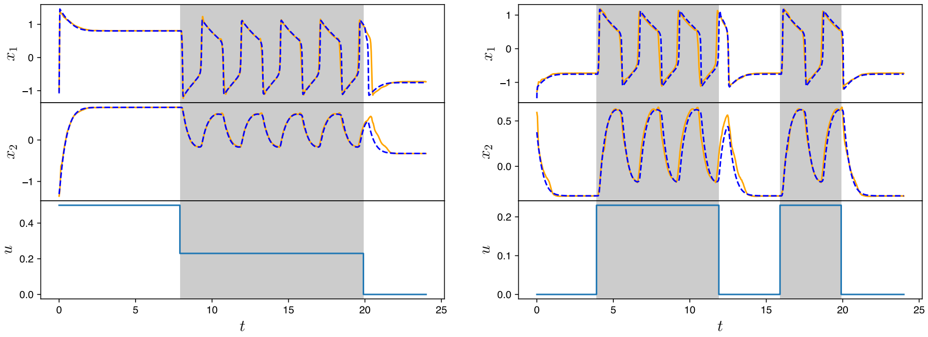
Figure 2.11: Validation loss as a function of learning rate and hidden state size for the FitzHugh-Nagumo system

Figure 2.12 shows two predicted trajectories on $[0, 40]$ for two new pairs of initial conditions and inputs drawn from P_x and P_u (i.e. unseen during training), and we can see that the model is able to faithfully reproduce the state trajectories of the system. Additionally, as we can observe in Figure 2.12a at about $t = 5$, although the trained model may fail to predict the peak value of the oscillations, it is able to recover from the error.



(a) (b)

Figure 2.12: Actual (blue, dashed) and simulated (orange) trajectories of the FitzHugh-Nagumo model with initial conditions and inputs drawn from the corresponding distributions.



(a) (b)

Figure 2.13: In (a) and (b), actual (blue, dashed) and predicted (black) trajectories demonstrate two distinct cases of excitability for the FitzHugh-Nagumo oscillator. Excitable regions are shaded in grey.

Finally, Figure 2.13 shows the state trajectories for two inputs chosen to exhibit the transition between the resting and excitable modes of the oscillator. In Figure 2.13a, we see that the learned model correctly predicts that, as the amplitude of the input is gradually decreased over time, the oscillator's response traverses from a higher resting state (constant response), passes through the excitable region (periodic spike train), and returns to a lower resting potential (constant response). In Figure 2.13b, the model correctly simulates the occurrence of two distinct excitable regions with two sets of spike trains.

2.7 Summary

We presented a recurrent neural network architecture to learn the flow of a time-invariant control system in continuous time from trajectory data. Exploiting the structure of the classes of control inputs commonly used in practice, we showed that the problem of learning the flow function can be cast as the problem of learning a discrete-time dynamical system, motivating the use of an RNN-based architecture. Our experimental results on the Van der Pol and FitzHugh-Nagumo oscillators show that the learned model has good prediction performance, and demonstrate that the model is able to generalise to longer prediction time horizons and new classes of input signals.

Chapter 3

Surrogate modelling of spiking systems

In this chapter we consider the problem of learning surrogate models of spiking systems from samples of state trajectories. Spiking behaviours abound in dynamical system models of biological neurons, and there is significant interest in reproducing such behaviour in electronic devices. We propose a data-driven framework based on a recurrent neural network (RNN) architecture to approximate the flow function of conductance-based state-space models of spiking systems in continuous time.

3.1 Introduction

Motivation

Spiking systems (Sepulchre, 2022) are dynamical systems whose stability behaviour is highly input-dependent. Determined by the input excitation, the state typically either remains close to an equilibrium or enters into a limit cycle with large-amplitude oscillations called spikes. Whether the input excites the system into the oscillatory regime and how many spikes are emitted depends on both the input amplitude and frequency (Sepulchre, Drion and Franci, 2018). These systems thus possess a mixed continuous–discrete character, as the spikes can be seen as encoding digital information into a continuous-time signal. Models of biological neurons provide prototypical examples of spiking systems, where the input is a current and the spiking phenomenon is observed in the membrane potential.

Significant effort in computational neuroscience has been devoted to modelling the spiking behaviour of neurons. Hodgkin and Huxley (1952) proposed modelling the relationship between the membrane voltage and the applied current as a parallel interconnection of nonlinear conductances, which can be identified using a voltage-clamp experiment, which Burghi, Schoukens and Sepulchre (2021) relate to a general output-feedback system identification scheme. The parameter identification problem is shown to be tractable thanks to stability properties of the inverse dynamics of the conductance-based models.

Circuit-theoretic models of biological neurons suggest the possibility of building electronic devices with spiking behaviour, conceivably combining the best features of analogue and digital electronics in a single physical device (Mead, 1990; DeWeerth et al., 1991; Sepulchre, 2022). Using these biologically-inspired components in circuit design requires the ability to efficiently simulate their behaviour numerically, possibly in interconnection with many other circuit elements. However, the differential equations models of these systems are typically stiff, which suggests that surrogate models may provide computational advantages over direct integration of the differential equations. Furthermore, continuous-time representations are desirable, due to the possibly input-dependent nature of the periodicity of the spike trains, the high-frequency nature of the spike signals, and the multiple time scales involved in the dynamics.

Related work

Machine learning offers an attractive array of methods for constructing surrogate models of dynamical systems. Focusing on continuous-time models, we can distinguish between two approaches in the literature: those methods that attempt to learn a model of the system dynamics from data, and those where the goal is to directly learn a solution operator associated to the system. In the first class of methods, a neural network is typically used to parameterise the right-hand side of an ordinary differential equation (ODE), resulting in a class of models known as neural ODEs. This requires a way to automatically compute gradients of trajectory values with respect to network parameters during training, which may be done by differentiating through an ODE solver or using an adjoint method (Chen et al., 2018). In a surrogate modelling context, Yang et al. (2022) propose a neural ODE method for learning models of complex circuit elements, and derive a parameterisation of the network that guarantees input-to-state stability. Regarding the application of these models in system identification, Forgione and Piga (2021) discuss model structures and fitting criteria, and Beintema, Schoukens and Tóth (2023) propose an architecture and estimation method shown to compete with state-of-the-art nonlinear identification methods. The second group of methods, broadly known as operator learning, attempt instead to directly learn the solution map of a differential equation, i.e. the mapping from initial conditions, parameters, and external inputs to the solution. Directly parameterising the solution allows for fast evaluation for new inputs, and provided that standard deep learning toolchains are used, gradients with respect to the inputs of the model also become easy to compute. A great deal of research attention has focused on learning solution operators of partial differential equations using integral kernel parameterisations composed with neural networks (Lu et al., 2021; Li, Kovachki et al., 2021; Kissas et al., 2022). Lin, Moya and Zhang (2023) use one such parameterisation in a recursive architecture to predict trajectories of dynamical systems with external inputs. In a similar context, Qin et al. (2021) suggest a residual network-based architecture to approximate the one-step-ahead map of a dynamical system. Biloš et al. (2021) suggest a number of architectures inspired by flow functions of autonomous systems as a substitute for neural ODEs.

Contribution

Our main contribution is an operator learning framework for constructing continuous-time input–output surrogate models of spiking systems from samples of state trajectories. Starting from the flow function description of the spiking system, we directly parameterise its state trajectories. This is particularly suited for systems exhibiting spiking behaviour, as it allows for a continuous-time description of the state trajectories, while not requiring sampling the derivatives of the states (which due to the spikes can widely vary in amplitude). Indeed, from the flow function point of view the problem can be formulated as a standard (albeit infinite-dimensional) regression problem. Furthermore, by imposing the non-restrictive assumption that the input signal is piecewise constant, we show that there is an exact correspondence between the flow function and a discrete-time dynamical system. This suggests that one can approximate the flow function by an RNN, resulting in an architecture that uses only standard learning components and can be easily implemented and trained with established deep learning toolchains. Due to the nature of the spike signals, the trajectories must be densely sampled in order to correctly capture the timing and height of each spike. We propose a simple data reduction method based on rejection sampling, which enables a significant reduction of the required amount of time samples per trajectory by focusing on the most important regions of the signal. This is made possible by our continuous-time approach, which naturally allows for data that is irregularly sampled in time. Moreover, we show how the complexity of the optimisation problem can be subdued by considering segments of trajectories, using the properties of the flow function. We numerically evaluate the approach through simulations of conductance-based models of a single neuron and an interconnection of two neurons.

Outline

The remainder of the chapter is organised as follows. In Section 3.2 we introduce conductance-based neuron models in state-space form as a prototype for spiking behaviours, define the concept of flow function of a control system, and formulate the problem of constructing a surrogate model of such a system as an optimisation problem. In Section 3.3 we describe the proposed architecture and two methods for reducing the complexity of the training process. Finally, in Section 3.4 we report and discuss results from numerical experiments, and in Section 3.5 we summarise the chapter.

3.2 Problem formulation

We begin by introducing a class of state-space models of biological neurons and their interconnections, which serve as prototypical examples of systems exhibiting spiking behaviour, before introducing the definition of the flow function of a control system and the mathematical formulation of the surrogate modelling problem for these systems.

Conductance-based models

The general conductance-based model of a neuron is given by the system of differential equations (Burghi, Schoukens and Sepulchre, 2021)

$$\begin{aligned} C_m \dot{V}(t) &= u(t) - g_{\text{leak}}(V(t) - V_{\text{leak}}) - \sum_{k=1}^{N_I} I_k(V(t), m_k(t), n_k(t)) \\ \dot{m}(t) &= A_m(V(t))m(t) + b_m(V(t)) \\ \dot{n}(t) &= A_n(V(t))n(t) + b_n(V(t)), \end{aligned} \tag{3.1}$$

where V is the membrane potential, u the external (input) current, $C_m > 0$ the membrane capacitance, g_{leak} the leak conductance, and V_{leak} the reversal potential. The gating variables $m, n \in [0,1]^{N_I}$ are dimensionless, and $A_m(V), A_n(V)$ are diagonal matrices depending on V. The ionic currents are given by

$$I_k(V, m_k, n_k) = g_k m_k^{\alpha_k} n_k^{\beta_k} (V - V_{I_k}),$$

where $g_k > 0$ and $V_{I_k} \in \mathbb{R}$ are constants and α_k, β_k are nonnegative integers. A detailed description of this class of models and their biological motivation is given in Hodgkin and Huxley (1952) and Pospischil et al. (2008). These models are prototypes of spiking systems: for certain choices of the input current one observes spiking behaviour in the membrane potential signal V (Sepulchre, Drion and Franci, 2018).

One can interconnect several neuron models (3.1) to obtain more complex spiking behaviours (Giannari and Astolfi, 2022). In particular, we can model the interconnection of n_V neurons through electrical synapses

$$\begin{aligned} C_m^i \dot{V}_i(t) &= u_i(t) - g_{\text{leak}}^i(V_i(t) - V_{\text{leak}}^i) \\ &\qquad - \sum_{k=1}^{N_I^i} I_k^i(V_i(t), m_k^i, n_k^i) + \sum_{j=1}^{n_V} \epsilon_{ij}(V_j(t) - V_i(t)) \\ \dot{m}^i(t) &= A_m^i(V_i(t))m^i(t) + b_m^i(V_i(t)) \\ \dot{n}^i(t) &= A_n^i(V_i(t))n(t) + b_n^i(V_i(t)), \end{aligned} \tag{3.2}$$

for each $i \in \{1, \dots, n_V\}$ and the variables and dynamics of each individual neuron have the same meaning as described above for the case $n_V = 1$. The weight of the electrical synapse from neuron j to neuron i is given by $\epsilon_{ij} \geq 0$.

Flow functions

Consider a dynamical system described in state-space form by

$$\begin{aligned}\dot{\xi}(t) &= f(\xi(t), u(t)),\ \xi(0) = x\\ \eta(t) &= h(\xi(t)),\end{aligned} \tag{3.3}$$

with state $\xi(t) \in \mathbb{R}^{d_x}$, input $u(t) \in \mathbb{R}^{d_u}$ and output $\eta(t) \in \mathbb{R}^{d_y}$. Assume that f is such that solutions to (3.3) exist on $\mathbb{R}_{\geq 0}$ for $x \in \mathcal{X}$, with $\mathcal{X} \subset \mathbb{R}^{d_x}$ being an invariant set, and $u : \mathbb{R}_{\geq 0} \to \mathcal{U}$, $\mathcal{U} \subset \mathbb{R}^{d_u}$, is measurable and essentially bounded. Define a map $\varphi : \mathbb{R}_{\geq 0} \times \mathcal{X} \times \mathbb{U} \to \mathcal{X}$, $\mathbb{U} := L^\infty(\mathbb{R}_{\geq 0}, \mathcal{U})$, such that for any such x and u it holds that $\xi(t) = \varphi(t, x, u),\ t \geq 0$. The map φ is called the flow function of (3.3). The flow function satisfies the identity property: for any $\xi \in \mathcal{X}$ and $u \in \mathbb{U}$, $\varphi(0, x, u) = x$; and the semigroup property: for any $x \in \mathcal{X}$, $t, s \geq 0$ and $u, v \in \mathbb{U}$, $\varphi(t + s, x, u \underset{s}{\wedge} v) = \varphi(t, \varphi(s, x, u), v)$. Here $u \underset{s}{\wedge} v$ denotes the concatenation of u and v at time $s \geq 0$, defined as

$$[u \underset{s}{\wedge} v](t) = \begin{cases} u(t) & 0 \leq t < s \\ v(t - s) & t \geq s \end{cases}.$$

Henceforth we assume that the considered controls are piecewise constant with sampling period $\Delta > 0$. Thus, let $\mathbb{U}^0_\Delta \subset \mathbb{U}$, where $\Delta > 0$ and $u \in \mathbb{U}^0_\Delta$ if and only if there exists a sequence $(\omega_k)_{k=0}^\infty \subset \mathcal{U}$ such that

$$u(k\Delta + t) = \omega_k,\ k \geq 1,\ t \in [0, \Delta).$$

We restrict φ to the set

$$\mathbb{U}_\Delta := \bigcup_{s \geq 0} \sigma^s(\mathbb{U}^0_\Delta),$$

where σ is the time-shift operator defined by $(\sigma^s u)(t) = u(t + s)$ for $s \geq 0$. Thus, $\mathbb{U}_\Delta$ is the set of piecewise constant controls with sampling period Δ, where the first period does not necessarily start at $t = 0$. If $u \in \mathbb{U}_\Delta$, $u = \sigma^s v$ with $s \geq 0$ and $v \in \mathbb{U}^0_\Delta$, we may write $s = k\Delta + \delta$ for some $k \in \mathbb{Z}_{\geq 0}$ and $\delta \in [0, \Delta)$, so that $u = \sigma^\delta(\sigma^{k\Delta} v)$. Since $\sigma^{k\Delta} v \in \mathbb{U}^0_\Delta$, we have that $\mathbb{U}_\Delta = \bigcup_{s \in [0, \Delta)} \sigma^s(\mathbb{U}^0_\Delta)$.

The conductance-based models (3.1) and (3.2) can be written in the form (3.3), with the state ξ given by the membrane potentials $(V_1, \ldots, V_{n_V})$ together with the respective gating variables m^i_k, n^i_k, $i = 1, \ldots, n_V$, $k = 1, \ldots, N^i_I$, and the input signal given by the collection of input currents, $u = (u_1, \ldots, u_{n_V})$. We take the output to be the collection of membrane potentials, $\eta = h(\xi) = (V_1, \ldots, V_{n_V})$, as these are the spiking signals we are interested in simulating and, as in (3.2), the relevant signals when interconnecting neuron models.

Problem statement

Let us now state the problem considered in this chapter. Let

$$y(t,x,u) := h(\varphi(t,x,u))$$

be the trajectory of the output signal. We define the problem of obtaining a surrogate model of the system (3.3) as that of solving the optimisation problem

$$\underset{\hat{y}\in\mathcal{H}}{\text{minimise}}\ \ \ell_T(\hat{y}) := \mathbf{E}_{x,u}\,\frac{1}{T}\int_0^T \|y(t,x,u)-\hat{y}(t,x,u)\|_1\,\mathrm{d}t, \tag{3.4}$$

where x, u are assumed to be independent and distributed according to probability distributions P_x and P_u, describing the initial conditions and control inputs of interest, respectively. The distribution P_u has its support in $\mathbb{U}_\Delta$, so that sampling the values of a control $u \sim P_u$ on $[0,T]$ is equivalent to sampling a finite sequence of control values. We use the 1-norm to measure the approximation error in (3.4), since the spikes have a very short duration in time, and so the non-smoothness of the loss is beneficial for correctly capturing the shape of the signal. The set $\mathcal{H}$ is the hypothesis class from which the surrogate model $\hat{y}$ is to be selected. We consider a hypothesis class given by the RNN architecture described in the following section.

3.3 Methodology

In this section, we first motivate the approximation of the flow function by an RNN and describe the proposed architecture. We then discuss the data collection process and two issues arising in the training of the RNN architecture, as well as methods to mitigate them.

Architecture

Fix $t \geq 0$, $x \in \mathcal{X}$ and $u \in \mathbb{U}^0_\Delta$. Let (ω_k) be the sequence of values of u, and define for $\omega \in \mathcal{U}$ the constant control u_ω through $u_\omega(t) := \omega$. With $k_t := \lfloor t/\Delta \rfloor$, the value of $\varphi(t,x,u)$ can be evaluated recursively as follows:

$$\begin{aligned} x_0 &= x \\ x_{k+1} &= \varphi(\Delta, x_k, u_{\omega_k}),\ 0 \leq k < k_t \\ x_{k_t+1} &= \varphi(t - k_t\Delta, x_{k_t}, u_{\omega_{k_t}}). \end{aligned} \tag{3.5}$$

By the semigroup property, we then have $x_{k_t+1} = \varphi(t,x,u)$. Defining the function

$$\begin{aligned} \Phi &: [0,1]\times\mathcal{X}\times\mathcal{U} \to \mathcal{X} \\ \Phi(\tau,x,\omega) &= \varphi(\tau\Delta, x, u_\omega), \end{aligned}$$

and $\tau_k, k = 0, \ldots, k_t$ by

$$\tau_k = \begin{cases} 1, & k < k_t \\ (t - k_t\Delta)/\Delta, & k = k_t \end{cases}$$

we can rewrite (3.5) as

$$x_{k+1} = \Phi(\tau_k, x_k, \omega_k),\ 0 \leq k \leq k_t.$$

More generally, if $u \in \mathbb{U}_\Delta$, there is some $v \in \mathbb{U}^0_\Delta$ and $\delta \in [0, \Delta)$ such that $u = \sigma^\delta v$, and by the semigroup property $\varphi(t, x, u) = \varphi(t - \delta, \varphi(\delta, x, u_{\omega_0}), \sigma^\Delta v)$ for $t \geq \delta$. Thus, since $\sigma^\Delta v, u_{\omega_0} \in \mathbb{U}_\Delta$, one can proceed as above also in this case. This shows that the flow can be exactly computed at any time instant by a discrete-time finite-dimensional dynamical system (with inputs (τ, ω)), suggesting that one can approximate φ (and thus y) through this representation.

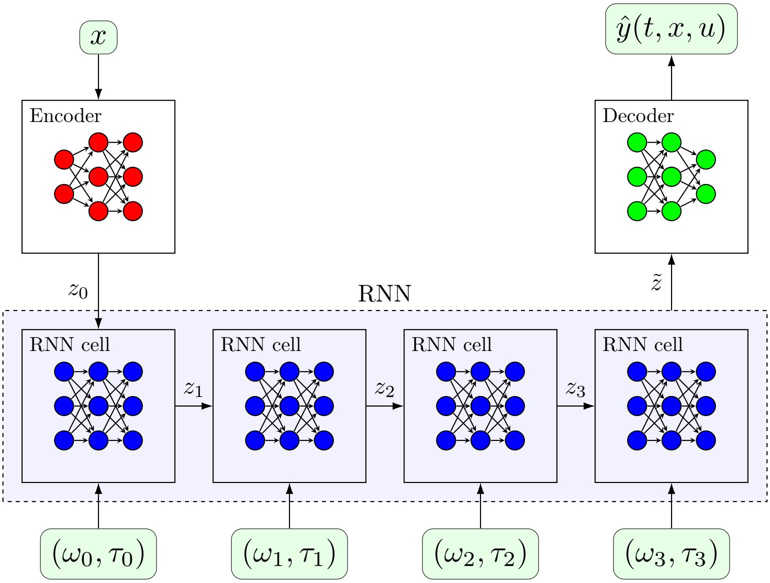

x
Encoder
z_0
RNN
RNN cell
z_1
RNN cell
z_2
RNN cell
z_3
RNN cell
$\tilde{z}$
Decoder
$\hat{y}(t, x, u)$
(ω_0, τ_0)
(ω_1, τ_1)
(ω_2, τ_2)
(ω_3, τ_3)

Figure 3.1: Schematic of the proposed architecture for $k_t = 3$.

The previous derivation motivates the choice of the hypothesis class $\mathcal{H}$ given by a parameterisation of $\hat{y}$ based on the composition of an RNN with a pair of encoder–decoder networks, as illustrated in Figure 3.1.

Data collection

To generate training data, we integrate N trajectories of the differential equation (3.3) with initial conditions x_i and inputs u_i, $i = 1, \ldots, N$, sampled from P_x and P_u, respectively, obtaining samples

$$\xi_{ik} = \varphi(t_{ik}, x_i, u_i), \; k = 1, \ldots, K, \; i = 1, \ldots, N, \tag{3.6}$$

where t_{ik} are sampled uniformly on $[0, T]$ and increasing in k. Using these samples, we construct an approximation of ℓ_T in (3.4) as

$$\hat{\ell}_T(\hat{y}) := \frac{1}{N}\frac{1}{K}\sum_{i,k} \left\| h(\xi_{ik}) - \hat{y}(t_{ik}, x_i, u_i) \right\|_1 . \tag{3.7}$$

Because φ is the flow of a spiking system, the optimisation of (3.7) is not without challenges. The spikes can be very thin, which can imply that a large number of samples are required when the sampling times t_{ik} are uniformly distributed, increasing the computational load during training. Furthermore, the spikes might be relatively infrequent, and consequently underrepresented in the data, which can make it harder to learn the spiking behaviour properly. In the next subsection we describe a simple rejection sampling algorithm that alleviates these issues.

Rejection sampling for data reduction

We propose a method that simultaneously reduces the amount of samples per trajectory needed to represent the spike signal and weights the loss function in order to emphasise learning the spiking behaviour. This is done as follows: first, sample data (3.6) with t_{ik} uniform and dense, and then use rejection sampling to 'prune' the data, i.e. select which samples to remove so that the remaining t_{ik} have a distribution which favours learning the spiking behaviour correctly.

Consider first the case of a single output, i.e. y is scalar-valued. Roughly speaking, the output signal $y(\cdot, x, u)$ has higher frequency content when its amplitude is higher (i.e. when a spike is emitted). One should thus sample more densely when the value of $y(\cdot, x, u)$ is higher. In other words, we would like that the sampling times t_{ik} be distributed according to the density function

$$p(t, x_i, u_i) \propto \left[y(t, x_i, u_i) - \min_{s \in [0,T]} y(s, x_i, u_i) \right]$$

(or, more generally, $p \propto \alpha(y)$, where α is an increasing function). If t_{ik} are sampled with this density, $\hat{\ell}_T$ in (3.7) is the empirical estimate of the weighted loss function

$$\mathbf{E}_{x,u} \, \frac{1}{T} \int_0^T p(t, x, u) \left\| y(t, x, u) - \hat{y}(t, x, u) \right\|_1 \mathrm{d}t,$$

giving higher weight to parts of the trajectories where spiking occurs.

In our case, where t_{ik} are given a priori and uniformly distributed, we can use rejection sampling (Ross, 2013) to discard certain samples so that the remaining t_{ik} are distributed approximately according to p. This implies the following procedure for each time sample t_{ik}:

- Draw $\Upsilon_{ik} \sim \mathrm{Uniform}([0,1])$;
- If $\Upsilon_{ik} > \dfrac{p(t_{ik}, x_i, u_i)}{M_i}$, where $M_i := \max_k p(t_{ik}, x_i, u_i)$, remove the sample.

After a single pass through the dataset, the undiscarded t_{ik} will be approximately distributed with density $p(\cdot, x_i, u_i)$. Furthermore, the probability of accepting a sample from the ith trajectory is approximately equal to $1/M_i$, giving the approximate fraction of samples which will be retained.

We are thus at once able to reduce the volume of data while preserving a faithful representation of the spiking signals, and to increase the weight of the spiking regions in the loss function. Other choices of p, e.g. involving the derivative or frequency content of the output signal, could of course also be used.

If there are several outputs, so y is vector-valued, one may combine the outputs into a scalar signal that contains the spikes of all outputs, for instance, $p(t,x,u) \propto \max_{i=1,\dots,d_y} \alpha_i(y_i(t,x,u))$, where α_i are monotone functions to ensure that y_i, $i = 1, \dots, d_y$, are normalised to the same range.

Windowed loss using the semigroup property

The complexity of optimising the empirical loss (3.7) with an RNN is highly dependent on the length of the input sequences. This dependence is twofold: the computational effort of the forward and backward passes through the recurrent network depends linearly on the simulation length, and simultaneously the loss function becomes less smooth with respect to the network parameters as the sequence length increases (Ribeiro et al., 2020). We describe here how this issue can be addressed in the context of our method, by reducing the length of input sequences while still making use of all training data. In a similar way to Ribeiro et al. (2020) and Beintema, Schoukens and Tóth (2023), we construct a new loss function by considering shorter segments of output trajectories. This is easy to do in our setting, as we can take advantage of properties of the flow function.

It follows from the semigroup property of φ that for $k \le j \le K$ we have

$$\xi_{ij} = \varphi(t_{ij} - t_{ik}, \xi_{ik}, \sigma^{t_{ik}} u_i).$$

Hence, with the same training data we can construct the loss function

$$\hat{\ell}^{\mathrm{win}}(\hat{y}) := \frac{1}{N} \frac{1}{K} \sum_{i,k} \frac{1}{|J_{ik}|} \sum_{j \in J_{ik}} \left\| h(\xi_{ij}) - \hat{y}(t_{ij} - t_{ik}, \xi_{ik}, \sigma^{t_{ik}} u_i) \right\|_1,$$

where J_{ik} are sets of indices such that $J_{ik} \subset [k, K]$, and $|J_{ik}|$ denotes the cardinality of J_{ik}. In this case, the maximum length of the input sequences is given by

$L := 1 + \max_{i,k} \max_{j \in J_{ik}} \lfloor \frac{t_{ij} - t_{ik}}{\Delta} \rfloor$, which we can choose by appropriate selection of the index sets J_{ik}. This allows for controlling the time complexity of the training epochs and the smoothness of the loss function. Of course, it is not necessarily the case that $\hat{\ell}^{\text{win}}$ is the empirical mean approximation of ℓ_T, so care must be taken to avoid overfitting.

3.4 Numerical experiments

We perform two experiments with data from simulations of two conductance-based models: a single neuron model, and a model of the feedforward interconnection of two neurons with an electrical synapse.

Model of a single fast-spiking neuron

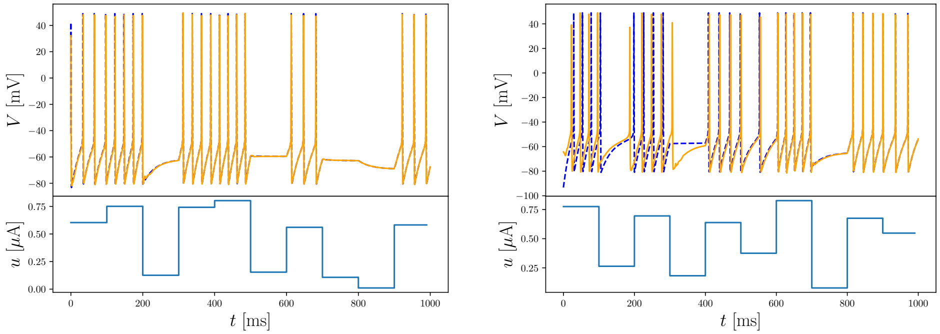


Figure 3.2: Predictions of output trajectories for a single fast-spiking neuron. The surrogate model prediction is shown in orange; the real output is plotted in blue (dashed). Note that the surrogate model closely reproduces the spiking behaviour of the neuron model, and is able recover from errors in the prediction of the initial condition.

Dynamics We consider a conductance-based model of a single neuron as in (3.1) with a sodium current,

$$I_1(V, m_1, n_1) = g_1 m_1^3 n_1 (V - V_1),$$

and a potassium ionic current,

$$I_2(V, m_2, n_2) = g_2 n_2^4 (V - V_2),$$

so that $N_I = 2$. The model can be described with four states, since $\alpha_2 = 0$ and so the equation for m_2 can be removed. This corresponds to a fast-spiking neuron, and is identical to the model structure originally proposed in Hodgkin and Huxley (1952). The full equations and parameter values may be found in Giannari and Astolfi (2022).

Data collection We integrate $N = 800$ state trajectories of the model over $t \in [0, T]$, $T = 500$ ms, using a backwards differentiation formula integrator, and collect $K = 50000$ samples from each trajectory, with t_{ik} sampled uniformly using Latin hypercube sampling. The initial conditions are sampled uniformly with $V(0) \sim \text{Uniform}([-100, 100])$ mV and $m_k(0), n_k(0) \sim \text{Uniform}([0, 1])$. The control inputs have period $\Delta = 10$ ms and input values are sampled according to

$$\begin{aligned} \omega_{10k} &\overset{\text{i.i.d.}}{\sim} \text{Uniform}([0,1])\ \mu\text{A},\ k \geq 0 \\ \omega_{10k+j} &= \omega_{10k},\ k \geq 0,\ 0 \leq j < 10, \end{aligned}$$

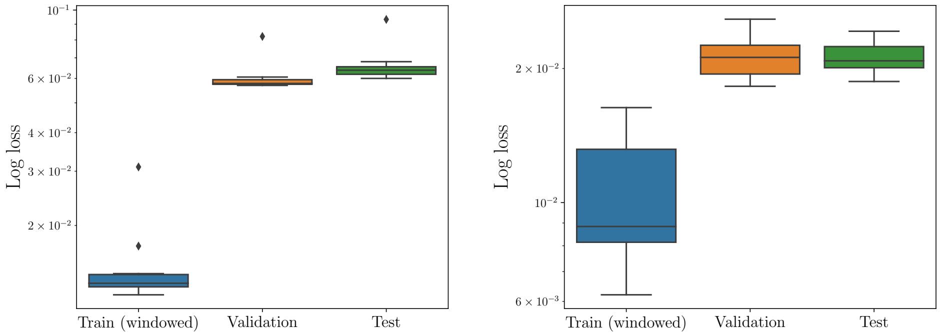
i.e. the input changes every 100 ms.

We reduce the dataset using the rejection sampling method described above, sampling according to the density $p(t, x, u) \propto \tilde{y}(t, x, u)$, where $\tilde{y}$ is the normalisation of the output y to $[0, 1]$. We ensure that the local maxima of the signal (i.e. the spike peaks) and the initial state are included in the final dataset. The resulting sampled trajectories are split into training, validation, and testing sets according to a 60/20/20% random split.

Architecture and training We train 10 models with the architecture described in 3.3. Each RNN is a long short-term memory (LSTM) network with 24 hidden states. The encoder and decoder are feedforward networks with tanh activations and three hidden layers. The encoder network maps the initial condition to the initial hidden state of the LSTM. The cell state of the LSTM is always zero-initialised.

We apply the windowing technique described in Section 3.3, where J_{ik} has at most 5 elements drawn uniformly (without repetition) from $\{k, \ldots, k + 20\}$, so that the input sequences to the RNN have maximum length $L = 20$. The windowed empirical loss is minimised using the Adam algorithm with an initial learning rate of 1×10^{-3}. The learning rate is reduced by a factor of 10 whenever the empirical loss (3.7) constructed with the validation data does not decrease for 5 consecutive epochs. Training is stopped when the validation loss does not decrease for 15 consecutive epochs.

Results Figure 3.2 shows two trajectory predictions with unseen test inputs and initial conditions from the model with the smallest validation loss among the 10 models. We observe that the surrogate model is able to closely capture the timing and height of the spikes. In the right-hand side figure, we see that the model is not able to predict the initial condition of the system correctly and consequently misses the timing of the first few spikes, but nonetheless correctly captures the spikes emitted after $t = 400$ ms. It is interesting to note that although the system does not have fading memory, i.e. the effect of initial conditions does not necessarily disappear as $t \to \infty$, the error in the initial conditions is not persistent in the output of the surrogate model. Figure 3.3a shows the distributions of the losses for the 10 models, and we observe that the training procedure is robust to the initialisation of the network parameters.



(a) (b)

Figure 3.3: Plot of the loss distributions for the 10 models trained with data from each system: (a) single fast-spiking neuron; (b) feedforward interconnection of two neurons.

Feedforward interconnection of two neuron models

Dynamics We consider a model of the form (3.2) with $n = 2$. Each of the neurons has $N_I = 3$ with I_1, I_2 as in the previous subsection and an additional potassium current given by

$$I_3(V, m_3, n_3) = g_3 n_3 (V - V_3),$$

so that each neuron has 5 states, and thus the interconnection results in a model with 10 states. We take $\epsilon_{12} = 0.1$ S, $\epsilon_{21} = 0$ S, and $u_2 \equiv 0$, corresponding to a feedforward interconnection of two regular spiking with adaptation type neurons, as described in Giannari and Astolfi (2022).

Data collection We follow the procedure described in the previous subsection, collecting 200 trajectories on the time interval $[0, 1000]$ ms, with the same distributions for the initial conditions of the membrane voltages and the gating variables, and the same distribution for the current input $u = u_1$. The rejection sampling is performed with the density $p(t, x, u) \propto \max_{i=1,2} \tilde{y}_i(t, x, u)$.

Architecture and training The details of the architecture and training procedure are as in the previous subsection, the sole difference being that the LSTM network now has 32 hidden states.

Results Figure 3.4 shows two trajectory predictions with unseen test inputs and initial conditions. As in the previous subsection, we observe that the surrogate model faithfully reproduces the behaviour of the spiking system. Similarly, in Figure 3.3b we verify the robustness of the training procedure with respect to the training parameters.

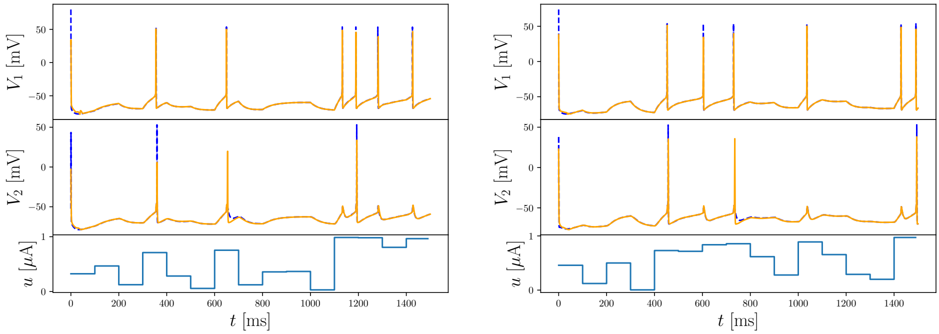


Figure 3.4: Predictions of output trajectories for the feedforward interconnection of two neurons. The surrogate model prediction is shown in orange; the real output is plotted in blue (dashed).

3.5 Summary

We proposed a framework for surrogate modelling of spiking systems based on approximating the flow function of a class of state-space models exhibiting spiking behaviour. The flow function approximation was performed using an RNN architecture which allows for a direct continuous-time parameterisation of the output trajectories. We discussed two issues which arise when training this architecture on data from a spiking system, namely, the amount of data required to accurately represent the spike signals and the complexity of the optimisation problem, and show how these can be addressed in the context of our method. Finally, we presented results from two numerical experiments which illustrate the feasibility of using our framework for constructing surrogate models of spiking systems.

Chapter 4

A universal approximation theorem

In this chapter, we consider the problem of approximating the flow function of a dynamical system by a discrete-time recurrent neural network. We give conditions under which the architecture proposed in Chapter 2 is a universal approximator of flows of a general control system. We further study the case of systems with dynamics given by controlled ordinary differential equations.

4.1 Introduction

Motivation

A hypothesis space is said to possess the universal approximation property if it is dense in a function space of interest. This means that, excluding the presence of other errors, a target function in that function space may be arbitrarily well approximated by elements of the hypothesis space. Thus, this notion corresponds to a form of well-posedness for a learning problem.

In Chapter 2 we proposed an architecture based on a discrete-time recurrent neural network for solving the flow function learning problem, hinging on a discrete structure induced in the flow function when considering the most commonly used classes of input signals. The results of the two previous chapters have confirmed the practical applicability of the architecture. Here, we approach the problem theoretically and ask under which conditions the architecture is able to approximate the flow function of an arbitrary control system.

Related work

A general discussion of universal approximation is given in (Kratsios, 2021). Hornik (1991) proves universal approximation properties for neural networks with multiple layers. For residual networks, Tabuada and Gharesifard (2022) establish a relationship between universal approximation and controllability of a related dynamical system. Schäfer and Zimmermann (2006) show that discrete-time recurrent neural networks are universal approximators of discrete-time controlled dynamical systems. In (Hanson and Raginsky, 2019), it is shown that convolutional architectures approximate discrete-time input-output operators possessing a property called approximately finite memory. This property is connected to the fading memory property, which in (Boyd and Chua, 1985) is shown to be related to universal approximation of input-output systems by Volterra series.

It is well known that continuous-time recurrent neural networks can approximate large classes of continuous-time dynamical systems with inputs, see (Sontag, 1992; Li, Ho and Chow, 2005) and references therein. Hanson and Raginsky (2020) show that these networks are able to approximate flows of incrementally stable continuous-time control systems on unbounded time intervals. Approximation of general continuous-time input-output operators by these networks is studied in (Hanson, Raginsky and Sontag, 2021).

A discussion of universal approximation properties of neural ordinary differential equations may be found in (Kidger, 2021); Veeravalli and Raginsky (2023) consider the reverse problem of studying the function class generated by a stochastic neural ordinary differential equation. Universal approximation properties of neural operator architectures are discussed in (Kovachki, Lanthaler and Mishra, 2021; Lu et al., 2021; Kissas et al., 2022).

Contribution

The contribution is twofold. Firstly, we prove that the architecture proposed in Chapter 2 is a universal approximator of flow functions of control systems, which implies that the learning problem we have formulated there is well-posed. Secondly, we show by system-theoretic arguments that the required assumptions hold for systems whose dynamics are given by well-behaved ordinary differential equations (ODEs), with rather general and practically relevant classes of input signals.

Outline

The remainder of this chapter is organised as follows. Section 4.2 introduces notation and some basic definitions. In Section 4.3 we describe the proposed architecture and the considered class of input signals. This is followed by the statement and proof of the main result, Theorem 1, in Section 4.4. Section 4.5 treats the case of flows of controlled ODEs and the assumptions of Theorem 1 are shown to hold in that setting. A summary of the chapter is given in Section 4.6.

4.2 Preliminaries

Notation

The indicator function of a set A is written $\mathbf{1}_A$, and the identity function on A is written id_A. Sequences are written $(z_k)_{k=0}^{\infty}$, or in short-hand (z_k). The space of sequences with values in A is written $S(A) := \{(z_k)_{k=0}^{\infty} : z_k \in A\}$. The space of continuous functions $f : A \to \mathbb{R}^n$ on a compact set $A \subset \mathbb{R}^m$ is written $C_n(A)$. For $A \subset \mathbb{R}^n$ and $\varepsilon > 0$, $N_\varepsilon(A)$ denotes the (closed) ε-neighbourhood of A, i.e. the set of points at most ε distance away from A. If A is compact, then so is $N_\varepsilon(A)$. Vectors $v \in \mathbb{R}^d$ are written $v = (v_1, \ldots, v_d)$. We denote by $\|\cdot\|$ the Euclidean norm on $\mathbb{R}^d$, and for a matrix $M \in \mathbb{R}^{m\times n}$, $\|M\|$ denotes the induced operator norm.

Flow functions

We consider finite-dimensional time-invariant control systems in continuous time with state evolving in an open set $X \subset \mathbb{R}^{d_x}$. Such systems can be described abstractly by a flow function

$$\varphi : \mathbb{R}_{\geq 0} \times X \times \mathbb{U} \to X \tag{4.1}$$

where $\mathbb{U}$ is a given set of control inputs $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^{d_u}$. The flow satisfies the following properties (Sontag, 1998, Chapter 2):

- Identity: $\varphi(0, x, u) = x$
- Semigroup: $\varphi(s + t, x, u) = \varphi(t, \varphi(s, x, u), u^s)$

for all $x \in X$, $u \in \mathbb{U}$ and $s, t \geq 0$. Here $u^s \in \mathbb{U}$ denotes the input u shifted by $s > 0$ time units, i.e. $u^s(t) := u(t + s)$. The function $t \mapsto \varphi(t, x, u), t \geq 0$ is the trajectory of the system with initial state x when the applied control is u.

Neural networks as function approximators

In this chapter, a (feedforward) neural network is any function $h : \mathbb{R}^m \to \mathbb{R}^n$ which can be written as

$$h(x) = C\sigma_p(Ax + b) + d,\ x \in \mathbb{R}^m \tag{4.2}$$

for $A \in \mathbb{R}^{p\times m}$, $b \in \mathbb{R}^p$, $C \in \mathbb{R}^{n\times p}$, $d \in \mathbb{R}^n$. Here $\sigma_p : \mathbb{R}^p \to \mathbb{R}^p$ is a diagonal mapping such that the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is applied to each coordinate, i.e. $\sigma_p(v) = (\sigma(v_1), \ldots, \sigma(v_p))$ for $v \in \mathbb{R}^p$. In practical applications, these are usually known as networks with one hidden layer.

We let $\mathfrak{N}_{\sigma,p}^{m,n}$ be the class of such networks and define

$$\mathfrak{N}_{\sigma}^{m,n} := \bigcup_{p=1}^{\infty} \mathfrak{N}_{\sigma,p}^{m,n}.$$

Throughout the chapter, we shall assume that $\sigma : \mathbb{R} \to \mathbb{R}$ is a fixed bounded, continuous and nonconstant function. Under this assumption it is well-known (Hornik, 1991) that $\mathfrak{N}_\sigma^{m,n}$ is dense in $C_n(K)$ for any compact set $K \subset \mathbb{R}^m$. That is, for any continuous function $f : K \to \mathbb{R}^n$ and $\varepsilon > 0$ there is a network $h \in \mathfrak{N}_\sigma^{m,n}$ such that

$$\sup_{x \in K} \|f(x) - h(x)\| < \varepsilon.$$

Let $\mathfrak{N}_{\sigma,p}^0 \subset \cup_{q \geq p} \mathfrak{N}_{\sigma,p}^{q,p}$ be the class of feedforward networks for which $C = I$ and $d = 0$ in (4.2). A Recurrent Neural Network (RNN) is then simply a difference equation whose right-hand side is a network in $\mathfrak{N}_{\sigma,p}^0$ for some $p \geq 0$:

Definition 1 (RNN)**.** An RNN is a difference equation of the form

$$z_{k+1} = \sigma_{d_z}(Az_k + Bu_k + b), \ k \in \mathbb{Z}_{\geq 0},$$

where $z \in \mathbb{R}^{d_z}$, $u \in \mathbb{R}^{d_u}$, $A \in \mathbb{R}^{d_z \times d_z}$, $B \in \mathbb{R}^{d_z \times d_u}$, and $b \in \mathbb{R}^{d_z}$.

4.3 Architecture definition

In this section we define a discrete-time RNN-based architecture to approximate flow functions of continuous-time dynamical systems. We focus in particular on systems for which the trajectories $t \mapsto \varphi(t, x, u)$ are continuous in time t. This is the case when φ arises from a differential equation, but excludes e.g. hybrid systems with state jumps. We shall show that φ can be approximated by a function $\hat{\varphi}$ on a finite time interval, where $\hat{\varphi}(t, x, u)$ is computed by an RNN. In the following sections we make this precise.

Class of inputs

In order to approximate φ, we must impose some structure on $\mathbb{U}$. In practice, the majority of systems are controlled by a computer with a zero-order hold digital-to-analog converter, so that the input signal will be piecewise constant, with the control value changing at regular time instants with some period $\Delta > 0$. Occasionally, first- or higher-order polynomial parameterisations are also used. In this chapter we consider a general parameterisation of control inputs which encompasses all of these cases. Namely, we assume that the control can be parameterised by a sequence of finite-dimensional parameters $(\omega_k)_{k=0}^\infty \subset \mathbb{R}^{d_\omega}$ as follows:

$$u(t) = \sum_{k=0}^{\infty} \alpha\left(\omega_k, \frac{t}{\Delta}\right) \mathbf{1}_{[k\Delta,(k+1)\Delta)}(t), \ t \geq 0. \tag{4.3}$$

Here $\alpha : \mathbb{R}^{d_\omega} \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{d_u}$ is periodic with period 1 in its second argument. In other words, we have for each $k \geq 0$

$$u(t) = \alpha(\omega_k, t/\Delta), \ k\Delta \leq t < (k+1)\Delta.$$

Throughout the chapter we assume that Δ and the function α are fixed and known. For a set $\Omega \subset \mathbb{R}^{d_\omega}$ we define the set $\mathbb{U}(\Omega)$ of controls u parameterised by sequences in $S(\Omega)$ according to (4.3), i.e.

$$\begin{aligned}\mathbb{U}(\Omega) := \Bigg\{ u : \mathbb{R}_{\geq 0} \to \mathbb{R}^{d_u} : (\omega_k)_{k=0}^{\infty} \in S(\Omega), \\ u(t) = \sum_{k=0}^{\infty} \alpha\left(\omega_k, \frac{t}{\Delta}\right) \mathbf{1}_{[k\Delta,(k+1)\Delta)}(t) \Bigg\}.\end{aligned} \tag{4.4}$$

Representing flows by discrete-time systems

We let u_ω be the control generated by the constant sequence with value ω, so that $u_\omega(t) = \alpha(\omega, t/\Delta),\ t \geq 0$ and define the function $\Phi : [0,1] \times X \times \mathbb{R}^{d_\omega} \to X$ by

$$\Phi(\tau, x, \omega) := \varphi(\tau\Delta, x, u_\omega).$$

Fix $t \in \mathbb{R}_{\geq 0}$ and define $k_t := \lfloor t/\Delta \rfloor,\ \tau_t := (t - k_t\Delta)/\Delta$. The value of $\varphi(t, x, u)$ can be computed recursively by Φ as follows:

$$\begin{aligned} x_0 &= x \\ x_{k+1} &= \Phi(1, x_k, \omega_k),\ 0 \leq k < k_t \\ x_{k_t+1} &= \Phi(\tau_t, x_{k_t}, \omega_{k_t}) = \varphi(t, x, u). \end{aligned} \tag{4.5}$$

This can be seen as representing φ by a discrete-time system with inputs $(\tau, \omega) \in [0,1] \times \mathbb{R}^{d_\omega}$. Note that such a representation does not amount to a discretisation of φ, so that no loss of information or generality is incurred, and we are able to compute the flow φ at any instant of time through this correspondence.

The discrete-time system defined by (4.5) can be approximated by an RNN as follows. Let $x \in X$, $t \geq 0$ and $u \in \mathbb{U}$ be parameterised according to (4.3) by a sequence (ω_k). Fixing networks $h \in \mathfrak{N}^0_{\sigma,d_z}$, $\beta \in \mathfrak{N}^{d_x,d_z}_\sigma$ and $\gamma \in \mathfrak{N}^{d_z,d_x}_\sigma$, compute the sequence

$$\begin{aligned} z_0 &= \beta(x) \\ z_{k+1} &= h(1, z_k, \omega_k),\ 0 \leq k < k_t \\ z_{k_t+1} &= h(\tau_t, z_{k_t}, \omega_{k_t}) \end{aligned} \tag{4.6}$$

and set

$$\hat{\varphi}(t, x, u) = \gamma((1 - \tau_t) z_{k_t} + \tau_t z_{k_t+1}).$$

The interpolation guarantees that $\hat{\varphi}$ is continuous in t. Note that it does not amount to a linear interpolation, as z_{k_t+1} depends on τ_t.

In order to express $\hat{\varphi}$ explicitly, the following definition is useful, and will be used throughout the following sections.

Definition 2 (Recursion map)**.** Let $f : A \times B \to A$. The associated recursion map $\rho_f : \mathbb{Z}_{\geq 0} \times A \times S(B) \to A$ is defined as

$$\begin{aligned} \rho_f(0, x, (u_k)) &= x, \\ \rho_f(n+1, x, (u_k)) &= f(\rho_f(n, x, (u_k)), u_n),\ n \geq 0. \end{aligned} \tag{4.7}$$

Now let $(\mathfrak{t}_k^t)_{k=0}^{\infty} \in S([0,1])$ be defined by

$$\mathfrak{t}_k^t = \begin{cases} 1, & 0 \leq k < k_t \\ \tau_t, & k = k_t \\ 0, & k > k_t. \end{cases} \tag{4.8}$$

Then we can rewrite (4.6) as $z_k = \rho_h(k, \beta(x), (\mathfrak{t}_k^t, \omega_k))$, $k \geq 0$, where, with a slight abuse of notation, we interpret h as a function mapping $\mathbb{R}^{d_z} \times ([0,1] \times \mathbb{R}^{d_\omega})$ to $\mathbb{R}^{d_z}$. Hence, $\hat{\varphi}$ can be written

$$\hat{\varphi}(t,x,u) = \gamma[(1-\tau_t)\rho_h(k_t, \beta(x), (\mathfrak{t}_k^t, \omega_k)) + \tau_t \rho_h(k_t+1, \beta(x), (\mathfrak{t}_k^t, \omega_k))].$$

We let $\mathcal{H}$ denote the set of functions $\hat{\varphi} : \mathbb{R}_{\geq 0} \times X \times \mathbb{U} \to \mathbb{R}^{d_x}$ defined in this way, that is,

$$\begin{aligned} \mathcal{H} := \Big\{ &\hat{\varphi} : \mathbb{R}_{\geq 0} \times X \times \mathbb{U} \to \mathbb{R}^{d_x} : d_z \in \mathbb{Z}_{\geq 0}, \\ &\gamma \in \mathfrak{N}_\sigma^{d_z, d_x},\ h \in \mathfrak{N}_{\sigma, d_z}^0,\ \beta \in \mathfrak{N}_\sigma^{d_x, d_z}, \\ &\hat{\varphi}(t,x,u) = \gamma[(1-\tau_t)\rho_h(k_t, \beta(x), (\mathfrak{t}_k^t, \omega_k)) \\ &\qquad + \tau_t \rho_h(k_t+1, \beta(x), (\mathfrak{t}_k^t, \omega_k))] \Big\}. \end{aligned} \tag{4.9}$$

4.4 Universal approximation of flow functions

We are now able to state the main result in this chapter.

Theorem 1. *Suppose the flow of a control system* $\varphi : \mathbb{R}_{\geq 0} \times X \times \mathbb{U} \to X$ *satisfies the following assumptions:*

1. *Given a compact set* $K_\omega \subset \mathbb{R}^{d_\omega}$*, define* $\mathbb{U}(K_\omega)$ *according to (4.4). Then* $\mathbb{U}(K_\omega) \subset \mathbb{U}$*, i.e. for any* $u \in \mathbb{U}(K_\omega)$*, the corresponding trajectory* $\varphi(\cdot, x, u)$ *is well-defined for all* $x \in X$*.*
2. *The function* $\Phi : [0,1] \times X \times \mathbb{R}^{d_\omega} \to X$ *defined as*

$$\Phi(\tau, x, \omega) := \varphi(\tau\Delta, x, u_\omega),\ u_\omega(t) = \alpha(\omega, t/\Delta)$$

   is right-differentiable at $\tau = 0$ *for every* $(x, \omega) \in X \times \mathbb{R}^{d_\omega}$*.*
3. *The function* $\Psi : [0,1] \times X \times \mathbb{R}^{d_\omega}$ *defined as*

$$\Psi(\tau, x, \omega) := \begin{cases} x + \tau^{-1}(\Phi(\tau, x, \omega) - x), & \tau \in (0,1] \\ \lim_{t \downarrow 0} \left[x + t^{-1}(\Phi(t, x, \omega) - x)\right], & \tau = 0 \end{cases} \tag{4.10}$$

   is continuous and locally Lipschitz in x*.*

Then, for any $\varepsilon > 0$*,* $T \geq 0$ *and compact sets* $K_x \subset X$*,* $K_\omega \subset \mathbb{R}^{d_\omega}$*, there exists* $\hat{\varphi} \in \mathcal{H}$*, defined according to* (4.9)*, such that* $\|\varphi(t,x,u) - \hat{\varphi}(t,x,u)\| < \varepsilon$ *holds for all* $t \in [0,T]$*,* $x \in K_x$ *and* $u \in \mathbb{U}(K_\omega)$*. Furthermore,* γ *and* β *in* (4.9) *can be chosen to be affine with* $\gamma \circ \beta = \mathrm{id}_{\mathbb{R}^{d_x}}$*.*

Note that assumptions 2 and 3 implicitly represent assumptions on φ and α. In Section 4.5 we shall give conditions under which they are satisfied for flows of differential equations. The proof of Theorem 1 will require the following result on universal approximation of discrete-time systems using RNNs.

Theorem 2 (Universal approximation for discrete-time dynamical systems)**.** *Let* $f : \mathbb{R}^{d_x} \times \mathbb{R}^{d_u} \to \mathbb{R}^{d_x}$ *be a continuous function that is locally Lipschitz in the first variable, in the sense that for any compact set* $K \subset \mathbb{R}^{d_x}$ *there exists a locally bounded function* $\nu_K : \mathbb{R}^{d_u} \to \mathbb{R}_{\geq 0}$ *such that*

$$\|f(x_2,u) - f(x_1,u)\| \leq \nu_K(u)\,\|x_2 - x_1\|\,,\ x_1, x_2 \in K.$$

Then for any $\varepsilon > 0$*,* $N \in \mathbb{Z}_{\geq 0}$ *and compact sets* $K_x \subset \mathbb{R}^{d_x}$ *and* $K_u \subset \mathbb{R}^{d_u}$ *there exist networks* $h \in \mathfrak{N}^0_{\sigma,d_z}$*,* $\gamma \in \mathfrak{N}^{d_z,d_x}_\sigma$ *and* $\beta \in \mathfrak{N}^{d_x,d_z}_\sigma$ *such that for any* $x \in K_x$ *and* $u \in S(K_u)$ *we have*

$$\|\rho_f(n,x,u) - \gamma(\rho_h(n,\beta(x),u))\| < \varepsilon,\ n = 0,\dots,N, \tag{4.11}$$

where ρ_f *is defined as in* (4.7)*. Furthermore,* γ *and* β *can be chosen to be affine with* $\gamma \circ \beta = \mathrm{id}_{\mathbb{R}^{d_x}}$*.*

That RNNs are universal approximators of discrete-time dynamical systems is well-known (Sontag, 1992; Schäfer and Zimmermann, 2006). However, we have stated it here in the form most appropriate for the proof of Theorem 1, and for the sake of completeness we provide a proof in Section 4.7.

Proof of Theorem 1. We begin with some intuition on the definition of Ψ in (4.10) and the stated assumptions. Fix $\hat{\varphi} \in \mathcal{H}$ and let β, h, γ be the corresponding networks as in (4.9). Assume for the moment that $\gamma = \beta = \mathrm{id}_{\mathbb{R}^{d_x}}$. In the first control period, i.e. for $0 < \tau \leq 1$ it holds that

$$\begin{aligned}\varphi(\tau\Delta,x,u_\omega) - \hat{\varphi}(\tau\Delta,x,u_\omega) &= \Phi(\tau,x,\omega) - [(1-\tau)x + \tau h(\tau,x,\omega)]\\ &= \tau\big(h(\tau,x,\omega) - \big[x + \tau^{-1}(\Phi(\tau,x,\omega) - x)\big]\big)\\ &= \tau(h(\tau,x,\omega) - \Psi(\tau,x,\omega)).\end{aligned}$$

Furthermore, note that $\Phi(1,x,\omega) = \Psi(1,x,\omega)$, so if we replace Φ by Ψ in (4.5) we get the same result, provided we interpolate the final state, that is,

$$\varphi(t,x,u) = \Phi(\tau_t, x_{k_t}, \omega_{k_t}) = (1-\tau_t)x_{k_t} + \tau_t\Psi(\tau_t, x_{k_t}, \omega_{k_t}).$$

This motivates the idea that we should approximate the discrete dynamical system obtained by iterating Ψ:

$$x_{k+1} = \Psi(\tau_k, x_k, \omega_k),\ k \geq 0.$$

Using the recursion map notation, we can equivalently write

$$x_k = \rho_\Psi(k, x_0, (\tau_k, \omega_k)_{k=0}^\infty).$$

By Theorem 2, there exists a network $h \in \mathfrak{N}^0_{\sigma,d_z}$ and affine maps γ, β such that

$$\|\gamma(\rho_h(n, \beta(x), (\tau_k, \omega_k))) - \rho_\Psi(n, x, (\tau_k, \omega_k))\| < \varepsilon \tag{4.12}$$

for $n = 0,\ \ldots,\ k_T + 1$ and any $x \in K_x$ and $(\tau_k, \omega_k) \in S([0,1] \times K_\omega)$. Let $\hat{\varphi} \in \mathcal{H}$ be defined by these three networks according to (4.9), and recall that γ, β may be chosen so that $\gamma \circ \beta = \mathrm{id}_{\mathbb{R}^{d_x}}$.

Fix $x \in K_x$, $u \in \mathbb{U}(K_\omega)$ and $t \in [0,T]$. Let (ω_k) be a sequence parameterising the control u and define

$$\begin{aligned} z_0 &= \beta(x) \\ z_{k+1} &= h(1, z_k, \omega_k),\ 0 \leq k < k_t \\ z_{k_t+1} &= h(\tau_t, z_{k_t}, \omega_{k_t}). \end{aligned}$$

Then, as before

$$z_n = \rho_h(n, \beta(x), (\mathfrak{t}^t_k, \omega_k)),\ 0 \leq n \leq k_t + 1$$

(recall the definition of $(\mathfrak{t}^t_k)$ in (4.8)). It follows from (4.12) that

$$\|\varphi(k\Delta, x, u) - \gamma(z_k)\| < \varepsilon$$

for $k = 0, \ldots, k_t$ and with $x_{k_t} := \varphi(k_t\Delta, x, u)$

$$\|\Psi(\tau_t, x_{k_t}, \omega_{k_t}) - \gamma(z_{k_t+1})\| < \varepsilon.$$

Write

$$\begin{aligned} \varphi(t,x,u) - \hat{\varphi}(t,x,u) &= \varphi(t,x,u) - \gamma((1-\tau_t)z_{k_t} + \tau_t z_{k_t+1}) \\ &= \Phi(\tau_t, x_{k_t}, \omega_{k_t}) - \gamma((1-\tau_t)z_{k_t} + \tau_t z_{k_t+1}) \\ &= x_{k_t} + \tau_t(\Psi(\tau_t, x_{k_t}, \omega_{k_t}) - x_{k_t}) - \gamma((1-\tau_t)z_{k_t} + \tau_t z_{k_t+1}) \\ &= (1-\tau_t)x_{k_t} + \tau_t\Psi(\tau_t, x_{k_t}, \omega_{k_t}) - (1-\tau_t)\gamma(z_{k_t}) - \tau_t\gamma(z_{k_t+1}) \\ &= (1-\tau_t)(x_{k_t} - \gamma(z_{k_t})) + \tau_t(\Psi(\tau_t, x_{k_t}, \omega_{k_t}) - \gamma(z_{k_t+1})). \end{aligned}$$

If $t < \Delta$ then $k_t = 0$, so that

$$\begin{aligned} \varphi(t,x,u) - \hat{\varphi}(t,x,u) &= (1-\tau_t)(x - \gamma(z_0)) + \tau_t(\Psi(\tau_t, x, \omega_0) - \gamma(z_1)) \\ &= (1-\tau_t)(x - \gamma(\beta(x))) + \tau_t(\Psi(\tau_t, x, \omega_0) - \gamma(z_1)) \\ &= \tau_t(\Psi(\tau_t, x, \omega_0) - \gamma(z_1)), \end{aligned}$$

and thus

$$\|\varphi(t,x,u)-\hat{\varphi}(t,x,u)\| = \tau_t\|\Psi(\tau_t,x,\omega_0)-\gamma(z_0)\| < \varepsilon\tau_t \leq \varepsilon.$$

For $t \geq \Delta$, we have

$$\begin{aligned}\|\varphi(t,x,u)-\hat{\varphi}(t,x,u)\| &\leq (1-\tau_t)\|x_{k_t}-\gamma(z_{k_t})\| + \tau_t\|\Psi(\tau_t,x_{k_t},\omega_{k_t})-\gamma(z_{k_t+1})\| \\ &< \varepsilon,\end{aligned}$$

and the proof is complete. □

4.5 Flows of controlled ODEs

In this section, we consider the class of flows φ arising from a controlled ODE of the form

$$\dot{\xi}(t) = f(\xi(t),u(t)),\ \xi(0)=x. \tag{4.13}$$

If the function $f : X \times \mathbb{R}^{d_u} \to \mathbb{R}^{d_x}$ is sufficiently regular, the flow of such a system is well-defined for all Borel measurable and essentially bounded controls (Sontag, 1998, Appendix C), and satisfies the ODE in the following sense:

$$\varphi(t,x,u) = x + \int_0^t f(\varphi(s,x,u),u(s))\mathrm{d}s,\ t \in \mathbb{R}_{\geq 0}. \tag{4.14}$$

In particular, if f is continuous and the control u is right-continuous at time $s \geq 0$, it then holds that

$$\left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=s} \varphi(t,x,u) = f(\varphi(s,x,u),u(s)). \tag{4.15}$$

We now show that the assumptions of Theorem 1 are satisfied case under mild conditions on the input parameterisation α and the right-hand side f of the ODE.

Lemma 1. *Assume that the functions f in* (3.3) *in and α in* (4.3) *satisfy the following assumptions:*

I) The function α is measurable, and for each $\omega \in \mathbb{R}^{d_\omega}$ the function $\alpha(\omega,\cdot)$ is bounded on $[0,1]$ and right-continuous at $t=0$. Furthermore, the family of functions $\{\alpha(\cdot,t) : \mathbb{R}^{d_\omega} \to \mathbb{R}^{d_u} : t \in [0,1)\}$ is equicontinuous, i.e. if $\omega_n \to \omega$ then for any $\varepsilon > 0$ there exists $N \in \mathbb{Z}_{\geq 0}$ such that for all $t \in [0,1)$ and $n \geq N$ it holds that $\|\alpha(\omega_n,t)-\alpha(\omega,t)\| < \varepsilon$.

II) The function f is continuously differentiable in (x,u), and solutions to (3.3) *exist in the sense of* (4.14) *for $t \in \mathbb{R}_{\geq 0}$, for all $x \in X$ and all measurable and essentially bounded controls $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^{d_u}$.*

Then the flow φ associated to the ODE (3.3) *satisfies the assumptions of Theorem 1.*

Recalling the definition of u_ω in Theorem 1, Assumption I implies that u_ω is continuous from the right at $t = 0$ and that $u_{\omega_n} \to u_\omega$ uniformly when $\omega_n \to \omega$. Assumption II above is sometimes referred to as forward completeness of (3.3). There is no single condition on f that can guarantee forward completeness; examples of possible conditions are discussed in (Sontag, 1998; Angeli and Sontag, 1999). It implies the existence of φ satisfying (4.14) for all $t \geq 0$, and that $\mathbb{U}$ can be chosen to be the set of all measurable essentially bounded functions $u : \mathbb{R}_{\geq 0} \to \mathbb{R}^{d_u}$.

Proof. First, we show that Assumption 1 in Theorem 1 holds. Let $K_\omega \subset \mathbb{R}^{d_\omega}$ be a compact set, and let $u \in \mathbb{U}(K_\omega)$ be parameterised by the sequence (ω_k). Let $a_k := \sup_{t\in[0,1]} \|\alpha(\omega_k, t)\|$. Suppose (a_k) is unbounded, and pick a subsequence (a_{k_j}) such that $\lim_{j\to\infty} a_{k_j} = \infty$. By compactness, there is a subsequence $(\omega_{k'_j})$ of (ω_{k_j}) with $\lim_{j\to\infty} \omega_{k'_j} = \omega \in K_\omega$. Then, by equicontinuity we have that for j large enough

$$\left\|\alpha(\omega_{k'_j}, t)\right\| < 1 + \|\alpha(\omega, t)\|\,,\ t \in [0, 1),$$

which implies that $(a_{k'_j})$ is bounded, a contradiction. Hence a_k is bounded, and so u is bounded. Since α is measurable, so is u, and thus $u \in \mathbb{U}$, as desired.

We now show that Assumption 2 in Theorem 1 holds. Since α is right-continuous at $t = 0$, (4.15) gives

$$\begin{aligned}\left.\frac{\mathrm{d}}{\mathrm{d}\tau}\right|_{\tau=0} \Phi(\tau, x, \omega) &= \left.\frac{\mathrm{d}}{\mathrm{d}\tau}\right|_{\tau=0} \varphi(\Delta\tau, x, u_\omega) \\ &= f(\varphi(0, x, u_\omega), u_\omega(0))\Delta \\ &= f(x, \alpha(\omega, 0))\Delta,\end{aligned}$$

and hence Φ is differentiable from the right at $\tau = 0$, as desired.

Finally, we show that Assumption 3 in Theorem 1 holds. Let

$$\Psi_0(\tau, x, \omega) = \Psi(\tau, x, \omega) - x.$$

We will show that the differential of Ψ_0 with respect to x is continuous, and thus bounded on compact sets, from which it follows that Ψ_0 (and thus Ψ) is locally Lipschitz. The remainder of the proof requires a few additional properties of the flow φ which we state in the following sublemmata.

Sublemma 1. *Let $(\omega_n) \subset \mathbb{R}^{d_\omega}$ and $(x_n) \subset X$ be such that $\omega_n \to \omega$ and $x_n \to x \in X$. Then $\varphi(t, x_n, u_{\omega_n}) \to \varphi(t, x, u_\omega)$ uniformly in $t \in [0, \Delta]$.*

Proof. By Assumption I on the function α we have that $u_{\omega_n} \to u_{\omega_n}$ uniformly, and thus the result follows from (Sontag, 1998, Theorem 1). □

Sublemma 2. *The flow φ is differentiable with respect to the initial condition x and and its differential with respect to x, $D_x\varphi$, satisfies*

$$D_x\varphi(t, x, u)\xi = \lambda_{x,u}(t; \xi)$$

for all $\xi \in \mathbb{R}^n$*, where* $\lambda_{x,u}$ *is the solution of the linear initial value problem*

$$\begin{aligned} \dot{\lambda}_{x,u}(s;\xi) &= D_x f(\varphi(s,x,u),u(s))\lambda(s;\xi) \\ \lambda_{x,u}(0;\xi) &= \xi. \end{aligned} \tag{4.16}$$

Equivalently, $D_x\varphi(t,x,u) = \Lambda_{x,u}(t)$ *where* $\Lambda_{x,u}$ *is the state transition matrix associated to the linear system* (4.16)*.*

Proof. See (Sontag, 1998, Theorem 1). ☐

We also need the following result on the continuity of solutions of linear ODEs with respect to the coefficient matrix.

Sublemma 3. *Let* $x, z : [t_1, t_2] \to \mathbb{R}^{d_x}$ *satisfy*

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) \\ \dot{z}(t) &= B(t)z(t) \end{aligned} \qquad t \in (t_1, t_2)$$

with $A, B : [t_1, t_2] \to \mathbb{R}^{d_x \times d_x}$ *measurable and essentially bounded. Then, with* $d(t) = x(t) - z(t)$ *and* $D(t) := A(t) - B(t)$*,*

$$\|d(t)\| \le \left(\|d(t_1)\| + \int_{t_1}^{t} \|D(s)\|\,\|x(s)\|\,\mathrm{d}s\right)\mathrm{e}^{\int_{t_1}^{t}\|B(s)\|\mathrm{d}s} \tag{4.17}$$

for $t \in [t_1, t_2]$*.*

Proof. Write

$$d(t) = d(t_1) + \int_{t_1}^{t} [B(s)d(s) + (A(s) - B(s))x(s)]\,\mathrm{d}s,$$

so that

$$\|d(t)\| \le \|d(t_1)\| + \int_{t_1}^{t} \|B(s)\|\,\|d(s)\|\,\mathrm{d}s + \int_{t_1}^{t} \|D(s)\|\,\|x(s)\|\,\mathrm{d}s,$$

and Grönwall's inequality gives the desired result. ☐

We shall use the inequality (4.17) to show that Λ_{x,u_ω} is continuous with respect to (x,ω). To this end, let $x_n \to x$ and $\omega_n \to \omega$ and, for $t \in [0,\Delta]$, set

$$A(t) = D_x f(\varphi(t,x,u_\omega), u_\omega(t)), \tag{4.18}$$

$$B_n(t) = D_x f(\varphi(t,x_n,u_{\omega_n}), u_{\omega_n}(t)). \tag{4.19}$$

By continuity of φ and α, there exist compact sets K'_x and K'_u such that $\varphi(t,x,u_\omega)$ and $\varphi(t,x_n,u_{\omega_n})$ remain in K'_x and $u_\omega(t), u_{\omega_n}(t) \in K'_u$ holds for all n and $t \in [0,\Delta]$. Let

$$\bar{F} := \sup\{\|D_x f(z,u)\| :\ z \in K'_x,\ u \in K'_u\}. \tag{4.20}$$

We have $\bar{F} < \infty$, by continuity of $D_x f$, and $\|B(s)\| \leq \bar{F}$ for $s \in [0, \Delta]$. Sublemma 3 applied on the interval $[0, t]$ now gives

$$\begin{aligned}\left\|\lambda_{x,u_\omega}(t;\xi) - \lambda_{x_n,u_{\omega_n}}(t;\xi)\right\| &\leq \mathrm{e}^{t\bar{F}} \int_0^t \|A(s) - B_n(s)\| \, \|\lambda_{x,u_\omega}(s;\xi)\| \,\mathrm{d}s \\ &\leq \mathrm{e}^{t\bar{F}} \sup_{s\in[0,\Delta]} \|\Lambda_{x,u_\omega}(s)\| \left(\int_0^t \|A(s) - B_n(s)\| \,\mathrm{d}s\right) \|\xi\|,\end{aligned}$$

so that

$$\left\|\Lambda_{x,u_\omega}(t) - \Lambda_{x_n,u_{\omega_n}}(t)\right\| \leq \mathrm{e}^{t\bar{F}} \sup_{s\in[0,\Delta]} \|\Lambda_{x,u_\omega}(s)\| \int_0^t \|A(s) - B_n(s)\| \,\mathrm{d}s \tag{4.21}$$

for each t. By continuity of $D_x f$ and the uniform convergence of $\varphi(t, x_n, u_{\omega_n})$ and u_{ω_n}, B_n converges uniformly to A, and thus $\Lambda_{x_n,u_{\omega_n}} \to \Lambda_{x,u_\omega}$ uniformly on $[0, \Delta]$.

Returning to our original goal, for $\tau > 0$ we have

$$\begin{aligned}D_x\Psi_0(\tau, x, \omega) &= \tau^{-1}(D_x\Phi(\tau, x, \omega) - I) \\ &= \tau^{-1}(D_x\varphi(\tau\Delta, x, u_\omega) - I) \\ &= \tau^{-1}(\Lambda_{x,u_\omega}(\tau\Delta) - I).\end{aligned}$$

Due to the continuity of Λ_{x,u_ω}, $D_x\Psi_0$ is continuous in (τ, x, ω) for $\tau > 0$.

For $\tau = 0$ we have $D_x\Psi_0(0, x, \omega) = \Delta D_x f(x, u_\omega(0))$. If $\tau_n \to 0$ with $\tau_n > 0$, $x_n \to x$ and $\omega_n \to \omega$ then

$$\begin{aligned}D_x\Psi_0(0, x, \omega) - D_x\Psi_0(\tau_n, x_n, \omega) &= \Delta D_x f(x, u_\omega(0)) - \tau_n^{-1}\big(\Lambda_{x_n,u_{\omega_n}}(\tau_n\Delta) - I\big) \\ &= \Delta D_x f(x, u_\omega(0)) - \tau_n^{-1}(\Lambda_{x,u_\omega}(\tau_n\Delta) - I) \\ &\quad + \tau_n^{-1}(\Lambda_{x,u_\omega}(\tau_n\Delta) - \Lambda_{x_n,u_{\omega_n}}(\tau_n\Delta)).\end{aligned}$$

The first term of the last equality goes to zero, hence we are left to investigate the second term. With $A, B_n, \bar{F}$ defined in (4.18)-(4.20), from (4.21) we find

$$\begin{aligned}&\left\|\tau_n^{-1}(\Lambda_{x,u_\omega}(\tau_n\Delta) - \Lambda_{x_n,u_{\omega_n}}(\tau_n\Delta))\right\| \\ &\quad\leq \mathrm{e}^{\tau_n\Delta\bar{F}} \sup_{t\in[0,\Delta]} \|\Lambda_{x,u_\omega}(t)\| \frac{1}{\tau_n} \int_0^{\tau_n\Delta} \|A(t) - B_n(t)\| \,\mathrm{d}t.\end{aligned}$$

Pick $\varepsilon > 0$, and let n be large enough that $\|A(t) - B_n(t)\| < \varepsilon$ for $t \in [0, \Delta]$, so that

$$\tau_n^{-1} \left\|\Lambda_{x,u_\omega}(\tau_n\Delta) - \Lambda_{x_n,u_{\omega_n}}(\tau_n\Delta)\right\| \leq \left(\Delta\mathrm{e}^{\tau_n\Delta\bar{F}} \sup_{t\in[0,\Delta]} \|\Lambda_{x,u_\omega}(t)\|\right) \varepsilon$$

and $\tau_n^{-1}(\Lambda_{x,u_\omega}(\tau_n\Delta) - \Lambda_{x_n,u_{\omega_n}}(\tau_n\Delta)) \to 0$ as $n \to \infty$, as desired.

□

4.6 Summary

We have shown that the RNN architecture proposed in Chapter 2 is a universal approximator of flow functions of continuous-time control systems. The considered parameterisation of the control inputs, from which the discrete structure of the flow emerges, motivated the RNN architecture. We proved the main result, Theorem 1, for general control systems, under smoothness assumptions on the flow function. We then showed that these assumptions hold in the important case of flows of control systems with dynamics given by controlled ODEs, under the condition that the equation is continuously differentiable.

4.7 Appendix

Proof of Theorem 2

The case $N = 0$ is trivial, and $N = 1$ corresponds to the standard universal approximation theorem proved in Hornik (1991), so we assume $N \geq 2$ in what follows.

Define the sets $K_x^0, \dots, K_x^{N-1}$ recursively by $K_x^{n+1} = f(K_x^n, K_u)$ with $K_x^0 = K_x$. By continuity of f, the K_x^n are compact. For any input sequence $u \in S(K_u)$ and initial state $x_0 \in K_x$, we then have that

$$\rho_f(n, x_0, u) \in K_x^n,\ n = 0, 1, \dots, N-1.$$

Define also $L_f^n, \eta_f^n \geq 0$ and sets $\tilde{K}^n$, $n = 1, \dots, N-1$ recursively as follows:

$$\begin{aligned} \eta_f^1 &= 1 \\ \tilde{K}^n &= N_{\varepsilon \eta_f^n}(K_x^n) \\ L_f^n &= \max\left\{1, \sup_{u \in K_u} \nu_{\tilde{K}^n}(u)\right\} \\ \eta_f^{n+1} &= 1 + L_f^n \eta_f^n, \end{aligned}$$

and let

$$\begin{aligned} K &= K_x \cup \bigcup_{n=1}^{N-1} \tilde{K}^n \\ \varepsilon_n &= \frac{1}{2^{N-n} \prod_{k=n}^{N-1} L_f^k} \varepsilon,\ n = 1, \dots, N. \end{aligned}$$

Pick a neural network $g \in \mathfrak{N}_\sigma^{d_x + d_u, d_x}$ such that

$$\sup_{x \in K, u \in K_u} \| f(x,u) - g(x,u) \| < \min_{n=1,\dots,N} \varepsilon_n. \tag{4.22}$$

In particular, $\sup_{x \in K, u \in K_u} \| f(x,u) - g(x,u) \| < \varepsilon$.

Now, pick $x \in K_x$ and $u \in S(K_u)$. We have (omitting the (x, u) arguments since they are fixed everywhere)

$$\begin{aligned}\|\rho_f(n+1, x, u) - \rho_g(n+1, x, u)\| &= \|f(\rho_f(n)) - g(\rho_g(n))\| \\ &\leq \|f(\rho_f(n)) - f(\rho_g(n))\| + \|f(\rho_g(n)) - g(\rho_g(n))\|\end{aligned}$$

Assuming that $\|\rho_f(n) - \rho_g(n)\| < \varepsilon\eta_f^n$, we have $\rho_g(n) \in \tilde{K}^n \subset K$ and so

$$\begin{aligned}\|\rho_f(n+1) - \rho_g(n+1)\| &< L_f^n \|\rho_f(n) - \rho_g(n)\| + \varepsilon \\ &\leq \varepsilon\eta_f^{n+1}.\end{aligned}$$

Since (4.22) implies

$$\|\rho_f(1, x, u) - \rho_g(1, x, u)\| < \varepsilon \, (= \varepsilon\eta_f^1),$$

by induction we have that $\|\rho_f(n, x, u) - \rho_g(n, x, u)\| < \varepsilon\eta_f^n$ for $n = 1, \ldots, N-1$, so that $\rho_g(n, x, u) \in \tilde{K}^n$.

Now, we show by induction that

$$\|\rho_f(n, x, u) - \rho_g(n, x, u)\| < \varepsilon_n$$

for each $n \geq 0$. For $n = 1$ this holds by (4.22):

$$\|\rho_f(1, x, u) - \rho_g(1, x, u)\| = \|f(x, u_0) - g(x, u_0)\| < \varepsilon_1.$$

Assume that $\|\rho_f(n, x, u) - \rho_g(n, x, u)\| < \varepsilon_n$. Then

$$\begin{aligned}\|\rho_f(n+1, x, u) - \rho_g(n+1, x, u)\| &\leq \|f(\rho_f(n)) - f(\rho_g(n))\| + \|f(\rho_g(n)) - g(\rho_g(n))\| \\ &< L_f^n \varepsilon_n + \varepsilon_n \\ &= \frac{\varepsilon_{n+1}}{2} + \varepsilon_n \\ &\leq \varepsilon_{n+1}.\end{aligned}$$

And since $\varepsilon_n \leq \varepsilon$ for $n = 1, \ldots, N$, we have that $\|\rho_f(n, x, u) - \rho_g(n, x, u)\| < \varepsilon$, as desired.

Since it is not necessarily the case that $g \in \mathfrak{N}_{\sigma,p}^0$ for some p, it remains to obtain an equivalent recurrent neural network. Write g explicitly as

$$g(x, u) = T\sigma_p(Ax + Bu + b) + c,$$

and rank-factorise T as

$$T = M \begin{bmatrix} T_1 \\ 0 \end{bmatrix}$$

with $M \in \mathbb{R}^{d_x \times d_x}$ invertible and $T_1 \in \mathbb{R}^{r \times p}$ of full row rank. Then, with

$$g_1(x, u) := \begin{bmatrix} T_1\sigma_p(AMx + Bu + b) + c_1' \\ c_2' \end{bmatrix}, \; M^{-1}c = \begin{bmatrix} c_1' \\ c_2' \end{bmatrix},$$

it follows that

$$M\rho_{g_1}(n, M^{-1}x, u) = \rho_g(n, x, u)$$

for all (n, x, u). Let now T_1^+ be a right inverse of T_1 (i.e. $T_1 T_1^+ = I_r$) and

$$Q := M \begin{bmatrix} T_1 & 0 \\ 0 & I_{d_x - r} \end{bmatrix}, \; Q^+ := \begin{bmatrix} T_1^+ & 0 \\ 0 & I_{d_x - r} \end{bmatrix} M^{-1}.$$

Then with

$$g_2(z, u) := \begin{bmatrix} \sigma_p(AQz + Bu + b) + T_1^+ c_1' \\ c_2' \end{bmatrix},$$

we get

$$Q\rho_{g_2}(n, Q^+x, u) = \rho_g(n, x, u).$$

Finally, let

$$\tilde{A} := \begin{bmatrix} AQ \\ 0_{(d_x - r)\times(p + d_x - r)} \end{bmatrix}, \; \tilde{B} := \begin{bmatrix} B \\ 0_{(d_x - r)\times d_u} \end{bmatrix},$$

$$\tilde{b} := \begin{bmatrix} b \\ 0_{d_x - r} \end{bmatrix}, \; \tilde{c} := \begin{bmatrix} T_1^+ c_1' \\ c_2' - \sigma_{d_x - r}(0) \end{bmatrix}$$

and define the maps $\gamma : \mathbb{R}^{p + d_x - r} \to \mathbb{R}^{d_x}$ and $\beta : \mathbb{R}^{d_x} \to \mathbb{R}^{p + d_x - r}$ as

$$\gamma(z) = Q(z + \tilde{c})$$
$$\beta(x) = Q^+ x - \tilde{c}.$$

Then with $d_z := p + d_x - r$ and

$$h(z, u) := \sigma_{d_z}(\tilde{A}z + \tilde{B}u + \tilde{b} + \tilde{A}\tilde{c})$$

we get

$$\gamma(\rho_h(n, \beta(x), u)) = \rho_g(n, x, u),$$

so that $h \in \mathfrak{N}^0_{\sigma, d_z}$ and

$$\|\gamma(\rho_h(n, \beta(x), u)) - \rho_f(n, x, u)\| < \varepsilon$$

for all $x \in K_x$, $u \in S(K_u)$ and $n = 0, \ldots, N$, as desired. Note also that γ and β have the desired properties.

Chapter 5

Conclusions

5.1 Summary

In this thesis, we have studied the problem of learning flow functions of continuous-time control systems from measurements of state trajectories. First, we proposed a discrete-time recurrent neural network (RNN) architecture for approximating the flow function. Considering classes of control inputs with a discrete structure, corresponding to inputs typically used in practical applications, we showed that these inputs induce a discrete structure in the flow function, and established an exact representation of the flow function as a discrete-time dynamical system. We thus proposed to approximate this discrete-time system by an architecture composed by an RNN together with a pair of encoder-decoder networks. We evaluated the architecture experimentally on data from the Van der Pol and FitzHugh-Nagumo nonlinear oscillators. On both systems, models trained using our method were able to accurately reproduce the state trajectories, and the training procedure was shown to be robust to initialisation and to the choice of the architecture hyperparameters. For the Van der Pol oscillator model, we further investigated the simulation capabilities of the trained model for long time horizons (relative to the length of the training data) and for a different distribution of the control inputs. We also showed that the training procedure is robust to measurement noise, and studied the dependence of the test loss on the number of measurements. For the FitzHugh-Nagumo oscillator model, we demonstrated that the trained model was able to capture the excitable behaviour of the system.

Secondly, we considered an application to surrogate modelling of spiking systems. We used conductance-based models as prototypical state-space realisations of spiking behaviour, and proposed a framework for constructing surrogate models of these systems from trajectory data, based on our flow function learning formulation. We highlighted two challenges arising in training simulation models for these systems. First, as the output signal contains high-frequency components, many samples are required in order to represent it accurately, leading to a data-heavy training procedure, and possibly to under-representation of the spikes in the dataset. To address this, we proposed a data reduction method based on rejection sampling, equivalent to giving higher weight in the loss function to those parts of the trajectories which contain spikes. Second, we considered the complexity of optimising the empirical loss. For recurrent models, it is known that the Lipschitz constant of the loss function with respect to the model parameters increases exponentially with the simulation length. We addressed this by using the semigroup property of the flow function to create a windowed loss function using shorter segments of the measured trajectories. We evaluated our methodology on two systems, namely a model of a fast spiking neuron, and a model of two regular spiking with adaptation type neurons interconnected in feedforward through an electrical synapse. In both cases, the trained models are able to closely reproduce the spiking behaviour, demonstrating the feasibility of our approach for constructing surrogate models of spiking systems.

Finally, we studied the universal approximation of flow functions of continuous-time control systems by discrete-time RNNs. Considering the architecture we had proposed earlier, we derived conditions on the flow function of a general continuous-time control system such that it can be approximated arbitrarily well by elements of the hypothesis space defined by the architecture. We then specialised to the case of flows of systems given by controlled ordinary differential equations. Namely, for systems with continuously differentiable dynamics, we showed that the conditions for universal approximation hold, for a broad class of input signals.

5.2 Future work

A number of avenues for developing the work we have presented here are in view. It would be interesting to study the application of flow function models in inverse problems and design optimisation. Inverse problem formulations can be used in parameter estimation, for instance, while design optimisation methods could be an interesting way to optimise controller parameters. Another application of the latter is in the design of a network of conductance-based neuron models so that it replicates a given desired behaviour, as described in Section 1.1.

A second application is learning observers for complex systems. Consider a system with spatiotemporal dynamics, e.g. a traffic flow model, from which we can sample low-dimensional measurements at a constant rate. An observer for this type of system can be regarded as a continuous-time control system with piecewise-constant inputs. Our formulation can thus be used to learn the flow function of such an observer, from data of measurement and state trajectories. In this connection, an interesting problem is that of learning an observer in this way from simulated data and deploying it on a real system.

The use of flow function models in continuous-time predictive control is another promising direction of research. Using the models we propose in this thesis allows for direct computation of the simulated state trajectory, so that the receding-horizon optimisation problem can be formulated without differential equation constraints.

Furthermore, automatic differentiation allows for the efficient computation of gradients of the state trajectory with respect to the control input. This application also suggests an experiment design problem, namely selecting the distribution of the inputs used for training the flow function model so that the model has good simulation accuracy under the inputs generated by the predictive controller in feedback.

Further experimental validation of the architecture is also in order, in particular its scalability with the number of states. A variety of extensions of the proposed architecture are in sight, such as to systems with spatiotemporal dynamics, possibly using techniques from neural operators for partial differential equations; estimating models from input-output data, for instance employing a subspace encoder similarly to what is done in (Beintema, Schoukens and Tóth, 2023); as well as considering other (non-parametric) classes of input signals.

Finally, a number of directions for further theoretical studies of the flow function learning problem can be considered. This includes extensions of our result in Chapter 4, using stability conditions to obtain approximation guarantees on unbounded time intervals, or obtaining bounds on the number of parameters, as in (Hanson and Raginsky, 2020). Another possible direction is the study of the sample complexity of the flow learning problem. Lastly, the results of Chapters 2 and 3 suggest that our architecture is capable of approximating systems without fading memory. The identification of systems without fading memory is a difficult and mostly open problem (Burghi, 2020); theoretical studies from this point of view and possible relations to stability of discrete-time recurrent neural networks would be valuable.

References

Aguiar, M., A. Das and K. H. Johansson (2023a). ‘Learning Flow Functions from Data with Applications to Nonlinear Oscillators’. In: *Proceedings of the 22nd IFAC World Congress.*

— (2023b). ‘Learning Flow Functions of Spiking Systems’. In: *arXiv e-prints* arXiv:2312.11913.

— (2023c). ‘Universal Approximation of Flows of Control Systems by Recurrent Neural Networks’. In: *2023 62nd IEEE Conference on Decision and Control (CDC).*

Almog, M. and A. Korngreen (2016). ‘Is Realistic Neuronal Modeling Realistic?’ In: *Journal of Neurophysiology.*

Angeli, D. and E. D. Sontag (1999). ‘Forward Completeness, Unboundedness Observability, and Their Lyapunov Characterizations’. In: *Systems & Control Letters.*

Arnol’d, V. I. (1991). *Ordinary Differential Equations.* MIT Press.

Atherton, D. P. (1987). ‘Simulation in Control System Design’. In: *Simulation of Control Systems.* Ed. by I. Troch, P. Kopacek and F. Breitenecker. IFAC Symposia Series.

Azizzadenesheli, K. et al. (2024). ‘Neural Operators for Accelerating Scientific Simulations and Design’. In: *arXiv e-prints* arXiv:2309.15325.

Beintema, G. I., M. Schoukens and R. Tóth (2023). ‘Continuous-Time Identification of Dynamic State-Space Models by Deep Subspace Encoding’. In: *The Eleventh International Conference on Learning Representations.*

Bevanda, P., S. Sosnowski and S. Hirche (2021). ‘Koopman Operator Dynamical Models: Learning, Analysis and Control’. In: *Annual Reviews in Control.*

Bhan, L., Y. Shi and M. Krstic (2023). ‘Operator Learning for Nonlinear Adaptive Control’. In: *Proceedings of The 5th Annual Learning for Dynamics and Control Conference.* PMLR.

Biloš, M. et al. (2021). ‘Neural Flows: Efficient Alternative to Neural ODEs’. In: *Advances in Neural Information Processing Systems.*

Bold, K. et al. (2003). ‘The Forced van der Pol Equation II: Canards in the Reduced System’. In: *SIAM Journal on Applied Dynamical Systems.*

Boyd, S. and L. Chua (1985). ‘Fading Memory and the Problem of Approximating Nonlinear Operators with Volterra Series’. In: *IEEE Transactions on Circuits and Systems.*

Brezinski, C. and L. Wuytack (2001). ‘Numerical Analysis in the Twentieth Century’. In: *Numerical Analysis: Historical Developments in the 20th Century.* Ed. by C. Brezinski and L. Wuytack. Elsevier.

Brunton, S. L., J. L. Proctor and J. N. Kutz (2016). ‘Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems’. In: *Proceedings of the National Academy of Sciences.*

Burghi, T. (2020). ‘Feedback for Neuronal System Identification’. PhD thesis. University of Cambridge.

Burghi, T. B., M. Schoukens and R. Sepulchre (2021). ‘Feedback Identification of Conductance-Based Models’. In: *Automatica.*

Cellier, F. E. and E. Kofman (2006). *Continuous System Simulation.* Kluwer Academic Publishers.

Chen, R. T. Q. et al. (2018). ‘Neural Ordinary Differential Equations’. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems.*

De Brouwer, E. et al. (2019). ‘GRU-ODE-Bayes: Continuous Modeling of Sporadically-Observed Time Series’. In: *Advances in Neural Information Processing Systems.*

DeWeerth, S. P. et al. (1991). ‘A Simple Neuron Servo’. In: *IEEE Transactions on Neural Networks.*

Drgoňa, J. et al. (2020). ‘All You Need to Know about Model Predictive Control for Buildings’. In: *Annual Reviews in Control.*

FitzHugh, R. (1961). ‘Impulses and Physiological States in Theoretical Models of Nerve Membrane’. In: *Biophysical Journal.*

Floryan, D. and M. D. Graham (2022). ‘Data-Driven Discovery of Intrinsic Dynamics’. In: *Nature Machine Intelligence.*

Forgione, M. and D. Piga (2021). ‘Continuous-Time System Identification with Neural Networks: Model Structures and Fitting Criteria’. In: *European Journal of Control.*

Frangos, M. et al. (2010). 'Surrogate and Reduced-Order Modeling: A Comparison of Approaches for Large-Scale Statistical Inverse Problems'. In: *Large-Scale Inverse Problems and Quantification of Uncertainty.* John Wiley & Sons, Ltd. Chap. 7.

Garnier, H. (2015). 'Direct Continuous-Time Approaches to System Identification. Overview and Benefits for Practical Applications'. In: *European Journal of Control* 24.

Geneva, N. and N. Zabaras (2022). 'Transformers for Modeling Physical Systems'. In: *Neural Networks.*

Ghadami, A. and B. I. Epureanu (2022). 'Data-Driven Prediction in Dynamical Systems: Recent Developments'. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences.*

Giannari, A. G. and A. Astolfi (2022). 'Model Design for Networks of Heterogeneous Hodgkin–Huxley Neurons'. In: *Neurocomputing.*

González, R. A. (2022). 'Continuous-Time System Identification : Refined Instrumental Variables and Sampling Assumptions'. PhD thesis. Kungliga Tekniska högskolan.

Guckenheimer, J., K. Hoffman and W. Weckesser (2003). 'The Forced van der Pol Equation I: The Slow Flow and Its Bifurcations'. In: *SIAM Journal on Applied Dynamical Systems.*

Hanson, J. and M. Raginsky (2019). 'Universal Approximation of Input-Output Maps by Temporal Convolutional Nets'. In: *Advances in Neural Information Processing Systems.*

— (2020). 'Universal Simulation of Stable Dynamical Systems by Recurrent Neural Nets'. In: *Proceedings of the 2nd Conference on Learning for Dynamics and Control.*

Hanson, J., M. Raginsky and E. Sontag (2021). 'Learning Recurrent Neural Net Models of Nonlinear Systems'. In: *Proceedings of the 3rd Conference on Learning for Dynamics and Control.*

Hardt, M. and B. Recht (2022). *Patterns, Predictions, and Actions: Foundations of Machine Learning.* Princeton University Press.

Hodgkin, A. L. and A. F. Huxley (1952). 'A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve'. In: *The Journal of Physiology.*

Holmes, P. (1990). 'Poincaré, Celestial Mechanics, Dynamical-Systems Theory and "Chaos"'. In: *Physics Reports.*

Hornik, K. (1991). 'Approximation Capabilities of Multilayer Feedforward Networks'. In: *Neural Networks.*

Indiveri, G. et al. (2011). ‘Neuromorphic Silicon Neuron Circuits’. In: *Frontiers in Neuroscience.*

Izhikevich, E. M. (2006). *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting.* The MIT Press.

Karniadakis, G. E. et al. (2021). ‘Physics-Informed Machine Learning’. In: *Nature Reviews Physics.*

Kidger, P. (2021). ‘On Neural Differential Equations’. PhD thesis. University of Oxford.

Kissas, G. et al. (2022). ‘Learning Operators with Coupled Attention’. In: *Journal of Machine Learning Research.*

Kovachki, N., S. Lanthaler and S. Mishra (2021). ‘On Universal Approximation and Error Bounds for Fourier Neural Operators’. In: *Journal of Machine Learning Research.*

Koziel, S., D. E. Ciaurri and L. Leifsson (2011). ‘Surrogate-Based Methods’. In: *Computational Optimization, Methods and Algorithms.* Ed. by S. Koziel and X.-S. Yang. Studies in Computational Intelligence. Springer.

Koziel, S. and L. Leifsson (2016). *Simulation-Driven Design by Knowledge-Based Response Correction Techniques.* Springer International Publishing.

Kratsios, A. (2021). ‘The Universal Approximation Property’. In: *Annals of Mathematics and Artificial Intelligence.*

Leifsson, L. and S. Koziel (2015). *Simulation-Driven Aerodynamic Design Using Variable-Fidelity Models.* Imperial College Press.

Li, X.-D., J. Ho and T. Chow (2005). ‘Approximation of Dynamical Time-Variant Systems by Continuous-Time Recurrent Neural Networks’. In: *IEEE Transactions on Circuits and Systems II: Express Briefs.*

Li, Z., N. B. Kovachki et al. (2021). ‘Fourier Neural Operator for Parametric Partial Differential Equations’. In: *International Conference on Learning Representations.*

Lin, G., C. Moya and Z. Zhang (2023). ‘Learning the Dynamical Response of Nonlinear Non-Autonomous Dynamical Systems with Deep Operator Neural Networks’. In: *Engineering Applications of Artificial Intelligence.*

Ljung, L. (1999). *System Identification: Theory for the User.* 2nd ed. Prentice Hall Information and System Sciences Series. Prentice Hall PTR.

Lu, L. et al. (2021). ‘Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators’. In: *Nature Machine Intelligence.*

Mead, C. A. (1990). ‘Neuromorphic Electronic Systems’. In: *Proceedings of the IEEE.*

Nelson, M. E. (2005). ‘Electrophysiological Models’. In: *Databasing the Brain: From Data to Knowledge Neuroinformatics.* Ed. by Koslow, Stephen H. and Subramanian, Shankar. Wiley-Liss.

Nghiem, T. X. et al. (2023). ‘Physics-Informed Machine Learning for Modeling and Control of Dynamical Systems’. In: *2023 American Control Conference (ACC).*

Otter, M. and F. E. Cellier (2000). ‘Software for Modeling and Simulating Control Systems’. In: *Control System Fundamentals.* Ed. by Levine, William S.

Pillonetto, G. et al. (2023). ‘Deep Networks for System Identification: A Survey’. In: *arXiv e-prints* arXiv:2301.12832.

Pintelon, R. and J. Schoukens (2012). *System Identification: A Frequency Domain Approach.* 2nd ed. Wiley IEEE Press.

Pospischil, M. et al. (2008). ‘Minimal Hodgkin–Huxley Type Models for Different Classes of Cortical and Thalamic Neurons’. In: *Biological Cybernetics.*

Qin, T. et al. (2021). ‘Data-Driven Learning of Nonautonomous Systems’. In: *SIAM Journal on Scientific Computing.*

Rahman, A. et al. (2022). ‘Neural Ordinary Differential Equations for Nonlinear System Identification’. In: *2022 American Control Conference (ACC).*

Raissi, M., P. Perdikaris and G. E. Karniadakis (2019). ‘Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations’. In: *Journal of Computational Physics.*

Ribar, L. and R. Sepulchre (2021). ‘Neuromorphic Control: Designing Multiscale Mixed-Feedback Systems’. In: *IEEE Control Systems.*

Ribeiro, A. H. et al. (2020). ‘On the Smoothness of Nonlinear System Identification’. In: *Automatica.*

Ross, S. (2013). ‘Chapter 5 - Generating Continuous Random Variables’. In: *Simulation (Fifth Edition).* Academic Press.

Schäfer, A. M. and H. G. Zimmermann (2006). ‘Recurrent Neural Networks Are Universal Approximators’. In: *Artificial Neural Networks – ICANN 2006.* Ed. by S. D. Kollias et al. Lecture Notes in Computer Science. Springer.

Schoukens, J. and L. Ljung (2019). ‘Nonlinear System Identification: A User-Oriented Road Map’. In: *IEEE Control Systems Magazine.*

Schwenzer, M. et al. (2021). ‘Review on Model Predictive Control: An Engineering Perspective’. In: *The International Journal of Advanced Manufacturing Technology.*

Semeraro, C. et al. (2021). ‘Digital Twin Paradigm: A Systematic Literature Review’. In: *Computers in Industry.*

Sepulchre, R. (2022). ‘Spiking Control Systems’. In: *Proceedings of the IEEE.*

Sepulchre, R., G. Drion and A. Franci (2018). ‘Excitable Behaviors’. In: *Emerging Applications of Control and Systems Theory: A Festschrift in Honor of Mathukumalli Vidyasagar*. Ed. by R. Tempo, S. Yurkovich and P. Misra. 1st ed. Lecture Notes in Control and Information Sciences - Proceedings. Springer.

Sontag, E. D. (1992). ‘Neural Nets as Systems Models and Controllers’. In: *Seventh Yale Workshop on Adaptive and Learning Systems.*

— (1998). *Mathematical Control Theory.* Ed. by J. E. Marsden et al. Texts in Applied Mathematics. Springer.

Stiefel, K. M. and D. S. Brooks (2019). ‘Why Is There No Successful Whole Brain Simulation (Yet)?’ In: *Biological Theory.*

Tabuada, P. and B. Gharesifard (2022). ‘Universal Approximation Power of Deep Residual Neural Networks Through the Lens of Control’. In: *IEEE Transactions on Automatic Control.*

Veeravalli, T. and M. Raginsky (2023). ‘A Constructive Approach to Function Realization by Neural Stochastic Differential Equations’. In: *2023 62nd IEEE Conference on Decision and Control (CDC).*

Willems, J. C. (1972). ‘Dissipative Dynamical Systems Part I: General Theory’. In: *Archive for Rational Mechanics and Analysis.*

Yang, A. et al. (2022). ‘Input-to-State Stable Neural Ordinary Differential Equations with Applications to Transient Modeling of Circuits’. In: *Proceedings of The 4th Annual Learning for Dynamics and Control Conference.*