# A Behavioral Approach to Robust Machine Learning

**Max Revay,**
**BE (Hons 1) & BSc**

A thesis submitted in fulfillment
of the requirements of the degree of
Doctor of Philosophy

Australian Centre for Field Robotics
School of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

September 2021

# Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

**Max Revay**

September 2021

# Abstract

Machine learning is revolutionizing almost all fields of science and technology and has been proposed as a pathway to solving many previously intractable problems such as autonomous driving and other complex robotics tasks. While the field has demonstrated impressive results on certain problems, many of these results have not translated to applications in physical systems, partly due to the cost of system failure and partly due to the difficulty of ensuring reliable and robust model behavior. Deep neural networks, for instance, have simultaneously demonstrated both incredible performance in game playing and image processing, and remarkable fragility. This combination of high average performance and a catastrophically bad worst case performance presents a serious danger as deep neural networks are currently being used in safety critical tasks such as assisted driving.

In this thesis, we propose a new approach to training models that have built in robustness guarantees. Our approach to ensuring stability and robustness of the models trained is distinct from prior methods; where prior methods learn a model and then attempt to verify robustness/stability, we directly optimize over sets of models where the necessary properties are known to hold.

Specifically, we apply methods from robust and nonlinear control to the analysis and synthesis of recurrent neural networks, equilibrium neural networks, and recurrent equilibrium neural networks. The techniques developed allow us to enforce properties such as incremental stability, incremental passivity, and incremental $\ell_2$ gain bounds / Lipschitz bounds. A central consideration in the development of our model sets is the difficulty of fitting models. All models can be placed in the image of a convex set, or even $\mathbb{R}^N$, allowing useful properties to be easily imposed during the training procedure via simple interior point methods, penalty methods, or unconstrained optimization.

In the final chapter, we study the problem of learning networks of interacting models with guarantees that the resulting networked system is stable and/or monotone, i.e., the order relations between states are preserved. While our approach to learning in this chapter is similar to the previous chapters, the model set that we propose has a separable structure that allows for the scalable and distributed identification of large-scale systems via the alternating directions method of multipliers (ADMM).

# Acknowledgements

Firstly, I would like to thank Ian for his guidance and support. My interest in control, machine learning and mathematics is in no small part inspired by the insight and creativity that he displays when tackling complex problems. The ideas developed throughout this work would have remained in their infancy if not for Ian's help.

Ray, you have been a tremendous help and it has been a pleasure working with you. I hope that we can work together again in the future.

Mum, Dad, Alex, David, Murli, Penny, Aiden, Layla and Noodle. Thank you for your endless support and patience as I have navigated the windy roads of a Ph.D. Thankyou for feigning interest in my work. None of this would have been possible without you.

To the Cult of the Green Dots and the extended ACFR family: Felix, Felix, Jasper, Vera, Wei, James, Wilhelm, Jacob, Stu, Jackson, Tara, Jen, Johnny, Nic (and more...). I was told that a Ph.D. is supposed to be hard and stressful. It turns out that with the right people, it is also really fun.

Nic, Jack, Sam, Marcus, and Calv, thank you for the endless boardgames, flips and wholesome hangs. Calvin, thanks for dinner; keep flipping, I hear its zero EV.

Ali and Marley, thank you for keeping me sane during timeless lockdowns and for your patience as I finished my thesis and if Putin manages to steal all the algebra, so be it.

*Sometimes science is more art than science.*
*- Richard D. Sanchez III*

# Contents

# Nomenclature

## List of Common Sets

| | |
|---|---|
| $\mathbb{R}$ | The set of real numbers |
| $\mathbb{R}^n$ | The set of real valued $n$-dimensional vectors |
| $\mathbb{R}^{n\times m}$ | The set of real valued $n$ by $m$ dimensional matrices |
| $\mathbb{S}^n$ | The set of $n$ by $n$, symmetric matrices |
| $\ell_{2e}^n$ | The set of $n$-dimensional sequences |
| $\ell_2^n$ | The set of square summable $n$-dimensional sequences |
| $\mathbb{D}$ | The set of diagonal matrices |
| $\mathbb{D}_+$ | The set of diagonal, positive definite matrices |

## List of Symbols

| | |
|---|---|
| $A \succ B$ | $A - B$ is positive definite |
| $A \succeq B$ | $A - B$ is positive semidefinite |
| $A \prec B$ | $A - B$ is negative definite |
| $A \preceq B$ | $A - B$ is negative semidefinite |
| $A > B$ | $A - B$ is elementwise positive |
| $A < B$ | $A - B$ is elementwise negative |
| $\lvert v\rvert_p$ | the p-norm of $v$ |
| $\lvert v\rvert_M$ | shorthand for $v^\top M v$ |

## List of Acronyms

| | |
|---|---|
| ADMM | Alternating Directions Methods of Multipliers |
| aREN | Acyclic Recurrent Equilibrium Network |
| CCP | Closed, Convex, Proper |
| ci-RNN | Contracting Implicit Recurrent Neural Network |
| c-aREN | Contracting Acyclic Recurrent Equilibrium Network |
| c-REN | Contracting Recurrent Equilibrium Network |
| DNN | Deep Neural Network |
| GAM | Generalized Additive Model |
| IQC | Integral Quadratic Constraint |
| iQC | Incremental Quadratic Constraint |
| ISS | Input to State Stability |
| LBEN | Lipschitz Bounded Equilibrium Network |
| LMI | Linear Matrix Inequality |
| LMT | Lipschitz Margin Training [216] |
| LSTM | Long Short Term Memory [94] |
| LTI | Linear Time Invariant |
| MON | Monotone Operator equilibrium Network [238] |
| NARX | Nonlinear AutoReggressive model with eXogenous input |
| NEE | Normalized Equation Error |
| NSE | Normalized Simulation Error |
| NRMSE | Normalized Root Mean Squared Error |
| ODE | Ordinary Differential Equation |
| PDE | Partial Differential Equation |
| ReLU | Rectified Linear Unit |
| REN | Recurrent Equilibrium Network |
| R-REN | Robust Recurrent Equilibrium Network |
| RNN | Recurrent Neural Network |
| SDP | Semidefinite Program |
| SNR | Signal to Noise Ratio |

# Chapter 1

# Introduction

## 1.1 Learning Stable and Robust Models

Machine learning and related fields such as system identification and reduced order modelling study the automated extraction of patterns from data. The goal is to construct models that are able to explain the observed data and then make predictions when exposed to new data. Recently, the increasing power of computational resources and availability of data has revolutionized the field of machine learning, and in particular deep learning, and led to unexpected and astounding progress in areas such as image classification [87], game playing [196], robotics [121] and it is expected to impact many more fields of science and engineering.

The fundamental object of interest in deep learning is the deep neural network: a model class whose apparent flexibility, accuracy, and scalability has led to renewed interest in neural networks, and more generally learning and data-driven methods in control, identification, and related areas [246, 121, 56].

On the other hand, it has been observed that neural networks can be very sensitive to small input perturbations in, e.g., image classification [209], reinforcement learning [180], language modelling [41], and many more. This type of extreme sensitivity is not limited to contrived adversarially chosen examples; operations such as image

compression, cropping, and resizing have been shown to destroy classifier performance [255]. These issues present a significant risk when deploying models as components in safety-critical systems such as autonomous driving [29], aircraft collision avoidance [102] and surgical robotics [142]. As noted by a recent RAND Corporation report on the state of AI [210]:

*"The current state of AI verification, validation, test, and evaluation (VVT&E) is nowhere close to ensuring the performance and safety of AI applications, particularly where safety-critical systems are concerned."*

The apparent brittleness of neural networks has inspired a considerable amount of research into neural network verification. This is the problem of finding formal guarantees that a bounded perturbation will not destroy a model's performance. There are many types of verification that have been developed for different problems. For instance, [239, 59, 67] construct convex bounds on network outputs subject to convex perturbation sets, [112, 148] verify the stability of recurrent models, [245] verify the stability of neural network controllers and [66] certify bounds on the Lipschitz constant of neural networks. See [240] for a thorough survey on the topic.

This brittleness of learned models is not a problem unique to deep learning. For instance, the sensitivity and instability of model structures is a well-known problem in the system identification community. Furthermore, the problem of training models with stability or robustness guaranteed *a priori* is significantly harder than the verification problem, and many methods have been developed for training stable linear [131, 117, 147] and nonlinear models [214, 219, 110].

Current approaches to robust learning first train a model and then attempt to verify certain properties of the learned model. Few researchers have considered the problem of specifying these properties *a-priori*, and then training a model within the constraints of that specification. Ensuring such properties a-priori is useful for a number of reasons:

1. Firstly, there is ample empirical evidence suggesting that joint learning of models and their stability or robustness certificates can improve generalization

[80, 176, 221, 180].

2. Secondly, there are many situations where it is essential that a learned model satisfies some property, e.g., stability in learned observers, or monotonicity for models that are to be used in conjunction with control strategies that rely on monotonicity [173, 184]. One could learn a black-box model and then attempt to verify this property, but if it does not satisfy the property then it is not clear how to correct this. Learning models with the property guaranteed fixes this issue.

3. Finally, in situations such as on-line adaptation or continual life-long learning, it is necessary that a learned module will not destabilize a system with which it is interacting. This can be achieved by learning models that are certified to satisfy the conditions of the small gain or passivity theorems, or by ensuring that the interconnected system has a stability certificate.

The main focus of this thesis is bridging the gap between certifying model robustness and training models with robustness certified *a priori*. Our interest stems from control applications; we aim to develop tools for training learning enabled components for control systems that are gauranteed to maintain system robustness. Some essential characteristics of such methods are:

1. Flexibility: Trained models should be as expressive as possible while still satisfying the specified robustness constraints.

2. Tractability: The problem of training a model with a robustness certificate should not be significantly harder than training a model without a robustness certficiate.

3. Compatability: The methods developed should be compatible with other tools that are commonly used in control engineering.

Understanding properties such as stability and sensitivity of interconnections of (possibly nonlinear) systems has been a long-standing problem in control theory [252]

and many methods have been developed including: passivity methods in robotics [86], networked systems analysis via dissipation inequalities [10], $\mu$ analysis [256] a number of other alternate methods for nonlinear control [225]. These approaches are unified under the the theory of integral quadratic constraints (IQC) [145].

Recently, the IQC framework has been applied to the problem of certifying a neural network's Lipschitz constant to achieve state of the art results [66, 67], however, as mentioned above, IQC theory has many further connections to methods used in robust control. Consequently, we believe that there is significant utility in further extending the IQC framework to both neural network analysis and synthesis.

### Contributions

The work in this thesis can be roughly divided into two sections. In chapters 3 to 5, we study the problem of training neural networks with certain properties guaranteed a priori. The particular properties certified are closely related to those studied in IQC theory and include: stability, incremental gain bounds (a.k.a. Lipschitz bounds), and incremental passivity. Our approach achieves the same quality of bounds as [66] (i.e. the current state of the art), however, it applies to a much larger class of neural networks including feedforward, recurrent and a recently proposed model class known as *equilibrium networks* [14, 15, 238]. Additionally, our approach leads to simpler constraints, allowing them to be easily imposed during training via constrained optimization, or in some cases via unconstrained optimization schemes.

In Chapter 6, we study the problem of simultaneously learning multiple interacting models and a stability and monotonicity certificates for those interacting models. The method developed applies to a broad class of models that includes the neural networks studied in the previous chapters.

A complete list of contributions for each chapter is as follows:

- In chapter 3, we propose a new convex parameterization of recurrent neural networks satisfying stability and robustness conditions. The stability and robustness conditions are based on incremental quadratic constraints and are less

conservative than prior methods. We then show that the proposed model is more expressive than previous model sets and contains all previously published sets of stable RNNs and all stable linear time-invariant (LTI) systems.

By using an implicit parameterization, we ensure that the parameterization is jointly convex in the model parameters, stability certificate, and the multipliers associated with the incremental quadratic constraint. This allows us to train models using an interior point method. Numerical experiments suggest that these constraints are beneficial for model robustness, generalizability, and lead to tight bounds on the Lipschitz constant.

- In chapter 4, we study the problem of learning static neural network mappings with guaranteed Lipschitz bounds. The results are developed in the context of a new class of neural networks: *equilibrium networks*, however, it is easy to show that this class contains many standard neural network structures. Consequently, the results generalize the incremental quadratic constraint approach of [66] to the analysis of a much larger class of neural networks. We show that the proposed model class permits a so-called direct parameterization, I.e., the model can be parameterized by a vector in $\mathbb{R}^n$. This allows for the training of Lipschitz bounded equilibrium networks via unconstrained gradient descent with almost no extra computational cost required. In addition, we prove the existence and uniqueness of solutions to the equilibrium network equation with less restrictive conditions on the weight matrix and more natural assumptions on the activation functions via novel connections to convex optimization and contracting dynamical systems. Finally, we show via small-scale image classification experiments that the proposed parameterizations can provide significant improvement in robustness to adversarial attacks with little degradation in nominal accuracy. Furthermore, we observe small gaps between certified Lipschitz upper bounds and observed lower bounds computed via adversarial attack.

- In chapter 5, we introduce a new model class, the *recurrent equilibrium network* (RENs). All RENs are contracting – a strong form of nonlinear stability – and the models can satisfy prescribed incremental integral quadratic constraints

(IQC), including Lipschitz bounds and incremental passivity. We show that the set of RENs contains all models developed in chapters 3 and 4 as special cases. Additionally, unlike the sequence-to-sequence mappings from Chapter 3, the REN permits a direct parameterization, significantly simplifying model training. The performance and robustness of the new model set is evaluated on benchmark nonlinear system identification problems. We also present novel applications of stable and robust model sets in data-driven nonlinear observer design and control with stability guarantees.

- The approaches taken in chapters 3 to 5 have limited scalability due to the requirement of centralized computation and a centralized model. In chapter 6, we propose methods for learning large-scale networked models with guarantees that their interconnection will be stable and/or monotone, i.e., the order relations between states are preserved. The main challenges that we address are: simultaneously searching for model parameters and a certificate of stability, and scalability to networks with hundreds or thousands of nodes. We propose a model set that admits convex constraints for stability and monotonicity, and has a separable structure that allows distributed identification via the alternating directions method of multipliers (ADMM). The performance and scalability of the approach is illustrated on a variety of linear and nonlinear case studies, including a nonlinear traffic network with a 200-dimensional state space.

## 1.2 Publications

The following is a list of publications whose content features in this thesis. Note that publication 4 is still under review.

Max Revay and Ian R. Manchester. Contracting implicit recurrent neural networks: Stable models with improved trainability. *L4DC*, 2020.

Max Revay, Ruigang Wang, and Ian R. Manchester. A convex parameterization of robust recurrent neural networks. *Control Systems Letters*, 2021.

Max Revay, Ruigang Wang, and Ian R. Manchester. Distributed identification of contracting and/or monotone network dynamics. *Transactions on Automatic Control*, 2021.

Max Revay, Ruigang Wang, and Ian R. Manchester. Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness. *Transaction on Automatic Control (Under Review)*, 2021.

Max Revay, Ruigang Wang, and Ian R. Manchester. Recurrent equilibrium networks: Unconstrained learning of stable and robust dynamical models. *Conference on Decision and Control*, 2021.

The following content of the following publication also appears in this thesis, however, it is currently hosted on arXiv with future plans for publication.

Max Revay, Ruigang Wang, and Ian R. Manchester. Lipschitz bounded equilibrium networks. *arXiv preprint arXiv:2104.05942*, 2021.

I was the second author for the following publication. While the experience I gained from working on this publication has helped shaped many of my ideas, no content explicitly appears in this thesis.

Humberto Stein Shiromoto, Max Revay, and Ian R. Manchester. Distributed nonlinear control design using separable control contraction metrics. *Transactions on Control of Networked Systems*, 2021.

# Chapter 2

# Background

In this chapter, we attempt to contextualize the contributions of this thesis. We do not attempt a complete literature review as the ideas developed are influenced by many areas and a complete review would be unwieldy. Instead, we provide an overview of our motivations and the foundational ideas that form the basis for each chapter.

This chapter is divided into five sections. Firstly, we provide a brief introduction to statistical learning theory with a focus on the importance of robustness, generalization, and regularization. We then introduce a taxonomy of model structures that we will study in the later chapters. In section 2.3, we motivate the need for incorporating known behavioral properties of a model into the training procedure. This is motivated from a regularization point of view as a means of improving model generalization and/or robustness. The properties discussed in Section 2.3 have a common structure, they can all be written as infinite dimensional inequality constraints that are intractable for many model classes. In Section 2.4 we present a number of techniques that can be used to convert the intractable inequality constraints into constraints that may be tackled via numerical optimization. Finally, in Section 2.5, we present a collection of numerical methods that can be used to fit models with the desired constraints.

## 2.1 Learning from Data

The term machine learning was first used in 1952 by Arthur Samuel, however, the foundational ideas can be traced back to early methods in statistics, e.g., Bayes' theorem and the least squares method in the early 19th century.

Today, the problem of learning models from data appears within many fields under different names. For instance, areas such as statistical learning theory, frequentist and Bayesian inference, deep learning, parameter estimation, system identification, and model reduction differ philosophically and in aspects such as data type and source, model structure, and objectives, however, all these methods amount to learning problems where a model is fit by optimizing a 'loss' function over a model set depending on data.

In this section, we provide an introduction to some core concepts in statistical learning theory.

#### Empirical Risk Minimization

Empirical risk minimization originates from statistical learning theory and provides a well-reasoned approach to solving learning problems. We assume that we have labeled data of the form $(u_1, y_1), ..., (u_N, y_N)$ where $u_i \in \mathcal{U}$ and $y_i \in \mathcal{Y}$ are the $i$th model input and output, respectively. This is a very general framework where the $\mathcal{U}$ and $\mathcal{Y}$ could be signals, text, images, labels, or numerical values. In this work, we are largely concerned with learning sequence-to-sequence mappings where $\mathcal{U}$ and $\mathcal{Y}$ are sequences. In Chapter 4, however, we will also consider the problem of image classification where $\mathcal{U}$ is the space of images and $\mathcal{Y}$ are class labels.

We assume that we have a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$, that compares the predicted and true outputs, and provides a score of how good that prediction was. In this thesis, we use perhaps the two most common loss functions: mean squared error and the cross entropy loss functions. The goal of supervised learning is to construct a model $\hat{y} = f(u)$ which produces accurate outputs when supplied with new data. We

can write the supervised learning problem succinctly as:

$$f^* = \underset{f\colon \mathcal{U} \mapsto \mathcal{Y}}{\arg\min} \ \mathbb{E}_{\ p(u,y)} \left[ \ell(f(u), y) \right], \tag{2.1}$$

where $p(u, y)$ is a fixed joint probability distribution from which the training data is sampled. In practice, (2.1) is intractable for two reasons: firstly, we do not know the distribution $p(u, y)$, and secondly, solving an optimization problem over the space of functions $f : \mathcal{U} \mapsto \mathcal{Y}$ is intractable. We make two simplifications to arrive at a tractable approximation of (2.1). Firstly, we approximate the expectation over $p(u, y)$ by sampling. Secondly, we select a parameterization $f := f_\theta$ where $\theta \in \Theta$. This leads to the following problem:

$$\theta^* = \underset{\theta \in \Theta}{\arg\min} \ \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(u_i), y_i). \tag{2.2}$$

Problem (2.2) is known as *empirical risk minimization* and is the basis of many supervised learning algorithms. A considerable advantage of this framework is that it allows for a principled approach to analyzing model generalization and convergence of learning algorithms.

We will explore some of the key concepts in learning using the cartoon shown in Figure 2.1. Here, we have represented the joint probability distribution $p(u, y)$ by the gray area, and the black points represent training samples drawn from the distribution $p(u, y)$. Problem (2.2) searches for a function $f_\theta$ that fits the training data (black dots). The learning problem, however, is ill-posed; there are many possible functions that could have generated the observed data, e.g., the red and blue lines. To overcome ill-posedness of empirical risk minimization, we can add further conditions specifying what makes a good model, e.g., we could specify that we want the smoothest model. We then hope that this model will continue to produce accurate predictions when new data is drawn from $p(u, y)$. We will now make some of these concepts more precise.

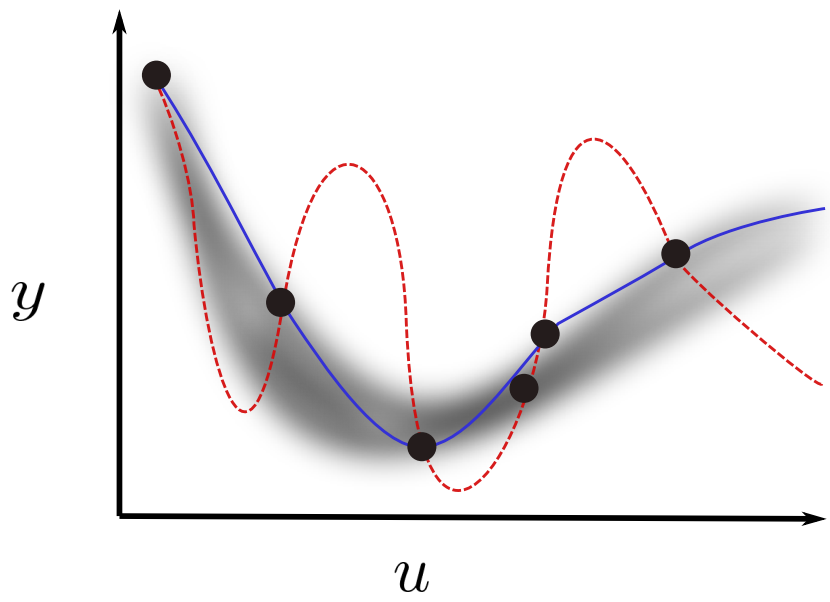



**Figure 2.1** – Here, we show two possible solutions to the same learning problem. The two solutions have the same loss however one is much simpler. In such cases regularization is used to choose the simplest solution. In this case, the black line.

**Generalization:**

As mentioned above, problem (2.1) is not tractable so instead we approximate it by Problem (2.2). The difference in loss between these two problems is often referred to as the *generalization gap*; ensuring that the generalization gap is small is a key problem in machine learning. Many approaches to analyzing generalization in statistical learning theory attempt to construct a generalization bound, i.e., a bound on (2.1) that depends on the solution to (2.2), the data and a measure of model complexity, e.g., [18]. Such bounds then allow a practitioner to reason about how much data is required to train effective models.

**Bias Variance Trade-off:**

Learning is an inherently noisy process as models are estimated from randomly sampled training data. When analyzing how a model generalizes, it is helpful to decompose the expected error into two components: the error resulting from the best model in our model set and the error due to mis-estimation of that model due to the random nature of the learning process. The tension between these two components is mediated by the model complexity, leading to a trade-off commonly known as the *bias-variance trade-off*.

Roughly speaking, when a model set is more expressive (complex), the best model in our model set can capture the true system behavior and lead to a good solution to (2.1). However, the solutions to (2.2) will have high variance and there will be a large generalization gap. This is the error due to variance. When the model set is less expressive, the results of Problem (2.2) will be closer to the best model in the model class, however, the model may not be expressive enough to capture the true behavior of the system, leading to a bias. When a model has too much error from bias, it has been *under-fit*. When the error due to variance outweighs the error due to bias, the model is said to have been *over-fit*. The need to balance these competing sources of error has motivated methods for controlling model complexity in a process known as regularization.

**Regularization:**

Regularization refers to a collection of techniques that are used to constrain or reduce model complexity as a means of balancing the bias-variance trade-off. Regularization also plays a distinct role in choosing between models with equivalent training set performance. Consequently, regularization techniques have a large effect on the generalization performance and the resulting model properties, often called an *inductive bias*.

Regularization can be seen as introducing a secondary optimization objective; within the set of models with small training error, find a simple model that will have a small generalization gap.

Techniques such as dropout [205], weight decay [114], Tikhonov/Ridge regression [60, Sec. 7], subset selection/regressor pruning in system identification [28, 200] and deep learning [95], Lipschitz regularization [80] encourage model simplicity by reducing model size or minimizing some measure that is closely related to model capacity [18, 104, 150, 77, 122]. Regularization is being used to discard models with undesirable properties, in this case, models that may have poor generalization performance.

While it is preferable to select models that have good generalization performance, there are many ways of achieving this and there are other properties that it is also useful or even necessary for a model to possess. For instance:

1. Robustness: Recent work has shown that deep learning models can be extremely sensitive to input perturbations [79]. In such cases it may be necessary to regularize the sensitivity of the neural network.

   A less obvious type of robustness appears when studying models that are defined by solutions to implicit equations or optimization problems which can have no, one or many solutions. For such a model to be robust, there must be a unique solution with a robust iterative method for finding solutions.

2. Stability: When learning dynamic models, it is not uncommon to encounter unstable models, even if the underlying system is stable. Such models have

extremely poor predictive performance which necessitates the development of specialized methods that guarantee the stability of the resulting model [220, 214, 148, 113]. Furthermore, many problems such as observer design and imitation learning require the learned model to be stable [110]. There is also a large amount of empirical evidence from the system identification community that stability constraints can drastically improve the generalization performance [220, 218, 214].

3. Domain Knowledge: Many learning problems have known underlying properties derived from theory or common sense. Incorporating known model properties as constraints is sometimes referred to as side information [3] and can be an extremely effective regularizer as it can reduce the estimator's variance with a minimal cost to the bias.

   For instance, *isotonic regression* enforces a known ordering relationship between the input and output variables; a property called *monotonicity*. This is useful in certain machine learning applications for security or fairness reasons [125]. Monotonicity also has many other applications, e.g., ensuring well-posedness [238, 214], guaranteeing convexity of an input-output mapping [35, 6, 235] or invertibility [96].

The methods required to enforce these types of properties depend on the model class being optimized over. In the next section, we will introduce some common model classes for learning nonlinear mappings. In Section 2.3, we will then expand on points 1-3 above and provide a survey of the current state of the art for verifying or enforcing those properties for different model structures.

## 2.2 Model Structures

This section introduces different model parameterizations for learning nonlinear mappings. In particular, we introduce basis function expansions, neural networks, and

equilibrium networks. We then discuss methods for extending these models to learning dynamic models.

### 2.2.1 Static Models

#### Basis Function Expansions

Basis function expansions are constructed by taking linear combinations of particular basis functions. Consider the problem of learning a function $y = f_\theta(u)$ where in $u \in \mathbb{R}^m$ and $y \in \mathbb{R}^p$. Then, a basis function expansion can be written as

$$y = \theta\Phi(u) \tag{2.3}$$

where $\theta \in \mathbb{R}^{p\times q}$ is a set of parameters and $\Phi(u) : \mathbb{R}^m \mapsto \mathbb{R}^q$ defines the basis. This includes linear models as well as polynomials which can be shown to be universal approximators of continuous functions via the Stone-Weierstrass theorem. Non-parametric models are widely used in statistical learning theory and construct $\Phi(u)$ from kernel functions evaluated at the observed data and find universal approximation properties via the representer theorem [23].

A considerable advantage of using basis function expansions is that model predictions are linear functions of the parameters allowing models to be fit via least squares or convex optimization. Additionally, for some basis function expansions, there exist powerful tools for fitting models with additional constraints. See, for instance, sum-of-squares programming discussed in Section 2.4.3.

#### Neural Networks

A model class that has recently gained a lot of popularity is the *neural network*. The scalability and accuracy of neural networks on practical tasks such as image classification has led to an enormous volume of research on different model structures and their properties [118].

A single hidden layer neural network is a function of the form:

$$y = W_o \phi(Wu + b), \tag{2.4}$$

where $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$ are the input and output, $W \in \mathbb{R}^{q \times m}, b \in \mathbb{R}^q$ and $W_o \in \mathbb{R}^{p \times q}$ are learnable parameters and $\phi(\cdot)$ is an element-wise non-linearity called an activation function. The term *neurons* is sometimes used to refer to the individual elements of $\phi(\cdot)$. Under mild assumptions, (2.4) is a universal function approximator over bounded domains as $q \to \infty$ [165].

While single hidden layer networks can approximate any function with arbitrarily high accuracy in theory, for problems such as image processing, it has been found that using multiple sequentially stacked layers can lead to significant improvements in model performance. The intuition is that using multiple sequential layers allows a model to learn hierarchies of features [169, 247]. A multi-layer neural network can be written as:

$$x^0 = u \tag{2.5}$$

$$x^{\ell+1} = \phi(W^\ell x^\ell + b^\ell), \quad \ell = 0, ..., L-1 \tag{2.6}$$

$$y = W^L x^L + b^L. \tag{2.7}$$

where the weights $W_\ell$ and biases $b_\ell$ are the learned parameters. Many additional architectures and mechanisms have been proposed, including the use of convolutional weights to learn spatially invariant features [175], deep residual networks to allow the learning of deeper networks [87] and normalization layers for preventing internal covariate shift [100, 33]. All of these methods are variations of the network structure (2.5) - (2.7), however, notably for this thesis, they can still all be written as compositions of simple element-wise nonlinearities and affine transforms.

### Equilibrium Networks

A recently introduced generalization of the neural network is the equilibrium network [14, 15, 238, 63]. This model structure is characterized by implicit equations that contain neural networks. Consider the dynamical systems

$$z_{t+1} = \phi(Wz_t + Bu + b) \tag{2.8}$$

and

$$\dot{z} = -z(t) + \phi(Wz(t) + Bu + b). \tag{2.9}$$

We can view (2.8) as either a discrete-time dynamical system or a weight-tied neural network. The model structure (2.9) is a neural ordinary differential equation [39]; a continuous time analogue of a neural network. Traditional deep learning approaches evaluate the models (2.8) and (2.9) by simulating the relevant equation over some fixed interval. Equilibrium networks instead consider the limiting behavior as $t \to \infty$, and solves directly for $z^\star$ such that

$$z^\star = \phi(Wz^\star + Bu + b). \tag{2.10}$$

By enforcing particular structures on the parameters $W$ and $B$, it is relatively easy to show that the set of equilibrium networks contains many popular network structures as a special case [62]. Consequently, (2.10) provides a good target for analysis as the results then generalize to large classes of neural networks.

It is important to note, however, that equilibrium networks have a number of complications including questions about the existence and uniqueness of solutions and computational issues with evaluating solutions to (2.10) and computing their gradients. We further discuss some of these issues in Chapter 4.

### 2.2.2 Dynamic Model Structures

The previous section studied common model structures used for learning nonlinear static mappings. In this section, we study how these ideas can be extended to learning nonlinear dynamical mappings.

When learning dynamical systems or sequential models, the prediction of a model depends not only on the current input but also the history of previous inputs and possibly an initial state. The model structures presented in Section 2.2.1 can all be extended to the dynamic case either by explicitly including a dependence on previous inputs and outputs, or by incorporating an internal state in the model. The section presents a rough outline of the different approaches to extending the models in Section 2.2.1 to learning dynamic models.

#### Finite Impulse Response Models

Nonlinear finite impulse response (FIR) models construct a prediction of the output from a finite history of inputs:

$$y_t = f_\theta(u_{t-\tau:t}), \tag{2.11}$$

where $\tau$ is a parameter that controls the memory of the model. Here, we have used the notation $u_{t-\tau:t}$ to refer to the the signal $u$ on the interval $[t-\tau : t]$ so $u_{t-\tau:t} = (u_{t-\tau}, ..., u_{t-1}, u_t)$.

In the case where $f_\theta(u_{t-t\tau:t}) = \sum_{k=0}^{\tau} \alpha_k u_{t-k}$, this model structure recovers the Linear Finite Impulse Model where $\alpha_0, ..., \alpha_\tau$ is the impulse response of a single input single output system. When learning nonlinear mappings, common choices for $f_\theta$ include Wiener and Volterra series models, and neural networks and temporal convolutional networks [13].

FIR models are fit via a simple regression problem which is convex in the case of a linearly parameterized model and a convex quality of fit criterion. FIR models are also inherently stable as they do not contain feedback. Unfortunately, these models

are inefficient representations when the system being modelled is highly resonant or operating near the edge of stability. When learning the models of such systems, it may be necessary to include feedback in the model to obtain a more parsimonious representation.

### Autoregressive Models

Perhaps the simplest feedback models are autoregressive models. These models construct predictions from a truncated history of inputs and outputs:

$$y_t = f_\theta(y_{t-1:t-\tau}, u_{t-\tau:t}). \tag{2.12}$$

This model class includes autoregressive exogenous input (ARX), output error and autoregressive moving average (ARMAX) models, where the main difference between these models comes from an assumption about how noise enters the model.

The inclusion of feedback in these models introduces stability issues. In particular, when simulating these models with new inputs but unknown output measurements, it is common to feedback the predicted output value at the next time step. In this case, it is quite common to observe unstable model models.

### State Space Models

State space models introduce an internal state that captures the effects of previous inputs. A deterministic state space model takes the form

$$x_{t+1} = f_\theta(x_t, u_t), \tag{2.13}$$

$$y_t = g_\theta(x_t, u_t). \tag{2.14}$$

Here, $f_\theta : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is known as the dynamics and $g_\theta : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ is known as the output mapping. This model class is extremely general and can be shown to contain LTI state space models when $f_\theta$ and $g_\theta$ are linear, recurrent neural networks [64] when $f_\theta$ is parameterized by a neural network (2.5) and the LSTM [94].

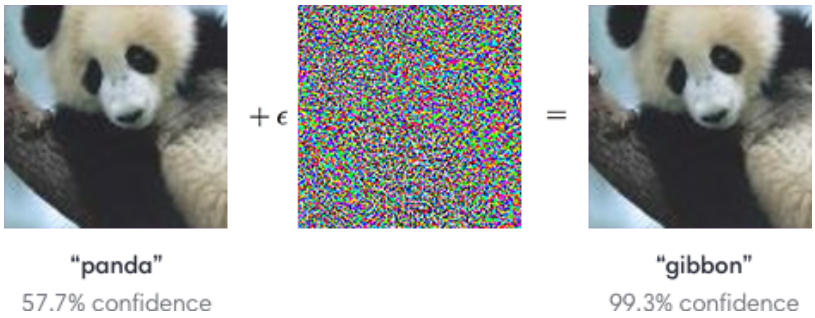



**Figure 2.2** – Example of an adversarial attack taken from [79]. While the perturbation is imperceptible to the human eye, the classifier changes its prediction to an incorrect label with high certainty.

State space models are widely studied in dynamics and control and even defining and checking the stability of the dynamics (2.13) can be very difficult. We will further discuss this problem in Section 2.3.3.

## 2.3 Behavioral Constraints

This section expands on the properties discussed at the end of Section 2.1 and provides an introduction to the current state of the art for enforcing those properties for different model structures.

### 2.3.1 Robustness

Deep learning has trended towards using larger and more expressive models with impressive results. While these models display a remarkable ability to generalize, it is important to note that model expressiveness can come at the cost of robustness. In particular, recent work has shown that imperceptible targeted adversarial attacks can consistently fool networks that otherwise perform well. This is shown, for instance, in Figure 2.2 where the learned classifier is extremely sensitive to an imperceptible input perturbation.

The existence of these adversarial examples is not specific to the problem of image classification. They have also been observed in reinforcement learning [181], language

modelling [41] and as we will see in chapter 5, system identification.

### Neural Network Robustness Verification and Lipschitz Continuity

The susceptibility of neural networks to adversarial attacks has motivated a large amount of research into network verification; methods for bounding a model's performance degradation when subjected to a bounded input perturbation.

Currently, adversarial training [79, 132] is the most effective way of improving network robustness to adversarial attacks [133, 159]. This approach adds adversarial examples to the training data set. While the method is simple and has proven empirically effective, it provides no formal guarantees.

To obtain a formal guarantee that a model is robust to an adversarial attack, a certification method must be used. Certification methods can be classed as either *exact* or *conservative*. Exact methods are based on satisfiability modulo theory [38, 61] or mixed integer linear programming [40, 211] and confirm whether or not there exists a perturbation that can fool a classifier. Conservative methods, on the other hand, only guarantee that a particular perturbation cannot fool a classifier and may fail even if there does not exist such a perturbation.

While exact methods induce no conservatism, they are computationally demanding and limited to relatively small-scale networks. For larger models, conservative methods must be used. These can be broadly grouped into *local* and *global* methods.

Local methods certify that a network is robust in a region centered on a specific input, usually the training data. A number of approaches have been proposed for local certification based on convex outer bounds [67], convex outer polytopes [239], tracking an outer reachable set that a perturbation can reach by stepping through network layers [149, 199, 236], or by calculating a local Lipschitz constant [88].

Global methods, on the other hand, certify adversarial robustness for all possible inputs by calculating the global Lipschitz constant. For a model $y = f(u)$, the global

Lipschitz constant is defined as the smallest constant $L$ such that

$$|f(u_1) - f(u_2)| \leq L|u_1 - u_2| \quad \forall u_1, u_2 \in \mathcal{U}. \tag{2.15}$$

Note that (2.15) is equivalent to the following quadratic inequality:

$$L^2|u_1 - u_2|^2 - |f(u_1) - f(u_2)|^2 \geq 0 \quad \forall u_1, u_2 \in \mathcal{U}. \tag{2.16}$$

Therefore, global Lipschitz constant certification is equivalent to checking the quadratic inequality (2.16) for all possible inputs. The Lipschitz constant of a function measures the worst-case sensitivity of the function, i.e., the maximum "amplification" of the difference in inputs to differences in the outputs. A Lipschitz bound can therefore be used to limit the maximum possible effect of an input perturbation.

While it has been argued that local methods are required to improve network expressivity [242, 98], recent work has shown that any locally Lipschitz bounded classifier admits an equivalent globally Lipschitz bounded network [120]. Furthermore, global Lipschitz bounds provide many further useful properties beyond adversarial robustness, e.g., robustness to distribution shift [50].

The key features of a good Lipschitz bounded learning approach include a tight estimation for Lipschitz constant and a computationally tractable training method with bounds enforced. For deep networks, [216] and [80] propose computationally efficient but conservative methods since their Lipschitz constant estimates are based on the composition of estimates for different layers, [8] proposed a combination of a novel activation function and weight constraints. For equilibrium networks, [62] proposed an estimation of Lipschitz bounds via input-to-state (ISS) stability analysis and [158] propose conditions based on monotone operator theory.

The fundamental problem in most Lipschitz bounded training approaches is that the calculation of the Lipschitz constant of a neural network is NP-hard [230]. The difficulty in estimating the Lipschitz constant means that most approaches resort to using highly conservative Lipschitz bounds. The tightest bounds known to date are given by [66] which estimates the Lipschitz constant for deep networks based on incremental

quadratic constraints and semidefinite programming (SDP). This was, however, limited to the analysis of previously trained networks. The SDP test was incorporated into training via the alternating direction method of multipliers (ADMM) in [164], however due to the complexity of the SDP, the training times recorded were almost 50 times longer than for unconstrained networks.

### 2.3.2 Monotonicity

Incorporating monotonicity into the learning process has a long history in the statistics community under the name of isotonic regression [155, 27]. A model $y = f(u)$ is termed monotone if

$$(f(u_1) - f(u_1))^\top (u_1 - u_2) \geq m|u_1 - u_2|^2 \quad \forall u_1, u_2 \in \mathcal{U}, \tag{2.17}$$

for some constant $m \geq 0$. Note that much like the Lipschitz condition (2.15), we can write (2.17) as a quadratic inequality constraint:

$$(f(u_1) - f(u_2))^\top (u_1 - u_2) - m|u_1 - u_2|^2 \geq 0 \quad \forall u_1, u_2 \in \mathcal{U}. \tag{2.18}$$

Incorporating monotonicity into a learning procedure has a number of benefits:

1. In many applications monotonicity can be used to encode domain knowledge. For instance, in pharmacology, it is known that toxicity increases monotonically with dosage [206]. Incorporating such constraints can improve model generalization and ensure that a model continues to produce realistic predictions when deployed.

2. Ensuring monotonicity can play a key role in developing secure, explainable [154] or fair models [44].

3. Monotonicity models can be used to construct invertible transformations, see for instance [214, Theorem 1]. This property has been used for instance to construct

autoregressive flows for modelling continuous probability distributions, e.g., [96, 54].

Despite the many uses for monotonicity in machine learning, training and verifying the monotonicity of deep neural networks remains challenging. The current approaches can be placed into two categories. Heuristic methods encourage monotonicity via penalization and scale, however, they provide no formal guarantee that the resulting model is monotone [82]. Other methods build monotonicity constraints into the network structure [52, 125, 235]. The second class of methods, however, can suffer from conservatism and scalability issues.

### 2.3.3 Stability of Discrete Time Systems

In this section, we present some definitions and results from the theory of discrete-time dynamical systems that are used throughout this thesis. Our main motivation is to characterize the stability of the discrete-time state space models (2.13), (2.14). Consider a discrete-time dynamical system

$$x_{t+1} = f(x_t, u_t), \tag{2.19}$$

$$y_t = g(x_t, u_t), \tag{2.20}$$

where $x_t \in \mathbb{R}^n$ is the model state, $u_t \in \mathbb{R}^m$ is a known input and $y_t \in \mathbb{R}^p$ is the output. Given the initial condition $x_0 = a$, the dynamical system (2.19), (2.20) defines a sequence-to-sequence mapping $\mathfrak{R}_a : \ell_{2e}^m \mapsto \ell_{2e}^p$.

When the functions $f$ and $g$ in (2.19), (2.20) are linear, stability is a well-understood concept and most definitions that one might reasonably construct lead to equivalent conditions. For nonlinear systems, the situation is more complex and there exist many definitions of stability describing the different behaviors that a nonlinear system might exhibit. In this section, we will introduce the main types of stability used throughout this thesis.

### Types of Stability for Nonlinear systems

There are many definitions of stability for nonlinear systems including asymptotic, exponential, and Lyapunov stability [201], input-output stability, [252, 253, 57], input-to-state stability [204], incremental stability, contraction analysis and convergent dynamics [215, 129].

In this thesis, we use the following definition for the internal stability of the model dynamics:

**Definition 2.1** (Contraction)**.** *The system (2.19) is termed contracting with rate $\alpha$, where $0 < \alpha < 1$, if for any two initial conditions $x_0^a$, $x_0^b$, given the same input sequence $u_t$, and some $p \in [1, \infty]$, there exists a continuous function $b_p(x_0^a, x_0^b) > 0$ such that the corresponding trajectories $x_t^a, x_t^b$ satisfy $|x_t^a - x_t^b|_p < \alpha^t b_p(x_0^a, x_0^b)$.*

This definition implies that the dynamics (2.19) have an exponentially fading memory with rate $\alpha$. For input-output systems, we characterize the stability as follows:

**Definition 2.2** (Incremental $\ell_2$ stability)**.** *The system (2.19), (2.20) is termed incrementally $\ell_2$ stable if for any two initial conditions $a$ and $b$, given the same input sequence $u$, the corresponding output trajectories $y^a = \mathfrak{R}_a(u)$ and $y^b = \mathfrak{R}^b(u)$ satisfy $y^a - y^b \in \ell_2^p$.*

This definition implies that the initial conditions of the sequence-to-sequence map $\mathfrak{R}_a(\cdot)$ are forgotten regardless of the input signal, however, the outputs can still be sensitive to small changes in the input. In such cases, it is natural to measure the system's robustness in terms of the incremental $\ell_2$-gain.

**Definition 2.3** (Incremental $\ell_2$ gain)**.** *The system (2.19), (2.20) is said to have an* incremental $\ell_2$-gain *bound of $\gamma$ if for all pairs of solutions with initial conditions $a, b \in \mathbb{R}^n$ and input sequences $u^a, u^b \in \ell_{2e}^m$, the output sequences $y^a = \mathfrak{R}_a(u^a)$ and $y^b = \mathfrak{R}_b(u^b)$ satisfy*

$$\left\|y^a - y^b\right\|_T^2 \leq \gamma^2 \left\|u^a - u^b\right\|_T^2 + d(a, b), \quad \forall T \in \mathbb{N}, \tag{2.21}$$

*for some function* $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ *with* $d(a, a) = 0$.

Condition (2.21) implies incremental $\ell_2$ stability, since if $u^a = u^b$ then $\|y^a - y^b\|_T^2 \leq d(a, b)$ for all $T \in \mathbb{N}$. It also implies that with fixed initial conditions, the input-output operator defined by (2.19) and (2.20) is Lipschitz continuous with Lipschitz constant $\gamma$, i.e. for any $a \in \mathbb{R}^n$ and all $T \in \mathbb{N}$

$$\|\mathfrak{R}_a(u) - \mathfrak{R}_a(v)\|_T \leq \gamma \|u - v\|_T, \quad \forall u, v \in \ell_{2e}^m. \tag{2.22}$$

This suggests further benefits in regulating the incremental $\ell_2$ constant as per the arguments in Section 2.3.1.

### Incremental Storage Functions

A common framework can be used to prove properties such as those in definitions 2.1, 2.2 and 2.3. Suppose we have two solutions to the system (2.19),(2.20) denoted $u^a \in \ell_2^m, x^a \in \ell_2^n, y^a \in \ell_2^p$ and $u^b \in \ell_2^m, x^b \in \ell_2^n, y^b \in \ell_2^p$. Then certain properties of the system are proved by selecting a supply rate $\sigma(\cdot)$ and showing that there exists a storage function $V(x_t^a, x_t^b) > 0$ with $V(x, x) = 0$, such that the following incremental dissipation inequality holds:

$$V(x_{t+1}^a, x_{t+1}^b) - V(x_t^a, x_t^b) \leq \sigma(\cdot) \tag{2.23}$$

Summing (2.23) over a time interval $t = 0, ..., T - 1$ gives:

$$V(x_T^a, x_T^b) - V(x_0^a, x_0^b) \leq \sum_{t=0}^{T-1} \sigma(\cdot), \tag{2.24}$$

$$\implies -V(x_0^a, x_0^b) \leq \sum_{t=0}^{T-1} \sigma(\cdot). \tag{2.25}$$

So the term $\sum_{t=0}^{T-1} \sigma(\cdot)$ is lower bounded by a constant that depends only on the initial condition. If $x_0^a = x_0^b$, then $\sum_{t=0}^{T-1} \sigma(\cdot) \geq 0$.

Choosing different functions $\sigma(\cdot)$ allows different properties of the system (2.19), (2.20) to be proved. For example:

1. If $\sigma(\cdot) = -\epsilon|x_t^a - x_t^b|^2$, the dissipation inequality (2.23) implies that $\sum_{t=0}^{T-1} |x_t^a - x_t^b|^2 < \frac{1}{\epsilon}V(x_0^a, x_0^b)$. Taking the limit as $T \to \infty$, gives $\sum_{t=0}^{\infty} |x_t^a - x_t^b|^2 < \frac{1}{\epsilon}V(x_0^a, x_0^b)$ which implies that the system is asymptotically incrementally stable.

2. If $\sigma(\cdot) = \gamma^2|u_t^a - u_t^b|^2 - |y_t^a - y_t^b|^2$, the dissipation inequality (2.23) implies that $||y^a - y^b||_{T-1}^2 < \gamma^2||u^a - u^b||_{T-1}^2 + V(x_0^a, x_0^b)$ which implies that the system has an incremental $\ell_2$ gain of $\gamma$ (see Definition 2.3).

Other properties such as incremental passivity and exponential incremental stability can be proved using the same approach. To make this concrete, consider the following example:

**Example 2.1 — Stability of LTI systems**

An LTI system takes the form:

$$x_{t+1} = Ax_t. \tag{2.26}$$

The incremental dynamics are $\Delta x_{t+1} = A\Delta x_t$. For LTI systems, a necessary and sufficient condition for exponential stability with rate $\lambda \in (0, 1)$ is the existence of a storage function $V = x^\top Px$ such that the dissipation inequality (2.23) holds with supply rate $\sigma(x_t) = -\lambda x_t^\top Px_t$. The resulting dissipation inequality is equivalent to the linear matrix inequality:

$$A^\top PA - (1 + \lambda)P \preceq 0. \tag{2.27}$$

The linear matrix inequality (2.27) can be easily verified by solving a convex feasibility problem.

Example 2.1 highlights an essential point; the LMI (2.27) is convex in $P$ for a given $A$ and convex in $A$ for a given $P$. However, it is not convex in both $A$ and $P$, complicating the problem of simultaneously learning both the dynamics and stability certificate. The problem stems from the nonconvex function composition

$V(x_{t+1}^a, x_{t+1}^b) - V(x_t^a, x_t^b) = V(f(x_t^a, u_t^a), f(x_t^b, u_t^b))$ in the dissipation inequality (2.23) and arises for most dynamic systems and choices of storage function.

### Differential Storage Functions

Another framework that can be used to prove stability properties such as those in definitions 2.1, 2.2 and 2.3 originates from contraction analysis [129]. Contraction analysis studies the differential dynamics of the system described by (2.19) and (2.20) given by

$$\delta x_{t+1} = F(x_t, u_t)\delta x_t + B(x_t, u_t)\delta u_t, \tag{2.28}$$

$$\delta y_t = C(x_t, u_t)\delta x_t + D(x_t, u_t)\delta u_t, \tag{2.29}$$

where $F(x_t, u_t) = \frac{\partial f}{\partial x}$, $B(x_t, u_t) = \frac{\partial f}{\partial u}$, $C(x_t, u_t) = \frac{\partial g}{\partial x}$ and $D(x_t, u_t) = \frac{\partial g}{\partial u}$. The stability properties of the differential dynamics, (2.28), (2.29), can be studied by searching for a differential storage function $V(x_t, \delta x_t) > 0$ and $V(x_t, 0) = 0$, and a differential supply rate $\sigma(\cdot)$ such that

$$V(x_{t+1}, \delta x_{t+1}) - V(x_t, \delta x_t) \leq \sigma(\cdot). \tag{2.30}$$

The vector $\delta x$ describes the relative dynamics between two neighboring trajectories and can be viewed as a differential analogue of the dissipation inequality (2.23). An incremental storage function can be constructed from a differential storage function by integrating along particular paths, allowing for global system properties to be analyzed.

The main benefit of the differential framework is that for certain types of storage functions, the inequality (2.30) is convex whereas (2.23) might not be. In this thesis, we use a differential approach in Chapter 6.

## 2.4 Relaxations

In Section 2.3, we saw that many useful properties can be encoded via inequality constraints. For instance, a Lipschitz bound holds if (2.15) holds, monotonicity is implied by the inequality (2.18), and various forms of stability hold if either of the inequalities (2.23) or inequalities (2.30) hold.

The ability to check such inequalities, of course, depends a great deal on the specific model class under consideration, however, for all but the simplest of the model classes, the corresponding inequalities are intractable. The primary difficulty with certifying properties such as (2.15), (2.23) or (2.30) is non-convexity of the relevant inequalities in the signals, which prevents any conclusions from being drawn about global inequality satisfaction from local properties. A common approach in such situations is to construct a convex relaxation of the inequality or the model. We will demonstrate a number of such relaxations in this section.

### 2.4.1 Convex Relaxations of Dissipation Inequalities

As we saw in the example at the end of Section 2.3.3, optimization over the set of stable models is complicated by the nonconvexity of the dissipation inequality (2.23) in the model parameters and the certificate of stability. The approach developed through the work: [144, 214, 212, 219, 218, 220] is to construct a convex relaxation of (2.23).

The main idea is that while the dissipation inequality is nonconvex due to the function composition of $V$ and $f$, the following inequality is convex for fixed multipliers $\lambda$:

$$V(x^a_{t+1}, x^b_{t+1}) - V(x^a_t, x^b_t) + \lambda^a(\cdot)^\top (x^a_{t+1} - f(x^a_t, u^a_t)) + \lambda^b(\cdot)^\top (x^b_{t+1} - f(x^b_t, u^b_t)) \leq \sigma(\cdot). \tag{2.31}$$

Whenever $x_{t+1} = f(x_t, u_t)$, we recover (2.23) and the model must therefore be stable. This approach is known as the S-Procedure and has a long history in control [166]. While (2.31) provides a convex, sufficient condition for stability for fixed multipliers $\lambda$,

fixing the multipliers causes the condition to be quite conservative. A key development in [214, 212] was to employ an implicit, redundant model structure that reduces the impact of fixing the multipliers. We will further explore this approach in Example 2.2.

**Example 2.2 — Convex Parameterizations of Stable LTI models**

The problem of learning LTI models has received a great deal of attention with methods proposed based on constrained optimization and regularization [131, 226, 117, 147, 134]. Following the approach in Section 2.4.1, we construct a redundant implicit parameterization of LTI systems

$$E\Delta x_{t+1} = F\Delta x_t, \tag{2.32}$$

where $E$ is invertible. Choosing $V(x_t^a, x_t^b) = \Delta x_t^\top P \Delta x_t$ with supply rate $\sigma(\cdot) = -\epsilon|\Delta x_t|^2$ and fixing the multipliers $\lambda = -\Delta x_t$, condition (2.31) can be written as

$$\begin{aligned} &\Delta x_{t+1}^\top P \Delta x_{t+1} - \Delta x_t^\top P \Delta x_t - 2\Delta x_t^\top (E\Delta x_{t+1} - F\Delta x_t) < -\epsilon|\Delta x_t|^2 \\ &\iff \begin{bmatrix} E + E^\top - P - \epsilon I & F^\top \\ F & P \end{bmatrix} \succeq 0. \end{aligned} \tag{2.33}$$

The model set (2.32), (2.33) is a convex parameterization of all stable LTI systems [138].

Note that in example 2.2, an implicit parameterization is required to ensure that that parameterization contains all LTI systems. Roughly speaking, the implicit parameterization reduces the conservatism by allowing for optimization over $E$ in place of $\lambda$.

### 2.4.2 Integral Quadratic Constraints

The theory of Integral Quadratic Constraints (IQC) was developed in [145], however, the foundational ideas have a long history in systems analysis and control [167, 241, 252, 237, 57]. IQC theory studies the stability properties of a system

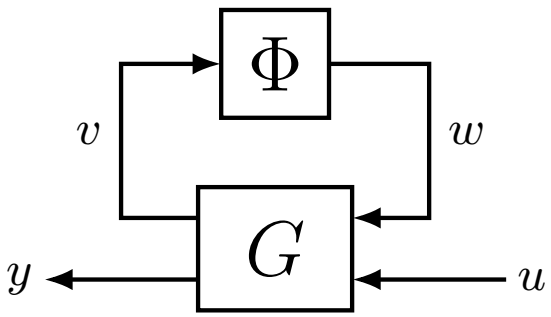



**Figure 2.3** – Lur'e Feedback Interconnection

interconnection called the Lur'e system, depicted in Figure 2.3. Here, we assume that we have a known, stable, nominal system $G$, in feedback with a nonlinear or uncertain component $\Phi$.

The principle difficulty behind analyzing the stability of the feedback interconnection depicted in Figure 2.3 is the nonlinearity/uncertainty $\Phi$. IQC theory replaces the mapping $w = \Phi(v)$ with a quadratic description of the signals that it could produce, and then guarantees the stability of the interconnection for that set of signals via the S-procedure [166]. Various classical theorems can be derived depending on the conditions placed on $\Phi$, e.g., the small gain theorem [108] when $\Phi$ has bounded gain, the circle criterion [253] when $\Phi$ is static and sector bounded and the passivity theorem [86] when $\Phi$ is passive.

Recent work has shown that IQCs have great potential for the analysis of neural networks. The main idea is to capture the behavior of the neural network's activation function in $\Phi$ and the neural network weights in $G$. For instance, using this approach, [66] has developed the tightest known bound on the Lipschitz constant for deep neural networks, [65] provides a method for bounding the effects of adversarial perturbations, and [246, 245, 163] provide methods for analyzing the stability of systems with neural network controllers.

In full generality, IQC theory permits a wide variety of different IQC descriptions including sector bounded, slope restricted, Popov, pure integrator, and Zames-Falb IQCs [145] and methods that obtain tighter descriptions by exploiting repeated nonlinearities [51]. In [115], however, it was shown that many traditional IQCs do not extend to the study of incremental stability as the positivity of an IQC does not imply its incremental positivity. To our knowledge, it remains to be seen if there exist any IQCs besides the slope restricted IQC that can be applied to incremental analysis. As the properties that we are interested in enforcing are all incremental properties, we restrict our discussion in this section to incremental Quadratic Constraints (iQCs).

**Definition 2.4** (incremental Quadratic Constraint [1])**.** *A function* $\phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ *said*

*to satisfy the iQC defined by* $\Pi \in \mathbb{S}^{2n}$ *if*

$$\begin{bmatrix} v_t^a - v_t^b \\ w_t^a - w_t^b \end{bmatrix}^\top \Pi \begin{bmatrix} v_t^a - v_t^b \\ w_t^a - w_t^b \end{bmatrix} \geq 0, \tag{2.34}$$

*for all* $w_t^a = \phi(v_t^a)$ *and* $w_t^b = \phi(v_t^b)$.

We can think of the iQC (2.34) as providing a rough description of the possible behaviors of the nonlinearity $\phi$. We will demonstrate how such a description is useful by way of example.

**Example 2.3 — iQC for LTI system**

Consider a dynamical system of the form

$$x_{t+1} = Ax_t + B\Phi(Cx_t) \tag{2.35}$$

where $A$, $B$ and $C$ are matrices of appropriate size and $\Phi$ is a nonlinearity satisfying the incremental quadratic constraint

$$\begin{bmatrix} v_t^a - v_t^b \\ w_t^a - w_t^b \end{bmatrix}^\top \Pi \begin{bmatrix} v_t^a - v_t^b \\ w_t^a - w_t^b \end{bmatrix} > 0 \tag{2.36}$$

for all $v_t^a, v_t^b \in \mathbb{R}^q, w_t^a = \Phi(v_t^a), w_t^b = \Phi(v_t^b)$. Recall that we can prove incremental stability by establishing the existence of an incremental storage function $V(x_t^a, x_t^b) > 0$, where $V(x, x) = 0$ that satisfies the incremental dissipation inequality (2.23) with $\sigma(\cdot) = -\epsilon|x_t^a - x_t^b|^2$. Verifying this inequality, however, is complicated by the dependence of $x_{t+1}$ on the uncertain component $\Phi$.

The approach taken in IQC analysis is to treat (2.35) as a feedback interconnection between the linear system

$$G : \begin{cases} x_{t+1} = Ax_t + Bw_t \\ v_t = Cx_t \end{cases} \tag{2.37}$$

and the uncertain component $w = \Phi(x)$. If we consider the extended space of signals $(x, w)$, then we can show that the dissipation inequality holds for

all signals satisfying $w = \Phi(v)$ by using the fact that the iQC holds for all such signals.

To make this concrete, take $\Delta x = x_t^a - x_t^b$, $\Delta w_t = w_t^a - w_t^b$ and $\Delta v_t = v_t^a - v_t^b$ and take $V(x_t^a, x_t^b) = \Delta x_t{}^\top P \Delta x_t$. Suppose that the following inequality holds:

$$V(x_{t+1}^a, x_{t+1}^b) - V(x_t^a, x_t^b) + \epsilon |\Delta x_t|^2 \leq - \begin{bmatrix} \Delta v_t \\ \Delta w_t \end{bmatrix}^\top \begin{bmatrix} \Pi_{11} & \Pi_{12} \\ \Pi_{12}^\top & \Pi_{22} \end{bmatrix} \begin{bmatrix} \Delta v_t \\ \Delta w_t \end{bmatrix}. \quad (2.38)$$

The left hand side is the dissipation inequality; if it is less than zero, then the system is stable. The right hand side is the iQC (2.36) which holds whenever $w_t = \Phi(v_t)$. As all terms are quadratic, they can be easily combined to give

$$\begin{bmatrix} \Delta x_t \\ \Delta w_t \end{bmatrix}^\top \begin{bmatrix} P - A^\top P A - \epsilon I - C^\top \Pi C & PB - C\Pi_{12} \\ B^\top P - \Pi_{12}^\top C^\top & B^\top P B - \Pi_{22} \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta w_t \end{bmatrix} \geq 0. \quad (2.39)$$

For fixed matrices $A, B$ and $C$, this is a convex constraint in $P$ and the IQC multipliers that can be solved by using semidefinite programming techniques.

### 2.4.3 Sum of Squares Programming

Sum-of-squares programming is a general approach for proving the global non-negativity of a multivariate polynomial first proposed in [194]. When a model is parameterized via polynomials, sum-of-squares programming provides an attractive relaxation for checking the behavioral properties discussed in Section 2.3. For example, for polynomial state space models and polynomial storage functions, sum-of-squares programming provides a convex relaxation of the dissipation inequality (2.23).

In the context of this thesis, we use sum-of-squares programming in Chapter 6 for the distributed learning of stable systems.

**Definition 2.5** (Monomial)**.** *A monomial in $x \in \mathbb{R}^n$ is a product*

$$m_d(x) = x_1^{d_1} \cdot x_2^{d_2} \cdot \ldots \cdot x_n^{d_n}$$

*where $d_i$ are nonnegative integers. The degree of the monomial is $D = d_1 + d_2 + ... + d_n$.*

The set of monomials with degree less than or equal to $D$ form of basis for degree $D$ polynomials.

**Definition 2.6.** *Let $m(x)$ be the vector of monomials with degree less than or equal to $D$. Then, a polynomial $p : \mathbb{R}^n \mapsto \mathbb{R}$ can be constructed by*

$$p(x) = \alpha^\top m(x) \tag{2.40}$$

*where $\alpha$ is a vector of coefficients.*

While proving the global non-negativity of a polynomial of degree four or higher is NP-hard [4], a simple sufficient condition is given by the existence of a sum-of-squares decomposition. That is, a polynomial $p(x) : \mathbb{R}^n \mapsto \mathbb{R}$ of degree $2D$ is nonnegative if it can be written as

$$p(x) = \sum_i p_i(x)^2. \tag{2.41}$$

Sum-of-squares programming provides a convenient and efficient method for searching for such a decomposition by formulating the search as a convex semidefinite program. A necessary and sufficient condition for a polynomial to have a SOS decomposition is that it can be written as

$$p(x) = m(x)^\top Q m(x) \tag{2.42}$$

where $Q$ is a positive definite matrix called the Gram matrix. The link between the Gram matrix and a SOS decomposition can be established via the Cholesky decomposition $Q = L^\top L$

$$p(x) = m(x)^\top Q m(x) \tag{2.43}$$

$$= (Lm(x))^\top (Lm(x)) \tag{2.44}$$

$$= \begin{bmatrix} p_1(x) \\ p_2(x) \\ \vdots \end{bmatrix}^\top \begin{bmatrix} p_1(x) \\ p_2(x) \\ \vdots \end{bmatrix} \tag{2.45}$$

$$= \sum_i p_i(x)^2 \tag{2.46}$$

where $p_i(x) = \sum_j L_{ij} m_j(x)$.

The link between the existence of sum-of-squares decomposition and a convex semidefinite program was made in [161, 162] and lead to widespread uptake in the control community [12, 136, 213]. There now exist a number of high-quality toolboxes for formulating and solving sum-of-squares programming problems [128, 119, 160]

Finally, we note that while the existence of a sum-of-squares decomposition is generally just a sufficient condition for positivity, there are a number of situations where it is both necessary and sufficient. These are when $p(x)$ is: uni-variate, quadratic or a quartic polynomial in two variables.

## 2.5 Numerical Methods

We saw in section 2.1 that model training can be formulated as an optimization problem. Once such an optimization problem has been formulated, it is usually necessary to employ a numerical method to iteratively search for a solution. The difficulty of this search, however, depends on the structure of the formulated optimization problem. A central theme of this thesis is the construction of model classes that have certain properties that simplify the optimization procedure.

In this section, we will provide an introduction to the necessary background information on numerical optimization, however we refer to [156] and [26] for a thorough treatment.

Suppose that we have a smooth function $J : \mathbb{R}^N \mapsto \mathbb{R}$, and a set of allowable parameter

values $\Theta \subseteq \mathbb{R}^N$. An optimization problem can then be formulated as

$$\min_{\theta \in \Theta} \quad J(\theta). \tag{2.47}$$

The function $J(\theta)$ is called the *objective function* and $\Theta$ is called the *feasible set*. It is also common to write an optimization problem with a set of constraints, e.g.,

$$\min_{\theta} \qquad J(\theta), \tag{2.48a}$$

$$\text{subject to} \quad g(\theta) \geq 0. \tag{2.48b}$$

Note however that (2.48a), (2.48b) can always be written as (2.47) by choosing $\Theta = \{\theta \subset \mathbb{R}^N \ : \ g(\theta) \geq 0\}$.

In the context of machine learning, the objective function $J$ usually contains a combination of model performance measures such as the loss function, regularizers, and possibly some penalty terms that encourage 'good' behavior. The constraint $\theta \in \Theta$, on the other hand, usually contains hard requirements for a model; these must be satisfied for the model to be useful. For example, this might include the requirement that the model is stable or well-posed.

In this section, we will first consider methods for unconstrained optimization. Methods for unconstrained optimization are usually simpler as there are no 'hard' model requirements that must be satisfied. We will then introduce some methods for constrained optimization.

### 2.5.1 Unconstrained Optimization Methods

We first consider the problem of *unconstrained optimization*. An unconstrained optimization problem is a problem of the form (2.47) where $\Theta = \mathbb{R}^N$, i.e.,

$$\min_{\theta \in \mathbb{R}^N} \quad J(\theta). \tag{2.49}$$

As the objective functions in this work are nonconvex, we will only search for a local solution to (2.49), that is a point $\theta^*$ that locally minimizes $J$. We can solve (2.49) via an iterative algorithm, i.e., given a solution estimate $\theta^{(k)}$ at iteration $k$, the iterative algorithm updates our estimates according to an update rule

$$\theta^{(k+1)} = \theta^{(k)} + \alpha^{(k)} d^{(k)}, \tag{2.50}$$

where $\alpha^{(k)} \in \mathbb{R}$ is called the step size and $d^{(k)} \in \mathbb{R}^N$ is called the search direction. A good choice for $\theta^{(k+1)}$ is one that ensures that $J(\theta^{(k+1)}) < J(\theta^{(k)})$, however, this leaves considerable flexibility in how to choose $\alpha^{(k)}$ and $d^{(k)}$ and many methods have been proposed. Broadly speaking, there are two classes of methods for choosing $\alpha^{(k)}$ and $d^{(k)}$:

1. *First order methods* choose an update by approximating the objective function by a first order Taylor series

$$J(\theta) \approx J(\theta^{(k)}) + \frac{\partial J(\theta^{(k)})}{\partial \theta}(\theta - \theta^{(k)})$$

   The most common first order method is gradient descent or where $\theta^{(k+1)} = \theta^{(k)} - \alpha \frac{\partial J(\theta^{(k)})}{\partial \theta}$ where $\alpha$ is chosen sufficiently small to ensure that the Taylor expansion is accurate. Many variants of gradient descent have been proposed to speed up the convergence and reduce sensitivity to hyper-parameter choice and scaling [111, 124].

2. *Second order methods* choose an update by approximating the objective function with a second order Taylor series

$$J(\theta) \approx J(\theta^{(k)}) + \frac{\partial J(\theta^{(k)})}{\partial \theta}(\theta - \theta^{(k)}) + (\theta - \theta^{(k)})^\top \frac{\partial^2 J(\theta^{(k)})}{\partial \theta^2}(\theta - \theta^{(k)}).$$

   Newtons method solves for the value of $\theta$ that optimizes the above quadratic approximation resulting in an iterative method $\theta^{(k+1)} = \theta^{(k)} - \left[\frac{\partial^2 J(\theta^{(k)})}{\partial \theta^2}\right]^{-1} \frac{\partial J(\theta^{(k)})}{\partial \theta}$.

Second order methods have much better local convergence properties than first-order

methods. For smooth objective functions, the neighborhood of a local optima is well approximated by a quadratic, allowing second-order methods to refine the solution to extremely high accuracy in a few steps. Despite its superior convergence properties, Newton's method is often avoided for two reasons: Firstly, it requires a good initial guess of $\theta^{(0)}$ to make use of its superior convergence. In the absence of good initialization, the Levenberg Marquadt algorithm [151] or line search methods [156, Chapter 3] are required to ensure robust convergence. Secondly, the algorithm scales poorly with the number of parameters. In particular, storing the Hessian requires and $\mathcal{O}[N^2]$ memory and calculating $\left[\frac{\partial J(\theta^k)}{\partial \theta^2}\right]^{-1}$ requires $\mathcal{O}[N^3]$ operations.

First order methods, on the other hand, have relatively slow convergence rates, but the individual steps are very cheap. These methods have become extremely popular in the machine learning community where there has been a trend towards training giant models on large amounts of data. In these situations, second-order methods are unable to sufficiently scale. Furthermore, the use of redundant, overparameterized models used in modern deep learning seems to significantly simplify training, and it has been observed that stochastic gradient descent is able to achieve near zero training error [254], in a reasonable number of iterations.

We frequently use a first-order method, known as Adaptive Momentum Estimation (Adam). The algorithm is shown in Algorithm 2.1. All operations are elementwise, e.g. $\sqrt{\hat{v}}$ is the elementwise square root of $\hat{v}$. The operations $\beta_1^k$ and $\beta_2^k$ refer to the $k$th power of $\beta_1$ and $\beta_2$, respectively. Common hyper-parameter choices are $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and $\alpha = 1E - 3$.

### 2.5.2 Constrained Optimization

In this section, we introduce some methods for constrained optimization. In general, a constrained optimization problem can be written as follows

$$\min_{\theta \in \Theta} \quad J(\theta). \tag{2.51}$$

**Algorithm 2.1:** Adaptive Momentum Estimation [111]

Initialize step size $\alpha \in R_+$
Initialize hyperparameters $\beta_1, \beta_2 \in [0, 1)$
Initialize initial guess $\theta^{(1)} \in \Theta$;
$m^{(1)} \leftarrow 0$
$v^{(1)} \leftarrow 0$
**for** $k = 1, 2, ...$ **do**
  $g^{(k)} \leftarrow \nabla J(\theta^{(k)})$
  $m^{(k+1)} \leftarrow \beta_1 m^{(k)} + (1 - \beta_1) g^{(k)}$
  $v^{(k+1)} \leftarrow \beta_2 v^{(k)} + (1 - \beta_2) {g^{(k)}}^2$
  $\hat{m} \leftarrow m^{(k+1)} \ / \ (1 - \beta_1^k)$
  $\hat{v} \leftarrow v^{(k+1)} \ / \ (1 - \beta_2^k)$
  $\theta^{(k+1)} = \theta^{(k)} - \alpha \cdot \hat{m} \ / \ (\sqrt{\hat{v}} + \epsilon)$
  **if** *Converged* **then**
    Return result $\theta^{(k+1)}$;

where the constraints are specified by a set of permitted parameter values $\Theta$. Typically, $\Theta$ will be described by the union of a finite set of equality and/or inequality constraints, e.g., $\Theta = \left\{\theta \in \mathbb{R}^N \; ; \; g_i(\theta) \geq 0, i = 1, ..., n_{ineq}, \; h_i(\theta) = 0, i = 1, ..., n_{eq}\right\}$. A constrained optimization problem can always be converted to an unconstrained optimization problem by introducing an indicator function

$$\mathcal{I}_\Theta(\theta) = \begin{cases} 0, & \theta \in \theta, \\ \infty, & \theta \notin \Theta. \end{cases}$$

The constrained optimization problem (2.51) is equivalent to the following unconstrained problem:

$$\min_{\theta \in \mathbb{R}^N} \quad \hat{J}(\theta), \quad \hat{J}(\theta) := J(\theta) + \mathcal{I}_\Theta(\theta) \tag{2.52}$$

Note, however, that the indicator function is not smooth or differentiable, so gradient descent or Newton's method cannot be applied to solve the optimization problem. Methods such as *interior point methods* or *penalty methods* solve (2.51) by constructing a series of unconstrained problems $\min_{\theta \in \mathbb{R}^N} \hat{J}_k(\theta)$ where $\lim_{k \to \infty} \hat{J}_k(\theta) = \hat{J}(\theta)$.

### Interior Point Methods

Interior point methods are a general purpose approach to solving nonlinear optimization problems and form the basis for a number of high-quality, widely used solvers, e.g. IPOPT [231] and Mosek [9].

An interior point method constructs a smooth approximation of (2.52) by introducing a *barrier function* $\phi : \mathbb{R}^N \mapsto \mathbb{R}$ that: is smooth, finite on the interior of $\Theta$, tends towards infinity as the $\theta$ approaches the boundary of the feasible set and infinite when $\theta \notin \Theta$. We now define an approximation of (2.52) as

$$J_\mu(\theta) = J(\theta) + \mu\phi(\theta). \tag{2.53}$$

Here, $\mu$ is a barrier parameter which controls the tradeoff between the accuracy of

the approximation and smoothness. I.e., when $\mu$ is small, $\phi(\theta)$ only has an effect near the edge of $\Theta$, however, $J_\mu(\cdot)$ will have very high curvature which may slow down the convergence of Newton's method or gradient descent. Conversely, when $\mu$ is large, the barrier function will push the minimizer of $J_\mu$ away from the edge of $\Theta$, however, the $J_\mu$ will also have better conditioning. To obtain both high accuracy and good conditioning, an interior point method solves a sequence of optimization problems with decreasing $\mu$.

**Algorithm 2.2:** Path Following Interior Point Method

```
Initialize μ_1 > 0, θ_1 ∈ Θ;
for k = 1, 2, ... do
    Solve θ^(k+1) = arg min_θ J_μ(θ) using a local search method initialized at θ^(k);
    if Converged then
        Return result θ^(k+1);
    else
        Reduce barrier parameter μ^(k+1) < μ^(k)
```

A path following interior point method effectively warm starts the local search method at each iteration with the solution from the previous iteration, and in doing so ensures rapid convergence of the local search method. The sequence of solutions $\theta^{(1)}, \theta^{(2)}, \ldots$ is called the central path and converges to the true minimizer of (2.51) as $\mu \to 0$.

To write an effective interior point method, it is of course necessary to choose a suitable barrier function. This task is made somewhat simpler when $\Theta$ is convex. For convex constraints, [153, Section 2.5] shows that there always exists a barrier function with a suitable smoothness property called *self concordance*. While there are no guarantees that a barrier function or its derivative will be easy to evaluate, the model sets constructed in this thesis all have tractable barrier functions. We will discuss some examples in the following sections.

**Linear Constraints**

A linear constraint on $\theta$ is described by $\{\theta \mid a_i^\top \theta \leq b_i\}$ where $a_i \in \mathbb{R}^N$ and $b_i \in \mathbb{R}$. The barrier function for a single linear constraint is then given by

$$\phi(\theta) = \begin{cases} -\log(b_i - a_i^\top \theta), & b_i - a_i^\top \theta > 0, \\ \infty, & b_i - a_i^\top \theta \leq 0. \end{cases}$$

**Semidefinite Programming**

A semidefinite constraint requires that a certain matrix is positive semidefinite. These are extremely common when studying the stability of linear dynamical systems and we have already seen a number of these in previous sections, e.g., equations (2.27), (2.33) and (2.39). A simple barrier function for imposing the semidefinite constraint $P \succ 0$ is the $\log\det$ barrier function:

$$\phi(P) = \begin{cases} -\log\det(P), & P \succ 0, \\ \infty, & P \preceq 0. \end{cases} \tag{2.54}$$

The gradient of the $\log\det$ barrier function is given by

$$\frac{\partial \phi(P)}{\partial P} = -\frac{1}{\det P}\frac{\partial \det P}{\partial P} = -P^{-\top}. \tag{2.55}$$

### 2.5.3 Operator Splitting and ADMM

When solving extremely large optimization problems, it is necessary to exploit some structure in the problem. A particular family of methods that has recently received a lot of attention are operator splitting methods, which search for a zero in the sum of multiple operators. The term splitting is derived from the fact that operator splitting methods 'split' a problem into a collection of more easily solved subproblems. Such methods find application in convex optimization as finding the optima of a convex

objective function is equivalent to finding a zero in the monotone subdifferential of that objective function.

A significant benefit of the operator splitting approach is the well-developed theory that it builds on, known as *monotone operator theory*. Monotone operator theory allows for the derivation of many common algorithms used for large-scale optimization and provides a rigorous analysis of the convergence properties of those algorithms.

The relevance of operator splitting methods to this thesis are two-fold: Firstly, the models developed in Chapters 4 and 5 are described by fixed point equations that can be related to operator splitting methods on non-Euclidean spaces. Secondly, in Section 6, we apply a splitting method called the Alternating Directions Method of Multipliers (ADMM) to scalably fit large models.

The structure of this section is as follows: Firstly, we provide a quick introduction to monotone operator theory. For further details, we recommend [183]. We then introduce a number of splitting methods that can be used to solve monotone operator inclusion problems. Finally, we introduce ADMM as a specific case of an operator splitting method applied to convex optimization.

### Monotone Operator Theory

An operator $A$ has a Lipschitz bound of $L$ if for any $(x, u), (y, v) \in A$

$$\|u - v\|_Q \leq L\|x - y\|_Q. \tag{2.56}$$

An operator $A$ is non-expansive if $L = 1$ and contractive if $L < 1$. An operator $A$ is monotone if

$$\langle u - v, x - y\rangle_Q \geq 0, \quad \forall (x, u), (y, v) \in A. \tag{2.57}$$

It is strongly monotone with parameter $m$ if

$$\langle u - v, x - y\rangle_Q \geq m\|x - y\|_Q^2, \quad \forall (x, u), (y, v) \in A. \tag{2.58}$$

A monotone operator $A$ is maximal monotone if no other monotone operator strictly contains it, which is a property required for the convergence of most fixed point iterations. Specifically, an affine operator $A(x) = Wx + b$ is (maximal) monotone if and only if $QW + W^\top Q \succeq 0$ and strongly monotone if $QW + W^\top Q \succeq mI$. A subdifferential $\partial f$ is maximal monotone if and only if $f$ is a convex closed proper function.

The resolvent and Cayley operators for an operator $A$ are denoted $R_A$ and $C_A$ respectively, defined as

$$R_A = (I + \alpha A)^{-1}, \quad C_A = 2R_A - I \tag{2.59}$$

for any $\alpha > 0$. The resolvent and Cayley operators are nonexpansive for any maximal monotone $A$, and are contractive for strongly monotone $A$. Operator splitting methods consider finding a zero in a sum of operators (assumed here to be maximal monotone), i.e., find $z$ such that $0 \in (A + B)(z)$. An important property of the resolvent and Cayley operators is that $0 \in A(x) \iff x = R_A(x) = C_A(x)$.

### Operator Splitting Algorithms

Here we give some popular operator splitting methods for this problem as follows.

- Forward-backward splitting: $z^{k+1} = R_B(z^k - \alpha A(z^k))$, i.e.,

$$\begin{aligned} z^{k+1/2} &= R_B(z^k - \alpha A z^k) \\ z^{k+1} &= z^{k+1/2} - \alpha(A z^{k+1/2} - A z^k) \end{aligned} \tag{2.60}$$

- Peaceman-Rachford splitting: $z^{k+1} = C_A C_B(z^k)$, $x^k = R_B(z^k)$, i.e.,

$$\begin{aligned} x^{k+1/2} &= R_B(z^k), \\ z^{k+1/2} &= 2x^{k+1/2} - z^k, \\ x^{k+1} &= R_A(z^{k+1/2}), \\ z^{k+1} &= 2x^{k+1} - z^{k+1/2}. \end{aligned} \tag{2.61}$$

- Douglas-Rachford splitting: $z^{k+1} = 1/2(I + C_A C_B)(u^z)$, $x^k = R_B(z^k)$, i.e.,

$$\begin{aligned} x^{k+1/2} &= R_B(z^k), \\ z^{k+1/2} &= 2x^{k+1/2} - z^k, \\ x^{k+1} &= R_A(z^{k+1/2}), \\ z^{k+1} &= z^k + x^{k+1} - x^{k+1/2}. \end{aligned} \tag{2.62}$$

A sufficient condition for forward-backward splitting to converge is $\alpha < 2m/L^2$. The Peaceman-Rachford and Douglas-Rachford methods converge for any $\alpha > 0$, although the convergence speed will often vary substantially based upon $\alpha$.

### ADMM

Monotone operator splitting methods are closely related to convex optimization as the problem of minimizing a convex function is equivalent to finding a zero in its subgradient, which is monotone.

Consider the following optimization problem:

$$\min_{x,y} \quad f(x) + g(y) \tag{2.63}$$

$$s.t. \quad Ax + By = c \tag{2.64}$$

Applying the Douglas Rachford splitting (2.62) to the problem of finding a zero in the subdifferential of the dual problem to (2.63),(2.64) leads to the following algorithm [183]:

$$x(k+1) = \arg\min_x f(x) + \frac{\rho}{2}||x - y(k) + v(k)||^2, \tag{2.65}$$

$$y(k+1) = \arg\min_y g(y) + \frac{\rho}{2}||x(k+1) - y - v(k)||^2, \tag{2.66}$$

$$v(k+1) = v(k) - x(k+1) + y(k+1). \tag{2.67}$$

This is an extremely well-known algorithm called the Alternating Directions Method

of Multipliers (ADMM). See [31] for a detailed review of its variations and applications. Note that the update steps steps in (2.65) and (2.66) decouple the optimization of the two variables $x$ and $y$.

# Chapter 3

# Robust Recurrent Neural Networks

In this chapter, we propose a new convex parameterization of RNNs satisfying stability and robustness conditions. The approach taken is to treat an RNN as a linear system in feedback with monotone, slope-restricted nonlinearities, and to apply methods from robust control to develop stability conditions that are less conservative than prior methods.

The specific method that we employ is to describe the nonlinearities by incremental quadratic constraints. This approach was similarly used in [66] for the analysis of Lipschitz constants of feedforward neural networks. In this chapter, we apply it to the problem of training RNNs. To this end, we present two main developments: Firstly, the extension to sequence-to-sequence maps requires the introduction of a suitable storage function. Secondly, the use of an implicit model allows for a parameterization that is jointly convex in the model parameters, stability certificate, and the multipliers associated with incremental quadratic constraints. This technique of combining an implicit model structure with a quadratic description of the nonlinearity is also used in chapters 4 and 5.

The proposed model set contains all previously published sets of stable RNNs and all stable linear time-invariant (LTI) systems. In our construction, there is a certain

term that is set to zero to avoid an algebraic feedback loop. It turns out that this term significantly improves the expressibility of the model class, however, the details of this term are left until Chapter 4.

In this chapter, we also present a simplified version of the Robust RNN called the Contracting Implicit RNN (ci-RNN) that we first presented in [176]. This can be viewed as a special case of the Robust RNN that occurs when you restrict certain parameters.

### Publications

Some of the content of this chapter has previously appeared in the following publications:

Max Revay and Ian R. Manchester. Contracting implicit recurrent neural networks: Stable models with improved trainability. *L4DC*, 2020.

Max Revay, Ruigang Wang, and Ian R. Manchester. A convex parameterization of robust recurrent neural networks. *Control Systems Letters*, 2021.

## 3.1 Introduction

Recurrent neural networks (RNNs) are nonlinear state-space models incorporating neural networks, and are widely used to model dynamical systems and sequence-to-sequence mappings in system identification and machine learning. It has long been observed that RNNs can be difficult to train in part due to model instability, referred to as the exploding gradients problem [20], and recent work shows that RNNs can be highly sensitive to input perturbations [41], posing challenges for reliable learning from data. Both stability and sensitivity of nonlinear dynamical systems are long-standing concerns in control theory, see e.g. [252] and many others.

For nonlinear dynamical systems, there are many distinct notions of stability appropriate for different purposes.

The most common notion is the stability of a particular solution, e.g., an equilibrium at the origin, and this can be verified by finding a Lyapunov function. However, this notion is not suitable when learning dynamical systems with inputs, since the stability can be input-dependent and the purpose of the model is to predict outputs with previously unseen inputs. In contrast, incremental stability [57] and contraction [129] are more appropriate since they imply the stability of solutions for *all* possible inputs.

Even if a model is stable, it is usually problematic if its output is very sensitive to small changes in the input. This sensitivity can be quantified by the model's incremental $\ell_2$ gain. Finite incremental $\ell_2$ gain implies both boundedness and continuity of the input-output map [57], and also bounds the Lipschitz constant of the sequence-to-sequence mapping. In machine learning, the Lipschitz constant is used in proofs of generalization bounds [18], analysis of expressiveness [257] and guarantees of robustness to adversarial attacks [97, 168].

The problems of training models with stability or robustness guarantees have seen significant attention for both linear [117, 147, 220] and nonlinear [214, 222, 112] models. For analysis of RNN models, several convex stability conditions have been given in terms of linear matrix inequalities (LMIs), e.g. [103, 17, 42], while incremental quadratic constraints [252] have been applied to (non-recurrent) neural networks to develop the tightest bounds on the Lipschitz constant known to date [66]. However, these tests are convex for a fixed model and are *not* jointly convex in the model parameters and stability certificates [220], making it difficult to apply them during training.

A simple but conservative approach is to fix the stability certificate and only search for the model parameters [148]. However, it has recently been shown that implicit parameterizations allow for joint convexity of the model and a stability certificate for linear [220], polynomial [214] and RNN [176] models. It has also been observed that

stability constraints serve as an effective regularizer and can improve generalization performance [219, 176].

It should be noted that when learning neural network models, the cost function being minimized is generally a nonconvex function of parameters. Nevertheless, a tractable convex representation of stable models is still useful since stability is often a "hard" constraint that must be satisfied by any trained model. With a convex model set, this can be added to standard training procedures via projection or barrier terms, without adding the significant extra challenge of finding a feasible solution to the nonconvex constraints.

## 3.2 Problem Formulation

The main goal of this chapter is to construct a rich parameterization of the state space models:

$$x_{t+1} = f_\theta(x_t, u_t), \tag{3.1}$$

$$y_t = g_\theta(x_t, u_t), \tag{3.2}$$

where the dynamical system defined by (3.1) and (3.2) have robustness guarantees. We use the following terminology:

1. A model is called *robust* if the system (3.1), (3.2) has finite incremental $\ell_2$ gain (see definition 2.2).

2. A model is called $\gamma$-*robust* if the system (3.1), (3.2) has an incremental $\ell_2$ gain bound of $\gamma$ (see definition 2.3).

We also refer to the model sets $\Theta_*$ and $\Theta_\gamma$ as robust and $\gamma$-robust if they contain only robust and $\gamma$-robust models.

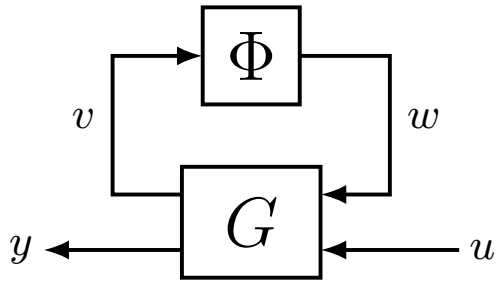



**Figure 3.1** – Feedback interconnection for RNNs.

## 3.3 Robust RNNs

### 3.3.1 Model Structure

We parameterize (3.1) and (3.2) as a feedback interconnection between a linear system $G$ and a static, memoryless nonlinearity $\Phi$:

$$G \begin{cases} x_{t+1} = \bar{F}x_t + \bar{B}_1 w_t + \bar{B}_2 u_t + \bar{b}_x \\ y_t = C_1 x_t + D_{11} w_t + D_{12} u_t + \bar{b}_y \\ v_t = \bar{C}_2 x_t + \bar{D}_{22} u_t + \bar{b}_v \end{cases}, \tag{3.3}$$

$$w = \Phi(v), \tag{3.4}$$

where $\Phi(v) = [\phi(v_1) \ \cdots \ \phi(v_q)]^\top$ with $v_i$ as the $i$th component of the $v \in \ell_{2e}^q$. This feedback interconnection is shown in Fig. 3.1. We assume that the slope of $\phi$ is restricted to the interval $[0, \beta]$, with $\beta$ known and fixed:

$$0 \leq \frac{\phi(y) - \phi(x)}{y - x} \leq \beta, \quad \forall x, y \in \mathbb{R}, \ x \neq y. \tag{3.5}$$

In the neural network literature, such functions are referred to as "activation functions", and common choices (e.g. tanh, ReLU, sigmoid) are slope restricted [78].

**Remark 3.1.** Note that we have excluded the term $\tilde{D}_{21}$ from (3.3) due to the well-posedness issues implied by the algebraic feedback loop $v = \tilde{D}_{21}\phi(v)$. This issue will be addressed in subsequent chapters.

The proposed model structure is highly expressive and contains many commonly used model structures. For instance, LTI systems are obtained when $\bar{B}_1 = 0$ and $D_{11} = 0$, whereas RNNs of the form [64]:

$$x_{t+1} = \mathcal{B}_1 \Phi(\mathcal{A} x_t + \mathcal{B} u_t + b), \tag{3.6}$$

$$y_t = \mathcal{C} x_t + \mathcal{D} u_t, \tag{3.7}$$

are obtained with the choice $\bar{F} = D_{11} = \bar{B}_2 = 0$, $\bar{B}_1 = \mathcal{B}_1$, $C_1 = \mathcal{C}$, $D_{12} = \mathcal{D}$, $\bar{C}_2 = \mathcal{A}$, $\bar{D}_{22} = \mathcal{B}$ and $\bar{b} = b$. This implies (3.3), (3.4) is a universal approximator for dynamical systems over bounded domains as $q \to \infty$ [165].

Even for linear systems, the set of robust or $\gamma$-robust models is non-convex [220]. Constructing a set of parameters for which (3.3), (3.4) is robust or $\gamma$-robust is further complicated by the presence of the nonlinear activation function in $\Phi$. We will simplify the analysis by replacing $\Phi$ with incremental quadratic constraints.

### 3.3.2 Description of by Incremental Quadratic Constraints

Multiplying (3.5) through by $(y - x)^2$, and combining the two inequalities, we get:

$$\begin{bmatrix} y - x \\ \phi(y) - \phi(x) \end{bmatrix}^\top \begin{bmatrix} 0 & \beta \\ \beta & -2 \end{bmatrix} \begin{bmatrix} y - x \\ \phi(y) - \phi(x) \end{bmatrix} \geq 0. \tag{3.8}$$

For $v^a, v^b \in \mathbb{R}^q$ and $w^a = \Phi(v^a)$, $w^b = \Phi(v^b)$, (3.8) holds for each element with $y = v_i^a$ and $x = v_i^b$. Sector conditions for multiple activation functions can be combined via the "S-Procedure", i.e. introducing multipliers $\lambda_i > 0$, we have:

$$\begin{bmatrix} v_t^a - v_t^b \\ w_t^a - w_t^b \end{bmatrix}^\top \underbrace{\begin{bmatrix} 0 & \beta\Lambda \\ \beta\Lambda & -2\Lambda \end{bmatrix}}_{M(\Lambda)} \begin{bmatrix} v_t^a - v_t^b \\ w_t^a - w_t^b \end{bmatrix} \geq 0, \tag{3.9}$$

where $\Lambda = \mathrm{diag}(\lambda_1, ..., \lambda_q)$.

### 3.3.3 Convex Parametrization of Robust RNNs

Corresponding to the linear system (3.3), we introduce the following implicit, redundant parameterization:

$$G \begin{cases} Ex_{t+1} = Fx_t + B_1 w_t + B_2 u_t + b_x \\ y_t = C_1 x_t + D_{11} w_t + D_{12} u_t + b_y \\ \Lambda v_t = C_2 x_t + D_{22} u_t + b_v \end{cases} \tag{3.10}$$

where $\theta = (E, F, B_1, B_2, C_1, D_{11}, D_{12}, \Lambda, C_2, b_x, b_y, b_v, D_{22})$ are the model parameters with $E$ invertible and $\Lambda \in \mathbb{D}_+$ is the multiplier from (3.9). The explicit system (3.3) can be easily constructed from (3.10) by inverting $E$ and $\Lambda$.

For the robust model set we introduce the following constraint, which is jointly convex in $E, F, C_2, B_1, P, \Lambda$:

$$\begin{bmatrix} E + E^\top - P & -\beta C_2^\top \\ -\beta C_2 & 2\Lambda \end{bmatrix} - \begin{bmatrix} F^\top \\ B_1^\top \end{bmatrix} P^{-1} \begin{bmatrix} F^\top \\ B_1^\top \end{bmatrix}^\top \succ 0. \tag{3.11}$$

The robust model set, we propose is given by equations (3.10), (3.4) parameterized by:

$$\Theta_* := \{\theta : \exists P \succ 0,\ \Lambda \in \mathbb{D}_+ \text{ s.t. } (3.11)\}.$$

For the $\gamma$-robust model set, we introduce the following constraint, which is jointly convex in the model parameters $E, F, C_1, C_2, B_1, B_2, D_{11}, D_{12}$ the stability certificate $P$, multipliers $\Lambda$ and gain bound $\gamma$:

$$\begin{bmatrix} E + E^\top - P & -\beta C_2^\top & 0 \\ -\beta C_2 & 2\Lambda & -\beta D_{22}^\top \\ 0 & -\beta D_{22} & \gamma I \end{bmatrix} - \begin{bmatrix} F^\top \\ B_1^\top \\ B_2^\top \end{bmatrix} P^{-1} \begin{bmatrix} F^\top \\ B_1^\top \\ B_2^\top \end{bmatrix}^\top - \frac{1}{\gamma} \begin{bmatrix} C_1^\top \\ D_{11}^\top \\ D_{12}^\top \end{bmatrix} \begin{bmatrix} C_1^\top \\ D_{11}^\top \\ D_{12}^\top \end{bmatrix}^\top \succ 0. \tag{3.12}$$

The $\gamma$-robust model set is then parameterized by:

$$\Theta_\gamma := \{\theta : \exists P \succ 0,\ \Lambda \in \mathbb{D}_+ \text{ s.t. } (3.12)\}.$$

Note that (3.11) and (3.12) can be written as LMIs via Schur complement, and are jointly convex in the model parameters, stability certificate, multipliers $\Lambda$, and the incremental $\ell_2$ gain bound $\gamma$. Note also that since $P \succ 0$, the upper-left block of each implies that $E + E^\top \succ 0$, hence $E$ is invertible.

The following three theorems establish all models in these sets are in fact robust and $\gamma$-robust as claimed, and furthermore that they are contracting [129].

**Theorem 3.1.** *Suppose that $\theta \in \Theta_\gamma$, then the Robust RNN (3.10), (3.4) has a incremental $\ell_2$-gain bound of $\gamma$.*

*Proof.* See Section 3.5. □

**Theorem 3.2.** *Suppose that $\theta \in \Theta_*$, then the Robust RNN (3.10), (3.4) has a finite incremental $\ell_2$-gain.*

*Proof.* See Section 3.5. □

**Theorem 3.3.** *All models in $\theta \in \Theta_*$ and $\theta \in \Theta_\gamma$ are contracting, i.e. for any input signal, initial conditions are forgotten exponentially.*

*Proof.* See Section 3.5. □

### 3.3.4 Contracting Implicit Recurrent Neural Networks

This section introduces a simplified version of the Robust RNN known as the contracting implicit RNN (ci-RNN), first presented in [176]. Consider the state space model (3.1), (3.2) with the dynamics parameterized as an $L$ layer neural network:

$$z^0 = x_t \quad z^{\ell+1} = \Phi(A_\ell z^\ell + B^\ell + b^\ell) \quad f_\theta(x_t, u_t) = z^L \tag{3.13}$$

for $\ell = 1, ..., L$ and a linear output layer. The dynamics in (3.13) can also be parameterized using the following implicit, redundant parameterization:

$$E_0 h^0 = x, \qquad \mathcal{E}_{\ell+1} h^{\ell+1} = \phi(\mathcal{F}_\ell h^\ell + B_\ell u + b_\ell) \qquad f_\theta(x, u) = E_L h^L, \tag{3.14}$$

for $\ell = 1, ..., L$ where $W_\ell$ and $E_\ell$ are learnable weight matrices and $E_\ell$ are invertible. Note that the implicit and explicit models are input/output equivalent under the coordinate transformation $z_\ell = E_\ell h_\ell$ and $A_\ell = \mathcal{F}_\ell E_{\ell-1}^{-1}$.

We can treat multi-layer networks as a time-varying, periodic, non-linear system by dividing up each $k$ step into $L$ sub-steps so that

$$h_k^{\ell+1} = f^\ell(h_k^\ell, u_k), \quad \ell = 0, ..., L-1 \tag{3.15}$$

where $f^\ell$ refers to the step at one layer in (3.13) and $h_{k+1}^0 = h_k^L$.

We now define the set of contracting implicit RNNs (ci-RNNs): A ci-RNN is a parameterized state space model (3.1), (3.2) with $f_\theta(x, u)$ defined in (3.14) with an additional contraction constraint. We propose to use the following constraints to ensure model stability:

$$\begin{pmatrix} \mathcal{E}_\ell + \mathcal{E}_\ell^\top - P_\ell & \mathcal{F}_\ell^\top \\ \mathcal{F}_\ell & P_{\ell+1} \end{pmatrix} \succeq 0, \quad \ell = 0, ..., L-1 \tag{3.16}$$

with $P_0 = \lambda P_L$. The set of ci-RNNs, denoted $\Theta_{ci}$ is defined as:

$$\Theta_{ci} := \left\{ \theta \ : \ \exists P_0, ..., P_L \in \mathbb{D}_+ \text{ s.t. } P_0 = \lambda P_L, \ E + E^\top \succ 0, \ (3.16) \right\}$$

Note that $\Theta_{ci}$ is convex as it is the intersection a number of semi-definite cones and a linear equality constraint, and for all $\theta \in \Theta_{ci}$, there exists a corresponding explicit RNN (3.13). Fixing $E_\ell = I$ and $P_\ell = I$ recovers the model set used by [148].

**Theorem 3.4.** *Suppose that $\theta \in \Theta_{ci}$, then the dynamics (3.13) are contracting with rate $\lambda$ in the metric $V = \Delta_x^\top E_0^\top P_0^{-1} E_0 \Delta_x$.*

*Proof.* See Section 3.5 □

Unlike the robust RNN, the ci-RNN does not rely on quadratic constraints. Instead, the main idea is to exploit the fact that the elementwise, slope-restricted activation functions are contracting in any diagonal metric. This fact implies that if $A_\ell$ in (3.13) is contracting in a diagonal metric, then the composition of $A_\ell$ and $\phi$ will also be contracting in that metric.

Note that we have presented the model set for the ci-RNN in the multi-layer setting and the model set for Robust-RNN in the single layer setting. This was done to match our approach in [176]. While in principle, it is easy to extend the ideas used in the Robust RNN to multilayer networks, a more effective method to improving the model class expressivity can be achieved via the term $\tilde{D}_{21}$ that is missing from (5.19). This will be further explored in chapters 4 and 5.

### 3.3.5 Expressivity of the Robust RNN Model Set

To be able to learn models for a wide class of systems, the flexibility or expressivity of a model set is important. Our stability and robustness conditions are only sufficient conditions, and could be quite conservative since the only information they use about the activation functions is the slope restriction. Nevertheless, we will see in the empirical results in Section 3.4 that they can give remarkably tight bounds on the incremental $\ell^2$-gain, and in this section we show that our model set contains several previously proposed sets as special cases. First, we show that our parameterization is not restrictive for the case of linear systems.

**Theorem 3.5.** *The Robust RNN set $\Theta_*$ contains all stable LTI models of the form*

$$x_{t+1} = \mathcal{A}x_t + \mathcal{B}u_t, \quad y_t = \mathcal{C}x_t + \mathcal{D}u_t. \tag{3.17}$$

*Proof.* A necessary and sufficient condition for stability of (3.17) is the existence of some $\mathcal{P} \succ 0$ such that:

$$\mathcal{P} - \mathcal{A}^\top \mathcal{P} \mathcal{A} \succ 0. \tag{3.18}$$

For any stable LTI system, the implicit RNN with $\theta$ such that $E = P = \mathcal{P}$, $F = \mathcal{P}\mathcal{A}$, $B_1 = 0$, $B_2 = \mathcal{P}\mathcal{B}$, $C = \mathcal{C}$ and $D = \mathcal{D}$, $C_2 = 0$ and $D_{22} = 0$ has the same dynamics and output. To see that that $\theta \in \Theta_*$,

$$\begin{aligned}(3.18) \Rightarrow & E + E^\top - P - F^\top P^{-1} P P^{-1} F \succ 0 \\ \Rightarrow & \begin{bmatrix} E + E^\top - P - F^\top P^{-1} F & 0 \\ 0 & 2\Lambda \end{bmatrix} \succeq 0 \Rightarrow (3.11)\end{aligned}$$

for any $\Lambda \succ 0$. □

**Remark 3.2.** Essentially the same proof technique but with the strict Bounded Real Lemma can be used to show that $\Theta_\gamma$ contains all LTI models with an $H_\infty$ norm bound of $\gamma$.

Next, we show that our model set includes previously proposed sets of stable RNNs. The equations for a single layer ci-RNN are given by:

$$\mathcal{E} z_{t+1} = \Phi(\mathcal{F} z_t + \mathcal{B} u_t + \mathfrak{b}), \quad y_t = \mathcal{C} z_t + \mathcal{D} u_t \tag{3.19}$$

such that the following contraction condition holds

$$\begin{bmatrix} \mathcal{E} + \mathcal{E}^T - \mathcal{P} & \mathcal{F}^T \\ \mathcal{F} & \mathcal{P} \end{bmatrix} \succ 0 \tag{3.20}$$

where $\mathcal{P} \in \mathbb{D}_+$. The stable RNN (s-RNN), proposed in [148] is a special case of the ci-RNNs with $\mathcal{E} = P = I$.

**Theorem 3.6.** *The Robust RNN set $\Theta_*$ contains all ci-RNNs, and hence also all s-RNNs.*

*Proof.* See Section 3.5 □

## 3.4 Numerical Example

We will compare the proposed Robust RNN with the (Elman) RNN [64] described by (3.6), (3.7) with $B_1 = I$ and Long Short Term Memory (LSTM) [93], which is a widely used model class that was originally proposed to resolve issues related to stability. In addition, we compare with two previously published stable model sets, the contracting implicit RNN (ci-RNN) [176] and stable RNN (sRNN) [148]. The LSTM used a tanh activation function and all other models used a ReLU activation function. All outputs are linear functions of the state.

The LSTM is described by the following equations:

$$\text{LSTM} \begin{cases} i_{t+1} = \sigma(W_{xi}x_t + W_{ii}u_{t+1} + b_i), \\ f_{t+1} = \sigma(W_{xf}x_t + W_{if}u_{t+1} + b_f), \\ g_{t+1} = \sigma(W_{xg}x_t + W_{ig}u_{t+1} + b_g), \\ o_{t+1} = \sigma(W_{xo}x_t + W_{io}u_{t+1} + b_o), \\ c_{t+1} = f_{t+1} \odot c_t + i_{t+1} \odot g_{t+1}, \\ x_{t+1} = o_{t+1} \odot \tanh(c_{t+1}), \end{cases} \tag{3.21}$$

where $c_t, x_t \in \mathbb{R}^n$, are the cell state and hidden state, $u_t \in \mathbb{R}^m$ is the input and $\odot$ is the Hadamard product and $\sigma$ is the sigmoid function.

To generate data, we use a simulation of four coupled mass spring dampers. The goal is to identify a mapping from the force on the initial mass to the position of the final mass. Nonlinearity is introduced through the springs' piecewise linear force profile

$$F_i(d) = k_i\Gamma(d), \quad \Gamma(d) = \begin{cases} d + 0.75, & -d \leq -1, \\ 0.25d, & -1 < d < 1, \\ d - 0.75, & d \geq 1, \end{cases} \tag{3.22}$$

where $k_i$ is the spring constant for the $i$th spring and $d$ is the displacement between the carts. A schematic is shown in Fig. 3.2. The masses are $[m_1, ..., m_4] =$

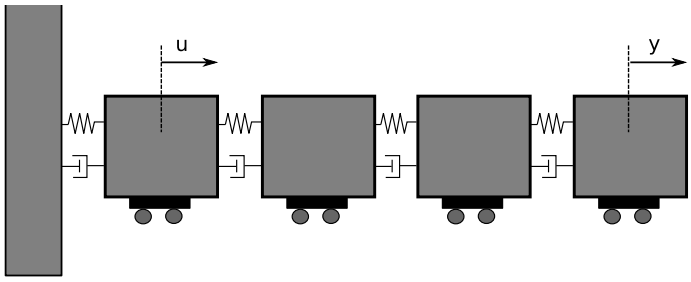



**Figure 3.2** – Nonlinear mass spring damper schematic.

$[1/4, 1/3, 5/12, 1/2]$, the linear damping coefficients used are $[c_1, ..., c_4] = [1/4, 1/3, 5/12, 1/2]$ and spring constants used in (3.22) are $[k_1, ..., k_4] = [1, 5/6, 2/3, 1/2]$.

Note that the dynamics of this system are representable in the proposed model set. Although the true state dimension is 8, a state dimension of 10 was used for each model as this substantially improved the fit over all model classes. Excessive model dimension is commonly observed to improve the trainability of neural networks [71].

We excite the system with a piecewise constant input signal that changes value after an interval distributed uniformly in $[0, \tau]$ and takes values that are normally distributed with standard deviation $\sigma_u$. Measurements have Gaussian noise of approximately 30dB added. To generate data, we simulate the system for $T/5$ seconds and sample the system at 5Hz to generate $T$ data points with an input signal characterized by $\tau = 20s$ and $\sigma_u = 3N$. The training data consists of 100 batches of length 1000. We also generate a validation set with $\tau = 20s$, $\sigma_u = 3N$ and length 5000 that is used for early stopping. To test the model performance, we generate test sets of length 1000 with $\tau = 20s$ and varying $\sigma_u$.

### 3.4.1 Training Procedure and Model Evaluation

We fit Robust RNNs to the data by minimizing the simulation error in $\ell^2$ norm:

$$\min_{\theta \in \Theta, a^k \in \mathbb{R}^n} J_{se} = ||\tilde{y}^k - S_{a^k}(\tilde{u}^k)||^2.$$

Here, $\tilde{u}^k$ and $\tilde{y}^k$ are the input and output data for the $k$th batch, $\Theta$ refers to one of the model sets $\Theta_\gamma$ or $\Theta_*$ characterized by (3.11) or (3.12). We treat the initial condition $a^k$ for batch $k$ as a parameter to be trained during optimization.

The constraint $\theta \in \Theta$ is enforced using logarithmic barrier functions, i.e., we minimize the following objective function:

$$J = ||\tilde{y}^k - S_{a^k}(\tilde{u}^k)||^2 - \alpha \log\det(M - \epsilon I), \tag{3.23}$$

for some small $\epsilon > 0$, where $M$ is the Schur complement of (3.11) or (3.12) and $\alpha$ is the barrier parameter. The matrix $M$ has dimension $2n + q$ or $2n + q + p$, and computing the gradient of the logarithmic barrier term requires $M^{-1}$. This can be calculated efficiently for moderately sized models. For large problems, the Burer-Monteiro method [36] provides a computationally simpler, albeit nonconvex alternative.

The objective (3.23) is minimized using stochastic gradient descent and the ADAM optimizer [111] The term epoch refers to one complete pass through the training data. A backtracking line search ensures strict feasibility throughout optimization. After 10 epochs without an improvement in validation performance, we decrease the learning rate by a factor of 0.25 and decrease $\alpha$ by a factor of 10. When $\alpha$ reaches a final value of $1 \times 10^{-7}$, we finish training. All code is written using Pytorch 1.60 and run on a standard desktop CPU. The code is available at the following link: https://github.com/imanchester/RobustRNN/.

Quality of fit is measured by normalized simulation error:

$$\text{NSE} = \frac{||\tilde{y} - y||}{||\tilde{y}||}$$

where $y, \tilde{y} \in \ell_2^p$ are the simulated and measured outputs, respectively. Since we are learning input-output dynamics, during testing the first 200 samples are ignored to

let initial condition transients fade. Robustness is assessed by solving:

$$\hat{\gamma} = \max_{u,v,a} \frac{||S_a(u) - S_a(v)||}{||u - v||}, \quad u \neq v.$$

using gradient ascent. The value of $\hat{\gamma}$ is the "worst-case observed sensitivity" and is a lower bound on the true Lipschitz constant of the model.

### 3.4.2 Results

The validation performance versus the number of epochs is shown in Fig. 3.3. Each epoch training the Robust RNN takes twice as long as the LSTM due to the evaluation of the logarithmic barrier functions and the backtracking line search, however, we will see that the model offers both stability/robustness guarantees and superior generalizability.

Figure 3.4 presents the boxplots of NSE and a comparison of the medians on test sets with input signals with varying $\sigma_u$. In each plot, there is a trough around $\sigma_u = 3$ corresponding to the training data distribution. For the LSTM and RNN, the model performance quickly degrades with varying $\sigma_u$, whereas all the stable models exhibit a more gradual decline in performance. This supports the claim that model stability constraints can improve model generalization. The Robust RNN set $\Theta_*$ uniformly outperforms all other models.

Figure 3.5 plots the worst-case observed sensitivity versus median nominal test performance ($\sigma_u = 3$). The Robust RNNs show the best trade-off between nominal performance and robustness, i.e., closest to the lower-left corner. For instance, if we compare the LSTM with the Robust RNN ($\Theta_*$), we observe that the Robust RNN has a slightly lower NSE but a Lipschitz constant almost fifteen times smaller. We also observe that the guaranteed upper bounds are quite tight to the observed lower bounds on the Lipschitz constant, especially for the set $\Theta_3$.

Fig. 3.6 shows model predictions for the RNN, LSTM and Robust RNN for an input with $\sigma_u = 10$. We can see that even with an input much larger than the training

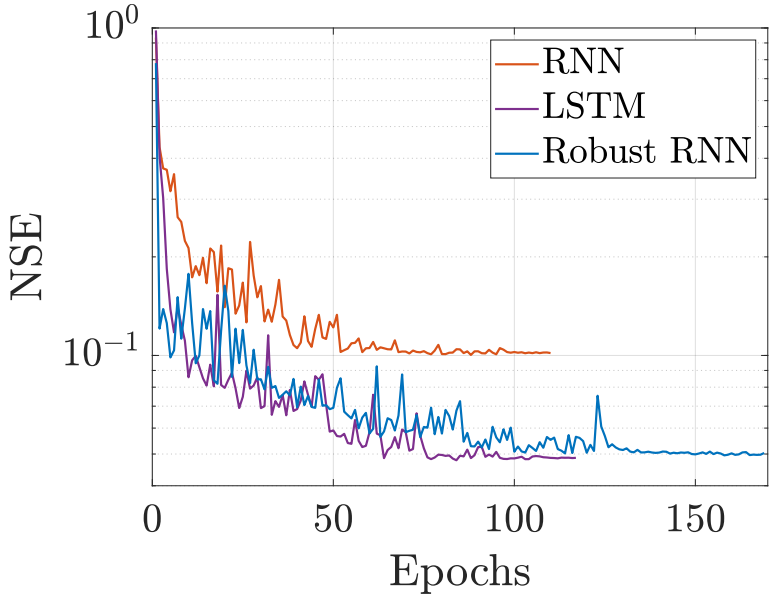



**Figure 3.3** – Validation performance versus epochs. The RNN, LSTM and Robust RNN take an average of 10.1, 18.7 and 37.6 seconds per epoch, respectively.

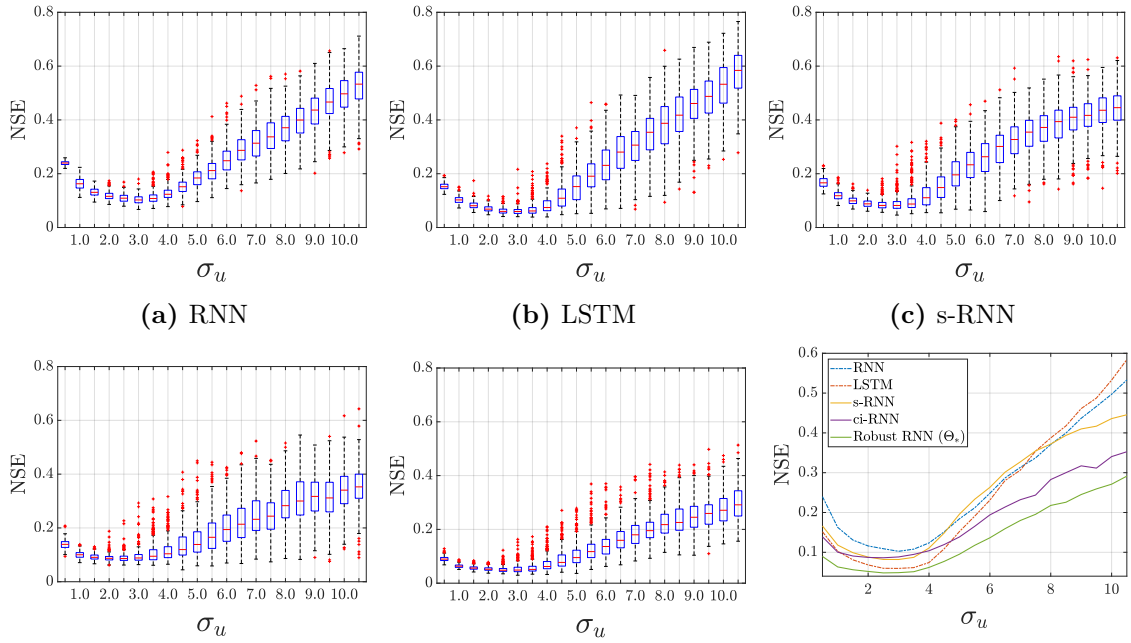
data, the Robust RNN predicts the measured data fairly well while LSTM and RNN deviate significantly from the measured data.

## 3.5 Proofs

### Proof of Theorem 3.1

*Proof.* Consider two solutions $x^a, x^b \in \ell_{2e}^n$ and outputs $y^a, y^b \in \ell_{2e}^p$ to the system (3.1), (3.2) with initial conditions $a, b \in \mathbb{R}^n$ and inputs $u^a, u^b \in \ell_{2e}^m$. Let $\Delta u = u^a - u^b$, $\Delta x = x^a - x^b$, $\Delta v = v^a - v^b$, $\Delta w = w^a - w^b$ and $\Delta y = y^a - y^b$.

To establish the incremental $\ell_2$-gain bound, we first left and right multiply (3.12) by the vectors $\left[\Delta x_t^\top, \Delta w_t^\top, \Delta u_t^\top\right]$ and $\left[\Delta x_t{}^\top, \Delta w_t{}^\top, \Delta u_t{}^\top\right]^\top$. Applying the bound $-E^\top P^{-1} E \preceq P - E - E^\top$ [214] and introducing the storage function $V_t = \Delta x_t^\top E^\top P^{-1} E \Delta x_t$




**(a)** RNN **(b)** LSTM **(c)** s-RNN

**(d)** ci-RNN **(e)** Robust RNN ($\Theta_*$) **(f)** Comparison of median model performance.

**Figure 3.4** – Boxplots showing model NSE performance for 300 input realizations for varying $\sigma_u$.

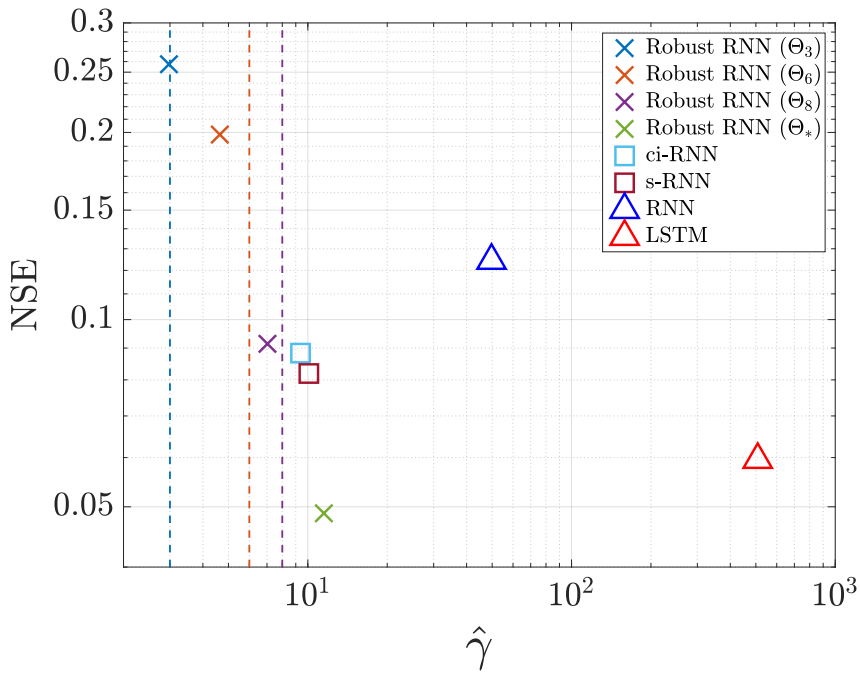



**Figure 3.5** – Test performance with ($\sigma_u = 3$) versus observed worst case sensitivity. The vertical lines are the upper bound on the Lipschitz constant.

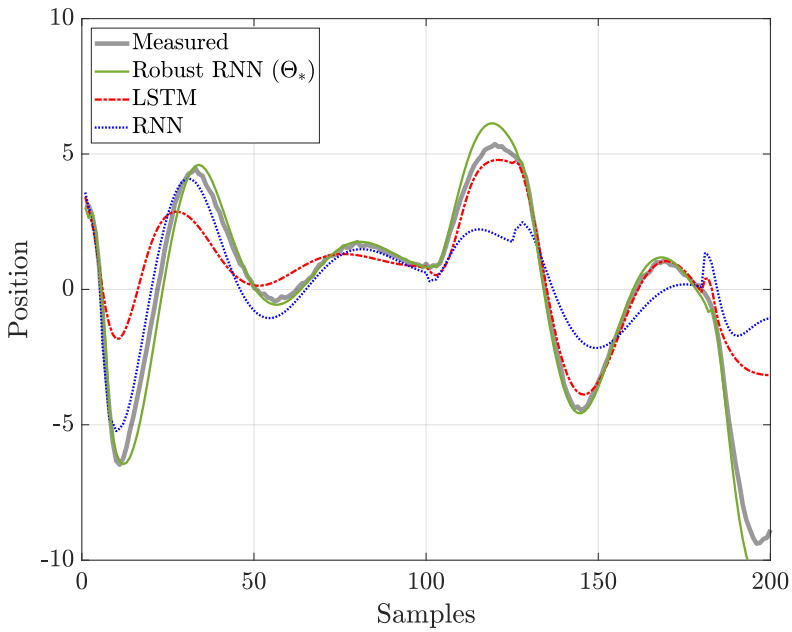



**Figure 3.6** – Example simulation of models with $\sigma_u = 10$, the robust RNN significantly outperforms the other models.

gives

$$V_{t+1} - V_t < \gamma|\Delta u_t|^2 - \frac{1}{\gamma}|\Delta y_t|^2 - \begin{bmatrix} \Delta v_t \\ \Delta w_t \end{bmatrix}^\top M(\Lambda) \begin{bmatrix} \Delta v_t \\ \Delta w_t \end{bmatrix}$$

for $\Delta x_t \neq 0$. Using (3.9) and summing over $[0, T]$ gives

$$V_T - V_0 < \gamma\|\Delta u\|_T^2 - \frac{1}{\gamma}\|\Delta y\|_T^2,$$

for $\Delta_x \neq 0$, so the incremental $\ell_2$-gain condition (2.21) follows with $d(a, b) = \gamma V_0$. □

## Proof of Theorem 3.2

*Proof.* Note that if the LMI condition (3.11) is satisfied, there exists a sufficiently large $\gamma$ such that (3.12) holds for any choice of $B_2$, $C_1$, $D_{11}$, $D_{12}$ and $D_{22}$. Since (3.11) implies (3.12) for some sufficiently large $\gamma$, from Theorem 3.1 the Robust RNN (3.10), (3.4) has a finite incremental $\ell_2$-gain bound $\gamma$. □

## Proof of Theorem 3.3

*Proof.* From the proof of Theorem 3.1, if $\Delta u = 0$ then $V_{t+1} < V_t$ when $\Delta x_t \neq 0$. Since both $V_t$ and $V_{t+1}$ can be expressed as quadratic forms in $\Delta x_t$, it follows that $V_{t+1} \leq \alpha V_t$ for some $\alpha \in [0, 1)$ and $V_t \leq \alpha^t V_0$. Since $P \succ 0$ and $E$ is non-singular this implies $|\Delta x_t| \to 0$ exponentially. □

## Proof of Theorem 3.4

*Proof.* We would like to show that the condition (3.16) implies the existence of a contraction metric $V_k$ for the system (3.13), for which $V_{k+1} \leq \lambda V_k$. Via Schur complement (3.16) is equivalent to:

$$E_\ell + E_\ell^\top - P_\ell - W_\ell^\top P_{\ell+1}^{-1} W_\ell \succeq 0.$$

For all admissible activation functions (slope restricted to the interval $[-1, 1]$) and diagonal $P \succ 0$, we have $(h^a - h^b)^\top P^{-1}(h^a - h^b) \geq (\phi(h^a) - \phi(h^b))^\top P^{-1}(\phi(z^a) - \phi(z^b))$. Left and right multiplying by $\Delta_h$ gives

$$\Delta_{h_{\ell+1}}^\top E_{\ell+1}^\top P_{\ell+1}^{-1} E_{\ell+1} \Delta_{h_{\ell+1}} - \Delta_h^\top (E + E^\top - P)\Delta_h \leq 0.$$

Introducing the storage function $V_k^\ell(\Delta_{h_k}^\ell) = {\Delta_{h_k}^\ell}^\top E_\ell^\top P_\ell^{-1} E_\ell \Delta_{h_k}^\ell$ and using the bound $E^\top P^{-1} E \succ E + E^\top - P$, we can see that $V_k^{\ell+1} - V_k^\ell < 0$. Summing this from $\ell = 0, ...., L-1$ gives $V_k^L - V_k^0 \leq 0$. Due to the periodicity, we have $V_{k+1}^0 = \lambda V_k^L$, so $V_{k+1}^0 \leq \lambda V_k^0$, and the system is contracting in the metric $V_k^0$. □

## Proof of Theorem 3.6

*Proof.* For any ci-RNN, there is a robust RNN with the same dynamics and output with $\theta$ such that $F = 0$, $E = \mathcal{E}$, $B_1 = I$, $B_2 = 0$, $C_1 = \mathcal{C}$, $D_{11} = 0$, $D_{12} = \mathcal{D}$, $\Lambda^{-1} C_2 = \mathcal{F}$, $\Lambda^{-1} D_{22} = \mathcal{B}$, $\mathfrak{b} = \Lambda^{-1} b$, $\Lambda = P^{-1}$ and $P = \mathcal{P}$. By substituting $\theta$ into (3.10) and (3.4), we recover the dynamics and output of the ci-RNN in (3.19). For these parameters, $\theta \in \Theta_*$:

$$\begin{aligned}
(3.20) &\implies E + E^\top - P - C_2^\top \Lambda^{-1} P^{-1} \Lambda^{-1} C_2 \succ 0 \\
&\implies \begin{bmatrix} E + E^\top - P & C_2^\top \\ C_2 & 2\Lambda - P^{-1} \end{bmatrix} \succ 0 \implies (3.11).
\end{aligned}$$

The remaining conditions $P \succ 0$, $\Lambda \in \mathbb{D}_+$ and $E + E^\top \succ 0$ follow by definition. □

# Chapter 4

# Lipschitz Bounded Equilibrium Networks

In Chapter 3, we studied a class of recurrent neural networks that is closely related to the Lure feedback interconnection and showed that a convex parameterization of sequence-to-sequence maps with prescribed stability and Lipschitz bounds can be constructed. In the proposed model, the term $\tilde{D}_{21}$ was set to zero, due to well-posedness concerns implied by the algebraic loop $v = \tilde{D}_{21}\phi(v)$ that would otherwise appear in the feedback interconnection between (3.3) and (3.4).

In this chapter, we study this algebraic loop in more detail and show that it is actually an instance of a class of neural networks called an *Equilibrium Network*. We derive less conservative well-posedness conditions than those that appeared previously in the literature and show that the tightest known bounds on the Lipschitz constant currently known can be easily imposed during training using unconstrained optimization: no projections or barrier functions are required. These results are proved by establishing novel operator splitting on non-Euclidean spaces and contracting neural ODEs. In image classification experiments on the MNIST and CIFAR10 datasets, we show that the Lipschitz bounds are very accurate and improve the robustness to adversarial attacks.

### Publications

Some of the content of this chapter can be found in the following preprint:

Max Revay, Ruigang Wang, and Ian R. Manchester. Lipschitz bounded equilibrium networks. *arXiv preprint arXiv:2104.05942*, 2021.

## 4.1 Introduction

Deep neural network models have revolutionized the field of machine learning: their accuracy on practical tasks such as image classification and their scalability have led to an enormous volume of research on different model structures and their properties [118]. In particular, deep residual networks with skip connections [87] have had a major impact, and neural ODEs have been proposed as an analog with "implicit depth" [39]. Recently, a new structure has gained interest: *equilibrium networks* [14, 238], a.k.a. *implicit deep learning models* [62], in which model outputs are defined by implicit equations incorporating neural networks. This model class is very flexible: it is easy to show that it includes many previous structures as special cases, including standard multilayer networks, residual networks, and (in a certain sense) neural ODEs.

However, model flexibility in machine learning is always in tension with model *regularity* or *robustness*. While deep learning models have exhibited impressive generalization performance in many contexts it has also been observed that they can be very brittle, especially when targeted with adversarial attacks [209]. In response to this, there has been a major research effort to understand and certify robustness properties of deep neural networks, e.g. [170, 211, 123, 43] and many others. Global Lipschitz bounds (a.k.a. incremental gain bounds) provide a somewhat crude but nevertheless highly useful proxy for robustness [216, 66], and appear in several analyses of generalization (e.g. [18, 257]).

Inspired by both of these lines of research, we propose new parameterizations of equilibrium networks with guaranteed Lipschitz bounds. We build directly on the monotone operator framework of [238] and the work of [66] on Lipschitz bounds.

Our main contribution is the ability to enforce tight bounds on the Lipschitz constant of an equilibrium network during training with essentially *no extra computational effort*. In addition, we prove the existence of solutions with less restrictive conditions on the weight matrix and more natural assumptions on the activation functions via novel connections to convex optimization and contracting dynamical systems. Finally, we show via small-scale image classification experiments that the proposed parameterizations can provide significant improvement in robustness to adversarial attacks with little degradation in nominal accuracy. Furthermore, we observe small gaps between certified Lipschitz upper bounds and observed lower bounds computed via adversarial attack.

## 4.2 Related work

**Equilibrium networks, Implicit Deep Models, and Well-Posedness.** As mentioned above, it has been recently shown that many existing network architectures can be incorporated into a flexible model set called an equilibrium network [14, 238] or implicit deep model [62]. In this unified model set, the network predictions are made not by forward computation of sequential hidden layers, but by finding a solution to an implicit equation involving a single layer of all hidden units. One major question for this type of networks is its well-posedness, i.e. the existence and uniqueness of a solution to the implicit equation for all possible inputs. [62] proposed a computationally verifiable but conservative condition on the spectral norm of hidden unit weight. In [238], a less conservative condition was developed based on monotone operator theory. Similar monotonicity constraints were previously used to ensure well-posedness of a different class of implicit models in the context of nonlinear system identification [214, Theorem 1]. On the question of well-posedness, our contribution is a more flexible model set and more natural assumptions on the activation functions: that they

are monotone and slope-restricted.

**Neural Network Robustness and Lipschitz Bounds.** The Lipschitz constant of a function measures the worst-case sensitivity of the function, i.e., the maximum "amplification" of the difference in inputs to differences in the outputs. The key features of a good Lipschitz bounded learning approach include a tight estimation for Lipschitz constant and a computationally tractable training method with bounds enforced. For deep networks, [216] proposed a computationally efficient but conservative approach since its Lipschitz constant estimation method is based on the composition of estimates for different layers, while [8] proposed a combination of a novel activation function and weight constraints. For equilibrium networks, [62] proposed an estimation of Lipschitz bounds via input-to-state (ISS) stability analysis. [66] estimates for deep networks based on incremental quadratic constraints and semidefinite programming (SDP) were shown to give state-of-the-art results, however, this was limited to the analysis of previously trained networks. The SDP test was incorporated into training via the alternating direction method of multipliers (ADMM) in [164], however due to the complexity of the SDP, the training times recorded were almost 50 times longer than for unconstrained networks. Our approach uses a similar condition to [66] applied to equilibrium networks, however, we introduce a novel direct parameterization method that enables learning robust models via unconstrained optimization, removing the need for computationally expensive projections or barrier terms.

## 4.3 Problem Formulation and Preliminaries

### 4.3.1 Problem statement

We consider the weight-tied network in which $x \in \mathbb{R}^d$ denotes the input, and $z \in \mathbb{R}^n$ denotes the hidden units, $y \in \mathbb{R}^p$ denotes the output, given by the following implicit

equation

$$z = \sigma(Wz + Ux + b_z), \quad y = W_o z + b_y \tag{4.1}$$

where $W \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times d}$, and $W_o \in \mathbb{R}^{p\times n}$ are the hidden unit, input, and output weights, respectively, $b_z \in \mathbb{R}^n$ and $b_y \in \mathbb{R}^p$ are bias terms. The implicit framework includes most current neural network architectures (e.g., deep and residual networks) as special cases. To streamline the presentation, we assume that $\sigma : \mathbb{R} \to \mathbb{R}$ is a single nonlinearity applied elementwise, although our results also apply in the case that each channel has a different activation function, nonlinear or linear.

Equation (4.1) is called an equilibrium network since its solutions are the equilibrium points of the difference equation $z^{k+1} = \sigma(Wz^k + Ux + b_z)$ or the ODE $\dot{z}(t) = -z(t)+\sigma(Wz(t)+Ux+b_z)$. Our goal is to learn equilibrium networks (4.1) possessing the following two properties:

- **Well-posedness:** For every input $x$ and bias $b_z$, equation (4.1) admits a unique solution $z$.
- **$\gamma$-Lipschitz:** It has a finite Lipschitz bound of $\gamma$, i.e., for any input-output pairs $(x_1, y_1)$, $(x_2, y_2)$ we have $\|y_1 - y_2\|_2 \le \gamma\|x_1 - x_2\|_2$.

### 4.3.2 Preliminaries

**Monotone operator theory.** In this chapter, we will use the monotone operator theory on non-Euclidean spaces [19], see Section 2.5.3 for a brief review of the relevant aspects of monotone operator theory or [183] for a more in depth treatment. In particular, we are interested in a finite-dimensional Hilbert space $\mathcal{H}$, which we identify as $\mathbb{R}^n$ equipped with a weighted inner product $\langle x, y\rangle_Q := y^\top Qx$ where $Q \succ 0$. The benefit of using a non-Euclidean space is that we can construct a more expressive set of equilibrium networks.

An operator is a set-valued or single-valued function defined on a subset of the space $A \subseteq \mathcal{H}\times\mathcal{H}$; we use the notation $A(x) = \{y \mid (x, y) \in A\}$. A function $f : \mathcal{H} \to \mathbb{R}\cup\{\infty\}$

is proper if $f(x) < \infty$ for at least one $x$. The subdifferential and proximal operators of a proper function $f$ are defined as

$$\partial f(x) := \{g \in \mathcal{H} \mid f(y) \geq f(x) + \langle y - x, g\rangle_Q, \forall y \in \mathcal{H}\},$$
$$\mathbf{prox}_f^\alpha(x) := \{z \in \mathcal{H} \mid z = \arg\min_u \frac{1}{2}\|u - x\|_Q^2 + \alpha f(u)\}$$

respectively, where $\|x\|_Q := \sqrt{\langle x, x\rangle_Q}$ is the induced norm. For $n = 1$, we only consider the case of $Q = 1$. An operator $A$ is monotone if $\langle u - v, x - y\rangle_Q \geq 0$ and strongly monotone with parameter $m$ if $\langle u - v, x - y\rangle_Q \geq m\|x - y\|_Q^2$ for all $(x, u), (y, v) \in A$. The operator splitting problem is that of finding a zero of a sum of two operators $A$ and $B$, i.e. find an $x$ such that $0 \in (A + B)(x)$.

**Dynamical systems theory.** Another approach to analyzing the properties of (4.1) is to treat it as the equilibrium point of a dynamical system

$$\dot{z}(t) = f(z(t)). \tag{4.2}$$

Then, the well-posedness and robustness properties of (4.1) can be guaranteed by imposing the corresponding properties on the dynamical system's solution set. A central focus in robust and nonlinear control theory for more than 50 years – and largely unified by the modern theory of integral quadratic constraints [145] – has been on systems which are interconnections of linear mappings and "simple" nonlinearities, i.e. those easily bounded in some sense by quadratic functions. Fortuitously, this characteristic is shared with deep, recurrent, and equilibrium neural networks, a connection that we use heavily in this chapter and has previously been exploited by [66, 62, 177] and others. We are interested in ensuring contraction:

**Definition 4.1** (Contraction [129]). *A continuous time dynamical system (4.2) is termed contracting with rate $\lambda$ if for any two solutions $x^a(t)$ and $x^b(t)$, given the same input $u(t)$, there exists some function $b(x^a(0), x^b(0)) > 0$ with $b(x, x) = 0$, such that the resulting trajectories satisfy $\|x^a(t) - x^b(t)\| \leq b(x^a(0), x^b(0))e^{-\lambda t}$.*

Contraction can be established by finding a Riemannian metric with respect to which nearby trajectories converge, which is a differential analog of a Lyapunov function.

For time-invariant systems, contraction implies a unique equilibrium exists and possesses a certain level of robustness. Moreover, the contraction can also be linked to monotone operators, i.e., a system is contracting w.r.t. to a constant (state-independent) metric $Q$ if and only if the operator $-f$ is strongly monotone w.r.t. $Q$-weighted inner product.

## 4.4 Main Results

This section contains the main theoretical results of the chapter: conditions implying well-posedness and Lipschitz-boundedness of equilibrium networks, and direct (unconstrained) parameterizations such that these conditions are automatically satisfied.

**Assumption 1.** The activation function $\sigma$ is piecewise differentiable and slope-restricted in $[0, 1]$, i.e.,

$$0 \leq \frac{\sigma(x) - \sigma(y)}{x - y} \leq 1, \ \forall x, y \in \mathbb{R}, \ x \neq y. \tag{4.3}$$

**Remark 4.1.** In [45], it is shown that Assumption 1 is equivalent to the assumption on $\sigma$ in [238], i.e. that $\sigma(\cdot) = \mathbf{prox}_f^1(\cdot)$ for some piecewise twice-differentiable and proper convex function $f$. However, the above assumption is arguably more natural, since it is easily verified for standard activation functions. Note also that if different channels have different activation functions, then we simply require that they all satisfy (4.3).

The following conditions are central to our results on well-posedness and Lipschitz bounds:

**Condition 4.1.** There exists a $\Lambda \in \mathbb{D}^+$, with $\mathbb{D}^+$ denoting diagonal positive-definite matrices, such that $W$ satisfies

$$2\Lambda - \Lambda W - W^T \Lambda \succ 0. \tag{4.4}$$

**Condition 4.2.** Given a prescribed Lipschitz bound $\gamma > 0$, there exists $\Lambda \in \mathbb{D}^+$ such that $W, W_o, U$ satisfy

$$2\Lambda - \Lambda W - W^T\Lambda - \frac{1}{\gamma}W_o^T W_o - \frac{1}{\gamma}\Lambda U U^T \Lambda \succ 0. \tag{4.5}$$

**Remark 4.2.** Note that Condition 4.2 implies Condition 4.1 since $1/\gamma(W_o^T W_o + \Lambda U U^T \Lambda) \succeq 0$. As a partial converse, if Condition 4.1 holds, then for any $W_o, U$ there exist a sufficiently large $\gamma$ such that Condition 4.2 is satisfied.

The main theoretical results of this chapter are the following:

**Theorem 4.1.** *If Assumption 1 and Condition 4.1 hold, then the equilibrium network (4.1) is well-posed, i.e. for all $x$ and $b_z$, equation (4.1) admits a unique solution $z$. Moreover, it has a finite Lipschitz bound from $x$ to $y$.*

*Proof.* See Section 4.7 □

**Theorem 4.2.** *If Assumption 1 and Condition 4.2 hold, then the equilibrium network (4.1) is well-posed and has a Lipschitz bound of $\gamma$.*

*Proof.* See Section 4.7 □

As a consequence, we call (4.1) a *Lipschitz bounded equilibrium network* (LBEN) if its weights satisfy (4.4). It is called an LBEN $\gamma$ if its weights also satisfy (4.5). The full proofs appear in Section 4.7, but here we sketch some of the main ideas. We can represent (4.1) as an algebraic interconnection between linear and nonlinear parts:

$$v = Wz + Ux + b_z, \quad z = \sigma(v), \quad y = W_o z + b_y. \tag{4.6}$$

It can be shown that for any pair of solutions to the nonlinear part $z_a = \sigma(v_a), z_b = \sigma(v_b)$, if we define $\Delta_v = v_a - v_b$ and $\Delta_z = z_a - z_b$ then Assumption 1 implies the following:

$$\langle \Delta_v - \Delta_z, \Delta_z \rangle_\Lambda \geq 0. \tag{4.7}$$

for any $\Lambda \in \mathbb{D}^+$. This and Condition 4.1 can be used to prove global stability of a unique equilibrium of the differential equation

$$\dot{v} = -v + W\sigma(v) + Ux + b_z,$$

for any input and bias, which proves there is a unique solution to (4.1). Next, straightforward manipulations of Condition 4.2 show that any pairs of inputs $x_a, x_b$ and outputs $y_a, y_b$ satisfy the following, where $\Delta_x = x_a - x_b$ and $\Delta_y = y_a - y_b$:

$$\gamma\|\Delta_x\|_2^2 - \frac{1}{\gamma}\|\Delta_y\|_2^2 \geq 2\langle \Delta_v - \Delta_z, \Delta_z\rangle_\Lambda \geq 0,$$

where the inequality comes (4.7). This implies the Lipschitz bound $\|\Delta_y\|_2 \leq \gamma\|\Delta_x\|_2$.

### 4.4.1 Direct Parameterization for Unconstrained Optimization

Training a network that satisfies Condition 4.1 or 4.2 can be formulated as an optimization problem with convex constraints. In fact, Condition 4.1 is a linear matrix inequality (LMI) in the variables $\Lambda$ and $\Lambda W$, from which $W$ can be determined uniquely. Similarly, via Schur complement, Condition 4.2 is an LMI in the variables $\Lambda, \Lambda W, \Lambda U, W_o$, and $\gamma$, from which all network weights can be determined. In a certain theoretical sense LMI constraints are tractable – [153] proved they are polynomial-time solvable – however for even moderate-scale networks (e.g. $\leq 100$ activations) the associated barrier terms, projections and/or line searches become a major computational bottleneck. Furthermore, the constrained optimization schemes for incorporating constraints (4.1) or (4.2) can be difficult implement and introduce a large number of hyperparameters.

In this chapter, we propose a 'direct' parameterization that allows learning via unconstrained optimization, i.e. all network parameters are constructed from free (unconstrained) matrix variables in such a way that LMI constraints (4.4) or (4.5) are automatically satisfied.

For well-posedness, i.e. Condition (4.1), we parameterize via the following free variables: $V \in \mathbb{R}^{n\times n}$, $d \in \mathbb{R}^n$, and skew-symmetric[1] matrix $S = -S^T \in \mathbb{R}^{n\times n}$, from which the hidden unit weight is

$$W = I - \Psi(V^T V + \epsilon I + S), \tag{4.8}$$

where $\Psi = \operatorname{diag}\left(e^d\right)$ and $\epsilon > 0$ is some small constant to ensure strict positive-definiteness. Then it follows from straightforward manipulations that Condition 4.1 holds with $\Lambda = \Psi^{-1}$ if and only if $W$ can be written as (4.8). When $\Psi = I$, this reduces to the exactly the same parameterization as was used in [238].

Similarly, for a specific Lipschitz bound, i.e. Condition 4.2, we add to the parameterization the free input and output weights $U$ and $W_o$, and arbitrary $\gamma > 0$. We can construct

$$W = I - \Psi\left[V^T V + \epsilon I + S + \frac{1}{2\gamma}(W_o^\top W_o + \Psi^{-1} U U^\top \Psi^{-1})\right], \tag{4.9}$$

for which (4.5) is automatically satisfied. Again, it can easily be verified that this construction is necessary and sufficient, i.e. any $W$ satisfying (4.5) can be constructed via (4.9).

### 4.4.2 Monotone Operator Perspective

The LBEN (4.1) is closely related to an operator splitting problem:

**Proposition 4.1.** *Finding a solution of LBEN (4.1) is equivalent to solving the well-posed operator splitting problem* $0 \in (A + B)(z)$ *with the operators*

$$A(z) = (I - W)(z) - (Ux + b_z), \quad B = \partial \mathfrak{f} \tag{4.10}$$

*where* $\mathfrak{f}(z) := \sum_{i=1}^n \lambda_i f(z_i)$ *with* $\lambda_i$ *as the ith diagonal element of* $\Lambda$.

[1]Note that $S$ can be parameterized via its upper or lower triangular components, or via $S = N - N^T$ with $N$ free, which can be more straightforward if $W$ is defined implicitly via linear operators, e.g. convolutions.

*Proof.* See Section 4.7 □

The proof appears in Section 4.7 and Theorem 4.1 follows directly since the above operator splitting problem has a unique solution for any $x, b_z$. The solution can be efficiently computed via various of operator splitting algorithms, e.g., ADMM [31] and Peaceman-Rachford splitting [107]. In [238], it was found that Peaceman-Rachford splitting converges very rapidly when properly tuned, and our experience agrees with this. The convergence properties of the equilibrium solving algorithm can also be improved by using suitable regularization techniques [16].

**Gradient Backpropagation.** As shown in [238, Section 3.5], the gradients of the loss function $\ell(\cdot)$ can be represented by

$$\frac{\partial \ell}{\partial(\cdot)} = \frac{\partial \ell}{\partial z_\star}(I - JW)^{-1} J \frac{\partial(Wz_\star + Ux + b_z)}{\partial(\cdot)} \tag{4.11}$$

where $z_\star$ denotes the solution of (4.1), $(\cdot)$ denotes some learnable parameters in the parameterization (4.8) or (4.9), and $J \in \mathrm{D}\sigma(Wz_\star + Ux + b_z)$ with $\mathrm{D}\sigma$ as the Clarke generalized Jacobian of $\sigma$. Since $\sigma$ is piecewise differentiable, then the set $\mathrm{D}\sigma(Wz_\star + Ux + b_z)$ is a singleton almost everywhere. The following proposition reveals that (4.11) is well-defined, see proof in Section 4.7.

**Proposition 4.2.** *The matrix $I - JW$ is invertible for all $z_\star$, $x$ and $b_z$.*

*Proof.* See Section 4.7 □

### 4.4.3 Contracting Neural ODEs

In this section, we will prove the existence of a solution to (4.1) from a different perspective: by showing it is the equilibrium of a contracting dynamical system (a "neural ODE"). We first add a smooth state $v(t) \in \mathbb{R}^n$ to avoid the algebraic loop in (4.6). This idea has long been recognized as helpful for well-posedness questions

[251]. We define the dynamics of $v(t)$ by the following ODE:

$$(I - \tfrac{1}{2}W)\dot{v}(t) = -v(t) + Wz(t) + Ux + b_z, \quad z(t) = \sigma(v(t)). \tag{4.12}$$

The well-posedness of (4.1) is equivalent to the existence and uniqueness of an equilibrium of (4.12) for all $x$ and $b_z$, which is established by the following proposition.

**Proposition 4.3.** *If Assumption 1 and Condition 4.1 hold, then the neural ODE (4.12) is contracting w.r.t. some constant metric $P \succ 0$.*

*Proof.* See Section 4.7 □

### 4.4.4 Model Expressivity

In [238], a set of well-posed equilibrium network, called monotone operator equilibrium network (MON), is introduced via the following parameterization

$$W = (1-m)I - A^\top A + B^\top - B \tag{4.13}$$

where $m > 0$ is a hyper-parameter, and $A, B$ are learnable matrices. The MON parameterization can be understood as a special case of LBEN with a fixing $\Psi = I$. In this section, we will illustrate the extra expressivity of LBEN compared with MON.

**Example 4.1**

Consider a toy example with $W \in \mathbb{R}^{2\times 2}$ and take a slice near $W = 0$ of the form

$$W = \begin{bmatrix} 0 & W_{12} \\ 0 & W_{22} \end{bmatrix}, \tag{4.14}$$

for which we have:

$$2I - W - W^T = \begin{bmatrix} 2 & -W_{12} \\ -W_{12} & 2 - 2W_{22} \end{bmatrix}. \tag{4.15}$$

By Sylvester's criterion, this matrix is positive-definite if and only if $W_{22} < 1$ and $\det(2I - W - W^T) = 4(1 - W_{22}) - W_{12}^2 > 0$, which defines a parabolic region in the $W_{12}, W_{22}$ plane.

Applying our condition (4.4), without loss of generality take $\Lambda = \mathrm{diag}(1, \alpha)$ with $\alpha > 0$ and we have

$$2\Lambda - \Lambda W - W^T\Lambda = \begin{bmatrix} 2 & -W_{12} \\ -W_{12} & 2\alpha - 2\alpha W_{22} \end{bmatrix}.$$

The positivity test is now $W_{22} < 1$ and $4\alpha(1 - W_{22}) - W_{12}^2 > 0$. For each $W_{12}$ there is sufficiently large $\alpha$ such that the second condition is satisfied, since the first implies $1 - W_{22} > 0$. Hence the only constraint on $W$ is that $W_{22} < 1$, which yields a much larger region in the $W_{12}, W_{22}$ plane (see Figure 4.1). Interestingly, in this simple example with ReLU activation, the condition $W_{22} < 1$ is also a necessary condition for well-posedness [62, Theorem 2.8].

Now we consider general $L$-layer feedforward networks, described by the following recursive equation

$$z^0 = x,\ z^{\ell+1} = \sigma(W^\ell z^\ell + b^\ell),\ y = W^L z^L + b^L \tag{4.16}$$

with $\ell = 0, \ldots, L-1$. It can be rewritten as an equilibrium network (4.1) with hidden units $z = \mathrm{col}(z^1, \ldots, z^L)$ and weights $W_o = \begin{bmatrix} 0 & \cdots & 0 & W^L \end{bmatrix}$,

$$W = \begin{bmatrix} 0 & & & \\ W^1 & \ddots & & \\ \vdots & \ddots & 0 & \\ 0 & \cdots & W^{L-1} & 0 \end{bmatrix},\ U = \begin{bmatrix} W^0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{4.17}$$

The above equilibrium network is clearly well-posed as a unique solution always exists. The following propositions show that (4.17) is an LBEN but does not necessarily belong to the MON model set.

**Proposition 4.4.** *The LBEN parameterization (4.8) contains all feedforward net-*

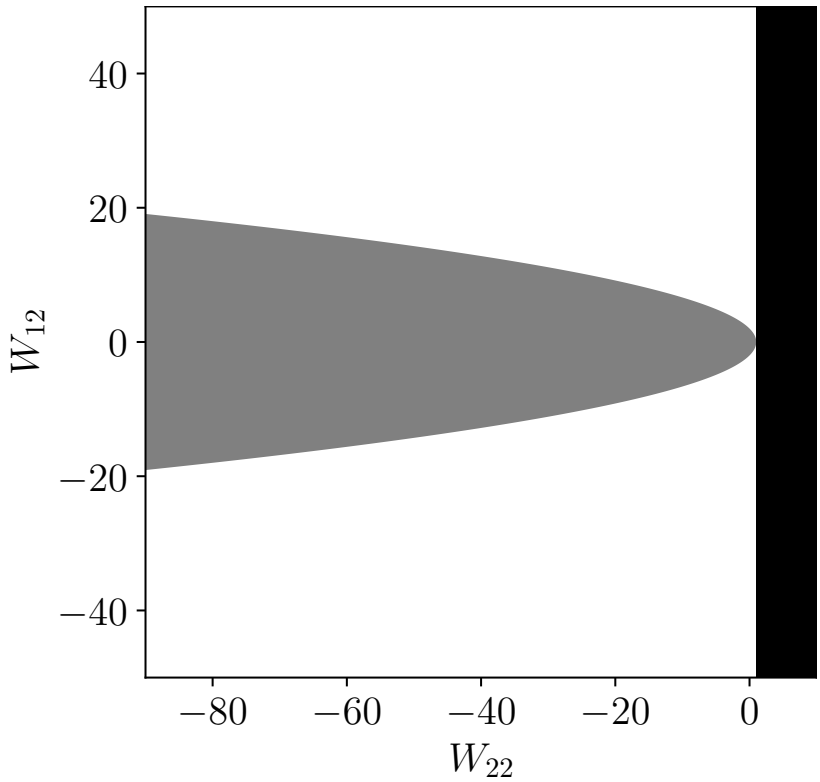



**Figure 4.1** – Valid coefficient ranges for Example 4.1.

*Gray region:* the condition from [238] is feasible: $2I - W - W^T \succ 0$.

*White region (including gray region):* our well-posedness condition is feasible: $\exists \Lambda \in \mathbb{D}^+ : 2\Lambda - \Lambda W - W^T \Lambda \succ 0$.

*Black region:* neither condition feasible.

*works (4.16).*

*Proof.* See Section 4.7 □

**Proposition 4.5.** *The MON parameterization (4.13) does not contain all feedforward networks (4.16), and if $m \geq 1$ it does not contain any feedforward networks.*

*Proof.* See Section 4.7 □

From the proof in Section 4.7.1, the set of feedforward networks in MON shrinks as the hyper-parameter $m$ increases. Most experiments in [238] use $m = 1$, which excludes all feedforward networks.

Another interesting question is: can we further improve the model expressivity by considering more general $\Lambda$, i.e., relaxing the requirement $\Lambda \in \mathbb{D}^+$ for (4.7)? In [66] it was claimed that (4.7) holds with a richer (more powerful) class of multipliers $\Lambda$ which were previously introduced for robust stability analysis of systems with repeated nonlinearities [42, 51, 116]. However this is not true: a counterexample was given in [164], and here we provide a brief explanation: even if the nonlinearities $\sigma(v_i)$ are repeated when considered as functions of $v_i$, their increments $\Delta_{zi} = \sigma(v_i + \Delta_{vi}) - \sigma(v_i)$ are not repeated when considered as functions of $\Delta_{vi}$, since they depend on the particular $v_i$ which generally differs between units.

## 4.5 Experiments

In this section, we test our approach on the MNIST and CIFAR-10 image classification problems. Our numerical experiments focus on model robustness, the trade-off between model performance and the Lipschitz constant, and the tightness of the Lipschitz bound. We compare the proposed LBEN to unconstrained equilibrium networks, the monotone operator equilibrium network (MON) of [238], and fully connected networks trained using Lipschitz margin training (LMT) [216]. When studying model robustness to adversarial attacks, we use the L2 Fast Gradient Method, implemented

as part of the Foolbox toolbox [174]. All models are trained on either a standard desktop computer with an NVIDIA GeForce RTX 2080 graphics card or using a google cloud instance with a Nvidia Tesla V100 graphics card. Details of the models and training procedure can be found in Section 4.8.

### 4.5.1 MNIST Experiments with Fully-Connected Networks

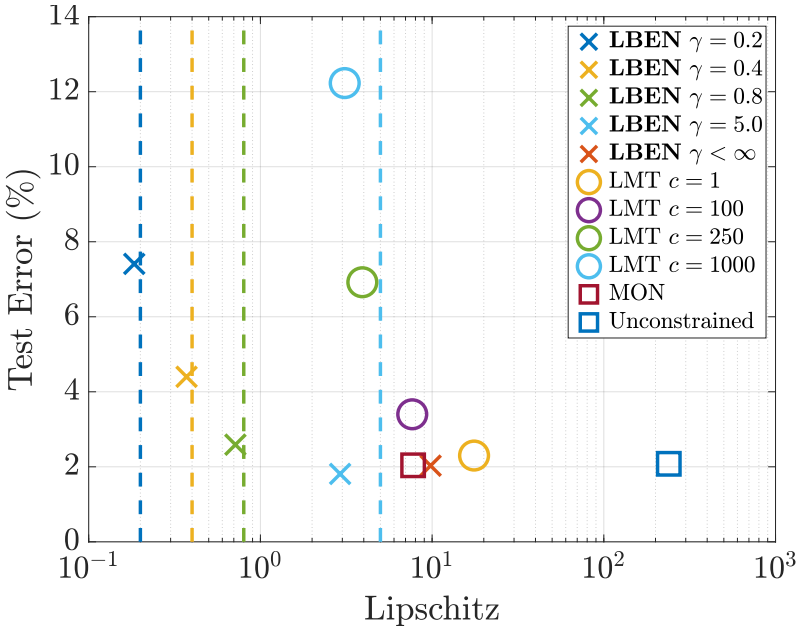
In Figure 4.2a we plot the test error versus the observed Lipschitz constant, computed via an adversarial attack for each of the models trained. We can see clearly that the parameter $\gamma$ in LBEN offers a trade-off between test error and Lipschitz constant. Comparing the $\text{LBEN}_{\gamma=5}$ with both MON and $\text{LBEN}_{\gamma<\infty}$, we also note a slight regularizing effect in the lower test error.

By comparison, LMT [216] with $c$ as a tunable regularization parameter displays a qualitatively similar trade-off, but underperforms LBEN in terms of both test error and robustness. If we examine the unconstrained equilibrium model, we observe a Lipschitz constant more than an order of magnitude higher, i.e., this model has regions of extremely high sensitivity, without gaining any accuracy in terms of test error.

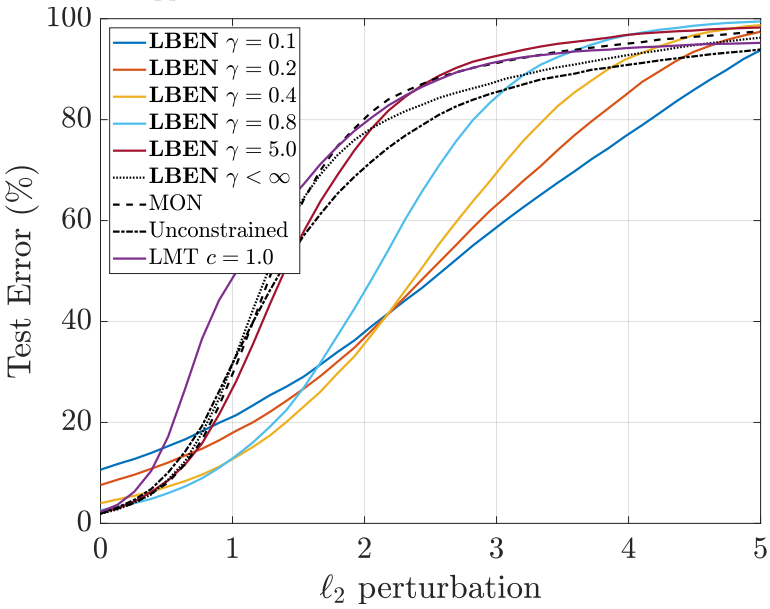
For the LBEN models, the lower and upper bounds on the Lipschitz constant are very close: the markers are very close to their corresponding lines in Section 4.2a, see also the table of numerical results in Section 4.6 in which the approximation accuracy is in many cases around 90%.

Next, we tested the robustness of classification accuracy to adversarial attacks of various sizes, the results are shown in Figure 4.2b and summarized in Table 4.1. We can clearly see that decreasing $\gamma$ (i.e. stronger regularization) in the LBENs results in a far more gradual degradation of performance as the perturbation size increases, with only a mild impact on the nominal (zero perturbation) test error.

Next, we examined the impact of our parameterization on computational complexity compared to other equilibrium models. The test and training errors versus the number




**(a)** Nominal test error vs Lipschitz constant estimates: markers indicate observed lower bounds for all methods, vertical lines indicate certified upper bounds for LBEN




**(b)** Test error with adversarial perturbation versus size of adversarial perturbation. Lower is better.

**Figure 4.2** – Image classification results on MNIST character recognition data set.

of epochs are plotted in Figure 4.5, and we can see that all models converge similarly, and take roughly the same amount of time per epoch. This is a clear contrast to the results of [164] in which imposing Lipschitz constraints resulted in fifty-fold increase in training time. Interestingly, we can also see in Figure 4.5 the effect of regularization for LBEN with $\gamma = 5$: higher training error but lower test error. We have observed several cases where the unconstrained equilibrium model became unstable during training, LBEN never exhibits this problem.

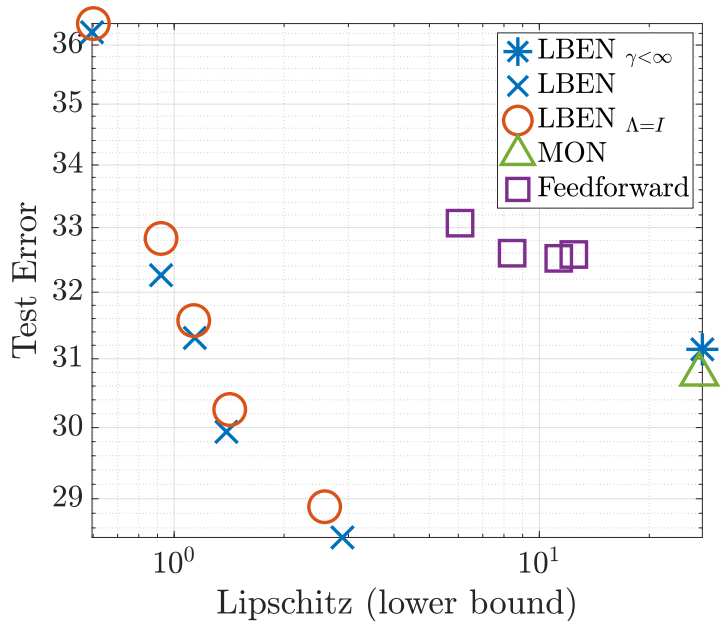
Finally, we examined the quality of the Lipschitz bounds as a function of network size, comparing the upper and lower bounds on fully connected networks with width 20 to 1000. The results are shown in Figure 4.6. It can be observed that the network size only has a mild effect on the quality of the Lipschitz bounds, which decrease slightly as width is increased by a factor of 50.

### 4.5.2 CIFAR-10 Experiments With Convolutional Networks

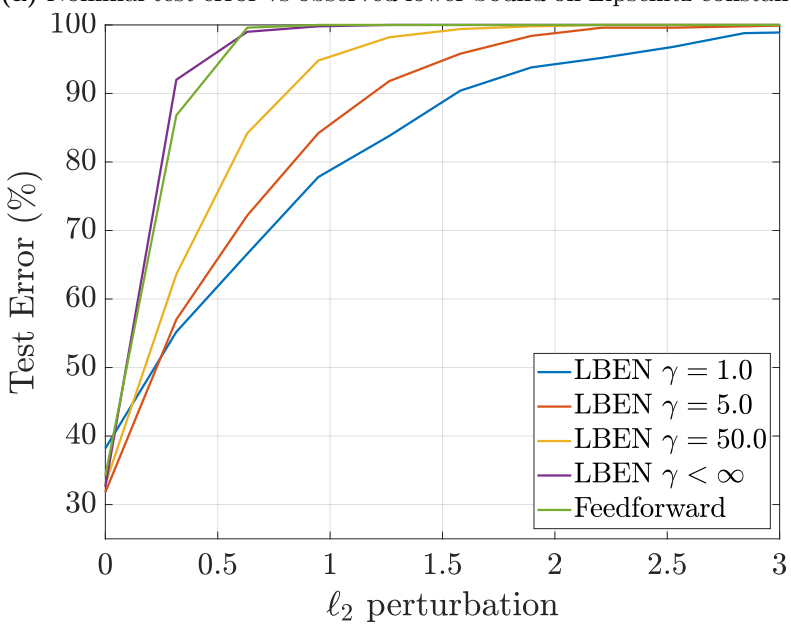
In this section, we report several experiments exploring the use of LBEN with convolutional layers on the CIFAR-10 dataset. We compare to feedforward convolutional networks of similar sizes and to LBENs with their metric set to the identity, denoted LBEN $_{\Lambda=I}$ to observe the effect of a flexible metric. Note that the model set LBEN $_{\Lambda=I,\gamma<\infty}$ corresponds to the MON. Additional model details can be found in Section 4.8.2.

In Figure 4.3a, we have plotted the test performance versus the observed Lipschitz constant for the LBEN and LBEN $_{\Lambda=I}$ for varying Lipschitz bound $\gamma = 1, 2, 3, 5, 50$, along with the LBEN$_{\gamma<\infty}$, MON, and feed-forward convolutional networks with 40, 81, 160, and 200 channels. As with the fully connected network on MNIST, we see that the Lipschitz bound has a regularizing effect, trading off between nominal fit and robustness. Additionally, we see that the LBEN provides both better performance and robustness than the traditional feedforward convolutional networks of similar sizes, highlighting the benefit of the equilibrium network structure.

Comparing LBEN and LBEN$_{\Lambda=I}$, we can see that the metric gives higher quality




**(a)** Nominal test error vs observed lower bound on Lipschitz constant.




**(b)** Test error with adversarial perturbation versus size of adversarial perturbation. Lower is better.

**Figure 4.3** – Image classification results on CIFAR-10 data set.

models for LBEN with specified $\gamma$, but it is slightly worse for LBEN $\gamma < \infty$ compared to MON. This is likely due to the extra expressiveness of the model leading to some overfitting. This can also be seen in the training curves in Figure 4.7 in the Section 4.6.

Note that the accuracy of the MON shown in Figure 4.3a is lower than the reported accuracy of 74.0% in Winston and Kolter [238]. This discrepancy is due to the reduced number of epochs that we used to train the models, using 25 epochs instead of 40. The number of training epochs was reduced as we found that applying the forward-backward algorithm caused the number of iterations required to solve for the equilibria of the LBEN increased over time and would significantly slow down training after 25 epochs.

Figure 4.3b shows the test error versus the size of adversarial perturbation for the LBEN and 162 channel feedforward convolutional network. We observe that the LBEN provides a much more gradual loss in performance than the feedforward network, with $\gamma = 5$ offering an excellent mix of nominal performance and robustness. The feed-forward networks of different sizes exhibited similar results, however only one is plotted in Figure 4.3b for clarity.

## 4.6 Experimental Results on MNIST & CIFAR Image Classification

This section contains tables of results on MNIST and CIFAR-10 data sets.

Legend:

- Err: Test error (%),
- $\|a\|_2$: $\ell^2$ norm of adversarial attack.
- $\gamma_{up}$: certified upper bound on Lipschitz constant (for models that provide one).
- $\gamma_{low}$: observed lower bound on Lipschitz constant via adversarial attack.

| Model | Err: $\|a\|_2 = 0$ | Err: $\|a\|_2 \leq 2$ | Err: $\|a\|_2 \leq 4$ | $\gamma_{up}$ | $\gamma_{low}$ | $\gamma$ approx |
|---|---|---|---|---|---|---|
| $\text{LBEN}_{\gamma<\infty}$ | **1.89** | 77.4 | 92.8 | - | 9.8 | - |
| $\text{LBEN}_{\gamma=5}$ | 2.21 | 76.4 | 96.8 | 5 | 2.912 | 58.2% |
| $\text{LBEN}_{\gamma=0.8}$ | 2.59 | 45.9 | 96.8 | 0.8 | 0.715 | 89.4% |
| $\text{LBEN}_{\gamma=0.4}$ | 3.99 | 35.6 | 92.2 | 0.4 | 0.372 | 93% |
| $\text{LBEN}_{\gamma=0.2}$ | 7.59 | **36.8** | 85.4 | 0.2 | 0.184 | 92% |
| $\text{LBEN}_{\gamma=0.1}$ | 10.36 | 37.9 | **77.1** | 0.1 | 0.0996 | 99.6% |
| MON | 1.95 | 80.2 | 95.1 | - | 7.75 | - |
| UNC | 2.06 | 70.5 | 90.9 | - | 239.0 | - |
| $\text{LMT}_{c=1}$ | 2.3 | 79.3 | 94.2 | - | 17.5 | - |
| $\text{LMT}_{c=100}$ | 3.4 | 86.1 | 95.4 | - | 7.66 | - |
| $\text{LMT}_{c=250}$ | 6.92 | 85.9 | 99.5 | - | 6.92 | - |
| $\text{LMT}_{c=1000}$ | 12.23 | 92.6 | 99.9 | - | 3.10 | - |
| Lip-NN | 3.55 | - | - | 8.74 | - | - |

**Table 4.1** – Results from MNIST experiments.

- $\gamma$ approx: approximation ratio of Lipschitz constant as percentage $= 100 \times \left(\frac{\gamma_{low}}{\gamma_{up}}\right)$.

Models:

- LBEN: the proposed Lipschitz bounded equilibrium network..
- MON: the monotone operator equilibrium network of [238].
- UNC: an unconstrained equilibrium network, i.e. $W$ directly parameterized.
- LMT: Lipschitz Margin Training model as in [216].
- Lip-NN: The Lipschitz Neural Network model of [164]. Note these figures are as reported in [164].

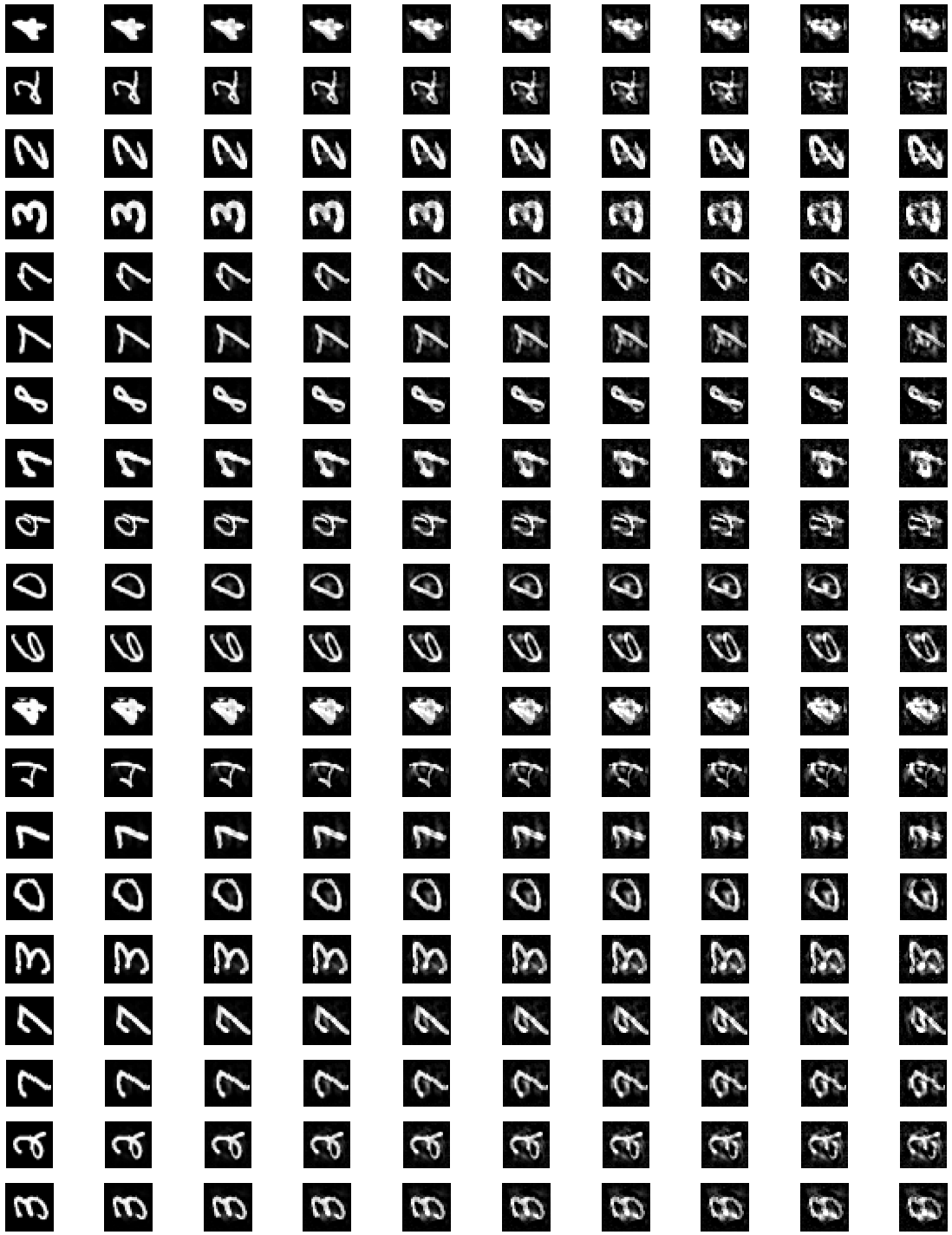


**Figure 4.4** – Random selection of MNIST adversarial examples from Figure 4.2b. Top to bottom is increasing perturbation size. Left to right are different examples.

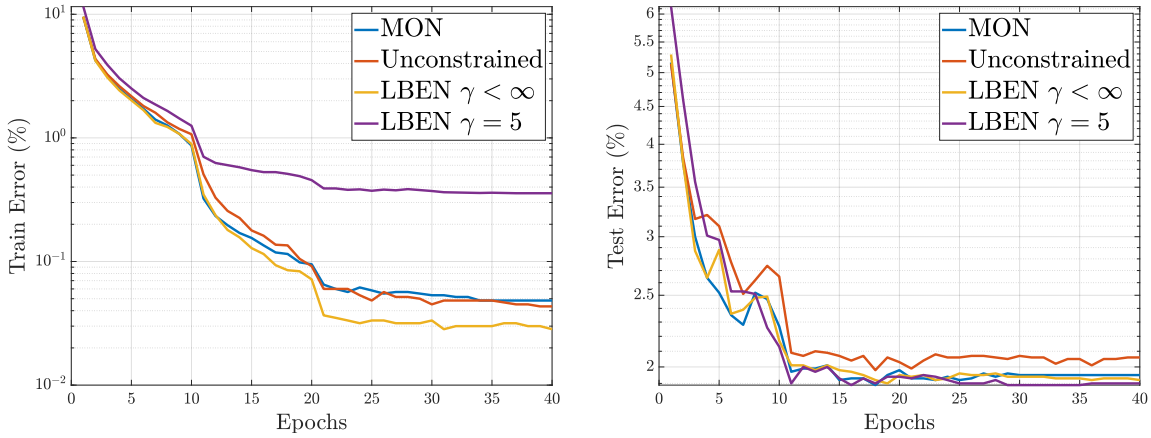



**Figure 4.5** – **Left:** Training set error versus epochs. **Right:** Test set error versus epochs. Note that the left and right plots are on different scales. The time per epoch for the MON, unconstrained, $\text{LBEN}_{\gamma<\infty}$ and $\text{LBEN}_{\gamma=5}$ networks are 14.4, 16.1, 14.9 and 14.8 seconds per epoch respectively.

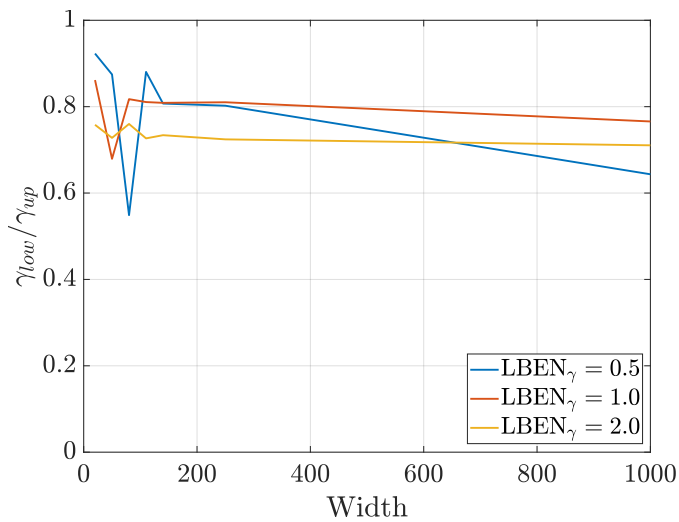



**Figure 4.6** – Approximation accuracy of the Lipschitz bound versus the network width of LBEN from the MNIST example. The certified upper bound is $\gamma_{up}$ and the observed lower bound is $\gamma_{low}$.

| Model | Err: $\|a\|_2 = 0$ | Err: $\|a\|_2 \leq 0.5$ | Err: $\|a\|_2 \leq 1.0$ | $\gamma_{up}$ | $\gamma_{low}$ | $\gamma$ approx |
|---|---|---|---|---|---|---|
| $\text{LBEN}_{\gamma<\infty}$ | 31.1 | 96.1 | 100 | - | 31.1 | - |
| $\text{LBEN}_{\gamma=50}$ | **28.4** | 75.5 | 95.4 | 50 | 2.89 | 5.7% |
| $\text{LBEN}_{\gamma=5}$ | 29.9 | 65.8 | 85.5 | 5 | 1.39 | 27.8% |
| $\text{LBEN}_{\gamma=3}$ | 31.3 | 64.2 | 83.5 | 3 | 1.14 | 38.0% |
| $\text{LBEN}_{\gamma=2}$ | 37.9 | 62.5 | 80.5 | 2 | 0.92 | 46.0% |
| $\text{LBEN}_{\gamma=1}$ | 36.2 | **61.8** | **78.8** | 1 | 0.60 | 60.0 % |
| $\text{FF}_{W=40}$ | 33.07 | 91.5 | 99.8 | - | 6.06 | - % |
| $\text{FF}_{W=81}$ | 32.6 | 93.3 | 100 | - | 8.42 | - % |
| $\text{FF}_{W=162}$ | 32.5 | 95.0 | 100 | - | 11.3 | - % |
| $\text{FF}_{W=200}$ | 32.6 | 94.5 | 100 | - | 12.4 | - % |

**Table 4.2** – Results from CIFAR experiments. FF refers to the feed-forward convolutional network.

## 4.7 Proofs

### Proof of Theorem 4.1

We present two proofs for the well-posedness of equilibrium network (4.1). All these proofs are based on the following lemma.

**Lemma 4.1** ([198])**.** For a time-invariant contracting dynamical system, all its solutions converge to a unique equilibrium.

(*Monotone operator perspective*): This proof is mainly based on Proposition 4.1, which states that the solution of (4.1) is also a zero of the operator splitting problem $0 \in (A+B)(z)$, where the operators $A$ and $B$ are given in (4.10). Condition 4.1 implies that the operator $A$ is strongly monotone while Assumption 1 implies that the operator $B$ is maximal monotone. Furthermore, the Clay operator $C_A$ is contractive and $C_B$ is non-expansive. Thus, applying Peaceman-Rachford algorithm to $0 \in (A+B)(z)$ yields a contracting discrete-time system (2.61) since $C_A C_B$ is a contractive operator. Since (2.61) is time-invariant, it yields a unique solution $z$ for any $x$ and $b_z$.

(*Neural ODE perspective*): This proof is built on Proposition 4.3, which states that the neural ODE (4.12) is a contracting continuous-time dynamical system under the

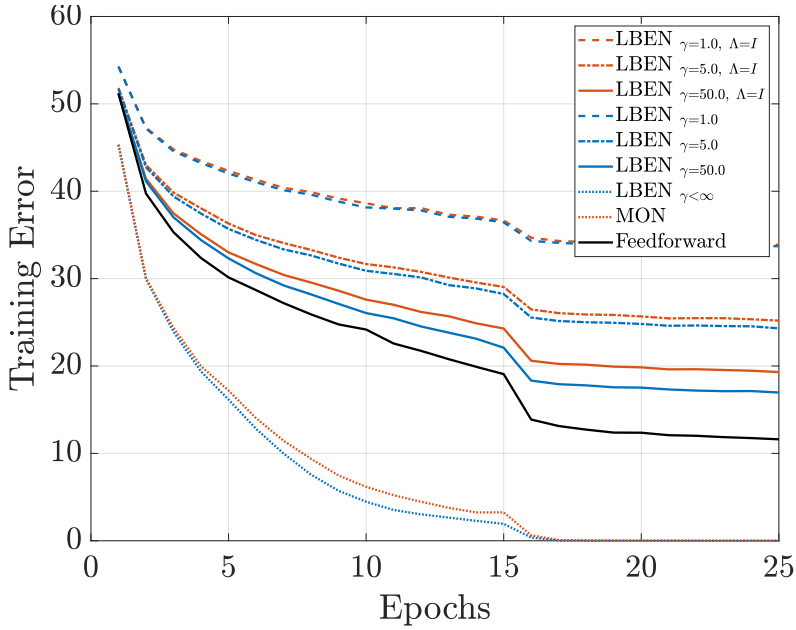



**Figure 4.7** – LBEN and MON training error versus epochs on CIFAR-10 dataset. The red curves have the metric set so that $\Lambda = I$ whereas the blue curves optimize over the metric. The line styles correspond to different gain bounds. Note that both MON and LBEN$_{\gamma<\infty}$ achieve zero training error.

Assumption 1 and Condition 4.1. For any fixed input $x$ and $b_z$, system (4.12) is also time-invariant and hence its solution converges to a unique equilibrium, which is also the solution of (4.1).

We now prove the Lipschitz boundedness of a well-posed equilibrium network. Condition 4.1 implies that there exists a constant $\epsilon > 0$ such that

$$2\Lambda - \Lambda W - W^T\Lambda \succeq \epsilon I.$$

For any $\delta \in (0, \epsilon)$ and weights $W_o, U$, we can find a sufficiently large but finite $\gamma$ such that

$$\frac{1}{\gamma}(W_o^T W_o + \Lambda U U^\top \Lambda) \preceq (\epsilon - \delta) I.$$

Then, Condition 4.2 holds for $\Lambda$ and $\gamma$ since

$$2\Lambda - \Lambda W - W^T\Lambda - \frac{1}{\gamma}(W_o^T W_o + \Lambda U U^\top \Lambda) \succeq \delta I \succ 0.$$

From Theorem 4.2, $\gamma$ is a Lipschitz bound for the well-posed equilibrium network (4.1).

## Proof of Theorem 4.2

Rearranging Eq. (4.5) yields

$$2\Lambda - \Lambda W - W^T\Lambda \succ \frac{1}{\gamma}(W_o^T W_o + \Lambda U U^T \Lambda) \succeq 0.$$

The well-posedness of the equilibrium network (4.1) follows by Theorem 4.1. To obtain the Lipschitz bound, we first apply Schur complement to (4.5):

$$\begin{bmatrix} 2\Lambda - \Lambda W - W^\top\Lambda - \frac{1}{\gamma}W_o^\top W_o & -\Lambda U \\ -U^\top\Lambda & \gamma I \end{bmatrix} \succ 0.$$

Left-multiplying $\begin{bmatrix}\Delta_z^\top & \Delta_x^\top\end{bmatrix}$ and right-multiplying $\begin{bmatrix}\Delta_z^\top & \Delta_x^\top\end{bmatrix}^\top$ gives

$$2\Delta_z^\top \Lambda \Delta_z - 2\Delta_z^\top \Lambda W \Delta_z - \frac{1}{\gamma}\Delta_z^\top W_o^\top W_o \Delta_z - 2\Delta_z^\top \Lambda U \Delta_x + \gamma\|\Delta_x\|_2^2 \geq 0.$$

Since (4.6) implies $\Delta_v = W\Delta_z + U\Delta_x$ and $\Delta_y = W_o\Delta_z$, the above inequality is equivalent to

$$\gamma\|\Delta_x\|_2^2 - \frac{1}{\gamma}\|\Delta_y\|_2^2 \geq 2\Delta_z^\top \Lambda \Delta_z - 2\Delta_z \Lambda \Delta_v = 2\langle \Delta_v - \Delta_z, \Delta_z\rangle_\Lambda. \tag{4.18}$$

Then, the Lipschitz bound of $\gamma$ for the equilibrium network (4.1) follows by (4.7).

## Proof of Proposition 4.1

Similar to [238], we first show that the solution of (4.1), if it exists, is an fixed point of the forward-backward iteration (2.60) with $\alpha = 1$:

$$z^{k+1} = R_B(z^k - \alpha A z^k) = \mathbf{prox}_{\mathfrak{f}}^1(z^k - \alpha(I-W)z^k + \alpha(Ux + b_z)) = \sigma(Wz^k + Ux + b_z).$$

The last equality follows by

$$\sigma(x) = \begin{bmatrix} \arg\min_{z_1} \frac{1}{2}(z_1 - x_1)^2 + f(z_1) \\ \vdots \\ \arg\min_{z_n} \frac{1}{2}(z_n - x_n)^2 + f(z_n) \end{bmatrix} = \arg\min_z \frac{1}{2}\|z - x\|_\Lambda^2 + \sum_{i=1}^n \lambda_i f(z_i) = \mathbf{prox}_{\mathfrak{f}}^1(x).$$

Note that the necessary condition for $\sigma(\cdot)$ to be diagonal is that the weight $\Lambda$ is positive diagonal.

Now we prove the well-posedness of LBEN by showing that the operator splitting problem $0 \in (A+B)(z)$ has a unique solution for any $x$ and $b_z$. Both Condition 4.1 and 4.2 implies that the operator $A$ is strongly monotone and its Cayley operator $C_A$ is contractive. Then, the Peaceman-Rachford iteration (2.61) is contracting and hence it converges to a unique fixed point.

## Proof of Proposition 4.2

The matrix $J$ is diagonal with elements in $[0, 1]$. Decompose $\Lambda = \Pi(J + \mu I)$ for some small $\mu > 0$, i.e. $\Pi = \Lambda(J + \mu I)^{-1}$, which is diagonal and positive-definite. By denoting $H = \Pi(I - W) + (I - W)^T \Pi$ we obtain the following inequality from (4.4):

$$\Pi J(I - W) + (I - W)^T J\Pi + \mu H \succeq \epsilon I,$$

which can be rearranged as

$$\Pi(I - JW) + (I - JW)^T \Pi \succeq \epsilon I + 2\Pi(I - J) - \mu H.$$

Since $2\Pi(I - J) \succeq 0$, we can choose a sufficiently small $\mu$ such that

$$\Pi(I - JW) + (I - JW)^T \Pi \succ 0,$$

which further implies that $I - JW$ is strongly monotone w.r.t. $\Pi$-weighted inner product, and is therefore invertible.

### 4.7.1 Proof of Proposition 4.3

Firstly, we note that the condition $2\Lambda - \Lambda W - W^T \Lambda \succ 0$ implies that $I - W$ is positive-stable (real part of eigenvalues positive), and hence so is $(I - \frac{1}{2}W)$, which is therefore invertible, and so so the ODE is well-posed.

Secondly, if the ODE is contracting it converges to a unique equilibrium for any input $x$, and it is clear that this equilibrium solves the equilibrium network, since $\dot{v} = 0 \Rightarrow v = W\sigma(v) + Ux + b_z$.

It remains to be shown that the ODE is contracting. We consider the feedback interconnection of a linear part $z \mapsto v$ and the activation functions $v \mapsto z$. A sufficient condition for contraction is that the linear part, denoted $G(s)$, is stable and

satisfies:

$$2\Lambda - \Lambda G(j\omega) - G(j\omega)^T\Lambda \succ 0 \quad \forall w, \quad G(0) = W, \quad G(\infty) = 0. \tag{4.19}$$

the first time ensures contraction via the IQC theorem. The second ensures the equilibrium condition solves the network. the third ensures $G$ is strictly proper and hence there is no algebraic loop in the interconnection (which would essentially require solving the LBEN). This can be considered an interpolation problem (at points 0 and $\infty$) with a frequency-domain constraint.

We will proceed by transforming to a positive-real interpolation problem, then a bounded real interpolation problem, and back again.

First, note that (4.19) can be re-written as

$$Z(s) + Z(s^*)^T \succ 0 \quad \forall s \geq 0 \tag{4.20}$$

where $Z(s) = I - LG(s)L^{-1}$ and $L$ is the diagonal matrix with $\Lambda = L^2$ (recall $\Lambda$ is diagonal and strictly positive). Then requiring $G(0) = W$ corresponds to the constraint $Z(0) = Z_0 = I - LWL^{-1}$, and $G(\infty) = 0$ corresponds to the constraint $Z(\infty) = I$.

We next convert this positive-real interpolation problem into a bounded-real interpolation problem. Recall that (4.20) is equivalent to $\|H(s)\| < 1$ for all $s \geq 0$ where

$$H(s) = (I + Z(s))^{-1}(I - Z(s)). \tag{4.21}$$

We will construct $H$, then transform back to $Z$, then finally back to $G$.

Now let

$$H_0 = (I + Z_0)^{-1}(I - Z_0)$$

and set

$$H(s) = \frac{1}{s+1}H_0$$

and note that since by construction $\|H_0\| < 1$ and $|1/(s+1)| \leq 1$ for all $s \geq 0$, we

have $\|H(s)\| < 1$ for all $s \geq 0$.

Now set

$$Z(s) = (I + H(s))^{-1}(I - H(s))$$

and note that $Z(0) = Z_0$ and $Z(\infty) = I$, and $Z(s) + Z(s^*)^T \succ 0$ for all $s \geq 0$. And finally the construction

$$G(s) = L^{-1}(I - Z(s))L$$

satisfies (4.19) and the interpolation conditions.

Now we show it can be written as $((I - \frac{1}{2}Ws + I)^{-1}W$, via straightforward (but slightly laborious) calculations. Now, we want to construct

$$G(s) = L^{-1}(I - Z(s))L = L^{-1}((I + H(s))^{-1}2H(s))L$$

Write $p = 1/(s+1)$ and $W_L = LWL^{-1}$, and we go through the above steps in detail:

$$\begin{aligned}
H_0 &= (I + Z_0)^{-1}(I - Z_0) = (2I - W_L)^{-1}W_L && (4.22)\\
H(s) &= p(2I - W_L)^{-1}W_L && (4.23)\\
I + H(s) &= (2I - W_L)^{-1}(2I - W_L) + (2I - W_L)^{-1}pW_L && (4.24)\\
&= (2I - W_L)^{-1}(2I + (p-1)W_L) && (4.25)\\
(I + H(s))^{-1} &= (2I + (p-1)W_L)^{-1}(2I - W_L) && (4.26)\\
I - Z(s) &= (I + H(s))^{-1}2H(s) && (4.27)\\
&= (2I + (p-1)W_L)^{-1}2pW_L && (4.28)\\
&= ((s(I - \tfrac{1}{2}W_L) + I)^{-1}W_L && (4.29)\\
G(s) &= L^{-1}(I - Z(s))L && (4.30)\\
&= L^{-1}((s(I - \tfrac{1}{2}W_L) + I)^{-1}W_L L && (4.31)\\
&= (s(I - \tfrac{1}{2}W) + I)^{-1}W && (4.32)
\end{aligned}$$

and the result is proved, since this transfer function corresponds to the linear part of the ODE above.

## Proof of Proposition 4.4

The following lemma shows that any feedforward network (4.16) is an LBEN since its hidden weight $W$ (as shown in (4.17)) is strictly lower triangular.

**Lemma 1.** Condition 4.1 holds for any strictly lower triangular $W$.

*Proof.* We prove it by showing that for any $\delta > 0$, there exists a $\Lambda \in \mathbb{D}^+$ such that

$$H(\Lambda_n, W_n) := \Lambda_n(I - W_n) + (I - W_n)^\top \Lambda_n \succ 2^{2-n}\delta I. \tag{4.33}$$

where $\Lambda_n, W_n$ are the upper left $n \times n$ elements of $\Lambda, W$, respectively. For $n = 1$, $\lambda_1 > \delta$ is sufficient since $W_1 = 0$. Assuming that (4.33) holds for $\Lambda_n$ and $W_n$, then we have

$$H(\Lambda_{n+1}, W_{n+1}) - 2^{1-n}\delta I = \begin{bmatrix} H(\Lambda_n, W_n) - 2^{1-n}\delta I & -\Lambda_n w_{n+1}^\top \\ -w_{n+1}\Lambda_n & 2(\lambda_{n+1} - 2^{-n}\delta) \end{bmatrix}, \tag{4.34}$$

where $\Lambda_{n+1} = \mathrm{diag}(\Lambda_n, \lambda_{n+1})$ and $W_{n+1} = \begin{bmatrix} [\, W_n \; 0 \,] & 0 \\ w_{n+1} & 0 \end{bmatrix}$. By applying Schur complement to (4.34), Inequality (4.33) holds for the case of $n + 1$ if $\lambda_{n+1} > 2^{-n}\delta + 2^{n-2}|\Lambda_n w_{n+1}|^2/\delta$. □

From the above lemma, we can construct a $V$ such that $V^\top V = 1/2[\Lambda(I - W) + (I - W)^\top \Lambda] - \epsilon I$ with $\epsilon = 2^{1-n}\delta$. By choosing $\Psi = \Lambda^{-1}$ and $S = (\Lambda W - W^\top \Lambda)/2$, the LBEN parameterization (4.8) recovers the exact $W$ in (4.17). Thus, LBEN contains all feedforward networks (4.16).

We note that "skip connections" as in a residual network can easily be added to the above structure via additional non-zero blocks in the lower-left part of the weight $W$.

### Proof of Proposition 4.5

From the MON parameterization (4.13) we have

$$H(m, W) := 2(1-m)I - W - W^\top = 2A^\top A \succeq 0$$

where $m > 0$. Let $\mathcal{W}_m$ be the set of non-zero and strictly lower triangular $W$ such that $H(m, W) \succeq 0$. Note that $\mathcal{W}_{m_1} \subset \mathcal{W}_{m_2}$ if $m_1 > m_2$. Because $H(m_1, W) \succeq 0$ implies $H(m_2, W) = H(m_1, W) + 2(m_1 - m_2)I \succ 0$.

Here we show that $\mathcal{W}_{m\to 0}$ does not contain all feedforward networks. Since $W$ is a strictly lower triangular for a feedforward network, $H(0, W)$ is a semidefinite matrix whose diagnoal elements are 2. As the norm of $W$ increases, $H(0, W)$ becomes indefinite. Taking the feedforward network (4.16) with $L = 2$ as an example, the set of $\mathcal{W}_{m\to 0}$ only contains those networks whose hidden unit weight $W_1$ satisfies $W_1 W_1^\top \preceq 4I$ since

$$H(0, W) = \begin{bmatrix} 2I & -W_1^\top \\ -W_1 & 2I \end{bmatrix} \succeq 0.$$

Now we show that $\mathcal{W}_m = \emptyset$ for all $m \geq 1$. Since the diagnoal elements of $H(m, W)$ are non-positive when $m \geq 1$, the matrix $H(m, W)$ is not semi-definite for any strictly lower triangular $W$.

## 4.8 Training Details

### 4.8.1 MNIST Example

This section contains the model structures and the details of the training procedure used for the MNIST examples. All models are trained using the ADAM optimizer [111] with an initial learning rate of $1 \times 10^3$. All models are trained for 40 Epochs, and the learning rate is reduced by a factor of 10 every 10 epochs.

The models in the MNIST example are all fully connected models with 80 hidden neurons and ReLU activations. For the equilibrium models, the forward and backward passes models are performed using the Peaceman-Rachford iteration scheme with $\epsilon = 1$ and a tolerance of $1 \times 10^{-2}$. When evaluating the models, we decrease the tolerance of the spitting method to $1 \times 10^{-4}$. We use the same $\alpha$ tuning procedure as [238]. All models were trained using the same initial point. Note that for LBEN, this requires initializing the metric $\Lambda = I$.

The feed-forward models trained using Lipschitz margin training were trained using the original author's code which can be found at https://github.com/ytsmiling/lmt.

### 4.8.2 CIFAR-10 Example

This section contains the model structures and the details of the training procedure used for the CIFAR-10 examples. All models are trained using the ADAM optimizer [111] with an initial learning rate of $1 \times 10^3$. The models were trained for 25 epochs and the learning rate was reduced by a factor of 10 after 15 epochs. Each model contains a single convolutional layer, an average pooling layer with kernel size 2, and a linear output layer.

The convolutional LBEN has 81 channels and is parametrized as discussed below. The MON similarly has 81 channels. Unless otherwise stated, the feed-forward convolutional network has 162 channels which gives it approximately the same number of parameters as the LBEN.

The MON was evaluated using the Peaceman-Rachford Iteration scheme.

#### Convolutional LBEN

Following the approach of [238], we parametrize $U$ and $V$ in (4.9) via convolutions. The skew symmetric matrix is constructed by taking the skew symmetric part of a convolution $\bar{S}$, so that $S = \frac{1}{2}(\bar{S} - \bar{S}^\top)$. Similar, to [238], we also find that using a

weight normalized parametrization improves performance. Specifically, we use the following parametrization: $V = \sqrt{\alpha}\frac{\hat{V}}{|\hat{V}|}$, $\bar{S} = \beta\frac{\hat{S}}{|\hat{S}|}$, $U = \sqrt{\eta}\frac{\hat{U}}{|\hat{U}|}$ and $W_o = \sqrt{\xi}\frac{\hat{W}_o}{|\hat{W}_o|}$.

In [238] Peaceman-Rachford is used and the operator $I - W$ can be quickly inverted using the fast Fourier transform. This situation is more complicated in our case as the term $W_{\text{out}}^\top W_{\text{out}}$ cannot be represented as a strict convolution and this is not diagonalized by the Fourier matrix,. Instead, we apply Forward-Backward Splitting algorithm shown in (2.60) which does not require a matrix inversion.

We have observed that the rate of convergence of the Forward-Backward splitting algorithm is highly dependent on the monotonicity parameter $m$. In particular, for the convolutional models, we found there was a strong trade-off between the ease of solve for the equilibrium versus the model expressibility and the accuracy of the Lipschitz bound.

# Chapter 5

# Recurrent Equilibrium Networks

In chapter 3 we studied recurrent neural networks as an interconnection between a linear system and a static elementwise nonlinearity. Due to well-posedness concerns about an algebraic loop, a particular term in the system was set to zero. In chapter 4, we studied this algebraic loop in more detail, constructed well-posedness conditions, and showed that including it significantly expands the model expressivity to contain all feedforward neural networks, residual networks, convolutional networks and monotone operator equilibrium networks.

In this chapter, we combine the methods developed in Chapter 3 and Chapter 4 to construct a new class of models that we call *recurrent equilibrium networks* (RENs). RENs are a class of nonlinear dynamical models for applications in machine learning, system identification, and control. The new model class has "built-in" guarantees of stability and robustness: all models in the class are contracting and the models can satisfy prescribed incremental integral quadratic constraints (IQC), including Lipschitz bounds and incremental passivity. RENs are otherwise very flexible: they can represent all stable linear systems, all previously known sets of contracting recurrent neural networks and echo state networks, all deep feedforward neural networks, and all stable Wiener/Hammerstein models.

An additional benefit of the RENs compared to the Robust RNNs from chapter 3 is that they permit a *direct parameterization*, i.e., they can be parameterized directly

by a vector in $\mathbb{R}^N$. This simplifies learning since generic methods for unconstrained optimization can be used. The performance and robustness of the new model set is evaluated on benchmark nonlinear system identification problems. We also present applications in data-driven nonlinear observer design and control with stability guarantees.

### Publications

Some of the content of this chapter has previously appeared in the following publications:

Max Revay, Ruigang Wang, and Ian R. Manchester. Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness. *Transaction on Automatic Control (Under Review)*, 2021.

Max Revay, Ruigang Wang, and Ian R. Manchester. Recurrent equilibrium networks: Unconstrained learning of stable and robust dynamical models. *Conference on Decision and Control*, 2021.

Note that the first of these publications is currently under review.

## 5.1 Introduction

In this chapter, we introduce a new model structure: the *recurrent equilibrium network* (REN).

1. RENs are highly *flexible* and include many standard models as special cases, including DNNs, RNNs, echo-state networks and stable linear dynamical systems.

2. RENs have *built in guarantees* of stability, robustness or other properties that are relevant to safety critical systems or physics informed learning.

3. RENs are *easy to use* as they permit a direct (unconstrained) parameterization enabling learning of large-scale models via simple first-order methods such as stochastic gradient descent.

RENs are guaranteed to be contracting [129], a strong form of nonlinear stability, and their built-in robustness guarantees take the form of incremental integral quadratic constraints (IQCs) [145]. This class of constraints includes user-definable bounds on the Lipschitz constant (incremental gain) of the network, and the IQC framework is inherently compatible with many commonly used tools for certifying system interconnection, including passivity methods in robotics [86], networked-system analysis via dissipation inequalities [10], $\mu$ analysis [256], and standard tools for analysis of nonlinear control systems [225]. The code to run all experiments is available via the following link `github.com/imanchester/REN`.

### 5.1.1 Learning and Identification of Stable Models

The problem of learning dynamical systems with stability guarantees appears frequently in system identification. When learning models with feedback, it is not uncommon for the model to be unstable even if the data-generating system is stable.

Even in the case of linear models, guaranteeing the stability of an identified model is complicated by the fact that the set of stable matrices is nonconvex, and various methods have been proposed to guarantee stability via regularization and constrained optimization [131, 226, 117, 152, 147, 218, 134].

For nonlinear models, there has also been a substantial volume of research on stability guarantees, e.g., for polynomial models [212, 214, 220, 219], Gaussian mixture models [110], and recurrent neural networks [148, 218, 112, 176, 177]. However, the problem is substantially more complex than the linear case due to the many different definitions of nonlinear stability. Indeed, even verification of stability of a given model is challenging. Contraction is a strong form of nonlinear stability [129] which is particularly well suited to problems in learning and system identification since it guarantees

stability of *all* solutions of the model, irrespective of inputs or initial conditions. This is important in learning since the purpose of a model is to simulate responses to previously unseen inputs. In particular, the works [212, 214, 220, 219, 148, 176, 177] are guaranteed to find contracting models.

### 5.1.2 Robustness Certification of Neural Networks

Beyond stability, model *robustness* can be characterized in terms of sensitivity to small perturbations in the input. It has recently been shown that recurrent neural network models can be extremely fragile [41], i.e. small changes to the input produce dramatic changes in the output.

Formally, sensitivity and robustness can be quantified via *Lipschitz bounds* on the input-output mapping defined by the model, e.g., incremental $\ell_2$ gain bounds and related properties such as incremental passivity, which have a long history in systems analysis [57]. In machine learning, Lipschitz constants are used in the proofs of generalization bounds [18], analysis of expressiveness [257] and guarantees of robustness to adversarial attacks [97, 168]. There is also ample empirical evidence to suggest that Lipschitz regularity (and model stability, where applicable) improves generalization in machine learning [80], system identification [176] and reinforcement learning [180].

Unfortunately, even calculation of the Lipschitz constant of a feedforward (static) neural network is NP-hard [230] and instead approximate bounds must be used. The tightest bound known to date is found by using quadratic constraints to construct a behavioral description of the neural network activation functions [66]. Extending this approach to network synthesis (i.e., training new neural networks with a prescribed Lipschitz bound) is complicated by the fact that model parameters and IQC multipliers are not jointly convex. In [164], Lipschitz bounded feedforward models were trained using the Alternating Direction Method of Multipliers, and in Chapter 3, a convexifying implicit parameterization and an interior point method were used to train Lipschitz bounded recurrent neural networks. Empirically, both works suggest generalization and robustness advantages to Lipschitz regularization, however, the

requirements to satisfy linear matrix inequalities at each iteration mean that these methods are limited to relatively small-scale networks.

### 5.1.3 Applications of Stable and Robust Models in Data-Driven Control and Estimation

Beyond system identification, the ability to learn flexible dynamical models with contraction, robustness and behavioral constraints has many applications in control and related fields, some of which we explore in this chapter.

The problem of nonlinear observer design (state estimation) can be posed as the search for a contracting dynamical system that can reproduce the true system trajectories [135] [243]. In this chapter, we formulate the observer design problem as a supervised learning problem over a set of contracting nonlinear systems, and apply it to a nonlinear reaction diffusion PDE.

In the optimization of linear controllers, a classical and widely-used approach is via the Youla-Kucera (or Q) parameterization, which represents all stabilizing controllers for a given system via a "free" stable system [256, 91]. This approach can be extended to nonlinear systems [73], [225] in which the "free parameter" is a stable nonlinear model. In this chapter, we show how learning over stable nonlinear models can be used to optimize nonlinear feedback policies for constrained linear control. This can be considered a data-driven approach to explicit model predictive control [5] with stability guarantees.

Beyond these settings, there are many further applications of flexible models with stability and robustness guarantees. In reinforcement learning [207], it has recently been found that the Lipschitz constant of policies has a strong effect on their robustness to adversarial attack [180]. In robotics, many approaches to control use passivity constraints to ensure stable interactions with physical environments, e.g. [69, 190]. In [105] it was shown that privacy preservation in dynamic feedback policies can be represented as an incremental $\ell^2$ gain bound, and thus the models we present here can be used for learning feedback policies with privacy guarantees.

### 5.1.4 Convex and Direct Parameterizations

In this chapter, we provide convex parameterizations of contracting and robust RENs via linear matrix inequality (LMI) constraints. Although convex, LMIs can be computationally challenging to incorporate for large-scale models. For example, a path-following interior point method, as proposed in [177] generally requires computing gradients of barrier functions, line search procedures, and a combination of "inner" and "outer" iterations as the barrier parameter changes.

A major benefit of RENs is that we can also provide *direct* i.e. unconstrained parameterizations of contracting and robust models. That is, we construct a smooth mapping from $\mathbb{R}^N$ to the model weights such that every model in the image of this mapping satisfies the desired behavioral constraints.

The approach is somewhat similar to the method of [36] for semidefinite programming, in which a positive-semidefinite matrix $X$ is represented by its factors $X = VV^\top$, so that $V$ can be treated as a "free" variable ensuring $X \succeq 0$. Despite introducing non-convexity, this approach has proven to be beneficial for large-scale semidefinite programming problems. Our parameterization differs in that the method of [36] generally requires nonlinear equality constraints to be satisfied, whereas in our method the associated optimization problems are completely unconstrained. The major benefit of this is that a wide variety of algorithms from unconstrained optimization can be directly applied, e.g. methods such as stochastic gradient descent and Adam [111] that have been developed for large-scale machine learning applications.

Another advantage of a direct parameterization is that it allows easy *random sampling* of nonlinear models with the required stability and robustness constraints. In this sense, our parameterization is also similar in spirit to that proposed in [37] for randomized design of robust linear control systems, and also has application to the generation of *echo state networks*, i.e. large-scale recurrent networks with fixed dynamics and learnable output maps (see, e.g., [34, 244] and references therein).

## 5.2 Learning Stable and Robust Models

This chapter is concerned with the learning of nonlinear dynamical models, i.e., finding a particular model within a set of candidates based on some data. The overall objective is to construct models that are *flexible* enough to make use of available data, and yet *guaranteed* to be well-behaved in some sense.

Given a dataset $\tilde{z}$, we consider the problem of learning a nonlinear state-space dynamical model of the form

$$x_{t+1} = f(x_t, u_t, \theta), \quad y_t = g(x_t, u_t, \theta) \tag{5.1}$$

that minimizes some loss or cost function depending (in part) on the data, i.e., to solve a problem of the form

$$\min_{\theta \in \Theta} \mathcal{L}(\tilde{z}, \theta). \tag{5.2}$$

In the above, $x_t \in \mathbb{R}^n, u_t \in \mathbb{R}^m, y_t \in \mathbb{R}^p, \theta \in \Theta \subseteq \mathbb{R}^N$ are the model state, input, output and parameters, respectively. Here $f : \mathbb{R}^n \times \mathbb{R}^m \times \Theta \to \mathbb{R}^n$ and $g : \mathbb{R}^n \times \mathbb{R}^m \times \Theta \to \mathbb{R}^p$ are piecewise continuously differentiable functions.

In the context of system identification, we may have $\tilde{z} = (\tilde{y}, \tilde{u})$ consisting of finite sequences of input-output measurements, and aim to minimize the simulation error: *simulation error*:

$$\mathcal{L}(\tilde{z}, \theta) = \|y - \tilde{y}\|_T^2 \tag{5.3}$$

where $y = \mathfrak{R}_a(\tilde{u})$ is the output sequence generated by the nonlinear dynamical model (5.1) with initial condition $x_0 = a$ and inputs $u_t = \tilde{u}_t$. Here the initial condition $a$ may be part of the data $\tilde{z}$, or considered a learnable parameter in $\theta$.

In this chapter, we are primarily concerned with constructing model parameterizations that have favorable stability and robustness properties, and we make the following definitions:

**Definition 5.1.** *A model parameterization (5.1) is called a* convex parameterization *if $\Theta \subseteq \mathbb{R}^N$ is a convex set. Furthermore, it is called a* direct parameterization *if*

$\Theta = \mathbb{R}^N$.

Direct parameterizations are useful for learning large-scale models since many scalable unconstrained optimization methods (e.g., stochastic gradient descent) can be applied to solve the learning problem (5.2).

**Definition 5.2.** *A model (5.1) is said to satisfy the* incremental integral quadratic constraint *(IQC) defined by* $(Q, S, R)$ *where* $0 \succeq Q \in \mathbb{R}^{p\times p}$, $S \in \mathbb{R}^{m\times p}$, *and* $R = R^\top \in \mathbb{R}^{m\times m}$, *if for all pairs of solutions with initial conditions* $a, b \in \mathbb{R}^n$ *and input sequences* $u, v \in \ell_{2e}^m$, *the output sequences* $y^a = \mathfrak{R}_a(u)$ *and* $y^b = \mathfrak{R}_b(v)$ *satisfy*

$$\sum_{t=0}^{T} \begin{bmatrix} y_t^a - y_t^b \\ u_t - v_t \end{bmatrix}^\top \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} y_t^a - y_t^b \\ u_t - v_t \end{bmatrix} \geq -d(a, b),\ \forall T \tag{5.4}$$

*for some function* $d(a, b) \geq 0$ *with* $d(a, a) = 0$.

Important special cases of incremental IQCs include:

- $Q = -\frac{1}{\gamma}I, R = \gamma I, S = 0$: the model satisfies an $\ell^2$ Lipschitz bound, a.k.a. incremental $\ell^2$-gain bound, of $\gamma$:

$$\|\mathfrak{R}_a(u) - \mathfrak{R}_a(v)\|_T \leq \gamma\|u - v\|_T,\ \forall u, v \in \ell_{2e}^m,\ T \in \mathbb{N}.$$

- $Q = 0, R = -2\nu I, S = I$ where $\nu \geq 0$: the model is incrementally input passive:

$$\sum_{t=0}^{T} (\mathfrak{R}_a(u_t) - \mathfrak{R}_a(v_t))^\top (u_t - v_t) \geq \nu\|u - v\|_T^2$$

  for all $u, v \in \ell_{2e}^m$ and $T \in \mathbb{N}$.

- $Q = -2\rho I, R = 0, S = I$ where $\rho > 0$: the model is incrementally strictly output passive:

$$\sum_{t=0}^{T} (\mathfrak{R}_a(u_t) - \mathfrak{R}_a(v_t))^\top (u_t - v_t) \geq \rho\|\mathfrak{R}_a(u) - \mathfrak{R}_a(v)\|_T^2$$

for all $u, v \in \ell_{2e}^m$ and $T \in \mathbb{N}$.

In other contexts, $Q, S, R$ may be decision variables in a separate optimization problem to validate the stability of interconnected systems.

**Remark 5.1.** Given a model class guaranteeing an incremental IQC defined by constant matrices $Q, S, R$, it is straightforward to construct models satisfying *frequency-weighted* IQCs. E.g., by constructing a model $\mathfrak{R}$ that is contracting and satisfies an $\ell^2$ Lipschitz bound, and choosing stable linear filters $W_1, W_2$, with $W_1$ having a stable inverse, the new model

$$y = \mathfrak{W}_a(u) = W_1^{-1}\mathfrak{R}_a(W_2 u)$$

is contracting and satisfies the frequency-weighted bound

$$\|W_1(\mathfrak{W}_a(u) - \mathfrak{W}_a(v))\|_T \leq \gamma \|W_2(u - v)\|_T.$$

This can be useful e.g. if a model should be sensitive over only a selected range of frequencies.

## 5.3 Recurrent Equilibrium Networks

The model structure we propose – the *recurrent equilibrium network* (REN) – is a state-space model of the form (5.1) with

$$x_{t+1} = Ax_t + B_1 w_t + B_2 u_t + b_x, \tag{5.5}$$

$$y_t = C_2 x_t + D_{21} w_t + D_{22} u_t + b_y, \tag{5.6}$$

where $w_t$ is the solution of an *equilibrium network*, a.k.a. *implicit network* [14, 238, 62] and Chapter 4:

$$w_t = \sigma(D_{11} w_t + C_1 x_t + D_{12} u_t + b_v) \tag{5.7}$$

and $\sigma$ is a scalar nonlinearity applied elementwise. We will show below how to ensure that a unique solution $w_t^*$ to (5.7) exists and can be computed efficiently.

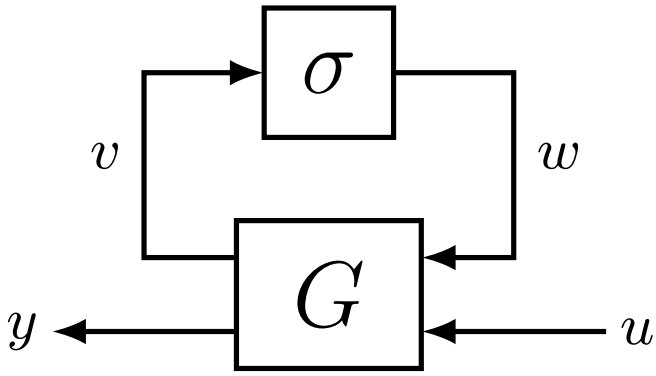



**Figure 5.1** – REN as a feedback interconnection of a linear system $G$ and a nonlinear activation $\sigma$.

The term "equilibrium" comes from the fact that any solution of the above implicit equation is also an equilibrium point of the difference equation $w_t^{k+1} = \sigma(Dw_t^k + b_w)$ or the ordinary differential equation $\frac{d}{ds}w_t(s) = -w_t(s) + \sigma(Dw_t(s) + b_w)$, where $b_w = C_1 x_t + D_{12} u_t + b_v$ is "frozen" for each time-step.

One interpretation of the REN model is that it represents a two-timescale or singular perturbation model, in which the "fast" dynamics in $w$ are assumed to reach equilibrium well within each time-step of the "slow" dynamics $x_t \to x_{t+1}$.

RENs can be conveniently represented as a feedback interconnection of a linear system $G$ and a static, memoryless nonlinear operator $\sigma$, as depicted in Fig. 5.1:

$$\begin{bmatrix} x_{t+1} \\ v_t \\ y_t \end{bmatrix} = \overbrace{\left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right]}^{W} \begin{bmatrix} x_t \\ w_t \\ u_t \end{bmatrix} + \overbrace{\begin{bmatrix} b_x \\ b_v \\ b_y \end{bmatrix}}^{b}, \tag{5.8}$$

$$w_t = \sigma(v_t) := \begin{bmatrix} \sigma(v_t^1) & \sigma(v_t^2) & \cdots & \sigma(v_t^q) \end{bmatrix}^\top, \tag{5.9}$$

where $v_t, w_t \in \mathbb{R}^q$ are the input and output of neurons respectively. The learnable parameters are the weight matrix $W \in \mathbb{R}^{(n+p+q)\times(n+m+q)}$, and the bias vector $b \in \mathbb{R}^{n+p+q}$. The nonlinear "activation function" $\sigma$ is fixed, and for simplicity we assume the same nonlinearity is applied to each channel, although this is not essential.

### 5.3.1 Flexibility of Equilibrium Networks

The equilibrium network (5.7) is quite flexible as it contains many feedforward network architectures as special cases. For example, a standard $L$-layer deep neural network takes the form

$$\begin{aligned} z_0 &= u, \\ z_{l+1} &= \sigma(W_l z_l + b_l), \quad l = 0, ..., L-1 \\ y &= W_L z_L + b_L \end{aligned} \tag{5.10}$$

where $z_l$ is the output of the $l$th hidden layer. This can be written as an equilibrium network with

$$\begin{gathered} w = \mathrm{col}(z_1, \ldots, z_L), \quad b_v = \mathrm{col}(b_0, \ldots, b_{L-1}), \quad b_y = b_L \\ D_{12} = \mathrm{col}(W_0, 0, \ldots, 0), \quad D_{21} = \begin{bmatrix} 0 & \cdots & 0 & W_L \end{bmatrix}, \\ D_{11} = \begin{bmatrix} 0 & & & \\ W_1 & \ddots & & \\ \vdots & \ddots & 0 & \\ 0 & \cdots & W_{L-1} & 0 \end{bmatrix}. \end{gathered}$$

Equilibrium networks can represent many other interesting structures including residual, convolution, and other feedforward networks. The reader is referred to [62, 238] and chapter 4 for further details.

### 5.3.2 Acyclic RENs and Well-Posedness

A useful subclass of REN is the *acyclic REN* (aREN) where the weight $D_{11}$ is constrained to be strictly lower triangular. We can interpret $D_{11}$ as the adjacency matrix of a directed graph defining interconnections between the neurons (activations functions) in the equilibrium network. If $D_{11}$ is strictly lower triangular, then this graph is guaranteed to be acylic. Compared to the general REN, the aREN is simpler to

implement since model evaluation does not require a fixed point equation to be solved and in our experience often provides similar quality of solutions.

The general REN with full $D_{11}$ may include many cycles. Well posedness of such a model is given by the Theorem 4.1, i.e., if

$$2\Lambda - \Lambda D - D^\top \Lambda \succ 0. \tag{5.11}$$

then the equilibrium network is well-posed.

### 5.3.3 Computational Details of RENs

For a well-posed REN with full $D_{11}$, solutions can be computed by formulating an equivalent monotone operator splitting problem [183].

When training an equilibrium network via gradient descent, we need to compute the Jacobian $\partial w_t^* / \partial(\cdot)$ where $w_t^*$ is the solution of the implicit equation (5.7), and $(\cdot)$ denotes the input to the network or model parameters. By using the implicit function theorem, $\partial w_t^* / \partial(\cdot)$ can be computed via

$$\frac{\partial w_t^*}{\partial(\cdot)} = (I - JD)^{-1} J \frac{\partial(Dw_t^\star + b_w)}{\partial(\cdot)} \tag{5.12}$$

where $J$ is the Clarke generalized Jacobian of $\sigma$ at $Dw_t^* + b_w$. From Assumption 5.1 in Section 5.3.4, we have that $J$ is a singleton almost everywhere. In particular, $J$ is a diagonal matrix satisfying $0 \preceq J \preceq I$. The matrix $I - JD$ is invertible by Condition (5.11), see Proposition 4.2.

### 5.3.4 Contracting and Robust RENs

We call the model of (5.8), (5.9) a contracting REN (C-REN) if it is contracting and a robust REN (R-REN) if it satisfies the incremental IQC defined by $(Q, S, R)$. Similarly, contracting a-REN (C-aREN) and robust a-REN (R-aREN) can be defined by imposing an additional structural constraint (i.e., $D_{11}$ is strictly lower-triangular).

Here we present the conditions for C-RENs and R-RENs based on incremental analysis.

For any two sequences $z^a = (x^a, w^a, v^a, u^a)$ and $z^b = (x^b, w^b, v^b, u^b)$ generated by (5.1), the dynamics of $\Delta z := z^a - z^b$ can be represented by

$$\begin{bmatrix} \Delta x_{t+1} \\ \Delta v_t \\ \Delta y_t \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta w_t \\ \Delta u_t \end{bmatrix}, \tag{5.13}$$

$$\Delta w_t = \sigma(v_t + \Delta v_t) - \sigma(v_t). \tag{5.14}$$

We make the following assumption on $\sigma$, which holds for most activation functions in the literature [78].

**Assumption 5.1.** The activation function $\sigma$ is piecewise differentiable and slope-restricted in $[0, 1]$, i.e.,

$$0 \leq \frac{\sigma(y) - \sigma(x)}{y - x} \leq 1, \quad \forall x, y \in \mathbb{R},\ x \neq y. \tag{5.15}$$

By taking a conic combination of the above constraint in the $i$th channel with multipliers $\lambda_i > 0$, we obtain the following incremental quadratic constraint

$$\Gamma_t = \begin{bmatrix} \Delta v_t \\ \Delta w_t \end{bmatrix}^\top \begin{bmatrix} 0 & \Lambda \\ \Lambda & -2\Lambda \end{bmatrix} \begin{bmatrix} \Delta v_t \\ \Delta w_t \end{bmatrix} \geq 0, \quad \forall t \in \mathbb{N} \tag{5.16}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_q) \in \mathbb{D}_+$.

The following proposition gives conditions for contracting and robust RENs using the IQC framework [145].

**Proposition 5.1.** *A REN in (5.8), (5.9) is contracting if there exist $P \succ 0$ and $\Lambda \in \mathbb{D}_+$ satisfying*

$$\begin{bmatrix} P & -C_1^\top \Lambda \\ -\Lambda C_1 & W \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix}^\top \succ 0 \tag{5.17}$$

*where* $W = 2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda$. *It satisfies the incremental IQC defined by* $(Q, S, R)$ *if there exist* $P \succ 0$ *and* $\Lambda \in \mathbb{D}_+$ *such that*

$$\begin{bmatrix} P & -C_1^\top \Lambda & C_2^\top S^\top \\ -\Lambda C_1 & W & D_{21}^\top S^\top - \Lambda D_{12} \\ SC_2 & SD_{21} - D_{12}^\top \Lambda & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \\ B_2^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \\ B_2^\top \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top \succ 0. \tag{5.18}$$

*Proof.* See Section 5.11 □

**Remark 5.2.** Note that neither of the Conditions (5.17) and (5.18) are not jointly convex in the model parameter $\theta$, stability certificate $P$, and multiplier $\Lambda$.

**Remark 5.3.** If only *non-incremental* forms of stability are considered, i.e. signal boundedness rather than contraction, then we can incorporate a richer (more powerful) class of multipliers for repeated nonlinearities, as previously discussed in [42, 51, 116]. However, these multipliers are not valid for the incremental case since the $\Delta w_t^i$ explicitly depend on the values of $v_t^i$ which may differ among channels [178].

## 5.4 Convex Parameterizations of RENs

In this section, we propose convex parameterizations for C-RENs/R-RENs, which are based on the following implicit representation of the linear component $G$:

$$\begin{bmatrix} Ex_{t+1} \\ \Lambda v_t \\ y_t \end{bmatrix} = \begin{bmatrix} F & \mathcal{B}_1 & \mathcal{B}_2 \\ \mathcal{C}_1 & \mathcal{D}_{11} & \mathcal{D}_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x_t \\ w_t \\ u_t \end{bmatrix} + \tilde{b} \tag{5.19}$$

where $E$ is an invertible matrix and $\Lambda$ is a positive-definite diagonal matrix. Note that the explicit linear model (5.8) can be easily constructed from (5.19) by inverting $E$ and $\Lambda$. While the parameters $E$ and $\Lambda$ do not expand the model set, the extra

degrees of freedom will allow us to formulate sets of C-RENs and R-RENs that are jointly convex in the model parameter, stability certificate, and multipliers.

**Definition 5.3.** *A model of the form (5.19), (5.9) is said to be well-posed if it yields a unique $(w_t, x_{t+1})$ for any $x_t, u_t$ and $b$, and hence a unique response to any initial conditions and input.*

To construct a convex parameterization of C-RENs, we introduce the following LMI constraint:

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} \succ 0, \tag{5.20}$$

where $\mathcal{W} = 2\Lambda - \mathcal{D}_{11} - \mathcal{D}_{11}^\top$. The convex parameterization of C-RENs is then given by

$$\Theta_C := \{\theta \mid \exists\, \mathcal{P} \succ 0 \text{ s.t. } (5.20)\}.$$

To construct convex parameterization of R-RENs, we propose the following convex constraint:

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & C_2^\top S^\top \\ -\mathcal{C}_1 & \mathcal{W} & D_{21}^\top S^\top - \mathcal{D}_{12} \\ SC_2 & SD_{21} - \mathcal{D}_{12}^\top & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} \\ - \begin{bmatrix} F^\top \\ \mathcal{B}_1^\top \\ \mathcal{B}_2^\top \end{bmatrix} \mathcal{P}^{-1} \begin{bmatrix} F^\top \\ \mathcal{B}_1^\top \\ \mathcal{B}_2^\top \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top \succ 0 \tag{5.21}$$

where $Q \preceq 0$, $S$, and $R$ are given. The convex parameterization of R-RENs is then defined as

$$\Theta_R := \{\theta \mid \exists \mathcal{P} \succ 0 \text{ s.t. } (5.21)\}.$$

The following result relates the above parameterizations to the desired model behavioral properties:

**Theorem 5.1.** *All models in $\Theta_C$ are well-posed and contracting. All models in $\Theta_R$ are well-posed, contracting, and satisfy the IQC defined by $(Q, S, R)$.*

*Proof.* See Section 5.11. □

**Remark 5.4.** We can modify Condition (5.17) to characterize C-RENs with a specified contraction rate $\alpha \in [0, 1)$:

$$\begin{bmatrix} E + E^\top - \frac{1}{\alpha}\mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} \succ 0.$$

One can even allow $\alpha > 1$ to characterize "slowly expanding" networks if desired.

**Remark 5.5.** It is straightforward to enforce an acyclic property on the REN or any other sparsity structure on $D_{11}$ e.g. corresponding to the standard feedforward network in Section 5.3.1 with specified layers. Since $\Lambda$ is diagonal, the sparsity structures of $\mathcal{D}_{11}$ and $D_{11} = \Lambda^{-1}\mathcal{D}_{11}$ are identical, and so the desired structure can be added as a convex constraint on $\mathcal{D}_{11}$.

The following result implies that all contracting models in our set are also Lipschitz bounded;

**Theorem 5.2.** *All models in $\Theta_C \supset \Theta_R$ have a finite $\ell_2$ Lipschitz bound.*

*Proof.* See Section 5.11 □

The proof is based showing that (5.21) is equivalent to

$$\mathcal{R} := R + SD_{22} + D_{22}^\top S^\top + D_{22}^\top Q D_{22} \succ 0, \tag{5.22a}$$

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} \succ \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top - \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top, \tag{5.22b}$$

where $\mathcal{C}_2 = (D_{22}^\top Q + S)C_2$ and $\mathcal{D}_{21} = (D_{22}^\top Q + S)D_{21} - \mathcal{D}_{12}^\top$.

## 5.5 Direct Parameterizations of RENs

In the previous section, we gave convex parameterizations of contracting and robust RENs. While convexity of a model set is useful, the conditions involve linear matrix inequalities which can be challenging to verify for large-scale models.

In this section, we provide *direct* parameterizations, i.e. smooth mappings from $\mathbb{R}^N$ to the weights and biases of contracting and robust RENs. We first present the direct parametrization for contracting RENs in Section 5.5.1, and then present the direction parametrization for robust RENs in Section 5.5.2. These Direct parametrizations allows learning to be done via unconstrained optimization, significantly enhancing the ease of use of RENs, and enables random sampling of REN models with prescribed stability and robustness conditions.

### 5.5.1 Direct Parameterizations of Contracting RENs

The main idea of our construction is to notice that the matrix in the contraction LMI (5.20) is dense and quite simple in its relation to the model parameters, so we can parameterize it directly as $X^\top X + \epsilon I$ with $X$ a free matrix variable and $\epsilon$ a small

positive constant, and from this extract the required model parameters. This idea is closely relate to the Burer-Monteiro method [36] for solving large-scale semidefinite programs.

To be precise, let

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} = X^\top X + \epsilon I \tag{5.23}$$

which is positive-definite by construction, where we have partitioned $H$ into blocks of size $n, n$, and $q$. Comparing (5.23) to (5.20) we get

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}. \tag{5.24}$$

We can calculate the implicit model parameters directly from the matrix H, giving

$$F = H_{31}, \quad \mathcal{B}_1 = H_{32}, \quad \mathcal{P} = H_{33}, \quad \mathcal{C}_1 = -H_{21}. \tag{5.25}$$

Furthermore, it can easily be verified that the construction

$$E = \frac{1}{2}(H_{11} + \mathcal{P} + Y_1 - Y_1^\top), \tag{5.26}$$

where $Y_1$ is a free matrix variable, results in $H_{11} = E + E - \mathcal{P}$.

To obtain an aREN, we need to construct a strictly lower-triangular $\mathcal{D}_{11}$ satisfying

$$H_{22} = \mathcal{W} = 2\Lambda - \mathcal{D}_{11} - \mathcal{D}_{11}^\top. \tag{5.27}$$

By taking the following matrix partition

$$H_{22} = D - L - L^\top \tag{5.28}$$

where $D$ is a diagonal matrix and $L$ is a strictly lower triangular matrix, we have

$$\Lambda = \frac{1}{2}D, \quad \mathcal{D}_{11} = L. \tag{5.29}$$

Other model parameters do not affect model stability and can be treated as free parameters.

To summarize, the parameter vector $\theta$ of a C-aREN consists of free variables: $X \in \mathbb{R}^{(2n+q)\times(2n+q)}$, $\mathcal{B}_2 \in \mathbb{R}^{n\times m}$, $C_2 \in \mathbb{R}^{p\times n}$, $\mathcal{D}_{12} \in \mathbb{R}^{q\times m}$, $D_{21} \in \mathbb{R}^{p\times q}$, $D_{22} \in \mathbb{R}^{p\times m}$ and $Y_1 \in \mathbb{R}^{n\times n}$, and model parameters are constructed via (5.25), (5.26) and (5.29).

The construction of a contracting REN with full (not acyclic) $D_{11}$ is the same except that we introduce two additional free variables: $g \in \mathbb{R}^q$ and $Y_2 \in \mathbb{R}^{q\times q}$, and then construct a positive diagonal matrix $\Lambda = e^{\mathrm{diag}(g)}$ and

$$\mathcal{D}_{11} = \Lambda - \frac{1}{2}(H_{22} + Y_2 - Y_2^\top), \tag{5.30}$$

which also results in (5.27).

### 5.5.2 Direct Parameterizations of Robust RENs

We now provide a direct parameterization of RENs satisfying the robustness condition (5.21) which is equivalent to conditions (5.22a) and (5.22b). Our direct parametrization contains two steps:

1. We first construct a direct parametrization of the matrix $D_{22}$ satisfying (5.22a).
2. We then construct a direct parametrization of the remaining model parameters satisfying (5.22b).

In many applications it is acceptable to have a model in which $D_{22}$, the direct feedthrough from input to output, is zero. E.g., for LTI systems, this corresponds to a strictly proper model. In such a case (5.22a) reduces to $R \succ 0$ and the first step below can be skipped.

### Construction of $D_{22}$ Satisfying (5.22a)

We first define $\mathcal{Q} = Q - \epsilon I \prec 0$ where $\epsilon > 0$ as a small constant, which may be fixed or parameterized, and rewrite (5.22a) as

$$R + SD_{22} + D_{22}^\top S^\top + D_{22}^\top \mathcal{Q} D_{22} \succ 0 \tag{5.31}$$

note that $\epsilon$ can be omitted if $Q \prec 0$.

Now, the direct parameterization of $D_{22}$ proceeds as follows: let $X_3, Y_3 \in \mathbb{R}^{s\times s}$ be the free variables, where $s = \max(p, m)$, and define

$$M = X_3^\top X_3 + Y_3 - Y_3^\top + \epsilon I. \tag{5.32}$$

Now set

$$Z = \left[(I - M)(I + M)^{-1}\right]_{p\times m} \tag{5.33}$$

where $[A]_{n\times m}$ denotes the upper-left $(n, m)$-block of the matrix $A$.

Now we construct the following factorizations:

$$L_Q^\top L_Q = -\mathcal{Q}, \quad L_R^\top L_R = R - S\mathcal{Q}^{-1}S^\top \tag{5.34}$$

and finally we construct $D_{22}$ as:

$$D_{22} = -\mathcal{Q}^{-1}S^\top + L_Q^{-\top} Z L_R. \tag{5.35}$$

and we have the following proposition:

**Proposition 5.2.** *The construction of $D_{22}$ om (5.32), (5.33), (5.34), (5.35) satisfies Condition (5.31).*

*Proof.* See Section 5.11. □

### Construction of Remaining Model Parameters

The construction of remaining parameters is similar to the case above for contracting RENs. Condition (5.22b) is automatically satisfied if we choose

$$\begin{aligned} H^{QSR} =& X^\top X + \epsilon I + \\ & \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top - \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top \succ 0, \end{aligned} \tag{5.36}$$

and then compute the model parameters of robust RENs based on the following matrix partition

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} = \begin{bmatrix} H_{11}^{QSR} & H_{12}^{QSR} & H_{13}^{QSR} \\ H_{21}^{QSR} & H_{22}^{QSR} & H_{23}^{QSR} \\ H_{31}^{QSR} & H_{32}^{QSR} & H_{33}^{QSR} \end{bmatrix}, \tag{5.37}$$

in a similar manner to Section 5.5.1.

### Examples of Robust RENs

We will now illustrate these ideas with some common REN parameterizations:

**Example 5.1 — Lipschitz REN**

A Lipschitz REN is described by the behavioral parameters $Q = -\frac{1}{\gamma}I$, $S = 0$ and $R = \gamma I$. In this case, we construct $L_Q = \frac{1}{\sqrt{\gamma}}I$ and $L_R = \sqrt{\gamma}I$ and construct the implicit model parameters as follows:

$$M = X_3^\top X_3 + Y_3 - Y_3^\top, \quad D_{22} = \gamma \left[ (I - M)(I + M)^{-1} \right]_{p \times m}, \tag{5.38}$$

$$\mathcal{R} = \gamma I - \frac{1}{\gamma} D_{22}^\top D_{22} \tag{5.39}$$

We now construct $H^{QSR}$ as in (5.36), giving

$$\begin{aligned} H^{QSR} =& X^\top X + \epsilon I + \\ & \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \left[\gamma I - \frac{1}{\gamma} D_{22}^\top D_{22}\right]^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top + \gamma \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top \succ 0, \end{aligned} \tag{5.40}$$

where $\mathcal{C}_2 = -\frac{1}{\gamma} D_{22}^\top C_2$ and $\mathcal{D}_{21} = -\frac{1}{\gamma} D_{22}^\top D_{21} - \mathcal{D}_{12}^\top$. The free parameters are given by $\theta = (C_2, D_{21}, D_{12}, \mathcal{B}_2, X, X_3, Y_1, Y_2, Y_3)$.

The remaining implicit model parameters are then calculated from the matrix partition (5.37).

**Example 5.2 — Output Passive REN**

An output passive REN is described by the behavioral parameters $Q = -2\rho I$, $S = I$ and $R = 0$, where $\rho > 0$. Note that for a REN to be passive we must have $p = m$. In this case, we construct $L_Q = \sqrt{2\rho} I$ and $L_R = \frac{1}{\sqrt{2\rho}} I$ and we can construct the implicit model parameters as follows:

$$M = X_3^\top X_3 + Y_3 - Y_3^\top, \tag{5.41}$$

$$D_{22} = \frac{1}{2\rho}\left[I + (I - M)(I + M)^{-1}\right] = \frac{1}{\rho}\left[(I + M)^{-1}\right], \tag{5.42}$$

$$\mathcal{R} = D_{22} + D_{22}^\top - 2\rho D_{22}^\top D_{22} \tag{5.43}$$

We now construct $H^{QSR}$ as in (5.36), giving

$$\begin{aligned} H^{QSR} = X^\top X + \epsilon I + & \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \left[D_{22} + D_{22}^\top - 2\rho D_{22}^\top D_{22}\right]^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top \\ & + 2\rho \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top \succ 0, \end{aligned} \tag{5.44}$$

where $\mathcal{C}_2 = (I - 2\rho D_{22}) C_2$ and $\mathcal{D}_{21} = (I - 2\rho D_{22}^\top) D_{21} - \mathcal{D}_{12}^\top$. The free parameters are given by $\theta = (C_2, D_{21}, D_{12}, \mathcal{B}_2, X, X_3, Y_1, Y_2, Y_3)$.

The remaining implicit model parameters are then calculated from the ma-

trix partition (5.37).

**Example 5.3 — Input Passive REN**

An input passive REN is described by the behavioral parameters $Q = 0$, $S = I$ and $R = -2\nu I$, where $\nu > 0$. Note that for a REN to be passive we must have $p = m$. In this case, (5.22a) is simply $\mathcal{R} = -2\nu I + D_{22} + D_{22}^\top$ which is satisfied by any $D_{22} \succ \nu I$. We can write this direct parameterization via

$$M = X_3^\top X_3 + Y_3 - Y_3^\top, \tag{5.45}$$

$$D_{22} = \nu I + M, \tag{5.46}$$

$$\mathcal{R} = D_{22} + D_{22}^\top - 2\nu I \tag{5.47}$$

We now construct $H^{QSR}$ as in (5.36), giving

$$H^{QSR} = X^\top X + \epsilon I + \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \left[ D_{22} + D_{22}^\top - 2\nu I \right]^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top \succ 0, \tag{5.48}$$

where $\mathcal{C}_2 = C_2$ and $\mathcal{D}_{21} = D_{21} - \mathcal{D}_{12}^\top$. The free parameters are given by $\theta = (C_2, D_{21}, D_{12}, \mathcal{B}_2, X, X_3, Y_1, Y_2, Y_3)$.

The remaining implicit model parameters are then calculated from the matrix partition (5.37).

### 5.5.3 Random Sampling of Nonlinear Systems and Echo State Networks

One benefit of the direct parameterizations of RENs is that it is straightforward to randomly sample systems with the desired behavioral properties. Since contracting and robust RENs are constructed as the image of $\mathbb{R}^N$ under a smooth mapping (Sections 5.5.1 and 5.5.2), one can sample random vectors in $\mathbb{R}^N$ and map them to random stable/robust nonlinear dynamical systems.

An "echo state network" is a state-space model with randomly sampled, but fixed,

state dynamics, and a learnable output map:

$$x_{t+1} = f(x_t, u_t) \tag{5.49}$$

$$y_{t+1} = g(x_t, u_t, \theta) \tag{5.50}$$

where $f$ is fixed and $g$ is affinely parameterized by $\theta$, i.e.

$$g(x_t, u_t, \theta) = g_0(x_t, u_t) + \sum_i \theta_i g^i(x_t, u_t).$$

Then, e.g., system identification with a simulation error criteria can be solved as a basic least squares problem. This approach is reminiscent of system identification via a basis of stable linear responses (see, e.g., [232]).

For this approach to work, it is essential that the random dynamics are stable. In [34, 244] and references therein, contraction of (5.49) is referred to as the "echo state property", and simple parameterizations are given for which contraction is guaranteed.

The direct parameterization of REN can be used to randomly sample from a rich class of contracting models, by sampling $X, Y_1, Y_2, \mathcal{B}_2, \mathcal{D}_{12}$ to construct the state-space dynamics and equilibrium network. Such a model can be used, e.g., for system identification by simulating its response to inputs to generate data $\tilde{u}_t, \tilde{x}_t, \tilde{w}_t$, and then the output mapping

$$y_t = C_2\tilde{x}_t + D_{21}\tilde{w}_t + D_{22}\tilde{u}_t + b_y$$

can be fit to $\tilde{y}_t$, minimizing (5.3) via least-squares to obtain the parameters $C_2, D_{21}, D_{22}, b_y$. We will also see in Section 5.9 how this approach can be applied in data-driven feedback control design.

## 5.6 Expressivity of REN Model Class

The set of RENs contain many prior model structures as special cases. We now discuss the relationship to some prior model types:

### Robust and Contracting RNNs and Echo State Networks

If we set $D_{11} = 0$, then the nonlinearity is not an equilibrium network but a single-hidden-layer neural network, and our model set $\Theta_C$ reduces to the model set proposed in Chapter 3. Therefore, the REN model class also includes all other models that were proven to be in that model set in Chapter 3, including:

1. all stable linear time-invariant (LTI) systems satisfying the corresponding IQC.
2. all prior sets of contracting RNNs including the ciRNN, s-RNN[148].

We note that the stability test for the ciRNN, contraction with respect to a diagonal metric, is the same as that proposed for echo state networks [34, 244], by randomly sampling RENs as in Section 5.5.3 we sample from a strictly larger set of echo state networks than previously known.

### Block Structured Models

Block structured models are constructed from series interconnections of LTI systems and static nonlinearities [188, 75]. The REN contains block structured models as a subset, where the built-in equilibrium network can approximate any continuous static nonlinearity and the linear system represents the LTI block. For simplicity, we only consider two simple block-oriented models:

1. *Wiener systems* consist of an LTI block followed by a static non-linearity. This structure is replicated in (5.8), (5.9) when $B_1 = 0$ and $C_2 = 0$. In this case the linear dynamical system evolves independently of the non-linearities and feeds into a equilibrium network.
2. *Hammerstein systems* consist of a static non-linearity connected to an LTI system. This is represented in the REN when $B_2 = 0$ and $C_1 = 0$. In this case the input passes through a static equilibrium network and into an LTI system.

Other more complex block-oriented models such as [189] can also be constructed as RENs in a similarly straightforward manner.

### Nonlinear Finite Impulse Response Models

Finite impulse response models are finite memory nonlinear filters. These have a similar construction to Wiener systems, where the LTI system contains a delay system that stores a finite history of inputs. The REN recovers a set of finite memory filters when

$$A = \begin{bmatrix} 0 & & \\ I & 0 & \\ & I & \ddots \\ & & \ddots \end{bmatrix}, \quad B_2 = \begin{bmatrix} I \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \quad B_1 = 0. \tag{5.51}$$

The output is then a nonlinear function of a truncated history of inputs.

## 5.7 Use Case: Stable and Robust Nonlinear System Identification

We demonstrate the proposed model set on the F16 ground vibration [157] and Wiener Hammerstein with process noise [187] system identification benchmark. We will compare the C-aREN and Lipschitz bounded R-aREN (i.e., robust REN with some prescribed Lipschitz bound of $\overline{\gamma}$, denoted by R-aREN $\gamma < \overline{\gamma}$) with an LSTM and RNN with a similar number of parameters. We will also compare to the Robust RNN proposed in Chapter 3.

An advantage of using a direct parameterization is that unconstrained optimization techniques can be applied. We fit models by minimizing simulation error:

$$\mathcal{L}_{se}(\tilde{z}, \theta) = ||\tilde{y} - \mathfrak{R}_a(\tilde{u})||_T^2 \tag{5.52}$$

using minibatch gradient descent with the Adam optimizer [111].

When training RENs, we use the Peaceman-Rachford monotone operator splitting algorithm [183, 238, 178] to solve for the fixed points in (4.1). We use the conjugate

gradient method to solve for the gradients with the respect to the equilibrium layer (4.1). The remaining gradients are calculated using the automatic differentiation tool, Zygote [99].

Model performance is measured using normalized root mean square error on the test sets, calculated as:

$$\text{NRMSE} = \frac{||\tilde{y} - \mathfrak{R}_a(\tilde{u})||_T}{||\tilde{y}||_T}. \tag{5.53}$$

Model robustness is measured in terms of the maximum observed sensitivity:

$$\underline{\gamma} = \max_{u,v,a} \frac{||\mathfrak{R}_a(u) - \mathfrak{R}_a(v)||_T}{||u - v||_T}. \tag{5.54}$$

We find a local solution to (5.54) using gradient ascent with the Adam optimizer. Consequently $\underline{\gamma}$ is a lower bound on the true Lipschitz constant of the sequence-to-sequence map.

### 5.7.1 Benchmark Datasets and Training Details

#### F16 System Identification Benchmark

The F16 ground vibration benchmark dataset [157] consists of accelerations measured by three accelerometers, induced in the structure of an F16 fighter jet by a wing-mounted shaker. We use the multisine excitation dataset with full frequency grid. This dataset consists of 7 multisine experiments with 73,728 samples and varying amplitude. We use datasets 1, 3, 5 and 7 for training and datasets 2, 4 and 6 for testing. All test data was standardized before model fitting.

All models fit have approximately 118,000 parameters. That is, the RNN has 340 neurons, the LSTM has 170 neurons and the RENs have width $n = 75$ and $q = 150$. Models were trained for 70 epochs with a sequence length of 1024. The learning rate was initalized at $10^{-3}$ and was reduced by a factor of 10 every 20 Epochs.

#### Wiener-Hammerstein With Process Noise Benchmark

The Wiener Hammerstein with process noise benchmark dataset [187] involves the estimation of the output voltage from two input voltage measurements for a block-oriented Wiener-Hammerstein system with large process noise that complicates model fitting. We have used the Multi-sine fade-out dataset consisting of two realizations of a multi-sine input signal with 8192 samples each. The test set consists of two experiments, a random phase multi-sine and a sine sweep, conducted without the added process noise.

All models trained have approximately 42,000 parameters. That is, the RNN has 200 neurons, the LSTM has 100 neurons, and the RENs have $n = 40$ and $q = 100$. Models were trained for 60 epochs with a sequence length of 512. The initial learning rate was $1 \times 10^{-3}$. After 40 epochs, the learning rate was reduced to $1 \times 10^{-4}$.

### 5.7.2 Results and Discussion

We have plotted the mean test performance versus the observed sensitivity of the models trained on the F16 and Wiener-Hammerstein Benchmarks in Fig. 5.2 and 5.3, respectively. The dashed vertical lines show the guaranteed upper bounds on the Lipschitz constant for the RENs. In all cases, we observe that the REN provides the best trade-off between nominal performance and robustness, with the REN slightly outperforming the LSTM in terms of nominal test error for large $\gamma$. By tuning $\gamma$, nominal test performance can be traded-off for robustness, signified by the consistent trend moving diagonally up and left with decreasing $\gamma$. In all cases, we found that the REN was significantly more robust than the RNN, typically having about 10% of the sensitivity for the F16 benchmark and 1% on the Wiener-Hammerstein benchmark. Also note that for small $\gamma$, the observed lower bound on the Lipschitz constant is very close to the guaranteed upper bound, showing that the real Lipschitz constant of the models is close to the upper bound.

Compared to the robust RNN (from Chapter 3), the REN has similar bounds on the incremental $\ell_2$ gain, however the added flexibility from the term $D_{11}$ significantly

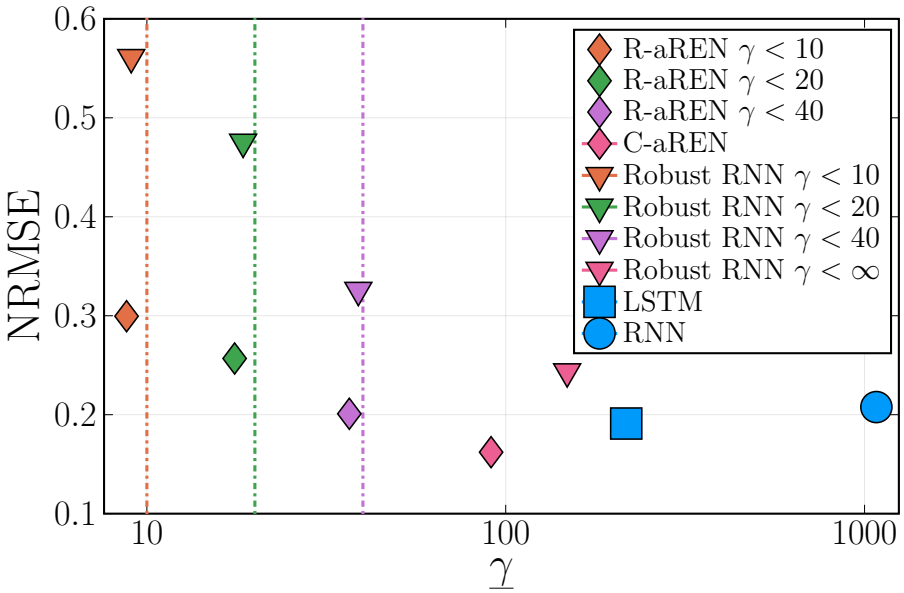



**Figure 5.2** – Nominal performance versus robustness for models trained on F16 ground vibration benchmark dataset. The dashed vertical lines are the guaranteed upper bounds on $\gamma$ corresponding to the models with matching color.

improves the nominal model performance for a given gain bound. Additionally, while both the C-aREN and Robust RNN $\gamma < \infty$ are contracting models, we note that the C-aREN is significantly more expressive with a NRMSE of 0.16 versus 0.24.

It is well known that many neural networks are very sensitive to adversarial perturbations. This is shown, for instance, in Fig. 5.4 and 5.5, where we have plotted the change in output for a small adversarial perturbation $||\Delta u|| < 0.05$, for a selection of models trained on the F16 benchmark dataset. Here, we can see that both the RNN and LSTM are very sensitive to the input perturbation. The R-aREN on the hand, has guaranteed bounds on the effect of the perturbation and is significantly more robust.

We have also trained R-RENs and C-RENs for the F16 and Wiener Hammerstein Benchmark datasets. The resulting nominal performance and sensitivities for the aRENs and RENs are shown in Table 5.1. We do not observe a significant difference in performance between the cyclic and acyclic model classes.

Finally, we have plotted the loss versus the number of epochs in Fig. 5.6 for some

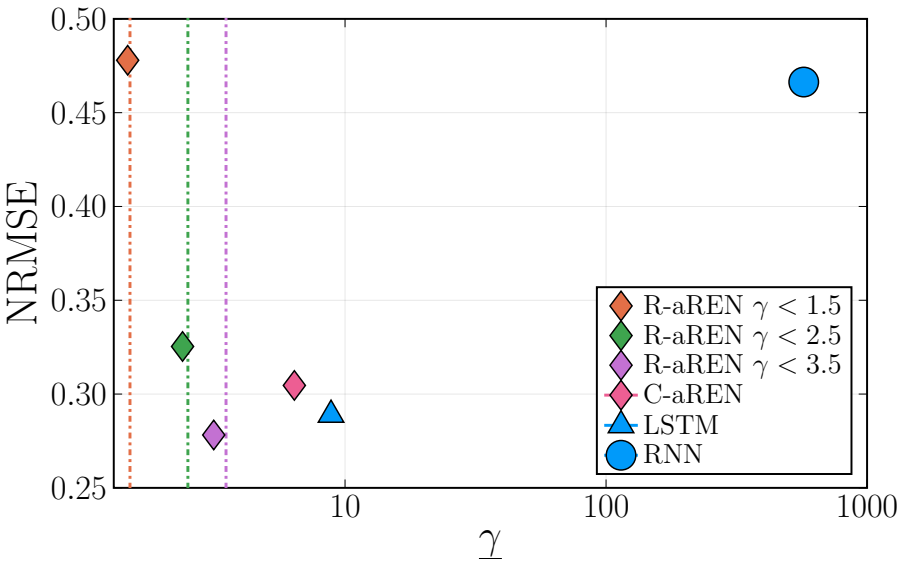



**Figure 5.3** – Nominal performance versus robustness for models trained on Wiener-Hammerstein with process noise benchmark dataset. The dashed vertical lines are the guaranteed upper bounds on $\gamma$ corresponding to the models with matching color.

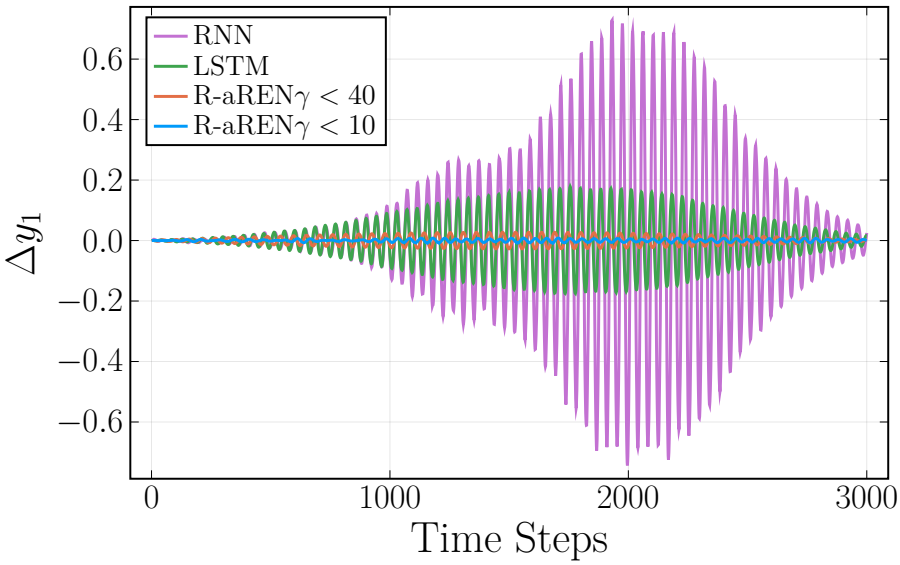



**Figure 5.4** – Change in output of models subject to an adversarial perturbation of $||\Delta u|| < 0.05$. The incremental gains from $\Delta u$ to $\Delta y$ are 980, 290, 37 and 8.6 respectively.

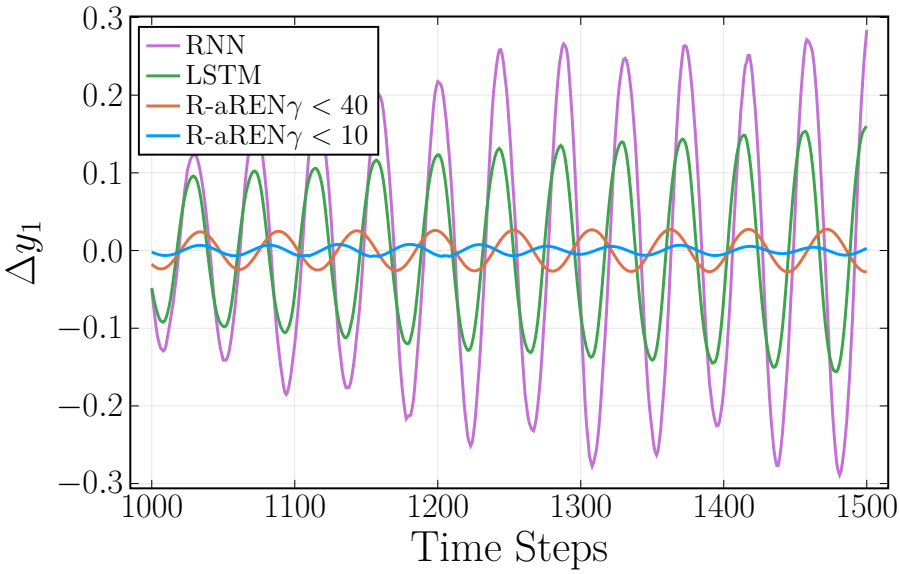



**Figure 5.5** – Zoomed in version of Fig. 5.4.

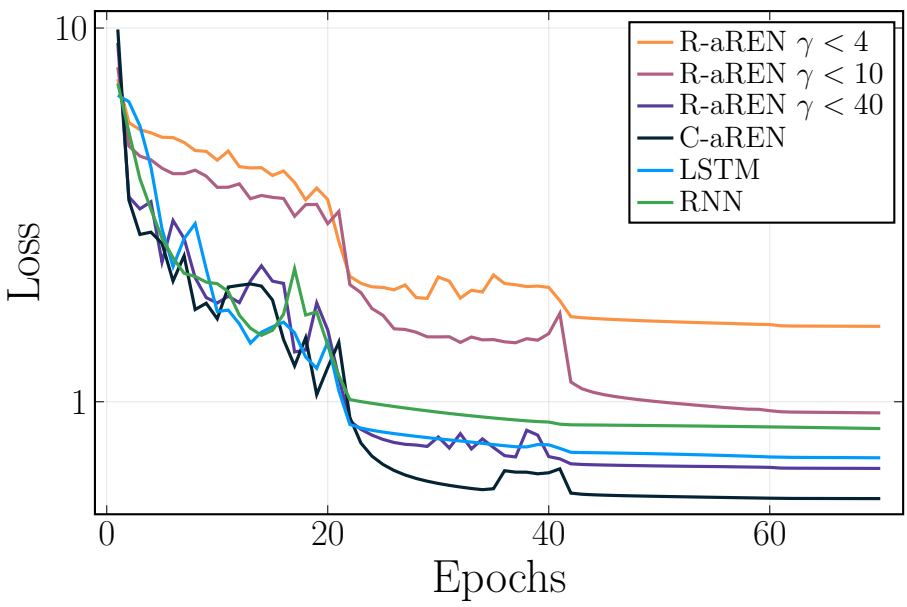



**Figure 5.6** – Nominal performance versus robustness for models trained on F16 ground vibration benchmark dataset.

**Table 5.1** – Upper and lower bounds on incremental $\ell_2$ gain and nominal performance for aREN and REN.

| | $\bar{\gamma}$ | 10 | 20 | 40 | 60 | 100 | $\infty$ |
|---|---|---|---|---|---|---|---|
| aREN | $\underline{\gamma}$ | 8.8 | 17.5 | 36.7 | 44.9 | 60.56 | 91.0 |
| | NRMSE (%) | 30.0 | 25.7 | 20.1 | 18.5 | 17.2 | 16.2 |
| REN | $\underline{\gamma}$ | 9.1 | 17.1 | 36.0 | 44.6 | 57.9 | 85.26 |
| | NRMSE (%) | 30.3 | 26.8 | 21.8 | 19.9 | 19.3 | 16.8 |

of the models on the F16 dataset. Compared to the LSTM, the REN takes a similar number of steps and achieves a slightly lower training loss. The LSTM is a model designed to be easy to train.

## 5.8 Use Case: Learning Nonlinear Observers

Estimation of system states from incomplete and/or noisy measurements is an important problem in many practical applications. For linear systems with Gaussian noise, a simple and optimal solution exists in the form of the Kalman filter, but for nonlinear systems even achieving estimation stability is non-trivial and many approaches have been investigated, e.g. [11, 109, 25]. State estimation a.k.a. observer design was one of the original motivations for contraction analysis [129], and in this section, we show how a flexible set of contracting models can be used to learn stable state observers via *snapshots* of a nonlinear system model.

Given a nonlinear system of the form

$$x_{t+1} = f_m(x_t, u_t, w_t), \quad y_t = g_m(x_t, u_t, w_t) \tag{5.55}$$

where $x_t \in \mathbb{X}$ is an internal state to be estimated, $y_t$ is an available measurement, $u_t \in \mathbb{U}$ is a known (e.g. control) input, and $w_t$ comprises unknown disturbances and sensor noise. Here $\mathbb{X}, \mathbb{U}$ are some compact sets. We consider $w_t = 0$ to represent a nominal deterministic model.

A standard structure, pioneered by Luenberger, is an observer of the form

$$\hat{x}_{t+1} = f_m(\hat{x}_t, u_t, 0) + l(\hat{x}_t, u_t, y_t) \tag{5.56}$$

i.e. a combination of a model prediction $f_m$ and a measurement correction function $l$. A common special case is $l(\hat{x}_t, u_t, y_t) = L(\hat{x})(y_t - g_m(\hat{x}_t, u_t, 0))$ for some gain $L(\hat{x})$.

In many practical cases the best available model $f_m, g_m$ is highly complex, e.g. based

on finite element methods or algorithmic mechanics [68]. This poses two major challenges to the standard paradigm:

1. How to design the function $l$ such that the observer (5.56) is stable (preferably globally) and exhibits good noise/disturbance rejection.
2. The model itself may be so complex that evaluating $f_m(\hat{x}_t, u_t, 0)$ in real-time is infeasible, e.g. for stiff systems where short sample times are required.

Our parameterization of contracting models enables an alternative paradigm, first suggested for the restricted case of polynomial models in [135].

**Proposition 5.3.** *If we construct an observer of the form*

$$\hat{x}_{t+1} = f_o(\hat{x}_t, u_t, y_t) \tag{5.57}$$

*such that the following two conditions hold:*

1. *The system (5.57) is contracting with rate $\alpha \in (0, 1)$ for some constant metric $P \succ 0$.*
2. *The following "correctness" condition holds:*

$$f_m(x, u, 0) = f_o(x, u, g_m(x, u, 0)), \ \forall (x, u) \in \mathbb{X} \times \mathbb{U}. \tag{5.58}$$

*Then when $w = 0$ we have $\hat{x}_t \to x_t$ as $t \to \infty$. Suppose that the observer (5.57) satisfies Conditions 1) and*

*3) The following error bound holds:*

$$|e(x, u)| \leq \rho, \quad \forall (x, u) \in \mathbb{X} \times \mathbb{U}, \tag{5.59}$$

*where $e(x, u) := f_o(x, u, g_m(x, u, 0)) - f_m(x, u, 0)$.*

*Then when $w = 0$ we have*

$$|\hat{x}_t - x_t| \leq \frac{2\rho}{1-\alpha}\sqrt{\frac{\overline{\sigma}(P)}{\underline{\sigma}(P)}}, \quad \forall t \geq T \tag{5.60}$$

*for some sufficiently large $T \in \mathbb{N}$, where $\overline{\sigma}(P)$ and $\underline{\sigma}(P)$ denote the maximum and minimum singular values of $P$, respectively.*

The reasoning is simple: (5.58) implies that if $\hat{x}_0 = x_0$ then $\hat{x}_t = x_t$ for all $t \geq 0$, i.e. the true state is a particular solution of the observer. But contraction implies that all solutions of the observer converge to each other. Hence all solutions of the observer converge to the true state. The proof of the estimation error bound can be found in Section 5.11.

In this section we propose designing such observers as a supervised learning problem over our class of contracting models.

1. Sample a dataset $\tilde{z} = \{x^i, u^i, i = 1, 2, ..., N\}$ where $(x^i, u^i) \in \mathbb{X} \times \mathbb{U}$.
2. Compute $g_m^i = g_m(x^i, u^i, 0)$ and $f_m^i = f_m(x^i, u^i, 0)$ for each $i$.
3. Learn a contracting system $f_o$ minimizing the loss function

$$\mathcal{L}_o(\tilde{z}, \theta) = \sum_{i=1}^{N} \left| f_m^i - f_o(x^i, u^i, g_m^i) \right|^2 \tag{5.61}$$

**Remark 5.6.** An observer of traditional form (5.56) with $l(\hat{x}_t, u_t, y_t) = L(\hat{x})(y_t - g_m(\hat{x}_t, u_t, 0))$ will always satisfy the correctness condition, but designing $L(\hat{x})$ to achieve global convergence may be difficult. In contrast, an observer design using the proposed procedure will always achieve global convergence, but may not achieve correctness exactly. If the dataset is sufficiently dense, and the loss (5.61) can be driven near zero, then we can establish the estimation error bound based on Proposition 5.3 and the generalization bound of the observer learning problem.

### 5.8.1 Example: Reaction-Diffusion PDE

We illustrate this approach by designing an observer for the following semilinear reaction-diffusion partial differential equation:

$$\frac{\partial \xi(z,t)}{\partial t} = \frac{\partial^2 \xi(z,t)}{\partial z^2} + R(\xi, z, t), \tag{5.62}$$

$$\xi(z,0) = 1, \quad \xi(1,t) = \xi(0,t) = b(t) \tag{5.63}$$

$$y = g(\xi, z, t) \tag{5.64}$$

where, the state $\xi(z,t)$ is a function of both the spatial coordinate $z \in [0,1]$ and time $t \in \mathbb{R}_+$. Models of the form (5.62) model processes such as combustion [74], bioreactors [146] or neural spiking dynamics [74]. The observer design problem for such systems has been considered using complex back-stepping methods that guarantee only local stability [146].

We consider the case where the local reaction dynamics have the following form, which appears in models of combustion processes [74]:

$$R(\xi, z, t) = \frac{1}{2}\xi(1-\xi)(\xi - \tfrac{1}{2}).$$

We consider the boundary condition $b(t)$ as a known input and assume that there is a single measurement taken from the center of the spatial domain so $y(t) = \xi(0.5, t)$.

We discretize $z$ into $N$ intervals $z^1, ..., z^N$ where $z^i = (i-1)\Delta z$. The state at spatial coordinate $z^i$ and time $t$ is then described by $\bar{\xi}_t = (\xi_t^1, \xi_t^2, ..., \xi_t^N)$ where $\xi_t^i = \xi(z^i, t)$. The dynamics over a time period $\Delta t$ can then be approximated using the following finite differences:

$$\frac{\partial \xi(z,t)}{\partial t} \approx \frac{\xi_{t+\Delta t}^i - \xi_t^i}{\Delta t}, \tag{5.65}$$

$$\frac{\partial^2 \xi(z,t)}{\partial z^2} \approx \frac{\xi_t^{i+1} + \xi_t^{i-1} - 2\xi_t^i}{\Delta z^2}. \tag{5.66}$$

Substituting (5.65) and (5.66) into (5.62) and rearranging for $\bar{\xi}_{t+\Delta t}$ leads to an $N$

dimensional state space model of the form:

$$\bar{\xi}_{t+\Delta t} = a_{rd}(\bar{\xi}_t, b_t) \tag{5.67}$$

$$y_t = c_{rd}(\bar{\xi}_t) \tag{5.68}$$

We generate training data by simulating the system (5.67), (5.68) with $N = 51$ for $10^5$ time steps with the stochastic input $b_{t+1} = b_t + 0.05\omega_t$ where $\omega_t \sim \mathcal{N}[0, 1]$. We denote this training data by $\tilde{z} = (\tilde{\xi}_t, \tilde{y}_t, \tilde{b}_t)$ for $t = 0, \ldots, 10^5\Delta t$.

To train an observer for this system, we construct a C-aREN with $n = 51$ and $q = 200$. We optimize the one step ahead prediction error:

$$\mathcal{L}(\tilde{z}, \theta) = \frac{1}{T}\sum_{t=0}^{T-1} |a(\tilde{\xi}_t, \tilde{b}_t) - f_o(\tilde{\xi}_t, \tilde{b}_t, \tilde{y}_t)|^2,$$

using stochastic gradient descent with the Adam optimizer [111]. Here, $f_o(\xi, b, y)$ is a C-aREN described by (5.8), (5.9) using direct parametrization discussed in Section 5.5.1. Note that we have taken the output mapping in (5.8) to be $[C_2, D_{21}, D_{22}] = [I, 0, 0]$.

We have plotted results of the PDE simulation and the observer state estimates in Fig. 5.7. The simulation starts with an initial state of $\xi(z, 0) = 1$ and the observer has an initial state estimate of $\bar{\xi}_0 = 0$. The error between the state estimate and the PDE simulation's state quickly decays to zero and the observer state continues to track the PDE's state.

We have also provided a comparison to a free run simulation of the PDE with initial condition $\xi(z, 0) = 0$ in Fig. 5.8 and 5.9. Here we can see that the simulated trajectories with different initial conditions do not converge. This suggests that the system is not contracting and the state cannot be estimated by simply running a parallel simulation. The state estimates of the observer, however, quickly converge on the true model state.

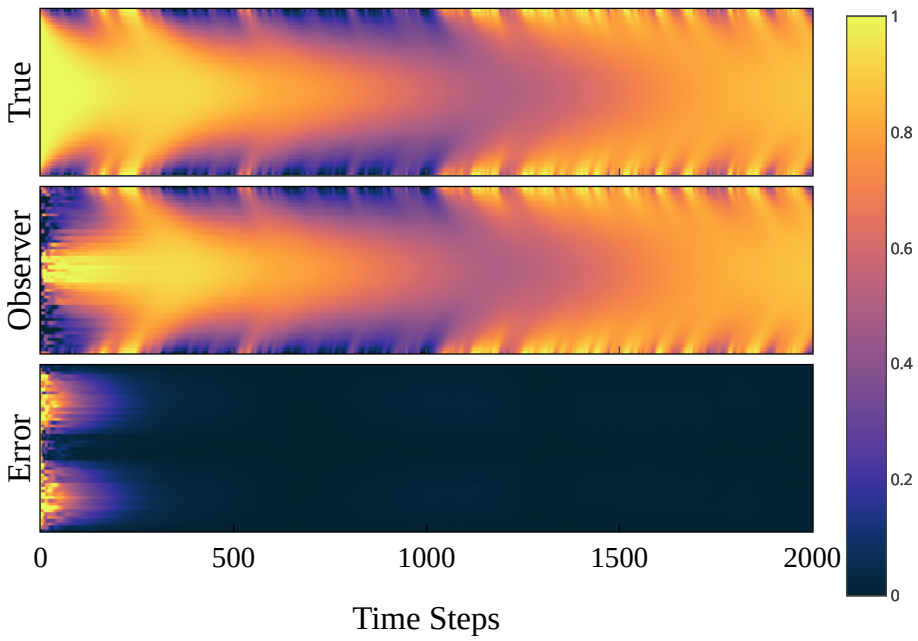
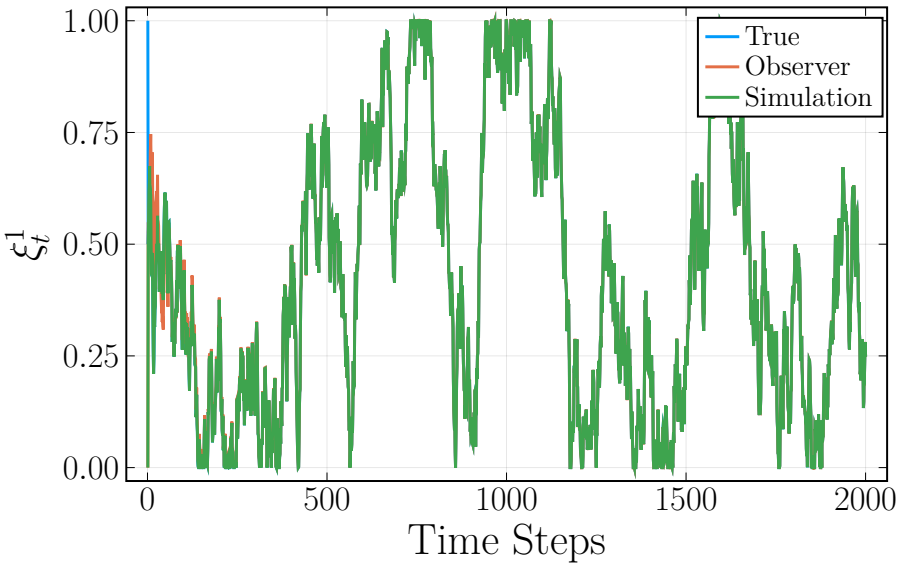
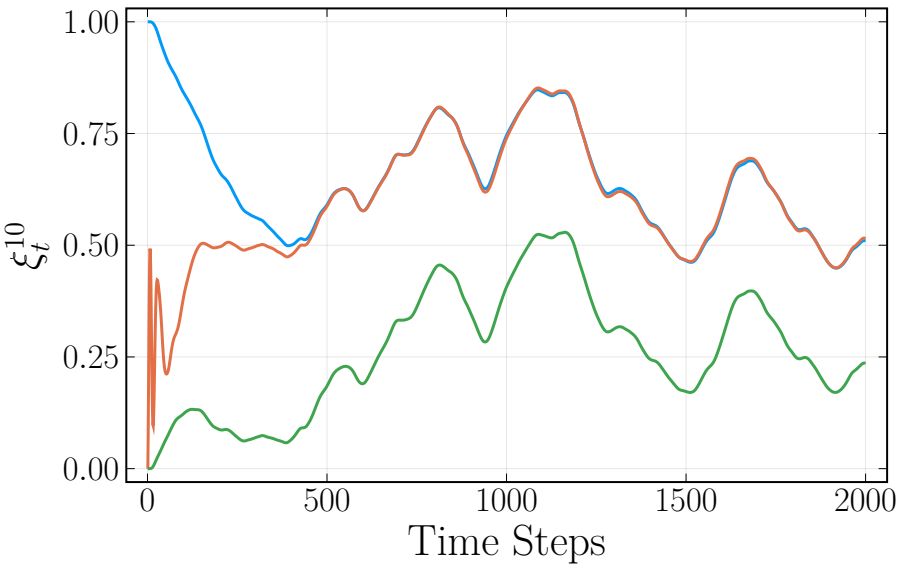



**Figure 5.7** – Simulation of a semi-linear reaction diffusion equation and the observer's state estimate, with a measurement in the centre of the spatial domain. The $y$-axis corresponds to the spatial dimension and the $x$-axis corresponds to the time dimension.




**(a)** True and estimated states for $\xi_t^1$, located at PDE boundary.




**(b)** True and estimated states for $\xi_t^{10}$.

**Figure 5.8** – True state and state estimates from the designed observer and a free run simulation of the PDE.

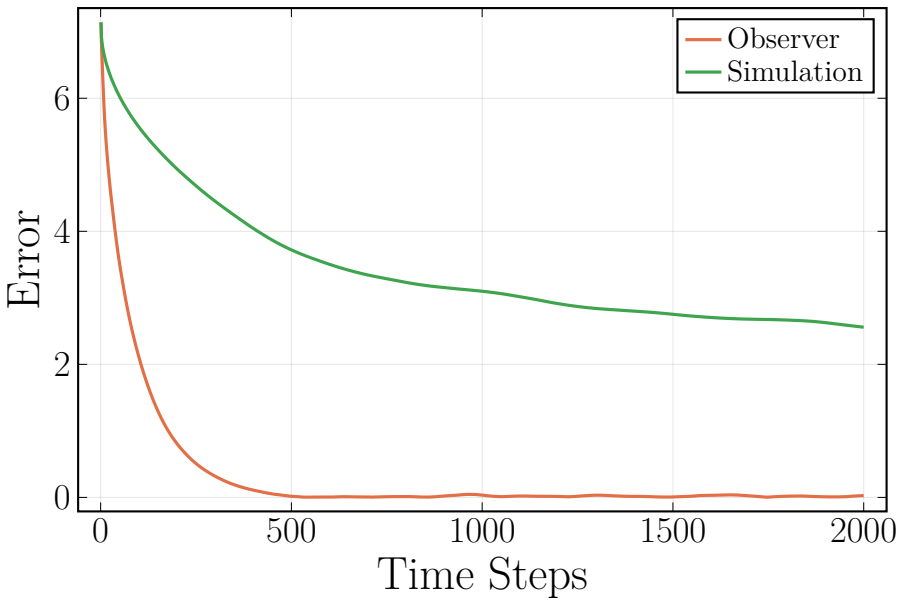



**Figure 5.9** – State estimation error for the nonlinear observer compared to a free run simulation from the same initial conditions.

## 5.9 Use Case: Data-Driven Feedback Control Design

In this section we show how a rich class of contracting nonlinear models can be useful for nonlinear feedback design for *linear* dynamical systems with stability guarantees. Even if the dynamics are linear, the presence of constraints, non-quadratic costs, and non-Gaussian disturbance can mean that non-linear policies are superior to linear policies. Indeed, in the presence of constraints, model predictive control is a common solution.

The basic idea we illustrate in this section is to build on a standard method for *linear* feedback optimization: the Youla-Kucera parameterization, a.k.a Q-augmentation [248, 256]. For a linear system model

$$x_{t+1} = \mathbb{A}x_t + \mathbb{B}_1 w_t + \mathbb{B}_2 u_t, \tag{5.69}$$

$$\zeta_t = \mathbb{C}_1 x_t + \mathbb{D}_{11} w_t + \mathbb{D}_{12} u_t. \tag{5.70}$$

$$y_t = \mathbb{C}_2 x_t + \mathbb{D}_{21} w_t. \tag{5.71}$$

with $x$ the state, $u$ the controlled input, $w_t$ external inputs (reference, disturbance, measurement noise), $y$ a measured output, and $\zeta$ comprises the "performance" outputs to kept small (e.g. tracking error, control signal). We assume the system is detectable and stabilizable, i.e. there exist $\mathbb{L}$ and $\mathbb{K}$ such that $\mathbb{A} - \mathbb{L}\mathbb{C}$ and $\mathbb{A} - \mathbb{B}\mathbb{K}$ are Schur stable. Note that if $\mathbb{A}$ is stable we can take $\mathbb{L} = 0, \mathbb{K} = 0$. Consider a feedback controller of the form:

$$\hat{x}_{t+1} = \mathbb{A}\hat{x}_t + \mathbb{B}_2 u_t + \mathbb{L}\tilde{y} \tag{5.72}$$

$$\tilde{y}_t = y_t - \mathbb{C}_2 \hat{x}_t \tag{5.73}$$

$$u_t = -\mathbb{K}\hat{x}_t + v_t \tag{5.74}$$

i.e. a standard output-feedback structure with $v_t$ an additional control augmentation.

The closed-loop input-output dynamics can be written as the transfer matrix

$$\begin{bmatrix} \zeta \\ \tilde{y} \end{bmatrix} = \begin{bmatrix} P_{\zeta w} & P_{\zeta v} \\ P_{\tilde{y} w} & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} \tag{5.75}$$

where we have used the fact that $v$ maps to $x$ and $\hat{x}$ equally, as the mapping from $v$ to $\tilde{y}$ is zero.

It is well-known that the set of all stabilizing linear feedback controllers can be parameterised by stable linear systems $Q : \tilde{y} \mapsto v$, and moreover this convexifies the closed-loop dynamics. A standard approach (e.g. [91, 32]) is to construct an affine parameterization for $Q$ via a finite-dimensional truncation of a complete basis of stable linear systems, and optimize to meet various criteria on frequency response, impulse response, and response to application-dependent test inputs.

However, if the control augmentation $v$ is instead generated by a contracting nonlinear system $v = Q(\tilde{y})$, then the closed-loop dynamics $w \mapsto \zeta$ are nonlinear but contracting and have the representation

$$\zeta = P_{zw} w + P_{zv} Q(P_{\tilde{y} w} w). \tag{5.76}$$

This presents opportunities for learning stabilizing controllers via parameterizations of stable nonlinear models.

### 5.9.1 Echo State Network and Convex Optimization

Here we describe a particular setting in which the data-driven optimization of nonlinear policies can be posed as a *convex* problem. Suppose we wish to design a causal feedback controller solving (at least approximately) a problem of the form:

$$\min_{\theta} J(\zeta) \tag{5.77}$$

$$\text{s.t. } c(\zeta) \leq 0 \tag{5.78}$$

where $\zeta$ is the response of the performance outputs to a *particular class* of inputs $w$, $J$ is a convex objective function, and $c$ is a set of convex constraints, e.g. state and control signal bounds.

If we take $Q$ as an echo state network, c.f. Section 5.5.3:

$$\begin{aligned} q_{t+1} &= f_q(q_t, \tilde{y}_t) \\ v_t &= g_q(q_t, \tilde{y}_t, \theta) \end{aligned}$$

where $f_q$ is fixed and $g_q$ is linearly parameterized by $\theta$, i.e.

$$g_q(q_t, \tilde{y}_t, \theta) = \sum_i \theta_i g_q^i(q_t, \tilde{y}_t).$$

Then $Q$ has the representation

$$Q(\tilde{y}) = \sum_i \theta_i Q^i(\tilde{y})$$

where $Q^i$ is a state-space model with dynamics $f_q$ and output $g_q^i$. Then, we can perform data-driven controller optimization in the following way:

1. Construct (e.g. via random sampling, experiment) a finite set of test signals $w^j$.
2. Compute $\tilde{y}_t^j = P_{\tilde{y}w} w^j$ for each $j$.
3. For each $j$, compute the response to $\tilde{y}^j$:

$$q_{t+1} = f_q(q_t, \tilde{y}_t^j), \quad v_t^{ij} = g_q^i(q_t, \tilde{y}_t^j).$$

4. Construct the affine representation

$$\zeta^j = P_{zw} w^j + \sum_i \theta P_{zv} v^{ij}.$$

5. Solve the convex optimization problem:

$$\theta^\star = \arg\min_\theta \; J(\zeta) + R(\theta)$$
$$\text{s.t. } c(\zeta^j) \leq 0$$

where $R(\theta)$ is an optional regularization term.

The result will of course only be approximately optimal, since $w^j$ are but a representative sample and the echo state network provides only a finite-dimensional span of policies. However it will be *guaranteed* to be stabilizing.

**Remark 5.7.** This framework can be extended to parameterizing robustly stabilizing controllers [256] and stabilizing controllers for nonlinear systems [225], and adaptive control via online convex optimization, e.g. [2, 197] use the Youla parameterization with $Q$ parameterized via its impulse response.

### 5.9.2 Example

We illustrate the approach on a simple discrete-time linear system with transfer function

$$P_{\zeta w} = P_{\zeta v} = -P_{\tilde{y} w} = \frac{1}{q^2 + 2\rho\cos(\phi)q + \phi^2}$$

with $q$ the shift operator, $\rho = 0.8$, and $\phi = 0.2\pi$. We consider the task of minimizing the $\ell^1$ norm of the output in response to step disturbances, while keeping the control signal $u$ bounded: $|u_t| \leq 5$ for all $t$.

Training data is generated by a 25,000 sample piecewise constant disturbance that has a hold time of 50 samples and a magnitude uniformly distributed in the interval [-10, 10]. An example is shown in Fig. 5.10.

We construct a contracting model $Q$ with $n = 50$ states and $q = 500$ neurons by randomly sampling a matrix $X_{ij} \sim \mathcal{N}\left[0, \frac{4}{2n+q}\right]$ where $X \in \mathbb{R}^{2n+q \times 2n+q}$ and constructing a C-aREN using the method outline in Section 5.5.1. The remaining parameters

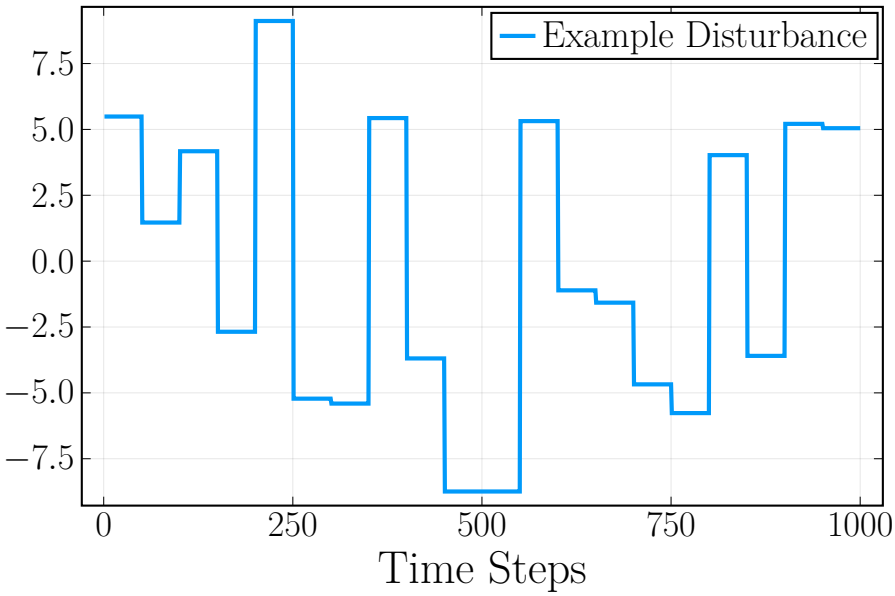



**Figure 5.10** – Example of the training disturbance.

are then sampled from the Glorot normal distribution [76]. We construct a linear $Q$ parameter of the form

$$q_{t+1} = A_q q_t + B_q \tilde{y}_t, \quad v_{t+1} = C_q q_t + D_q \tilde{y}_t.$$

We construct $A_q$ by first sampling a matrix $\bar{A}_{ij} \sim \mathcal{N}\left[0, \frac{1}{2n+q}\right]$ and then normalizing the eigenvalues so that $A_q = (1-\lambda)\frac{\bar{A}}{\rho(\bar{A})}$ so that $A_q$ is a stable matrix with a contraction rate of $\lambda$. We sample $B_q$ from the Glorot normal distribution [76].

The response to test inputs are shown in Figure 5.11. The benefits of learning a nonlinear $Q$ parameter are that the control can respond aggressively to small disturbances, driving the output quickly to zero, but respond less aggressively to large disturbances to stay within the control bounds. In contrast, the linear control policy must respond proportionally to disturbances of all sizes. Therefore, since the control constraints require less aggressive response to large disturbances, the controller also responds less aggressively to small disturbances and does not drive the output to zero.

## 5.10 Conclusions and Future Work

In this chapter, we have introduced RENs as a new model class for learning dynamical systems with stability and robustness constraints. The model set is flexible and admits a direct parameterization, allowing learning via unconstrained optimization.

We have illustrated the benefits of the new model class on problems in system identification, observer design, and control. On system identification benchmarks, the REN structure outperformed the widely-used RNN and LSTM models in terms of model fit while achieving far lower sensitivity to input perturbations. We illustrated the use in state estimation via data-driven observer design for an unstable nonlinear PDE. And finally, we illustrated the benefits for control design by linear constrained nonlinear feedback policies for linear systems.

Our future work will further explore uses of RENs for robust reinforcement learning, learning of robot motion primitives with passivity constraints, and privacy-preserving

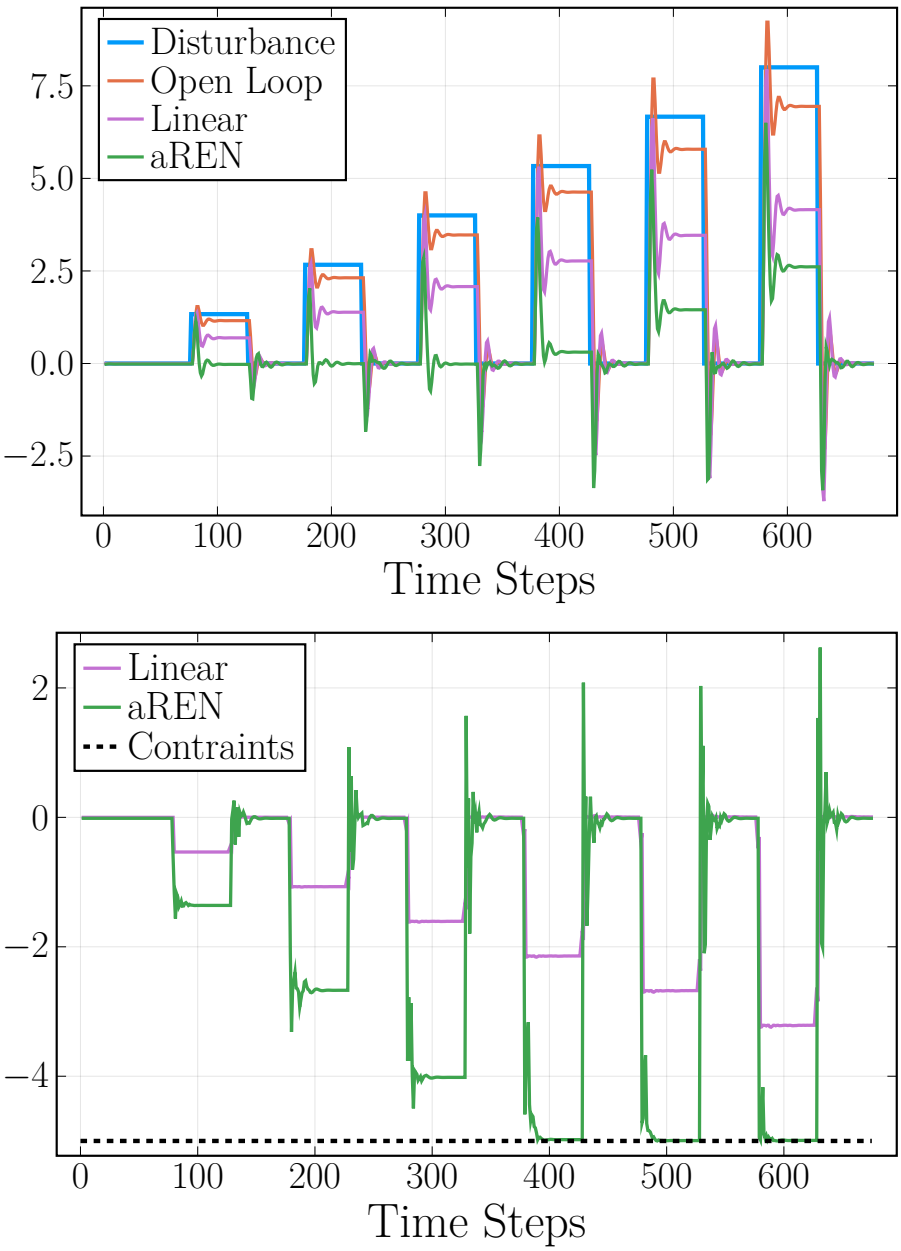



**Figure 5.11** – Output (top) and control signal (bottom) responses to step disturbances for nonlinear (aREN) and linear data-driven optimization of feedback controllers.

control policy design, as well as applications in online learning of feedback policies.

## 5.11 Proofs

### Proof of Proposition 5.1

To prove contraction, we consider (5.13) with $\Delta_u = 0$. By left-multiplying $\begin{bmatrix}\Delta x_t^\top & \Delta w_t^\top\end{bmatrix}$ and right-multiplying $\begin{bmatrix}\Delta x_t^\top & \Delta w_t^\top\end{bmatrix}^\top$ to (5.17), we obtain the following incremental Lyapunov inequality:

$$V_{t+1} - V_t < \Gamma_t \leq 0, \tag{5.79}$$

where $V_t = \Delta x_t^\top P \Delta x_t$. The second inequality follows by the incremental quadratic constraint (5.16). Since $V_t$ is a quadratic form in $\Delta x_t$, it follows that $V_{t+1} \leq \alpha V_t$ for some $\alpha \in [0, 1)$ and $V_t \leq \alpha^t V_0$.

Similarly, from (5.18) we can prove that the following incremental dissipation inequality holds for (5.13), (5.14):

$$V_{t+1} - V_t - \begin{bmatrix}\Delta y_t \\ \Delta u_t\end{bmatrix}^\top \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \begin{bmatrix}\Delta y_t \\ \Delta u_t\end{bmatrix} < \Gamma_t \leq 0. \tag{5.80}$$

### Proof of Theorem 5.1

To show well-posedness, from (5.20) we have $E + E^\top \succ P \succ 0$ and $\mathcal{W} = 2\Lambda - \Lambda\Lambda^{-1}\mathcal{D}_{11} - \mathcal{D}_{11}^\top\Lambda^{-1}\Lambda \succ 0$. The first LMI implies that $E$ is invertible [214] and thus (5.8) is well-posed. The second one ensures that the equilibrium network (4.1) is well-posed by the main result of [178].

To prove contraction, applying the inequality $E^\top\mathcal{P}^{-1}E \succeq E + E^\top - P$ and Schur complement to (5.20) gives

$$\begin{bmatrix} E^\top\mathcal{P}^{-1}E & -\mathcal{C}_1^\top \\ -\mathcal{C}_1 & \mathcal{W} \end{bmatrix} - \begin{bmatrix} F^\top \\ \mathcal{B}_1^\top \end{bmatrix} \mathcal{P}^{-1} \begin{bmatrix} F^\top \\ \mathcal{B}_1^\top \end{bmatrix}^\top \succ 0.$$

By substituting $F = EA$, $\mathcal{B}_1 = EB_1$, $\mathcal{B}_2 = EB_2$, $\mathcal{C}_1 = \Lambda C_1$ and $\mathcal{D}_{11} = \Lambda D_{11}$ into the above inequality, we obtain (5.17) with $P = E^\top \mathcal{P}^{-1} E$. Thus, $\Theta_C$ is a set of C-RENs.

Similarly, we can show that (5.21) implies (5.18), i.e., $\Theta_R$ is a set of R-RENs.

## Proof of Theorem 5.2

We first show that $\Theta_R \subset \Theta_C$. Applying Schur complement to (5.21) yields

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & \mathcal{C}_2^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{D}_{21}^\top & \mathcal{B}_1^\top \\ \mathcal{C}_2 & \mathcal{D}_{12} & \mathcal{R} & \mathcal{B}_2^\top \\ F & \mathcal{B}_1 & \mathcal{B}_2 & \mathcal{P} \end{bmatrix} \succ - \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \\ 0 \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \\ 0 \end{bmatrix}^\top .$$

By swapping the last two rows and columns for the above inequality, and then applying Schur complement to the component $\mathcal{R}$, we obtain Inequality (5.22). Then, Condition (5.20) follows by the facts $\mathcal{R} \succ 0$ and $Q \preceq 0$.

Now we show that any REN in $\Theta_C$ has a finite $\ell_2$ Lipschitz bound. That is, Condition (5.20) implies that there exists a sufficiently large but finite $\gamma$ such that (5.22) holds for $Q = -\frac{1}{\gamma} I, R = \gamma I, S = 0$, i.e.,

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} \succ \Phi_\gamma \tag{5.81}$$

where

$$\Phi_\gamma = \frac{1}{\gamma}\begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}\begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top + \begin{bmatrix} -\frac{1}{\gamma}C_2^\top D_{22} \\ -\frac{1}{\gamma}D_{21}^\top D_{22} - \mathcal{D}_{12} \\ \mathcal{B}_2 \end{bmatrix}$$
$$\left(\gamma I - \frac{1}{\gamma}D_{22}^\top D_{22}\right)^{-1}\begin{bmatrix} -\frac{1}{\gamma}C_2^\top D_{22} \\ -\frac{1}{\gamma}D_{21}^\top D_{22} - \mathcal{D}_{12} \\ \mathcal{B}_2 \end{bmatrix}^\top .$$

For sufficiently large $\gamma$, we have

$$\Phi_\gamma \approx \frac{1}{\gamma}\left(\begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}\begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top + \begin{bmatrix} 0 \\ -\mathcal{D}_{12} \\ \mathcal{B}_2 \end{bmatrix}\begin{bmatrix} 0 \\ -\mathcal{D}_{12} \\ \mathcal{B}_2 \end{bmatrix}^\top\right),$$

which is a positive-semidefinite matrix with arbitrary small spectral radius. Thus, for any model in $\Theta_C$, Condition (5.81) holds for some sufficiently large but finite gain bound $\gamma$.

## Proof of Proposition 5.2

By construction $M + M^\top \succ 0$, so we have

$$I - Z^\top Z \succ 0,$$

by the properties of the Cayley transform. Then by direct substition, $D_{22}$ as constructed in (5.35) satisfies

$$R - S\mathcal{Q}^{-1}S^\top \succ \left(L_Q D_{22} - L_Q^{-\top} S^\top\right)^\top \left(L_Q D_{22} - L_Q^{-\top} S^\top\right),$$

which can rearranged as (5.31) via the factorizations (5.34).

## Proof of Proposition 5.3

When the correctness condition (5.58) holds, we have that $\hat{x}_t = x_t$ for all $t \geq 0$ if $\hat{x}_0 = x_0$, i.e. the true state trajectory is a particular solution of the observer. But contraction implies that all solutions of the observer converge to each other. Hence when $w = 0$ we have $\hat{x}_t \to x_t$ as $t \to \infty$.

Now we consider the case where the correctness condition does not hold but its error is bounded, i.e., $|e(x, u)| \leq \rho$. The dynamics of $\Delta x := \hat{x} - x$ can be written as

$$\begin{aligned}\Delta x_{t+1} &= f_o(\hat{x}_t, u_t, y_t) - f_m(x_t, u_t) \\ &= F(x_t, u_t)\Delta x_t + e_t\end{aligned}$$

where $F(x_t, u_t)\Delta x_t := f_o(x_t + \Delta x_t, u_t, y_t) - f_o(x_t, u_t, y_t)$ and $e_t := e(x_t, u_t)$. Letting $V_t := \Delta x_t^\top P \Delta x_t$, we have

$$\begin{aligned}V_{t+1} - V_t =& 2e_t^\top P\Delta x_t + \Delta x_t^\top F^\top P F \Delta x_t - \Delta x_t^\top P \Delta x_t \\ \leq& 2e_t^\top P\Delta x_t - (1-\alpha)\Delta x_t^\top P \Delta x_t.\end{aligned}$$

The above inequality is based on $F^\top P F - \alpha P \preceq 0$ (i.e., contraction). When $t \geq T$ with $T$ sufficiently large, the estimation error satisfies $2e_t^\top P\Delta x_t - (1-\alpha)\Delta x_t^\top P\Delta x_t \geq 0$, which gives the bound in (5.60).

# Chapter 6

# Distributed Identification Of Monotone and/or Contracting Networks

In the previous chapters, we developed methods for training neural network and recurrent neural network models with behavioural guarantees in the form of incremental IQCs. These model sets can guarantee properties such as contraction, Incremental $\ell_2$ gain bounds and incremental passivity using quadratic constraints, however, the reliance on quadratic constraints inherently limits the scalability of the methods.

In this chapter, we study a new problem setting; the problem of learning with behavioral constraints in a large-scale networked setting. Specifically, we propose methods for the identification of large-scale networked systems with guarantees that the resulting model will be contracting and/or monotone, i.e., the order relations between states are preserved. The main challenges that we address are: simultaneously searching for model parameters and a certificate of stability while also scaling to networks with hundreds or thousands of nodes. We propose a model set that admits convex constraints for stability and monotonicity, and has a separable structure that allows distributed identification via the alternating directions method of multipliers (ADMM). The performance and scalability of the approach is illustrated on a vari-

ety of linear and nonlinear case studies, including a nonlinear traffic network with a 200-dimensional state space.

Note that the methods developed in this chapter are distinct from the prior chapters and it might be best to view this as a stand-alone chapter.

### Publications

Some of the content of this chapter has previously appeared in the following publication:

> Max Revay, Ruigang Wang, and Ian R. Manchester. Distributed identification of contracting and/or monotone network dynamics. *Transactions on Automatic Control*, 2021.

## 6.1 Introduction

System identification is the process of generating dynamic models from data [127], and is also referred to as *learning dynamical systems* (e.g. [208]). When scaling control and identification algorithms to large-scale systems, it can be useful to treat a system as a sparse network of local subsystems interconnected through a graph [21, 195, 223].

However, if the models of local subsystems are identified separately and then recombined as a networked model, the quality of fit can deteriorate dramatically due to unmodelled dynamic interactions between subsystems. In particular, model instability is a common outcome even if the underlying data-generating system is stable. Furthermore, many large-scale systems have known behavioral properties such as monotonicity or positivity (see Sec 6.1.3), and it is often important that the identified network model preserves these.

In this chapter, we propose model structures and algorithms for the identification of networked systems in state space form:

$$x_{t+1} = a(x_t, u_t, u_{t+1}), \tag{6.1}$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the state and input respectively, and the model dynamics $a(\cdot, \cdot, \cdot)$ can be either linear or nonlinear. We assume that measurements (or estimates) of state and input sequences are available.

Our approach:

1. uses distributed computation, i.e., network nodes only share data and parameters with immediate neighbors,
2. can (optionally) generate models with certificates of a strong form of nonlinear stability called contraction [129],
3. can (optionally) generate monotone and positive models, i.e. with prescribed ordering and sign relations between states [202].

If the system to be modelled is known to be contracting and/or monotone, then requiring these behavioral properties of the identified models provides two main benefits:

1. Firstly, it may be *required* that the model satisfies these properties for it to be useful, e.g. an unstable model is useless for long-term predictions, while a model of traffic density in a road network is non-sensical if it predicts a negative number of vehicles will be present, and similarly for chemical concentrations in a reaction model. Furthermore, several algorithms for control design make use of monotonicity properties of models (see Section 6.1.3).
2. Even if not a model requirement, these behavioral properties comprise *prior information* about the system, and imposing them on the model can be thought of as a form of *regularization*. We observe in numerical experiments in Section

6.6 that this can result in significantly-improved quality of fit compared to unconstrained models.

The key technical difficulty we address is the identification of large-scale systems while simultaneously guaranteeing model stability and/or monotonicity. We propose a convex model set with scalable behavioral constraints and an algorithm based on ADMM that decomposes the identification problem into easily solvable subproblems that require only local communication between subsystems.

### 6.1.1 Identification of Networked Systems

Standard approaches to system identification do not work well for large-scale networked systems for three reasons [83]: firstly, the dataset must be collected at a central location, a process which may be prohibitive for complex systems; secondly, the computational and memory complexities prohibit the application of many methods to large systems; finally, the network structure may not be preserved by identification. For instance, standard subspace identification methods have $\mathcal{O}[n^3]$ an $\mathcal{O}[n^2]$ computational and memory complexities respectively, and the sparsity patterns corresponding to network structure are destroyed through a dense similarity transformation [229]. These issues motivate approaches based on modelling subsystems within a network.

Previous work in networked system identification can be loosely categorized into two areas; the identification of a network's topology [139, 140, 185], and the identification of a system's dynamics with known topology. In the latter category, almost all prior work has focused on the case where the subsystems are linear time invariant (LTI) and described by state space models [83, 249, 250] or transfer functions (a.k.a. modules) [224, 53].

When identifying the subsystem dynamics, the states or outputs of neighbors are treated as exogenous inputs, ignoring the feedback loops induced by the network topology. This improves scalability as the identification of each subsystem can be performed in parallel. However, accurate identification of the individual subsystems

does not imply accurate identification of the full network, because the ignored feedback loops may have a strong effect and even introduce instability. A simple case with two subsystems which has received significant attention is closed-loop identification (see e.g. [70]).

In prior works on networked system identification, the only sense in which model stability or other behavioral properties have been guaranteed is via identifiability [234] and consistency [223]. That is, if it can be proven that in the limit of infinite data the model converges to the true system, and the true system is stable, then with sufficient data a stable model will be found. However, in real applications, the data set is always finite and the real system is never in the model set, and as such there are no hard guarantees of model stability with existing methods.

### 6.1.2 Identification of Stable and Contracting Models

While the problem of model instability is exacerbated by the interconnection of local models, even in standard (centralized) identification, it is a significant and well-studied problem. Even for linear system identification, the problem is non-trivial, since the set of stable models is not convex using standard parameterizations, and several methods have been suggested based on regularization or model constraints [226, 131, 117, 147, 220].

For linear systems most definitions of stability are equivalent. For nonlinear systems the situation is more nuanced, and the definition used depends on the requirements of the problem at hand. Classical Lyapunov stability is arguably not appropriate in system identification as the stability certificate must be constructed about a known stable solution, whereas the purpose of system identification is to predict a system's response to previously unseen inputs. Contraction [129] and incremental stability (e.g. [215]) are more appropriate since they ensure the stability of all possible solutions and consequently, do not require a-priori knowledge of the inputs and state trajectories.

A central technical problem in identifying stable nonlinear models is the following: while the search for a stability certificate such as a Lyapunov function or contraction

metric for a given model is convex, the *joint search* for a model and a certificate of its stability is not. Some approaches, e.g., [222, 148], effectively fix the stability certificate, but this can be conservative even for linear systems.

This chapter builds on previous work in jointly-convex parameterization of models and their stability certificates via implicit models [144, 30, 212, 214] and chapters 3 and 5, and associated convex bounds for model fidelity via Lagrangian relaxation [214, 219, 218].

While convex, these approaches are not well-suited to large-scale systems, since they are based on linear matrix inequalities and sum-of-squares constraints that scale poorly with large state dimension. A key development in this chapter is to significantly improve scalability of this approach via a novel model parameterization and new contraction constraints that are *separable* into many smaller constraints which can be enforced in parallel.

### 6.1.3 Monotone and Positive Systems

Monotone systems are a class of dynamic system characterized by the preservation of an order relation for solutions (c.f. Definition 6.2 below). A closely related class is *positive systems*, for which state variables remain non-negative for all non-negative inputs (c.f. Definition 6.3 below). A monotone system is also positive if the zero state is a solution with zero input, and for linear systems positivity and monotonicity are equivalent. Processes such as traffic networks [46, 130, 49], chemical reactions [55], combination therapies [173, 89, 90, 101], wildfires [203] and power scheduling [173] exhibit monotone and/or positive dynamics.

A useful property of monotone systems is that they often admit simplified stability tests. In particular, for linear positive systems the existence of *separable* Lyapunov functions, i.e. those representable as the sum or maximum over functions of individual state variables, is necessary and sufficient for stability [24]. This property has been used to simplify analysis [84], control [172] and identification [217] of positive systems. Separable stability certificates have also been shown to exist for certain classes of

nonlinear monotone systems [92, 58, 137, 106]. and have been used for distributed stability verification [48] and control [193]. Monotonicity can also simplify nonlinear model predictive control [173] and formal verification using signal temporal logic [184].

There are however, few identification algorithms that guarantee monotonicity. In [191], monotone gene networks are identified using the monotone P-splines developed in [192]. This approach, however, does no guarantee model stability.

### 6.1.4 Model Quality-of-Fit Criteria

Identification and learning typically involve the optimization of a quality of fit metric, a.k.a. a loss function, over a model set. Arguably the simplest and most widely-applied quality-of-fit metric in system identification is least-squares equation error (a.k.a. one-step-ahead prediction error):

$$J_{ee}(\theta) = \sum_{t=0}^{T-1} |a(\tilde{x}_t, \tilde{u}_t) - \tilde{x}_{t+1}|^2, \tag{6.2}$$

where $\tilde{x}_t \in \mathbb{R}^n$ and $\tilde{u}_t \in \mathbb{R}^m$ are state and input measurements or estimates.

Least-squares equation error is a natural choice for optimizing short-term predictions. In many contexts, system state measurements are not available. Nevertheless, equation error frequently arises as a sub-problem via estimated states, e.g. in subspace identification algorithms [227, 228, 250], where states are estimated using using matrix factorizations, or in maximum likelihood identification via the expectation maximization (EM) algorithm where they are estimated from the joint smoothing distribution [186, 220].

Long-term prediction performance can be evaluated via simulation error, defined as

$$J_{se}(\theta) = \sum_{t=0}^{T-1} |x_t - \tilde{x}_t|^2, \;\; s.t. \; x_{t+1} = a(x_t, \tilde{u}_t), \tag{6.3}$$

where the initial conditions may be fixed or estimated. The dependence on simulated states, however, renders the cost function non-convex [200, 127] and notoriously dif-

ficult to optimize [179]. Consequently, equation error optimization is often used to initialize local optimization of simulation error (e.g. via gradient descent) or used as a surrogate for simulation error that is easier to optimize. In the latter context, model stability is particularly important since a model can have small equation error but be unstable and therefore exhibit very large simulation error. In fact, when a model is contracting, it can be shown that small equation error implies small simulation error [144].

## 6.2 Preliminaries and Problem Setup

### Notation

We will introduce some additional notation that is specific to this chapter.

A graph $\mathscr{G}$ is defined by a set of nodes (vertices) $\mathscr{V} = [1, ..., N]$ and edges $\mathscr{E} \subset \mathscr{V} \times \mathscr{V}$. The vector $\mathbf{1}$ is the column vector of ones, with size inferred from context. For vectors $v$, $v > 0$ refers to the element-wise inequality. For matrices $M$, $M \geq 0$ and $M \leq 0$ refer to element-wise inequalities. For symmetric matrices $M$, $M \succ 0$ means that $M$ is positive definite. For a vector $v$, $diag(v)$ is the matrix with the elements of $v$ along the diagonal. The set of $n \times n$ non-singular M-matrices is denoted $\mathbb{M}^n$. For a matrix $A$, $A \in \mathbb{M}^n$ means $A^{ij} \leq 0$, $\forall i \neq j$ and $\text{real}(\lambda_i) > 0$ for $i = 1, ..., n$, where $\lambda_i$ are the eigenvalues of $A$. For brevity, we will sometimes drop the arguments from a function where the meaning may be inferred from context.

### 6.2.1 Behavioural Properties via Differential Dynamics

Both contraction [129] and monotonicity [202] can be verified by way of a system's *differential dynamics*, a.k.a. linearized, variational, or prolonged dynamics. For the system (6.1), the differential dynamics are

$$\delta_{x_{t+1}} = A(x_t, u_t, u_{t+1})\delta_{x_t} + B(x_t, u_t, u_{t+1})\delta_{u_t}. \tag{6.4}$$

where $A = \frac{\partial a}{\partial x}$ and $B = \frac{\partial a}{\partial u}$. In conjunction with (6.1), the differential dynamics describe the linearized dynamics along *all* solutions of the system.

### Contraction Analysis

We use the following definition of nonlinear stability:

**Definition 6.1** (Contraction)**.** *A system is termed contracting with rate $\alpha$, where $0 < \alpha < 1$, if for any two initial conditions $x_0^a$, $x_0^b$, given the same input sequence $u_t$, and some $p \in [1, \infty]$, there exists a continuous function $b_p(x_0^a, x_0^b) > 0$ such that the corresponding trajectories $x_t^a, x_t^b$ satisfy $|x_t^a - x_t^b|_p < \alpha^t b_p(x_0^a, x_0^b)$.*

Contraction can be proven by finding a contraction metric which verifies conditions on the differential dynamics [129]. A contraction metric is a function $V(t, x, \delta_x)$ such that:

$$V(t, x, 0) = 0, \quad V(t, x, \delta) \geq \mu|\delta|_p, \tag{6.5}$$

$$V(t+1, x_{t+1}, \delta_{x_{t+1}}) \leq \alpha V(t, x, \delta_x). \tag{6.6}$$

for some $\mu > 0$

The choice of contraction metric $V(t, x, \delta)$ is problem dependent. Prior works have proposed quadratic contraction metrics for which (6.6) is linear in the stability certificate and can be verified using semi-definite programming. A number of works have also noted that using a weighted $\ell_1$ norm can lead to separable constraints [182, 48] allowing for stability verification of large-scale networked systems.

In the context of system identification, the joint search for model $a$ in (6.1) and contraction metric $V$ is non-convex due to the nonlinear function composition $V(t+1, x_{t+1}, \delta_{x_{t+1}}) = V(t+1, a(x, u), A(x, u)\delta_{x_t})$.

### Monotone and Positive Systems

We now define system monotonicity and positivity of dynamical systems.

**Definition 6.2** (Monotone System)**.** *A system (6.1) is termed monotone if for inputs $u_t^a$ and $u_t^b$ and initial conditions $x_0^a$, $x_0^b$, the following implication holds:*

$$x_0^a \geq x_0^b,\ u_t^a \geq u_t^b\ \forall t \implies x_t^a \geq x_t^b\ \forall t.$$

Monotonicity results from $A(x,u) \geq 0$ and $B(x,u) \geq 0$ where $A$ and $B$ come from the differential dynamics (6.4).

**Definition 6.3** (Positive System)**.** *A system (6.1) is positive if for all inputs $u_0, ..., u_T \geq 0$ and initial conditions $x_0 \geq 0$, the resulting trajectory has $x_1, ..., x_T \geq 0$.*

A sufficent condition for a system to positive is for it to be monotone and admit $x_t = 0, u_t = 0\,\forall t$ as a solution, i.e. $a(0,0,0) = 0$ in (6.1).

### 6.2.2 Network Structure

We assume model (6.1) is partitioned into $N$ subsystems. The interactions between these subsystems is described by a directed graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. Here, we have a set of nodes denoted $\mathscr{V} = \{1, ..., N\}$ corresponding to the subsystems. Each subsystem has its own state denoted $x^i \in \mathbb{R}^{n_i}$ and may take an input denoted $u^i \in \mathbb{R}^{m_i}$ (we allow for the case $m_i = 0$). The whole-network state and input is attained by concatenating the states and inputs of each subsystem,

$$x = \begin{bmatrix} x^1 \\ \vdots \\ x^N \end{bmatrix}, \quad u = \begin{bmatrix} u^1 \\ \vdots \\ u^N \end{bmatrix}. \tag{6.7}$$

The set of edges $\mathscr{E} \subseteq \mathscr{V} \times \mathscr{V}$ describes how the subsystems interact with each other. In particular, $(j,i) \in \mathscr{E}$ means that the state of subsystem $j$ affects the state of subsystem $i$. The edge list $\mathscr{E}$ may arise naturally from the context of the problem, e.g. in traffic networks where edges come from the physical topology of the road network, or may be identified from data [139, 141].

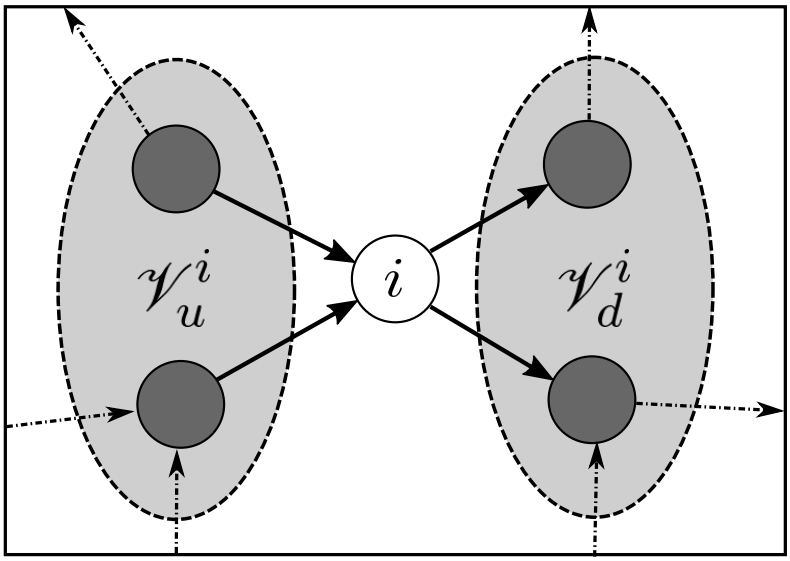



**Figure 6.1** – Illustration of upstream/downstream notation: in a directed graph, node $i$'s upstream neighbours $\mathscr{V}_u^i$ have edges going towards $i$, while its downstream neighbours $\mathscr{V}_d^i$ have edges coming from $i$.

For each subsystem $i \in \mathscr{V}$, we define the set of upstream neighbours $\mathscr{V}_u^i = \{j|(j,i) \in \mathscr{E}\}$ and the set of downstream neighbours $\mathscr{V}_d^i = \{j|(i,j) \in \mathscr{E}\}$. The term *upstream neighbours* of $i$ refers to the subsystems whose state affects the state of subsystem $i$, and the term *downstream neighbours* refers to the subsystems whose state is affected by subsystem $i$'s state. In general, we allow self-loops so that a node can (and generally will) be both upstream and downstream to itself. This notation is illustrated in Fig. 6.1.

We can write the dynamics of the individual interacting subsystems as follows:

$$x_{t+1}^i = a^i(\breve{x}_t^i, \breve{u}_t^i, \breve{u}_{t+1}^i), \quad i = 1, ..., N. \tag{6.8}$$

where $a^i$ corresponds to the $i^{th}$ element in (6.1) and $\breve{x}^i = \{x_j \mid j \in \mathscr{V}_u^i\}$ and $\breve{u}^i = \{u^j \mid j \in \mathscr{V}_u^i\}$.

### 6.2.3 Separable Optimization using ADMM

Consider an optimization problem of the form,

$$\min_{\theta} J(\theta), \tag{6.9}$$

which may include constraints on $\theta$ via indicator functions appearing in $J$. The indicator function for the constraint $\theta \in \Theta$ is the function $\mathcal{I}_{\Theta}(\theta)$ which is zero for $\theta \in \Theta$ and infinite otherwise.

**Definition 6.4** (Separable)**.** *The problem (6.9) is termed separable with respect to the partitioning $\theta = \{\theta^i \mid i = 1, .., N\}$ if it can be written as $J(\theta) = \sum_{i=1}^{N} J^i(\theta^i)$.*

In this cahpter we encounter problems of the form:

$$\min_{\theta} \sum_{i=1}^{N} J_a^i(\theta_a^i) + \sum_{j=1}^{M} J_b^j(\theta_b^j), \tag{6.10}$$

where $\{\theta_a^i \mid i = 1, ..., N\}$ and $\{\theta_b^j \mid j = 1, ..., M\}$ are two different partitions of the same vector $\theta$. In our context, these partitionings correspond to the sets of upstream or downstream neighbors discussed in the previous section. For such problems, the alternating directions method of multipliers (ADMM) can be applied [31]. We write (6.10) as

$$\begin{aligned} \min_{\theta,\phi} \quad & \sum_{i=1}^{N} J_a^i(\theta_a^i) + \sum_{j=1}^{M} J_b^j(\phi_b^j), \\ s.t. \quad & \theta - \phi = 0. \end{aligned} \tag{6.11}$$

Applying ADMM results in iterations in which each step is separable with respect to the partition $\theta_a$ or $\theta_b$, and can thus be solved via distributed computing. For convex problems, ADMM is guaranteed to converge to the optimal solution [31].

### 6.2.4 Problem Statement

To summarise, the main objective of this chapter is: Given state and input measurements $\{\tilde{x}_t, \tilde{u}_t \mid t = 1, .., T\}$, and a graph $\mathscr{G}$ describing the network topology, identify models (6.8) at each node such that:

1. During the identification procedure, each subsystem only communicates with immediate (upstream and downstream) neighbours;
2. Convergence of the algorithm is guaranteed and least-squares equation error is small at each subsystem;
3. Model behavioural constraints such as contraction, monotonicity, and/or positivity can be guaranteed for the interconnected model (6.1).

## 6.3 A Model Class with Convex Behavioural Constraints

In this section, we develop a convex parametrization of models with contraction, monotonicity and/or positivity guarantees. As described in subsection 6.2.1, jointly searching for a model (6.1) and contraction metric is nonconvex.

Following [212, 214], we solve this problem by instead searching for models in the following implicit form:

$$e(x_{t+1}, u_{t+1}) = f(x_t, u_t). \tag{6.12}$$

The differential dynamics of (6.12) are:

$$E(x_{t+1}, u_{t+1})\delta_{x_{t+1}} = F(x_t, u_t)\delta_{x_t} + K(x_t, u_t)\delta_{u_t}, \tag{6.13}$$

where $E = \frac{\partial e}{\partial x}$, $F = \frac{\partial f}{\partial x}$ and $K = \frac{\partial f}{\partial u}$.

**Definition 6.5** (Well-Posed)**.** *An implicit model of the form (6.12) is termed well-posed if for every $x_t, u_t, u_{t+1}$ there is a unique $x_{t+1}$ satisfying (6.12).*

I.e., well-posedness means that $e(x, u)$ is a bijection with respect to its first argument, and implies the existence of an explicit model of the form (6.1) where $a = e^{-1} \circ f$. Furthermore, it implies that for any initial condition $x_0$ and sequence of inputs $u_0, ..., u_T$, there exists a unique trajectory $x_1, ..., x_T$ satisfying (6.12).

### 6.3.1 Stability and Monotonicity Constraints

In this section, we propose convex conditions on the implicit model (6.12) that guarantee well-posedness, monotonicity, positivity, and contraction. The main result is the following:

**Theorem 6.1.** *A model of the form (6.12) is:*

1. ***well-posed*** *if there exists $\epsilon > 0$ such that for all $(x, u)$,*

$$E(x, u) + E(x, u)^T \succ \epsilon I, \tag{6.14}$$

2. ***contracting*** *with rate $\alpha$ if (a) holds and there exists a matrix function $S(x, u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n\times m}$ such that for all $(x, u)$:*

$$-S(x, u) \leq F(x, u) \leq S(x, u), \tag{6.15}$$

$$\mathbf{1}^\top(\alpha E(x, u) - S(x, u)) \geq 0, \tag{6.16}$$

3. ***monotone*** *if (a) holds and for all $(x, u)$:*

$$F(x, u) \geq 0, \quad K(x, u) \geq 0, \quad E(x, u) \in \mathbb{M}^n, \tag{6.17}$$

4. ***positive*** *if (c) holds and:*

$$e(0) = f(0, 0), \tag{6.18}$$

5. ***contracting and monotone*** *if (a) and (c) hold, and for all* $(x, u)$

$$\mathbf{1}^\top(\alpha E(x,u) - F(x,u)) \geq 0. \tag{6.19}$$

*Positivity is also enforced if (6.18) holds.*

*Proof.* See Section 6.8. □

We refer to the stability conditions in Theorem 6.1 (b) or (e) as $\ell_1$ contraction conditions as they ensure contraction using a state dependent weighted $\ell_1$ norm of the differentials: $V(t, x, \delta) = |E(x, u)\delta|_1$, noting that for the purpose of contraction analysis the exogenous input $u$ can be considered as a time-variation.

**Remark 6.1.** Theorem 6.1 requires an exponential contraction rate $\alpha$ to be specified. A weaker form of incremental stability can also be imposed by replacing (6.16) with

$$\mathbf{1}^\top(E(x,u) - S(x,u)) \geq \mu\mathbf{1}^\top \tag{6.20}$$

for some $\mu > 0$, and similarly for (6.19). This implies that $\sum_{t=0}^{\infty} |x_t^a - x_t^b|_1 < \infty$ and hence $x_t^a \to x_t^b$ asymptotically, following a line of reasoning similar to [214].

### 6.3.2 Specific Model Parametrizations

As formulated above, Theorem 6.1 applies to models represented by the infinite-dimensional space of continuously differentiable functions $e$ and $f$. In practice, these functions are usually parametrized by a finite-dimensional vector. In this section we briefly discuss some common model parametrizations and how the constraints can be enforced.

For linear models, (6.14) is a semidefinite constraint, (6.15)-(6.19) are linear and can be enforced using semidefinite programming. Furthermore, if $E$ is diagonal, then (6.14) is also linear and the model set is polytopic.

If the functions $e$ and $f$ are multivariate polynomials or trigonometric polynomials, then the constraints can be enforced using sum of squares programming [162, 143].

The model set (6.12) also contains a class of recurrent neural networks with slope-restricted, invertible activation functions. In this case, $e(x)$ is the inverse of the activation functions, $f(x, u)$ is affine, and simulation of the explicit model $a = e^{-1} \circ f$ yields the equation of a standard recurrent neural network [64]. The conditions in Theorem 6.1 (b) or (d) then correspond to diagonal dominance conditions on the weight matrices which can be enforced via linear constraints.

Finally, if the requirement for global verification of these properties is relaxed, then these constraints can be applied pointwise for arbitrary parametrizations $e$ and $f$, which amount to linear and semidefinite constraints if $e$ and $f$ are linearly parametrized.

## 6.4 Distributed Identification

In this section we consider the problem of distributed identification of networked systems with the behavioral constraints introduced in Theorem 6.1. First, we propose a particular structure for (6.12) for which the constraints in Theorem 6.1 are separable. We then formulate an objective function that is separable (with respect to a different partition). Finally we propose an algorithm based on ADMM for fitting the proposed models that requires only local communication between subsystems at each step.

### 6.4.1 Distributed Model

We propose the following model structure for distributed identification, in which $e$ depends only on local states and inputs, and $f$ is a summation of nonlinear functions of states and inputs from upstream neighbours:

$$e^i(x^i_{t+1}, u^i_{t+1}) = \sum_{j \in \mathcal{V}^i_u} f^{ij}(x^j, u^j). \tag{6.21}$$

Models of the form (6.21) are widely used for statistical modelling, and are referred to as generalized additive models (GAMs) [85]. This class of models also includes linear systems, and a class of recurrent neural networks. We assume that each of the functions $e^i : \mathbb{R}^{n_i} \times \mathbb{R}^{m_i} \mapsto \mathbb{R}^{n_i}$ and $f^{ij} : \mathbb{R}^{n_j} \times \mathbb{R}^{m_i} \mapsto \mathbb{R}^{n_i}$ are linearly parametrized by $\theta_e^i$ and $\theta_f^{ij}$ respectively.

We define two partitions of the model parameters; the sets of upstream and downstream parameters. These are denoted $\theta_u^i = \{\theta_e^i, \theta_f^{ij} | j \in \mathscr{V}_u^i\}$ and $\theta_d^i = \{\theta_e^i, \theta_f^{ji} | j \in \mathscr{V}_d^i\}$ respectively. Objective functions, constraints and optimization problems are called *upstream-separable* or *downstream-separable* if they are separable with respect to these partitions. Upstream and downstream separable optimization problems are closely related to the column-wise and row-wise separable optimization problems used in [233].

For the parametrization (6.21), the differential dynamics have a sparsity pattern determined by the network topology. In particular, the $(i,k)^{th}$ block of $F$ is:

$$F^{ik} = \frac{\partial}{\partial x^k} \sum_{j \in \mathscr{V}_u^i} f^{ij}(x^j, u^j) = \begin{cases} \frac{\partial f^{ik}}{\partial x^k}, & k \in \mathscr{V}_u^i \\ 0, & k \notin \mathscr{V}_u^i \end{cases}.$$

and $E$ is block diagonal. This means $F^{ik}$ depends only on parameters $\theta_f^{ik}$ and the block $E^{ii}$ depends only on $\theta_e^i$. As each block of $E$ and $F$ has an independent parametrization, functions of disjoint sets of elements of $E$ or $F$ will be separable.

### 6.4.2 Convex Bounds for Equation Error

In Section 6.4.1 we propose a convex set of implicit models. However, this approach shifts the convexity problem from the model set to the objective function as equation error (6.2), with $a = e^{-1} \circ f$, is not convex with respect to $e$ and $f$.

A simple approach is be to minimize the implicit equation error

$$J_{iee} = \sum_{t=1}^{T-1} |e(\tilde{x}_{t+1}, \tilde{u}_{t+1}) - f(\tilde{x}_t, \tilde{u}_t)|^2 \tag{6.22}$$

as a surrogate for equation error. This approach, however, strongly biases the resulting model and leads to poor performance [218]. Instead we use the convex upper bound for equation error proposed in [218], which is based on Lagrangian relaxation.

The least-squares equation error (6.2) for the implicit model (6.12) is:

$$\begin{aligned} \min_{\theta, x_2, \ldots, x_T} \quad & J_{ee}(\theta) = \sum_{t=1}^{T-1} |x_{t+1} - \tilde{x}_{t+1}|^2 \\ s.t. \quad & e(x_{t+1}, \tilde{u}_{t+1}) = f(\tilde{x}_t, \tilde{u}_t), \quad \forall t = 1, \ldots, T-1. \end{aligned} \tag{6.23}$$

Note that this problem is not jointly convex in $x_{t+1}$ and $\theta$. The following convex upper bound was proposed in [218]:

$$\begin{aligned} J_{ee} \leq \hat{J}_{ee}(\theta) = \sum_{t=1}^{T-1} \sup_{x_{t+1}} \Big\{ & |x_{t+1} - \tilde{x}_{t+1}|^2 \\ & - 2\lambda(x_{t+1})^\top (e(x_{t+1}, \tilde{u}_{t+1}) - f(\tilde{x}_t, \tilde{u}_t)) \Big\}, \end{aligned} \tag{6.24}$$

where $\lambda_t(x_{t+1}) = x_{t+1} - \tilde{x}_{t+1}$ is a Lagrange multiplier. The function (6.24) is convex in $\theta$ as it is the supremum of an infinite family of convex functions [214].

For our parametrization (6.21), $E$ is block-diagonal which then implies that (6.24) is upstream separable, so it can be written as

$$\hat{J}_{ee}(\theta) = \sum_{i=1}^{N} \hat{J}^i_{ee}(\theta^i_u), \tag{6.25}$$

where

$$\begin{aligned} \hat{J}^i_{ee}(\theta^i_u) = \sum_{t=1}^{T-1} \sup_{x^i_t} \Big\{ & |x^i_{t+1} - \tilde{x}^i_{t+1}|^2 \\ & - 2(x^i_{t+1} - \tilde{x}^i_{t+1})^\top \Big( e^i(x^i_{t+1}, \tilde{u}^i_{t+1}) - \sum_{j \in \mathscr{V}^i_u} f^{ij}(\tilde{x}^j_t, \tilde{u}^j_t) \Big) \Big\}. \end{aligned}$$

The evaluation of $\hat{J}^i_{ee}$ is not trivial as it involves the calculation of the supremum of a nonlinear multivariate function. In this work, we linearise (6.25) with respect to $x^i_t$

and solve for the supremum of the resulting concave quadratic function, giving:

$$\hat{J}^i_{ee}(\theta^i_u) \approx \bar{J}^i_l(\theta^i_u) = \sum_{t=1}^{T-1} \epsilon^{i\top}_t (E^i_t + E^{i\top}_t - I)^{-1} \epsilon^i_t, \tag{6.26}$$

where $\epsilon^i_t = e^i(x^i_{t+1}, \tilde{u}^i_{t+1}) - \sum_{j \in \mathscr{V}^i_u} f^{ij}(\tilde{x}^j_t, \tilde{u}^j_t)$ is the implicit equation error and $E^i(x^i, u^i) = \partial e^i / \partial x^i$ and $E^i_t = E^i(\tilde{x}^i_t, \tilde{u}^i_t)$. The cost function (6.26) can be optimized via a semidefinite program. Alternative methods for minimizing LREE can also be found in [218].

### 6.4.3 Alternating Directions Method of Multipliers (ADMM)

In Section 6.4.1 we introduced a model set for which the constraints in Theorem 6.1 are downstream separable and in Section 6.4.2 we introduced an upstream separable objective function. Note however, that the constraints and objective are not jointly separable with respect to the same partition. We use ADMM to solve this problem.

We now develop the algorithm for the case where (6.12) is well-posed, monotone and contracting, however, a parallel construction without monotonicity or contraction constraints is straightforward. Consider the following set of parameters

$$\Theta_{m\ell_1} = \{\theta \mid (6.14), (6.17), (6.18), (6.19)\}. \tag{6.27}$$

Applying ADMM as discussed in Section 6.2.3 to the problem $\min_{\theta \in \Theta_{m\ell_1}} \hat{J}_{ee}$ gives the following iteration scheme for iteration $k$:

$$\theta(k+1) = \arg\min_{\theta} \hat{J}_{ee}(\theta) + \frac{\rho}{2} ||\theta - \phi(k) + v(k)||^2, \tag{6.28}$$

$$\phi(k+1) = \arg\min_{\phi \in \Theta_{m\ell_1}} \frac{\rho}{2} ||\theta(k+1) - \phi - v(k)||^2, \tag{6.29}$$

$$v(k+1) = v(k) - \theta(k+1) + \phi(k+1). \tag{6.30}$$

for $\rho > 0$.

When using a GAM structure (6.21), we have the following result:

**Proposition 6.1.** *For the model structure (6.21), the ADMM iteration (6.28) separates into $N$ upstream-separable optimization problems of the form (6.31) and the ADMM iteration (6.29) separates into $N$ downstream-separable optimization problems of the form (6.32).*

*Proof.* See Section 6.8. □

In particular, the ADMM approach corresponds to performing the following iterations locally at each node $i = 1, ..., N$:

$$\theta_u^i(k+1) = \arg\min_{\theta_u^i} \hat{J}_{ee}^i(\theta_u^i) + \frac{\rho}{2}||\theta_u^i - \phi_u^i(k) + v_u^i(k)||^2, \tag{6.31}$$

$$\phi_d^i(k+1) = \arg\min_{\phi_d^i} \mathcal{I}_{\Theta_{m\ell_1}}(\phi_d^i) + \frac{\rho}{2}||\theta_d^i(k+1) - \phi_d^i + v_d^i(k)||^2, \tag{6.32}$$

$$v_u^i(k+1) = v_u^i(k) - \theta_u^i(k+1) + \phi_u^i(k+1). \tag{6.33}$$

The distributed algorithm is listed in Algorithm 6.1. The steps (6.31) and (6.32) require access to the upstream and downstream parameters, respectively. These can be solved by the nodes in the graph, however, communication between both upstream and downstream parameters is necessary between steps. The update (6.33) is trivially separable and can be solved as either an upstream or downstream separable problem.

Termination of ADMM after a finite number of iterations means that in general the two parameter vectors $\theta$ and $\phi$ will disagree. For this reason, we take $\phi$ as the solution to ensure that the well-posedness, monotonicity, and contraction constraints (6.14), (6.17), (6.19) are satisfied.

## 6.5 Discussion

### 6.5.1 Conservatism of the Separable Model Structure

We have proposed searching over the model set (6.21) with $\theta \in \Theta_{m\ell_1}$ (6.27), and it is important to understand which systems may fall into this model set. A particular

**Algorithm 6.1:** Distributed Algorithm

```
Result: φ
Initialize ρ > 0;
Initialize: θ(0), φ(0), v(0);
for k = 1, ... do
    for i = 1, ..., N do
        Get: {x̆_t^i}_{t=1}^T;
        Compute θ_u^i(k + 1) using (6.31);
        Send θ_u^i(k + 1) to upstream neighbours;
        Compute φ_d^i(k + 1) using (6.32);
        Send φ_d^i(k + 1) to downstream neighbours;
        Compute v_u^i using (6.33);
        Send v_u^i(k + 1) to upstream neighbours;
    end
end
```

question of interest is whether there are contracting and monotone systems which *cannot* be represented by this structure, and there are two main reasons why this may occur: the separable structure of the model (6.21), and the assumption of a separable contraction metric in condition (6.19).

An exact characterization of the functions that be approximated via the GAM structure (6.21) is difficult to give, however, they have widely applied in statistical modelling, see [85] for details. Note that while the functions in the implicit system (6.21) are additive, the resulting explicit system (6.8) may not be. For example, the scalar functions $e(x) = \sqrt{x}$ and $f(x, y) = (x + y)$. Both $e$ and $f$ are additive; however, the function $e^{-1} \circ f(x, y) = x^2 + 2xy + y^2$ is not.

Conservatism may also be introduced by the assumption of a separable contraction metric. For the case of linear positive systems, it is has been shown that the existence of a separable Lyapunov functions is both necessary and sufficient [24]. This means that $\Theta_{m\ell_1}$ contains all positive linear systems [217]:

**Theorem 6.2.** *For the system (6.21), if $e$ and $f$ are affine in $(x, u)$, then the model set characterised by (6.14), (6.17) and (6.19) is a parametrization of all stable, discrete-time, positive linear systems.*

*Proof.* See Section 6.8. □

The situation is more complicated for nonlinear monotone systems. Separable contraction metrics have been shown to exist for certain classes of monotone systems [137] and separable weighted $\ell_1$ contraction metrics have been used for the analysis of monotone systems [46, 48]. For incrementally exponentially stable systems, it has been shown that the existence of weighted $\ell_1$ contraction metrics, are necessary and sufficient [106], however the state-dependant weighting depends on the all system states and is therefore not separable in the sense we use. To the authors' knowledge, a complete characterisation of the class of contracting monotone systems that admit separable metrics is still an open problem.

### 6.5.2 Consistency

It has be previously noted that system identification approaches that guarantee stability lead to a bias towards systems that are too stable [131, 117, 138]. Empirical evidence suggests that for methods based on Lagrangian relaxation [219, 218] this bias is smaller.

There are a number of situations that lend themselves towards consistent identification. Firstly, consider the situation where we have noiseless state and input measurements produced by a model with $\theta^* \in \Theta_{m\ell_1}$ such that $J_{ee}(\theta^*) = 0$. Then we also have $\bar{J}_l(\theta)^* = 0$ so the bound is tight and LREE recovers the true minimizer of equation error.

Now, consider the situation where the unconstrained minimizer of equation error (6.2), is a monotone, additive function that is contracting in the identity metric. That is, for the function $a_{\phi^*}(x, u)$ where $\phi^* = \arg\min J_{ee}(\phi)$, the following hold:

1. $a_{\phi^*}(x, u)$ is additive so that (6.8) can be written as $a^i(x, u) = \sum_{j \in \mathcal{V}_u^i} a^{ij}(x^j, u^j)$,
2. $\mathbf{1}^\top(\alpha I - A(x, u)) \geq 0$,
3. $A(x, u) \geq 0$.

where $A = \frac{\partial a}{\partial x}$. Then, optimizing (6.26) returns the same solution as the unconstrained least squares minimizer of $J_{ee}$.

**Proposition 6.2.** *Consider models of the form (6.21) with $e_\theta(x, u) = Ex$ and $f_\theta(x, u) = a_{\phi^*}(x, u)$ for some $\theta$. If properties 1, 2, 3 hold for $a_{\phi^*}(x, u)$ where $\phi^* = \arg\min J_{ee}(\phi)$, then for $\theta^* = \underset{\theta \in \Theta_{m\ell_1}}{\arg\min}\ \bar{J}_l(\theta)$, we have $a_{\phi^*}(x, u) = e_{\theta^*}^{-1} f_{\theta^*}(x, u)$.*

*Proof.* The proof follows the same line of argument as [218, Sec. IV Proposition 1]. □

### 6.5.3 Iteration Complexity of Distributed Algorithm

In this section, we investigate the computational complexity of each step in the distributed algorithm. In general, the complexity depends on the model parametrization used, however, we limit our discussion to the case where the models are parametrized by polynomials and the constraints are enforced using sum-of-squares programming.

The first step, (6.31), is a semi-definite program and can be solved using standard solvers. If no structural properties are exploited, a primal-dual interior point method (IPM), would require $\mathcal{O}\left[\max\{n_{\theta_u^i}^3,\ n_{\theta_u^i}{n_i}^3,\ n_{\theta_u^i}^2 n_i^2\}\right]$ operations per iteration per node [126], where $n_{\theta_u^i}$ is the number of upstream free parameters .

The second step, (6.32), is a sum-of-squares problem that can solved as a semidefinite program. If $e$ and $f$ both have degree $2d$, then the size of Gram matrix corresponding to (6.19) for the additive model (6.21) is $p = 1 + \sum_{j\in\mathscr{V}_d^i}\left[\binom{n_j+m_i+d}{d} - 1\right]$. Solving (6.32) using a primal-dual IPM requires approximately $\mathcal{O}\left[\max\{n_{\theta_d^i}^3,\ n_{\theta_d^i}p^3,\ n_{\theta_d^i}^2 p^2\}\right]$ operations per iteration per node [126], where $n_{\theta_d^i}$ is the number of downstream free parameters.

In a distributed computing setting, if a computational resource is associated with each node in the network, and the network graph has bounded degree (neighbours per node), then the iteration complexity is independent of the number of nodes. However, the computational complexity will grow with the number neighbours per node, the size of the local states, and the degrees of the polynomials used in the model.

### 6.5.4 Other Quality of Fit Criteria

Lagrangian relaxation of least-squares equation error was chosen as it is convex, upstream separable, quick to compute, and leads to a simple implementation of ADMM. Any method that treats neighbouring states as exogenous inputs will be upstream separable. However, any such approach will also be susceptible to instability due to the introduction of new feedback loops via the network topology, even if it guarantees

stability of the local models. Consequently, one can similarly apply any convex quality of fit criteria such us convex upper bounds on simulation error [214, 219] and still guarantee convergence of ADMM. Alternatively, a non-convex quality of fit criteria like simulation error can be used at the expense of ADMM's convergence guarantees.

We note that the conditions proposed in Section 6.3 can be used for non-separable models with centralized algorithms to ensure stability and/or monotonicity. Joint convexity of the model set and stability constraints is still an important as it simplifies constrained optimization allowing for the easy application of penalty, barrier or projected gradient methods as in chapter 3.

## 6.6 Numerical Experiments

In this section we present numerical results exploring the scalability and identification performance the proposed approach.

This section is structured as follows: first, we look at the identification of positive linear systems, and explore the computational complexity of the $\ell_1$ and $\ell_2$ contraction conditions; we then explore the consistency of fitting nonlinear models when the true system lies in the model set, essentially analysing the effect of convex bound on equation error; finally, we apply the method to the identification of a (simulated) nonlinear traffic network. The traffic network does not lie in the model set so only an approximate model can be identified. We explore the regularising effect of the model constraints and scalability of the method to large networks.

Previous methods for the identification of models with stability guarantees have ensured contraction using a quadratic metric [214, 219, 218]. Contraction is implied by the following semidefinite constraint:

$$W(x,u,\theta) \succeq 0 \quad \forall (x,u), \tag{6.34}$$

$$W(x,u,\theta) = \begin{bmatrix} E(x,u) + E(x,u)^\top - P - \eta I & F(x,u)^\top \\ F(x,u) & P \end{bmatrix}$$

where $P \in \mathbb{S}^{n\times n}, P \succ 0$, $\eta > 0$. We refer to (6.34) as an $\ell_2$ contraction condition as it implies the contraction conditions (6.6) with a state dependent weighted $\ell_2$ norm of the differentials $V = \delta_{x_t}^\top E(x_t, u_t)^\top P^{-1} E(x_t, u_t)\delta_{x_t}$.

We will make future reference to the following convex sets of parameters, in addition to $\theta_{ml_1}$ defined in (6.27):

$$\Theta_u = \{\theta \mid (6.14), (6.18)\}, \quad \Theta_m = \{\theta \mid (6.14), (6.17), (6.18)\}$$
$$\Theta_{m\ell_2} = \{\theta \mid (6.17), (6.18), (6.34)\}$$

Here the subscripts refer to the following properties:

- $m\ell_1$ - Monotone $\ell_1$ contracting models i.e. $\theta \in \Theta_{m\ell_1}$,
- $m$ - Monotone models i.e. $\theta \in \Theta_m$,
- $u$ - Models that are not constrained to be contracting or monotone i.e. $\theta \in \Theta_u$,
- $m\ell_2$ - Models that are monotone and contracting in $\ell_2$, i.e. $\theta \in \Theta_{m\ell_2}$,

All functions $e^i$, and $f^{ij}$ are polynomials in all monomials of their arguments up to a certain degree.

As a baseline for comparison, we will also compare to models denoted $Poly$, with explicit polyonomial models (6.1) fit by least-squares without any separable structure imposed. We will also compare to standard wavelet and sigmoid Nonlinear AutoRegressive with Exogenous input (NARX) models implemented as part of the Matlab system identification toolbox.

For the implicit models, the model class prefix is followed by the degrees of the polynomials in $e$ and $f$ in parenthesis. For example, the notation $u(3, 5)$ refers to unconstrained models with $e$ having degree 3 and $f$ having degree 5. For the explicit polynomial models $Poly$, the degree used follows in parenthesis, so $Poly(5)$ are explicit polynomial models of degree 5 in all arguments.

The NARX models were fit at each node using the regressors $(\breve{x}_t^i, \breve{u}_t^i, \breve{u}_{t+1}^i)$. The wavelet NARX models were set to automatically choose the number of basis functions and the sigmoid NARX models were set to use 10 basis functions. The focus for each model was set to produce the best performance. For the wavelet network, we used a focus on simulation and for the sigmoid network, we used a focus on prediction.

The constraints (6.14), (6.17), (6.18), (6.19) and (6.34) are enforced using sum of squares programming [162]. All programs are solved using the SDP solver MOSEK with the parser YALMIP [128] on a standard desktop computer (intel core i7, 16GB RAM).

### 6.6.1 Identification of Linear Positive Systems

In this subsection we study the scalability of the proposed method for the identification of linear positive systems.

We compare the computation time using the proposed $\ell_1$ contraction constraint to a previously proposed $\ell_2$ contraction constraint (i.e. quadratic Lyapunov function). Note that for linear systems, the model sets $m\ell_1$ and $m\ell_2$ both are parameterizations of all stable positive linear systems so no difference in quality of fit is observed.

#### Scalability of Separable Linear and Quadratic Metrics

We illustrate the difference in scalability between the models $m\ell_1(1,1)$ and $m\ell_2(1,1)$. Each experimental trial consists of the following steps:

1. A stable positive system with state dimension $n_x$ is randomly generated using Matlab's `rand` function; $A \in \mathbb{R}^{n_x \times n_x}$ has a banded structure with band width equal to 9. Stability was ensured by rescaling $A$ to have a spectral radius of 0.95.

2. The system is simulated for $T = 10^4$ time steps; $\tilde{x}_{1:T}$ is obtained by adding white noise to the simulated states at SNR equal to 40dB.

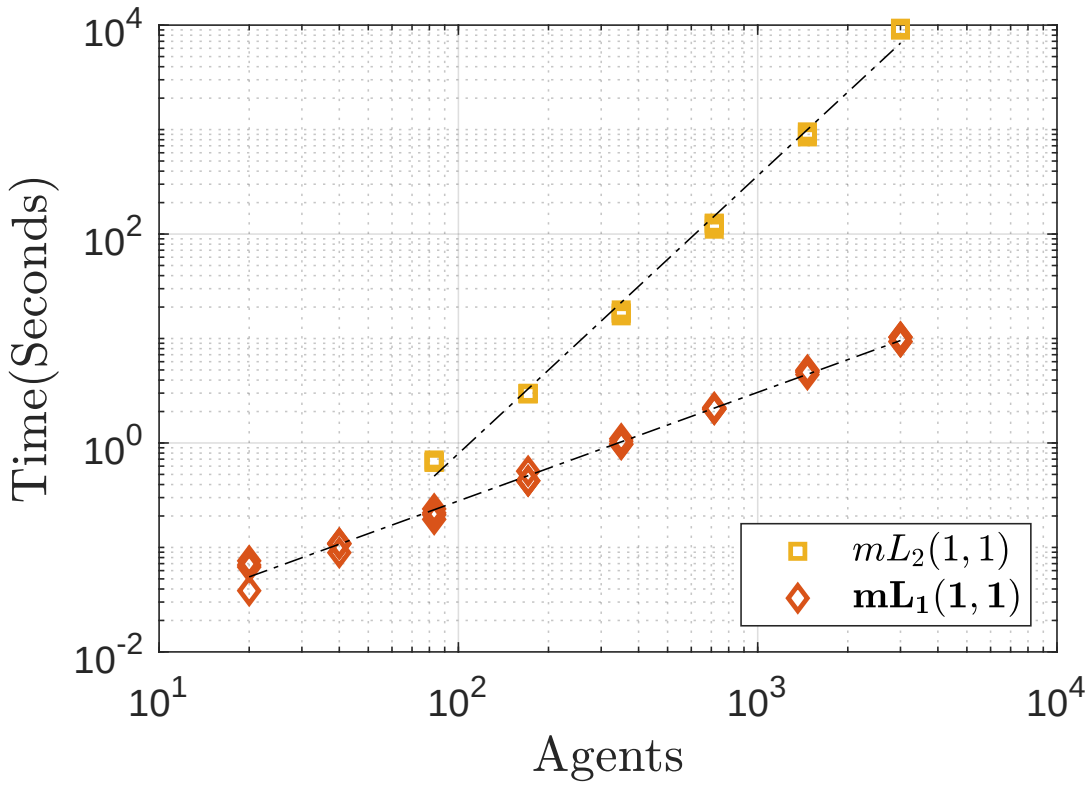



**Figure 6.2** – Computation time as function of system size. The slopes of the lines of best fit are: $m\ell_2(1,1)$ - 2.66, $m\ell_1(1,1)$ - 1.04 .

3. This process is repeated 5 times for each $n_x$.

The time taken to solve each optimization problem is shown in Fig. 6.2. Here, we see a significant improvement in the computational complexity from approximately cubic growth for $m\ell_2$ to linear growth for $m\ell_1$. The networked approach allows us to solve stable identification problems with at least 3000 states.

Note that no explicit attempts to exploit the sparsity of the system were made; use of solvers and parsers designed to exploit sparsity could improve performance, especially for the SDPs associated with the LMI parametrization, e.g. [7].

### 6.6.2 Identification of Nonlinear Models

In this section we study the consistency of fitting nonlinear implicit models via the LREE bound on equation error. In Section 6.5.2 we saw that in the noiseless case, optimization of LREE will return the true model parameters. We will now explore the effect of introducing noise on the model estimates. The experiments in this section can be seen to supplement those in [218, Sec. IV] which studied the effects of noise and model stability on consistency in the linear setting.

We generate models $a^*(x, u)$ by sampling a parameter vector $\theta$ and then projecting onto the set $\Theta_{m\ell_1}$. The models have degree 3, state size $n = 2$ and $m = 1$. We then generate training data with $T$ samples by randomly sampling $(\tilde{x}_t, \tilde{u}_t)$ from the uniform distribution on $[0, 1]$ and generated noisy measurements of $x_{t+1}$ by $\tilde{x}_{t+1} = a^*(\tilde{x}_t, \tilde{u}_t) + v_t$, where $v_t$ is normally distributed noise with a specified Signal to Noise Ratio (SNR). Models $a(x, u)$ are then trained by minimizing $\bar{J}_l$ with $\theta \in \Theta_{m\ell_1}$ and performance measured using Normalized Equation Error (NEE):

$$\text{NEE} = \frac{|a(x, u) - a^*(x, u)|_2^2}{|a^*(x, u)|_2^2} \tag{6.35}$$

where $a(x, u)$ is the identified dynamic model and $a^*(x, u)$ is the true where $|f(x)|_2 = \int_{x \in \mathcal{D}} |f(x)|^2 dx$ is the sample estimate of the 2-norm of the function $f$.

In Figure 6.3, we have plotted the NEE that results from fitting models from $m\ell_1(3, 3)$ by optimizing LREE (6.26) and implicit equation error (6.22). We can see that LREE provides a much better fit than implicit equation error, especially as the number of data points increases.

To explore the effect of noise on the consistency of LREE, we have plotted NEE versus the size of the dataset for varying noise level (measured in decibels) in Figure 6.4. If we had a consistent estimator of the explicit model (6.1), we would expect to see $\lim_{T \to \infty} NEE = 0$ with consistent slope for all SNR levels. What we in fact observe, however, is that in noisier conditions the NEE initially decreases and then plateaus at a certain level.

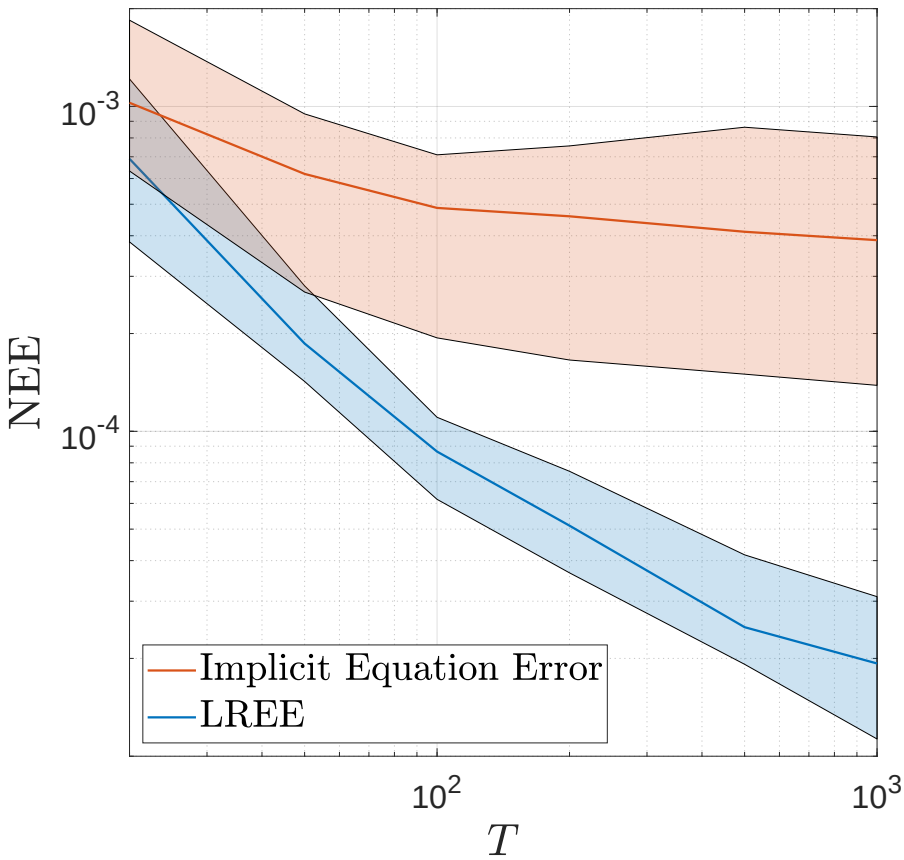



**Figure 6.3** – Comparison of implicit equation error and LREE: Normalized equation error versus number of training data points. The training data has gaussian noise with $SNR = 30$dB. For each method, the central line shows the median NEE for 50 model realizations and the shaded region shows the upper and lower quartiles.

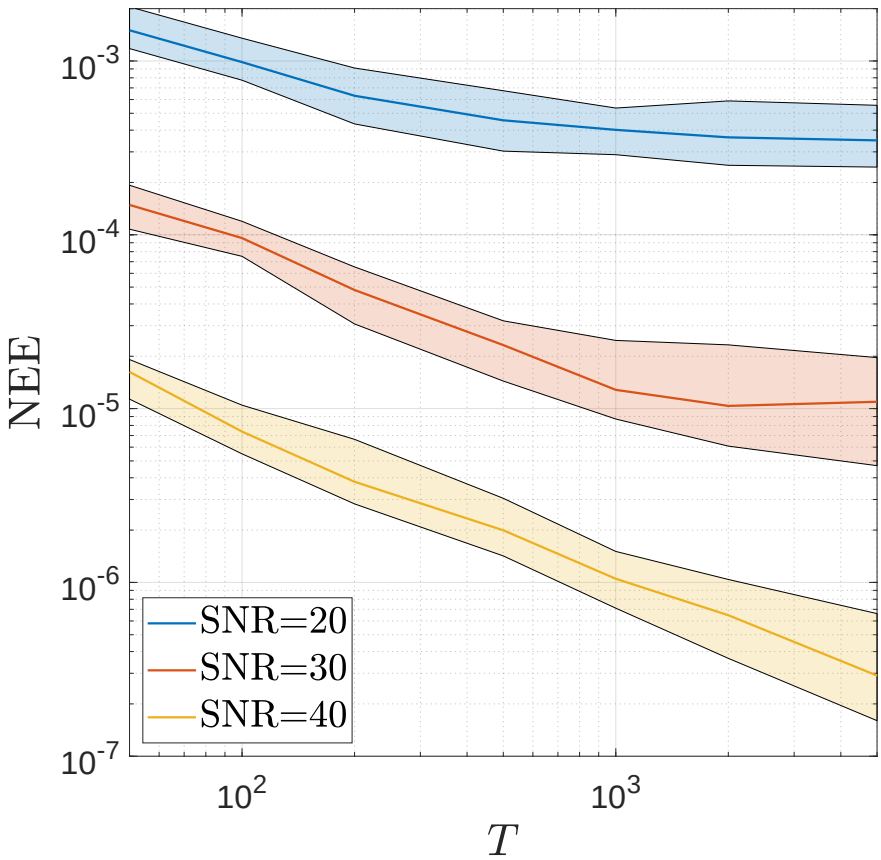



**Figure 6.4** – Normalized equation error versus number of training data points for three different SNRs. The central line shows the median NEE for 25 model realisations and the shaded region shows the upper and lower quartiles. The SNR is measured in decibels.

This phenomena can also be seen in [218, Sec. IV], where LREE produces models biased towards being too stable, even in the infinite data limit.

### 6.6.3 Identification of Traffic Networks

In this section we examine a potential application of our approach, the identification of a traffic network. The dynamics of traffic networks are thought to be monotone when operating in the free flow regime [130]. Note that monotonicity of some traffic models is lost when certain nodes are congested [47].

The data are generated using the model in [130], which is not in the proposed model set. Hence this section provides a test of robustness of the proposed approach to modelling assumptions.

For this application, we consider using equation error as a surrogate for simulation error. Model performance is therefore measured using Normalized Simulation Error (NSE):

$$\text{NSE} = \frac{\sum_t |x_t - \tilde{x}_t|^2}{\sum_t |\tilde{x}_t|^2}, \tag{6.36}$$

where $x_t$ are the simulated states.

We will first introduce the model, then study the effect of the model constraints by comparison to existing methods, and finally examine scalability to large networks.

#### Simulation of a traffic network

The dynamics are simulated over a graph (e.g. Fig. 6.6), where, each node $i$ represents a road with state corresponding to the density of traffic on the road, denoted $\rho^i$. Nodes marked *in* allow cars to flow into the network, and nodes marked *out* allow cars to flow out of the network. Each edge $(i, j)$ is randomly assigned a turning preference denoted $R_{ij}$ such that $\sum_i R_{ij} = 1$ (this ensures that the total number of cars at each intersection is conserved). Each node $i$ has a capacity of $C_i = 1$. Vehicles transfer

from roads $i$ to $j$ according to the routing policy,

$$f_{i\to j}(\rho) = R_{ji} d_i(\rho^i) \min\left\{1, \frac{s_j(\rho_j)}{\sum_{k\in\mathscr{V}_u^i} R_{kj} d_k(\rho^j)}\right\},$$

where $d_i(\rho) = \min(10, \rho)$ and $s_i(\rho) = \max(2C_i - \rho, 0)$ are monotone demand and supply curves for road $i$. The dynamics of the complete system are then found to be

$$\dot{\rho}^i = f_{in}^i - f_{out}^i, \tag{6.37}$$

where

$$f_{in}^i = \begin{cases} u^i & , \ i \in \text{in} \\ \sum_{j\in\mathscr{V}_u^i} f_{j\to i} & , \ i \notin \text{in} \end{cases}$$

$$f_{out}^i = \begin{cases} d_i(\rho_i) & , \ i \in \text{out} \\ \sum_{j\in\mathscr{V}_d^i} f_{i\to j} & , \ i \notin \text{out}. \end{cases}$$

The input nodes $i \in in$ take a time varying input $u^i$. We use the following method to generate data sets of size $T$:

(i) First, we generate an input signal for each $u^i$ of size $T$. This signal changes value every 5 seconds to a new value that is normally distributed with mean $\mu_u$ and standard deviation $\sigma_u$. Negative values of $u$ are set to zero. An example input signal is shown in Fig. 6.5.

(ii) The dynamics (6.37) are integrated over $t_f$ seconds.

(iii) A training set of size $T = 2t_f$ is generated by sampling every 0.5 seconds.

### Regularization Effect of Model Constraints

In this section we will explore the effects of introducing monotonicity, positivity, and contraction constraints.

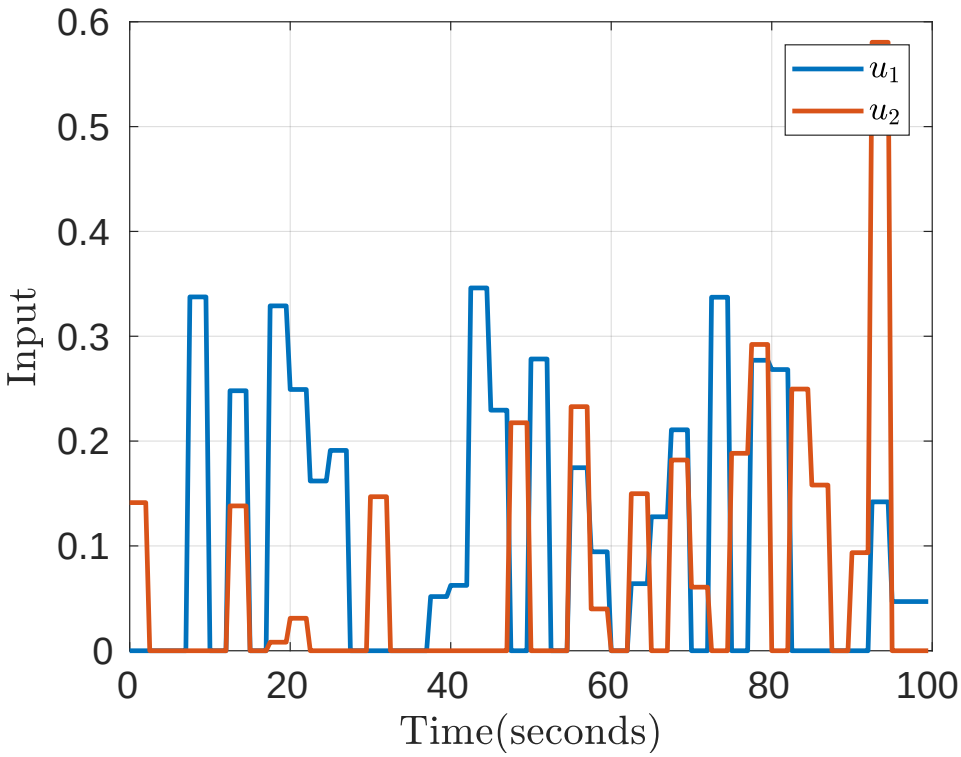



**Figure 6.5** – Example input signal to network ($\mu_u = 0, \sigma_u = 0.2$).

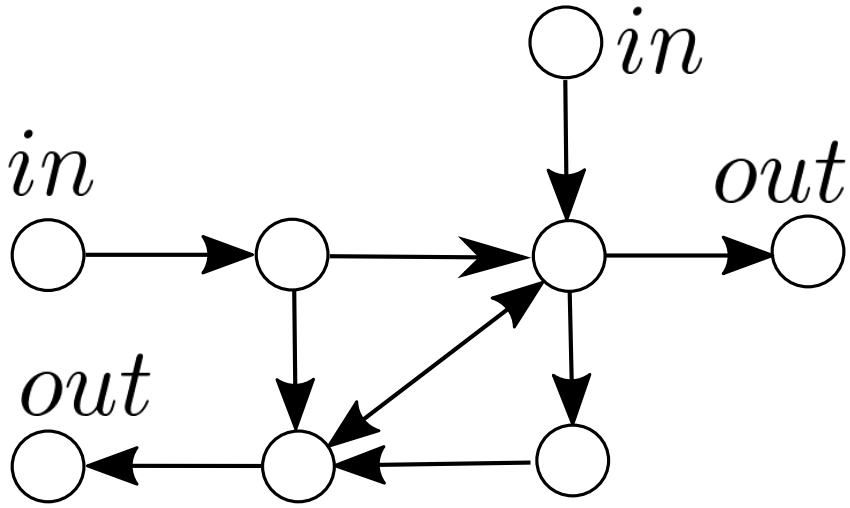


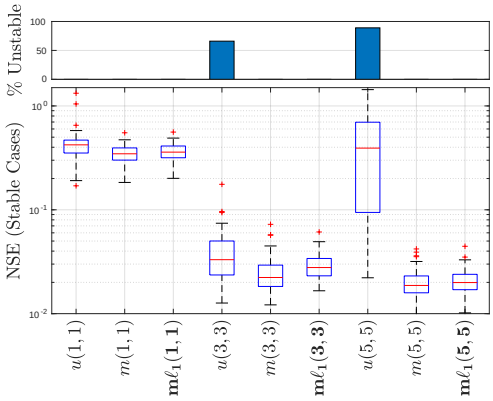

**Figure 6.6** – A small traffic network. Each node represents a road and each link represents an intersection.

Introducing model constraints limits the expressivity of our model. Consequently, one might expect the estimator bias to increase and the variance to decrease [72, Chapter 7]. Empirical evidence in this section suggests that a judicious choice of constraints can reduce the variance with a minimal increase in bias.

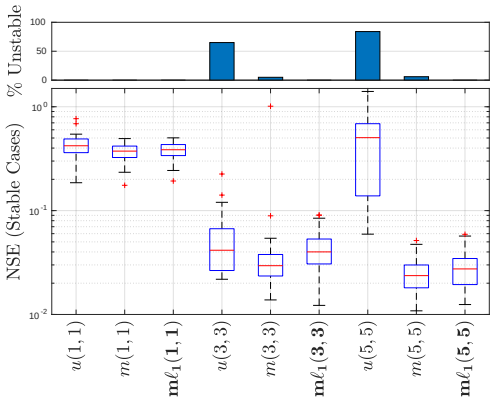
Using the method outlined in Section 6.6.3 for simulating a traffic network and the graph depicted in Fig. 6.6, we generate 100 different training sets of size $T = 1000$ with $\mu_u = 0, \sigma_u = 0.2$ and then compare the results on three different validation sets. The first validation set has inputs generated with parameters $\mu_u = 0, \sigma_u = 0.2$ (the same as the training set). The second and third validation sets have parameters $\mu_u = 0, \sigma_u = 0.3$ and $\mu_u = 0, \sigma_u = 0.4$ respectively. These are used to test the generalizability of our model to inputs outside the training set.

In figures 6.7 and 6.8, we have plotted the $NSE$ on both the training set and validation sets 1 and 2 for our proposed model sets, the polynomial model, the NARX models and the model set $m\ell_2$. The percentage of total models that displayed instability is indicated in both the bar graph in the upper portion of the figures.

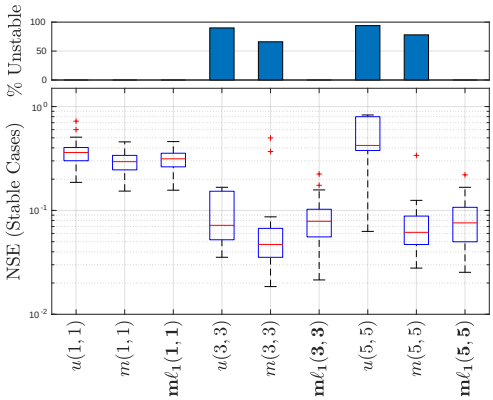
In all cases, the identified linear models performed poorly. This is unsurprising as the true system is highly non-linear.




**(a)** Training set ($\sigma_u = 0.2, \mu_u = 0$) over 100 realizations.

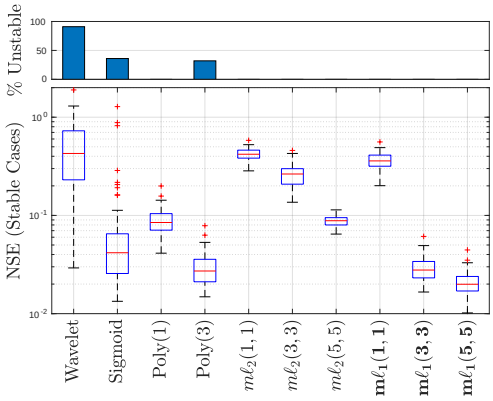



**(b)** Validation set 1 ($\sigma_u = 0.2, \mu_u = 0$) over 100 realizations.

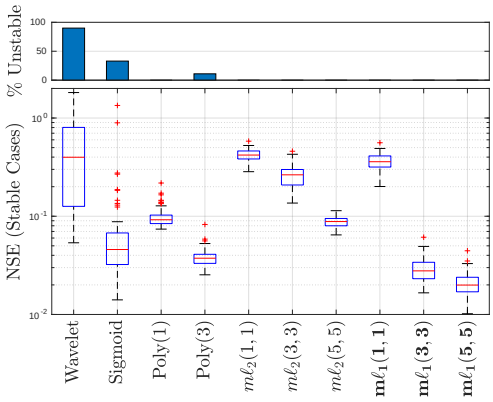



**(c)** Validation set 2 ($\sigma_u = 0.3, \mu_u = 0$) over 100 realizations.

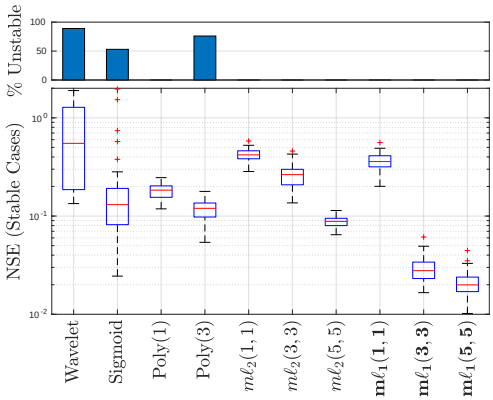
**Figure 6.7** – Box plots showing normalized simulation error for 100 model realizations for different behavioural constraints. The bar graph shows the percentage of models that displayed instability.




**(a)** Training set ($\sigma_u = 0.2, \mu_u = 0$) over 100 realizations.




**(b)** Validation set 1 ($\sigma_u = 0.2, \mu_u = 0$) over 100 realizations.




**(c)** Validation set 2 ($\sigma_u = 0.3, \mu_u = 0$) over 100 realizations.

**Figure 6.8** – Box plots showing normalized simulation error for 100 model realizations for different model structures. The bar graph shows the percentage of models that displayed instability.

| | $\boldsymbol{m\ell_1(1,1)}$ | $m(1,1)$ | $u(1,1)$ | $\boldsymbol{m\ell_1(3,3)}$ | $m(3,3)$ | $u(3,3)$ | $\boldsymbol{m\ell_1(5,5)}$ | $m(5,5)$ | $u(5,5)$ | Wavelet | Sigmoid |
|---|---|---|---|---|---|---|---|---|---|---|---|
| train. ($\sigma_u = 0.2$) | 0% | 0% | 2% | 0% | 0% | 66% | 0% | 0% | 89% | 88% | 36% |
| val. 1 ($\sigma_u = 0.2$) | 0% | 0% | 0% | 0% | 5% | 65% | 0% | 6% | 84% | 87% | 31% |
| val. 2 ($\sigma_u = 0.3$) | 0% | 0% | 0% | 0% | 64% | 90% | 0% | 78% | 94% | 88% | 51% |
| val. 3 ($\sigma_u = 0.4$) | 0% | 0% | 0% | 0% | 88% | 95% | 0% | 97% | 89% | 88% | 63% |

**Table 6.1** – Percentage of unstable models that diverged on training and validation data. In each case the input $u$ has $\mu_u = 0$.

Comparing the models $m\ell_1$ and $m\ell_2$ with the remaining models, we can see that the stability constraints have a regularizing effect where increasing the degree of the polynomials reduces the median NSE; in other words, increasing model complexity improves model fidelity. The other models on the other hand perform worse with increasing the complexity. This is most clearly seen in the models $u$, where increase the polynomial degree results in poorer fits on validation data.

Our results also suggest that model stability constraints significantly improve robustness. Without stability constraints, a model that appears stable during training may turn out to be unstable under a slight shift in the input data distribution. This can be seen most clearly in the models $m(5,5)$ and $Poly(3)$, where on the training data distribution, most models are stable. However, increasing the variance of the inputs to the network results a large number of unstable models with unbounded NSE, c.f. Fig. 6.7c and Fig. 6.8c. Further evidence is shown in Table 6.1, where we can see that once the variance of the input data doubles, almost all models that do not have stability constraints are unstable.

To compare to a standard approach, we also compare to wavelet and sigmoid NARX models fit using the Matlab system identification tool box. The resulting $NSE$ is shown in Fig. 6.8 and show the number of models producing unstable models and negative state estimates in tables 6.1 and 6.2 respectively. While we observed extremely high performance of the individually identified sub-systems, simulating the network interconnection of those sub-systems produces many unstable models, many negative state estimates and poor quality of fit.

For positive linear systems, both $\Theta_{m\ell_1}$ and $\Theta_{m\ell_2}$ are parameterizations of the same set of models. This is not the case for nonlinear monotone systems and the choice of parametrization impacts the resulting model performance. This can be seen in

| | $\boldsymbol{m\ell_1(1,1)}$ | $m(1,1)$ | $u(1,1)$ | $\boldsymbol{m\ell_1(3,3)}$ | $m(3,3)$ | $u(3,3)$ | $\boldsymbol{m\ell_1(5,5)}$ | $m(5,5)$ | $u(5,5)$ | Wavelet | Sigmoid |
|---|---|---|---|---|---|---|---|---|---|---|---|
| train. ($\sigma_u = 0.2$) | 0% | 0% | 2% | 0% | 0% | 66% | 0% | 0% | 89% | 100% | 91% |
| val. 1 ($\sigma_u = 0.2$) | 0% | 0% | 0% | 0% | 0% | 65% | 0% | 0% | 84% | 100 % | 93% |
| val. 2 ($\sigma_u = 0.3$) | 0% | 0% | 2% | 0% | 0% | 90% | 0% | 0% | 94% | 100 % | 99% |
| val. 3 ($\sigma_u = 0.4$) | 0% | 0% | 1% | 0% | 0% | 95% | 0% | 0% | 99% | 100 % | 100% |

**Table 6.2** – Percentage of total models that predicted negative states. In each case the input $u$ has $\mu_u = 0$.

Fig. 6.8a, Fig. 6.8b and Fig. 6.8c where the models fit using our proposed $\ell_1$ contraction constraint outperform those fit using the previously-proposed $\ell_2$ contraction constraint.

Finally, looking at Table 6.2, we can see that when models were not constrained to be positive $u$ and $Poly$, a large number of models producing negative state estimates were identified. This can lead to non-sensical results in many applications, and prevents the application of synthesis methods that depend on monotonicity.

### Scalability Comparison of $\ell_1$ and $\ell_2$ contraction

We now explore the scalability of the $\ell_1$ and $\ell_2$ contraction constraints for nonlinear models.

We construct traffic networks consisting of $N = P + 2M$ nodes by placing $P$ points randomly in a unit square and triangulating. $M$ in nodes and $M$ out nodes are then randomly assigned throughout the network. We generate training data using the method described in Section 6.6 with $T = 600, \mu_u = 0, \sigma_u = 0.4$ and a corresponding validation set. We then fit models $m\ell_1(3,3)$ and $m\ell_2(3,3)$ using an interior point method. This is repeated 5 times for a varying number of nodes.

Figure 6.9 shows a plot of the time taken to solve each problem versus the total number of nodes. We observe that fitting models with an $\ell_2$ contraction constraint has a complexity $\mathcal{O}[N^3]$ in the number of nodes while models using the $\ell_1$ contraction constraint have a complexity of $\mathcal{O}[N^{1.5}]$ in the number of nodes. The improved complexity of the $\ell_1$ constraint is a result of its separable structure.

The validation NSE versus the number of agents is shown in Fig. 6.10 for the model set in $m\ell_1(3,3)$. We observe no deterioration of model performance as the number

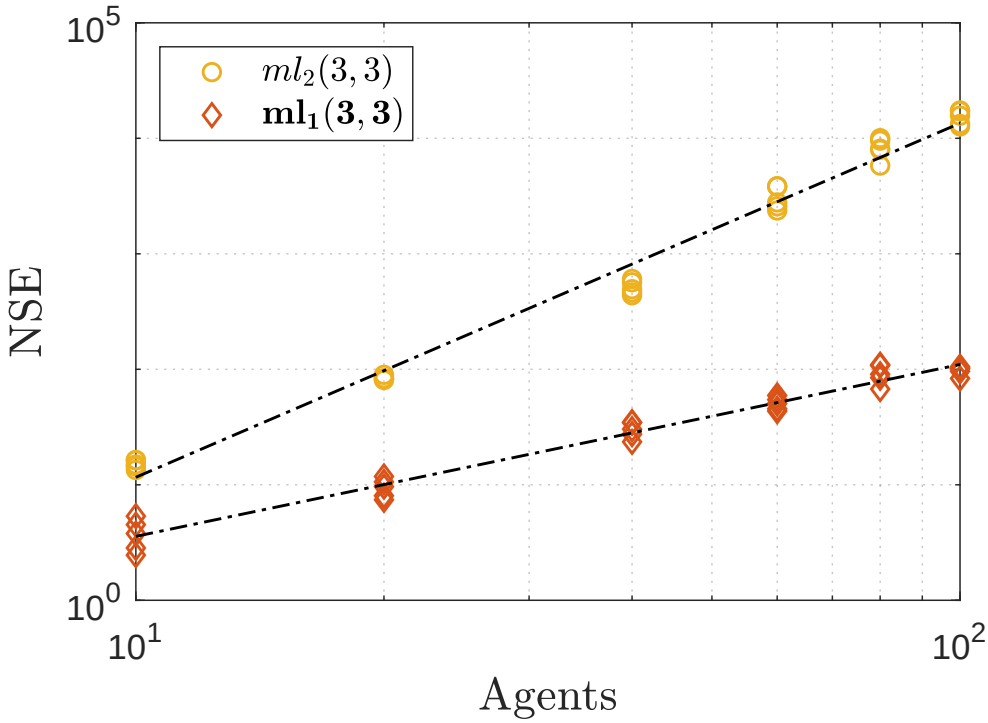



**Figure 6.9** – Computation time for models $m\ell_2(3,3)$ and $m\ell_1(3,3)$ for a varying system size. The slopes of the lines are 3.06 and 1.49 respectively.

of agents increases, suggesting that our method can be effective when scaled to large networks.

### Scalability Compared to Interior Point Methods

We conclude our numerical experiments with a comparison of the computational complexity of the proposed distributed algorithm to centralized optimization via standard interior point methods.

We introduce additional notation to distinguish between the centralized and distributed algorithms. We will use a subscript $C$ to refer to models fit using the off-the-shelf interior point method. The subscript $D$ is used to denote models fit using ADMM. For example, $m\ell_1(3,3)_D$ is the problem of fitting the model $m\ell_1(3,3)$ solved using the distributed algorithm.

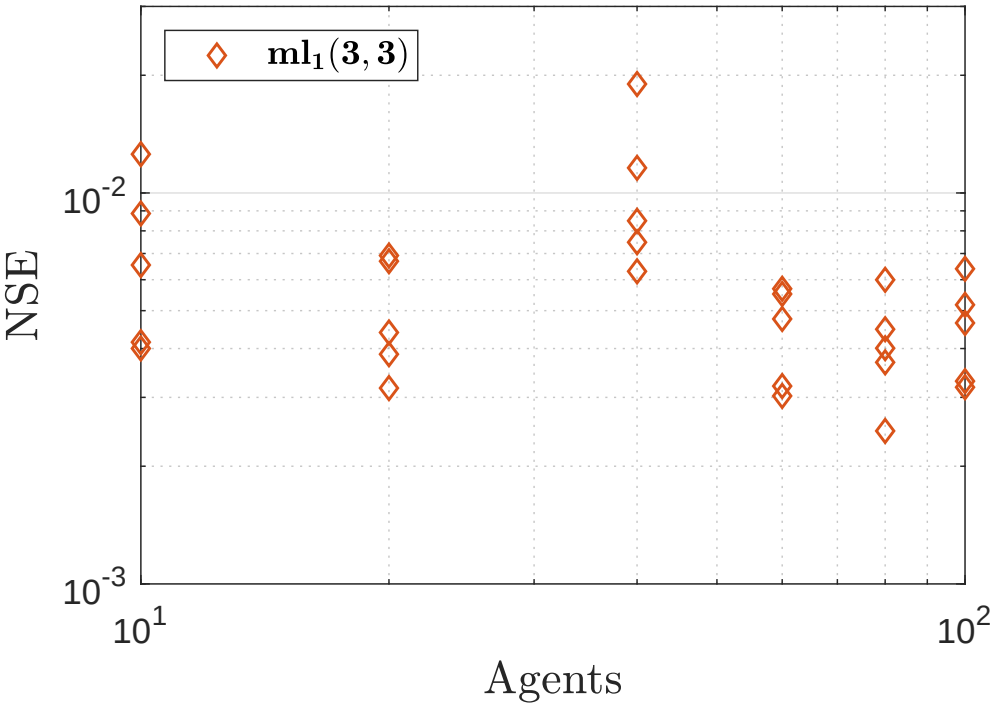



**Figure 6.10** – NSE for models $m\ell_1(3,3)$ for varying system size.

To control for the number of neighbors of each node, we generate random, connected, regular graphs of size $N$ and degree 4 and randomly assign $\frac{P}{2}$ *in* nodes and $\frac{P}{2}$ *out* nodes. Training data is generated according to Section 6.6.3 with $T = 500$ and $\sigma_u = 0.2$.

We then solve the problems $m\ell_1(3,3)_C$ and $m\ell_1(3,3)_D$ using the stopping criteria from [31, Section 3.3] ($\epsilon_{abs} = 10^{-4}$, $\epsilon_{rel} = 10^{-3}$).

The results are displayed in Fig. 6.11. The line $m\ell_1(3,3)_{D-serial}$ indicates the total time taken to fit a model using ADMM, where the sub-problems (6.31), (6.32) are solved without parallelization (consecutively, on a single computer). Additionally, we calculate the total time that would be taken if the computation had been distributed among $N$ nodes, indicated by the line $m\ell_1(3,3)_{D-parallel}$.

While the program $m\ell_1(3,3)_{D-serial}$ takes longer on the selected problems than $m\ell_1(3,3)_C$, it has superior scalability with $\mathcal{O}[N^{1.05}]$ compared to $\mathcal{O}[N^{1.36}]$, suggesting that for a larger number of nodes, it will be faster.

Of more interest is $m\ell_1(3,3)_{D-parallel}$ with an observed complexity of $\mathcal{O}[N^{0.05}]$ in the number of nodes. This suggests that if the computation is distributed, the problem can be solved in near constant time. It is important to note, however, that this does not take into account many of the complexities of distributed computing, for example the overhead associated with communication between nodes.

## 6.7 Conclusion

In this chapter we have proposed a model set for system identification that allows model behavioral guarantees such as stability (contraction), monotonicity, and positivity. Furthermore, we have introduced a particular separable structure that allows distributed identification and scalability to large networked systems via local node-to-node communication.

We have examined the proposed approach via a selection of numerical case studies including a nonlinear traffic network. The main conclusions are that the approach

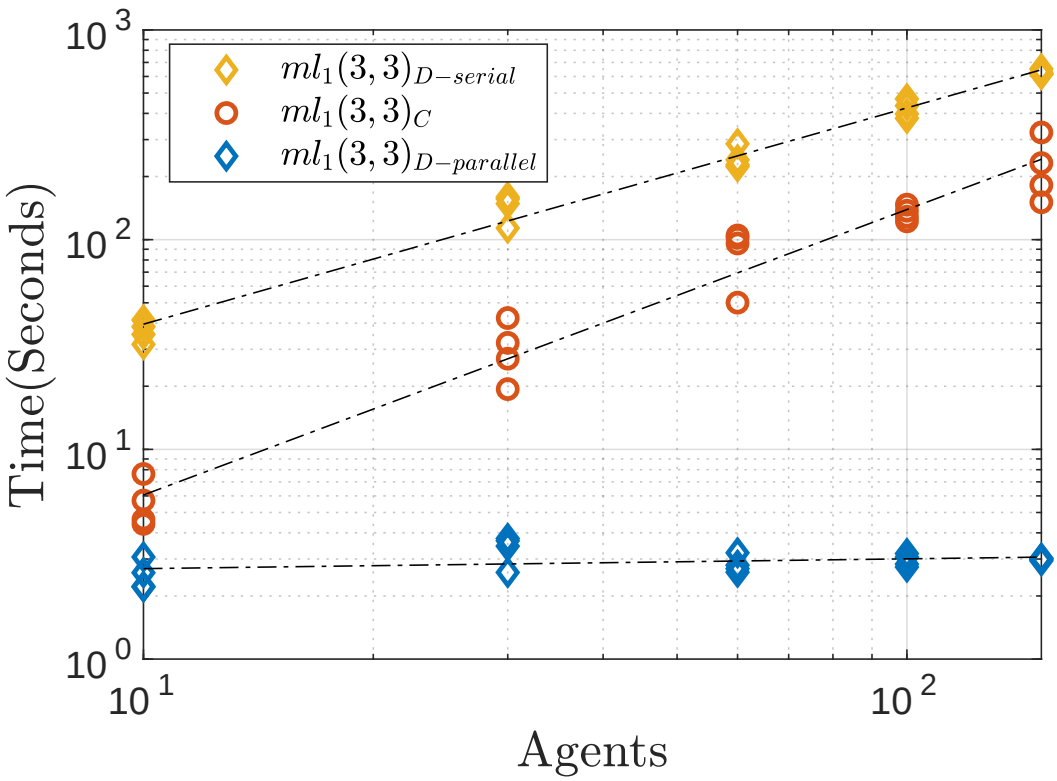



**Figure 6.11** – Runtime of ADMM compared to IPM where the number of threads is one or equal to the number of nodes. When calculating the results for "simulated" distributed computing, ADMM is run in series and time per iteration is taken to be the sum of the maximum times to solve each step. The slopes of the lines are 1.05, 1.36 and 0.047 respectively.

scales much better than previous approaches guaranteeing stability, and that behavioral constraints such as stability and monotonicity can have a regularizing effect that leads to superior model predictions.

## 6.8 Proofs

### Proof of Theorem 6.1

We use the following lemma in the proof of Theorem 6.1:

**Lemma 6.1.** Suppose that for the system (6.21), there exists a weighted differential $\ell_1$ storage function $V_t = |E(x_t, u_t)\delta_t|_1$, where $E : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{M}^n$ such that $V_{t+1} \leq \alpha V_t$ and there exists some $K \succ 0$ such that $|\delta_t|_1 \prec K|E_t\delta_t|_1$, then the system is contracting in the sense of definition 2.1.

*Proof.* Consider the family of solutions to (6.21), parametrized by $\rho \in [0, 1]$, having initial conditions $\rho x_1(0) + (1 - \rho)x_2(0)$ and input $u(t)$, denoted $x_\rho(t)$.

Define $\delta_\rho(t) = \frac{\partial x_\rho(t)}{\partial \rho}$. Now, consider:

$$\begin{aligned} |x_1(t) - x_2(t)|_1 &= \left| \int_0^1 \delta_\rho(t) d\rho \right|_1 \\ &\leq \int_0^1 |\delta_\rho(t)|_1 d\rho \\ &\leq \int_0^1 K|E_t\delta_\rho(t)|_1 d\rho \end{aligned}$$

By assumption, $V_{t+1} \leq \alpha V_t$ which means that $|E_t\delta(t)|_1 \leq \alpha|E_{t-1}\delta_{t-1}|_1$. This inequality can be applied repeatedly to give:

$$|x_1(t) - x_2(t)|_1 \leq K\alpha^t \int_0^1 |E_0\delta_\rho(0)|_1 d\rho$$

Taking $b(x_1(0), x_2(0)) = K \int_0^1 |E_0\delta_\rho(0)|_1 d\rho$ gives Definition 2.1. □

*Proof of Theorem 6.1.* First we will show well-posedness and monotonicity. We will then prove stability of monotone contracting systems and finally just contracting systems. For brevity of the equations, we will use a subscript $t$ to refer to the evaluation of a function at a specific time, so $E_t = E(x_t, u_t)$.

*Well-posedness:* Assume (6.14). Since $E$ is a non-singular M matrix, there exists a diagonal matrix $D$ such that $ED + DE^\top \succ 0$. Well posedness follows from the same argument as [214, Theorem 5].

*Monotonicity:* Assume (6.17). Since $E$ as an M-matrix, it is inverse positive and $E^{-1}F \geq 0$. The differential dynamics of the explicit system (6.8) can be written as $\delta_{x_{t+1}} = E_{t+1}^{-1} F \delta_{x_t}$. Therefore, the explicit system is monotone.

*Contraction:* Assume conditions (6.15) and (6.16). Condition (6.15) implies that

$$|F(x,u)| \leq S(x,u). \tag{6.38}$$

Condition (6.16) then implies,

$$\begin{aligned}
&\mathbf{1}^\top(\alpha E(x,u) - S(x,u)) \geq 0, && (6.39)\\
\implies &\mathbf{1}^\top(\alpha E(x,u) - |F(x,u)|) \geq 0, && (6.40)\\
\implies &\mathbf{1}^\top(\alpha - |F(x,u)|E^{-1}(x,u)) \geq 0, && (6.41)\\
\implies &\mathbf{1}^\top(\alpha - |F(x,u)E^{-1}(x,u)|) \geq 0, && (6.42)\\
\implies &(\alpha - ||F(x,u)E^{-1}(x,u)||_1) \geq 0, && (6.43)
\end{aligned}$$

where $||\cdot||_1$ is the induced matrix norm , $||M|| := \max_j \sum_i M^{ij}$. Stability follows from the same argument as in the proof of Theorem 6.1. Multiply by $|E_t\delta_t|_1$, we get:

$$\begin{aligned}
&\left(\alpha - ||F(x,u)E^{-1}(x,u)||_1)\right)|E_t\delta_t|_1 \geq 0, && (6.44)\\
\implies &\alpha|E_t\delta_t|_1 - |F(x,u)E_t^{-1}E_t\delta_t|_1) \geq 0, && (6.45)\\
\implies &|F_t\delta_t|_1 - \alpha|E_t\delta_t|_1 \leq 0, && (6.46)\\
\implies &|E_{t+1}\delta_{t+1}|_1 - \alpha|E_t\delta_t|_1 \leq 0. && (6.47)
\end{aligned}$$

Contraction then follows from Lemma 1 with contraction metric $V_t = |E(x_t, u_t)\delta_t|_1$.
*Monotonicity and Contraction* Finally, to see how contraction follows from (6.17) and (6.19), note that they imply conditions (6.15) and (6.16).

□

## Proof of Proposition 6.1

*Proof.* The first step (6.28) can be broken up into the following sum:

$$\theta(k+1) = \arg\min_{\theta} \sum_{i=1}^{N} \hat{J}^i_{ee}(\theta^i_u) + \frac{\rho}{2}||\theta^i_u - \phi^i_u(k) + u^i_u(k)||^2,$$

which is equivalent to the $N$ optimization problems in (6.31). The second step (6.29) can be written as

$$\phi(k+1) = \arg\min_{\phi} \mathcal{I}_{\Theta_{m\ell_1}}(\phi) + \sum_{i=1}^{N} \frac{\rho}{2}||\theta^i_d(k+1) - \phi^i_d + u^i_d(k)||^2. \tag{6.48}$$

We will show that the indicator function can be written as a sum over $i = 1, ..., N$ indicator functions each depending on $\phi^i_d$. Splitting it up in terms of the individual constraints, we get

$$\mathcal{I}_{\Theta_{m\ell_1}}(\phi) = \mathcal{I}_{F_x \geq 0}(\phi) + \mathcal{I}_{F_u \geq 0}(\phi) + \mathcal{I}_{E \in \mathbb{M}}(\phi) + \mathcal{I}_{\mathbf{1}^\top(\alpha E - F \geq 0)}(\phi). \tag{6.49}$$

The first two terms can be written as element-wise SOS constraints. The last two terms can then be written as a sum over the columns of the matrices $E$ and $F$. We can therefore right (6.49) as:

$$\mathcal{I}_{\Theta_{m\ell_1}}(\phi) = \sum_i \mathcal{I}_{\phi^i_d \in \Theta^i_{m\ell_1}}(\phi^i_d)$$

where,

$$\mathcal{I}_{\phi_d^i \in \Theta_{m\ell_1}^i}(\phi_d^i) = \mathcal{I}_{\alpha E^{ii} - \sum_{k \in \mathscr{V}_d^i} F^{ki} \geq 0}(\phi_d^i) + \mathcal{I}_{E^{ii} + E^{ii\top} > \epsilon}(\phi_d^i) + \sum_{k \in \mathscr{V}_d^i} \mathcal{I}_{E^{ki} \geq 0}(\phi_d^i) + \sum_{k \in \mathscr{V}_d^i} \mathcal{I}_{F^{ki} \geq 0}(\phi_d^i).$$

□

## Proof of Theorem 6.2

Sufficiency follows from Theorem 1.

We now prove necessity, i.e. that if a positive linear system is Schur stable, then $\theta \in \Theta_{m\ell_1}$. Suppose a matrix $A$ is Schur stable. Then by [171, proposition 2], there exists some $z > 0$ such that $z^\top A - z^\top < 0$. We can always rescale $z$ such that $z^\top A - z^\top \leq -\epsilon \mathbf{1}$. With this $z$, we choose $E = diag(z) \geq 0$ and $F = EA \geq 0$. Then

$$z^\top A - z^\top \leq -\epsilon \mathbf{1}^\top \implies \mathbf{1}^\top (F - E) \leq -\epsilon \mathbf{1}^\top \implies \theta \in \Theta_{m\ell_1}$$

# Chapter 7

# Conclusion

The central theme of this thesis has been the development of model sets that are expressive, yet also allow for certain behavioral constraints to be easily imposed during training. These constraints are useful for: improving model robustness, improving model generalization, and ensuring safety when the model must interact with other systems. As discussed in Section 2.3, the behaviors that we wish to encode can be described by infinite dimensional inequality constraints which are intractable for most model sets. To guarantee that models obey these behavioral constraints, we introduce a series of relaxation methods in section 2.4 that we use throughout the thesis.

In chapter 3, we studied recurrent neural networks and showed that by using a combination of implicit parameterizations and incremental quadratic constraints, we could construct a convex description of stable and Lipschitz bounded RNNs. We then showed that the proposed model is highly expressive and contains all previously published sets of stable RNNs and all stable linear time-invariant (LTI) systems. Our numerical experiments showed that these constraints are beneficial for model robustness, generalizability, and lead to good bounds on the Lipschitz constant.

In Chapter 4, we studied a particular algebraic loop that can be viewed as a fixed point of a neural ode or RNN. We showed that this algebraic loop results in a class of equilibrium networks which contains many standard neural network structures. Using incremental quadratic constraints, we developed conditions guaranteeing well-

posedness and Lipschitz boundedness for that class of equilibrium networks. A significant advantage of the model class is that it permits a *direct parameterization* allowing for training via unconstrained optimization methods. We ran experiments with the model set on small-scale image classification tasks and showed that the proposed parameterization allows for tight Lipschitz bounds to be imposed during training. We also found that imposing these Lipschitz bounds significantly improved the model robustness to adversarial attacks with limited loss in performance.

In Chapter 5, we combined the methods developed in the previous two chapters to develop a new model class, the *recurrent equilibrium network*. We showed that the recurrent equilibrium network has direct parameterizations which ensure: contraction, Lipschitz bounds, or incremental passivity. We also showed that the recurrent equilibrium network is highly expressive and contains many commonly used model classes for learning static and dynamic mappings. We demonstrated the model class on benchmark nonlinear system identification problems and demonstrated two novel applications of behaviorally constrained models: data-driven nonlinear observer design and control with stability guarantees.

Finally, in Chapter 6, we extended our approach to the learning of large-scale networks of models with a guarantee that their interconnection is stable and/or monotone. We proposed a particular class of storage function that is well suited to monotone systems and has a separable structure, allowing for the distributed identification of both a model and its stability certificate via the alternating directions method of multipliers (ADMM). We then demonstrated the approach on a variety of linear and nonlinear case studies, including a nonlinear traffic network with a 200-dimensional state space.

## 7.1 Future Research Directions

We will conclude with some remarks on future research directions and open problems.

### Model Scalability

The work in this thesis has demonstrated that imposing behavioral constraints can significantly improve model generalizability and robustness. An obvious application for such models is the training of large-scale models in computer vision, which are known to be very brittle. We were able to demonstrate our approach on small-scale image processing tasks in Chapter 4 and found promising results, however, we had difficulty extending the approach to large-scale convolutional models. The main difficulty was that multilayer convolutional LBENs were not easily compatible with the Peaceman-Rachford operator splitting, requiring us to use the Forward-Backward splitting scheme. While the Forward-Backward Scheme worked for small models, it could take many iterations to converge for larger models, making training difficult.

As the scalability issues result from having to solve for an equilibrium condition, we see two approaches to improving model scalability: a) remove the need to solve for equilibria, or b) make the equilibria easier to find.

a) As we demonstrated in chapter 5, it is possible to construct robust, direct parameterizations where it is not necessary to solve for the equilibria, e.g. the acyclic REN. The acyclic REN is not well suited to image processing tasks as model evaluation would be slow on a GPU as there are many sequential layers. There may, however, exist parameterizations that are similar in structure to traditional deep feedforward neural networks that are quick to evaluate on a GPU.

b) Recent work on equilibrium networks has shown that by regularizing the conditioning of the equilibrium equation can make model evaluation much faster [16]. This could easily applied to the LBEN and it is likely that as the study of equilibrium networks progresses, tools and training processes will emerge that further extend the scalability of equilibrium networks.

## Further Applications of the REN

In Chapter 5, we introduced the REN and demonstrated its application in system identification, observer design, and nonlinear feedback design, however, the REN also has many further applications in reinforcement learning and online learning.

Recent work in reinforcement learning has suggested that including Lipschitz bounds on policies may improve system robustness [180]. There is also a great need for reinforcement learning methods with stability certificates for safety critical systems [22, 81, 245]. The REN may find application in stability certified reinforcement learning by exploiting the compositional properties of IQCs [10]. For example, learning passive feedback policies for robotic manipulators would ensure stability of the manipulator when interacting with passive environments.

At the end of Chapter 5, we demonstrated how a REN could be used to construct a stable Q-parameter which was fit via a convex program. This problem could also be solved online to construct stable nonlinear adaptive controllers.

## Distributed Identification

In Chapter 6, we used Lagrangian Relaxation of equation error as a quality-of-fit metric as it significantly simplified the model fitting. For many problems, however, the state estimates required by Lagrangian relaxation of equation error are not available.

When state estimates are unavailable, it is common to combine equation error minimization with a method for estimating the states, e.g., via subspace methods [228] or a maximum likelihood framework [220]. In general, subspace and maximum likelihood methods do not provide state estimates that are consistent with a positive system realization, even if one exists. It would be useful to develop subspace or maximum likelihood methods that return a positive system realization.

An alternative approach to fitting models in the absence of state estimates is to use a prediction error framework [127] using simulation error as a quality of fit criteria. It

should be noted, however, that in this case, each ADMM step becomes much slower, which might complicate model fitting.

Finally, we note that the model set developed in this Chapter 6 is compatible with the observer design approach from Chapter 5. This could be used for a data-driven approach to design large-scale observers for positive/monotone systems.

[1] Behçet Açıkmeşe and Martin Corless. Observers for systems with nonlinearities satisfying incremental quadratic constraints. *Automatica*, 47(7): 1339–1348, 2011.

[2] Naman Agarwal, Brian Bullins, Elad Hazan, Sham Kakade, and Karan Singh. Online Control with Adversarial Disturbances. In *International Conference on Machine Learning*, pages 111–119. PMLR, May 2019.

[3] Amir Ali Ahmadi and Bachir El Khadir. Learning dynamical systems with side information. In *Learning for Dynamics and Control*, pages 718–727. PMLR, 2020.

[4] Amir Ali Ahmadi, Alex Olshevsky, Pablo A Parrilo, and John N Tsitsiklis. Np-hardness of deciding convexity of quartic polynomials and related problems. *Mathematical Programming*, 137(1):453–476, 2013.

[5] Alessandro Alessio and Alberto Bemporad. A Survey on Explicit Model Predictive Control. In Lalo Magni, Davide Martino Raimondo, and Frank Allgöwer, editors, *Nonlinear Model Predictive Control: Towards New Challenging Applications*, Lecture Notes in Control and Information Sciences, pages 345–369. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-01094-1.

[6] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR, 2017.

[7] Martin S Andersen, Joachim Dahl, and Lieven Vandenberghe. Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. *Mathematical Programming Computation*, 2(3-4):167–201, 2010.

[8] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*, pages 291–301. PMLR, 2019.

[9] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[10] Murat Arcak, Chris Meissen, and Andrew Packard. *Networks of dissipative systems: compositional certification of stability, performance, and safety.* Springer, 2016.

[11] Alessandro Astolfi, Dimitrios Karagiannis, and Romeo Ortega. *Nonlinear and Adaptive Control with Applications.* Springer Science & Business Media, 2007.

[12] Erin M Aylward, Pablo A Parrilo, and Jean-Jacques E Slotine. Stability and robustness analysis of nonlinear systems via contraction metrics and sos programming. *Automatica*, 44(8):2163–2170, 2008.

[13] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[14] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, pages 690–701, 2019.

[15] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *arXiv preprint arXiv:2006.08656*, 2020.

[16] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Stabilizing equilibrium models by jacobian regularization. *arXiv preprint arXiv:2106.14342*, 2021.

[17] Nikita. E. Barabanov and Danil. V. Prokhorov. Stability analysis of discrete-time recurrent neural networks. *IEEE Transactions on Neural Networks*, 13(2):292–303, March 2002.

[18] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[19] Heinz H Bauschke, Patrick L Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[20] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[21] Peter Benner. Solving large-scale control problems. *IEEE Control Systems Magazine*, 24(1):44–59, 2004.

[22] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, pages 908–919, 2017.

[23] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[24] Abraham Berman and Robert J Plemmons. *Nonnegative matrices in the mathematical sciences*, volume 9. Siam, 1994.

[25] Pauline Bernard. *Observer Design for Nonlinear Systems*. Springer, 2019.

[26] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[27] Michael J Best and Nilotpal Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1):425–439, 1990.

[28] Stephen A Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.

[29] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[30] Bradley N Bond, Zohaib Mahmood, Yan Li, Ranko Sredojevic, Alexandre Megretski, Vladimir Stojanovi, Yehuda Avniel, and Luca Daniel. Compact modeling of nonlinear analog circuits using system identification via semidefinite programming and incremental stability certification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(8):1149–1162, 2010.

[31] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[32] Stephen P Boyd and Craig H Barratt. *Linear controller design: limits of performance*, volume 7. Citeseer, 1991.

[33] Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

[34] M. Buehner and P. Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, May 2006. ISSN 1941-0093.

[35] Felix Bünning, Adrian Schalbetter, Ahmed Aboudonia, Mathias Hudoba de Badyn, Philipp Heer, and John Lygeros. Input convex neural networks for building mpc. In *Learning for Dynamics and Control*, pages 251–262. PMLR, 2021.

[36] Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[37] Giuseppe Calafiore and Fabrizio Dabbene. Control design with hard/soft performance specifications: A Q -parameter randomization approach. *International Journal of Control*, 77(5):461–471, March 2004. ISSN 0020-7179, 1366-5820.

[38] Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. Provably minimally-distorted adversarial examples. *arXiv preprint arXiv:1709.10207*, 2017.

[39] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[40] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer, 2017.

[41] Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Association for the Advancement of Artificial Intelligence*, pages 3601–3608, 2020.

[42] Yun-Chung Chu and Keith Glover. Bounds of the induced norm and model reduction errors for systems with repeated scalar nonlinearities. *IEEE Transactions on Automatic Control*, 44(3):471–483, 1999.

[43] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320, 2019.

[44] Guy W Cole and Sinead A Williamson. Avoiding resentment via monotonic fairness. *arXiv preprint arXiv:1909.01251*, 2019.

[45] Patrick L Combettes and Jean-Christophe Pesquet. Deep neural network structures solving variational inequalities. *Set-Valued and Variational Analysis*, pages 1–28, 2020.

[46] Giacomo Como, Enrico Lovisari, and Ketan Savla. Throughput optimality and overload behavior of dynamical flow networks under monotone distributed routing. *IEEE Transactions on Control of Network Systems*, 2(1):57–67, 2015.

[47] Giacomo Como, Enrico Lovisari, and Ketan Savla. Convexity and robustness of dynamic traffic assignment and freeway network control. *Transportation Research Part B: Methodological*, 91:446–465, 2016.

[48] Samuel Coogan. A contractive approach to separable lyapunov functions for monotone systems. *Automatica*, 106:349–357, 2019.

[49] Samuel Coogan and Murat Arcak. Dynamical properties of a compartmental model for traffic networks. In *American Control Conference (ACC), 2014*, pages 2511–2516. IEEE, 2014.

[50] Zac Cranko, Zhan Shi, Xinhua Zhang, Richard Nock, and Simon Kornblith. Generalised lipschitz regularisation equals distributional robustness. In *International Conference on Machine Learning*, pages 2178–2188. PMLR, 2021.

[51] Fernando J D'Amato, Mario A Rotea, AV Megretski, and UT Jönsson. New results for analysis of systems with repeated nonlinearities. *Automatica*, 37(5): 739–747, 2001.

[52] Hennie Daniels and Marina Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

[53] Arne Dankers, Paul MJ Van den Hof, Xavier Bombois, and Peter SC Heuberger. Identification of dynamic models in complex networks with prediction error methods: Predictor input selection. *IEEE Transactions on Automatic Control*, 61(4):937–952, 2016.

[54] Nicola De Cao, Wilker Aziz, and Ivan Titov. Block neural autoregressive flow. In *Uncertainty in Artificial Intelligence*, pages 1263–1273. PMLR, 2020.

[55] Patrick De Leenheer, David Angeli, and Eduardo D Sontag. Monotone chemical reaction networks. *Journal of mathematical chemistry*, 41(3): 295–314, 2007.

[56] Sarah Dean, Nikolai Matni, Benjamin Recht, and Vickie Ye. Robust guarantees for perception-based control. In *Learning for Dynamics and Control*, pages 350–360. PMLR, 2020.

[57] Charles A Desoer and Mathukumalli Vidyasagar. *Feedback systems: input-output properties*, volume 55. SIAM, 1975.

[58] Gunther Dirr, Hiroshi Ito, Anders Rantzer, and Björn Rüffer. Separable Lyapunov functions for monotone systems: Constructions and limitations. *Discrete Contin. Dyn. Syst. Ser. B*, 20(8):2497–2526, 2015.

[59] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *UAI*, volume 1, page 3, 2018.

[60] Bradley Efron and Trevor Hastie. *Computer age statistical inference*, volume 5. Cambridge University Press, 2016.

[61] Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

[62] Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Y. Tsai. Implicit deep learning. *arXiv:1908.06315*, 2019.

[63] Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021.

[64] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[65] Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *arXiv preprint arXiv:1903.01287*, 2019.

[66] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 11427–11438, 2019.

[67] Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 2020.

[68] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 2014.

[69] F. Ferraguti, N. Preda, A. Manurung, M. Bonfè, O. Lambercy, R. Gassert, R. Muradore, P. Fiorini, and C. Secchi. An Energy Tank-Based Interactive Control Architecture for Autonomous and Teleoperated Robotic Surgery. *IEEE Transactions on Robotics*, 31(5):1073–1088, October 2015. ISSN 1941-0468.

[70] Urban Forssell and Lennart Ljung. Closed-loop identification revisited. *Automatica*, 35(7):1215–1241, 1999.

[71] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[72] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[73] Kenji Fujimoto and Toshiharu Sugie. Characterization of all nonlinear stabilizing controllers via observer-based kernel representations. *Automatica*, 36(8):1123–1135, August 2000. ISSN 0005-1098.

[74] Brian H Gilding and Robert Kersner. *Travelling Waves in Nonlinear Diffusion-Convection Reaction*, volume 60. Birkhauser, 2012.

[75] Fouad Giri and Er-Wei Bai. *Block-oriented nonlinear system identification*, volume 1. Springer, 2010.

[76] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[77] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pages 297–299. PMLR, 2018.

[78] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[79] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[80] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing Lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

[81] Fangda Gu, He Yin, Laurent El Ghaoui, Murat Arcak, Peter Seiler, and Ming Jin. Recurrent neural network controllers synthesis with stability guarantees for partially observed systems. *arXiv preprint arXiv:2109.03861*, 2021.

[82] Akhil Gupta, Naman Shukla, Lavanya Marla, Arinbjörn Kolbeinsson, and Kartik Yellepeddi. How to incorporate monotonicity in deep networks while preserving flexibility? *arXiv preprint arXiv:1909.10662*, 2019.

[83] Aleksandar Haber and Michel Verhaegen. Subspace identification of large-scale interconnected systems. *IEEE Transactions on Automatic Control*, 59(10):2754–2759, 2014.

[84] Wassim M Haddad, VijaySekhar Chellaboina, and Qing Hui. *Nonnegative and compartmental dynamical systems*. Princeton University Press, 2010.

[85] Trevor J Hastie and Robert J Tibshirani. *Generalized additive models*, volume 43. CRC press, 1990.

[86] Takeshi Hatanaka, Nikhil Chopra, Masayuki Fujita, and Mark W Spong. *Passivity-based control and estimation in networked robotics*. Springer, 2015.

[87] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[88] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NIPS*, 2017.

[89] Esteban Hernandez-Vargas, Patrizio Colaneri, Richard Middleton, and Franco Blanchini. Discrete-time control for switched positive systems with application to mitigating viral escape. *International journal of robust and nonlinear control*, 21(10):1093–1111, 2011.

[90] Esteban A Hernandez-Vargas, Patrizio Colaneri, and Richard H Middleton. Optimal therapy scheduling for a simplified HIV infection model. *Automatica*, 49(9):2874–2880, 2013.

[91] Joao P Hespanha. *Linear Systems Theory*. Princeton university press, 2018.

[92] Morris W Hirsch and Hal Smith. Monotone dynamical systems. In *Handbook of differential equations: ordinary differential equations*, volume 2, pages 239–357. Elsevier, 2006.

[93] Sepp Hochreiter and Jurgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9:1735–1780, 1997.

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

[95] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint arXiv:2102.00554*, 2021.

[96] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR, 2018.

[97] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the Lipschitz constant as a defense against adversarial examples. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 16–29. Springer, 2018.

[98] Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the lipschitz constant as a defense against adversarial examples. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 16–29. Springer, 2018.

[99] Michael Innes. Don't unroll adjoint: Differentiating ssa-form programs. *arXiv preprint arXiv:1810.07951*, 2018.

[100] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[101] Vanessa Jonsson, Anders Rantzer, and Richard M Murray. A scalable formulation for engineering combination therapies for evolutionary dynamics of disease. In *American Control Conference (ACC), 2014*, pages 2771–2778. IEEE, 2014.

[102] Kyle D Julian, Jessica Lopez, Jeffrey S Brush, Michael P Owen, and Mykel J Kochenderfer. Policy compression for aircraft collision avoidance systems. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2016.

[103] Eugenius Kaszkurewicz and Amit Bhaya. *Matrix Diagonal Stability in Systems and Computation*. Birkhäuser Basel, 2000.

[104] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[105] Yu Kawano and Ming Cao. Design of Privacy-Preserving Dynamic Controllers. *IEEE Transactions on Automatic Control*, 65(9):3863–3878, September 2020. ISSN 1558-2523.

[106] Yu Kawano, Bart Besselink, and Ming Cao. Contraction analysis of monotone systems via separable functions. *IEEE Transactions on Automatic Control*, 2019.

[107] R Bruce Kellogg. A nonlinear alternating direction method. *Mathematics of Computation*, 23(105):23–27, 1969.

[108] Hassan K Khalil. *Nonlinear systems*. Prentice-Hall, 2002.

[109] Hassan K Khalil. *High-Gain Observers in Nonlinear Feedback Control*. SIAM, 2017.

[110] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[111] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations*, 2015.

[112] J. Zico Kolter and Gaurav Manek. Learning stable deep dynamics models. In *Advances in Neural Information Processing Systems 32*, pages 11128–11136. Curran Associates, Inc., 2019.

[113] J Zico Kolter and Gaurav Manek. Learning stable deep dynamics models. *Advances in Neural Information Processing Systems*, 32:11128–11136, 2019.

[114] Anders Krogh and John Hertz. A simple weight decay can improve generalization. In J. Moody, S. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1992.

[115] V. V. Kulkarni and M. G. Safonov. Incremental positivity nonpreservation by stability multipliers. *IEEE Trans. Autom. Control*, 47(1):173–177, Jan. 2002.

[116] Vishwesh V Kulkarni and Michael G Safonov. All multipliers for repeated monotone nonlinearities. *IEEE Transactions on Automatic Control*, 47(7): 1209–1212, 2002.

[117] Seth L Lacy and Dennis S Bernstein. Subspace identification with guaranteed stability using constrained optimization. *IEEE Transactions on automatic control*, 48(7):1259–1263, 2003.

[118] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[119] Benoît Legat, Chris Coey, Robin Deits, Joey Huchette, and Amelia Perry. Sum-of-squares optimization in Julia. In *The First Annual JuMP-dev Workshop*, 2017.

[120] Klas Leino, Zifan Wang, and Matt Fredrikson. Globally-robust neural networks. *ICML*, 2021.

[121] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[122] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 888–896. PMLR, 2019.

[123] Changliu Liu, Tomer Arnon, Christopher Lazarus, Clark Barrett, and Mykel J Kochenderfer. Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*, 2019.

[124] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[125] Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks. *arXiv preprint arXiv:2011.10219*, 2020.

[126] Zhang Liu and Lieven Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1235–1256, 2009.

[127] Lennart. Ljung. *System identification theory for the user*. Prentice Hall, Upper Saddle River, N.J, 2nd edition, 1999. ISBN 0-13-244193-4.

[128] Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289. IEEE, 2004.

[129] Winfried Lohmiller and Jean-Jacques E Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998.

[130] Enrico Lovisari, Giacomo Como, and Ketan Savla. Stability of monotone dynamical flow networks. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 2384–2389. IEEE, 2014.

[131] J. M. Maciejowski. Guaranteed stability with subspace methods. *Systems & Control Letters*, 26(2):153–156, September 1995. ISSN 0167-6911.

[132] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[133] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pages 6640–6650. PMLR, 2020.

[134] Giorgos ('Yorgos') Mamakoukas, Orest Xherija, and Todd Murphey. Memory-Efficient Learning of Stable Linear Dynamical Systems for Prediction and Control. *Advances in Neural Information Processing Systems*, 33: 13527–13538, 2020.

[135] Ian R Manchester. Contracting nonlinear observers: Convex optimization and learning from data. In *2018 American Control Conference (ACC)*, pages 1873–1880. IEEE, 2018.

[136] Ian R Manchester and Jean-Jacques E Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 62(6):3046–3053, 2017.

[137] Ian R Manchester and Jean-Jacques E Slotine. On existence of separable contraction metrics for monotone nonlinear systems. *IFAC-PapersOnLine*, 50 (1):8226–8231, 2017.

[138] Ian R Manchester, Mark M Tobenkin, and Alexandre Megretski. Stable nonlinear system identification: Convexity, model class, and consistency. *IFAC Proceedings Volumes*, 45(16):328–333, 2012.

[139] Donatello Materassi and Giacomo Innocenti. Topological identification in networks of dynamical systems. *IEEE Transactions on Automatic Control*, 55 (8):1860–1871, 2010.

[140] Donatello Materassi and Murti V Salapaka. On the problem of reconstructing an unknown topology via locality properties of the wiener filter. *IEEE transactions on automatic control*, 57(7):1765–1777, 2012.

[141] Donatello Materassi, Giacomo Innocenti, Laura Giarré, and M Salapaka. Model identification of a network as compressing sensing. *Systems & Control Letters*, 62(8):664–672, 2013.

[142] Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jürgen Schmidhuber. A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, 22(13-14): 1521–1537, 2008.

[143] Alexandre Megretski. Positivity of trigonometric polynomials. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 4, pages 3814–3817. IEEE, 2003.

[144] Alexandre Megretski. Convex optimization in robust identification of nonlinear feedback. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 1370–1374. IEEE, 2008.

[145] Alexandre Megretski and Anders Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6): 819–830, 1997.

[146] Thomas Meurer. On the extended luenberger-type observer for semilinear distributed-parameter systems. *IEEE Transactions on Automatic Control*, 58 (7):1732–1743, 2013.

[147] Daniel N Miller and Raymond A De Callafon. Subspace identification with eigenvalue constraints. *Automatica*, 49(8):2468–2473, 2013.

[148] John Miller and Moritz Hardt. Stable Recurrent Models. *In Proceedings of ICLR 2019*, May 2018.

[149] Matthew Mirman, Timon Gehr, and Martin T Vechev. Differentiable abstract interpretation for provably robust neural networks. In *ICML*, 2018.

[150] Ari S Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *NeurIPS*, 2018.

[151] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[152] U. Nallasivam, B. Srinivasan, V.Kuppuraj, M. N. Karim, and R. Rengaswamy. Computationally Efficient Identification of Global ARX Parameters With Guaranteed Stability. *IEEE Transactions on Automatic Control*, 56(6): 1406–1411, June 2011.

[153] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[154] An-phi Nguyen and María Rodríguez Martínez. Mononet: towards interpretable models by learning monotonic features. *arXiv preprint arXiv:1909.13611*, 2019.

[155] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

[156] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[157] J.P. Noël and M. Schoukens. F-16 aircraft benchmark based on ground vibration test data. *Workshop on Nonlinear System Identification Benchmarks*, pages 15–19, 2017.

[158] Chirag Pabbaraju, Ezra Winston, and J Zico Kolter. Estimating lipschitz constants of monotone deep equilibrium models. In *International Conference on Learning Representations*, 2020.

[159] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *International Conference on Learning Representations*, 2020.

[160] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P.A. Parrilo. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2013.

[161] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[162] Pablo A Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320, 2003.

[163] Patricia Pauli, Dennis Gramlich, Julian Berberich, and Frank Allgöwer. Linear systems with neural network nonlinearities: Improved stability analysis via acausal zames-falb multipliers. *arXiv preprint arXiv:2103.17106*, 2021.

[164] Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgower. Training robust neural networks using Lipschitz bounds. *IEEE Control Systems Letters*, 2021.

[165] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8(1):143–195, 1999.

[166] Imre Pólik and Tamás Terlaky. A survey of the s-lemma. *SIAM review*, 49(3): 371–418, 2007.

[167] VM Popov. Absolute stability of nonlinear systems of automatic control. *Automation and Remote Control*, 22(8):857–875, 1962.

[168] Haifeng Qian and Mark N Wegman. L2-nonexpansive neural networks. *International Conference on Learning Representations (ICLR)*, 2019.

[169] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR, 2017.

[170] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

[171] Anders Rantzer. Distributed control of positive systems. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 6608–6611. IEEE, 2011.

[172] Anders Rantzer. Scalable control of positive systems. *European Journal of Control*, 24:72–80, 2015.

[173] Anders Rantzer and Bo Bernhardsson. Control of convex-monotone systems. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 2378–2383. IEEE, 2014.

[174] Jonas Rauber, Roland Zimmermann, Matthias Bethge, and Wieland Brendel. Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax. *Journal of Open Source Software*, 5(53):2607, 2020.

[175] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9): 2352–2449, 2017.

[176] Max Revay and Ian R Manchester. Contracting implicit recurrent neural networks: Stable models with improved trainability. *L4DC 2020*, 2020.

[177] Max Revay, Ruigang Wang, and Ian R Manchester. A convex parameterization of robust recurrent neural networks. *IEEE Control Systems Letters*, 2020.

[178] Max Revay, Ruigang Wang, and Ian R. Manchester. Lipschitz bounded equilibrium networks. *arXiv:2010.01732*, 2020.

[179] Antônio H. Ribeiro, Koen Tiels, Jack Umenberger, Thomas B. Schön, and Luis A. Aguirre. On the smoothness of nonlinear system identification. *Automatica*, 121:109158, 2020. ISSN 0005-1098.

[180] Alessio Russo and Alexandre Proutiere. Optimal attacks on reinforcement learning policies. *arXiv preprint arXiv:1907.13548*, 2019.

[181] Alessio Russo and Alexandre Proutiere. Optimal Attacks on Reinforcement Learning Policies. *arXiv:1907.13548 [cs, stat]*, July 2019.

[182] Giovanni Russo, Mario di Bernardo, and Eduardo D Sontag. Stability of networked systems: a multi-scale approach using contraction. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 6559–6564. IEEE, 2010.

[183] Ernest K Ryu and Stephen Boyd. Primer on monotone operator methods. *Appl. Comput. Math*, 15(1):3–43, 2016.

[184] Sadra Sadraddini and Calin Belta. Formal synthesis of control strategies for positive monotone systems. *IEEE Transactions on Automatic Control*, 64(2): 480–495, 2018.

[185] Borhan M Sanandaji, Tyrone L Vincent, and Michael B Wakin. Exact topology identification of large-scale interconnected dynamical systems from compressive observations. In *Proceedings of the 2011 American Control Conference*, pages 649–656. IEEE, 2011.

[186] Thomas B Schön, Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

[187] M Schoukens and JP Noël. Wiener-hammerstein benchmark with process noise. In *Workshop on nonlinear system identification benchmarks*, pages 15–19, 2016.

[188] Maarten Schoukens and Koen Tiels. Identification of block-oriented nonlinear systems starting from linear approximations: A survey. *Automatica*, 85: 272–292, 2017. ISSN 0005-1098.

[189] Maarten Schoukens, Anna Marconato, Rik Pintelon, Gerd Vandersteen, and Yves Rolain. Parametric identification of parallel wiener–hammerstein systems. *Automatica*, 51:111–122, 2015.

[190] E. Shahriari, A. Kramberger, A. Gams, A. Ude, and S. Haddadin. Adapting to contacts: Energy tanks and task energy for passivity-based dynamic movement primitives. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 136–142, November 2017.

[191] Jinglai Shen and Xiao Wang. Estimation of shape constrained functions in dynamical systems and its application to gene networks. In *Proceedings of the 2010 American Control Conference*, pages 5948–5953. IEEE, 2010.

[192] Jinglai Shen and Xiao Wang. Estimation of monotone functions via p-splines: A constrained dynamical optimization approach. *SIAM Journal on Control and Optimization*, 49(2):646–671, 2011.

[193] Humberto Stein Shiromoto, Max Revay, and Ian R Manchester. Distributed nonlinear control design using separable control contraction metrics. *IEEE Transactions on Control of Network Systems*, 6(4):1281–1290, 2018.

[194] Naum Z Shor. Class of global minimum bounds of polynomial functions. *Cybernetics*, 23(6):731–734, 1987.

[195] Dragoslav D Siljak. *Decentralized control of complex systems*. Courier Corporation, 2011.

[196] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[197] Max Simchowitz, Karan Singh, and Elad Hazan. Improper Learning for Non-Stochastic Control. In *Conference on Learning Theory*, pages 3320–3436. PMLR, July 2020.

[198] John W Simpson-Porco and Francesco Bullo. Contraction theory on riemannian manifolds. *Systems & Control Letters*, 65:74–80, 2014.

[199] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T Vechev. Fast and effective robustness certification. *NeurIPS*, 1(4):6, 2018.

[200] Jonas Sjöberg, Qinghua Zhang, Lennart Ljung, Albert Benveniste, Bernard Delyon, Pierre-Yves Glorennec, Håkan Hjalmarsson, and Anatoli Juditsky. Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12):1691–1724, 1995.

[201] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[202] Hal L Smith. *Monotone dynamical systems: an introduction to the theory of competitive and cooperative systems*. American Mathematical Soc., 2008.

[203] Vera L. J. Somers and Ian R. Manchester. Priority maps for surveillance and intervention of wildfires and other spreading processes. *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[204] Eduardo D Sontag. Input to state stability: Basic concepts and results. In *Nonlinear and optimal control theory*, pages 163–220. Springer, 2008.

[205] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[206] Mario Stylianou and Nancy Flournoy. Dose finding using the biased coin up-and-down design and isotonic regression. *Biometrics*, 58(1):171–177, 2002.

[207] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*, volume 2. MIT press, 2018.

[208] Andreas Svensson and Thomas B Schön. A flexible state–space model for learning nonlinear dynamical systems. *Automatica*, 80:189–199, 2017.

[209] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR: International Conference on Learning Representations*, 2014.

[210] Danielle C. Tarraf, William Shelton, Edward Parker, Brien Alkire, Diana Gehlhaus, Justin Grana, Alexis Levedahl, Jasmin Leveille, Jared Mondschein, James Ryseff, Ali Wyne, Daniel Elinoff, Edward Geist, Benjamin N. Harris, Eric Hui, Cedric Kenney, Sydne Newberry, Chandler Sachs, Peter Schirmer, Danielle Schlang, Victoria M. Smith, Abbie Tingstad, Padmaja Vedula, and Kristin Warren. *The Department of Defense Posture for Artificial Intelligence: Assessment and Recommendations*. RAND Corporation, Santa Monica, CA, 2019.

[211] Vincent Tjeng, Kai Y Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2018.

[212] M. M. Tobenkin, I. R. Manchester, J. Wang, A. Megretski, and R. Tedrake. Convex optimization in identification of stable non-linear state space models. In *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010.

[213] Mark M Tobenkin, Ian R Manchester, and Russ Tedrake. Invariant funnels around trajectories using sum-of-squares programming. *IFAC Proceedings Volumes*, 44(1):9218–9223, 2011.

[214] Mark M Tobenkin, Ian R Manchester, and Alexandre Megretski. Convex parameterizations and fidelity bounds for nonlinear identification and reduced-order modelling. *IEEE Transactions on Automatic Control*, 62(7): 3679–3686, 2017.

[215] Duc N Tran, Björn S Rüffer, and Christopher M Kellett. Convergence properties for discrete-time nonlinear systems. *IEEE Transactions on Automatic Control*, 64(8):3415–3422, 2018.

[216] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in neural information processing systems*, pages 6541–6550, 2018.

[217] Jack Umenberger and Ian R Manchester. Scalable identification of stable positive systems. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 4630–4635. IEEE, 2016.

[218] Jack Umenberger and Ian R Manchester. Convex bounds for equation error in stable nonlinear identification. *IEEE control systems letters*, 3(1):73–78, 2018.

[219] Jack Umenberger and Ian R Manchester. Specialized interior-point algorithm for stable nonlinear system identification. *IEEE Transactions on Automatic Control*, 64(6):2442–2456, 2018.

[220] Jack Umenberger, Johan Wågberg, Ian R Manchester, and Thomas B Schön. Maximum likelihood identification of stable linear dynamical systems. *Automatica*, 96:280–292, 2018.

[221] Jack Umenberger, Johan Wagberg, Ian R Manchester, and Thomas B Schön. Maximum likelihood identification of stable linear dynamical systems. *Automatica*, 96:280–292, 2018.

[222] Jonas Umlauft, Armin Lederer, and Sandra Hirche. Learning stable gaussian process state space models. In *2017 American Control Conference (ACC)*, pages 1499–1504. IEEE, 2017.

[223] Paul MJ Van den Hof, Arne Dankers, Peter SC Heuberger, and Xavier Bombois. Identification of dynamic models in complex networks with prediction error methods—basic methods for consistent module estimates. *Automatica*, 49(10):2994–3006, 2013.

[224] Paul M.J. Van den Hof, Arne G. Dankers, and Harm H.M. Weerts. Identification in dynamic networks. *Computers & Chemical Engineering*, 109: 23–29, 2018. ISSN 0098-1354.

[225] Arjan van der Schaft. *L2-Gain and Passivity in Nonlinear Control*. Springer-Verlag, third edition, 2017.

[226] Tony Van Gestel, Johan AK Suykens, Paul Van Dooren, and Bart De Moor. Identification of stable models in subspace identification by using regularization. *IEEE Transactions on Automatic control*, 46(9):1416–1420, 2001.

[227] Peter Van Overschee and Bart De Moor. N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30 (1):75–93, 1994.

[228] Peter Van Overschee and BL De Moor. *Subspace identification for linear systems: Theory - Implementation - Applications*. Springer Science & Business Media, 2012.

[229] Michel Verhaegen and Vincent Verdult. *Filtering and system identification: a least squares approach*. Cambridge university press, 2007.

[230] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[231] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[232] B. Wahlberg and P. M. Mäkilä. On approximation of stable linear dynamical systems using Laguerre and Kautz functions. *Automatica*, 32(5):693–708, May 1996. ISSN 0005-1098.

[233] Yuh-Shyang Wang, Nikolai Matni, and John C Doyle. Separable and localized system-level synthesis for large-scale systems. *IEEE Transactions on Automatic Control*, 63(12):4234–4249, 2018.

[234] Harm HM Weerts, Paul MJ Van den Hof, and Arne G Dankers. Identifiability of linear dynamic networks. *Automatica*, 89:247–258, 2018.

[235] Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. *Advances in Neural Information Processing Systems*, 32:1545–1555, 2019.

[236] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*, 2018.

[237] Jan C Willems. Dissipative dynamical systems part i: General theory. *Archive for rational mechanics and analysis*, 45(5):321–351, 1972.

[238] Ezra Winston and J. Zico Kolter. Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 10718–10728, 2020.

[239] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[240] Weiming Xiang, Patrick Musau, Ayana A Wild, Diego Manzanas Lopez, Nathaniel Hamilton, Xiaodong Yang, Joel Rosenfeld, and Taylor T Johnson. Verification for machine learning, autonomy, and neural networks survey. *arXiv preprint arXiv:1810.01989*, 2018.

[241] VA Yakubovich. Frequency conditions for the absolute stability of control systems with several nonlinear or linear nonstationary blocks. *Avtomatika i telemekhanika*, 6:5–30, 1967.

[242] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In *NeurIPS*, 2020.

[243] Bowen Yi, Ruigang Wang, and Ian R Manchester. Reduced-order nonlinear observers via contraction analysis and convex optimization. In *2020 American Control Conference (ACC)*. IEEE, 2020.

[244] Izzet B Yildiz, Herbert Jaeger, and Stefan J Kiebel. Re-visiting the echo state property. *Neural networks*, 35:1–9, 2012.

[245] He Yin, Peter Seiler, and Murat Arcak. Stability analysis using quadratic constraints for systems with neural network controllers. *IEEE Transactions on Automatic Control*, 2021.

[246] He Yin, Peter Seiler, Ming Jin, and Murat Arcak. Imitation learning with stability and safety guarantees. *IEEE Control Systems Letters*, 2021.

[247] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3320–3328, 2014.

[248] D. Youla, H. Jabr, and J. Bongiorno. Modern Wiener-Hopf design of optimal controllers–Part II: The multivariable case. *IEEE Transactions on Automatic Control*, 21(3):319–338, June 1976. ISSN 2334-3303.

[249] C. Yu and M. Verhaegen. Subspace identification of 1d large-scale heterogeneous network. In *2017 13th IEEE International Conference on Control Automation (ICCA)*, pages 218–223, 2017.

[250] Chengpu Yu, Jie Chen, and Michel Verhaegen. Subspace identification of individual systems in a large-scale heterogeneous network. *Automatica*, 109: 108517, 2019.

[251] G. Zames. Realizability Condition for Nonlinear Feedback Systems. *IEEE Transactions on Circuit Theory*, 11(2):186–194, 1964.

[252] George Zames. On the input-output stability of time-varying nonlinear feedback systems part I: Conditions derived using concepts of loop gain, conicity, and positivity. *IEEE transactions on automatic control*, 11(2): 228–238, 1966.

[253] George Zames. On the input-output stability of time-varying nonlinear feedback systems–part II: Conditions involving circles in the frequency plane and sector nonlinearities. *IEEE transactions on automatic control*, 11(3): 465–476, 1966.

[254] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[255] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016.

[256] Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and Optimal Control*, volume 40. Prentice hall New Jersey, 1996.

[257] SiQi Zhou and Angela P Schoellig. An analysis of the expressiveness of deep neural network architectures based on their Lipschitz constants. *arXiv preprint arXiv:1912.11511*, 2019.