Contractive and Robust Deep Neural Network in Continuous Time

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria Dell'Automazione

Author: **Daniele Martinelli**

Student ID: 970953
Advisor: Prof. Riccardo Scattolini
Co-advisors: Prof. Giancarlo Ferrari Trecate, Luca Furieri, Clara Galimberti
Academic Year: 2021-22

Abstract

In this thesis, we present a new class of deep neural networks in continuous time that can be used in different learning tasks, from classification problems to control applications such as continuous-time system identification and optimal control. The name of this architecture is: Recurrent Equilibrium Network Ordinary Differential Equations (REN-ODEs). This new class of neural networks consists of nonlinear dynamical systems that assures contractivity (a powerful form of stability) by design and can also guarantee incremental *integral quadratic constraints* (IQCs). IQCs are used to enforce properties of incremental dissipativity and passivity, as well as Lipschitz bounds. These properties provide *robustness* to the model. With this term we mean a mitigation of the sensitivity of the system's outputs with respect to small perturbations in the inputs. This feature is important in applications in which signals are affected by noise (e.g., system identification from real acquired data). Being contractive and robust *by design* means that the N parameters, characterizing a REN-ODE, are unconstrained. This property makes possible, during the learning phase, to use *unconstrained* iterative first-order optimization methods such as *gradient descent* (and its variations). The structure of the class is inspired by the Recurrent Equilibrium Networks (RENs) which, however, are formulated in discrete time. Moreover, the REN-ODE's architecture belongs to the family of Neural Ordinary Differential Equations (Neural-ODEs). As a result, REN-ODEs inherit all the advantages of Neural-ODEs, including the possibility to use modern and sophisticated ODE solvers for the evaluation of the model's trajectories. Furthermore, latest ODE solvers can provide high level of precision and adapt the evaluation strategy on the fly to achieve the requested level of accuracy. In this work, the properties of contractivity and robustness are validated on a nonlinear system identification problem and an optimal control task. Moreover, we evaluate the performance of the REN-ODEs over benchmark binary classification problems.

Keywords: Contractivity, Robustness, Lipschitz bound, Deep Neural Networks, Neural-ODE, Recurrent Equilibrium Network.

Contents

Introduction

Nowadays, Machine Learning (ML) methods have become more and more popular: image processing [1], pattern recognition [2], speech-to-text conversion [3], protein structure prediction [4] and learning to play complex games [5] are just few of the many possible areas of applications.
Thanks to their generality, flexibility and ability of learning from experience (i.e., from data), a rapid and still growing spread of ML methods has been reported in the control field, especially for large systems or non conventional applications, like driving soft and continuum robots or smart buildings management [6, 7]. Indeed, thanks to the possibility of training flexible machine learning models, exploiting big amount of data and having access to high computational power, many solutions to problems in automation, that were previously not achievable, are now possible. For example, when considering applications such as fault detection and/or control over large-scale systems, it is now feasible to retrieve non-linear models that are closer to reality, rather than using standard linearized versions of them [8].
Among the ML community, the Deep Learning computing paradigm has become one of the most widely used computational approaches, achieving impressive results on several complex tasks, matching or even beating those provided by human performance [9]. Deep Neural Networks (DNNs) have been more and more exploited, thanks to their ability to learn from massive amounts of data. However it has been observed that neural networks can be very sensitive to small changes in inputs [10] and thus not ideal for control applications in their generic form.
In [11], the authors introduce an innovative DNN architecture: Recurrent Equilibrium Networks (RENs). RENs are able to guarantee properties of stability and robustness while keeping a smooth mapping from the space $\mathbb{R}^N$ to the N weights and biases (i.e., the parameters) of the REN: this unconstrained parametrization is defined by the authors as *direct parametrization* and it makes the training process be an *unconstrained* optimization problem. *Robustness*, in the control field, is a general concept. In this thesis, more details will be provided and, in particular, further concepts that will lead to robust networks will be formally introduced (e.g., Lipschitz bounded gain, dissipativity and passivity).

Moreover, about robustness it has been shown in [10, 12] that different recurrent neural network models suffer of high sensitivity, i.e., small changes to the input produce substantial changes in the output. This collateral effect may be problematic in case of control systems, in which signals are usually affected by noise and disturbances. On the other hand, there are many empirical proofs that suggest that limitating Lipschitz constants of DNNs can have a positive impact in terms of speed of training and model performance in system identification [13] and generalization in ML [14]. However, even just calculating Lipschitz constant of a feedforward-NN has been proven to be a NP-hard problem [15]. Also, during the last few years, many different approaches to enforce Lipschitz-bounds have been provided, but they required the use of *Alternating Direction Method of Multipliers* or *convex implicit parametrization* of the networks to train [13, 16]. These problems can be usually estimated only for small neural networks, due to the poor scaling.
On the other side, all the models in the class of RENs are contracting or they can satisfy prescribed *incremental integral quadratic constraints* (IQC), including Lipschitz bounds and incremental passivity. Moreover, this new architecture has shown really good performances, outperforming popular models like LSTM, RNNs and Resnets in many different tasks such as system identification, nonlinear observers and data-driven feedback control designs. However, in the control community are well known the problems of implementation of discrete-time and sampled-data controllers for linear and nonlinear continuous-time systems due to the presence of sampling zero dynamics and the possibility of losing closed-loop stability [17–19]. In fact, there is no guarantee of preserving the properties of stability once the system is sampled (Figure 1). Additionally, if RENs are used as controllers or observers, the model returned after being trained, will depend from the sampling time T_s of data used during this phase. If, on the other hand, the real system suffers of time-varying delays in the communication between devices, highly possible in case of large-scale systems, the trained discrete REN system may not work as designed, while a continuously-defined dynamical system could automatically include data arriving at arbitrary times.
In this thesis, we present a new version of REN in continuous time: REN-ODE. It will be also shown that it is able to keep guarantees on properties of contractivity and passivity/dissipativity. This new system is inspired by the state-of-the-art Neural-ODEs [20]. Indeed, the authors proposed an innovative deep neural network class in which the number of hidden layers is not specified; instead, a *continuous number of layers* is considered, where the derivative of the hidden state is parameterized using a neural network. Successively, the trajectories of the system are obtained using modern ODE solvers, that can guarantee high level of precision and adapt the evaluation strategy on the fly to achieve the requested level of accuracy.

Organization of the Thesis

The thesis is organized as follows:

- In chapter 1, we provide preliminary knowledge about the definition of contractivity and some formal concepts to characterize robustness in a dynamical system: incremental dissipativity, passivity and IQCs. Furthermore, an introduction to the Recurrent Equilibrium Network and the Neural-ODE architecture is given. Details on the training procedure, as well as their key features, are reported.
- In chapter 2, we show that all REN-ODEs are contractive *by design*. This means that this property is guaranteed independently of the parameters. Moreover, we introduce a subclass of REN-ODEs, called *robust* REN-ODEs (RREN-ODE), that can also guarantee implicitly prescribed properties of incremental dissipativity/passivity or satisfy any given IQC.
- In chapter 3, we validate through simulations the properties of the REN-ODEs. We evaluate their performance in three different kind of problems: system identification of a nonlinear system, binary classification and optimal control of a multi agent system with safety policies.
- Finally, chapter 4 presents the conclusions of our work, as well as some future developments of REN-ODEs.

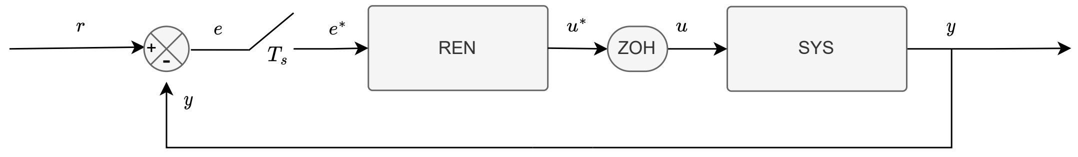


Figure 1: Equivalent scheme of the closed-loop system obtained using the *hold-equivalence*(HE) method.

1 | Preliminary Knowledge

Non-linear system analysis is still an open and hot topic due to the complexity of the subject. Concepts as *contractivity*, *dissipativity* and *passivity* are frequently used in this field. In this chapter, we fornally define these concepts, that will be of use for characterizing the properties of the REN-ODEs. Furthermore, we will introduce the architectures and properties of Neural-ODEs [20] and RENs [11]. The former is discussed, since REN-ODEs are a subclass of these models and thus, their properties will be exploited. The latter is a discrete-time NN architecture that also guarantees contractive and robustness properties. Our new models are inspired by RENs, while keeping the propertis of continuous-time NNs.

1.1. Contractivity

Based on a differential analysis of convergence, contractivity was introduced for the first time in [21] as a nonlinear system analysis method, inspired from fluid mechanics and differential geometry.
Consider a general deterministic dynamical system Σ of the form:

$$\Sigma = \begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ y(t) = g(x(t), u(t)) \end{cases} \tag{1.1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$, $y \in \mathcal{Y} \subseteq \mathbb{R}^p$, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ are respectively the system state, output and input. Furthermore, $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ and $g : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ are the state evolution and output functions. The functions f and g are $\mathcal{C}^1$, i.e., they are continuous and differentiable. We denote by $x^a(t)$ the state function starting from the initial condition $a \in \mathcal{X}$ at time t_0 up to the time instant t with an input u (from $t_0 \to t$).
It is assumed that the possible trajectories of the system Σ are restricted to have *left-compact support*, i.e., it exists a $t_0 \in \mathbb{R}$ such that (x, u, y) is zero outside $[t_0, \infty)$.
Thus, now we are ready to introduce the definition of a contracting system.

Definition 1.1.1 (Contracting System)**.** A system Σ in the form (1.1) is said to be *contracting* if for any two initial conditions a,$b\in\mathcal{X}$, $\forall t \geq t_0$, the state functions $x^a(t)$ and $x^b(t)$ with the same input function $u(t)$ satisfy:

$$\|x^a(t) - x^b(t)\| \leq \kappa e^{-c(t-t_0)}\|a-b\| \tag{1.2}$$

for some $c > 0$ and $\kappa > 0$.

The Definition 1.1.1 can be interpreted as follow: a contractive model is a model that "forgets" the initial condition with a certain rate c. Indeed, we have that

$$\lim_{t\to+\infty}\|x^a(t) - x^b(t)\| = 0.$$

This property can be useful especially in applications such as system identification or state estimation (i.e., state observer design), in which the initial state can be affected by uncertainty.
An equivalent definition of contractivity can be provided for discrete time systems. Let us consider a general discrete-time deterministic dynamical system Σ_{DT} of the form:

$$\Sigma_{DT} = \begin{cases} x(k+1) = f(x(k), u(k)) \\ y(k) = g(x(k), u(k)), \end{cases} \tag{1.3}$$

where $k \in \mathbb{N}$ denotes a time-step of the model and x, u, y, f and g have the same dimension/structure of the continuous-time counterpart. As done previously, we denote by $x^a(k)$ the state sequence starting from the initial condition $a \in \mathcal{X}$ at time k_0 up to the time step k with an input sequence u (from $k_0$to$k$). The following definition introduces the concept of a discrete-time contracting system.

Definition 1.1.2 (Contracting Discrete-Time System)**.** A system Σ_{DT} in the form (1.3) is said to be *contracting* if for any two initial conditions a,$b\in\mathcal{X}$, the state sequences $x^a(k)$ and $x^b(k)$ with same input sequence $u(k)$ satisfy $\forall k \geq k_0$:

$$\|x^a(k) - x^b(k)\| \leq c\,\alpha^k\|a-b\|, \tag{1.4}$$

for some $c > 0$ and $\alpha \in [0, 1)$.

1.2. Dissipativity

We define *supply rate* a function $s(u(t), y(t))$:

$$s\ :\ \mathcal{U} \times \mathcal{Y} \rightarrow \mathbb{R}. \tag{1.5}$$

The symbol $\mathbb{R}^+$ denotes $\mathbb{R}^+ = [0, \infty)$.
Dissipativity, in many physical systems, may be roughly interpreted as the way the system exchanges its internal energy with the external through the inputs and outputs. Thus, it is intuitive how the concept of dissipativity is important in control theory: under certain conditions of controllability and reachability, then the way the system stores and expels energy, based on inputs and outputs, can provide information about the overall stability of the process. Thus, a definition of a dissipative system can be provided [22].

1.2.1. General Dissipativity

Definition 1.2.1 (Generally Dissipative System)**.** Given a *supply rate* s, a system Σ in the form (1.1) is said to be dissipative with respect to s if there exists a function $\mathcal{S} : \mathcal{X} \rightarrow \mathbb{R}^+$, called *storage function*, such that for any initial condition $x(t_0) \in \mathcal{X}$ at any time t_0, and for any input $u(\cdot) \in \mathcal{U}$ and the following inequality holds:

$$\mathcal{S}(x(t_1)) \leq \mathcal{S}(x(t_0)) + \int_{t_0}^{t_1} s(u(t), y(t))dt\ , \quad \forall t_1 \geq t_0, \tag{1.6}$$

where the integral $\int_{t_0}^{t_1} s(u(t), y(t))dt$ is assumed to be well defined for all allowed $u(\cdot) \in \mathcal{U}$ and $y(\cdot) \in \mathcal{Y}$.

If the function $\mathcal{S}(x(t))$ is differentiable, (1.6) can be rewritten as:

$$\frac{d}{dt}\Big(\mathcal{S}(x(t))\Big) \leq\ s\big(u(t), y(t)\big), \tag{1.7}$$

and it is called *differentiated dissipation inequality.* If (1.6) holds with *equality* for all initial conditions $x(t_0)$, $t_1 \geq t_0$ and any allowed $u(\cdot)$, then the system Σ is said to be *conservative* with respect to the supply rate s.

Indeed, the storage function $\mathcal{S}(\cdot)$, as the name suggests, can be seen as the energy that the system can store and the supply rate $s(\cdot)$ as the law that describes how this energy balance varies with respect to inputs and outputs.

A definition of dissipativity can also be introduced for the discrete-time case.

Definition 1.2.2 (General Discrete-Time Dissipative System)**.** Given a *supply rate* s, a system Σ_{DT} in the form (1.3) is said to be *dissipative* with respect to s if there exists a function $\mathcal{S} : \mathcal{X} \to \mathbb{R}^+$, called *storage function*, such that for any initial condition $x(k_0) \in \mathcal{X}$ at any time step k_0, and for any input $u(\cdot) \in \mathcal{U}$ and $\forall k_1 \geq k_0$ the following inequality holds:

$$\mathcal{S}(x(k_1)) \leq \mathcal{S}(x(k_0)) + \sum_{i=k_0}^{k_1} s(u(i), y(i)) \tag{1.8}$$

1.2.2. Incremental Dissipativity

Dissipativity, as per Definition 1.2.1 or Definition 1.2.2, describes the way the system exchange its internal energy with the external. This notion can be extended to the *incremental* form of the system. Hence, *incremental dissipativity* is the study of the energy flow between any two trajectories of the system, where two trajectories may variate due to possible different initial conditions or sequence of inputs.
We denote with:

$$\Delta y(t) = y(t) - \bar{y}(t)\,, \quad \Delta u(t) = u(t) - \bar{u}(t)\,, \quad \Delta x(t) = x(t) - \bar{x}(t) \tag{1.9}$$

the finite differences between the two possible trajectories $(x, u, y), (\tilde{x}, \tilde{u}, \tilde{y})$ of the system Σ.

Definition 1.2.3 (Incrementally Dissipative System)**.** Given a *supply rate* s, a system Σ in the form (1.1) is said to be *incrementally dissipative* with respect to s if there exists a function $\mathcal{S} : \mathcal{X} \to \mathbb{R}^+$, called *storage function*, such that for any two possible trajectories $(x, u, y), (\bar{x}, \bar{u}, \bar{y})$:

$$\mathcal{S}(\Delta x(t_1)) \leq \mathcal{S}(\Delta x(t_0)) + \int_{t_0}^{t_1} s(\Delta u(t), \Delta y(t))dt\,, \quad \forall t_1 \geq t_0 \tag{1.10}$$

where the integral $\int_{t_0}^{t_1} s(\cdot)dt$ is assumed to be well defined for all allowed $u(\cdot) \in \mathcal{U}$ and $y(\cdot) \in \mathcal{Y}$, and $\Delta x, \Delta y, \Delta u$ are defined in (1.9).

1.3. Passivity

Once the definition of dissipativity has been stated, the property of passivity can be given. As a premise, a necessary condition for a system to be passive is that the input and output must have the same dimensions, i.e., $m \equiv p$ and $\mathcal{U}, \mathcal{Y} \subseteq \mathbb{R}^m$

Definition 1.3.1 (Passive System)**.** A system Σ in the form (1.1) is said to be *passive* if it is dissipative with respect to the supply rate $s(u, y) = u^\top y$.

Additionally, definitions of *input strictly passive* and *output strictly passive* systems can be given.

Definition 1.3.2 (Input Strictly Passive System)**.** A system Σ in the form (1.1) is said to be *input strictly passive* if $\exists \nu > 0$ such that Σ is dissipative with respect to the supply rate $s(u, y) = u^\top y - \nu \|u\|^2$.

Definition 1.3.3 (Output Strictly Passive System)**.** A system Σ in the form (1.1) is said to be *output strictly passive* if $\exists \varepsilon > 0$ such that Σ is dissipative with respect to the supply rate $s(u, y) = u^\top y - \varepsilon \|y\|^2$.

1.4. Integral Quadratic Constraints (IQCs

At this point, it would be interesting to find a parameterized form of the supply rate $s(\cdot)$, in such a way that, choosing properly the parameters, it is possible to retrieve the supply rate for specific cases (e.g., passivity, Lipschitz bounded system). This is the main idea behind the incremental integral quadratic constraints (IQCs). Before reporting the definition of IQCs, let $X \geq 0$ and $X \leq 0$ denote that the generic square matrix X is semi-positive definite and semi-negative definite, respectively.

Definition 1.4.1 (Incremental Integral Quadratic Constraint (IQC))**.** A system Σ in the form (1.1) is said to satisfy the *incremental integral quadratic constraints* (IQCs) defined by the matrices (Q, S, R) where $0 \geq Q \in \mathbb{R}^{p \times p}, S \in \mathbb{R}^{m \times p}$ and $R = R^T \in \mathbb{R}^{m \times m}$ if there exists a storage function $\mathcal{S} : \mathcal{X} \to \mathbb{R}^+$ such that for any pair of initial conditions $(x(t_0), \bar{x}(t_0))$ and input sequences $(u(t), \bar{u}(t)) \in \mathcal{U}$, the respective output sequences $y, \bar{y}$ (following (1.1)) satisfy:

$$\int_{t_0}^{t_1} \begin{bmatrix} y(t) - \bar{y}(t) \\ u(t) - \bar{u}(t) \end{bmatrix}^\top \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} y(t) - \bar{y}(t) \\ u(t) - \bar{u}(t) \end{bmatrix} dt \geq -\mathcal{S}(x(t_0) - \bar{x}(t_0)), \quad \forall t_1 \geq t_0. \tag{1.11}$$

where the integral $\int_{t_0}^{t_1}$ is assumed to be well defined for all allowed $u(\cdot) \in \mathcal{U}$ and $y(\cdot) \in \mathcal{Y}$, and $\Delta x, \Delta y, \Delta u$ are defined in (1.9).

It is interesting to note that (1.11) can be seen as a special reformulation of the incremental dissipation inequality (1.10) with $s(\Delta u, \Delta y) = \begin{bmatrix} \Delta y(t) \\ \Delta u(t) \end{bmatrix}^\top \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} \Delta y(t) \\ \Delta u(t) \end{bmatrix}$:

$$\int_{t_0}^{t_1} \begin{bmatrix} \Delta y(t) \\ \Delta u(t) \end{bmatrix}^\top \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} \Delta y(t) \\ \Delta u(t) \end{bmatrix} dt \geq \mathcal{S}(\Delta x(t_1)) - \mathcal{S}(\Delta x(t_0)) \tag{1.12}$$

For definition, $\mathcal{S}(x) \geq 0 \, \forall x$, thus:

$$\int_{t_0}^{t_1} \begin{bmatrix} \Delta y(t) \\ \Delta u(t) \end{bmatrix}^\top \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} \Delta y(t) \\ \Delta u(t) \end{bmatrix} dt \geq \mathcal{S}(\Delta x(t_1)) - \mathcal{S}(\Delta x(t_0)) \geq -\mathcal{S}(\Delta x(t_0)) \tag{1.13}$$

So we have just shown that (1.11) is the reformulation of (1.10) with a particular supply rate $s(\cdot)$ that now is parametrized with respect to the matrices (Q, S, R).
Let $\|y\|_T$ denote the finite 2-norm of the signal y from time t_0 up to time T; in formula:

$$\|y\|_T = \Big(\int_{t_0}^{T} |y(\tau)|^2 d\tau\Big)^{1/2}. \tag{1.14}$$

One of the main interesting applications of incremental IQCs is the possibility to test properties of the system (such as dissipativity and passivity) by choosing proper values of the fixed matrices (Q, S, R). Indeed, taking:

- $Q = -\frac{1}{\gamma} I, R = \gamma I, S = 0 \rightarrow$ the model satisfies an ℓ_2 Lipschitz bound of γ:

  Proof. Using the values of (Q, R, S) in (1.11) and the definition of $\Delta x, \Delta y, \Delta u$ from (1.9), where two possible trajectories are considered, i.e., $(x, u, y), (\bar{x}, \bar{u}, \bar{y})$:

$$\int_{t_0}^{t_1} \begin{bmatrix} \Delta y(t) \\ \Delta u(t) \end{bmatrix}^\top \begin{bmatrix} -\frac{1}{\gamma} I & 0 \\ 0 & \gamma I \end{bmatrix} \begin{bmatrix} \Delta y(t) \\ \Delta u(t) \end{bmatrix} dt \geq -\mathcal{S}(\Delta x(t_0)) \tag{1.15}$$

  Then it can be possible to compact everything using the notation (1.14) and denoting with $\mathcal{S}'(\cdot) = \gamma \mathcal{S}(\cdot)$:

$$-\frac{1}{\gamma}\|\Delta y\|_{t_1} + \gamma \|\Delta u\|_{t_1} \geq -\mathcal{S}(\Delta x(t_0)) \tag{1.16}$$

$$-\|\Delta y\|_{t_1} + \gamma^2 \|\Delta u\|_{t_1} \geq -\mathcal{S}'(\Delta x(t_0)) \tag{1.17}$$

$$\gamma^2 \|\Delta u\|_{t_1} + \mathcal{S}'(\Delta x(t_0)) \geq \|\Delta y\|_{t_1} \tag{1.18}$$

  Then it is possible to prove [22, Proposition 1.2.7] that the input-output map of the incremental model has ℓ_2-gain $\leq \gamma$, i.e.:

$$\|y-\bar{y}\|_{t_1} \leq \gamma\|u-\bar{u}\|_{t_1}, \quad \forall t_1 \geq t_0 \tag{1.19}$$

□

- $Q = 0, R = -2\nu I, S = I, \nu \geq 0 \rightarrow$ the system is incrementally input passive.

Proof. Using the values of (Q, S, R) into (1.11):

$$\int_{t_0}^{t_1} 2\Delta u^\top(t)\Delta y(t) - 2\nu\Delta u^\top(t)\Delta u(t)dt \geq -\mathcal{S}(\Delta x(t_0)), \tag{1.20}$$

Defining a storage function $\mathcal{S}' = \frac{1}{2}\mathcal{S}$, then (1.20) can be rewritten as:

$$\int_{t_0}^{t_1} \Delta u^\top(t)\Delta y(t) - \nu\Delta u^\top(t)\Delta u(t)dt \geq -\mathcal{S}'(\Delta x(t_0)), \tag{1.21}$$

where two possible trajectories are considered, i.e., $(x, u, y), (\bar{x}, \bar{u}, \bar{y})$ and $\Delta x, \Delta y, \Delta u$ are obtained as in (1.9). Finally, (1.21) is the definition of a incrementally dissipative system with supply rate
$s(\Delta u, \Delta y) = \Delta u^\top \Delta y - \nu\Delta u^\top \Delta u$, or, in other words, an incrementally input passive system (Definition 1.3.2). □

- $Q = -2\varepsilon I, R = 0, S = I, \varepsilon \geq 0 \rightarrow$ the system is incrementally output passive.

Proof. We can just use the values of (Q, S, R) into (1.11):

$$\int_{t_0}^{t_1} 2\Delta u^\top(t)\Delta y(t) - 2\varepsilon\Delta y^\top(t)\Delta y(t)dt \geq -\mathcal{S}(\Delta x(t_0)), \tag{1.22}$$

Defining a storage function $\mathcal{S}' = \frac{1}{2}\mathcal{S}$, then (1.22) can be rewritten as:

$$\int_{t_0}^{t_1} \Delta u^\top(t)\Delta y(t) - \varepsilon\Delta y^\top(t)\Delta y(t)dt \geq -\mathcal{S}'(\Delta x(t_0)), \tag{1.23}$$

where two possible trajectories are considered, i.e., $(x, u, y), (\bar{x}, \bar{u}, \bar{y})$ and $\Delta x, \Delta y, \Delta u$ are obtained as in (1.9). Finally, (1.23) is the definition of a incrementally dissipative system with supply rate
$s(\Delta u, \Delta y) = \Delta u^\top \Delta y - \varepsilon\Delta y^\top \Delta y$, or, in other words, an incrementally output passive system (Definition 1.3.3). □

1.5. Machine Learning background

RENs and REN-ODEs (the model proposed in this thesis) belong to the family of the *artificial neural networks* (ANNs). An ANN is a type of ML architecture composed, as the name suggests, by a network of artificial neurons, inspired by the human brain. These neurons are organized in three or more layers that connect the input features x (of dimension n), up to the outputs y (of dimension p). The main advantages of ANNs is that, according to the *universal function approximation theorem*, it is possible, with a finite number of neurons, to approximate any continuous function belonging on compact subsets of $\mathbb{R}^n$. This flexibility in the mapping between inputs and outputs makes ANN, theoretically, able to solve any kind of problem, and thus, applicable to many different cases. A neuron, sometimes called also "linear threshold unit" (LTU) with $(n+1)$ inputs and p outputs, is characterized by the following relation:

$$y = \sigma\left(\sum_{j=0}^{n} w_j x_j\right) \tag{1.24}$$

where x_j denotes the j^{th} input feature (with $w_0 x_0$ a constant, called *bias*), y the output, w_j a weight coefficient that multiplies the j^{th} input x_j and $\sigma(\cdot)$ a nonlinear function applied element-wise called *activation function* with $\sigma : \mathbb{R} \to \mathbb{R}$. Neurons are organized in layers that are interconnected in many possible ways, generating thus a network. Usually, in a feed-forward neural network, each layer, starting from the first one that takes as input the features x, is connected to the next one through nodes up to last one, that returns the p outputs. Each node is characterized by its own weights, one for each connection to another node. Thus, the net is composed mainly by three parts: (Figure 1.1):

1. a layer of neurons that are connected to the inputs, called "input layer";
2. a layer of neurons that return the outputs, called "output layer";
3. one or more layers of neurons that connect the input layer to the output one, called "hidden layers".

Furthermore, deep neural networks (DNNs) are ANNs with multiple hidden layers: indeed, the term *deep* refers to the large amount of layers that they can present inside.

A particular type of DNN are the recurrent neural networks (RNNs), which present a feedback interconnection inside the net. In other words, the output at time k (y_k) depends on the inputs at time k (u_k), but also on the states at the time $k-1$ (h_{k-1}). In Figure 1.2, a scheme of a generic RNN is reported. The dependencies on the previous state, allows the system to have a *memory of the past*, (i.e., as the concept of state in dynamical systems) making them useful in case of estimation or predictions from time sequences (such as audio or video).

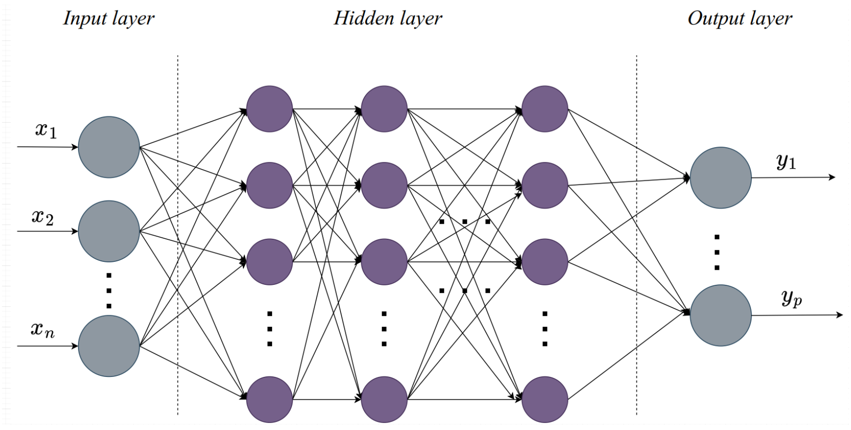


Figure 1.1: Scheme of a generic ANN having n input features and m outputs features.

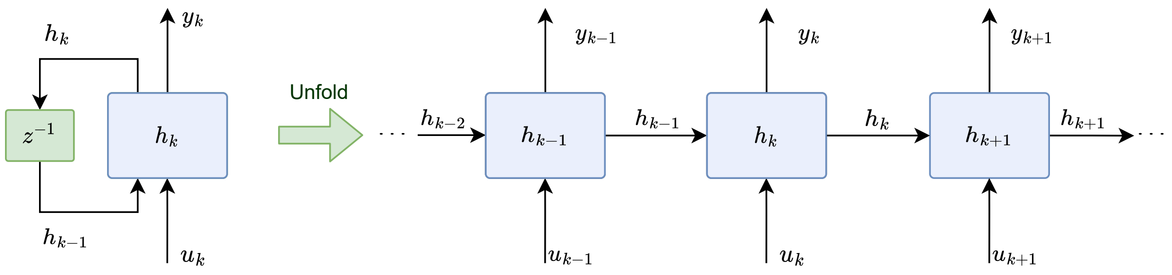


Figure 1.2: Compressed (left) and unfolded (right) generic RNN. The unitary delay is represented here with z^{-1}.

1.6. Recurrent Equilibrium Networks (RENs)

Firstly introduced by the authors in [11, 13, 23], Recurrent Equilibrium Networks (RENs) are a subset of RNNs. Architectures such as ResNets [24] and RNN encoder/decoders [25] are based on the fact that the hidden layers are given by a sequence of transformations that follow:

$$h_{k+1} = h_k + f(h_k, \theta) \tag{1.25}$$

where $k \in \{0, \dots, T-1\}$ is one of the T hidden-layers, $\theta \in \mathbb{R}^N_k$ are the parameters of the net and $h_k \in \mathbb{R}^n$ are the hidden-states of each hidden-layer with n neurons per layer. For this reason, RENs can be considered as discrete-time systems. A more general class of

RNN are given by the new architecture introduced as RENs by [11]:

$$\begin{bmatrix} x_{k+1} \\ v_k \\ y_k \end{bmatrix} = \overbrace{\begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix}}^{W} \begin{bmatrix} x_k \\ w_k \\ u_k \end{bmatrix} + \overbrace{\begin{bmatrix} b_x \\ b_v \\ b_y \end{bmatrix}}^{b}, \tag{1.26}$$

$$w_k = \sigma(v_k) \tag{1.27}$$

where $x_k \in \mathbb{R}^n$, $v_k \in \mathbb{R}^q$, $w_k \in \mathbb{R}^q$, $u_k \in \mathbb{R}^m$ and $y_k \in \mathbb{R}^p$ are respectively the state, the nonlinear output, the nonlinear input, the exogenous input and output. Moreover, the matrices in W, called *weights*, and the vectors in b, called *biases*, are the learnable parameters of the system. The function $\sigma(\cdot)$ is a nonlinear fixed "activation function" applied element-wise to each channel. Thus, RENs can be framed as a linear part (1.26), plus a nonlinear feedback (1.27). In Figure 1.3, we present a scheme of a REN where G denotes the linear part and σ the nonlinear one.

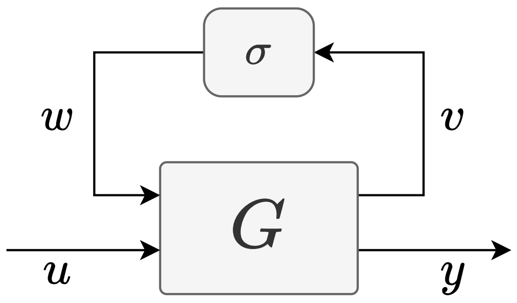


Figure 1.3: Recurrent Equilibrium Network(REN) schematized as a linear system(G) and a nonlinear static feedback(σ)

Once the model of the system has been given in (1.26)-(1.27), it is interesting to analyze how the system behaves if two initial conditions or input functions are considered. Consider the difference between two trajectories (x^a, w^a, v^a, u^a) and (x^b, w^b, v^b, u^b) of a REN, then we can denote with:

$$\Delta x_k = x_k^a - x_k^b\,, \quad \Delta v_k = v_k^a - v_k^b\,, \quad \Delta y_k = y_k^a - y_k^b\,, \quad \Delta u_k = u_k^a - u_k^b. \tag{1.28}$$

The two trajectories have been generated by (1.26)-(1.27) starting from two different initial conditions $a, b \in \mathbb{R}^n$ and from two input sequences u^a and u^b. At this point, it is possible to provide the *incremental form* of the REN:

$$\begin{bmatrix} \Delta x_{k+1} \\ \Delta v_k \\ \Delta y_k \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta w_k \\ \Delta u_k \end{bmatrix}, \tag{1.29}$$

$$\Delta w_k = \sigma(v_k + \Delta v_k) - \sigma(v_k), \tag{1.30}$$

We make the following assumption on σ, which is valid for most activation functions in the literature.

Assumption 1 (Rate-Limited σ)**.** The activation function $\sigma(\cdot)$ is piece-wise differentiable and slope-restricted in $[0, 1]$, i.e.,

$$0 \leq \frac{\sigma(y) - \sigma(x)}{y - x} \leq 1\,, \quad \forall x, y \in \mathbb{R}\,, \quad x \neq y \tag{1.31}$$

Note that (1.31) can also be rewritten for each j^{th} channel as a conic combination with multipliers $\xi_j > 0$, resulting in an incremental IQC:

$$\Gamma_k = \begin{bmatrix} \Delta v_k \\ \Delta w_k \end{bmatrix}^\top \begin{bmatrix} 0 & \Lambda \\ \Lambda & -2\Lambda \end{bmatrix} \begin{bmatrix} \Delta v_k \\ \Delta w_k \end{bmatrix} \geq 0\,, \quad \forall k \in \mathbb{N} \tag{1.32}$$

where $\Lambda = \text{diag}(\xi_1, \ldots, \xi_q)$. It is worth to highlight that the previous assumption is not restricting the use of the most popular activation functions used in ML literature, such as *ReLU*$(\cdot)$, *sigmoid*$(\cdot)$, *tanh*$(\cdot)$, since they do satisfy Assumption 1.
Subsequently a subclass of RENs will be presented, characterized by some assumptions made on the structure of the parameter D_{11}. Furthermore, direct parametrization of the contractive and robust RENs (called CREN and RREN, respectively) is obtained.

1.6.1. Acyclicity

A subset of RENs, as called by the authors in [11], are the *acyclic* RENs (aRENs). The name takes origin by the so called *directed acyclic graphs* in which there are no direct cycles. A directed cycle in a directed graph is a non-empty directed trail in which only the first and last vertices are equal (a directed graph is a graph in which the edges have a direction). The property of *acyclicity* can be enforced into the RENs assuming that the weight matrix D_{11} is always constrained to be strictly lower triangular; rephrasing it, it means that the i^{th} channel of $v(k)$ with $i \in [1, q]$ will depend only by the channels from $\{1, \dots (i-1)\}$. Assuming an acyclic grapch, allow to simplify the calculation of w_k at every layer/time k and consequently the simulation of x_k and y_k. However, this simplification is not necessary and different approaches can be used in order to solve the implicit equation [26, 27], resulting in a full matrix D_{11}.

1.6.2. Well-Posedness

Due to the presence of biases and of a nonlinear activation function σ, it is important to verify that the system is *well-posed*. *Well-posedness* means that, given any particular input $(\bar{x}_k, \bar{u}_k, \bar{b}_v)$, a unique solution $\bar{w}_k$ exists, where $(\bar{x}_k, \bar{u}_k, \bar{b}_v)$ correspond to all the inputs used to calculate $\bar{w}_k$. This property can be reformulated as follows.

Definition 1.6.1 (Well-posedness)**.** An equation in the form:

$$w_k = \sigma(v_k) = \sigma(D_{11}w_k + C_1x_k + D_{12}u_k + b_v) \tag{1.33}$$

is said to be *well-posed* if and only if:

$$\forall \bar{x}_k, \bar{u}_k, \bar{b}_v \quad \text{then} \quad \exists!\, \bar{w}_k \ :\ \bar{w}_k = \sigma\big(D_{11}\bar{w}_k + C_1\bar{x}_k + D_{12}\bar{u}_k + \bar{b}_v\big) \tag{1.34}$$

In [26] the authors show that if there exists a positive definite diagonal matrix Λ such that:

$$2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda > 0, \tag{1.35}$$

then the problem in (1.33) is well-posed. This important result will be used in the next section and in Chapter 2.

1.6.3. Learning Stable & Robust Models

One of the main results obtained in [11] is the possibility to obtain a set of learnable parameters (i.e., the matrix W and the vectors in b in (1.26)) such that the RENs can be contractive (called *C-RENs*) or robust (called *R-RENs*). With *robustness*, we mean the system's sensitivity of the model's output is small with respect to small perturbations in the input. This problem has been recently noticed in many different RNN models [12], making the use of standard RNN impractical in control theory, in which small perturbation in data are quite common. For sake of clarity, the propositions guaranteeing the necessary conditions for contractivity and robustness are reported here.

Proposition 1.1 (LMI for C-REN)**.** *A discrete-time REN in the form* (1.26)-(1.27) *is* *well-posed and contracting if there exists a matrix $P > 0$ and a diagonal matrix $\Lambda > 0$ such that*

$$\begin{bmatrix} P & -C_1^\top \Lambda \\ -\Lambda C_1 & W \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix}^\top > 0, \tag{1.36}$$

where

$$W = 2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda. \tag{1.37}$$

Proposition 1.2 (LMI for R-REN). *Given the matrices (Q, S, R), a discrete-time REN in the form* (1.26)-(1.27) *is well-posed and satisfies the incremental IQC described by (Q, S, R) if there exists a matrix $P > 0$ and a diagonal matrix $\Lambda > 0$ such that*

$$\begin{bmatrix} P & -C_1^\top \Lambda & C_2^\top S^\top \\ -\Lambda C_1 & W & D_{21}^\top S^\top - \Lambda D_{12} \\ SC_2 & SD_{21} - D_{12}^\top \Lambda & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \\ B_2^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \\ B_2^\top \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top > 0 \tag{1.38}$$

where W is defined in (1.37)*.*

The proofs of Propositions 1.1 and 1.2 can be found in the authors' paper[11].

1.6.4. Convex Parametrization

It is possible to notice that the matrix inequalities in (1.36), (1.38), are not convex in their parameters (e.g., the presence of the element $A^\top PA$). This detail would make not possible later a direct parametrization. For this reason, as reported in [23], a convex parametrization of C-RENs/R-RENs is done. First of all, the REN's model (1.26) can be rewritten multiplying the first two rows by two matrices (E, Λ), with E invertible and Λ positive definite and diagonal.

$$\begin{gathered} \begin{bmatrix} Ex_{k+1} \\ \Lambda v_k \\ y_k \end{bmatrix} = \begin{bmatrix} F & \mathcal{B}_1 & \mathcal{B}_2 \\ \mathcal{C}_1 & \mathcal{D}_{11} & \mathcal{D}_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} x_k \\ w_k \\ u_k \end{bmatrix} + \tilde{b}, \\ w_k = \sigma(v_k), \end{gathered} \tag{1.39}$$

where $\mathcal{F} = EA$, $\mathcal{B}_1 = EB_1$, $\mathcal{B}_2 = EB_2$, $\mathcal{C}_1 = \Lambda C_1$, $\mathcal{D}_{11} = \Lambda D_{11}$, $\mathcal{D}_{12} = ED_{12}$, and $\tilde{b}$ is the column concatenation of Eb_x, Λb_v, b_y. The passage from the initial matrices of (1.26) to the new ones (and vice versa) is guaranteed by the invertibility of E and Λ ($\Lambda > 0$). Thus, Propositions 1.1 and 1.2 can also holds for (1.39).

Proposition 1.3 (Convex Parametrization of C-REN). *Let a REN be in the form* (1.39), *then if there exists a matrix* $\mathcal{P} > 0$ *such that:*

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} > 0, \tag{1.40}$$

with

$$\mathcal{W} = 2\Lambda - \mathcal{D}_{11} - \mathcal{D}_{11}^T, \tag{1.41}$$

then the REN is well-posed and contractive.

Proposition 1.4 (Convex Parametrization of R-REN). *Let a REN be in the form* (1.39), *then, given the matrices* (Q, S, R), *if exists a matrix* $\mathcal{P} > 0$ *such that:*

$$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & C_2^\top S^\top \\ -\mathcal{C}_1 & \mathcal{W} & D_{21}^\top S^\top - \mathcal{D}_{12} \\ SC_2 & SD_{21} - \mathcal{D}_{12}^\top & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} - \begin{bmatrix} F^\top \\ \mathcal{B}_1^\top \\ \mathcal{B}_2^\top \end{bmatrix} \mathcal{P}^{-1} \begin{bmatrix} F^\top \\ \mathcal{B}_1^\top \\ \mathcal{B}_2^\top \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top > 0, \tag{1.42}$$

with $\mathcal{W}$ *given by* (1.41), *then the REN is well-posed and satisfies the incremental IQCs characterized by* (Q, S, R).

The proofs of Propositions 1.3 and 1.4 can be found in [11].

1.6.5. Direct Parametrization

Now, it is finally possible to obtain a *direct parametrization* of the RENs, i.e., find an unconstrained mapping from the free/learnable parameters $\theta \in \mathbb{R}^N$ to the matrices W and b from (1.26) that guarantees contractivity or robustness of the model.

Contractive Direct Parametrization

Let the matrix from the left handside of inequality (1.40) be H, with $H \in \mathbb{R}^{(2n+q)\times(2n+q)}$. Then, for any matrix $X \in \mathbb{R}^{(2n+q)\times(2n+q)}$ and an $\epsilon > 0$, imposing that:

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} = X^\top X + \epsilon I \,, \tag{1.43}$$

makes H satisfy the inequality (1.40) by construction. Comparing (1.43) with (1.40), leads to:

$$F = H_{31} \,, \quad \mathcal{B}_1 = H_{32} \,, \quad \mathcal{P} = H_{33} \,, \quad \mathcal{C}_1 = -H_{21}. \tag{1.44}$$

Additionally we can set:

$$E = \frac{1}{2}(H_{11} + \mathcal{P} + Y_1 - Y_1^\top) \tag{1.45}$$

where $Y_1 \in \mathbb{R}^{n\times n}$ is a free matrix that allows us to obtain also non-symmetric E matrices. If the system is *acyclic*, then $\mathcal{D}_{11}$ must be strictly lower triangular, thus from H_{22}:

$$H_{22} = \mathcal{W} = 2\Lambda - \mathcal{D}_{11} - \mathcal{D}_{11}^\top. \tag{1.46}$$

It is possible to obtain Λ from $1/2$ of the main diagonal of $\mathcal{W}$ and $\mathcal{D}_{11}$ from the strict lower triangular part of $\mathcal{W}$. The remaining parameters do not impact on the contractivity of the model and can be considered free. In conclusion, to define a CREN, first one can set $X \in \mathbb{R}^{(2n+q)\times(2n+q)}, \mathcal{B}_2 \in \mathbb{R}^{n\times m}, C_2 \in \mathbb{R}^{p\times n}, \mathcal{D}_{12} \in \mathbb{R}^{q\times m}, D_{21} \in \mathbb{R}^{p\times q}, D_{22} \in \mathbb{R}^{p\times m}, b \in \mathbb{R}^{(n+q+p)}$ and $Y_1 \in \mathbb{R}^{n\times n}$. Then, the remaining matrices are obtained through (1.44)-(1.46). Note that a possible implementation in case of full $\mathcal{D}_{11}$ can be carried out. In order to do so, the full matrix must be constructed as per:

$$\mathcal{D}_{11} = \Lambda - \frac{1}{2}(H_{22} + Y_2 - Y_2^\top), \tag{1.47}$$

where $Y_2 \in \mathbb{R}^{q\times q}$, $\Lambda = e^{diag(g)}$ and $g \in \mathbb{R}^q$.

Robust Direct Parametrization

We introduce an intermediate step that will be helpful for the direct parametrization of robust RENs. By defining:

$$\mathcal{R} = R + SD_{22} + D_{22}^\top S^\top + D_{22}^\top Q D_{22} > 0, \tag{1.48}$$

It can be shown that the inequality (1.42) is equivalent to:

$$\begin{bmatrix} E+E^\top-\mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} > \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top - \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top , \tag{1.49}$$

with

$$\mathcal{C}_2 = (D_{22}^\top Q + S)C_2 \,, \quad \mathcal{D}_{21} = (D_{22}^\top Q + S)D_{21} - \mathcal{D}_{12}^\top \tag{1.50}$$

The equivalence between ((1.42)) and (1.48)-(1.49) is based on the use of *Schur Complement*. The detailed proof can be found in [11].
However, it can be shown that, in case of the given matrix S to be null, then (1.48) becomes:

$$\mathcal{R} = R + D_{22}^\top Q D_{22} > 0, \tag{1.51}$$

Moreover, doing a direct parametrization as in the conctractive would imply solving (1.51) in D_{22} given a fixed value of $\mathcal{R}$. This may lead to complex solutions, not acceptable for real signals. Then, it is necessary to find a construction of D_{22} such that Equation (1.48) is satisfied, for any S. To do so, we can define:

$$s = \max(p, m) \,, \quad X_3, Y_3 \in \mathbb{R}^{s\times s} \tag{1.52}$$

$$M = X_3^\top X_3 + Y_3 - Y_3^\top + \epsilon I \tag{1.53}$$

$$Z = \Big[(I-M)(I+M)^{-1}\Big]_{p\times m} \tag{1.54}$$

where $\epsilon > 0$ and $[V]_{p\times m}$ indicates the block matrix with the first p rows and m columns of a matrix V. Assuming $Q < 0$, Q and R can be factorized as:

$$L_Q^\top L_Q = -Q \,, \quad L_R^\top L_R = R - SQ^{-1}S^\top \tag{1.55}$$

and finally:

$$D_{22} = -Q^{-1}S^\top + L_Q^{-1} Z L_R \tag{1.56}$$

Constructing D_{22} as per (1.56), guarantees that the matrix $\mathcal{R}$ is positive definitive (i.e., (1.48) holds true). The proof can be found in [11].
Now, the direct parametrization in the robust case proceeds in the same way as done in the contractive case.
Denoting with $\tilde{d} = (n+q+m)$, let $H^* \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ be the resulting matrix from the inequal-

ity (1.49):

$$H^* = \begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & C_2^\top S^\top \\ -\mathcal{C}_1 & \mathcal{W} & D_{21}^\top S^\top - \mathcal{D}_{12} \\ SC_2 & SD_{21} - \mathcal{D}_{12}^\top & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} - \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top . \tag{1.57}$$

Additionally, let H be defined as:

$$H = \begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & C_2^\top S^\top \\ -\mathcal{C}_1 & \mathcal{W} & D_{21}^\top S^\top - \mathcal{D}_{12} \\ SC_2 & SD_{21} - \mathcal{D}_{12}^\top & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} . \tag{1.58}$$

Combining (1.49) and (1.58) we have:

$$H^* = H - \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top \tag{1.59}$$

Then, for any matrix $X \in \mathbb{R}^{\tilde{d} \times \tilde{d}}$ and an $\epsilon > 0$:

$$H^* = X^\top X + \epsilon I > 0 . \tag{1.60}$$

Finally, using (1.59) and (1.60):

$$H = \underbrace{XX^\top + \epsilon I}_{H^*} + \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \mathcal{C}_2^\top \\ \mathcal{D}_{21}^\top \\ \mathcal{B}_2 \end{bmatrix}^\top - \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} > 0 . \tag{1.61}$$

Now, it is possible to follow the same partitions done for C-REN (using (1.44)-(1.46)) and obtain a direct parametrization of a R-REN, given the matrices (Q, S, R).

1.7. Neural-ODEs

Based on studies about relations between neural networks and differential equations [28, 29], Neural Ordinary Differential Equations (Neural-ODEs) have been introduced for the first time in [20]. The main idea about this new type of deep neural network is that, differently from the classical neural networks used in literature, the number of discrete sequences of the hidden layers is not specified; instead, the derivative of the hidden states using a neural network is parameterized (Figure 1.4). Normally, the hidden states inside general networks (e.g., ResNets) follow an evolution as in Equation (1.25), reported here for clarity:

$$h_{k+1} = h_k + f(h_k, \theta_k)$$

Afterwards, the numerical computation of this evolution depends on the considered model's architecture(for few cases, see Table 1.1). If for example, using Forward Euler(FE), infinitesimal steps are considered, Equation (1.25) can be rewritten in a *differential form*:

$$\frac{d}{dt}h(t) = f\big(h(t), t, \theta\big) \tag{1.62}$$

At this point the ODE in (1.62) can be solved using optimized and efficient black-box differential equation solvers. However, in order to perform the weights' updates during the training phase, the reverse-mode differentiation (a.k.a. *back-propagation step*) must be carried out through the use of a special *adjoint method* for ODEs. The adjoint method was introduced for the first time in [30] and it benefits of different advantages: it scales linearly with the problem size, it has low memory cost, and it explicitly controls numerical error. We will detailed these advantages in Section 1.7.1. The use of black-box ODE solvers can bring multiple benefits with respect to the traditional method of differentiating through the operations of the forward pass. These advantages are reported here for sake of clarity:

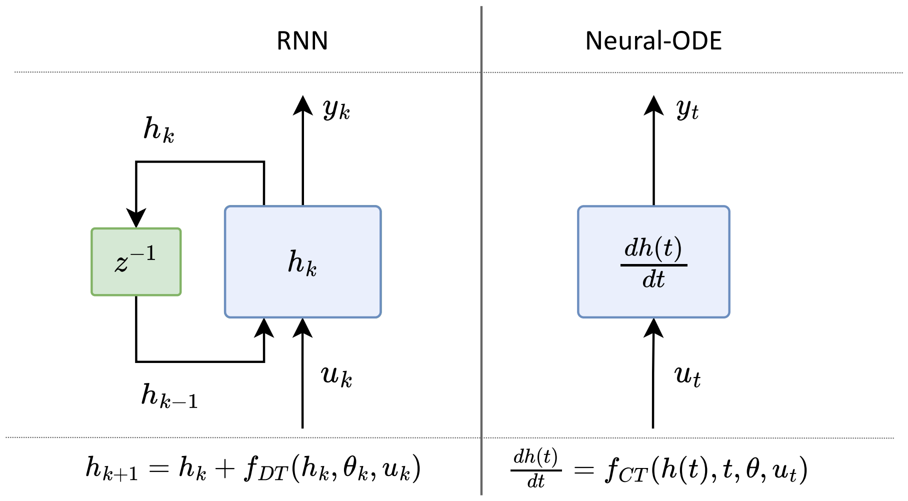

RNN
Neural-ODE
h_k
y_k
z^{-1}
h_k
u_k
h_{k-1}
$h_{k+1} = h_k + f_{DT}(h_k, \theta_k, u_k)$
y_t
$\frac{dh(t)}{dt}$
u_t
$\frac{dh(t)}{dt} = f_{CT}(h(t), t, \theta, u_t)$

Figure 1.4: Comparison of the representations of the hidden-layer between a generic RNN(left) and a Neural-ODE(right). The unitary delay is represented here with z^{-1}.

- **Memory efficiency** During the computation of the gradients of a scalar-valued loss

Neural Network	**Fixed-Step Numerical Scheme**
ResNet[24], ResNeXt[31], ...	Forward Euler
PolyNet[32]	Approx. to Backward Euler
FractalNet[33]	Runge-Kutta
DenseNet[34]	Runge-Kutta

Table 1.1: Interpretation of classical deep learning networks as ODE solvers.

with respect to all inputs of any ODE solver, it is not required to back-propagate through the operations of the solver (for details, see Section 1.7.1). Intermediate quantities of the forward pass don't need to be stored; thus, the memory cost to train the models is kept constant as a function of depth. This problem was one of the major bottlenecks of training deep models.

- **Adaptive computation** During the past few centuries many different ODE solvers have been developed, way more accurate then the simple Euler's method, achieving high level of precision and efficiency[35–37]. Modern ODE solvers can provide properties such as monitoring the level of the error, growth of the approximation error and adaptation of the evaluation strategy on the fly to achieve the requested level of accuracy. Thus, the cost of evaluating the cost of the model can be scaled functionally to the problem complexity. Indeed, after the training phase, the accuracy of the returned model can be reduced for real-time or low-power applications.
- **Continuous time-series models** Differently from RNNs, that are based on fixed-intervals observations and emissions, the continuous-defined dynamics of Neural-ODEs can automatically include data arriving at arbitrary times. This can bring high benefits in the control field, due to the presence of possible time-varying delays in the communications between devices.

1.7.1. Adjoint Method

Let's denote with $L(z(t))$ the loss/cost function evaluated on the sample $z(t)$ and with $\theta \in \mathbb{R}^N$ the parameters of the neural network. During the training phase, in order to update the weights for each batch of data, optimization techniques require the computation of the gradient of the loss function L with respect to the parameters θ of the network.
In formula:

$$\nabla_\theta L = \frac{\partial L}{\partial \theta} \tag{1.63}$$

Considering a continuous system governed by Equation (1.62), the evolution between two time instants t_0 and t_f can be made explicit:

$$z(t_1) = z(t_0) + \int_{t_0}^{t_f} f(z(t), t, \theta) dt \tag{1.64}$$

Thus, the calculus of the loss $L(z(\cdot))$ at time t_f can be rewritten as:

$$\begin{aligned} L(z(t_f)) &= L\Big(z(t_0) + \int_{t_0}^{t_f} f(z(t), t, \theta) dt \Big) \\ &= L\Big(\text{ODESOLVE}\big(f(\cdot), z(t_0), [t_0, t_f], \theta \big) \Big) \end{aligned} \tag{1.65}$$

Let's assume at first that the error depends only on the last time instant t_f. This assumption will be removed later. Hence a first call to the black-box ODE solver is done to compute $z(t_f)$ and then $L(z(t_f))$. At this point, the traditional back-stepping method would require differentiating through the operations. However, this can be done in a way more efficient way using the *adjoint method* firstly presented in [30].
In order to evaluate the gradient of L w.r.t. θ, the sensitivity of the loss function w.r.t. $z(t)$ is needed, i.e., $\partial L / \partial z$. Let's denote it as the *adjoint state* $a(t)$. In formula:

$$a(t) = \frac{\partial L}{\partial z(t)} \tag{1.66}$$

It can be shown that the evolution of $a(t)$:

$$\frac{d}{dt} a(t) = -a(t)^{\top} \frac{\partial f(z(t), t, \theta)}{\partial z} \tag{1.67}$$

The proof of Equation (1.67) can be found in [20] (Hint: it is obtained using the chain rule and the definition of derivative, i.e. $\frac{da(t)}{dt} = \lim_{\epsilon \to 0^+} \big(\frac{a(t+\epsilon) - a(t)}{\epsilon} \big) \big)$. However, for Equation (1.67), it is required to know the values of z(t) along its entire trajectory. But, instead of computing them forward, they can be obtained together backward with the adjoint evolution, starting from the final value $z(t_f)$. After that, computing the gradients with respect to θ requires evaluating the integral depending on both $z(t)$ and $a(t)$:

$$\frac{dL}{d\theta} = -\int_{t_f}^{t_0} a(t)^{\top} \frac{\partial f(z(t), t, \theta)}{\partial \theta} dt \tag{1.68}$$

As for Equation (1.67), the proof of Equation (1.68) can be found in the authors' paper[20][Appendix B.2]. Thus, for the backward-step, only one call to the black-box ODE solver for an augmented variable $s(\cdot)$ must be done (see Algorithm 1.1).

Algorithm 1.1 Backward Propagation of a *adjoint*-ODE solver

```
Input: param. $\theta$, start time $t_0$, final time $t_f$, final state $z(t_f)$, loss gradient $\partial L/\partial z(t_f)$
  $s_0 = [z(t_f), \frac{\partial L}{\partial z(t_f)}, 0_{|\theta|}]$        ▷ Define initial augmented state
  def aug_dynamics($[z(t), a(t), \cdot], t, \theta$):        ▷ Define dynamics on augmented state
      return $[f(z(t), t, \theta), -a(t)^\top \frac{\partial f}{\partial z}, -a(t)^\top \frac{\partial f}{\partial \theta}]$        ▷ Compute vector-Jacobian products
  $[z(t_0), \frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}]$ = ODESolve($s_0$, aug_dynamics, $t_f, t_0, \theta$)        ▷ Solve reverse-time ODE
return $\frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}$        ▷ Return gradients
```

What if the loss function $L(\cdot)$ depends also from N intermediate steps of $z(\cdot)$ between the time instants t_0 and t_f? Then the interval $[t_0, t_f]$ must be broken into N different sub-problems and then each parts must be solved individually backward, starting from the last one $[t_N, t_f]$ up to the first one $[t_0, t_1]$. A schematical representation of it is reported in Figure 1.5. A study case of loss function is the system identification of a dynamical model. Indeed, the loss function is described as the *mean squared error*(MSE) between the N-measured states $[z(t_1), \ldots, z(t_N)]$ and the N-simulated states $[\hat{z}(t_1), \ldots, \hat{z}(t_N)]$. An application of it will be provided in Section 3.4.

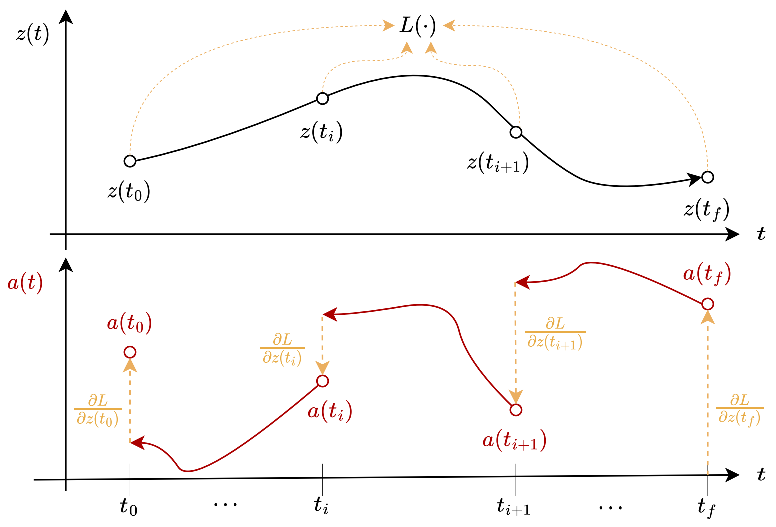

$z(t)$
$L(\cdot)$
$z(t_i)$
$z(t_0)$
$z(t_{i+1})$
$z(t_f)$
t
$a(t)$
$a(t_f)$
$a(t_0)$
$\frac{\partial L}{\partial z(t_{i+1})}$
$\frac{\partial L}{\partial z(t_i)}$
$\frac{\partial L}{\partial z(t_0)}$
$a(t_i)$
$\frac{\partial L}{\partial z(t_f)}$
$a(t_{i+1})$
t
t_0 $\cdots$ t_i t_{i+1} $\cdots$ t_f

Figure 1.5: Representation of the evolution of the augmented states in a *adjoint*-ODE solver. In this case, the loss function depends from the $z(t_0), z(t_f)$ and also from the values at intermediate steps.

1.7.2. Augmented Neural-ODEs

Despite the new capabilities of Neural-ODEs and its great promises on a number of tasks including modeling continuous time data and building normalizing flows with low computational cost [20, 38], many different limitations of this advanced deep neural network have been shown[39]. The authors of [39], however, have introduced a simple extension of the Neural-ODEs: Augmented Neural-ODEs (ANODEs). The brilliant idea is to enlarge the number of state of the net model. These new "degrees of freedom" allow the system to learn approximation of (sometimes even simple) functions, impossible to achieve with standard Neural-ODEs. For the proof of the impossibility of Neural-ODE to learn some functions, the reader is invited to the authors' paper [39]. From it, an example of function $g(z)$ that cannot be approximated is the following:

$$g(z) = \begin{cases} -1 & \text{if} \quad \|z\| \leq r_1, \\ +1 & \text{if} \quad r_1 \leq \|z\| \leq r_2 \end{cases} \tag{1.69}$$

where $\|\cdot\|$ is the Euclidean norm, $r_1, r_2, r_3 \in \mathbb{R}$ and $0 < r_1 < r_2 < r_3$. Additionally, another limitation of standard Neural-ODEs, is the computation burden when the flow gets complex: this is caused by the number of steps required by ODE solvers raising [20, 38]. Instead, as ANODEs learn simpler flows, they would presumably require fewer iterations to compute, without causing any increase in complexity. A graphical representation of $g(z)$ is reported in Figure 1.6, with $z \in \mathbb{R}^2$. It can be shown that the ANODEs are able to learn an approximation of $g(z)$. For this reason, from now on, the architecture introduced in this thesis, i.e., *REN-ODE*, is going to have an enlarged number of states w.r.t. the original data. Indeed, in Section 3.3 the mapping of function $g(z)$ will be obtained.

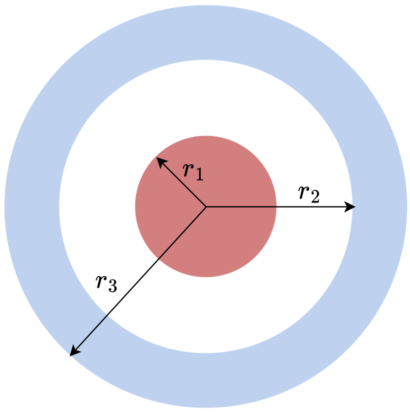


Figure 1.6: Graphical representation of function $g(z)$, with $z \in \mathbb{R}^2$.

2 | REN-ODEs

The contribution of this work is the introduction of a new deep neural network model: REN-ODE. As the name suggests, it consists in a particular architecture, capable of providing a direct parametrization with guarantees of stability (like a REN), while being in continuous time, and thus, trainable through the use of black-box ODE solvers (like a Neural-ODE). In this chapter the model of the system will be provided. Then, through similar steps as done in Section 1.6, direct parametrization for contractive and robust models (respectively defined as CREN-ODE and RREN-ODE) will be given.

2.1. REN-ODE model

REN-ODE shares similar structure to the original REN[11]. The system model is the following:

$$\begin{bmatrix} \dot{x}_t \\ v_t \\ y_t \end{bmatrix} = \overbrace{\begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix}}^{W} \begin{bmatrix} x_t \\ w_t \\ u_t \end{bmatrix} + \overbrace{\begin{bmatrix} b_x \\ b_v \\ b_y \end{bmatrix}}^{b}, \tag{2.1}$$

$$w_t = \sigma(v_t) \tag{2.2}$$

where $x_t \in \mathbb{R}^n$, $v_t \in \mathbb{R}^q$, $w_t \in \mathbb{R}^q$, $u_t \in \mathbb{R}^m$ and $y_t \in \mathbb{R}^p$ are respectively the state, the nonlinear output, the nonlinear input, the exogenous input and the linear output. Also in REN-ODE, $\sigma(\cdot)$ is a nonlinear "activation function" applied element-wise. Additionally, the *incremental form* of the system (for the definition, see Section 1.2.2) can be provided:

$$\begin{bmatrix} \Delta\dot{x}_t \\ \Delta v_t \\ \Delta y_t \end{bmatrix} = \begin{bmatrix} A & B_1 & B_2 \\ C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta w_t \\ \Delta u_t \end{bmatrix} \tag{2.3}$$

$$\Delta w_t = \sigma(v_t + \Delta v_t) - \sigma(v_t) \tag{2.4}$$

where the *finite* difference between two possible trajectories (x, w, v, u) and $(\bar{x}, \bar{w}, \bar{v}, \bar{u})$ of the system has been considered:

$$\Delta x_t = x_t^a - x_t^b\,, \quad \Delta v_t = v_t^a - v_t^b\,, \quad \Delta y_t = y_t^a - y_t^b\,, \quad \Delta u_t = u_t^a - u_t^b \tag{2.5}$$

2.1.1. Assumptions

In order to achieve properties of contractivity and robustness, some assumptions must be introduced. Indeed, as in the discrete case, also here the activation function $\sigma(\cdot)$ must be rate-limited and piece-wise differentiable (Assumption 1). However, the inequality (1.31) can be also rewritten in continuous time for each j^{th} channel as a conic combination with multipliers $\xi_j > 0$, resulting in an *incremental* IQC:

$$\Gamma_t = \begin{bmatrix} \Delta v_t \\ \Delta w_t \end{bmatrix}^\top \begin{bmatrix} 0 & \Lambda \\ \Lambda & -2\Lambda \end{bmatrix} \begin{bmatrix} \Delta v_t \\ \Delta w_t \end{bmatrix} \geq 0\,, \quad \forall t \in \mathbb{R} \tag{2.6}$$

where $\Lambda = \text{diag}(\xi_1, \dots, \xi_q)$. Δv_t and Δw_t represent the difference between the two trajectories having (v_t, w_t) and $(\bar{v}_t, \bar{w}_t)$.

2.1.2. Well-Posedness

The concept of *well-posedness* has already been established in Section 1.6.2. However, despite the new REN-ODE being in continuous time, the formulation does not change and, thus, it is still valid. Indeed, finding a positive definite diagonal matrix Λ such that:

$$2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda > 0 \tag{2.7}$$

makes the problem $w_t = \sigma(v_t)$ well-posed (proof in [26]).

2.2. Contracting and Robust REN-ODEs

In this section, LMIs in order to verify properties of contractivity and robustness of REN-ODEs are going to be provided.

Theorem 2.1 (Contractive REN-ODE). *A REN-ODE in the form* (2.1)-(2.2) *is con-*

tracting if there exists a matrix $P > 0$ and a diagonal matrix $\Lambda > 0$ such that:

$$\begin{bmatrix} -A^\top P - PA & -C_1^\top \Lambda - PB_1 \\ -\Lambda C_1 - B_1^\top P & W \end{bmatrix} > 0, \tag{2.8}$$

with:

$$W = 2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda. \tag{2.9}$$

Proof. By left-multiplying and right-multiplying the inequality (2.8) with $\left[\begin{smallmatrix} \Delta x_t^\top & \Delta w_t^\top \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} \Delta x_t \\ \Delta w_t \end{smallmatrix}\right]$ respectively:

$$\begin{bmatrix} \Delta x_t^\top & \Delta w_t^\top \end{bmatrix} \begin{bmatrix} -A^\top P - PA & -C_1^\top \Lambda - PB_1 \\ -\Lambda C_1 - B_1^\top P & W \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta w_t \end{bmatrix} > 0. \tag{2.10}$$

Then:

$$\begin{aligned} &\Delta x_t^\top (-A^\top P - PA)\Delta x_t + \Delta w_t^\top (-\Lambda C_1 - B_1^\top P)\Delta x_t \\ &\qquad + \Delta x_t^\top (-C_1^\top \Lambda - PB_1)\Delta w_t + \Delta w_t^\top (W)\Delta w_t > 0. \end{aligned} \tag{2.11}$$

Rearranging the inequality:

$$\begin{aligned} &-\overbrace{(A\Delta x_t + B_1 \Delta w_t)^\top}^{=\Delta \dot{x}_t^\top} P\Delta x_t - \Delta x_t^\top P \overbrace{(A\Delta x_t + B_1\Delta w_t)}^{=\Delta \dot{x}_t} \\ &\qquad - 2\Delta w_t^\top \Lambda C_1 \Delta x_t + \Delta w_t^\top W \Delta w_t^\top W \Delta w_t > 0. \end{aligned} \tag{2.12}$$

Using the definition of $\Delta \dot{x}_t$ from (2.3) in (2.12), with $(\Delta u = 0)$[1], we obtain:

$$-\Delta \dot{x}_t^\top P \Delta x_t - \Delta x_t^\top P \Delta \dot{x}_t - 2\Delta w_t^\top \Lambda C_1 \Delta x_t + \Delta w_t^\top W \Delta w_t \quad > 0. \tag{2.13}$$

At this point, we can define a function $V_\Delta(t)$ as:

$$V_\Delta(t) = \Delta x_t^\top P \Delta x_t \quad \rightarrow \quad \dot{V}_\Delta(t) = \Delta \dot{x}_t^\top P \Delta x_t + \Delta x_t^\top P \Delta \dot{x}_t \tag{2.14}$$

And, thus, plugging (2.14) in (2.13):

$$-V_\Delta(t) - 2\Delta w_t^\top \Lambda C_1 \Delta x_t + \Delta w_t^\top W \Delta w_t \ > 0. \tag{2.15}$$

[1]The study of contractivity considers two trajectories with different initial conditions, but same input sequences, i.e., $\Delta u = 0$

After that, the definition of W can be used (2.9):

$$-V_\Delta(t) - 2\Delta w_t^\top \Lambda C_1 \Delta x_t + \Delta w_t^\top (2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda)\Delta w_t \; > 0. \tag{2.16}$$

Then, rearranging and using the definition of $\dot{V}_\Delta$:

$$-\dot{V}_\Delta(t) \; > \; 2\Delta w_t^\top \Lambda \overbrace{(C_1 \Delta x_t + D_{11}\Delta w_t)}^{=\Delta v_t} - 2\Delta w_t^\top \Lambda \Delta w_t. \tag{2.17}$$

Using the definition of $\Delta v_t^\top$ from (2.1) in (2.17) and remembering the definition of Γ_t (2.6):

$$-\dot{V}_\Delta(t) \; > \; 2\Delta w_t^\top \Lambda \Delta v_t - 2\Delta w_t^\top \Lambda \Delta w_t \; = \; \Gamma_t. \tag{2.18}$$

Finally:

$$\dot{V}_\Delta(t) \; < -\Gamma_t \leq 0. \tag{2.19}$$

The function $V_\Delta(t)$ can be chosen as a valid Lyapunov function for the incremental system in (2.3)-(2.4) and thus, if there exists a matrix $P > 0$, then:

$$\begin{aligned} V_\Delta(t) &> 0\,, \\ \dot{V}_\Delta(t) &< 0. \end{aligned} \tag{2.20}$$

Since $V_\Delta(t)$ is a quadratic form in the vector $\left[\begin{smallmatrix} \Delta x_t^\top & \Delta w_t^\top \end{smallmatrix}\right]^\top$ then it follows that:

$$\dot{V}_\Delta(t) \leq -\alpha V_\Delta(t) \tag{2.21}$$

with $\alpha > 0$ (the proof is reported in Appendix B). The relation in (2.21), with (2.20), guarantees, thanks to the Lyapunov exponential stability theorem, that the system is globally exponentially stable, i.e.,

$$\|\Delta x(t)\| \leq \kappa e^{-c(t-t_0)}\|a - b\| \tag{2.22}$$

with $c = \frac{\alpha}{2} > 0$, that is the Definition 1.1.1 of a contractive system. $\square$

Theorem 2.2 (Robust REN-ODE). *A REN-ODE in the form* (2.1)-(2.2) *is well-posed and satisfies the incremental IQC described by* (Q, S, R) *if there exists a matrix* $P > 0$

and a diagonal matrix $\Lambda > 0$ *such that:*

$$\begin{bmatrix} -A^\top P - PA & -PB_1 - C_1^\top \Lambda & -PB_2 + C_2^\top S^\top \\ -B_1^\top P - \Lambda C_1 & W & D_{21}^\top S^\top - \Lambda D_{12} \\ -B_2^\top P + SC_2 & SD_{21} - D_{12}^\top \Lambda & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top > 0 \tag{2.23}$$

with:

$$W = 2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda. \tag{2.24}$$

Proof. By left-multiplying and right-multiplying the inequality (2.23) with $\begin{bmatrix} \Delta x_t^\top & \Delta w_t^\top & \Delta u_t^\top \end{bmatrix}$ and $\begin{bmatrix} \Delta x_t^\top & \Delta w_t^\top & \Delta u_t^\top \end{bmatrix}^\top$ respectively:

$$\begin{aligned} &\begin{bmatrix} \Delta x_t \\ \Delta w_t \\ \Delta u_t \end{bmatrix}^\top \begin{bmatrix} -A^\top P - PA & -PB_1 - C_1^\top \Lambda & -PB_2 + C_2^\top S^\top \\ -B_1^\top P - \Lambda C_1 & W & D_{21}^\top S^\top - \Lambda D_{12} \\ -B_2^\top P + SC_2 & SD_{21} - D_{12}^\top \Lambda & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta w_t \\ \Delta u_t \end{bmatrix} + \\ &+ \begin{bmatrix} \Delta x_t \\ \Delta w_t \\ \Delta u_t \end{bmatrix}^\top \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top \begin{bmatrix} \Delta x_t \\ \Delta w_t \\ \Delta u_t \end{bmatrix} > 0 \end{aligned} \tag{2.25}$$

Then:

$$\begin{aligned} &-\Delta x_t^\top A^\top P \Delta x_t - \Delta w_t^\top B_1^\top P \Delta x_t - \Delta u_t^\top B_2^\top P \Delta x_t - \Delta x_t^\top PA \Delta x_t \\ &-\Delta x_t^\top PB_1 \Delta w_t - \Delta x_t^\top PB_2 \Delta u_t - 2\Delta x_t^\top C_1^\top \Lambda \Delta w_t + 2\Delta x_t^\top C_2^\top S^\top \Delta u_t \\ &+\Delta w_t^\top W \Delta w_t + 2\Delta w_t^\top (D_{21}^\top S^\top - \Lambda D_{12}) \Delta u_t + \Delta u_t^\top (R + SD_{22} + D_{22}^\top S^\top) \Delta u_t \\ &+(\Delta x_t^\top C_2^\top + \Delta w_t^\top D_{21} + \Delta u_t^\top D_{22}^\top)\, Q\, (C_2 \Delta x_t + D_{21} \Delta w_t + D_{22} \Delta u_t) > 0 \end{aligned} \tag{2.26}$$

Using the definition of $\Delta \dot{x}_t$ from (2.3) in (2.26), we obtain:

$$\begin{aligned} &-\Delta \dot{x}_t^\top P \Delta x_t - \Delta x_t^\top P \Delta \dot{x}_t - 2\Delta x_t^\top C_1^\top \Lambda \Delta w_t + 2\Delta x_t^\top C_2^\top S^\top \Delta u_t \\ &+\Delta w_t^\top W \Delta w_t + 2\Delta w_t^\top (D_{21}^\top S^\top - \Lambda D_{12}) \Delta u_t + \Delta u_t^\top (R + SD_{22} + D_{22}^\top S^\top) \Delta u_t \\ &+(\Delta x_t^\top C_2^\top + \Delta w_t^\top D_{21} + \Delta u_t^\top D_{22}^\top)\, Q\, (C_2 \Delta x_t + D_{21} \Delta w_t + D_{22} \Delta u_t) > 0 \end{aligned} \tag{2.27}$$

At this point, we can define a function $V_\Delta(t)$ as:

$$V_\Delta(t) = \Delta x_t^\top P \Delta x_t \quad \rightarrow \quad \dot{V}_\Delta(t) = \Delta \dot{x}_t^\top P \Delta x_t + \Delta x_t^\top P \Delta \dot{x}_t \tag{2.28}$$

Thus, using (2.28) in (2.27):

$$\begin{gathered}-\dot{V}_\Delta(t) - 2\Delta x_t^\top C_1^\top \Lambda \Delta w_t + 2\Delta x_t^\top C_2^\top S^\top \Delta u_t + \Delta w_t^\top W \Delta w_t \\ +2\Delta w_t^\top (D_{21}^\top S^\top - \Lambda D_{12})\Delta u_t + \Delta u_t^\top (R + SD_{22} + D_{22}^\top S^\top)\Delta u_t \\ +(\Delta x_t^\top C_2^\top + \Delta w_t^\top D_{21} + \Delta u_t^\top D_{22}^\top)\, Q\, (C_2 \Delta x_t + D_{21}\Delta w_t + D_{22}\Delta u_t) > 0\end{gathered} \tag{2.29}$$

Considering the definition of Δy_t from (2.3), (2.29) becomes:

$$\begin{gathered}-\dot{V}_\Delta(t) - 2\Delta x_t^\top C_1^\top \Lambda \Delta w_t + 2\Delta x_t^\top C_2^\top S^\top \Delta u_t + \Delta w_t^\top W \Delta w_t \\ +2\Delta w_t^\top (D_{21}^\top S^\top - \Lambda D_{12})\Delta u_t + \Delta u_t^\top (R + SD_{22} + D_{22}^\top S^\top)\Delta u_t \\ +\Delta y_t^\top\, Q\, \Delta y_t > 0\end{gathered} \tag{2.30}$$

Rearranging (2.30):

$$\begin{gathered}-\dot{V}_\Delta(t) + \Delta y_t^\top Q \Delta y_t + \Delta u_t^\top R \Delta u_t + 2\Delta u_t^\top S(D_{22}\Delta u_t + C_2 \Delta x_t) \\ -2\Delta w_t^\top \Lambda (C_1 \Delta x_t + D_{11}\Delta w_t) + 2\Delta w_t^\top \Lambda \Delta w_t + 2\Delta w_t^\top (D_{21}^\top S^\top - \Lambda D_{12})\Delta u_t > 0\end{gathered} \tag{2.31}$$

From equations (2.3), it easy to verify that:

$$\begin{aligned}(C_1 \Delta x_t + D_{11}\Delta w_t) &= (\Delta v_t - D_{12}\Delta u_t) \\ (D_{22}\Delta u_t + C_2 \Delta x_t) &= (\Delta y_t - D_{21}\Delta w_t)\end{aligned} \tag{2.32}$$

Using (2.32) in (2.31):

$$\begin{gathered}-\dot{V}_\Delta(t) + \Delta y_t^\top Q \Delta y_t + \Delta u_t^\top R \Delta u_t + 2\Delta u_t^\top S(\Delta y_t - D_{21}\Delta w_t) \\ -2\Delta w_t^\top \Lambda (\Delta v_t - D_{12}\Delta u_t) + 2\Delta w_t^\top \Lambda \Delta w_t + 2\Delta w_t^\top (D_{21}^\top S^\top - \Lambda D_{12})\Delta u_t > 0\end{gathered} \tag{2.33}$$

Simplifying and gathering (2.33):

$$-\dot{V}_\Delta(t) + \big(\Delta y_t^\top Q \Delta y_t + \Delta u_t^\top R \Delta u_t + 2\Delta u_t^\top S \Delta y_t\big) - \big(2\Delta w_t^\top \Lambda \Delta v_t - 2\Delta w_t^\top \Lambda \Delta w_t\big) > 0 \tag{2.34}$$

Rearranging (2.34):

$$-\dot{V}_\Delta(t) + \begin{bmatrix}\Delta y_t^\top & \Delta u_t^\top\end{bmatrix} \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix}\Delta y_t \\ \Delta u_t\end{bmatrix} - \underbrace{\begin{bmatrix}\Delta v_t^\top & \Delta w_t^\top\end{bmatrix} \begin{bmatrix} 0 & \Lambda \\ \Lambda & -2\Lambda \end{bmatrix} \begin{bmatrix}\Delta v_t \\ \Delta w_t\end{bmatrix}}_{=\Gamma_t} > 0 \tag{2.35}$$

Remembering the definition of Γ_t from (2.6), (2.35) becomes:

$$-\dot{V}_\Delta(t) + \begin{bmatrix} \Delta y_t^\top & \Delta u_t^\top \end{bmatrix} \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} \Delta y_t \\ \Delta u_t \end{bmatrix} - \Gamma_t > 0 \tag{2.36}$$

In the end, changing the sign of (2.36):

$$\dot{V}_\Delta(t) - \begin{bmatrix} \Delta y_t^\top & \Delta u_t^\top \end{bmatrix} \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} \Delta y_t \\ \Delta u_t \end{bmatrix} < -\Gamma_t \leq 0 \tag{2.37}$$

Defining with $\tilde{s}(\cdot)$:

$$\tilde{s}(\Delta y_t, \Delta u_t) \; = \; \begin{bmatrix} \Delta y_t^\top & \Delta u_t^\top \end{bmatrix} \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} \Delta y_t \\ \Delta u_t \end{bmatrix} \tag{2.38}$$

Then we obtain that:

$$\dot{V}_\Delta(t) < s(\Delta u_t, \Delta y_t) \tag{2.39}$$

Assuming the function V_Δ is differentiable, then (2.39) can be rewritten as:

$$V_\Delta(t) < V_\Delta(t_0) + \int_{t_0}^{t} s(\Delta u_\tau, \Delta y_\tau) d\tau \tag{2.40}$$

That is exactly the inequality that a system must guarantee in order to satisfy the incremental IQC, defined by the supply rate $s(\cdot)$ with storage function V_Δ (see Definition 1.4.1). □

At this point, using the special choices of the matrices (Q, S, R) in Section 1.4, it is possible to verify properties of the architecture to be incrementally dissipative and passive.

2.2.1. Robust REN-ODEs are Contractive

It is important to notice that, observing the form of the inequalities (2.8),(2.23), it can be proven that, choosing a matrix Q negative definite, the property of incremental robustness implies contractivity.

Theorem 2.3 (Robust REN-ODE are Contractive). *A REN-ODE in the form* (2.1)-(2.2) *that is well-posed and satisfies the incremental IQC described by* (Q, S, R)*, with* Q *negative definite and* $R = R^\top$*, is also* ***contractive****.*

Proof. If the REN-ODE is well-posed and satisfies the incremental IQC described by

(Q, S, R), it means that there exists a matrix $P > 0$ and a diagonal matrix $\Lambda > 0$ such that (2.23) is satisfied. If the matrix Q is definite negative and $R = R^\top$, then (2.23) can be rewritten as:

$$\begin{bmatrix} -A^\top P - PA & -PB_1 - C_1^\top \Lambda & -PB_2 + C_2^\top S^\top \\ -B_1^\top P - \Lambda C_1 & W & D_{21}^\top S^\top - \Lambda D_{12} \\ -B_2^\top P + SC_2 & SD_{21} - D_{12}^\top \Lambda & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} > - \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top > 0 \tag{2.41}$$

Thus defining with $\mathcal{K}$ the matrix on the left-side of (2.41), i.e.:

$$\mathcal{K} = \begin{bmatrix} -A^\top P - PA & -PB_1 - C_1^\top \Lambda & -PB_2 + C_2^\top S^\top \\ -B_1^\top P - \Lambda C_1 & W & D_{21}^\top S^\top - \Lambda D_{12} \\ -B_2^\top P + SC_2 & SD_{21} - D_{12}^\top \Lambda & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} > 0 \tag{2.42}$$

Using the *Sylvester's criterion*, the positive definiteness of matrix $\mathcal{K}$ implies that its upper-left $(n+q)$-by-$(n+q)$ sub-matrix is positive definite, or, in another way:

$$\begin{bmatrix} -A^\top P - PA & -PB_1 - C_1^\top \Lambda \\ -B_1^\top P - \Lambda C_1 & W \end{bmatrix} > 0 \tag{2.43}$$

That is the LMI in (2.8) that guarantees the model to be contractive. Thus, robustness implies contractivity. □

2.2.2. Convex Parametrization

In order to obtain *direct parametrization*, the inequalities (2.8),(2.23) must be convex in the parameters $(A, B_1, B_2, \dots, D_{22}, P, \Lambda)$ all together. However, at the time being, they are not due to the presence of non-linearity. For this reason, the inequalities will be rewritten.

Convex Parametrization of CREN-ODE

It can be observed that (2.8) is not linear in its parameters (e.g., $A^\top P, C_1^\top \Lambda$). However, it is possible to cast it, under a suitable reformulation, as a linear one. To do so, we define the following matrices:

$$U = C_1^\top \Lambda\,, \quad U \in \mathbb{R}^{n\times q} \tag{2.44}$$

$$Y = PA\,, \quad Y \in \mathbb{R}^{n\times n} \tag{2.45}$$

$$Z = PB_1\,, \quad Z \in \mathbb{R}^{n\times q} \tag{2.46}$$

Then (2.8) becomes:

$$\begin{bmatrix} -Y^\top - Y & -U - Z \\ -U^\top - Z^\top & W \end{bmatrix} > 0 \tag{2.47}$$

Note that, now, the inequality (2.47) is linear and convex, having the independent variables (U, W, Y, Z) as free/design variables.

Convex Parametrization of RREN-ODE

It can be observed that (2.23) is not linear in its parameters. However, in order to obtain a convex version of the LMI, some intermediate steps must be introduced. We can define the following matrices:

$$\mathcal{V} = PB_2\,, \quad \mathcal{V} \in \mathbb{R}^{n\times m} \tag{2.48}$$

$$T = \Lambda D_{12}\,, \quad T \in \mathbb{R}^{q\times m} \tag{2.49}$$

Using these new definitions and (2.44)-(2.46), (2.23) becomes:

$$\begin{bmatrix} -Y^\top - Y & -U - Z & -\mathcal{V} + C_2^\top S^\top \\ -U^\top - Z^\top & W & D_{21}^\top S^\top - T \\ -\mathcal{V}^\top + SC_2 & SD_{21} - T^\top & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top > 0 \tag{2.50}$$

We can define, additionally, the matrices:

$$\tilde{\mathcal{V}} = -\mathcal{V} + C_2^\top S^\top + C_2^\top Q D_{22}\,, \quad \tilde{\mathcal{V}} \in \mathbb{R}^{n\times m} \tag{2.51}$$

$$\tilde{T} = -T + D_{21}^\top S^\top + D_{21}^\top Q D_{22}\,, \quad \tilde{T} \in \mathbb{R}^{q\times m} \tag{2.52}$$

$$\mathcal{R} = R + SD_{22} + D_{22}^\top S^\top + D_{22}^\top Q D_{22}\,, \quad \mathcal{R} \in \mathbb{R}^{m\times m} \tag{2.53}$$

It is easy to check that, using (2.51)-(2.53), (2.50) can be rearranged as:

$$\begin{bmatrix} -Y^\top - Y & -U - Z & \tilde{\mathcal{V}} \\ -U^\top - Z^\top & W & \tilde{T} \\ \tilde{\mathcal{V}}^\top & \tilde{T}^\top & \mathcal{R} \end{bmatrix} + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ 0 \end{bmatrix}^\top > 0 \tag{2.54}$$

At this point, using the *Schur Complement*, it is possible to rewrite the inequality (2.54) as:

$$\mathcal{R} > 0, \tag{2.55}$$

$$\begin{bmatrix} -Y^\top - Y & -U - Z \\ -U^\top - Z^\top & W \end{bmatrix} - \begin{bmatrix} \tilde{\mathcal{V}} \\ \tilde{T} \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \tilde{\mathcal{V}} \\ \tilde{T} \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix}^\top > 0 \tag{2.56}$$

Even though it might seem linear and ready for the direct parametrization, a problem can appear due to the definition of the matrix $\mathcal{R}$. Indeed, taking in consideration the cases in which S may be chosen equal to 0 (e.g., for dissipativity/passivity), then the matrix $\mathcal{R}$ would be:

$$\mathcal{R} = R + D_{22}^\top Q D_{22} \tag{2.57}$$

with Q definite negative. Fixed the value of $\mathcal{R}$, the equation (2.57), solved in D_{22} could lead to complex matrices, obviously not acceptable for real signals[2]. Thus, the matrix D_{22} cannot be a free learnable parameter and additional steps on how to choose it must be added in order to enforce (2.55).

We can define with:

$$s = \max(p, m)\ , \quad X_3, Y_3 \in \mathbb{R}^{s\times s} \tag{2.58}$$

$$M = X_3^\top X_3 + Y_3 - Y_3^\top + \epsilon I \tag{2.59}$$

$$Z = \Big[(I - M)(I + M)^{-1}\Big]_{p\times m} \tag{2.60}$$

with $\epsilon > 0$ and indicating with $[V]_{p\times m}$ the block matrix with the first p rows and m columns of the generic matrix V. Assuming $Q < 0$ and $R = R^\top$(in the special cases considered in Section 1.4, it is always verified), Q and R can be factorized as:

$$L_Q^\top L_Q = -Q\ , \quad L_R^\top L_R = R - SQ^{-1}S^\top \tag{2.61}$$

and finally:

$$D_{22} = -Q^{-1}S^\top + L_Q^{-1} Z L_R \tag{2.62}$$

Constructing D_{22} as (2.62), guarantees the inequality (2.55) always holds, for any choice of the IQC matrices, even for $S = 0$.

[2]The matrix $\mathcal{R}$ will be fixed during the direct parametrization, in which it will be put equal to the block matrix H_{33} (see Section 2.3).

2.2.3. Assumptions on the symmetry of P

In many steps, during the reformulation of the LMIs (2.8),(2.23) in (2.47),(2.54) respectively, it has been taken for granted that the matrix P is symmetric, i.e., $P = P^\top$. However, this assumption is not restrictive. Indeed, being P a square matrix, it can always be rewritten as:

$$P = \underbrace{\frac{P + P^\top}{2}}_{\text{symmetric}} + \underbrace{\frac{P - P^\top}{2}}_{\text{skew-symmetric}} = P_{symm} + P_{skew} \tag{2.63}$$

We can show that:

$$\begin{aligned}
x^\top P x &= x^\top (P_{symm} + P_{skew}) x && (2.64)\\
&= x^\top P_{symm} x + x^\top P_{skew} x && (2.65)\\
&= x^\top P_{symm} x + \frac{1}{2} x^\top P_{skew} x + \frac{1}{2} x^\top P_{skew} x && (2.66)\\
&= x^\top P_{symm} x + \frac{1}{2} x^\top P_{skew} x + \frac{1}{2} (x^\top P_{skew} x)^\top && (2.67)\\
&= x^\top P_{symm} x + \frac{1}{2} x^\top P_{skew} x - \frac{1}{2} x^\top P_{skew} x && (2.68)\\
&= x^\top P_{symm} x && (2.69)
\end{aligned}$$

Thus, determining the definiteness of a matrix there is no loss of generality in restricting the matrix to be symmetric. This property can be extended to the LMI in (2.8), due to the particular structure of it. Indeed, substituting (2.63) in the matrix:

$$\begin{bmatrix} -A^\top P - PA & -C_1^\top \Lambda - PB_1 \\ -\Lambda C_1 - B_1^\top P & W \end{bmatrix} \tag{2.70}$$

results in:

$$\begin{aligned}
&\begin{bmatrix} -A^\top P_{symm} - P_{symm} A & -C_1^\top \Lambda - P_{symm} B_1 \\ -\Lambda C_1 - B_1^\top P_{symm} & W \end{bmatrix} \\
&\qquad\qquad + \begin{bmatrix} -A^\top P_{skew} - P_{skew} A & -P_{skew} B_1 \\ -B_1^\top P_{skew} & 0 \end{bmatrix}
\end{aligned} \tag{2.71}$$

where it can be easily checked that the right matrix is still skew-symmetric, for any choice of A, B_1. Hence, as shown in (2.64)-(2.69), the skew-symmetric part will not affect the matrix inequality. Similarly, the same considerations can be done for (2.23).

2.3. Direct Parametrization

Now we are ready to retrieve a direct parametrization of CREN-ODEs and RREN-ODEs.

Contracting Direct Parametrization

Let the matrix from the left-part of the inequality (2.47) be called H, with $H \in \mathbb{R}^{(n+q)\times(n+q)}$. Then, for any matrix $X \in \mathbb{R}^{(n+q)\times(n+q)}$ and an $\epsilon > 0$, imposing that:

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} = X^\top X + \epsilon I \,, \qquad \forall X \tag{2.72}$$

satisfies necessarily the inequality (2.47) by construction. At this point, comparing (2.72) with (2.47):

$$-Y^\top - Y = H_{11} \tag{2.73}$$

$$-U - Z = H_{12} \tag{2.74}$$

From (2.73):

$$Y = -\frac{1}{2}(H_{11} + Y_1 - Y_1^\top) \tag{2.75}$$

where Y_1 is a free matrix that allows to obtain also non-symmetric Y matrices. Considering (2.74), it is possible to keep one of the matrices (U, Z) free (e.g., U), and then obtain the remaining one by the relation (2.74). In case U is chosen freely:

$$Z = -H_{12} - U \tag{2.76}$$

If the system is *acyclic*, then D_{11} must be strictly lower triangular, thus from H_{22}:

$$H_{22} = W = 2\Lambda - D_{11} - D_{11}^\top \tag{2.77}$$

it is possible to obtain Λ from $1/2$ of the main diagonal of W and D_{11} from the strict lower triangular part of W. It must be noted that the matrix P never appears by itself, thus it does not need to be calculated from H. However, P must be positive definite. In order to enforce this condition, P can be built as follows:

$$P = P_1 P_1^\top + \epsilon I > 0 \,, \qquad \forall P_1 \in \mathbb{R}^{n\times n} \tag{2.78}$$

where P_1 is a generic free matrix and $\epsilon > 0$.
From (2.73)-(2.78) the following matrices can be retrieved:

$$Y = PA \quad \rightarrow \quad A = P^{-1}Y \tag{2.79}$$

$$Z = PB_1 \quad \rightarrow \quad B_1 = P^{-1}Z \tag{2.80}$$

$$U = C_1^\top \Lambda \quad \rightarrow \quad C_1 = (U\Lambda^{-1})^\top \tag{2.81}$$

The remaining parameters do not impact the contractivity of the model and, thus, they can be considered free. Summing up, the free parameters (i.e., $X \in \mathbb{R}^{(n+q)\times(n+q)}, B_2 \in \mathbb{R}^{n\times m}, C_2 \in \mathbb{R}^{p\times n}, D_{12} \in \mathbb{R}^{q\times m}, D_{21} \in \mathbb{R}^{p\times q}, D_{22} \in \mathbb{R}^{p\times m}, b \in \mathbb{R}^{(n+q+p)}, P_1 \in \mathbb{R}^{n\times n}, U \in \mathbb{R}^{n\times q}$ and $Y_1 \in \mathbb{R}^{n\times n}$) can be updated without any restriction, and, afterwards, the constrained remaining matrices are obtained through (2.72)-(2.81). Please note that a possible application in case of full D_{11} can be carried out. In order to do so, the full matrix must be constructed in the following way:

$$D_{11} = \Lambda - \frac{1}{2}(H_{22} + Y_2 - Y_2^\top) \tag{2.82}$$

where $g \in \mathbb{R}^q$, $Y_2 \in \mathbb{R}^{q\times q}$ are free parameters and finally $\Lambda = e^{diag(g)}$.
The possibility of having an unconstrained parametrization of the network model, allows to use the most common iterative optimization methods for the update of the parameters in such a way to optimize a desired cost function. An example of iterative optimization methods can be *gradient descent* and its variations like SGD, Adam[40], AdaGrad[41], AMSGrad[42], etc. For a deeper and more accurate review about the topic, the reader is referred to [43]. Additionaly, in Chapter 3 more information about the implementations of them in Python will be provided.

Robust Direct Parametrization

Once the required steps to retrieve a convex parametrization of the matrix inequality(2.23) have been done (i.e.,(2.55)-(2.56)), the direct parametrization in the robust case proceeds similarly to the contractive one.
As first step, given the matrices (Q, S, R), the construction of the matrix D_{22} is carried out, using (2.58)-(2.62); consequently, the inequality (2.55) is guaranteed. After that, denoting with $\tilde{d} = (n + q)$, let $H^* \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ be the resulting matrix from the inequality

(2.56):

$$H^* = \begin{bmatrix} -Y^\top - Y & -U - Z \\ -U^\top - Z^\top & W \end{bmatrix} - \begin{bmatrix} \tilde{\mathcal{V}} \\ \tilde{T} \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \tilde{\mathcal{V}} \\ \tilde{T} \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix}^\top \tag{2.83}$$

Additionally, let H be defined as the matrix:

$$H = \begin{bmatrix} -Y^\top - Y & -U - Z \\ -U^\top - Z^\top & W \end{bmatrix} \tag{2.84}$$

Furthermore:

$$H^* = H - \begin{bmatrix} \tilde{\mathcal{V}} \\ \tilde{T} \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \tilde{\mathcal{V}} \\ \tilde{T} \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix}^\top \tag{2.85}$$

where matrices $(\tilde{\mathcal{V}}, \tilde{T})$ are obtained using (2.51) and (2.52) respectively, with $(\mathcal{V}, T)$ as free parameters. Then, for any matrix $X \in \mathbb{R}^{\tilde{d}\times\tilde{d}}$ and an $\epsilon > 0$:

$$H^* = X^\top X + \epsilon I > 0\,. \tag{2.86}$$

Finally, using (2.84)-(2.86):

$$H = \underbrace{XX^\top + \epsilon I}_{H^*} + \begin{bmatrix} \tilde{\mathcal{V}} \\ \tilde{T} \end{bmatrix} \mathcal{R}^{-1} \begin{bmatrix} \tilde{\mathcal{V}} \\ \tilde{T} \end{bmatrix}^\top - \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix}^\top > 0\,, \qquad \forall X \tag{2.87}$$

In addition, D_{12} is obtained:

$$T = \Lambda D_{12} \quad \rightarrow \quad D_{12} = \Lambda^{-1} T \tag{2.88}$$

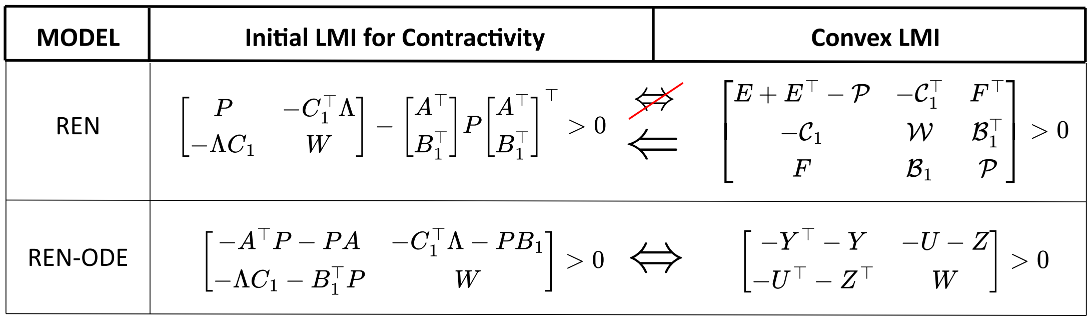
Now, it is possible to follow similar steps as done for contractive REN-ODEs (using (2.72)-(2.81)). Summing up, the free parameters (i.e., $X \in \mathbb{R}^{(n+q)\times(n+q)}, B_2 \in \mathbb{R}^{n\times m}, C_2 \in \mathbb{R}^{p\times n}, T \in \mathbb{R}^{q\times m}, D_{21} \in \mathbb{R}^{p\times q}, X_3, Y_3 \in \mathbb{R}^{s\times s}, b \in \mathbb{R}^{(n+q+p)}, P_1 \in \mathbb{R}^{n\times n}, U \in \mathbb{R}^{n\times q}$ and $Y_1 \in \mathbb{R}^{n\times n}$) can be updated without any restriction and, afterwards, the constrained remaining matrices are obtained through (2.72)-(2.81) and (2.88). Thus, we have been able to obtain a direct parametrization of a REN-ODE that satisfies the incremental IQC given by (Q, S, R). On how to update the parameters, in a way to minimize a desired cost function, the reader is invited, as said before, to [43]. As previously said, the assumption of acyclicity of the REN-ODE is not necessary and how to construct the matrix D_{11} is the same as it has been provided during the contractive direct parametrization.

2.4. Comments on REN-ODE

REN-ODEs introduce some differences with respect to its discrete-time counterpart. Indeed, even though it has not been showed in its entirety, during the passage from the initial LMIs (1.36), (1.38) to their convex versions (1.40), (1.42) respectively, the following inequality is used:

$$E^\top \mathcal{P}^{-1} E \geq E + E^\top - P \tag{2.89}$$

This guarantees that (1.40), (1.42) imply (1.36), (1.38), but not vice versa. On the other hand, due to the use of only linear reformulations and of the *Schur Complement*, between the REN-ODE's LMIs there is a *one-to-one correspondence* (i.e., the relations are *bijections*). A scheme of this statement is reported in Figure 2.1 just for the contractive case. This passage is relevant for what concerns the property of robust REN-ODE to be



MODEL	Initial LMI for Contractivity	Convex LMI
REN	$\begin{bmatrix} P & -C_1^\top \Lambda \\ -\Lambda C_1 & W \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix}^\top > 0$ $\cancel{\Leftrightarrow}$ $\Leftarrow$	$\begin{bmatrix} E + E^\top - \mathcal{P} & -\mathcal{C}_1^\top & F^\top \\ -\mathcal{C}_1 & \mathcal{W} & \mathcal{B}_1^\top \\ F & \mathcal{B}_1 & \mathcal{P} \end{bmatrix} > 0$
REN-ODE	$\begin{bmatrix} -A^\top P - PA & -C_1^\top \Lambda - PB_1 \\ -\Lambda C_1 - B_1^\top P & W \end{bmatrix} > 0$ $\Leftrightarrow$	$\begin{bmatrix} -Y^\top - Y & -U - Z \\ -U^\top - Z^\top & W \end{bmatrix} > 0$

Figure 2.1: Comparison between the passages from contractive LMIs and their convex version. The discrete-time RENs do not preserve one-to-one correspondence.

also contractive. It needs to be commented that the same steps done in Theorem 2.3 can be used, with some changes, also for the discrete-time REN matrix inequalities (1.36), (1.38). The proof is reported in Appendix A. However, during the passage to direct parametrization of the models, the discrete-time RENs lose the possibility to show that a robust direct parametrization implies a contractive one (the form of the inequalities are completely different). On the other hand, thanks to the fact that the REN-ODE's convex matrix inequalities use only linear reformulations and Schur complement, it is easy to verify that, even after direct parametrization, Theorem 2.3 is still valid.

Proposition 2.1. *If a REN-ODE is obtained through a robust direct parametrization, then the model is also contractive.*

Proof. Using the robust direct parametrization for a REN-ODE, it has been proven that (2.55), (2.56) are always satisfied. Additionally, (2.55), (2.56) can be rewritten equivalently as (2.23), report here for clarity:

$$\begin{bmatrix} -A^\top P - PA & -PB_1 - C_1^\top \Lambda & -PB_2 + C_2^\top S^\top \\ -B_1^\top P - \Lambda C_1 & W & D_{21}^\top S^\top - \Lambda D_{12} \\ -B_2^\top P + SC_2 & SD_{21} - D_{12}^\top \Lambda & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top > 0 \quad (2.90)$$

with:

$$W = 2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda. \quad (2.91)$$

Thanks to Theorem 2.2, the system is guaranteed to be robust. At this point, using Theorem 2.3, the robust REN-ODE is contractive as well. □

Summing up, a RREN-ODE is also a CREN-ODE, while the same cannot be guaranteed for discrete-time RENs (Table 2.1).

Property	**CREN**	**RREN**	**CREN-ODE**	**RREN-ODE**
Contractivity	✓		✓	✓
Robustness		✓		✓

Table 2.1: Properties of the different models.

This is an important result: given a robust (and contractive) RREN-ODE and an input function, if there exists a stable equilibrium point of the system for that input, then, for any initial condition, the model will always converge exponentially to the equilibrium state while keeping guarantees of *incremental* dissipativity. Additionally, this incremental form of robustness also becomes valid for the *general* form of the system, as shown in [44, Theorem 10]. This opens up to new possibilities: for example, if a passive REN-ODE is used to control a passive system (through an optimal policy) then, it can be proven that the overall negative feedback interconnection is still passive. This statement comes from [45][*Theorem* 4.23], reported here for completeness:

Theorem 2.4. *The feedback connection of two passive systems Σ_1, Σ_2 is passive.*

A scheme of a feedback connection of two passive systems is represented in Figure 2.2a. The proof of the theorem can be found in [45]. At this point, a negative feedback interconnection (Figure 2.2b) can be seen as a special case of feedback connection in which $u_2 = 0$ and $e_2 \equiv y_1$. In this way, all the techniques based on the properties of control using passivity, i.e., *Passivity Based Control* (PBC), can be applied. Indeed, this property is relevant in the study of networked system: it is possible to apply a modular method for the construction of large-scale passive networked systems while also providing some degrees of robustness to unmodeled passive dynamics. About the topic, the reader is invited to [46]. An example of application is provided in Section 3.5. It is also important to remind to the reader that, REN-ODE, being a subclass of Neural-ODEs, inherits from them all the benefits reported in Section 1.7. In fact, it is possible to implement state-of-the-art ODE solvers that make use of variable-step strategies in order to guarantee requested levels of accuracy and much more. The possibility to solve the ODE with many different methods allows the user to choose the solver based on the problem complexity and the available hardware. Additionally, different methods can be applied even between the training phase and the testing one: thus, a slow and heavy solver can be used during the training phase and then in real-time application the returned model could be implemented with a lighter integration method.

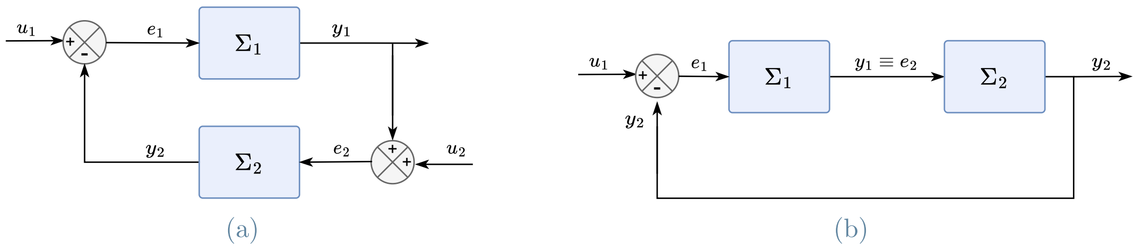


Figure 2.2: Scheme of a feedback connection of two systems (left) and of a negative feedback interconnection (right).

3 | Simulations & Results

In this chapter, the proposed REN-ODE will be used in different fields of application: binary classification, system identification and optimal control. Initially, details about the technical implementations are going to be provided and then simulations to show the properties of contractivity and robustness will be reported.

3.1. Implementation

The implementation was carried out using Python as main programming language. The main third-party libraries used for REN-ODEs are:

- PyTorch[47] for the neural network framework. It allows the possibility to use GPU acceleration.
- Torchdiffeq[48], developed by the authors of the Neural-ODE architecture. It allows the use of the adjoint-method for the back-propagation during the training phase (for details, check Section 1.7.2). It supports different integration methods.
- Scikit-learn[49] and NumPy[50] have been used respectively for the evaluation of classical observational errors (e.g., accuracy score, precision) and the simulation of the mechanical systems to be identified.

Additionally, the Torchdiffeq library allows the use of different integration methods, reported here with their *'code-name'* for sake of clarity:

- Methods with Variable-step of integration:
  - *'dopri8'*, Runge-Kutta of order 8 of Dormand-Prince-Shampine.
  - *'dopri5'*, Runge-Kutta of order 5 of Dormand-Prince-Shampine.
  - *'bosh3'*, Runge-Kutta of order 3 of Bogacki-Shampine.
  - *'fehlberg2'*, Runge-Kutta-Fehlberg of order 2.
  - *'adaptive_heun'*, Runge-Kutta of order 2.

- Methods with Fixed-step of integration:
  - *'euler'*, (Forward) Euler method.
  - *'midpoint'*, Midpoint method.
  - *'rk4'*, Fourth-order Runge-Kutta with 3/8 rule.
  - *'explicit_ adams'*, Explicit Adams-Bashforth.
  - *'implicit_ adams'*, Implicit Adams-Bashforth-Moulton.

The main used methods during this work have been: *'dopri5'*, *'euler'*, *'rk4'*. One of the most well-known limitations of the Neural-ODE package[20] is the possibility of setting only the initial state and the time vector for the simulation through the ODE solver: the introduction of any kind of predefined input sequence or arguments from outside(such as model parameters) is not allowed out-of-the-box. This can be really inconvenient in applications such as system identification, in which the training phase can be done using pre-recorded experiments(i.e., sequences of input-output that have been collected beforehand). For the REN-ODE, thus, in order to solve this problem, a simple solution has been considered. It is possible to introduce a composition of classes as schematized in Figure 3.1. The new component "signal generator", as the name suggests, has the role to evaluate at each time t the input $u(t)$ to pass to the REN-ODE system. A pseudo-code of the forward function of the REN-ODE is reported in Algorithm 3.1: this function will be called at each iteration by the ODE solver.

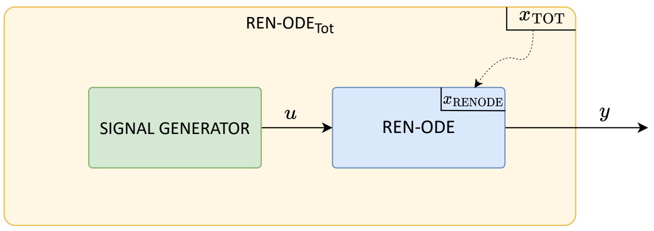


Figure 3.1: Scheme of the implementation of a REN-ODE with an input signal $u(t)$ using Torchdiffeq library.

Algorithm 3.1 Pseudo-code of the Feed-Forward function of REN-ODE$_{Tot}$

```
Input: model, time instant $t$, state at time $x(t)$.
  $u(t)$ = model.signal_generator($t$)          ▷ Get the input at time $t$
  $\dot{x}$ = model.RENODE($t, x(t), u(t)$)      ▷ Evolution of the dynamical part
return $\dot{x}$                                 ▷ Return the derivative
```

Similarly, this limitation can appear also in the realization of any kind of negative feedback interconnection (see Figure 3.2). In order to implement it, a composition with the REN-ODE and the system "SYS" to control is carried out. It must be noted that the system "SYS", in order to be simulated, has been assumed to be *strictly proper*, i.e., the output y at time t does not depend by the input at the same instant. Additionally, if the reference signal r is variable with respect to time t, then an additional "signal generator" (as seen before), must be added inside the final composed closed-loop system. A pseudo-code of the total system is reported in Algorithm 3.2 and an application of it can be found in Section 3.5.

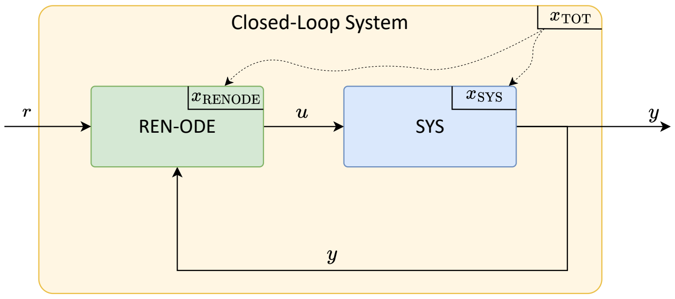


Figure 3.2: Scheme of the implementation of a feedback closed-loop system using Torchdiffeq library. The reference signal r must be defined a priori or generated by a "signal generator" inside the closed-loop system.

Algorithm 3.2 Pseudo-code of the Feed-Forward function in a Closed-Loop system.

```
Input: model (total system), time instant $t$, state of the total system $x_{TOT}(t)$.
    $x_{\text{RENODE}},\ x_{\text{SYS}} = \text{split}(x_{\text{TOT}}(t))$    ▷ Split state vector in two.
    $y(t) = \text{SYS.output}(t, x_{\text{SYS}})$    ▷ output evaluation of SYS
    $\dot{x}_{\text{REN-ODE}}(t) = \text{RENODE.forward}(t, x_{\text{SYS}})$    ▷ state evolution of RENODE
    $u_{\text{RENODE}}(t) = \text{RENODE.output}(t, x_{\text{SYS}})$    ▷ output evaluation of RENODE
    $\dot{x}_{\text{SYS}}(t) = \text{SYS.forward}(t, x_{\text{SYS}}, u_{\text{RENODE}}(t))$    ▷ state evolution of SYS
return $[\dot{x}_{RENODE}\ ,\ \dot{x}_{SYS}]$    ▷ Return the derivative of both states
```

3.2. Validation of Contractivity, ℓ_2-Bound and Passivity

In Chapter 2, it has been proven that properties of contractivity and robustness can always be guaranteed, using the direct parametrization described in Section 2.3. Thus, in this section, simulations are carried out to show practically that these properties are respected by the model's implementations. Starting with contractivity, in Figure 3.3 is shown that (1.2) is satisfied $\forall t$, simulating the model from two different initial conditions $a, b \in \mathbb{R}^n$. The two initial conditions were sampled randomly from a normal distribution with zero mean and unitary variance. The experiment's model is a CREN-ODE with $n = 4, q = 3, m = 1$. All the free parameter of the CREN-ODE were drawn from a normal distribution with zero mean and variance $\sigma^2 = 0.01$. The input $u(t)$ is chosen as:

$$u(t) = \mathbb{1}(t) \tag{3.1}$$

where $\mathbb{1}(t)$ denotes the unit step-signal. It is clearly visible how all the modules of the difference between the two trajectories' states (i.e., $|x^b - x^a|$) are always lower (or equal) than an exponential curve $\kappa e^{-ct}|b - a|$, with $c, \kappa > 0$. In this example, c was chosen equal to 0.5 and $\kappa = 1$.

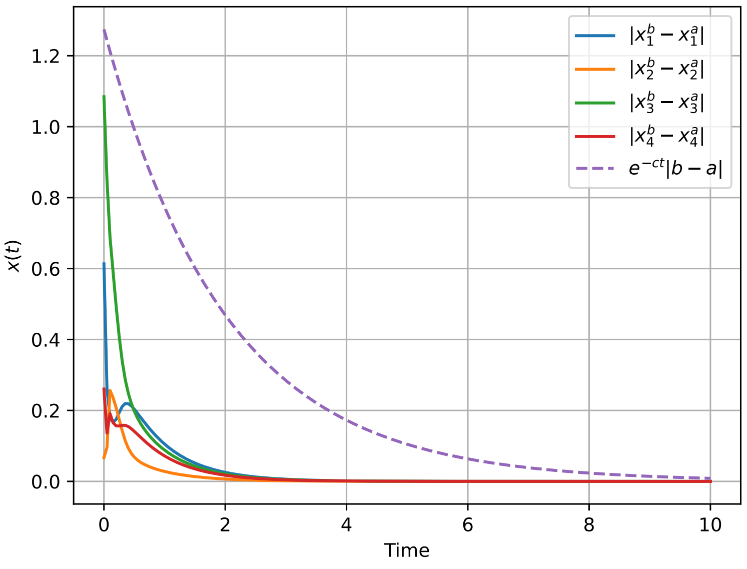


Figure 3.3: Plot of the module of the difference between two evolution of a model ($n = 4$) starting from two different initial conditions a, b. In violet, the curve $e^{-ct}|b - a|$.

In order to show that it is possible to guarantee that the model is *incremental* ℓ_2 Lipschitz bounded, a RREN-ODE is picked, with free parameters drawn randomly from a normal distribution with zero mean and variance $\sigma^2 = 0.01$. Then, the model has been simulated twice, using two different initial conditions $x_u(t_0), x_v(t_0)$ and two different inputs $u(t), v(t)$:

$$u(t) = -2e^{-0.2t} \sin\left(\frac{\pi}{2}t + \frac{\pi}{3}\right) \mathbb{1}(t) \tag{3.2}$$

$$v(t) = 3e^{-0.3t} \cos\left(\pi t\right) \mathbb{1}(t) \tag{3.3}$$

The initial conditions were randoly sampled from a normal distribution with zero mean and unitary variance. They were chosen differently, because, thanks to Theorem 2.3, the RREN-ODE is also contractive, thus the two evolutions will converge to each other after some time, no matter the two initial conditions. The plots of $u(t)$ and $v(t)$ are reported in Figure 3.4. Thus, given a value of γ bound, the parameters of the model are obtained as reported in Section 2.2. The experiments were carried out using different values of γ. Chosen one RREN-ODE, at each γ, only the constrained parameters were calculated, while the free ones were left the same. Fo each γ, the finite l2-norm of the difference between the two returned output trajectories, i.e., $\|y_u - y_v\|_t$ is computed and normalized with respect to the finite l2-norm of the difference between the inputs, i.e., $\|u - v\|_t$. The result are then compared with the bound γ and shown in Figure 3.5. It can be noticed that, for different values of γ, the Lipschitz bound is always kept, or, in other words:

$$\frac{\|y_u - y_v\|_t}{\|u - v\|_t} < \gamma \,, \qquad \forall t \tag{3.4}$$

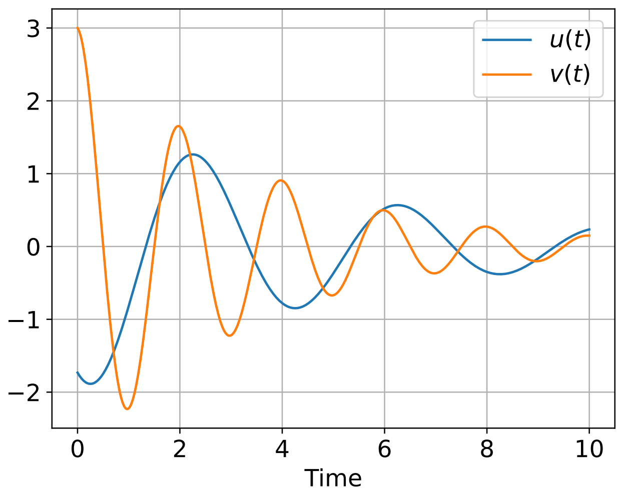
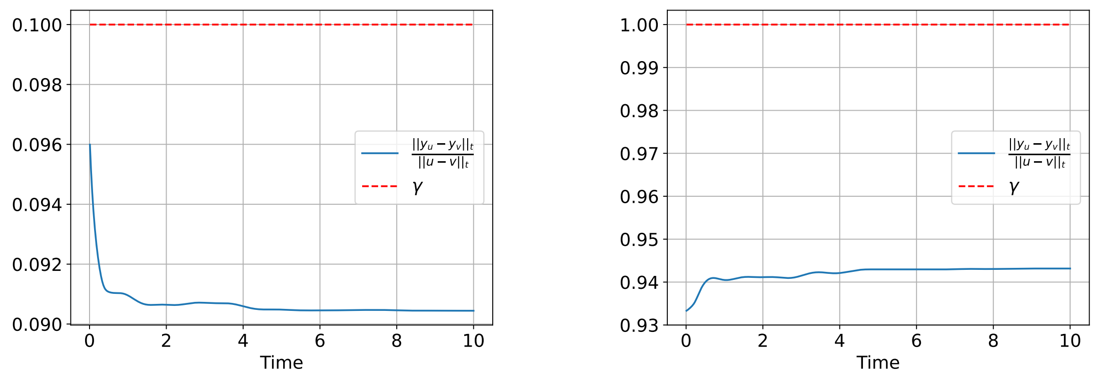


Figure 3.4: Plots of the inputs u, v (right).



(a) $\gamma = 0.1$ (b) $\gamma = 1$

Figure 3.5: Plots of the normalized finite l2-norm of the difference between the outputs $y_u(t), y_v(t)$ with $\gamma = 0.1$ (left) and $\gamma = 1$ (right).

Finally, experiments to show passivity of the RREN-ODE are carried out. The model must have the same number of inputs and outputs, i.e., $m \equiv n$. The free parameter of the model have been drawn from a normal distribution with zero mean and variance $\sigma^2 = 0.01$.

The system is perturbed with two different inputs $u(t), v(t)$, chosen for simplicity as in (3.2), (3.3), starting from two (random and different) initial conditions $x_u(t_0), x_v(t_0)$. The motivation is the same as before (a RREN-ODE is always contractive). In order to show that the system is passive, the differentiated form of the incremental dissipation inequality (2.40) must hold, reported here for clarity:

$$\dot{V}_\Delta(t) < s(\Delta u_t, \Delta y_t) \tag{3.5}$$

where $\dot{V}_\Delta(t)$ is the derivative of the storage function and $s(\Delta y_t, \Delta u_t)$ is the supply rate, respectively given by:

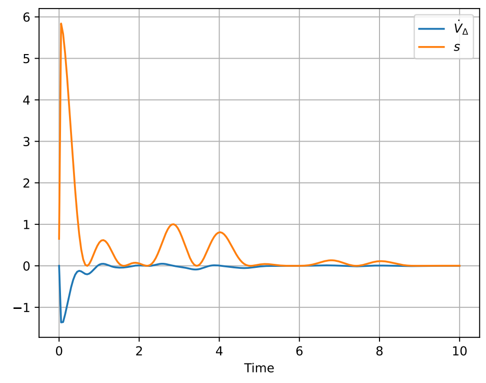
$$\dot{V}_\Delta(t) = \Delta \dot{x}_t^\top P \Delta x_t + \Delta x_t^\top P \Delta \dot{x}_t \tag{3.6}$$

$$s(\Delta y_t, \Delta u_t) = \begin{bmatrix} \Delta y_t^\top & \Delta u_t^\top \end{bmatrix} \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} \Delta y_t \\ \Delta u_t \end{bmatrix} \tag{3.7}$$

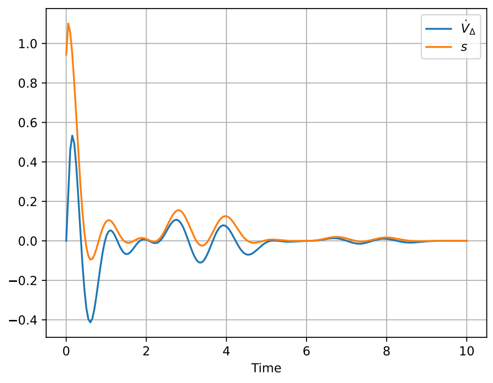
This result has been previously obtained in Section 1.4. The passage from the differentiated to the standard dissipativity inequality, and vice versa, can always be guaranteed if the function $V_\Delta(\cdot)$ is differentiable (see [22]). Two types of passivity have been enforced, using the matrices (Q, S, R) as provided in Section 1.4:

1. Strictly output passivity $\Rightarrow s(\Delta u_t, \Delta y_t) = \Delta u_t^\top \Delta y_t - \varepsilon \|\Delta y_t\|$
2. Strictly input passivity $\Rightarrow s(\Delta u_t, \Delta y_t) = \Delta u_t^\top \Delta y_t - \nu \|\Delta u_t\|$

Thus, during simulations, different values of ν and ε have been used. However, throughout the experiments, the free parameters of the model have been left constant, while only the constrained parameters have been modified, according to the robust direct parametrization in Section 2.2. The plots of $\dot{V}_\Delta(t)$ and $s(t)$ for each choice of ν and ε are reported in Figures 3.6 and 3.7. It is clear that the supply rate (orange) is always greater(or equal) than the derivative of the storage function(blue).

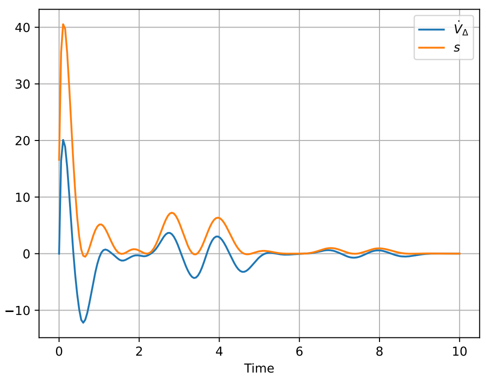


(a) $\varepsilon = 1$

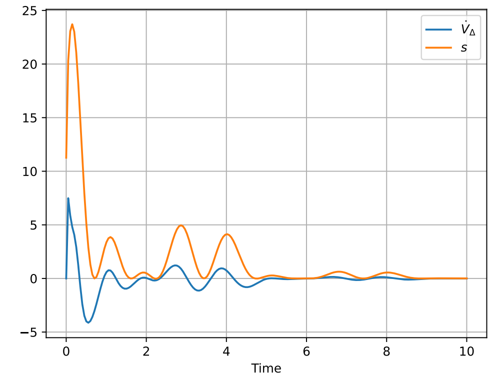


(b) $\varepsilon = 10$

Figure 3.6: Plots of the storage function $\dot{V}_\Delta$ and supply rate s of strictly output passive REN-ODE with $\varepsilon = 1$(left) and $\varepsilon = 10$(right).



(a) $\nu = 1$



(b) $\nu = 10$

Figure 3.7: Plots of the storage function $\dot{V}_\Delta$ and supply rate s of strictly input passive REN-ODE with $\nu = 1$(left) and $\nu = 10$(right).

3.3. Use Case 1: Binary Classification

The first study-case of the REN-ODE is the binary classification of different benchmarks from the state-of-the-art literature[51]. The data sets have always two classification labels $\{0, 1\}$ and two features (x_1, x_2). The classification is obtained in this way: the data-set has N points and a time window T_{end} is chosen. For any i^{th} point of the data-set (x_1^i, x_2^i), with $i \in \{1, N\}$, the sample is used as initial condition for the evolution of the REN-ODE, done up to time T_{end}; then the output $y(T_{end})$ is retrieved and passed through a Sigmoid layer returning a (normalized) value between 0 and 1. Finally, comparing this result with a 0.5 threshold, the binary classification p is obtained (see Figure 3.8). The goal of the

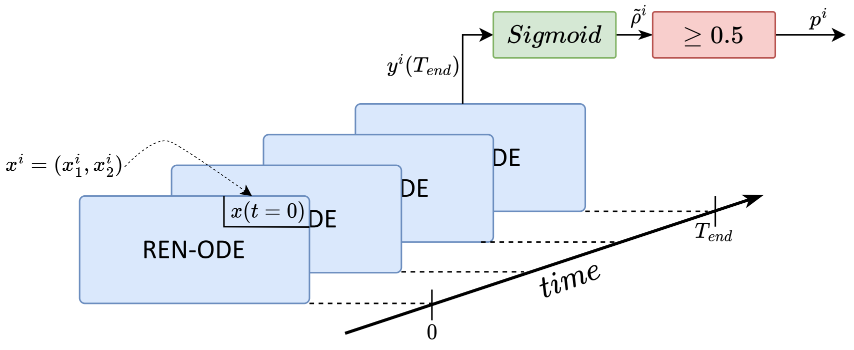


Figure 3.8: Scheme of the implementation of REN-ODE for a binary classification of the point x^i. The returned binary classification is p^i.

problem is to obtain a REN-ODE that, given any point (x_1^i, x_2^i) from the data-set with expected label $\hat{y}^i$, the returned binary classification p^i is always equal too the expected, i.e., $\hat{y}^i$. As loss function it has been used the Binary Classification Entropy(BCE) loss, typical of binary classification problems. The BCE is defined as follow:

$$\ell(\tilde{\rho}, \hat{y}) \;=\; -\big(\hat{y}\log(\tilde{\rho}) + (1-\hat{y})\log(1-\tilde{\rho})\big) \tag{3.8}$$

where $\tilde{\rho} = Sigmoid(y(T_{end}))$ and $\hat{y}$ is the expected one. However, during the training phase in iterative optimization techniques, the loss function is evaluated as the mean value of the loss function of η classification estimations, with $\eta \leq N$. Thus, the final loss function $L(\cdot)$ is defined as follow:

$$L_\eta = \frac{1}{\eta}\sum_{i=0}^{\eta} \ell(y_i, \hat{y}_i) \tag{3.9}$$

η is defined as a *hyper-parameter*, i.e., a parameter that is not learnt/trained through the training phase. The choice of the number η of points used to evaluate the loss function L

is an important topic still discussed in literature. Indeed, from η depends how many information is used at each time the parameters are updated to reach iteratively the optimum. In literature, η is called *batch-size*. Using a too small amount of data can produce a curve of values of $L(\cdot)$ with time to be highly non-smooth and monotone-decreasing, causing much more time to reach a stable system; from the other hand, choosing $\eta \approx N$ brings usually to a degradation of the ability of the model to generalize well, i.e., to perform well with data different from the ones used during the training. The following three types of optimizing techniques differ from the opted value of η :

- Gradient Descent(GD): $\eta = N$. The whole data-set is used to update the model.
- Stochastic Gradient Descent(SGD): $\eta = 1$. Only one sample is used to update the model (i.e., update its parameters).
- Mini-Batch Gradient Descent: $1 < \eta < N$. A group of data are used to update the model.

Furthermore, as explained in Section 1.7.1, after the evaluation of the loss function, through the back-propagation step, the evaluation of the gradient ∇L_θ of L with respect to the model's parameters θ is carried out. Then, the technique to update the parameters θ using ∇L_θ depends on the implemented optimizer. A simple version of the update rule of the parameters θ at iteration k can be represented as follows:

$$\theta_{k+1} = \theta_k - \ell_r \nabla L_{\theta_k} \tag{3.10}$$

where ℓ_r is the learning rate, an important hyper-parameter. Intuitively, it determines how much θ must be updated with respect to the error of the last batch of data considered for the loss evaluation. In the standard versions of gradient descent, the learning rate is fixed a priori. However, during the last decades, many variations of the update algorithm have been proposed: an example is *Adaptive Moment Estimation*(Adam)[40]. Adam is an adaptive learning rate optimization algorithm that utilises both momentum and scaling, combining the benefits of RMSProp and SGD with momentum[43]. The optimizer is designed to be appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. In the paper, the authors show empirically that Adam works well in practice and compares favorably to other adaptive learning-method algorithms. For this reason, Adam has been chosen for all classification applications. Another important question in ML is how many updates of the model must be done: a too short number may not bring the system to a minimum point, while a too big number may not be necessary due too over-training and leading also to over-fitting of the data used to train the model. In the years, many different ideas have been considered [52]. In this thesis, the implemented stopping-criteria for the training phase are:

- the loss function is smaller than a threshold value for a certain number of optimization steps (i.e., the loss has reached a desired value)
- the gradient of the loss function is smaller than a threshold value for a certain number of optimization steps (i.e., the loss is stuck in a local minimum)

Finally, it must be reported that, in order to verify that the trained model performs well even with cases not observed earlier, a common practice is to divide the data-set of experiments in two parts: training set (used by the model to optimize the parameter) and a testing set (used to test the performance of the model). During the experiments reported in this thesis the training set and the testing one have the same amount of samples, and, in the classification case, with the same amount of labelled data (i.e., each set is composed by half samples labelled '0' and half '1'). Before jumping into the results, the definition of accuracy in binary classification must be reported:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.11}$$

where:

- TP is the number of *true positive*, i.e. a point is classified correctly as positive.
- TN is the number of *true negative*, i.e. a point is classified correctly as negative.
- FP is the number of *false positive*, i.e., a point is classified as positive, but it is negative.
- FN is the number of *false negative*, i.e., a point is classified as negative, but it is positive.

Results

For the simulations, some of the benchmarks from the state-of-the-art have been used[51]. The following data-sets have been considered: "double moons"(Figure 3.10), "double circles"(Figure 3.11), "swiss roll"(Figure 3.12) and "checker board"(Figure 3.13). In red are represented the points with label '0', and in blue the ones with label '1'. The model used for this experiment is the CREN-ODE. However, as discussed in Section 1.7.2, using a CREN-ODE with the same amount of states as the input features (i.e., $n = 2$) would not allow the system to map correctly the space of the features (aka the data points). For this reason, the model has been considered with an augmented amount of initial states $\tilde{n} \geq 2$, thus, each 2D point has been augmented as:

$$x_{Point} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{2\times 1} \quad \rightarrow \quad \tilde{x}_{aug} = \begin{bmatrix} x_1 \\ x_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{\tilde{n}\times 1} \tag{3.12}$$

Indeed, in Figure 3.9 is reported the different values of accuracy obtained in function of the number of augmented states for the "double moons" benchmark.

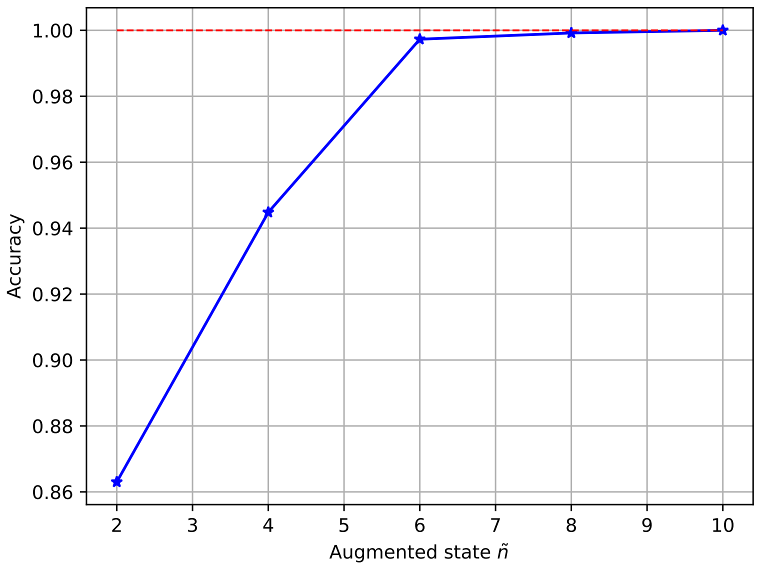
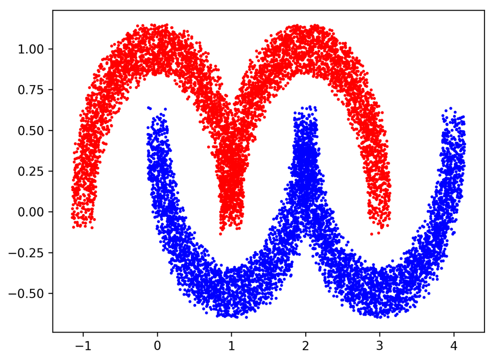


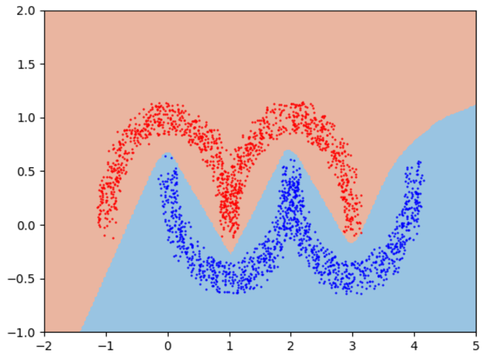
Figure 3.9: Plot of the accuracy obtained in function of the number $\tilde{n}$ of augmented states in the prediction of the "double moons" benchmark.

Below is reported a table with the minimum number $\tilde{n}$ of augmented states in order to obtain 100% for each data-set:

DataSet	$\tilde{n}$
Double Moons	10
Double Circles	16
Checker Board	40
Swiss Roll	45

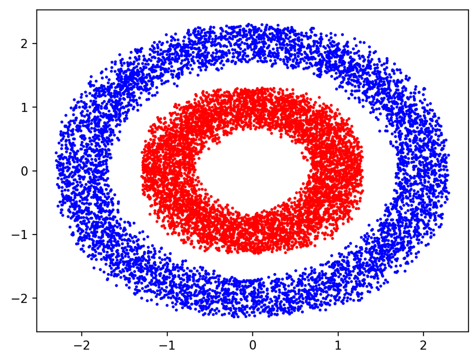


(a)

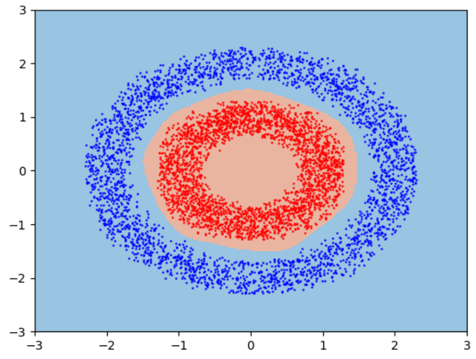


(b)

Figure 3.10: Labeled input features representing the "Double Moons" data-set (left), predictions of the REN-ODE (colors in the background) superimposed by the validation data.

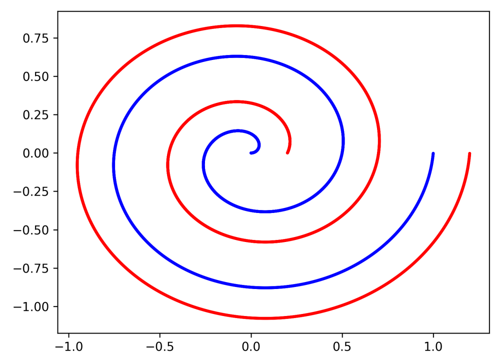


(a)



(b)

Figure 3.11: Labeled input features representing the "Double Circles" data-set (left), predictions of the REN-ODE (colors in the background) superimposed by the validation data.


(a)

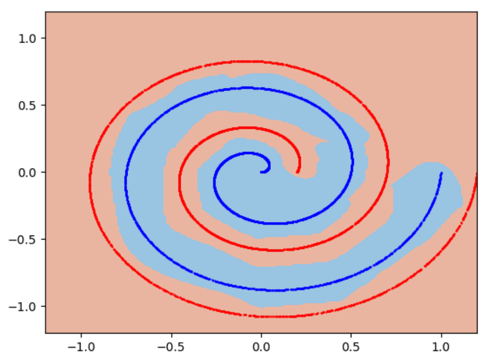
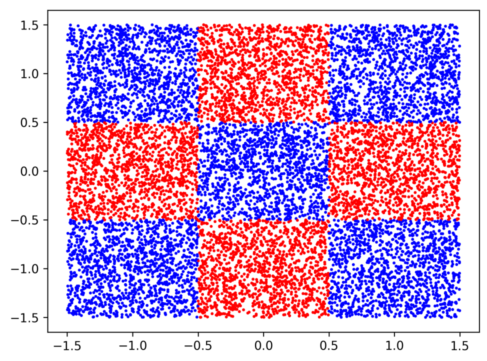

(b)

Figure 3.12: Labeled input features representing the "Swiss Roll" data-set (left), predictions of the REN-ODE (colors in the background) superimposed by the validation data.


(a)

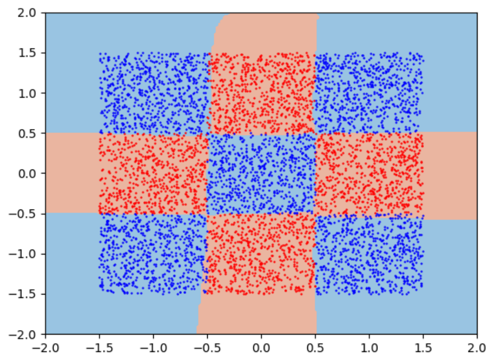

(b)

Figure 3.13: Labeled input features representing the "Checker Board" data-set (left), predictions of the REN-ODE (colors in the background) superimposed by the validation data.

3.4. Use Case 2: System Identification

REN-ODE has been tested for the system identification of a nonlinear pendulum in free evolution, namely building an approximated mathematical model of the system using the collected output signals. The main goal of the experiment is to find a REN-ODE that is able to find a good approximation of the real physical system, i.e., given the same initial conditions, to return the same outputs as the measurements of the pendulum's outputs.

The pendulum is governed by the following equation:

$$\ell\ddot{\alpha}(t) + \beta\dot{\alpha}(t) + g\sin\alpha(t) = 0 \tag{3.13}$$

where α is the angle of the pendulum with respect to the vertical axis, β is the viscous damping coefficient, g is the gravitational acceleration and ℓ is the length of the pendulum (see Figure 3.14). Then, (3.13) can be rewritten as:

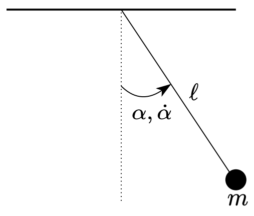


Figure 3.14: Scheme of the pendulum.

$$\ddot{\alpha}(t) + \frac{\beta}{\ell}\dot{\alpha}(t) + \omega^2\sin\alpha(t) = 0 \tag{3.14}$$

$$\omega = \sqrt{\frac{g}{\ell}} \tag{3.15}$$

The system can be reformulated in a state-space form defining with $x_1 = \alpha$, $x_2 = \dot{\alpha}$ and $x(t) = \begin{bmatrix} \alpha(t) & \dot{\alpha}(t) \end{bmatrix}^\top$:

$$\dot{x}(t) = \begin{bmatrix} \dot{\alpha}(t) \\ \ddot{\alpha}(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -\omega^2\sin x_1(t) - \frac{\beta}{\ell}x_2(t) \end{bmatrix} \tag{3.16}$$

$$y(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(t) \tag{3.17}$$

We want that the output of the REN-ODE at each time instant to be as close as possible to the measurements of the pendulum. In order to do so, the loss function used for this task is the Mean Squared Error(MSE):

$$L(y,\hat{y}) = MSE(y,\hat{y}) = \frac{1}{\eta}\sum_{i=0}^{\eta}\sum_{t=0}^{T_{end}}\|y^i(t) - \hat{y}^i(t)\|^2 \tag{3.18}$$

where the loss is evaluated on a batch of η experiments and $y, \hat{y}$ are respectively the measured outputs vector and the predicted one. In order to use REN-ODE for system identification, the following strategy has been used: starting from N random initial conditions on the angle α and the velocity $\dot{\alpha}$, the N experiments of the mechanical system are simulated from T_0 up to time T_{end}, chosen with respect to the dynamics of the pendulum; afterwards, η values are used as initial conditions of the neural network and then the model is let free to evolve up to the time T_{end}. Finally, the estimated outputs are compared with the previously simulated ones through the MSE loss function. Then the system's parameters θ are update using a learning strategy characterised by the chosen optimizer (for details, see Section 3.3). The scheme of the implementation is reported in Figure 3.15. The initial conditions have been sampled from the following distributions:

$$\begin{aligned} \alpha(0) &\sim \mathcal{U}\left(-\frac{\pi}{2}\,;\,\frac{\pi}{2}\right) \\ \dot{\alpha}(0) &\sim \mathcal{N}\left(0\,;\,\left(\frac{\pi}{180}\right)^2\right) \end{aligned}$$

where $\mathcal{U}(a, b)$ denotes a uniform distribution with boundaries a, b and $\mathcal{N}(\mu, \sigma^2)$ represents a normal distribution with μ mean and σ^2 variance. It must be noticed that for (free evolution) system identification, the properties of dissipativity and passivity (i.e., RREN-ODE) have not been enforced due to the lack of any sort of exogenous input $u(t)$. For this reason, experiments have been carried out only for CREN-ODEs. The simulations have been carried out with the following parameters:

Parameter	**Value**
β	0.75 m/s
ℓ	0.5 m
T_0	0 s
T_{end}	10.0 s
No. experiments	400
batch-size	20 exp.

The system has been chosen with an augmented dimension $\tilde{n} = 7$ and $q = 3$. CREN-ODE was trained mainly using two different integration methods: 'rk4' and 'euler' (for details Section 3.1). 'rk4' was able to obtain way better results, but with a longer time required by 'euler' to be trained: An additional comment about the choice of the integration method

Integration Method	**Time for Training**	**Final Loss (200 exp.)**
Forward Euler	1264 s	1.67 e-3
Runge-KuttaIV	3762	4.85 e-4

is that Euler is "higly" conditionally stable (i.e., the stability depends on the choice of the sampling time chosen for the ODE solver) and it has shown during the simulations an higher chance to reach numerical instability, causing the loss of all the results obtained up to that moment. On the other hand, being rk4 more accurate, has experienced less cases of numerical instability. For this reason, it is suggested to use, if possible, higher order methods.

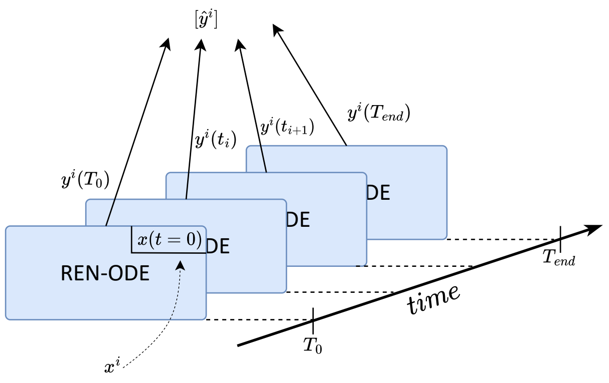


Figure 3.15: Scheme of the implementation of a REN-ODE for system identification of a free evolution system with initial condition x^i simulated from T_0 to T_{end}. The predicted output of the net is the vector $\hat{y}^i$.

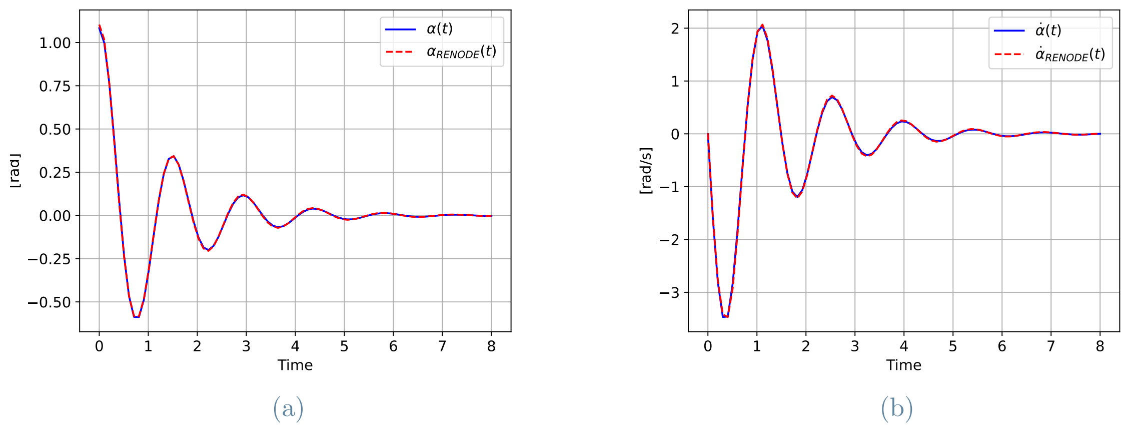


Figure 3.16: Validation of the REN-ODE, starting from an initial condition $(\alpha(0), \dot{\alpha}(0))$ not used during the training phase. In blue the actual trajectories and red the predicted one by the net.

In Figure 3.16 the trained model is tested comparing the evolution of the pendulum's states and the prediction of the REN-ODE, starting from an initial state condition $(\alpha(0), \dot{\alpha}(0))$ not used during the training phase.

3.5. Use Case 3: Optimal Control

The last considered use case of the REN-ODE is the optimal control of a dynamical system that consists of two robots that must reach two different targets, while avoiding any kind of collision and obstacles. A scheme of it is represented in Figure 3.17. Each robot has been modeled as a 2D point mass, subject to drag forces (e.g., friction) with state (p_t, q_t), where $p_t \in \mathbb{R}^2$ and $q_t \in \mathbb{R}^2$ represent the position and speed, respectively. For each agent:

$$\begin{bmatrix} \dot{p}_t \\ \dot{q}_t \end{bmatrix} = \begin{bmatrix} q_t \\ m^{-1}(-\mathcal{C}(q_t)q_t + F_t) \end{bmatrix} \tag{3.19}$$

where m is the mass of the robot, $F_t \in \mathbb{R}^2$ is the force control input and $\mathcal{C} : \mathbb{R}^2 \to \mathbb{R}$ is the *drag-force*, modeled as: $\mathcal{C}(q_t) = b_1 + b_2|q_t|$, with $b_1, b_2 \in \mathbb{R}^{1\times 2}$. The robots need to achieve the target position $\bar{p} \in \mathbb{R}^2$ with zero velocity, i.e., $\bar{q} = 0$ in a finite time window T_{end} (or less, if possible). For this setting, we consider that a base controller $\bar{F}_t = -K(p_t - \bar{p})$ has already been implemented for each agent, acting as a virtual spring able to push the robots linearly to the targets with $K = \text{diag}(k_1, k_2)$ and $k_1 = k_2 \in \mathbb{R}$. This base controller, thus, makes the target points a global asymptotically equilibrium of the system. For each agent, thus, the control input will be $F'_t \in \mathbb{R}^2$ and $F_t = \bar{F}_t + F'_t$. Finally the total model of the system can be considered, defining the overall state and input

$$x_t = [p_t^1, q_t^1, p_t^2, q_t^2] \in \mathbb{R}^8 \tag{3.20}$$
$$u_t = [F_t^1, F_t^2] \in \mathbb{R}^4 \tag{3.21}$$

This scenario is motivated by the examples in [46, 53]. The scheme of the implementation of a REN-ODE as a controller has been previously reported in Figure 3.2. The REN-ODE's control optimal policy is trained considering the following loss function, sampled with a sampling period T_s:

$$\sum_{k=0}^{T_{end}/T_s} l(x_{t_k}, u_{t_k}) = \sum_{k=0}^{T_{end}/T_s} l_{traj}(x_{t_k}, u_{t_k}) + l_{ca}(x_{t_k}) + l_{obst}(x_{t_k}) \tag{3.22}$$

where $(\cdot)_{t_k}$ denotes the value of a signal sampled at time $t = k\,T_s$ and each loss function is:

$$l_{traj}(x_{t_k}, u_{t_k}) = \sum_{i=1}^{2} \alpha_x [p_{t_k}^i, q_{t_k}^i]^\top \mathcal{Q} [p_{t_k}^i, q_{t_k}^i] + \alpha_u (u_{t_k}^i)^\top (u_{t_k}^i) \tag{3.23}$$

$$l_{ca}(x_{t_k}) = \begin{cases} \alpha_{ca}(d_{12}(t_k) + \epsilon)^{-2} & \text{if } d_{12} \leq D, \\ 0 & \text{otherwise.} \end{cases} \tag{3.24}$$

with $\alpha_x, \alpha_u, \alpha_{ca}$ weights of each policy, $d_{12}(t_k)$ the euclidean distance between the two robots at time t_k ($d_{12} = d_{21}$), D a "safe" value of distance, $\mathcal{Q}$ a positive definite matrix of weights and $\epsilon > 0$ small used to have (3.24) well-defined. Finally, we denote with:

$$\mathcal{N}(p;\mu;\Sigma) = \frac{1}{2\pi\sqrt{det(\Sigma)}} \exp\left(-\frac{1}{2}(p-\mu)^\top \Sigma^{-1}(p-\mu)\right) \tag{3.25}$$

a Gaussian density function with mean $\mu \in \mathbb{R}^2$ and covariance $\Sigma \in \mathbb{R}^{2\times 2}$. Then, the term $l_{obst}(x_{t_k})$ is given by:

$$l_{obst}(x_{t_k}) = \alpha_{obst} \sum_{i=1}^{2} \left(\mathcal{N}\left(p^i_{t_k}; \begin{bmatrix} 2.5 \\ 0 \end{bmatrix}; 0.2I\right) + \mathcal{N}\left(p^i_{t_k}; \begin{bmatrix} -2.5 \\ 0 \end{bmatrix}; 0.2I\right)\right) \tag{3.26}$$

$$+ \left(\mathcal{N}\left(p^i_{t_k}; \begin{bmatrix} 1.5 \\ 0 \end{bmatrix}; 0.2I\right) + \mathcal{N}\left(p^i_{t_k}; \begin{bmatrix} -1.5 \\ 0 \end{bmatrix}; 0.2I\right)\right) \tag{3.27}$$

where α_{obst} is a weight and the points $\begin{bmatrix} -1.5 & 0 \end{bmatrix}^\top$, $\begin{bmatrix} 1.5 & 0 \end{bmatrix}^\top$, $\begin{bmatrix} -2.5 & 0 \end{bmatrix}^\top$, $\begin{bmatrix} 2.5 & 0 \end{bmatrix}^\top$ are the locations of the obstacles (see Figure 3.17), denoted in [46] as "mountains". In order to study REN-ODE with different properties introduced in this thesis, the measured outputs of the system are the velocities of the robots: in this way $m \equiv p \equiv 4$. It must be pointed out that the model was trained on multiple initial positions of the two robots; indeed, their initial positions were sampled from the following normal distributions:

$$p_0^1 \sim \mathcal{N}\left(\begin{bmatrix} -2 \\ -2 \end{bmatrix}; 0.25I\right) \quad , \qquad p_0^2 \sim \mathcal{N}\left(\begin{bmatrix} 2 \\ -2 \end{bmatrix}; 0.25I\right)$$

where $I \in \mathbb{R}^{2\times 2}$ is the identity matrix. The model has been simulated with the following parameters:

Parameter	**Value**	**Parameter**	**Value**
m	1 kg	T_s	1 s
b_1	3 kg/s	T_{end}	70 s
b_2	0 kg/m	n	20
$k_1 = k_2$	0.5 N/m	q	4
α_x	100	$\sigma(\cdot)$	$tanh(\cdot)$
α_u	0.15	No. experiments	750 exp.
α_{obst}	1	Batch size	75 exp.
α_{ca}	50	Safe distance D	0.7 m
$\mathcal{Q}$	diag([1 1])		

The experiment has been performed considering different types of controllers guaran-

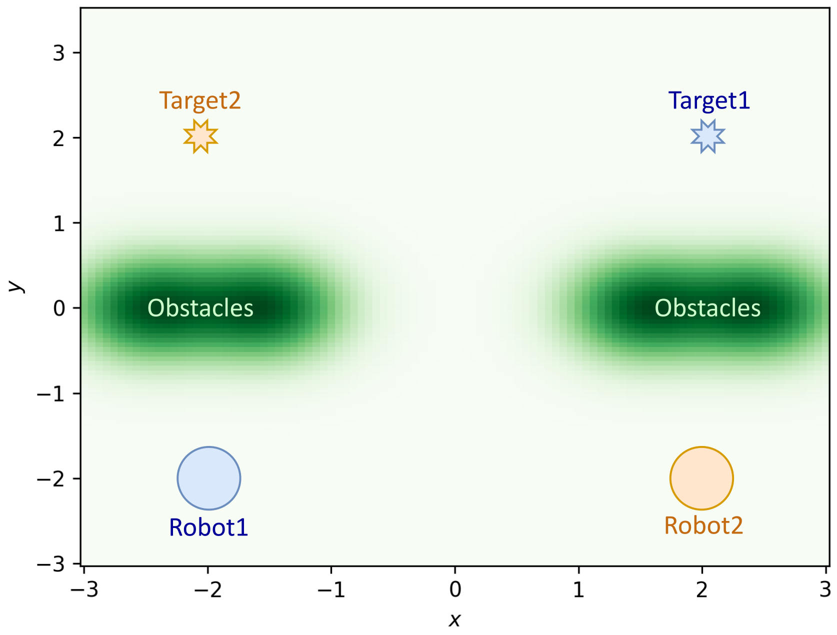


Figure 3.17: Scheme of the problem to optimally control. The two robots must reach the targets while avoiding crushes or collisions with the obstacles.

teeing contractivity and/or dissipativity. Due to different number of free parameters, CREN-ODE and RREN-ODEs did not have the same free parameters. The results of the simulations are reported in the following table, in which the same hyper-parameters and data have been used: Highlighted in blue the best result (a passive system with a

	Loss on Testing set				
Model	$loss_x$	$loss_u$	$loss_{ca}$	$loss_{obst}$	$Total$
Contractive	38	583	1.21 e3	744	2575
ℓ_2-bounded ($\gamma = 1$)	42	208	327	473	1050
Passive	39	244	105	124	511
Input Passive ($\nu = 0.01$)	39	269	0.0	126	433
Input Passive ($\nu = 0.1$)	38	18	314	121	661
Input Passive ($\nu = 1$)	62	2.76 e4	5.64 e5	184	5.92 e5

Table 3.1: Loss evaluated on the same Testing set of different models of REN-ODE.

small coefficient $\nu = 0.01$) whose plots of positions, velocities and inputs are reported in Figure 3.18 and, additionally, its optimal path is represented in Figure 3.19. The model has showed zero collision for any of the 375 experiments that compose the testing set, considering as safe distance the value $D = 0.7m$: this can be noted by $loss_{ca}$ that is equal to 0.

It is important to notice that the overall feedback-loop system is composed by passive systems. In fact, as shown in [44, Theorem 10], an incrementally passive system, is also *generally* passive (see Definition 1.2.3) for a given equilibrium point. In this experiment we chose to set the vector $\tilde{b}$ of biases to zero. Now, it is easy to verify that the system has an equilibrium point in zero. Additionally, we want to check if the two robots are passive. This property can be verified finding a matrix $\Phi = \Phi^\top \geq 0$ such that the following LMI are satisfied:

$$A^\top \Phi + \Phi A \leq 0\,, \qquad B^\top \Phi = C \tag{3.28}$$

where A, B, C are the matrix that characterized the linear system of the two robots (the coefficient $b_2 = 0$ and it follows that the system is linear). The proof of (3.28) can be found in [22, Proposition 4.1.2]. In this case, the matrices have the following values:

$$A_{robot} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.5 & 0 & -3 & 0 \\ 0 & -0.5 & 0 & -3 \end{bmatrix} \rightarrow A = \begin{bmatrix} A_{robot} & 0_{4\times4} \\ 0_{4\times4} & A_{robot} \end{bmatrix}\,, \tag{3.29}$$

$$B_{robot} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \rightarrow B = \begin{bmatrix} B_{robot} & 0_{4\times2} \\ 0_{4\times2} & B_{robot} \end{bmatrix}\,, \tag{3.30}$$

$$C_{robot} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow C = \begin{bmatrix} C_{robot} & 0_{2\times4} \\ 0_{2\times4} & C_{robot} \end{bmatrix}\,. \tag{3.31}$$

And thus, a solution can be:

$$\Phi = \mathrm{diag}\left(\begin{bmatrix} 0.5 & 0.5 & 1 & 1 & 0.5 & 0.5 & 1 & 1 \end{bmatrix}\right). \tag{3.32}$$

This discovery is important because the negative feedback interconnection of RREN-ODE and the two robots is still passive, thanks to Theorem 2.4. It has been tested that, even if the problem was solved for a fixed time horizon T_{end}, the system will not diverge if the time is prolonged. Indeed, the system will remain indefinitely in the equilibrium point, independently from time.

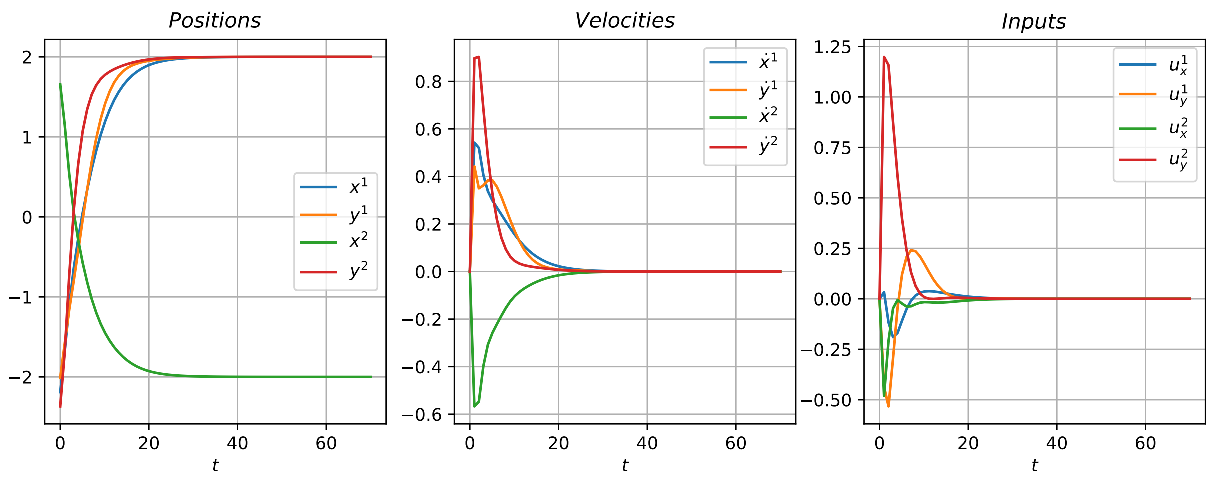


Figure 3.18: Plot of the positions, velocities and inputs with time, respectively simulated by the RREN-ODE strictly input passive with $\nu = 0.01$.

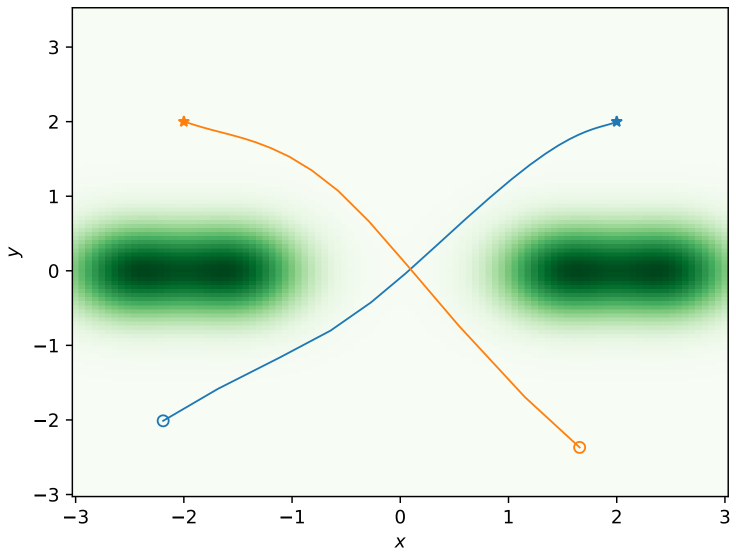


Figure 3.19: Path generated by the trained RREN-ODE strictly input passive with $\nu = 0.01$ starting from a random initial position of the robots.

4 | Conclusions and future developments

In this thesis, a new class of deep neural networks in continuous time, named REN-ODE, has been presented. We show that, through direct parametrization, it is always possible to obtain a contractive dynamical model by design and to, additionally, satisfy *integral quadratic constraints* (IQCs). IQCs can be used to enforce properties of incremental dissipativity and passivity, as well as Lipschitz bounds. The main results of my thesis are the mathematical characterizations of the class of REN-ODE that guarantee contractivity and robustness (called CREN-ODE and RREN-ODE, respectively). Hence, there are no constraints on the parameters and they can be learnt through the use of the most modern iterative optimization algorithms such as gradient descent and its variants (e.g., Adam, AMSGrad). The structure of the proposed class has been inspired by RENs[11] which, however, are formulated in discrete time. REN-ODE's architecture belongs to the family of Neural-ODEs [20]. As a result, REN-ODE inherits all the advantages of this neural network class, including the possibility to use modern and sophisticated ODE solvers for the evaluation of the output. Furthermore, these black-box ODE solvers are also adopted during the training phase, in which they can provide high level of precision and adapt the evaluation strategy on the fly to achieve the requested level of accuracy. Thanks to their flexibility, REN-ODEs can be used in different tasks in the control field. In this work, we have tested our new architecture, using it to identify the model of a nonlinear pendulum with different integration methods. Additionally, CREN-ODEs have been used in binary classification benchmarks from literature, obtaining great results. Finally, we implemented a RREN-ODE in a multi agent control scenario, where it was used as a regulator in order to be optimal with respect to a given policy function. RREN-ODE was able to achieve good performances, while trying to avoid obstacles and collisions between the agents. The guarantees of contractivity and robustness of this new class of neural networks, opens up the possibility in which REN-ODE's properties will be exploited, such as modeling of more challenging and complex systems like *reaction-diffusion systems*[54] or *continuous normalizing flows*[55]. Furthermore, future works will regard the use of REN-ODEs in distributed and decentralized cases, in which large-scale systems will be considered.

Bibliography

[1] R. Gargeya and T. Leng, "Automated identification of diabetic retinopathy using deep learning," *Ophthalmology*, vol. 124, no. 7, pp. 962–969, 2017.

[2] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[3] S. Ö. Arık, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, X. Li, J. Miller, A. Ng, J. Raiman, *et al.*, "Deep voice: Real-time neural text-to-speech," in *International Conference on Machine Learning*, pp. 195–204, PMLR, 2017.

[4] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, jul 2021.

[5] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[6] X. Wang, Y. Li, and K.-W. Kwok, "A survey for machine learning-based control of continuum robots," *Frontiers in Robotics and AI*, vol. 8, p. 730330, Sept. 2021.

[7] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, "Machine learning for smart building applications: Review and taxonomy," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–36, 2019.

[8] M. Said, K. b. Abdellafou, and O. Taouali, "Machine learning technique for data-driven fault detection of nonlinear processes," *Journal of Intelligent Manufacturing*, vol. 31, no. 4, pp. 865–884, 2020.

[9] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma,

J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, no. 1, pp. 1–74, 2021.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[11] M. Revay, R. Wang, and I. R. Manchester, "Recurrent Equilibrium Networks: Flexible Dynamic Models with Guaranteed Stability and Robustness," July 2021.

[12] M. Cheng, J. Yi, P.-Y. Chen, H. Zhang, and C.-J. Hsieh, "Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 3601–3608, Apr. 2020.

[13] M. Revay and I. R. Manchester, "Contracting implicit recurrent neural networks: Stable models with improved trainability," 2019.

[14] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing lipschitz continuity," *Machine Learning*, vol. 110, no. 2, pp. 393–416, 2021.

[15] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[16] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using lipschitz bounds," *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2021.

[17] K. J. Åström, P. Hagander, and J. Sternby, "Zeros of sampled systems," *Automatica*, vol. 20, no. 1, pp. 31–38, 1984.

[18] S. Liang, C. Zeng, I. Mitsuaki, and J. Zhong, "The roles of sampling zero dynamics in the discrete-time models for linear and nonlinear systems," in *Proceedings of the 33rd Chinese Control Conference*, pp. 3887–3892, 2014.

[19] K. Ogata, *Discrete-time control systems.* Prentice-Hall, Inc., 1995.

[20] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," 2018.

[21] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[22] A. Van der Schaft, *L2-gain and passivity techniques in nonlinear control.* Springer, 2000.

[23] M. Revay, R. Wang, and I. R. Manchester, “A convex parameterization of robust recurrent neural networks,” 2020.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[25] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[26] M. Revay, R. Wang, and I. R. Manchester, “Lipschitz bounded equilibrium networks,” 2020.

[27] E. K. Ryu and S. Boyd, “Primer on monotone operator methods,” *Appl. Comput. Math*, vol. 15, no. 1, pp. 3–43, 2016.

[28] Y. Lu, A. Zhong, Q. Li, and B. Dong, “Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations,” 2017.

[29] E. Haber and L. Ruthotto, “Stable architectures for deep neural networks,” *Inverse Problems*, vol. 34, p. 014004, dec 2017.

[30] L. S. Pontryagin, “The mathematical theory of optimal processes,” 1962.

[31] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[32] X. Zhang, Z. Li, C. Change Loy, and D. Lin, “Polynet: A pursuit of structural diversity in very deep networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 718–726, 2017.

[33] G. Larsson, M. Maire, and G. Shakhnarovich, “Fractalnet: Ultra-deep neural networks without residuals,” *arXiv preprint arXiv:1605.07648*, 2016.

[34] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[35] C. Runge, “Ueber die numerische auflösung von differentialgleichungen.,” *Mathematische Annalen*, vol. 46, pp. 167–178, 1895.

[36] W. Kutta, "Beitrag zur naherungsweisen integration totaler differentialgleichungen," *Z. Math. Phys.*, vol. 46, pp. 435–453, 1901.

[37] *Solving Ordinary Differential Equations I.* Springer Berlin Heidelberg, 1993.

[38] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," 2018.

[39] E. Dupont, A. Doucet, and Y. W. Teh, "Augmented neural odes," 2019.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[42] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[43] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[44] C. Verhoek, P. J. Koelewijn, R. Tóth, and S. Haesaert, "Convex incremental dissipativity analysis of nonlinear systems," *arXiv preprint arXiv:2006.14201*, 2020.

[45] P. Al Hokayem and E. Gallestey, "Lecture notes on nonlinear systems and control spring semester 2020 eth zurich,"

[46] L. Furieri, C. L. Galimberti, M. Zakwan, and G. Ferrari-Trecate, "Distributed neural network control with dependability guarantees: a compositional port-hamiltonian approach," in *Learning for Dynamics and Control Conference*, pp. 571–583, PMLR, 2022.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[48] R. T. Q. Chen, "torchdiffeq," 6 2021.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Courna-

peau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[50] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[51] E. Haber and L. Ruthotto, “Stable architectures for deep neural networks,” *Inverse problems*, vol. 34, no. 1, p. 014004, 2017.

[52] L. Prechelt, “Early stopping-but when?,” in *Neural Networks: Tricks of the trade*, pp. 55–69, Springer, 1998.

[53] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto, “A neural network approach applied to multi-agent optimal control,” in *2021 European Control Conference (ECC)*, pp. 1036–1041, IEEE, 2021.

[54] R. Martin and M. Pierre, “Nonlinear reaction-diffusion systems,” in *Nonlinear Equations in the Applied Sciences* (W. Ames and C. Rogers, eds.), vol. 185 of *Mathematics in Science and Engineering*, pp. 363–398, Elsevier, 1992.

[55] D. Rezende and S. Mohamed, “Variational inference with normalizing flows,” in *International conference on machine learning*, pp. 1530–1538, PMLR, 2015.

A | Appendix A

Proposition A.1. *If Q is a negative definite matrix and $R = R^\top$, then the LMI in* (1.38) *implies the* (1.36).

Proof. Considering the inequality in (1.38), reported here for clarity:

$$\begin{bmatrix} P & -C_1^\top \Lambda & C_2^\top S^\top \\ -\Lambda C_1 & W & D_{21}^\top S^\top - \Lambda D_{12} \\ SC_2 & SD_{21} - D_{12}^\top \Lambda & R + SD_{22} + D_{22}^\top S^\top \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \\ B_2^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \\ B_2^\top \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \\ D_{22}^\top \end{bmatrix}^\top > 0 \tag{A.1}$$

Using the *Sylvester's criterion*, it implies that:

$$\begin{bmatrix} P & -C_1^\top \Lambda \\ -\Lambda C_1 & W \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix}^\top + \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix}^\top > 0 \tag{A.2}$$

Being Q definite negative, (A.2) can be rewritten as:

$$\begin{bmatrix} P & -C_1^\top \Lambda \\ -\Lambda C_1 & W \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix}^\top > - \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix} Q \begin{bmatrix} C_2^\top \\ D_{21}^\top \end{bmatrix}^\top > 0 \tag{A.3}$$

Finally:

$$\begin{bmatrix} P & -C_1^\top \Lambda \\ -\Lambda C_1 & W \end{bmatrix} - \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix} P \begin{bmatrix} A^\top \\ B_1^\top \end{bmatrix}^\top > 0 \tag{A.4}$$

that corresponds to (1.36). □

B | Appendix B

We want to prove the following proposition:

Proposition B.1. *Given the Assumption 1 and a REN-ODE in the form* (2.1)-(2.2) *and its incremental form* (2.3)-(2.4) *such that:*

$$\begin{bmatrix} -A^\top P - PA & -C_1^\top \Lambda - PB_1 \\ -\Lambda C_1 - B_1^\top P & W \end{bmatrix} > 0 \tag{B.1}$$

with P a positive definite matrix, Λ a positive diagonal matrix and

$$W = 2\Lambda - \Lambda D_{11} - D_{11}^\top \Lambda. \tag{B.2}$$

then $\exists \alpha > 0$ such that:

$$\dot{V}_\Delta(t) \leq -\alpha V_\Delta(t) \tag{B.3}$$

where $V_\Delta(t)$ is a quadratic Lyapunov function defined as follows:

$$V_\Delta(t) = \Delta x_t^\top P \Delta x_t \tag{B.4}$$

Proof. Starting from the definition of $V_\Delta(t)$ and the definition of $\Delta \dot{x}_t$ from (2.3), then:

$$\begin{aligned} \dot{V}_\Delta(t) &= \Delta \dot{x}_t^\top P \Delta x_t + \Delta x_t^\top P \Delta \dot{x}_t && \text{(B.5)} \\ &= (\Delta x_t^\top A^\top + \Delta w_t^\top B_1^\top) P \Delta x + \Delta x^\top P (A \Delta x_t + B_1 \Delta w_t) && \text{(B.6)} \\ &= \begin{bmatrix} \Delta x_t^\top \\ \Delta w_t^\top \end{bmatrix}^\top \begin{bmatrix} A^\top P + PA & PB_1 \\ B_1^\top P & 0_{q \times q} \end{bmatrix} \begin{bmatrix} \Delta x_t^\top \\ \Delta w_t^\top \end{bmatrix} && \text{(B.7)} \end{aligned}$$

However, we showed during the proof of Theorem 2.1 that, using the definition of Γ_t (1.32) and Assumption 1, it is possible to write that:

$$\begin{bmatrix} \Delta x_t^\top \\ \Delta w_t^\top \end{bmatrix}^\top \begin{bmatrix} A^\top P + PA & PB_1 \\ B_1^\top P & 0_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta w_t \end{bmatrix} < -\Gamma_t \leq 0\,, \quad \forall t \tag{B.8}$$

Defining with M the matrix:

$$M = \begin{bmatrix} A^\top P + PA & PB_1 \\ B_1^\top P & 0_{q\times q} \end{bmatrix} \tag{B.9}$$

Then, due to (B.8), the square matrix $M \in \mathbb{R}^{(n+q)\times(n+q)}$ is definite negative, i.e.:

$$\begin{bmatrix} \Delta x_t^\top \\ \Delta w_t^\top \end{bmatrix}^\top M \begin{bmatrix} \Delta x_t \\ \Delta w_t \end{bmatrix} < 0\,; \qquad \forall \begin{bmatrix} \Delta x_t \\ \Delta w_t \end{bmatrix} \neq 0 \tag{B.10}$$

Being M a symmetric matrix, its eigenvalues are real and negative.
Defining with $\lambda_{min}(M)$ and $\lambda_{max}(M)$ the minimum and maximum eigenvalue of M, respectively (with $\lambda_{min}(M) \leq \lambda_{max}(M) < 0$), then it is always possible to write:

$$\lambda_{min}(M)\|\Delta z\|^2 \leq \Delta z^\top M \Delta z \leq \lambda_{max}(M)\|\Delta z\|^2 < 0 \quad \forall \Delta z \neq 0 \tag{B.11}$$

where Δz is defined as:

$$\Delta z = \begin{bmatrix} \Delta x_t \\ \Delta w_t \end{bmatrix} \tag{B.12}$$

Thus, using (B.7), (B.11) and (B.12):

$$\dot{V}_\Delta(t) = \Delta z^\top M \Delta z \leq \lambda_{max}(M) \left\| \begin{matrix} \Delta x_t \\ \Delta w_t \end{matrix} \right\|^2 < 0\,, \quad \forall \Delta x_t, \Delta w_t : \left\| \begin{matrix} \Delta x_t \\ \Delta w_t \end{matrix} \right\|^2 \neq 0 \tag{B.13}$$

Defining with γ:

$$\gamma = -\lambda_{max}(M) > 0 \tag{B.14}$$

Then (B.13) becomes:

$$\dot{V}_\Delta(t) \leq -\gamma \left\| \begin{matrix} \Delta x_t \\ \Delta w_t \end{matrix} \right\|^2 < 0\,, \quad \forall \Delta x_t, \Delta w_t : \left\| \begin{matrix} \Delta x_t \\ \Delta w_t \end{matrix} \right\|^2 \neq 0 \tag{B.15}$$

At this point, we can consider the symmetric and positive matrix P. P has only real and positive eigenvalues. Defining with $\lambda_{min}(P)$ and $\lambda_{max}(P)$ the minimum and maximum eigenvalue of P, respectively (with $0 < \lambda_{min}(P) \leq \lambda_{max}(P)$), then it always holds that:

$$0 < \lambda_{min}(P)\|\Delta x_t\|^2 \leq \Delta x_t^\top P \Delta x_t \leq \lambda_{max}(P)\|\Delta x_t\|^2 \,, \quad \forall \Delta x_t \neq 0 \tag{B.16}$$

We can now define the "augmented" matrix $P_0 \in \mathbb{R}^{(n+q)\times(n+q)}$ such that:

$$P_0 = \begin{bmatrix} P & 0_{n\times q} \\ 0_{q\times n} & 0_{q\times q} \end{bmatrix} \tag{B.17}$$

Notice that P_0 is a symmetric semi-definite matrix. Then it is always possible to write:

$$0 \leq \Delta z^\top P_0 \Delta z \leq \lambda_{max}(P)\|\Delta z\|^2 \,, \quad \forall\, \Delta z \tag{B.18}$$

Rearranging (B.18) and, being $\lambda_{max}(P) \neq 0$:

$$\|\Delta z\|^2 \geq \frac{\Delta z^\top P_0 \Delta z}{\lambda_{max}(P)} \tag{B.19}$$

Furthermore, using (B.4), (B.15) and (B.19):

$$\begin{aligned}
\dot{V}_\Delta(t) \leq -\gamma \left\| \begin{matrix} \Delta x_t \\ \Delta w_t \end{matrix} \right\|^2 &= -\gamma \frac{\Delta z^\top P_0 \Delta z}{\lambda_{max}(P)} && \text{(B.20)} \\
&= -\frac{\gamma}{\lambda_{max}(P)} \Delta z^\top \begin{bmatrix} P & 0_{n\times q} \\ 0_{q\times n} & 0_{q\times q} \end{bmatrix} \Delta z && \text{(B.21)} \\
&= -\frac{\gamma}{\lambda_{max}(P)} \begin{bmatrix} \Delta x_t^\top & \Delta w_t^\top \end{bmatrix}^\top \begin{bmatrix} P & 0_{n\times q} \\ 0_{q\times n} & 0_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta x_t \\ \Delta w_t \end{bmatrix} && \text{(B.22)} \\
&= -\frac{\gamma}{\lambda_{max}(P)} \Delta x_t^\top P \Delta x_t && \text{(B.23)} \\
&= -\frac{\gamma}{\lambda_{max}(P)} V_\Delta(t) && \text{(B.24)}
\end{aligned}$$

We can now define the parameter α as:

$$\alpha = \frac{\gamma}{\lambda_{max}(P)} > 0 \tag{B.25}$$

And finally, from (B.20) and the definition of α:

$$\dot{V}_\Delta(t) \leq -\alpha V_\Delta(t) \,, \qquad \forall t \tag{B.26}$$

□