Control and Cybernetics

vol. **44** (2015) No. 1

Dynamic network functional comparison via approximate-bisimulation*

by

Francesco Donnarumma[1], Aniello Murano[2] and Roberto Prevete[2]

[1]Institute of Cognitive Sciences and Technologies,
National Research Council of Italy
Via S. Martino della Battaglia, 44 - 00185, Rome, Italy
[2]Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione,
Università degli Studi di Napoli Federico II
Via Claudio, 21 - 80125, Napoli, Italy
francesco.donnarumma@istc.cnr.it,
aniello.murano@unina.it,
roberto.prevete@unina.it

Abstract: It is generally unknown how to formally determine whether different neural networks have a similar behaviour. This question intimately relates to the problem of finding a suitable similarity measure to identify bounds on the input-output response distances of neural networks, which has several interesting theoretical and computational implications. For example, it can allow one to speed up the learning processes by restricting the network parameter space, or to test the robustness of a network with respect to parameter variation. In this paper we develop a procedure that allows for comparing neural structures among them. In particular, we consider dynamic networks composed of neural units, characterised by non-linear differential equations, described in terms of autonomous continuous dynamic systems. The comparison is established by importing and adapting from the formal verification setting the concept of $\delta-$approximate bisimulations techniques for non-linear systems. We have positively tested the proposed approach over continuous time recurrent neural networks (CTRNNs).

Keywords: continuous time recurrent neural network, dynamic networks, bisimulation, network equivalence

1. Introduction

1.1. An outline for the problem

In recent years, a growing number of studies in computational neuroscience has focused on the question whether, in neural-network models having different

*Submitted: March 2014; Accepted: February 2015 parameters, similar input-output behaviours can be realized (see, for example, Prinz et al., 2004; Marder and Goaillard, 2006; DiMattina and Zhang, 2010). This question naturally leads to the problem of defining a suitable similarity measure to functionally compare distinct neural networks.

The problem of specifying a computational procedure for a similarity measure which enables one to identify bounds on the input-output response distances of distinct neural networks has several interesting theoretical and computational implications. In the field of computational neuroscience, for example, such a procedure would allow one to quantitatively compare the network behaviours resulting from widely differing combinations of intrinsic and synaptic properties (MacGregor and Tajchman, 1988).

Interestingly, the above question is of great relevance in many engineering applications such as pattern recognition problems. For example, in supervised learning approaches (Bishop, 2006), different kinds of algorithms (e.g. gradient descent and evolutionary optimization techniques) are used to find the network-parameter values that minimize any functional error on the basis of some given training data set. Multiple minima give rise to multiple equivalent solutions to the same problem. Thus, the search for parameters by a supervised learning approach can often result in an ill-posed problem, insofar as one cannot uniquely identify a neural network from the training data only. A well-posed problem can be obtained in some special cases (e.g., standard MLP models): under special assumptions on the neuron output functions, the overall input-output relationship of the network uniquely determines the values of all network parameters, up to a permutation of neurons and the regrouping of identical neurons (Albertini and Sontag, 1993; Albertini et al., 1993; Wu et al., 2006). As a consequence, there are situations in which one may completely recover, in principle, the entire structure of a neural network just from the training data (Albertini and Sontag, 1993; Fefferman and Markel, 1994). However, the uniqueness of such results relies on the assumption that noiseless and potentially unlimited training data are available, thus restricting their applicability in the modelling of real data. A similarity measure may enable one to extend this class of well-posed problems insofar as one may determine classes of parameters which correspond to "equivalent" or "sufficiently similar" neural networks. In this way, a functional mapping between training data and parameter classes would be possible, thereby turning the search for network parameters by a supervised approach into a well-posed problem.

In addition to this, a similarity measure would allow one to speed up the learning processes by restricting the parameter space during the learning phase (Neruda, 2000). Moreover, a similarity measure might be used to test and compare distinct neural networks (Horne and Giles, 1995) obtained by means of different learning algorithms. Consequently, this can enable one to choose between neural networks that look similar in terms of functionality by means of suitable heuristics over the complexity of the network architecture and specific requirements of the problem to solve.

The similarity question about neural-network models could be rephrased in the formal method framework (see Clarke et al., 2000). Such a framework provides powerful techniques to automatically and exhaustively check whether a system satisfies a desired behaviour by checking whether a mathematical model of the system meets a mathematical representation of the desired behaviour. However, for complex systems, such as non-linear dynamic networks, it is an open issue how to properly establish such a procedure (see Casagrande et al., 2012).

1.2. Our contribution

In this paper we develop an effective procedure allowing one to compare different given neural structures when the parameters are varied. In particular, we consider dynamic networks composed of neural units characterised by non-linear differential equations and describable as autonomous continuous dynamical systems (Gupta et al., 2003; Munakata, 1997). We emphasize that dynamic networks are widely used in computational neuroscience as models of brain regions or subsets of biological neurons (Sporns, 2011; Chersi et al., 2013; Pezzulo et al., 2013; MacGregor, 2012; Izhikevich, 2007).

We collect a series of results in dynamical system theory and bisimulation theory in order to establish a dynamic network similarity (DyNeS) algorithm enabling one to functionally compare different dynamic networks. The comparison of networks is established by finding upper bounds between distance of trajectories, solutions of the equation of two neural networks. This is obtained by importing and adapting from formal verification framework $\delta-$approximate bisimulations techniques for non-linear systems (Girard and Pappas, 2005) to dynamic networks.

We stress that our approach introduces the bisimulation framework to non-linear dynamic networks, which results in the possibility of comparing the behaviour of different networks by means of the estimation of a bound δ between the systems under investigation. Without loss of generality, the present approach is tested within the continuous time recurrent neural network (CTRNN) framework, a popular network model widely deployed both in robotics (see, e.g., Birch et al., 2002; De Falco et al., 2008; Paine and Tani, 2004; Montone et al., 2011) and in the field of computational models of biological neuronal phenomena (see, e.g., Dunn et al., 2004; Donnarumma et al., 2010, 2012). Finally, note that the techniques used in this paper stem from the very new research developments in dynamical system theory. This is the first attempt, to our best knowledge, to specialize bisimulation within the framework of non-linear dynamic networks.

1.3. Related work

In the field of static neural networks there exist a number of results about similarity between networks. It has been proved that the input-output functionality uniquely determines the network structure (Albertini and Sontag, 1993; Fefferman and Markel, 1994) in the case of infinite, noiseless data. Moreover, a biologically inspired method is presented in DiMattina and Zhang (2010), which allows for determining when the structure of a feed forward neural network can be gradually perturbed while preserving its functionality. This is accomplished by deriving a differential equation that specifies the conditions, under which the parameters of some given neural network can be continuously modified, while leaving the network functionality unchanged. Works by Amari and colleagues (Wei and Amari, 2008) characterised the behaviour of layered networks near singularities with the aim of helping in avoiding regions of parameters where standard gradient-based learning methods are stuck in large plateaus, greatly slowing training.

In the field of dynamic networks, a lot of efforts in the literature have been devoted to establish conditions related to stability (see, e.g., Cao et al., 2005; Yu and Yao, 2007; Chandrasekar et al., 2014): those approaches are linked to the method here presented, in the sense that the similarity we adopted can be seen as a relaxation of the stability conditions. On the other hand, in the formal method framework, different approximation techniques for extending formal methods to complex dynamical systems have been proposed (see, e.g., Fränzle, 1999; Ratschan, 2010; Girard and Pappas, 2007a; Casagrande et al., 2009; Prabhakar et al., 2009; Lall et al., 2002). Those techniques are crucial in order to effectively study properties related to dynamical systems by means of computational procedures. Hybrid automata with noise are presented in Fränzle (1999). The introduction of noise in many cases ensures the (semi-)decidability of the reachability problem. Another result of (semi-)decidability, again related to the concept of perturbation, is given in Ratschan (2010). Our approach is based on δ-(bi)simulation (Girard and Pappas, 2005) relations, which essentially corresponds to relaxations on the infinite precision, required by simulation and bisimulation. Such relations represent a tool capable of removing complexity and undecidability issues related to the analysis of the investigated model. Following this last strategy, Lyapunov-like conditions can be developed in order to find bisimulation functions, which are used to over-approximate the observational distance between two polynomial systems, defining the so called $\delta-$approximate bisimulations.

Unfortunately, when dealing with the standard non-linear form of output function (e.g., the sigmoidal function), none of these techniques can be straightforwardly applied to dynamic networks. In this respect, however, it should be noted that a number of simplified dynamic network models are considered in the literature. Studies reveal that approximation of output functions to cubic non-linearities still shows shapes, firing rates, and bursting behaviours throughout the physiological range (see Wilson, 1999). Second order polynomial approximations have also been considered in the literature (see, e.g., Rolls et al., 2006), receiving, however, less attention as to how those models could be related to the non-approximated ones. Linearizing the non-linear components of the original continuous model, by replacing the sigmoidal outputs with sign functions (Ghosh and Tomlin, 2001; De Jong et al., 2004), makes the continuous signals turn into discrete off-on signals. Unfortunately, it is proved that behaviours of those system models differ from the original one. A more sophisticated approximation of sigmoidals, based on a piecewise linear function, is exploited in the development of a hybrid automaton which simulates a single oscillator (Casagrande et al., 2012). We take a step further by considering "polynomialized" dynamic networks, for which we show the possibility of computing useful $\delta-$approximate bisimulations.

1.4. Work plan

The rest of the paper is organized as follows. In Section 2, we propose the mathematical background of the dynamic network similarity (DyNeS) procedure. The pseudocode is presented in Subsection 2.3. The remaining subsections are devoted to the explanation of the steps of the procedure and include the details on the systems in use (Subsection 2.2), the polynomial approximation of the output function of the networks (Subsection 2.4) and the $\delta-$ approximated bisimulation (Subsection 2.5). In Section 3, examples of the applications of the method are given, focusing on a particular model of dynamic networks, i.e. continuous time recurrent neural networks (CTRNN). Finally, in Section 4, conclusions are provided and future work on the approach is outlined.

2. Network similarity computation

2.1. Similarity measure

In this section, we fully describe the computational steps taken in order to obtain the network similarity measure for dynamic networks. The pseudo-code of a dynamic network similarity (DyNeS) procedure is presented in Subsection 2.3. We start by clarifying the idea underlying our approach and the kind of systems on which it is performed. To exemplify the idea of similarity measure, we make use of the popular framework of multi-layered perceptron (MLP) network models (see Gardner and Dorling, 1998).

Given an MLP network G, it is possible to write its input-output relation in terms of a functional relation $\mathbf{y} = f(\mathbf{x}, \boldsymbol{\theta})$, where f is a parametric non-linear function, $\mathbf{y} = (y^1, \ldots, y^c)$ is the output of the neurons belonging to the output layer, $\mathbf{x} = (x^1, \ldots, x^d)$ is the input to the network, and $\boldsymbol{\theta} = (\theta^1, \ldots, \theta^n)$ is the parameter set, encoding the network structure (e.g. synaptic weights and biases). Given two MLP networks G and $\bar{G}$ with different structures $\boldsymbol{\theta}$ and $\bar{\boldsymbol{\theta}}$, a neural network similarity measure should enable one to find δ-bounds on their responses, i.e., $\forall \mathbf{x} \in D \subseteq \mathbb{R}^d \left\| f(\mathbf{x}, \boldsymbol{\theta}) - f(\mathbf{x}, \bar{\boldsymbol{\theta}}) \right\| < \delta$, the value of such δ expressing a quantitative measure of how functionally close the networks G and $\bar{G}$ are. Thus, the problem of comparing two networks can be reformulated as finding a suitable δ that bounds so defined a distance between the two systems.

In the next section, we introduce the framework of continuous dynamic networks for which our approach has been developed. In contrast with MLP, continuous dynamic networks explicitly include time in the model. Moreover, we stress that this kind of neural networks models are widely used in neuroscience literature and engineering applications.

2.2. Dynamic networks framework

We consider the framework of continuous dynamic networks in which the evolution of the system is expressed by means of first-order differential equations. Consequently, it is possible to study them as dynamical systems:

Definition 1 *An* autonomous continuous dynamic system D *is a 3-ple* (Q, γ, T) *where*

- Q is a topological space named *state space*
- T is the *time set*
- $\gamma : (\mathbf{y}, t) \in Q \times T \longrightarrow Q$ is the *flow* given by the solution of the set of first-order ordinary differential equations (ODE)

$$\frac{d\mathbf{y}}{dt} = f(\mathbf{y}). \tag{1}$$

Thus, we refer to continuous dynamic networks when dealing with artificial neural network models satisfying Definition 1. In order to compare systems with a different number of variables, in Definition 2 the notion of a continuous dynamic network with observables is introduced.

Definition 2 *An* autonomous continuous dynamical system with observables $D = (Q, \gamma, T, h, H)$ *is an autonomous continuous dynamical system* $D = (Q, \gamma, T)$ *additionally equipped with*

- *an* observation space H*, which is a metric space* H *along with a metric* d
- *an* observation map $h : Q \to H$*, which maps variables of the state space* Q *to the observation space* H*, the variables in the space* H *are called* observables*.*

Accordingly, one can define *observable trajectories* given by

$$\{(t, h(\gamma(\mathbf{y}, t))) : \exists \mathbf{y} \in Q, t \in T \quad \gamma(\mathbf{y}, t) \in Q\}.$$

From Definition 2, it is clear that a continuous dynamic network with observables is a continuous dynamic network where a mapping between the state space and an observation space is introduced. Thus, two dynamic networks with a different number of variables can be compared if one considers the same number of observable variables from each system and compares their observable trajectories. In particular, in order to show our approach we focus our attention on a popular model of dynamic networks.

Definition 3 Continuous time recurrent neural networks (*CTRNNs*) *are networks of biologically inspired neurons (nodes) described by the following general equations* (*Hopfield and Tank, 1986; Beer, 1995*)*:*

$$\tau^i \frac{dy^i}{dt} = -y^i + \sum_{j=1}^{n} w^{ij} o(y^j + \theta^j) + I_e^i \qquad i \in \{1, \dots, n\} \tag{2}$$

where N *is the number of neurons in the network and for each neuron* i*:* τ^i *is the* time constant*,* y^i *is the* potential *or* activation *variable,* θ^i *is the* bias*,* $o(y^i + \theta^i)$ *is the* mean firing rate*, with* $o(\cdot)$ *the* output function*,* $I_e^i = \sum_{j=n+1}^{n+l} w^{ij} u^j$ *is an* external input *coming from* l *external sources* u^j*, and* w^{ij} *is the* weight *of the connection coming from the node* j*.*

From Equation (2), it is clear that CTRNN systems satisfy Definition 1 when network parameters together with the external inputs I_e^i are time independent. Usually, $o(x)$ is the sigmoidal function $\sigma(x)$. However, one can choose as $o(x)$ any *smooth*, *monotonic*, and *bounded* activation function; the resulting network being called additive CTRNN. For example, one may use the parametric form (Tino et al., 2001)

$$o(x) = \sigma_{a,b,c}(x) = \frac{a}{1 + e^{-c \cdot x}} + b \tag{3}$$

that has the advantage of reducing to the hyperbolic tangent function by substituting $a = 2$, $b = -1$, $c = 2$ in (3) as follows

$$\sigma_{2,-1,2}(x) = \frac{2}{1 + e^{-2 \cdot x}} - 1 = \frac{e^x}{e^x} \cdot \frac{1 - e^{-2 \cdot x}}{1 + e^{-2 \cdot x}} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \tanh(x)$$

or one may use the standard sigmoid with $a = 1$, $b = 0$, $c = 1$, formally obtained from (3) as follows

$$o(x) = \sigma_{1,0,1}(x) = \frac{1}{1 + e^{-x}} = \sigma(x)\,.$$

The sigmoidal output function $\sigma(x)$ is used throughout Section 3.

In the next subsection we will show how it is possible to find a clever approximation of the sigmoidal output function in order to apply the bisimulation techniques shown in Subsection 2.5. However, the approach presented here is quite general, insofar as similar results can be obtained for different activation functions fulfilling the smoothness, monotonicity and boundedness conditions (see Section 4).

2.3. DyNeS algorithm

It is possible to formalize in a pseudo-code fashion a similarity measure between dynamic network behaviours: in Algorithm 1 we present the main steps of the dynamic network similarity (DyNeS) procedure, which will be extensively clarified in the rest of this section.

Algorithm 1 DyNeS Algorithm

```
Require: Given two continuous dynamic network systems $D_1$ and $D_2$
1: select $k$ neurons from $D_1$ and $D_2$ respectively: this selection determines the comparing subspace on which the similarity will be computed;
2: select a proper polynomial approximation for the networks output function on the basis of the dynamic networks connection weights $W$;
3: choose a suitable form of a parametric bisimulation function $V^{\mathbf{c}}(\mathbf{y}) : \mathbf{y} \in \mathbb{R}^k \times \mathbb{R}^k \to V^{\mathbf{c}}(\mathbf{y}) \in \mathbb{R}^+$ with $\mathbf{y} = \{y_1^1, \ldots, y_1^k, y_2^1, \ldots, y_2^k\}$ being the concatenation of the observable variables of $D_1$ and $D_2$ and $\mathbf{c} = \{c_1, \ldots, c_L\}$ a suitable number $L$ of parameters. This bisimulation function enables one to establish a bisimilarity between $D_1$ and $D_2$, i.e., in a nutshell, that the solutions of the two systems are “sufficiently close”;
4: compute by means of bisimulation function $V^{\mathbf{c}}(\mathbf{y})$ an upper bound $\delta_{max}$ of the ”distance” between the solutions of $D_1$ and $D_2$.
```

The aim of the algorithm is to produce a δ_{max} between two dynamic networks, which bounds the trajectories of the two systems in time. This can be done by specifying a clever approximation of non-linear dynamic networks and then finding a bisimulation between the systems under investigation, which ensures that the trajectories have the property of being “sufficiently” close.

In Subsection 2.2 we described the kind of systems for which this procedure is applied (step 1). We will clarify the successive steps of Algorithm 1 in the rest of this section. Then, in Subsection 2.4 we show the approximation needed for these systems in order to find the required bisimulation (step 2); in Subsection 2.5 we describe the bisimulation framework that lets us define a superior bound on the measure of similarity between the systems we compare (steps 3-4).

2.4. Polynomial approximation of a network output function

In the presented framework of dynamic networks, it is possible to derive the following theorem:

Theorem 1 (Funahashi and Nakamura, 1993) *Given a dynamic network D from Definition 2, equipped with a sigmoidal output function, there exists C such that*

$$\|\mathbf{y}(t)\| \leq \sqrt{n} \cdot C$$

with

- $C = \max\{C^i\}$,
- $C^i = \max\{y^i(0), M\}$,
- $M = n \cdot w_{max} + I_{max}$,
- $w_{max} = \max\{w^{ij}\}_{i,j=1}^n$,

- $I_{max} = \max\{I_e^i\}_{i=1}^n$.

PROOF We can write the equations as

$$\dot{y}^i = -\frac{y^i}{\tau^i} + \frac{F^i(\mathbf{y}, W, I_e^i)}{\tau^i}.$$

It is possible to find a constant M such that $\left|F^i\right| \leq M$. In fact,

$$\begin{array}{lc} \left|F^i\right| & = \left|\sum_{j=1}^n w^{ij}\sigma(y^j+\theta^j) + I_e^i\right| \leq \sum_{j=1}^n \left|w^{ij}\right| \left|\sigma(y^j+\theta^j)\right| + \left|I_e^i\right| \quad < \\ < & \sum_{j=1}^n \left|w^{ij}\right| + \left|I_e^i\right| \leq n \cdot w_{max} + I_{max} = M \end{array}$$

with $w_{max} = \max\{\left|w^{ij}\right|\}_{i,j=1}^n$ and $I_{max} = \max\{\left|I_e^i\right|\}_{i=1}^n$. Consequently, we can write:

$$\left|y^i(t)\right| \leq \max\{\left|y^i(0)\right|, M\} = C^i$$

from which the statement directly descends. □

From Theorem 1 it directly results that the argument of the sigmoidal function σ is bounded in a finite interval depending on the constants $[-A, A]$, where $A = C + \theta_{max}$, with $\theta_{max} = \max\{\left|\theta^i\right|\}_{i=1}^n$. Thus, it is possible to use the following well known approximation result.

THEOREM 2 [Weierstrass, 1885] *For each continuous function $f : x \in [a,b] \to \mathbb{R}$ defined on the closed and bounded interval $[a,b]$, and $\forall\epsilon > 0$ there exists some polynomial $p(x)$ such that*

$$|p(x) - f(x)| < \epsilon.$$

Consequently, for each dynamic network, it is possible to effectively construct an interval in which to perform a clever approximation, "as good as we want" of the initial system (see Subsection 3.2). Of course, the better an approximation is, the more polynomial terms are required, making the search for the bisimulation a more computationally difficult problem (see Subsection 2.5). Therefore, a trade-off strategy must be chosen, depending on the specific purpose for which the procedure is applied (see Section 4 for discussions).

2.5. $\delta-$approximate bisimulation for dynamic networks

The concept of bisimulation, as applied to state transition systems in verification of hardware and software systems (Clarke et al., 2000), can be used to establish topological equivalence between dynamical systems (Hale and Koçac, 1991). Here, we show how to relax bisimulation equivalence in order to establish a functional similarity measure between continuous dynamic networks.

In Girard and Pappas (2005) the concept of δ-approximate bisimulation between transition systems (Clarke et al., 2000) was introduced. This approximate bisimulation, if established, identifies bounds on the distance between the trajectories of two distinct transition systems. The δ-approximate bisimulations between two transition systems can be achieved using classes of functions called *bisimulation functions*. In Girard and Pappas (2005), the guidelines for identifying parametric bisimulation functions are proposed and shown to work in some case studies. Since the autonomous continuous dynamical systems can be considered as a subclass of the class of transition systems (Brihaye, 2006), one can specialize the $\delta-$approximate bisimulation method for the purpose of finding bisimulation functions in the continuous dynamic network framework.

The classical notion of bisimulation, reported in Definition 4 (see Clarke et al., 2000) can be formally seen as an equivalence relation, inducing a partition of the states of the involved transition systems (Zhang, 1994).

Definition 4 A relation $\sim$ is a *bisimulation* between two transition systems $G_1 \equiv (Q_1, \Sigma, \underset{1}{\rightarrow})$ and $G_2 \equiv (Q_2, \Sigma, \underset{2}{\rightarrow})$ such that $\forall (q_1, q_2) \in Q_1 \times Q_2$, $(q_1 \sim q_2)$ if the following conditions are satisfied:

1. $\forall a \in \Sigma, \forall q_1' \in Q_1 \mid q_1 \overset{a}{\underset{1}{\rightarrow}} q_1' \implies \exists q_2' \in Q_2 \mid q_2 \overset{a}{\underset{2}{\rightarrow}} q_2'$ and $(q_1' \sim q_2')$
2. $\forall a \in \Sigma, \forall q_2' \in Q_2 \mid q_2 \overset{a}{\underset{2}{\rightarrow}} q_2' \implies \exists q_1' \in Q_1 \mid q_1 \overset{a}{\underset{1}{\rightarrow}} q_1'$ and $(q_1' \sim q_2')$.

Bisimulation from Definition 4 guarantees that two structures have the same behaviour. In particular, exact bisimulations between two transition systems entail that their observations are (and remain) identical. Consequently, exact bisimulations are very difficult to establish for non-linear dynamical systems like dynamic networks, and like CTRNNs in particular.

The approximate bisimulation approach presented in Definition 5 (see Girard and Pappas, 2005) is less rigid, as it requires the observations of approximately bisimilar systems to be (and remain) arbitrarily close to each other.

Definition 5 A relation $\sim_\delta$ is a *$\delta-$approximate bisimulation* between two continuous dynamic networks with observables $D_1 \equiv (Q_1, \gamma_1, T, h_1, H)$ and $D_2 \equiv (Q_2, \gamma_2, T, h_2, H)$ with a common time space T and observation space H, such that $(\forall (\mathbf{y}_1, \mathbf{y}_2) \in Q_1 \times Q_2$, $(\mathbf{y}_1 \sim_\delta \mathbf{y}_2)$ if the following conditions are satisfied:

1. $d(h_1(\mathbf{y}_1), h_2(\mathbf{y}_2)) \leq \delta$
2. $\forall t \in T, \forall \mathbf{y}_1' = \gamma_1(\mathbf{y}_1, t) \in Q_1 \implies \exists \mathbf{y}_2' = \gamma_2(\mathbf{y}_2, t) \in Q_2 \mid (\mathbf{y}_1' \sim_\delta \mathbf{y}_2')$
3. $\forall t \in T, \forall \mathbf{y}_2' = \gamma_2(\mathbf{y}_2, t) \in Q_2 \implies \exists \mathbf{y}_1' = \gamma_1(\mathbf{y}_1, t) \in Q_1 \mid (\mathbf{y}_1' \sim_\delta \mathbf{y}_2')$.

Note If $\delta = 0$, Definition 5 collapses into an exact bisimulation.

In other words, establishing a $\delta-$approximate bisimulation between two continuous dynamic networks guarantees that distances of the trajectories in the observation space are bounded by a value δ as stated by Theorem 3.

Theorem 3 (Girard and Pappas, 2007b) *If there exists a $\delta-$approximate bisimulation between the two continuous dynamical networks D_1 and D_2, then for all observable trajectories of D_1 there exists a trajectory of D_2 such that $\forall t \in T$, $d(h_1(\gamma_1(\mathbf{y}_1, t)), h_2(\gamma_2(\mathbf{y}_2, t)) \leq \delta$ and vice versa.*

The construction of approximate bisimulations between two transition systems, as well as the evaluation of their precision, can be performed using a class of functions called *bisimulation functions*, formally defined in Definition 6, which are positive functions defined on $Q_1 \times Q_2$, bounding the distance between the observations associated with a pair $(\mathbf{y}_1, \mathbf{y}_2)$, and non-increasing under the dynamics of the systems.

Definition 6 A *bisimulation function* V is a continuous function

$$V \,:\, Q_1 \times Q_2 \rightarrow \mathbb{R}^+$$

with

1. $V(\mathbf{y}_1, \mathbf{y}_2) \geq d(h_1(\mathbf{y}_1), h_2(\mathbf{y}_2))$
2. $V(\mathbf{y}_1, \mathbf{y}_2) \geq \max_{\mathbf{y}_1' = \gamma_1(\mathbf{y}_1, t)} \min_{\mathbf{y}_2' = \gamma_2(\mathbf{y}_2, t)} V(\mathbf{y}_1', \mathbf{y}_2')$
3. $V(\mathbf{y}_1, \mathbf{y}_2) \geq \max_{\mathbf{y}_2' = \gamma_2(\mathbf{y}_2, t)} \min_{\mathbf{y}_1' = \gamma_1(\mathbf{y}_1, t)} V(\mathbf{y}_1', \mathbf{y}_2')$.

Note that when restricting our study to the class of autonomous systems, Conditions 2 and 3 of Definition 6 become equivalent, and reduce to a Lyapunov-like condition. In particular, this can be considered as a weakening of Lyapunov stability conditions, in which one of the systems to compare collapses to the fixed-point solution. Similarly, the concept of bisimulation function is reminiscent of robust control Lyapunov functions (see Liberzon et al., 2002), though the latter require stronger conditions than bisimulation functions. Thanks to Theorem 4, the discovery of a bisimulation function is a sufficient condition for finding a bisimulation relation between two systems.

Theorem 4 (Girard and Pappas, 2007b) *If V is a bisimulation function, then $\forall \delta \geq 0$ the set*

$$B_\delta = \{(\mathbf{y}_1, \mathbf{y}_2) \in Q_1 \times Q_2, V(\mathbf{y}_1, \mathbf{y}_2) \leq \delta\}$$

is a δ-approximate bisimulation between dynamical networks D_1 and D_2.

In other words, Theorem 4 suggests the possibility of comparing observable trajectories of two continuous dynamic networks if one is able to describe a procedure to systematically compute suitable bisimulation functions for them.

Let us consider the case of two continuous dynamic networks D_i with observables, where $i \in \{1, 2\}$, with equations $\dot{\mathbf{y}}_i = \mathbf{f}_i(\mathbf{y}_i)$, where $\mathbf{y}_i \in \mathbb{R}^{n_i}$, and with observation maps $\mathbf{h}_i$, respectively. Let us further assume the hypotheses that D_1 and D_2 have the same observation space $\mathbb{R}^k$, equipped with the Euclidean distance.

Now, by letting $\mathbf{y} = [\mathbf{y}_1; \mathbf{y}_2]$ be an $(n_1 + n_2) \times 1$ column vector, $\mathbf{f}(\mathbf{y}) = \left[\ \mathbf{f}_1(\mathbf{y}_1); \mathbf{f}_2(\mathbf{y}_2)\ \right]$ an $(n_1 + n_2) \times 1$ column vector, and $\mathbf{h}(\mathbf{y}) = \mathbf{h}_1(\mathbf{y}_1) - \mathbf{h}_2(\mathbf{y}_2)$ a $p \times 1$ column vector, one can state the following important theorem (see Girard and Pappas, 2005):

Theorem 5 *Let $p : \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \rightarrow \mathbb{R}^+$ be a differentiable function with ∇p its gradient. If for all $\mathbf{y} \in \mathbb{R}^{n_1 + n_2}$ we have that $p(\mathbf{y})$ satisfies*

$$p(\mathbf{y}) \geq \mathbf{h}(\mathbf{y})^T \mathbf{h}(\mathbf{y}) \tag{4}$$

$$\nabla p(\mathbf{y})^T \mathbf{f}(\mathbf{y}) \leq 0 \tag{5}$$

then $V = \sqrt{p(\mathbf{y})}$ *is a bisimulation function.*

Thus, Theorem 5 ensures that finding a function $p(\mathbf{y})$ satisfying Conditions (4) and (5) lets us obtain a bisimulation function. Although this is a difficult task (Parrilo, 2003), there are techniques in semidefinite programming that allow one to find a sum of squares form for $p(\mathbf{y}) - \mathbf{h}(\mathbf{y})\mathbf{h}(\mathbf{y})$ and $-\nabla p(\mathbf{y})^T \mathbf{f}(\mathbf{y})$, resulting in a much simpler problem*. However, note that this approach restricts the possible solutions, insofar as a sum of squares condition implies a positive condition, but the converse is not true. We, indeed, reduce the space of the functions, in which solutions are searched, to a subspace in which this search is more tractable.

Accordingly, it is possible to write a simpler formulation of Theorem 5, if one assumes that the vector fields $\mathbf{f}_1(\mathbf{y})$ and $\mathbf{f}_2(\mathbf{y})$ are expressed by polynomials.

Theorem 6 (Girard and Pappas, 2005) *Assuming the hypotheses that the autonomous vector fields* $\mathbf{f}_1$ *and* $\mathbf{f}_2$ *and the observation maps* $\mathbf{h}_1$ *and* $\mathbf{h}_2$ *are vectors of polynomials, then the conditions*

$$p(\mathbf{y}) - \mathbf{h}(\mathbf{y})^T \mathbf{h}(\mathbf{y}) \quad \textit{is a sum of squares} \tag{6}$$

$$-\nabla p(\mathbf{y})^T \mathbf{f}(\mathbf{y}) \quad \textit{is a sum of squares} \tag{7}$$

imply that $V(\mathbf{y}) = \sqrt{p(\mathbf{y})}$ *is a bisimulation function, where* $p(\mathbf{y})$ *is a multivariate polynomial.*

With Theorem 6, the task of finding $p(\mathbf{y})$, satisfying Conditions (6) and (7), becomes manageable, and the bisimulation function can be computed in semidefinite programming†. Thus, one can rewrite the steps 3-4 of Algorithm 1 in the light of this theorem:

3a* choose a sum of squares form $p^{\mathbf{c}}(\mathbf{y})$
3b* find $\mathbf{c}'$ that minimizes $p^{\mathbf{c}}(\mathbf{y})$ satisfying Theorem 6
4* find $\delta_{max} = \max_{\mathbf{y}} \sqrt{p^{\mathbf{c}'}(\mathbf{y})}$.

In the next section, we apply this DyNeS procedure to CTRNN equations, and show that this measure enables one to capture the error it is possible to tolerate in order to consider CTRNN systems as functionally bisimilar systems.

*It has been shown (see Parrilo, 2003) that the condition "$p(\mathbf{x})$ is a sum of squares" is computationally more tractable than $p(\mathbf{x}) \geq 0$.

†In particular, the algorithm subsumed by Theorem 6, SOSTOOLS Matlab toolbox (Prajna et al., 2002) will be deployed.

3. CTRNN comparison by DyNeS measure

3.1. The outline

In this section, we prepare and test systems whose functional equivalence is going to be evaluated by means of the $\delta-$approximate bisimulation. First of all, we need a polynomial approximation of a CTRNN, due the fact that one cannot apply the bisumulation method explained in Subsection 2.5 directly to equations of CTRNNs whose flows are not polynomials. Following Subsection 2.4, this problem can be addressed by applying a machine learning procedure enabling one to approximate the CTRNN sigmoid function in a finite interval. In this way, in Subsection 3.2 we obtain a “polynomialized” version of the CTRNN system, which approximates the original one. Then, in Subsection 3.3 we prepared systems of one neuron networks, for which it is possible to theoretically characterize the space of the solutions when varying their parameters, in order to illustrate δ-approximate bisimulation search. Finally, we test the scalability procedure on two and three neuron networks.

3.2. A polynomial approximation for CTRNN flows

The possibility of comparing two continuous dynamical networks by using approximate bisimulation was illustrated in Section 2.5. However, the application of Theorem 6 affords a comparison procedure only in the presence of polynomial flows of the system. Thus, the sigmoid output function prevents one from directly searching a bisimulation between CTRNN systems. On the other hand, a polynomial approximation of the sigmoidal function, ”as good as we want”, can be computed (see Subsection 2.4). More specifically, in this section we use regularized least squares (see Bishop, 2006) in order to obtain a polynomial approximation for the sigmoid function

$$\sigma(x) \approx \sum_{m=0}^{M} c_i \cdot x^m = Pol_M(x)$$

which, in turn, enables one to obtain a polynomialized version of the CTRNN Equations (2):

$$\tau^i \frac{dy^i}{dt} = -y^i + \sum_{j=1}^{N} w^{ij} Pol_M(y^j - \theta^j) + I_e^i \qquad i \in \{1, \dots, N\}. \tag{8}$$

This procedure, given a suitable polynomial order M, allows one to approximate, to any desired precision, the behaviour of the sigmoid function in some given interval. In order to apply the regression algorithm, input-output pairs of the sigmoid function $\sigma(x)$ in a fixed interval $[x_{min}, x_{max}]$ are prepared. The coefficients found for a Pol_M are searched in an interval $[-30, 30]$. Fig. 1 shows

M	1	2	3	4	5	6	7	8
$\bar{E}$	0.1845	0.1845	0.1160	0.1160	0.0831	0.0831	0.0630	0.0630
dev	0.0133	0.0133	0.0077	0.0077	0.0046	0.0046	0.0029	0.0029

Table 1: Mean error $\bar{E}$ and standard deviation dev for Pol_M.

h

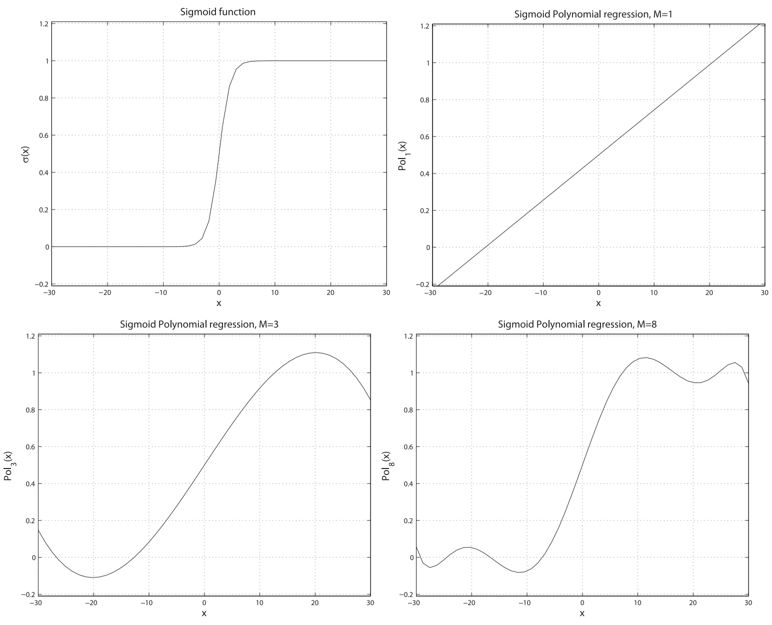


Figure 1: Sigmoid function $\sigma(x)$ versus the polynomial approximations $Pol_1(x)$, $Pol_3(x)$ and $Pol_8(x)$

the behaviour of this approximation for $M = 1$, $M = 3$, $M = 8$ compared to the sigmoid function $\sigma(x)$. Table 1 shows that the mean errors $\bar{E}$ corresponding to different Pol_M decrease with the order of the polynomial. In the application of the bisimilarity procedure we choose $M = 8$ and the corresponding $Pol_8(x)$. The two systems are comparable as long as each neuron activation value y_i of the CTRNNs is in $[x_{min}, x_{max}]$. However, if the solution of the systems lets the activations range over values outside this interval, then it is possible to regress a new $Pol_M(x)$, which approximates $\sigma(x)$ within a wider range.

3.3. Methods explained for simple CTRNNs

In this section we build different examples in order to show how to compare two CTRNNs by means of the DyNeS measure. In the first set of tests, we consider

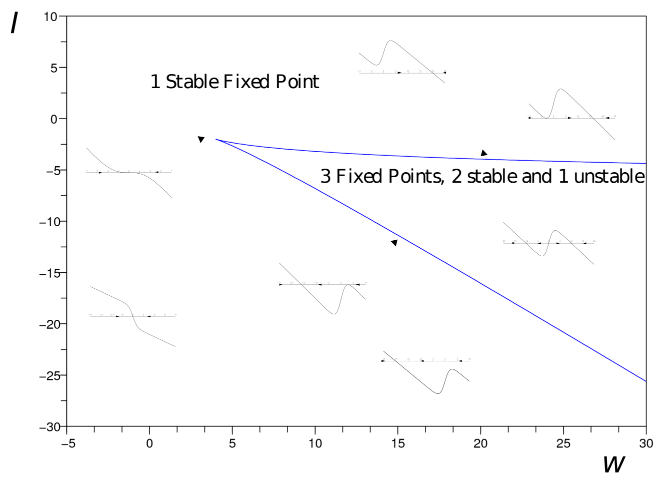


Figure 2: The two branches of the cusp of Equation (9) (where θ is set to 0), in the parameter space (w, I). Outside two cusp branches, the system has a global stable equilibrium point. Inside the branches there are three equilibria: an unstable one and two stable ones

simple systems made up of a single neuron:

$$\dot{y} = -y + w \cdot \sigma(y + \theta) + I \tag{9}$$

where, for simplicity, we set the time constant $\tau = 1$. Notice that no elementary expression for the solution of (9) exists. By contrast, it is possible to achieve a complete qualitative description of its dynamics (Beer, 1995). Specifically, one can describe the limit sets of (9), including their stability and their dependence on the parameters, as well as the bifurcations that can occur as the parameters are varied (see Fig. 2). Such system has a cusp point (Hale and Koçac, 1991). This cusp point $(\tilde{w}, \tilde{I})$ is the only bifurcation point in which the system undergoes a pitchfork bifurcation (Hale and Koçac, 1991). All other bifurcation points are saddle-node bifurcations.

Thus, let us consider two one-neuron systems, $\tilde{D}_1$ and $\tilde{D}_2$:

$$\tilde{D}_1 \equiv \left\{ \begin{array}{rcl} \dot{y}_1 & = & -y_1 + w_1 \cdot \sigma(y_1) + I \end{array} \right.$$

and

$$\tilde{D}_2 \equiv \left\{ \begin{array}{rcl} \dot{y}_2 & = & -y_1 + w_2 \cdot \sigma(y_2) + I \end{array} \right. .$$

In order to apply Theorem 6, the flows of the system have to be vectors of

w_2	c_0	c_1	c_2	c_3	c_4	c_5
-16	0.04	-0.27	0.26	-5.56	2.83	2.74
-17	0.08	-0.24	0.25	-5.23	2.69	2.55
-18	0.30	-0.90	0.91	-7.07	3.73	3.36
-19	0.50	-1.15	1.17	-7.18	3.86	3.35
-20	0.76	-1.38	1.41	-6.82	3.71	3.164

Table 2: The parameters of the bisimulation function $V(y_1, y_2, \{c_i\}_{i=0}^5)$ between D_1 and five versions of system D_2 with $w_2 \in \{-16, -17, -18, -19, -20\}$. We consider the case, in which the initial conditions of the observable variables are identical ($y_1^0 = y_2^0$) and are in the closed and bounded interval $[-1, 1]$

polynomials. Thus, we "polynomialize" the CTRNN equations as shown in the previous section, obtaining the new systems

$$D_1 \equiv \left\{ \begin{array}{lcl} \dot{y}_1 & = & -y_1 + w_1 \cdot Pol_M(y_1) + I \\ h_1 & = & y_1 \end{array} \right.$$

and

$$D_2 \equiv \left\{ \begin{array}{lcl} \dot{y}_2 & = & -y_1 + w_2 \cdot Pol_M(y_2) + I \\ h_2 & = & y_2 \end{array} \right. .$$

We choose the identity functions as the observation maps: this means that we are going to directly compare activations y_1 and y_2 of the two systems. In this case the term $\mathbf{h}(\mathbf{y})^T\mathbf{h}(\mathbf{y})$ in Condition 6 turns into $(y_1 - y_2)^2$. This suggests that the polynomial form of the bisimulation function must be at least of the second order. We choose quite a general form, including terms up to the second order:

$$V(y_1, y_2, \{c_i\}_{i=0}^5) = \sqrt{p(y_1, y_2)} =$$
$$\sqrt{c_0 + c_1 \cdot y_1 + c_2 \cdot y_2 + c_3 y_1 \cdot y_2 + c_4 \cdot (y_1)^2 + c_5 \cdot (y_2)^2}$$

Let us firstly consider the case, in which there is a global stable equilibrium point in the equation, so that, for any of the initial conditions of the variables y_1 and y_2, the asymptotic solution is unique. Consequently, we set $I = 0$ for both D_1 and D_2, the weight $w_1 = -15$ for system D_1, and for D_2 we choose w_2 belonging to the set $\{-16, -17, -18, -19, -20\}$, thus obtaining five different versions of D_2, one for each w_2 value. We compare the D_1 system so obtained with the five different versions of system D_2. Thus, minimizing the condition of $V(y_1, y_2, \{c_i\}_{i=0}^5)$ under which the initial conditions of the observable variables are identical ($y_1^0 = y_2^0$), and in the closed interval $[-1, 1]$, we obtain the parameters c_i of the bisimulation function for each system D_2.

Table 2 shows the values of the identified parameters, while in Table 3 the corresponding values of the similarity measure are shown. Note that, as we

w_2	-16	-17	-18	-19	-20
$\delta_{\max}$	0.20	0.32	0.58	0.75	0.92

Table 3: $\delta_{\max}$ obtained for the bisimulation function with parameter values from Table 2. Increasing values of $\delta_{\max}$ are found, as long as system D_2 behaves "more differently" with respect to D_1

w_2	c_0	c_1	c_2	c_3	c_4	c_5
-16	0.29	-0.66	0.75	-2.85	1.37	1.49
-17	0.50	-1.14	1.28	-3.48	1.66	1.83
-18	0.38	-1.21	1.25	-4.50	2.25	2.25
-19	0.59	-1.42	1.48	-4.50	2.28	2.23
-20	0.86	-1.63	1.72	-4.48	2.29	2.21

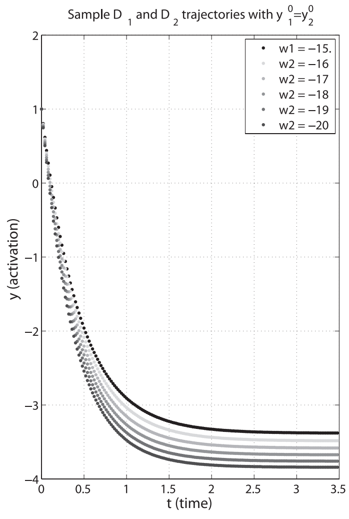
Table 4: The parameters of the bisimulation function $V(y_1, y_2, \{c_i\}_{i=0}^5)$ between D_1 and five versions of system D_2 with $w_2 \in \{-16, -17, -18, -19, -20\}$. Here, the case, in which the initial conditions of the observable variables are different ($y_1^0 \neq y_2^0$) and are in the closed and bounded interval $[-1, 1]$ is considered

expected, increasing values of $\delta_{\max}$ are found, as long as system D_2 behaves "more differently" with respect to D_1.

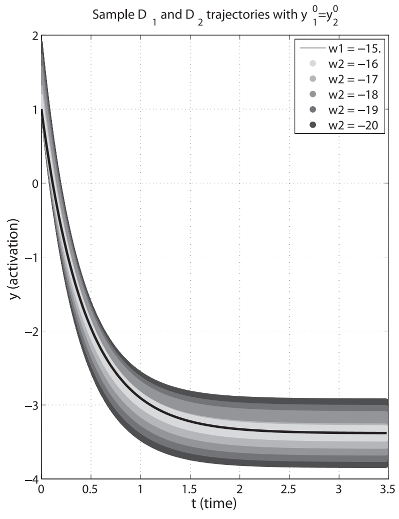
In Fig. 3 numerical simulations of the original system $\tilde{D}_1$ versus the different versions of system $\tilde{D}_2$ are shown. As it was expected, the errors on trajectories shown are completely consistent with the $\delta_{\max}$ error in Table 3. In fact, Fig. 3a shows sample trajectories of solutions of the compared systems, Fig. 3b shows that solutions of D_2 depicted with their $\delta_{\max}$-area include solutions of D_1. Furthermore, we simulated 500 different solutions for each D_2 system, with different initial conditions fulfilling ($y_1^0 = y_2^0$). Thus, at each time we computed an experimental $\delta_{exp}(t) = \max \left\{ \left| y_1(t) - y_2^{(i)}(t) \right| \right\}_{k=1}^{500}$. In Fig. 3c the ratio $(\delta_{\max} - \delta_{exp}(t))/\delta_{\max}$ is plotted, resulting in values greater than zero, meaning that the $\delta_{\max}$ bound is never crossed.

A second test is made on the same D_2 system, but now consisting in the search for an initial condition of the observable variables $y_1^0 \neq y_2^0$ in the same interval $[-1, 1]$. In this case, the minimization is run in order to find parameters that a posteriori minimize the bisimulation function $V(y_1, y_2, \{c_i\}_{i=1}^5)$ for any possible initial conditions in $[-1, 1]$. The parameters for the bisimulation function are shown in Table 4.

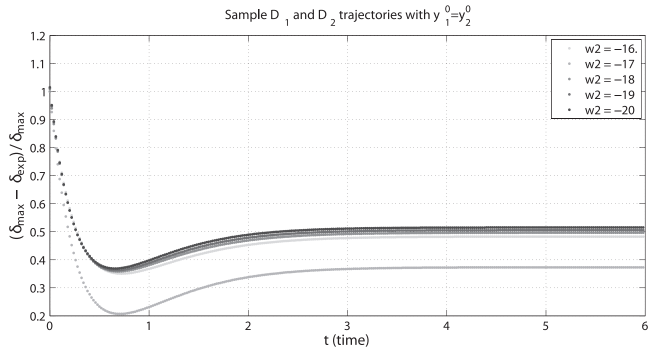
In Table 5, $\delta_{\max}$ values obtained for the bisimulation function with parameters from Table 4 are shown. Again, increasing values of $\delta_{\max}$ are found, as long as system D_2 behaves "more differently" with respect to D_1. The values of the upper bound given by $\delta_{\max}$ are bigger than the ones in Table 3. Note that this is again expected, insofar as we considered more general initial conditions for y_1 and y_2.



(a)

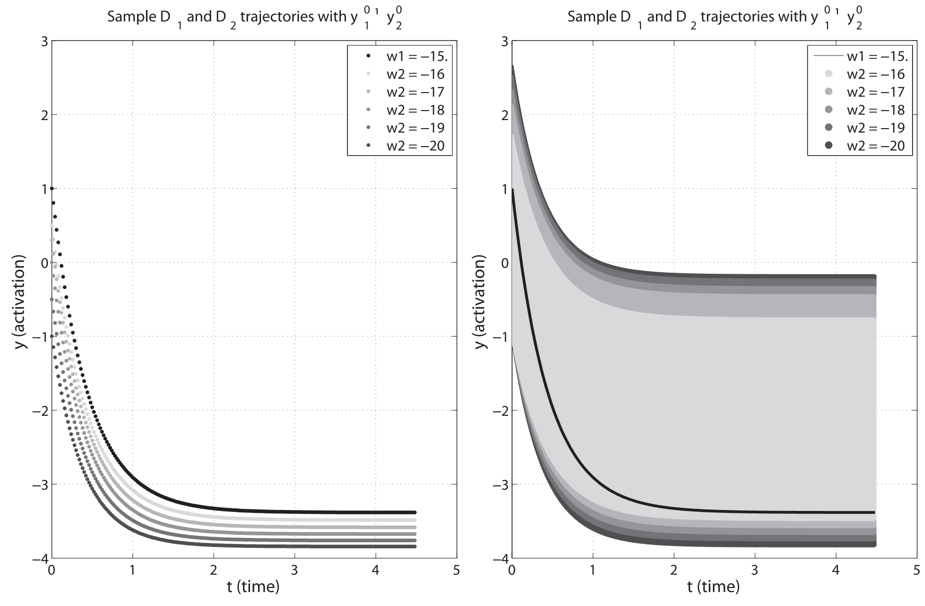


(b)

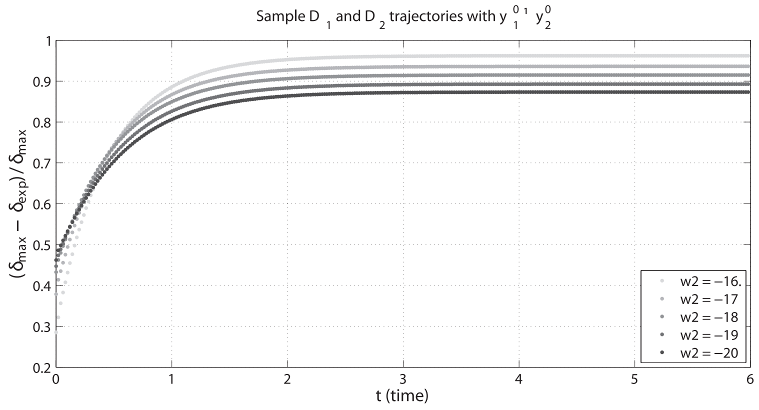


(c)

Figure 3: In diagram (a) some numerical simulations of system $\tilde{D}_1$ versus different versions of the system $\tilde{D}_2$ are shown. $\tilde{D}_1$ and $\tilde{D}_2$ are one-neuron CTRNNs. In diagram (b), each solution of $\tilde{D}_2$ is depicted as a $\delta_{\max}$-coloured area with values from Table 3. As an example, the area of $\tilde{D}_2$ with $w_2 = -20$ includes all areas with $w_2 > -20$. All the areas of $\tilde{D}_2$ include the solution of $\tilde{D}_1$. That means that the shown distance between trajectories of $\tilde{D}_1$ and $\tilde{D}_2$ is completely contained, as it was expected, in the $\delta_{\max}$ error. In diagram (c), the ratio $(\delta_{\max} - \delta_{exp})/\delta_{\max}$ is shown: again, as expected, values greater than zero mean that the bound $\delta_{\max}$ is never overcome. See text for more details



(a) (b)



(c)

Figure 4: In diagram (a), numerical simulations of system $\tilde{D}_1$ versus the different versions of system $\tilde{D}_2$ were tested. $\tilde{D}_1$ and $\tilde{D}_2$ are one-neuron CTRNNs. In diagram (b), areas of δ-solutions of $\tilde{D}_2$ are shown (see Fig. 3). Note that in this example areas are larger in order to bound all the different initial conditions we subsumed in the computation. In diagram (c), values $(\delta_{\max} - \delta exp)/\delta_{\max}$ are plotted: as a result, the errors on trajectories shown are completely contained, as it was expected, in the $\delta_{\max}$ error from Table 5

w_2	-16	-17	-18	-19	-20
$\delta_{\max}$	2.73	3.14	3.44	3.53	3.63

Table 5: $\delta_{\max}$ obtained for the bisimulation function with parameters from Table 4. Again, increasing values of $\delta_{\max}$ are found, as long as system D_2 behaves "more differently" with respect to D_1. The values of the upper bound given by $\delta_{\max}$ are bigger than the ones in Table 3. This is again expected, insofar as we considered more general initial conditions for y_1 and y_2

D_1 and D_2	c_0	c_1	c_2	c_3	c_4	c_5	$\delta_{\max}$
$I = -2.5$, $w_1 = 6$, $w_2 = 7$	42.30	119.89	-85.61	-131.06	93.63	46.27	22.76
$I = -5$, $w_1 = 10$, $w_2 = 11$	200.23	-377.25	-82.88	92.49	278.14	10.25	32.26

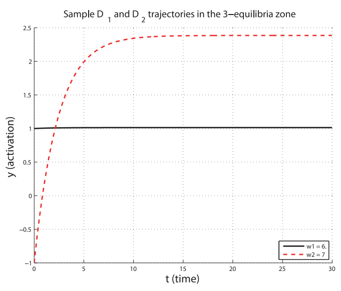
Table 6: Parameters for the bisimulation function and the relative $\delta_{\max}$ between systems D_1 and D_2. Two cases are presented: in the first we set $I = -2.5$, $w_1 = 6$, $w_2 = 7$, and in the second $I = -5$, $w_1 = 10$, $w_2 = 11$. In the two examples the initial conditions are different, $y_1^0 \neq y_2^0$, and considered in the interval $[-1, 1]$. In this case $\delta_{\max}$ measures the variability of the observable trajectories (here being higher with respect to the previous example), and increases consistently with the greater numerical values of the distance between the stable equilibrium points

In Fig. 4, numerical simulations of the original system $\tilde{D}_1$ versus the different version of system $\tilde{D}_2$ are shown. Coherently, even with different initial conditions, the errors on trajectories shown are completely consistent with the $\delta_{\max}$ error in Table 5.

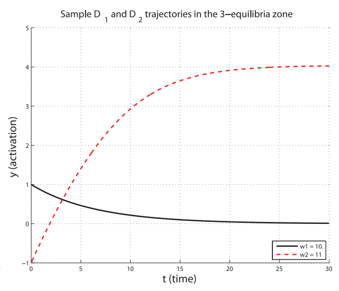
Now, we consider the case of parameters (I, w) (see Fig. 2), in which there are three equilibrium points. In this case, by considering different initial conditions one may end up with systems that are more different with respect to the one in the global stable equilibrium point zone that we have considered until now. We run the search for parameters of the bisimulation function in two illustrative cases: in the first one we set $I = -2.5$, $w_1 = 6$, $w_2 = 7$, and in the second $I = -5$, $w_1 = 10$, $w_2 = 11$. In the two examples the initial conditions are different, $y_1^0 \neq y_2^0$, and considered in the interval $[-1, 1]$.

Table 6 shows the results for this bisimulation function search and the relative $\delta_{\max}$. In this case $\delta_{\max}$ measures the variability of the observable trajectories (which is higher with respect to the previous example), and increases consistently with the greater numerical values of the distance between the stable equilibrium points. The numerical simulations, shown in Fig. 5, are coherent with the $\delta_{\max}$ bound found.

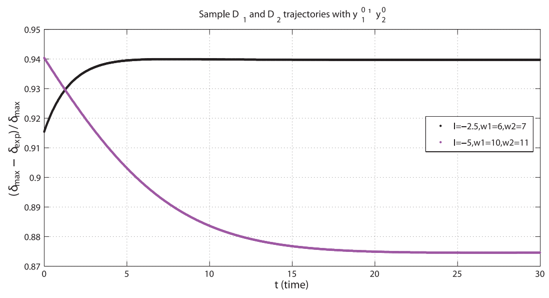
Finally, in the last session of tests, we explore the scalability of the proposed approach on systems of multiple neurons. We prepared two system comparisons



(a)



(b)



(c)

Figure 5: In diagrams (a) and (b), numerical simulations of system $\tilde{D}_1$ versus the different versions of system $\tilde{D}_2$ in the 3−equilibria zone are shown. In diagram (c), $(\delta_{\max} - \delta_{exp})/\delta_{\max}$ is shown: the errors on trajectories shown are completely contained, as it was expected, in the $\delta_{\max}$ error from Table 6

- a comparison between networks with two neurons each, D_1^{two} and D_2^{two}
- a comparison between networks with three neurons each, D_1^{three} and D_2^{three}.

In both cases the networks are fully connected. On the second system we put a fixed input signal I that acts as a perturbation. For example, in the case of systems D_1^{two} and D_2^{two} we consider

$$D_1^{two} \equiv \begin{cases} \dot{y}_1^1 & = & -y_1^1 + w^{11} \cdot Pol_M(y_1^1) + w^{12} \cdot Pol_M(y_1^2) \\ \dot{y}_1^2 & = & -y_1^2 + w^{12} \cdot Pol_M(y_1^1) + w^{22} \cdot Pol_M(y_1^2) \\ \mathbf{h}_1(\mathbf{y}_1) & = & \mathbf{y}_1 \end{cases}$$

and

$$D_2^{two} \equiv \begin{cases} \dot{y}_2^1 & = & -y_2^1 + w^{11} \cdot Pol_M(y_2^1) + w^{12} \cdot Pol_M(y_2^2) + I \\ \dot{y}_2^2 & = & -y_2^2 + w^{12} \cdot Pol_M(y_2^1) + w^{22} \cdot Pol_M(y_2^2) + I \\ \mathbf{h}_2(\mathbf{y}_2) & = & \mathbf{y}_2 \end{cases} .$$

To run the experiments, we consider a quadratic form for the bisimulation func-

D_1^{two} vs D_2^{two}	δ_{max}	σ
$I = 10$	14.7	1.9
$I = 1$	1.9	0.3
$I = 0.1$	0.19	0.16

D_1^{three} vs D_2^{three}	$\bar{\delta}_{max}$	σ
$I = 10$	36	3
$I = 1$	20	9
$I = 0.1$	13	8

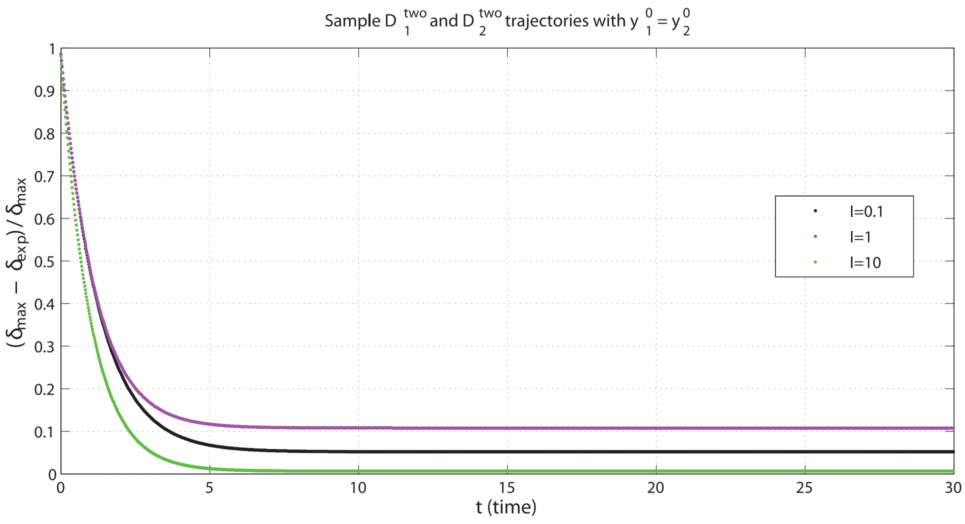
Table 7: Mean value for $\delta_{\max}$ and standard deviation computed on the set of random weights for bisimulation between D_1^{two} vs D_2^{two} and D_1^{three} vs D_2^{three}. The systems to be compared are identical except for the perturbation with a fixed input signal $I \in \{0.1, 1, 10\}$

tion in all the variables, as we did previously. Furthermore, to simplify the computations we set equal initial conditions for all the observation variables $\{\mathbf{y}_1, \mathbf{y}_2\}$ and in the interval $[0, 1]$. To calculate each $\delta_{\max}$, we randomly choose a set of weights in $[0, 1]$ and perturb the second system with a fixed input $I \in \{0.1, 1, 10\}$. We repeat this computation for 30 times and compute the mean value for $\delta_{\max}$ and its standard deviation. From the results in Table 7 and in Fig. 6, it is possible to appreciate the coherence of the measure with respect to the perturbation of the input signals and the increase in the number of neurons.

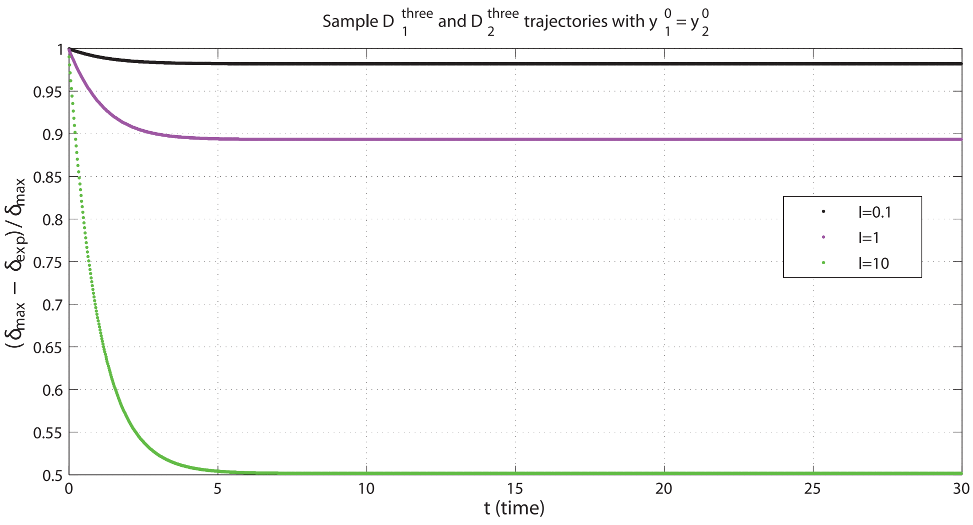
4. Conclusions

4.1. The results

We have presented the dynamic network similarity (DyNeS) Algorithm, i.e., a procedure enabling one to compare the behaviour of neural networks, by introducing for the first time the bisimulation method to the non-linear dynamic network framework. To do this, we have imported theorems from dynamical system theory and formal methods and adapted them to dynamic networks. We, then, have selected a particular model of dynamic networks, namely continuous time recurrent neural networks (CTRNN), and showed the applicability of our algorithm to them. A first step in the proposed procedure is the approximation of the dynamic network with a polynomial version with a behaviour as close as possible to the "original" network. To do this, a machine learning procedure is applied. Consequently, we have shown how to find the upper bounds on the distances between the trajectories of two distinct CTRNNs by computing approximate bisimulations on their "polynomialized" versions (Tables 3, 5, 6 and 7). Then, we have confirmed the upper bound validity by numerically integrating the chosen CTRNNs. The results shown in Figs. 3, 4, and 5 confirm



(a)



(b)

Figure 6: In diagrams (a) and (b), the ratio $(\delta_{\max} - \delta_{exp})/\delta_{\max}$ is shown for system $\tilde{D}_1^{two}$ versus system $\tilde{D}_2^{two}$ and for system $\tilde{D}_1^{three}$ versus system $\tilde{D}_2^{three}$, respectively, with different inputs $I \in \{0.1, 1, 10\}$. Again, as expected, values greater than zero mean that the bound $\delta_{\max}$ is never overcome

that the distances between the trajectories are inferior or equal to the predicted theoretical upper bounds. Notice that the DyNeS Algorithm can be performed on a wider class of networks with respect to the one we have selected in our experiments. For example, as a consequence of the studies presented in Haschke (2004); Beer (2006), all results obtained in this paper for the CTRNN models with neuron output function $\sigma(x)$ can be easily transferred to any CTRNN with neuron output function $\sigma_{a,b,c}$ shown in Section 2, since for each CTRNN with neuron output function $\sigma_{a,b,c}$ it is possible to construct an equivalent CTRNN with neuron output function $\sigma(x)$.

4.2. Discussion and future works

Although these preliminary results suggest that the application of the DyNeS Algorithm might be a viable effective approach for comparing dynamic networks, several issues arise, pointing at different directions, underlying the need of achieving further progress on them. Firstly, the more the polynomial approximation is improved leading to indistinguishable behaviour between approximate and "original" networks, the more complex the needed polynomial is, and the computational cost raises, especially when dealing with the argument of the sigmoidal function spreading in large intervals. However, one can also directly use "polynomialized" versions of CTRNNs as models of neural circuits *per se*, just as other dynamic neural models have been used in a "polynomialized" version (see, e.g., Wilson, 1999).

Another research line concerns finding a subclass of dynamic networks for which an effective ϵ error of the polynomial approximation can be combined with the δ of the bisimulation, thus constructing a unique theoretical upper bound for the entire procedure. Furthermore, in case the search for the bisimulation fails, we cannot establish a measure of similarity, but at the same time we cannot assert that a bisimulation does not exist, because of 1) the arbitrariness of the construction of the Lyapunov-like bisimulation function, and 2) the fact that we are trying to satisfy the sum of squares condition in Theorem 6, which implies the positive condition of Theorem 5, while the converse is not true. Thus, a critical aspect of the DyNeS Algorithm is related to the search for the appropriate bisimulation function, together with the computation of the coefficients of the polynomial. In order to simplify the search and the construction of the proper Lyapunov-like function, effective matrix measure approaches could be imported from the field of synchronisation of chaotic neural networks (He and Cao, 2009; Cao and Wan, 2014; Chandrasekar et al., 2014). Interestingly, the network similarity problem specified in Section 2 can be expressed in a very general way for different models of static and dynamic neural networks. In particular, the presented DyNeS Algorithm is designed to work with CTRNNs, but, in principle, could also be extended in order to encompass more general dynamic neural networks, thus establishing a similarity measure able to compare dynamic behaviours of different type of dynamic neural networks models.

Finally, we put further emphasis on how this kind of approach paves the way to interesting possibilities for neuroscientists: for example, the introduction of formal logics “talking about” trajectories (see, e.g., Fainekos et al., 2007) could enable one to ask more complex questions about dynamic network behaviours. Thus, also for those aspects, although in all the experiments described in this paper the similarity could be successfully accomplished by the computation of the coefficients of the proper bisimulation function, there are several issues for which additional theoretical and experimental investigations are needed.

Acknowledgments

The present research is funded by the Human Frontier Science Program (HFSP), award number RGY0088/2014, by the EU’s FP7 under grant agreement no FP7-ICT-270108 (Goal-Leaders), and by the ESF GAMES project, short visit grant n. 3014. The GEFORCE Titan used for this research was donated by the NVIDIA Corporation. The authors would also like to thank Patricia Bouyer, Giuseppe Trautteur, Guglielmo Tamburrini and Giovanni Pezzulo for the inspiring discussions, and the anonymous reviewers for their valuable comments and suggestions that greatly contributed to improving the final version of the paper.

References

Albertini, F. and Sontag, E.D. (1993) For neural networks, function determines form. *Neural Networks*, **6** (7), 975–990.

Albertini, F., Sontag, E.D. and Maillot, V. (1993) Uniqueness of weights for neural networks. In: *Artificial Neural Networks with Applications in Speech and Vision*. Chapman and Hall, 115–125.

Beer, R.D. (1995) On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, **3** (4), 469–509.

Beer, R.D. (2006) Parameter space structure of continuous-time recurrent neural networks. *Neural Computation*, **18** (12), 3009–3051.

Birch, M.C., Quinn, R.D., Hahm, G., Phillips, S.M., Drennan, B.T., Fife, A.J., Beer, R.D., Yu, X., Garverick, S.L., Laksanacharoen, S., Pollack, A.J. and Ritzmann, R.E. (2002) Cricket-based robots. *IEEE Robotics and Automation Magazine*, **9** (4), 20–30.

Bishop, C.M. (2006) *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Brihaye, T. (2006) *Verification and Control of O-minimal hybrid systems and weighted timed automata*. Ph.D. thesis, Université de Mons-Hainaut.

Cao, J., Huang, D.S. and Qu, Y. (2005) Global robust stability of delayed recurrent neural networks. *Chaos, Solitons and Fractals*, **23** (1), 221 – 229.

Cao, J. and Wan, Y. (2014) Matrix measure strategies for stability and synchronization of inertial bam neural network with time delays. *Neural Networks*, **53**, 165–172.

Casagrande, A., Dreossi, T. and Piazza, C. (2012) Hybrid automata and ϵ-analysis on a neural oscillator. In: *Proceedings of the First International Workshop on Hybrid Systems and Biology.* Newcastle Upon Tyne, UK, EPTCS, 58–72.

Casagrande, A., Piazza, C. and Policriti, A. (2009) Discrete semantics for hybrid automata. *Discrete Event Dynamic Systems*, **19** (4), 471–493.

Chandrasekar, A., Rakkiyappan, R., Cao, J. and Lakshmanan, S. (2014) Synchronization of memristor-based recurrent neural networks with two delay components based on second-order reciprocally convex approach. *Neural Networks*, **57**, 79–93.

Chersi, F., Donnarumma, F. and Pezzulo, G. (2013) Mental imagery in the navigation domain: a computational model of sensory-motor simulation mechanisms. *Adaptive Behavior*, **21** (4), 251–262.

Clarke, E.M., Grumberg, O. and Peled, D.A. (2000) *Model Checking.* The MIT Press.

De Falco, I., Della Cioppa, A., Donnarumma, F., Maisto, D., Prevete, R. and Tarantino, E. (2008) CTRNN parameter learning using Differential Evolution. In: M. Ghallab, C.D. Spyropoulos, N. Fakotakis and N. Avouris, eds., *18th European Conference on Artificial Intelligence* of *Frontiers in Artificial Intelligence and Applications. Frontiers in Artificial Intelligence and Applications*, **178**. IOS Press, Amsterdam, 783–784.

De Jong, H., Gouzé, J.L., Hernandez, C., Page, M., Sari, T. and Geiselmann, J. (2004) Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin of Mathematical Biology*, **66** (2), 301–340.

DiMattina, C. and Zhang, K. (2010) How to modify a neural network gradually without changing its input-output functionality. *Neural Computation*, **22** (1), 1–47.

Donnarumma, F., Prevete, R. and Trautteur, G. (2010) How and over what timescales does neural reuse actually occur? Commentary on "neural re-use as a fundamental organizational principle of the brain", by Michael L. Anderson. *Behavioral and Brain Sciences*, **33** (04), 272–273.

Donnarumma, F., Prevete, R. and Trautteur, G. (2012) Programming in the brain: a neural network theoretical framework. *Connection Science*, **24** (2-3), 71–90.

Dunn, N.A., Lockery, S.R., Pierce-Shimomura, J.T. and Conery, J.S. (2004) A neural network model of chemotaxis predicts functions of synaptic connections in the nematode caenorhabditis elegans. *Journal of Computational Neuroscience*, **17** (2), 137–147.

Fainekos, G.E., Girard, A. and Pappas, G.J. (2007) Hierarchical synthesis of hybrid controllers from temporal logic specifications. In: *Hybrid systems: computation and control.* Springer, 203–216.

Fefferman, C. and Markel, S. (1994) Recovering a feed-forward net from its output. In: *Advances in Neural Information Processing Systems* (*NIPS*). Morgan Kaufmann, 335–342.

Fränzle, M. (1999) Analysis of hybrid systems: An ounce of realism can save an infinity of states. In: *Computer Science Logic*. Springer, 126–139.

Funahashi, K. and Nakamura, Y. (1993) Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, **6** (6), 801–806.

Gardner, M. and Dorling, S. (1998) Artificial neural networks (the multilayer perception) - a review of applications in the atmospheric sciences. *Atmospheric Environment*, **32** (14-15), 2627–2636.

Ghosh, R. and Tomlin, C.J. (2001) Lateral inhibition through delta-notch signaling: A piecewise affine hybrid model. In: *Hybrid Systems: Computation and Control*. Springer, 232–246.

Girard, A. and Pappas, G. (2007a) Approximation metrics for discrete and continuous systems. *Automatic Control, IEEE Transactions on*, **52** (5), 782–798.

Girard, A. and Pappas, G.J. (2005) Approximate bisimulations for nonlinear dynamical systems. In: *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005.*. IEEE Press, 684–689.

Girard, A. and Pappas, G.J. (2007b) Approximation metrics for discrete and continuous systems. *Automatic Control, IEEE Transactions on*, **52** (5), 782–798.

Gupta, M.M., Homma, N. and Jin, L. (2003) *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory.* John Wiley & Sons, Inc., New York, NY, USA.

Hale, J.K. and Koçac, H. (1991) *Dynamics and Bifurcations.* Springer-Verlag.

Haschke, R. (2004) *Input space bifurcation manifolds of recurrent neural networks.* Ph.D. thesis, Bielefeld University, Neuroinformatics Group, Faculty of Technology, Bielefeld, Germany.

He, W. and Cao, J. (2009) Exponential synchronization of chaotic neural networks: a matrix measure approach. *Nonlinear Dynamics*, **55** (1-2), 55–65.

Hopfield, J.J. and Tank, D.W. (1986) Computing with neural circuits: A model. *Science*, **233**, 625–633.

Horne, B.G. and Giles, C.L. (1995) An experimental comparison of recurrent neural networks. In: G. Tesauro, D. Touretzky and T. Leen, eds., *Advances in Neural Information Processing Systems*, **7**. MIT Press, 697–704.

Izhikevich, E.M. (2007) *Dynamical Systems in Neuroscience: the Geometry of Excitability and Bursting. Computational Neuroscience.* MIT Press, Cambridge, Mass., London.

Lall, S., Marsden, J.E. and Glavascaronki, S. (2002) A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal of Robust and Nonlinear Control*, **12** (6), 519–535.

Liberzon, D., Sontag, E.D. and Wang, Y. (2002) Universal construction of feedback laws achieving iss and integral-iss disturbance attenuation. *Systems*

& Control Letters, **46** (2), 111–127.

MacGregor, R. (2012) *Neural and Brain Modeling.* Elsevier.

MacGregor, R.J. and Tajchman, G. (1988) Theory of dynamic similarity in neuronal systems. *Journal of Neurophysiology*, **60** (2), 751–768.

Marder, E. and Goaillard, J.M. (2006) Variability, compensation and homeostasis in neuron and network function. *Nature Reviews Neuroscience*, **7** (7), 563–574.

Montone, G., Donnarumma, F. and Prevete, R. (2011) A robotic scenario for programmable fixed-weight neural networks exhibiting multiple behaviors. In: A. Dobnikar, U. Lotric and B. Šter, eds., *Adaptive and Natural Computing Algorithms*, *Lecture Notes in Computer Science*, **6593**. Springer Berlin / Heidelberg, 250–259.

Munakata, T. (1997) *Fundamentals of the New Artificial Intelligence: Beyond Traditional Paradigms.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Neruda, R. (2000) Genetic algorithms and neural networks: Making use of parameter space symmetries. In: *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE - INNS - ENNS International Joint Conference on*, **1**, 293–298.

Paine, R.W. and Tani, J. (2004) Motor primitive and sequence self-organization in a hierarchical recurrent neural network. *Neural Networks*, **17** (8-9), 1291–1309. New Developments in Self-Organizing Systems.

Parrilo, P.A. (2003) *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* Ph.D. thesis, California Institute of Technology, Pasadena, CA.

Pezzulo, G., Donnarumma, F. and Dindo, H. (2013) Human sensorimotor communication: A theory of signaling in online social interactions. *PLoS ONE*, **8** (11), e79876.

Prabhakar, P., Vladimerou, V., Viswanathan, M. and Dullerud, G.E. (2009) Verifying tolerant systems using polynomial approximations. In: *Proceedings of the 30th IEEE Real-Time Systems Symposium.* IEEE Press, 181–190.

Prajna, S., Papachristodoulou, A. and Parrilo, P.A. (2002) Introducing sostools: a general purpose sum of squares programming solver. In: *Decision and Control 2002, Proceedings of the 41st IEEE Conference on*, **1**. IEEE Press, 741–746.

Prinz, A.A., Bucher, D. and Marder, E. (2004) Similar network activity from disparate circuit parameters. *Nature Neuroscience*, **7** (12), 1345–1352.

Ratschan, S. (2010) Safety verification of non-linear hybrid systems is quasi-semidecidable. In: *Theory and Applications of Models of Computation.* Springer, 397–408.

Rolls, E.T., Stringer, S.M. and Elliot, T. (2006) Entorhinal cortex grid cells can map to hippocampal place cells by competitive learning. *Network: Computation in Neural Systems*, **17** (4), 447–465.

Sporns, O. (2011) The human connectome: a complex network. *Annals of the New York Academy of Sciences*, **1224** (1), 109–125.

Tino, P., Horne, B.G. and Giles, C.L. (2001) Attractive periodic sets in discrete time recurrent networks (with emphasis on fixed point stability and bifurcations in two-neuron networks). *Neural Computation*, **13**, 1379–1414.

Wei, H. and Amari, S.i. (2008) Dynamics of learning near singularities in radial basis function networks. *Neural Networks*, **21** (7), 989–1005.

Wilson, H.R. (1999) Simplified dynamics of human and mammalian neocortical neurons. *Journal of Theoretical Biology*, **200** (4), 375–388.

Wu, M.C., David, S.V. and Gallant, J.L. (2006) Complete functional characterization of sensory neurons by system identification. *Annual Review of Neuroscience*, **29**, 477–505.

Yu, W. and Yao, L. (2007) Global robust stability of neural networks with time varying delays. *Journal of Computational and Applied Mathematics*, **206** (2), 679 – 687.

Zhang, Y. (1994) *A foundation for the design and analysis of robotic systems and behaviors.* Ph.D. thesis, University of British Columbia.