

Article

Approximating Heterotypic Bisimulations for Weighted Finite Automata over the Field of Real Numbers

Predrag Stanimirović ^{1,*} , Miroslav Ćirić ¹ , Darjan Karabašević ^{2,3,*}  and Dimitrios Gerontitis ⁴ ¹ Faculty of Sciences and Mathematics, University of Niš, Višegradska 33, 18108 Niš, Serbia; miroslav.ciric@pmf.edu.rs² Faculty of Applied Management, Economics and Finance, University Business Academy in Novi Sad, Jevrejska 24, 11000 Belgrade, Serbia³ College of Global Business, Korea University, Sejong 30019, Republic of Korea⁴ Department of Information and Electronic Engineering, International Hellenic University, 57400 Thessaloniki, Greece; dimitrios_gerontitis@yahoo.gr

* Correspondence: predrag.stanimirovic@pmf.edu.rs (P.S.); darjan.karabasevic@mef.edu.rs (D.K.)

Abstract

This paper examines the existence and approximation of bisimulations between weighted finite automata (WFAs) over the real numbers. It shows that forward–backward bisimulation (fbb) and backward–forward bisimulation (bfb) between two WFAs are equivalent to solving specific homogeneous Sylvester equations and two vector equations. The transition matrices of the automata serve as the coefficient matrices in these equations. This approach reformulates the WFA’s problem as a linear algebra task involving real coefficient matrices. Obtained systems of vector and matrix equations frequently lack consistency. Multi-criteria optimization can address these inconsistencies. We apply continuous-time zeroing neural network (ZNN) dynamics to find approximate solutions for these inconsistent vector-matrix systems. Since ZNN dynamics are globally convergent, they generate approximate solutions that evolve over time. Simulation experiments are conducted on random transition matrices, using various initial state matrices and two activation functions.

Keywords: weighted finite automata; bisimulation; Sylvester equations; Zhang neural network**MSC:** 15A24; 65F20; 68Q70; 68T05

1. Introduction and Problem Description

Weighted finite automata (WFAs) are computational models applied across computer science and related disciplines. They are essential for analyzing the behavior of software and hardware systems, such as operating systems and network protocols. WFAs are also used to process the syntax and semantics of programming languages and to assign weights to words and sentences in natural language processing. They are further applied in image compression, speech recognition, and pattern recognition in text and other data. Recently, WFAs have become popular in machine learning as an alternative to recurrent neural networks for specific tasks.

WFAs extend traditional automata by adding weights to transitions and states. These weights capture quantitative properties such as cost, resource availability, and transition reliability.

The concept of quantitative (bi)simulations for fuzzy finite automata was explored in [1,2]. A similar approach was applied to study (bi)simulations between non-deterministic automata [3], WFAs over additively idempotent semirings [4], max-plus automata [5,6],



Academic Editor: Jorge Delgado Gracia

Received: 12 May 2026

Revised: 9 July 2026

Accepted: 16 July 2026

Published: 23 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

and WFAs over semirings [7,8]. A comparable methodology was also applied to social networks [9] and Kripke models of fuzzy multimodal logic [10,11]. In [12] it was shown that two probabilistic finite automata are equivalent only if there exists a bisimulation between them.

We investigate bisimulations for WFAs over the real numbers. Weighted automata over the real numbers were studied in [13], which explored bisimulation seminorms and pseudometrics, and in [14], which examined linear bisimulations. The focus of our investigation will be on the existence and fundamental characteristics of bisimulations between weighted automata over the real numbers.

Forward–backward bisimulation (fbb) and backward–forward bisimulation (bfb) between two WFAs require solving a system of homogeneous Sylvester equations and two additional vector equations. This approach reformulates the (bi)simulation problem as a linear algebra task over the real numbers. However, these systems of vector and matrix equations are inconsistent.

To address the complex challenge of computing approximate bisimulations between WFAs with real-number weights, this study develops a zeroing neural network (ZNN) approach that integrates multiple matrix and vector error functions (EFs) into a comprehensive block matrix. ZNN dynamics effectively solve matrix algebra problems because of their global exponential convergence, distributed structure, ease of hardware implementation, and suitability for TV problems. Additionally, ZNN dynamics can ensure convergence within a specified time frame [15,16]. Extensive research has addressed classical and finite-time ZNN dynamical systems for solving time-varying (TV) Sylvester equations [17–27]. Recently, several finite-time and predefined-time convergent ZNN models have been developed to solve the TV Sylvester equation [28–31]. However, solving these TV Sylvester equations requires significant CPU time due to the intensive calculations involved [19].

This issue is complex because of several challenges. The ZNN design developed in this research is based on a couple of EFs in both matrix and vector form. Each of these EFs is systematically integrated into a comprehensive block matrix. Each matrix EF is specifically designed to address a homogeneous Sylvester equation. The primary objective of ZNN dynamics is to reduce these error functions to zero, as shown in previous studies [16,32]. However, the ZNN model presented here addresses an inconsistent system of matrix and vector equations. Our approach uses ZNN neurodynamics to compute approximate bisimulations between WFAs with real-number weights, based on approximate solutions to the related matrix-vector systems. Approximate bisimulations help identify WFAs that behave similarly within a defined tolerance. Previous works [33–35] explored approximate bisimulations among fuzzy automata, leading to fuzzy automata with fewer states and minimal behavioral differences from the original. Additionally, Yang and Li [36–38] investigated approximate bisimulations as crisp relations on WFA over a Gödel structure.

Because the linear system in the ZNN model is inconsistent, we use the normal system to obtain the best approximate solution. This approach relies on the use of the pseudoinverse matrix. Our objective aligns with numerical linear algebra, and we use a ZNN dynamics-based algorithm to address this problem. To construct a ZNN, the first requirement is to define an appropriate error function for each matrix and vector equation in the observed system. Many studies have examined ZNN dynamics using various error functions, most notably [39–43]. The ZNN models that utilize block-matrix error functions have successfully ensured the convergence of each involved error function to zero.

Quantitative simulations and bisimulations between WFAs over $\mathbb{R}$ are determined by solving specific systems of matrix and vector inequalities, as discussed in [44]. Bisimulation is defined through a system of homogeneous Sylvester equations alongside two additional vector equations. These systems of vector and matrix equations are often inconsistent.

We aim to use continuous recurrent neural networks, specifically ZNNs, to find optimal approximate solutions for these inconsistent matrix-vector systems. The approach outlined in [44] is based on the 2-equivalence of two WFAs and a simultaneous method involving two unknown matrices. Our goal is to evaluate the performance of standard ZNN dynamics in addressing inconsistent systems that occur when finding (bi)simulations, rather than comparing different ZNN models.

The main findings of this paper are as follows.

- The heterotypic bisimulations are reformulated as inconsistent systems of matrix-vector equations.
- A multicriteria optimization (MOO) interpretation of approximate bisimulations is introduced.
- A continuous-time ZNN framework is developed for computing least-squares approximations of such inconsistent systems. The main contribution is not merely the application of an existing ZNN to a Sylvester equation. Rather, the paper introduces a unified neurodynamic framework for computing approximate bisimulations of WFAs, particularly in situations where exact bisimulations do not exist. Furthermore, the formulation establishes a foundation for future extensions to time-varying weighted automata and evolving transition structures, which cannot be handled directly by conventional static algebraic methods.
- The propagation of approximation errors through weighted automata behaviors is theoretically analyzed.
- Simulink implementation of the ZNN model is described, and numerical experience is presented, including comparison with multi-objective optimization methods.

The structure of our presentation is organized into the following sections. Section 2 addresses WFAs with weights over real numbers. Section 2.2 defines WFA, while Section 2.3 explains the concepts of simulations and bisimulations between WFAs. Section 3 presents the justification for applying approximate bisimulations in the context of WFAs. Survey of numerical methods for solving the coupled linear system is presented in Section 4. Zeroing neural network (ZNN) neurodynamic systems for computing approximate bisimulations are developed in Section 5. Propagation of approximations of bisimulations is described in Section 6. Section 7 covers the Simulink implementation of ZNN models and shares numerical experiences. Computational performance and CPU execution times for the proposed ZNN_{fb} and ZNN_{bfb} models are given in Section 7.1. Some final considerations and possibilities for further research are given in Section 8.

2. Preliminaries and Motivation

WFAs with weights in the field of real numbers have been extensively studied since the beginning of weighted automata theory, beginning with Schützenberger’s research on WFAs over fields in [45]. Additionally, ZNNs are widely used dynamical systems in scientific research and diverse applications. This section provides a brief overview of these two areas, which will be explored further in the following sections.

2.1. Definition of ZNN Dynamics

The creation of a ZNN model consists of two global steps. The first step is to choose an appropriate matrix or vector error function (EF), denoted as $E(t)$. An error function $E(t)$ is well-defined if its zero point, $E(t) = 0$, matches the theoretical solution of the problem. To ensure the convergence of the underlying error function $E(t)$ to zero, the ZNN design utilizes the following dynamical relation between $E(t)$ and its time derivative $\dot{E}(t)$

$$\dot{E}(t) = -\kappa \odot (E(t)), \quad (1)$$

where the design parameter $\kappa \in \mathbb{R}$, $\kappa > 0$ is used to scale the convergence and $\odot$ assumes elementwise application of an appropriate activation function [16,32]. Experiments are typically designed to make κ as large as possible, as a higher κ facilitates faster convergence. The convergence speed of the dynamical system (1) is adjusted by the learning rate parameter $\kappa \in \mathbb{R}^+$. The dynamics (1) converges in proportion to increasing κ [46]. The primary effect of the learning postulate utilized in (1) is to ensure the convergence $E(t) \rightarrow 0$ as $t \rightarrow \infty$ at an exponential rate defined by κ [16,46].

2.2. Definition of WFAs over Real Numbers

In the following, $\mathbb{R}$ (resp. $\mathbb{N}$) will denote the field of real numbers (resp. the set of natural numbers without zero included). Moreover, X denotes a non-empty finite set $\{x_1, x_2, \dots, x_\rho\}$ consisting of $\rho \in \mathbb{N}$, termed as *alphabet*, while $X^+ = \{x_{i_1}x_{i_2} \dots x_{i_s} \mid s \in \mathbb{N}, x_{i_1}, x_{i_2}, \dots, x_{i_s} \in X\}$ will be the set of all finite sequences with entries from X , which represent *words* over the alphabet X , and $X^* = X^+ \cup \{\varepsilon\}$, where $\varepsilon \notin X^+$ represents a symbol called the *empty word*.

A WFA over $\mathbb{R}$ and the alphabet X is defined as a quadruple $\mathcal{A} \equiv (m, \sigma^A, \{M_x^A\}_{x \in X}, \tau^A)$, in which $m \in \mathbb{N}$ represents the *dimension* of $\mathcal{A}$; $\sigma^A \in \mathbb{R}^{m \times 1}$, $\tau^A \in \mathbb{R}^{1 \times m}$ are vectors known, respectively, as the *initial vector* and the *terminal vector*, and $\{M_x^A\}_{x \in X} \subset \mathbb{R}^{m \times m}$ denotes a set of matrices known as the *transition matrices* of $\mathcal{A}$.

The set $\{M_x^A\}_{x \in X}$ is extended up to the set $\{M_v^A\}_{v \in X^*} \subset \mathbb{R}^{m \times m}$ as follows: we put $M_\varepsilon^A = I_m$, where I_m denotes the $m \times m$ identity matrix, while for $v = x_1x_2 \dots x_s \in X^+$, $x_1, x_2, \dots, x_s \in X$, we put $M_v^A = M_{x_1}^A \cdot M_{x_2}^A \dots M_{x_s}^A$, where the operator $\cdot$ denotes the usual multiplication of matrices. The matrices M_v^A , $v \in X^*$ will be termed as *compound transition matrices* of $\mathcal{A}$.

An arbitrary function $f : X^* \rightarrow \mathbb{R}$ is known as a *word function*. They are sometimes referred to as *weighted languages*, and when expressed using notation borrowed from classical power series theory, they are also referred to as *formal power series*. Specifically, each WFA $\mathcal{A} \equiv (m, \sigma^A, \{M_x^A\}_{x \in X}, \tau^A)$ leads to a word function $\llbracket \mathcal{A} \rrbracket : X^* \rightarrow \mathbb{R}$ defined by

$$\llbracket \mathcal{A} \rrbracket(v) = \sigma^A \cdot M_v^A \cdot \tau^A = \sigma^A \cdot M_{x_1}^A \cdot M_{x_2}^A \dots M_{x_s}^A \cdot \tau^A, \quad (2)$$

for $v = x_1x_2 \dots x_s \in X^+$ and

$$\llbracket \mathcal{A} \rrbracket(\varepsilon) = \sigma^A \cdot M_\varepsilon^A \cdot \tau^A = \sigma^A \cdot \tau^A. \quad (3)$$

Note again that σ^A is treated as a row vector (a $1 \times m$ matrix), while τ^A is treated as a column vector (an $m \times 1$ matrix).

In this setup, the initial vector σ^A acts as a weighted selection of starting states, and the terminal vector τ^A acts as a weighted exit or acceptance criteria for each state. This is essentially the linear representation of a recognizable formal power series.

We now restate core decision problems for WFAs over the real field. WFAs $\mathcal{A}$ and $\mathcal{B}$ over $\mathbb{R}$ and the alphabet X are *equivalent* if $\llbracket \mathcal{A} \rrbracket = \llbracket \mathcal{B} \rrbracket$. In addition, if $\llbracket \mathcal{A} \rrbracket \leq \llbracket \mathcal{B} \rrbracket$, by which we mean that $\llbracket \mathcal{A} \rrbracket(v) \leq \llbracket \mathcal{B} \rrbracket(v)$, for every $v \in X^*$, then $\mathcal{A}$ is said to be *contained in* $\mathcal{B}$.

The set of all positive real numbers will be denoted by $\mathbb{R}_+$. Recall that $(\mathbb{R}_+, +, \cdot, 0, 1)$ is a semiring, or more precisely, it is a positive semiring (for the definitions of a semiring and a positive semiring we refer to the book [47]).

2.3. Simulations and Bisimulations of WFAs and Motivation

For the sake of completeness, here we repeat definitions of simulations and bisimulations in the context of WFA over the real numbers $\mathbb{R}$. The results available in [7,48] are mainly restated.

Let $\mathcal{A} = (m, \sigma^A, \{M_x^A\}_{x \in X}, \tau^A)$ and $\mathcal{B} = (n, \sigma^B, \{M_x^B\}_{x \in X}, \tau^B)$ be WFA over $\mathbb{R}$ and the alphabet X . A matrix $U \in \mathbb{R}^{m \times n}$ is known as a *forward simulation* (fs) between $\mathcal{A}$ and $\mathcal{B}$ if it satisfies the subsequent conditions

$$\begin{aligned} \text{(fs-1)} \quad & \sigma^A \leq \sigma^B \cdot U^\top \\ \text{(fs-2)} \quad & U^\top \cdot M_x^A \leq M_x^B \cdot U^\top \quad (x \in X) \\ \text{(fs-3)} \quad & U^\top \cdot \tau^A \leq \tau^B. \end{aligned}$$

Here and throughout, $U^\top$ denotes the transpose of the matrix U . Further, U is called a *backward simulation* (bs) between $\mathcal{A}$ and $\mathcal{B}$ if it satisfies

$$\begin{aligned} \text{(bs-1)} \quad & \tau^A \leq U \cdot \tau^B \\ \text{(bs-2)} \quad & M_x^A \cdot U \leq U \cdot M_x^B \quad (x \in X) \\ \text{(bs-3)} \quad & \sigma^A \cdot U \leq \sigma^B \end{aligned}$$

Moreover, if both U and $U^\top$ are forward simulations between $\mathcal{A}$ and $\mathcal{B}$ and vice versa, i.e., if they satisfy

$$\begin{aligned} \text{(fb-1)} \quad & \sigma^A \leq \sigma^B \cdot U^\top, \quad \sigma^B \leq \sigma^A \cdot U \\ \text{(fb-2)} \quad & U^\top \cdot M_x^A \leq M_x^B \cdot U^\top, \quad U \cdot M_x^B \leq M_x^A \cdot U \quad (x \in X) \\ \text{(fb-3)} \quad & U^\top \cdot \tau^A \leq \tau^B, \quad U \cdot \tau^B \leq \tau^A \end{aligned}$$

then U is called a *forward bisimulation* (fb) between $\mathcal{A}$ and $\mathcal{B}$. If both U and $U^\top$ are bs between $\mathcal{A}$ and $\mathcal{B}$ and vice versa, i.e., if they satisfy

$$\begin{aligned} \text{(bb-1)} \quad & \tau^A \leq U \cdot \tau^B, \quad \tau^B \leq U^\top \cdot \tau^A \\ \text{(bb-2)} \quad & M_x^A \cdot U \leq U \cdot M_x^B, \quad M_x^B \cdot U^\top \leq U^\top \cdot M_x^A \quad (x \in X) \\ \text{(bb-3)} \quad & \sigma^A \cdot U \leq \sigma^B, \quad \sigma^B \cdot U^\top \leq \sigma^A \end{aligned}$$

then U represents a *backward bisimulation* (bb) between $\mathcal{A}$ and $\mathcal{B}$.

Quantitative forward and backward simulations as well as fb and bb bisimulations between WFA over $\mathbb{R}$ are outlined in [44]. These (bi)simulations correspond to solutions of systems defined by several matrix inequalities and two vector inequalities. However, fs, bs, fb, and bb can be formulated as solvable linear programming (LP) problems. In general, these problems are solvable using standard linear programming methods.

Unlike the foregoing models, heterotypic bisimulations can be restated as systems of matrix and vector equations. If U is an fs between $\mathcal{A}$ and $\mathcal{B}$ and $U^\top$ is a bs between $\mathcal{B}$ and $\mathcal{A}$, then U and $U^\top$ satisfy

$$\begin{aligned} \text{(fbb-1)} \quad & \sigma^A = \sigma^B \cdot U^\top \\ \text{(fbb-2)} \quad & U^\top \cdot M_x^A = M_x^B \cdot U^\top \quad (x \in X) \\ \text{(fbb-3)} \quad & U^\top \cdot \tau^A = \tau^B \end{aligned} \tag{4}$$

and U is known as a *forward–backward bisimulation* (fbb) between $\mathcal{A}$ and $\mathcal{B}$, and if U is a bs between $\mathcal{A}$ and $\mathcal{B}$ and $U^\top$ is a fs between $\mathcal{B}$ and $\mathcal{A}$, i.e., if U and $U^\top$ satisfy

$$\begin{aligned} \text{(bfb-1)} \quad & \tau^A = U \cdot \tau^B \\ \text{(bfb-2)} \quad & M_x^A \cdot U = U \cdot M_x^B \quad (x \in X) \\ \text{(bfb-3)} \quad & \sigma^A \cdot U = \sigma^B \end{aligned} \tag{5}$$

then U is termed a *backward–forward bisimulation (bfb)* between $\mathcal{A}$ and $\mathcal{B}$.

Simulations and bisimulations via inequalities reduce to feasibility problems in LP. Heterotypic bisimulations (fbb and bfb) are expressed in terms of equalities and reduce to solving coupled systems of matrix equations, free of inequality constraints.

In general, these systems of vector and matrix equations are inconsistent. Our goal is to employ continuous recurrent neural networks, specifically the zeroing neural network (ZNN), to obtain optimal approximate solutions for such inconsistent matrix-vector systems. Accordingly, we propose ZNNs as an efficient tool for solving inconsistent problems. A specific approach, based on the 2-equivalence of two weighted finite automata (WFAs) and a simultaneous method involving two unknown matrices, is applied to solve the matrix-vector equations required in heterotypic bisimulations between WFAs, as described in [44]. This method relies on the use of two unknown matrices, U_1 and U_2 . Our objective in this paper is to solve inconsistent matrix and vector systems underlying heterotypic (bi)simulations, specifically fbb and bfb, between WFAs. The distinctive aspect of this research is its focus on addressing inconsistent problems.

Comparison between the algebraic framework and the ZNN-based implementation is presented in Table 1.

Table 1. Comparison between the algebraic framework and the ZNN-based implementation.

Feature	Algebraic Definition	ZNN Implementation
Target	Exact bisimulation matrix U	Approximate/least-squares tracking matrix $U(t)$
Equations	(fbb)/(bfb) algebraic equations	Continuous-time neural dynamic equations
Error formulation	Exact algebraic constraints	Composite error function $\mathcal{E}(t) \rightarrow \mathbf{0}$
Methodology	Matrix-algebraic analysis	Continuous-time dynamical systems
Stability behavior	Sensitive to inconsistency and perturbations	Globally convergent and robust
Solution nature	Static exact solution	Dynamic online tracking solution
Computational framework	Offline algebraic computation	Real-time recurrent neural computation
Robustness	Limited under noisy/time-varying data	Adaptive to time-varying environments

3. MOO as an Innovative Tool to Find Approximate Bisimulations

Let $\mathcal{A} \equiv (m, \sigma^A, \{M_{x_i}^A\}_{x_i \in X}, \tau^A)$ and $\mathcal{B} \equiv (n, \sigma^B, \{M_{x_i}^B\}_{x_i \in X}, \tau^B)$ be WFA over $\mathbb{R}$. Consider $M_{x_i}^A \in \mathbb{R}^{m \times m}$, $\sigma^A \in \mathbb{R}^{1 \times m}$, $\tau^A \in \mathbb{R}^{m \times 1}$ and $M_{x_i}^B \in \mathbb{R}^{n \times n}$, $\sigma^B \in \mathbb{R}^{1 \times n}$, $\tau^B \in \mathbb{R}^{n \times 1}$ with the aim to solve (5).

One of the basic properties of vectorization used throughout this work is given by the following well-known *vectorization identity* [49]:

$$\text{vec}(PQR) = (R^T \otimes P) \text{vec}(Q), \quad (6)$$

valid for arbitrary $P \in \mathbb{R}^{k \times l}$, $Q \in \mathbb{R}^{l \times m}$ and $R \in \mathbb{R}^{m \times n}$, where $\otimes$ denotes the Kronecker product of matrices. As a result, each homogeneous Sylvester equation (bfb-2) can be expressed in vector form $Ax = b$ using the $\text{vec}()$ operator as follows:

$$(\text{bfb-2}^*) \quad (I_n \otimes M_{x_i}^A - (M_{x_i}^B)^T \otimes I_m) \text{vec}(U) = \mathbf{0} \quad (\forall x_i \in X).$$

Since the eigenvalues of the coefficient matrix in (bfb-2*) are defined by

$$\lambda_{ik}(I_n \otimes M_{x_i}^A - (M_{x_i}^B)^T \otimes I_m) = \lambda_i(I_n \otimes M_{x_i}^A) - \lambda_k((M_{x_i}^B)^T \otimes I_m),$$

it is nonsingular if $\lambda_i(M_{x_i}^A) \neq \lambda_k(M_{x_i}^B)$ for all i, k . So, each matrix equation $M_{x_i}^A \cdot U(t) = U(t) \cdot M_{x_i}^B$, as indicated in equation (bfb-2), possesses a nonzero solution if and only if $M_{x_i}^A$ and $M_{x_i}^B$ share at least one common eigenvalue. To illustrate this concept, we will refer to Example 1, where $M_{x_i}^A$ and $M_{x_i}^B$ indeed have at least one common eigenvalue.

Example 1. Let $m = 3$, $n = 2$ and $X = \{x_1, x_2\}$, and consider $\mathcal{A} = (m, \sigma^A, \{M_{x_i}^A\}_{x_i \in X}, \tau^A)$ and $\mathcal{B} = (n, \sigma^B, \{M_{x_i}^B\}_{x_i \in X}, \tau^B)$. Clearly $M_{x_i}^A \in \mathbb{R}^{m \times m}$, $\sigma^A \in \mathbb{R}^{1 \times m}$, $\tau^A \in \mathbb{R}^{m \times 1}$ and $M_{x_i}^B \in \mathbb{R}^{n \times n}$, $\sigma^B \in \mathbb{R}^{1 \times n}$, $\tau^B \in \mathbb{R}^{n \times 1}$. Consider $\mathcal{A} = (m, \sigma^A, \{M_{x_i}^A\}, = 1, 2), \tau^A)$ with components

$$\sigma^A = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

$$M_{x_1}^A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ -2 & 2 & 3 \end{bmatrix}, \quad M_{x_2}^A = \begin{bmatrix} -2 & 2 & 1 \\ -5 & 5 & 1 \\ 2 & 0 & 1 \end{bmatrix}$$

and $\mathcal{B} = (n, \sigma^B, \{M_{x_i}^B\}, i = 1, 2), \tau^B)$ defined by

$$\sigma^B = \begin{bmatrix} 1 & 0.5 \end{bmatrix}, \quad \tau^B = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$$M_{x_1}^B = \begin{bmatrix} 4 & -1 \\ 2 & 1 \end{bmatrix}, \quad M_{x_2}^B = \begin{bmatrix} \frac{11}{3} & -\frac{4}{3} \\ \frac{4}{3} & \frac{1}{3} \end{bmatrix}.$$

Consider unknown matrix U generated in the programming language Mathematica by the command $U = \text{Table}[\text{Subscript}[u, i, j], \{i, m\}, \{j, n\}]$, whose entries are free variables u_{ij} :

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \\ u_{3,1} & u_{3,2} \end{bmatrix}.$$

The eigenvalues of the matrices involved are equal to

$$\Lambda(M_{x_1}^A) = \{3, 2, 1\}, \quad \Lambda(M_{x_2}^A) = \{3, 2, -1\}, \quad \Lambda(M_{x_1}^B) = \{3, 2\}, \quad \Lambda(M_{x_2}^B) = \{3, 1\}.$$

Matrices $M_{x_1}^A$ and $M_{x_1}^B$ share a common eigenvalue 3, making the homogeneous Sylvester equation $M_{x_1}^A U = U M_{x_1}^B$ solvable, with the solutions defined by

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{1,1} & u_{1,2} \\ 2u_{1,1} + 2u_{1,2} & -u_{1,1} - u_{1,2} \end{bmatrix}.$$

Further, matrices $M_{x_2}^A$ and $M_{x_2}^B$ share a common eigenvalue 3, which implies solvability of the homogeneous Sylvester equation $M_{x_2}^A U = U M_{x_2}^B$ with the general solution

$$U = \begin{bmatrix} u_{1,1} & -\frac{u_{1,1}}{2} \\ 2u_{1,1} & -u_{1,1} \\ u_{1,1} & -\frac{u_{1,1}}{2} \end{bmatrix}.$$

However, the system

$$(bfb-2) \quad M_{x_i}^A U = U M_{x_i}^B, \quad i = 1, 2$$

possesses only the zero solution $U = 0_{3 \times 2}$.

Further, the complete system (5) is not solvable.

A part of Mathematica[®] code used in this example is presented as follows.

```
m = 3; n = 2;
DgMxA = {{3, 0, 0}, {0, 2, 0}, {0, 0, 1}}; (* Diagonal matrix with wanted
eigenvalues *)
PxA = {{1, 1, 0}, {1, 1, 1}, {1, 0, -1}}; (* Invertible matrix *)
MxA = PxA.DgMxA.Inverse[PxA];
DgMyA = {{3, 0, 0}, {0, 2, 0}, {0, 0, -1}};
DgMyA = {{3, 0, 0}, {0, 2, 0}, {0, 0, -1}};
MyA = PyA.DgMyA.Inverse[PyA];
DgMxB = {{3, 0}, {0, 2}}; PxB = {{1, 1}, {1, 2}};
MxB = PxB.DgMxB.Inverse[PxB];
DgMyB = {{3, 0}, {0, 1}}; PyB = {{2, 1}, {1, 2}};
MyB = PyB.DgMyB.Inverse[PyB];
sA = {{0, 1, 0}}; tA = {{1}, {1}, {1}}; sB = {{1, 0.5}}; tB = {{1}, {1}};
U = Table[Subscript[u, i, j], {i, m}, {j, n}];
ret = Solve[{MxA.U == U.MxB}, Flatten[U]]; ret = vars /. Sort[Flatten[ret]];
Us = Table[ret[[ (i-1)*n + j]], {i, m}, {j, n}];
ret = Solve[{MyA.U == U.MyB}, Flatten[U]]; ret = vars /. Sort[Flatten[ret]];
Us = Table[ret[[ (i-1)*n + j]], {i, m}, {j, n}];
ret = Solve[{MxA.U == U.MxB, MyA.U == U.MyB}, Flatten[U]]; (* Solve (bfb-2) *)
ret = vars /. Sort[Flatten[ret]];
Us = Table[ret[[ (i-1)*n + j]], {i, m}, {j, n}];
Solve[{sA.U == sB, MxA.U == U.MxB, MyA.U == U.MyB, tA == U.tB}, Flatten[U]]
(* Solve (bfb-1), (bfb-2), (bfb-3) *)
```

The research problem addressed in this study is specific, as no solution exists that simultaneously satisfies all matrix equations (bfb-1), (bfb-2), and (bfb-3). Therefore, the objective is to identify approximate solutions. Example 2 serves to illustrate the selection of appropriate methods.

Example 2. The aim of this example is to find an approximate solution to the system discussed in Example 1 using MOO. Our objective is to minimize Frobenius norm of vectors e_1, e_2 and matrices E_1, E_2 , defined by

$$\|e_1\|_2 = U \cdot \tau^B - \tau^A; \quad \|e_2\|_2 = \sigma^A \cdot U - \sigma^B$$

$$\|E_1\|_F = M_{x_1}^A \cdot U - U \cdot M_{x_1}^B; \quad \|E_2\|_F = M_{x_1}^A \cdot U - U \cdot M_{x_1}^B$$

adopting Mathematica[®] (Version 11.2) software described in [50].

Four objectives in this experiment are equal to norms defined by

```
e1 = Norm[sA.U - sB, "Frobenius"]; e2 = Norm[tA - U.tB, "Frobenius"];
E1 = Norm[MxA.U - U.MxB, "Frobenius"]; E2 = Norm[MyA.U - U.MyB, "Frobenius"];
```

Minima of individual objectives are generated by the Mathematica[®] expression

```
NMinimize[e1, Flatten[U]][[1]], NMinimize[E1, Flatten[U]][[1]],
NMinimize[E2, Flatten[U]][[1]], NMinimize[e2, Flatten[U]][[1]]
```

and equal to

$$\{e1, E1, E2, e2\} = \{1.26657 \times 10^{-11}, 2.56881 \times 10^{-11}, 3.54634 \times 10^{-7}, 7.60217 \times 10^{-12}\}.$$

Weighting coefficients in the weighting sum method are defined by

Needs["Combinatorica"] w = Compositions[4, 4]/4;

Under the choice {w1,w2,w3,w4}=w[[21]]={1/4, 1/4, 1/4, 1/4}

the minimum of w1*e1+w2*E1+w3*E2+w4*e2 is obtained by

NMinimize[w1*e1+w2*E1+w3*E2+w4*e2,Flatten[U]], and it is equal to

{0.710186, {u_{1,1} → 0.06899, u_{1,2} → −0.0353, u_{2,1} → 0.1345, u_{2,2} → −0.0817, u_{3,1} → 0.0536, u_{3,2} → 0.01301}}

Minimal values of the objectives are equal to

{e1,E1,E2,e2}={1.04279, 0.112014, 0.0421582, 1.64378}

Moreover, each combination of weights leads to different results [51].

The lexicographic MOO method defined by the order $E1 < E2 < e1 < e2$ is implemented as

M1 = NMinimize[E1, Flatten[U]]

M2 = NMinimize[{E2, E1 <= M1[[1]]}, Flatten[U]]

m1 = NMinimize[{e1, E2 <= M2[[1]], E1 <= M1[[1]]}, Flatten[U]]

m2 = NMinimize[{e2, e1 <= m1[[1]], E2 <= M2[[1]], E1 <= M1[[1]]}, Flatten[U]]

and produces the result

{e1,E1,E2,e2}={1.11765,0.000374883,0.000637271,1.73154}.

The outcomes produced by multi-objective optimization (MOO) methods are significantly influenced by the decision maker's selection of weights or the specified priority order. To address this issue, ZNN dynamics were proposed as a means to identify optimal solutions.

4. Numerical Methods for Solving the Coupled Bisimulation System

Let $U \in \mathbb{R}^{m \times n}$ be a time-varying bfb between $\mathcal{A}$ and $\mathcal{B}$ (generated by ZNN) for solving

$$\begin{aligned} \text{(bfb-1)} \quad & \tau^A = U \cdot \tau^B \\ \text{(bfb-2)} \quad & M_{x_i}^A \cdot U = U \cdot M_{x_i}^B \quad (\forall x_i \in X) \\ \text{(bfb-3)} \quad & \sigma^A \cdot U = \sigma^B. \end{aligned} \quad (7)$$

Exploiting vectorization and the Kronecker product, Equation (7) is reformulated in the equivalent form:

$$\begin{aligned} \text{(bfb-1)} \quad & ((\tau^B)^\top \otimes I_m) \text{vec}(U) = \tau^A \\ \text{(bfb-2)} \quad & (I_n \otimes M_{x_i}^A - (M_{x_i}^B)^\top \otimes I_m) \text{vec}(U) = \mathbf{0}_{mn} \quad (\forall x_i \in X) \\ \text{(bfb-3)} \quad & (I_n \otimes \sigma^A) \text{vec}(U) = (\sigma^B)^\top, \end{aligned} \quad (8)$$

where $\mathbf{0}_{mn \times 1}$ is the zero vector. Let the alphabet X contains the letters $x_1, \dots, x_s$. The block matrix form of (8) is the following:

$$L \text{vec}(U) = r_1, \quad (9)$$

where

$$L = \begin{bmatrix} (\tau^B)^\top \otimes I_m \\ I_n \otimes M_{x_1}^A - (M_{x_1}^B)^\top \otimes I_m \\ \dots \\ I_n \otimes M_{x_s}^A - (M_{x_s}^B)^\top \otimes I_m \\ I_n \otimes \sigma^A \end{bmatrix} \in \mathbb{R}^{(m+mn \cdot s+n) \times mn}, \quad r_1 = \begin{bmatrix} \tau^A \\ \mathbf{0}_{mn \times 1} \\ \dots \\ \mathbf{0}_{mn \times 1} \\ (\sigma^B)^\top \end{bmatrix} \in \mathbb{R}^{(m+mn \cdot s+n) \times 1}. \quad (10)$$

Dimensions of expressions involved in the block matrix L are as follows:

$$(\tau^B)^\top \otimes I_m \in \mathbb{R}^{m \times mn}, \quad I_n \otimes M_{x_i}^A - (M_{x_i}^B)^\top \otimes I_m \in \mathbb{R}^{mn \times mn}, \quad i = 1, \dots, s, \quad I_n \otimes \sigma^A \in \mathbb{R}^{n \times mn}.$$

The coefficient matrix L of the coupled linear system (9) incorporates all terminal, transition, and initial-state constraints. Consequently, solving (9) is equivalent to computing a backward–forward bisimulation matrix.

The coefficient matrix L incorporates all terminal-state, transition, and initial-state constraints defining a backward–forward bisimulation. Consequently, solving (9) is equivalent to computing a bisimulation matrix satisfying all algebraic constraints simultaneously. The choice of an appropriate numerical method depends primarily on the rank, conditioning, sparsity, and consistency of the coupled system. This section reviews the principal computational approaches and highlights their advantages and disadvantages.

4.1. Direct Application of Linear Algebra

If L has full column rank and the system is consistent, then $\text{vec}(U)$ can be obtained by standard linear algebra techniques, including Gaussian elimination or QR-factorization. These methods compute exact solutions with computational complexity approximately $O((mn)^3)$, making them suitable for small and medium-dimensional automata.

Their principal limitation is that they require exact consistency and must be restarted whenever the coefficient matrix changes.

4.2. Application of QR Decomposition

Instead of forming the normal equations, it is possible to compute $L = QR$, where $Q^\top Q = I$, R is upper triangular. The solution of (9) satisfies

$$R \text{vec}(U) = Q^\top r_1.$$

QR factorization is numerically more stable than Gaussian elimination and avoids the conditioning deterioration associated with the normal equations. Its computational complexity remains approximately $O((mn)^3)$.

4.3. Application of Singular Value Decomposition (SVD)

The most reliable algebraic approach is based on the SVD decomposition $L = P\Sigma Q^\top$, where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_k)$ contains the singular values of L . The Moore–Penrose pseudoinverse solution of (9) is given by

$$\text{vec}(U) = L^\dagger r_1, \quad L^\dagger = Q\Sigma^\dagger P^\top.$$

The SVD possesses several advantages: it works for rank-deficient systems, produces the minimum-norm solution, provides numerical ranks, computes condition numbers, and remains stable even for ill-conditioned matrices. Its higher computational cost is compensated by its superior numerical robustness.

4.4. Least-Squares Approximation

If the system (9) is inconsistent, no exact bisimulation exists. Then the minimization problem can be solved

$$\min_U \|L \text{vec}(U) - r_1\|_2^2.$$

The corresponding normal equations are given in the form

$$L^\top L \text{vec}(U) = L^\top r_1.$$

If $L^T L$ is nonsingular, U is generated using $\text{vec}(U) = (L^T L)^{-1} L^T r_1$. If $L^T L$ is singular, the normal equations admit infinitely many least-squares solutions. In this case the Moore–Penrose pseudoinverse provides the unique minimum-norm solution. Although computationally inexpensive, this formulation squares the condition number, $\kappa(L^T L) = \kappa(L)^2$, which may reduce numerical accuracy. If $L^T L$ is singular, the normal equations admit infinitely many solutions and $L^\dagger$ could be exploited.

4.5. Moore–Penrose Pseudoinverse Solution

The Moore–Penrose pseudoinverse computes the unique minimum-norm least-squares solution $\text{vec}(U) = L^\dagger r_1$, satisfying

$$\text{vec}(U) = \arg \min_x \|x\|_2 \quad \text{subject to} \quad x \in \arg \min_y \|Ly - r_1\|_2.$$

It is particularly suitable for inconsistent or rank-deficient bisimulation systems. As a result, $U = L^\dagger r_1$ is the unique best approximate bisimulation.

4.6. Application of Regularized Least Squares

When L is severely ill-conditioned, Tikhonov regularization computes

$$\text{vec}(U_\lambda) = (L^T L + \lambda I)^{-1} L^T r_1, \quad \lambda > 0.$$

Regularization replaces $L^T L$ by $L^T L + \lambda I$, whose eigenvalues satisfy $\mu_i + \lambda > 0$. The regularization parameter $\lambda > 0$ balances stability and approximation accuracy. Larger values improve conditioning but introduce greater bias into the computed solution.

4.7. Krylov Subspace Methods

For large-scale automata, direct factorization becomes impractical. Iterative methods such as LSQR, particularly suited to least-squares systems, GMRES for nonsymmetric systems, MINRES for symmetric systems, CGNE for normal equations, construct approximate solutions inside Krylov subspaces. Their computational cost is approximately $O(kmn)$, where k denotes the number of iterations and m, n are dimensions of the model. These methods require only matrix-vector products and therefore scale well to very large problems.

4.8. Application of Multiobjective Optimization

Unlike least-squares formulations, MOO approach treats each bisimulation constraint as an independent objective, allowing different trade-offs among terminal, transition, and initial-state conditions. The MOO is formulated as minimizing the following residuals simultaneously:

$$\|\sigma^A U - \sigma^B\|, \|M_x^A U - U M_x^B\|, \|U \tau^B - \tau^A\|.$$

Typical formulations include weighted-sum optimization, lexicographic optimization, Pareto optimization. These methods are attractive because they naturally balance conflicting constraints. However, These formulations require external preference information (weights or priority order).

4.9. Zhang Neural Networks

The proposed approach reformulates (9) as a continuous-time neural dynamical system. Starting from the error matrix

$$E(t) = L \text{vec}(U(t)) - r_1,$$

the evolution law is prescribed as $\dot{E}(t) = -\lambda E(t)$, $\lambda > 0$. Unlike algebraic methods, ZNN computes the complete solution trajectory $U(t)$, rather than only its equilibrium. Consequently, $U(t)$ can continuously adapt to changes in $L(t)$, $r_1(t)$, without restarting the computation. Its principal advantages are simultaneous treatment of all constraints, exponential convergence, robustness against perturbations, online computation, direct extension to time-varying automata, automatic computation of approximate bisimulations when the system is inconsistent.

4.10. Final Comparison

Remark 1. The selection of a numerical solver for (9) is determined by the algebraic characteristics of the coefficient matrix.

- When L possesses full column rank and the system is consistent, direct methods yield exact solutions.
- If the system (9) is inconsistent, least-squares or Moore–Penrose methods compute minimum residual solutions.
- When L is severely ill-conditioned, regularization techniques enhance numerical stability.
- For sparse or large-scale problems, Krylov subspace methods are preferable.
- ZNN transforms bisimulation from a static algebraic problem into a dynamic tracking paradigm. Thus, continuous-time ZNN dynamics enable online tracking of time-varying systems without the need for repeated matrix computations.

Comparison of numerical methods for solving the coupled bisimulation system is presented in Table 2. The quantity k denotes the number of iterative steps, m, n denote dimensions of the underlying problem, and $\text{nnz}(L)$ stands for number of nonzeros. Furthermore, the symbol $\checkmark$ indicates that the corresponding condition is satisfied, while $\times$ indicates that it is not satisfied.

Table 2. Comparison of numerical methods for solving the coupled bisimulation system.

Method	Exact	Approx.	Rank-Deficient	Time-Varying	Complexity	Robustness
Gaussian elimination	$\checkmark$	$\times$	$\times$	$\times$	$O((mn)^3)$	Medium
QR factorization	$\checkmark$	$\checkmark$	$\checkmark$	$\times$	$O((mn)^3)$	High
Singular Value Decomposition (SVD)	$\checkmark$	$\checkmark$	$\checkmark$	$\times$	$O((mn)^3)$	Very High
Moore–Penrose pseudoinverse	$\checkmark$	$\checkmark$	$\checkmark$	$\times$	$O((mn)^3)$	Very High
Regularized least squares	$\times$	$\checkmark$	$\checkmark$	$\times$	$O((mn)^3)$	High
Krylov subspace methods	$\checkmark$	$\checkmark$	$\checkmark$	Limited	$O(k \text{nnz}(L))$	High
Weighted-sum optimization	$\times$	$\checkmark$	$\checkmark$	Limited	Iterative	Moderate
Lexicographic optimization	$\times$	$\checkmark$	$\checkmark$	Limited	Iterative	Moderate
Pareto optimization	$\times$	$\checkmark$	$\checkmark$	Limited	Iterative	Moderate
Zhang Neural Network (ZNN)	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	Online	Very High

The comparison given in Remark 1 and Table 2 demonstrates that classical numerical linear algebra provides efficient tools for solving static bisimulation equations, whereas optimization-based approaches are appropriate for inconsistent systems. The proposed ZNN framework unifies both situations within a single continuous-time dynamical model. It recovers exact bisimulations whenever they exist and automatically computes minimum residual approximate bisimulations otherwise, while naturally extending to time-varying weighted finite automata.

5. Neurodynamic Systems for Computing Approximate Bisimulations

Let $U(t) \in \mathbb{R}^{m \times n}$ be a time-varying bfb between $\mathcal{A}$ and $\mathcal{B}$ (generated by ZNN) for solving

$$\begin{aligned} \text{(bfb-1)} \quad & \tau^A = U(t) \cdot \tau^B \\ \text{(bfb-2)} \quad & M_{x_i}^A \cdot U(t) = U(t) \cdot M_{x_i}^B \quad (\forall x_i \in X) \\ \text{(bfb-3)} \quad & \sigma^A \cdot U(t) = \sigma^B. \end{aligned} \quad (11)$$

Exploiting vectorization and the Kronecker product, Equation (7) is reformulated in the equivalent form:

$$\begin{aligned} \text{(bfb-1)} \quad & ((\tau^B)^\top \otimes I_m) \text{vec}(U(t)) = \tau^A \\ \text{(bfb-2)} \quad & (I_n \otimes M_{x_i}^A - (M_{x_i}^B)^\top \otimes I_m) \text{vec}(U(t)) = \mathbf{0}_{mn} \quad (\forall x_i \in X) \\ \text{(bfb-3)} \quad & (I_n \otimes \sigma^A) \text{vec}(U(t)) = (\sigma^B)^\top, \end{aligned} \quad (12)$$

The coefficient matrix form of the system (12) is defined in the general form (9) and (10). Our intention is to develop a ZNN model for solving (7). Using (9), which is the matrix form of (7), the error function is defined as follows:

$$E(t) = L \text{vec}(U(t)) - r_1. \quad (13)$$

It is known that the general ZNN dynamics are defined based on the indefinite error function $E(t)$, which can be in scalar, vector or matrix form [16,52].

Time derivative of the error function in (13) is equal to

$$\dot{E}(t) = L \text{vec}(\dot{U}(t)), \quad (14)$$

and the expanded ZNN model of the pattern (1) is defined by

$$L \text{vec}(\dot{U}(t)) = -\kappa \odot (L \text{vec}(U(t)) - r_1). \quad (15)$$

A general issue arises with the dynamical system (15). The mass matrix L is required to be nonsingular. However, in most cases, $L(t)$ possesses full column rank. Consequently, the matrix $L^\top L$ is nonsingular in these instances. To address this issue, Equation (15) should be replaced by the corresponding normal equations for $\text{vec}(\dot{U}(t))$.

$$L^\top L \text{vec}(\dot{U}(t)) = -\kappa L^\top \odot (L \text{vec}(U(t)) - r_1). \quad (16)$$

Any solution $\text{vec}(\dot{U}(t))$ to the normal system (16) minimizes the vector norm

$$\begin{aligned} & \|L^\top L \text{vec}(\dot{U}(t)) + \kappa \odot (L^\top L \text{vec}(U(t)) - r_1)\|_2 \\ & = \min \|L^\top L \text{vec}(\dot{U}(t)) + \kappa \odot (L^\top L \text{vec}(U(t)) - r_1)\|_2. \end{aligned}$$

The best approximate solution $\text{vec}(\dot{U}(t))$ to (16) is obtained using the matrix pseudoinverse. The matrix $L^\top L$ is nonsingular if the columns of L are linearly independent. In such cases, $\text{vec}(\dot{U}(t))$ can be computed by the pseudoinverse solution

$$\begin{aligned} \text{vec}(\dot{U}(t)) & = -\kappa (L^\top L)^{-1} L^\top \odot (E(t)) = -\kappa (L^\top L)^{-1} L^\top \odot (L \text{vec}(U(t)) - r_1) \\ & = -\kappa L^+ \odot (L \text{vec}(U(t)) - r_1) \end{aligned} \quad (17)$$

in which L^+ denotes the pseudoinverse of L .

Theorem 1. Let $\mathcal{A} = (m, \sigma^A, \{M_{x_i}^A\}_{x_i \in X}, \tau^A)$ and $\mathcal{B} = (n, \sigma^B, \{M_{x_i}^B\}_{x_i \in X}, \tau^B)$ be WFAs over $\mathbb{R}$, such that $M_{x_i}^A \in \mathbb{R}^{m \times m}$, $\sigma^A \in \mathbb{R}^{1 \times m}$, $\tau^A \in \mathbb{R}^{m \times 1}$, and $M_{x_i}^B \in \mathbb{R}^{n \times n}$, $\sigma^B \in \mathbb{R}^{1 \times n}$, $\tau^B \in \mathbb{R}^{n \times 1}$, $i \in [1, \dots, s]$. Let (9) be the corresponding vectorized coupled bisimulation system. Assume that the system is consistent and admits an exact solution $u_S = \text{vec}(U_S)$, where $Lu_S = r_1$.

The dynamical system based on (15) inline with the linear ZNN design (1) satisfies the following statements:

- (i) The residual $E(t)$ satisfies $\lim_{t \rightarrow \infty} E(t) = 0$.
- (ii) $Lu(t) \rightarrow r_1$, that is, the coupled bisimulation constraints are asymptotically satisfied.
- (iii) If the exact bisimulation is unique (equivalently, L has full column rank), then

$$u(t) \rightarrow u_S \iff U(t) \rightarrow U_S.$$

- (iv) If the exact solution is not unique, then every accumulation point of $u(t)$ belongs to the affine solution set

$$\mathcal{S} = \{u \in \mathbb{R}^{mn} : Lu = r_1\},$$

and the limiting solution depends on the initial condition of the neural trajectory.

Moreover, the equilibrium corresponding to the exact bisimulation is globally exponentially stable in the sense of Lyapunov.

Proof. Since $E(t) = Lu(t) - r_1$, the linear ZNN design (1) possesses the explicit solution $E(t) = e^{-\lambda t}E(0)$, which immediately implies $\|E(t)\|_2 = e^{-\lambda t}\|E(0)\|_2 \rightarrow 0 \iff Lu(t) \rightarrow r_1$. To establish stability, consider the Lyapunov function $V(t) = \frac{1}{2}\|E(t)\|_2^2$. Its derivative is equal to

$$\dot{V}(t) = E(t)^T \dot{E}(t) = -\lambda E(t)^T E(t) = -\lambda \|E(t)\|_2^2 = -2\lambda V(t).$$

Hence, $V(t) = e^{-2\lambda t}V(0)$, which shows $V(t) > 0$ and $\dot{V}(t) < 0$ for $E \neq 0$. Therefore the equilibrium $E = 0$ is globally exponentially stable in the sense of Lyapunov.

If the exact bisimulation is unique, then the linear system (9) has the unique solution u_S . The convergence $Lu(t) \rightarrow r_1$ implies

$$u(t) \rightarrow u_S \iff U(t) \rightarrow U_S.$$

If the solution is nonunique, convergence of the residual only guarantees that every accumulation point satisfies (9). Hence every limit point belongs to the affine solution manifold $\mathcal{S} = \{u : Lu = r_1\}$, and the particular solution reached by the neural dynamics depends on the initial condition. $\square$

Theorem 2. Let $\mathcal{A} = (m, \sigma^A, \{M_{x_i}^A\}_{x_i \in X}, \tau^A)$ and $\mathcal{B} = (n, \sigma^B, \{M_{x_i}^B\}_{x_i \in X}, \tau^B)$ be WFAs defined by $M_{x_i}^A \in \mathbb{R}^{m \times m}$, $\sigma^A \in \mathbb{R}^{1 \times m}$, $\tau^A \in \mathbb{R}^{m \times 1}$ and $M_{x_i}^B \in \mathbb{R}^{n \times n}$, $\sigma^B \in \mathbb{R}^{1 \times n}$, $\tau^B \in \mathbb{R}^{n \times 1}$, $i \in [1, \dots, r]$. Starting from arbitrary initialization $\text{vec}(U(0))$, the generated neural trajectory converges exponentially to an exact bisimulation solution. More precisely, $E(t) = e^{-\lambda t}E(0)$, and consequently $L \text{vec}(U(t)) \rightarrow r_1$ as $t \rightarrow \infty$.

If the coupled system possesses a unique solution $\text{vec}(U_S(t))$, then $\text{vec}(U(t)) \rightarrow \text{vec}(U_S(t))$ exponentially.

If multiple exact bisimulations exist, then every accumulation point of $\text{vec}(U(t))$ belongs to the solution set of (9), and the limiting solution is determined by the initial condition.

Proof. The system (7) defines the solution $\text{vec}(U(t))$ which affiliates to the time-varying bfb between $\mathcal{A}$ and $\mathcal{B}$. Next, the system (7) is rewritten into (8) and then into (9) for generating $\text{vec}(U_S(t))$. Thirdly, the error function (13) is established to solve the system (9) and the ZNN evolution is exploited to generate the solution $\text{vec}(U(t))$ of (7). Later, Equation (15)

is generated by the ZNN design (1) aimed at zeroing (13). In accordance with Theorem 1, it follows $E(t) \rightarrow 0$ as $t \rightarrow \infty$. In consequence, the solution of the dynamical system (17) tends to $\text{vec}(U_S(t))$ as $t \rightarrow \infty$. Moreover, it is evident that (17) is another form of (15). $\square$

If $L^\top L \in \mathbb{R}^{mn \times mn}$ is singular we employ a small Tikhonov regularization parameter $\lambda > 0$, which leads to

$$(L^\top L + \lambda I_{mn}) \text{vec}(\dot{\mathbf{U}}(t)) = -\kappa L^\top \odot (E(t)), \quad (18)$$

where $\lambda > 0$ is a small regularization parameter. Since $L^\top L + \lambda I_{mn}$ is regular, Equation (18) is uniquely solvable by

$$\text{vec}(\dot{\mathbf{U}}(t)) = -\kappa (L^\top L + \lambda I_{mn})^{-1} L^\top \odot (L \text{vec}(\mathbf{U}(t)) - r_1), \quad (19)$$

The main idea is to use the Tikhonov regularization and the limit representation of the pseudoinverse $(L^\top L + \lambda I_{mn})^{-1} L^\top \rightarrow L^\dagger$ as $\lambda \rightarrow 0$ [53], which means that any solution $\text{vec}(\dot{\mathbf{U}}(t))$ to (19) converges to the solution of (17). The dynamical systems (17) in nonsingular case or the model (19) in singular case will be termed as ZNNbfb. We will say that ZNNbfb is “best approximate” dynamics to (15). Symbolically, we will write

$$\text{ZNNbfb} \div \mathcal{A} \equiv (m, \sigma^A, \{M_{x_i}^A\}_{x_i \in X}, \tau^A) \rightleftharpoons \mathcal{B} \equiv (n, \sigma^B, \{M_{x_i}^B\}_{x_i \in X}, \tau^B).$$

Remark 2. Clearly, the structure of the matrix L is sparse. So, it will be interesting to improve considered algorithms using techniques available for sparse matrices.

Now, our goal is to reduce ZNN dynamics ZNNbfb to approximate the fbb (4) between $\mathcal{A}$ and $\mathcal{B}$ to the developed dynamics ZNNbfb. Let $U(t) \in \mathbb{R}^{m \times n}$ be a time-varying matrix of state variables in the ZNN dynamics which defines fbb between $\mathcal{A}$ and $\mathcal{B}$ for solving

$$\begin{aligned} \text{(fbb-1)} \quad & \sigma^A = \sigma^B \cdot U^\top(t) \\ \text{(fbb-2)} \quad & U^\top(t) \cdot M_{x_i}^A = M_{x_i}^B \cdot U^\top(t) \quad (\forall x_i \in X) \\ \text{(fbb-3)} \quad & U^\top(t) \cdot \tau^A = \tau^B. \end{aligned} \quad (20)$$

Let $U(t) \in \mathbb{R}^{m \times n}$ be a time-varying matrix of state variables in the ZNN dynamics which defines fbb between $\mathcal{A}$ and $\mathcal{B}$ for solving (20). Our intention is to apply the notion of transposed automaton from [54] to reverse the transitions’ flow direction. Transposition of equations involved in (20) leads to the following equivalent form of fbb between $\mathcal{A}$ and $\mathcal{B}$:

$$\begin{aligned} \text{(fbb-1T)} \quad & (\sigma^A)^\top = U(t) \cdot (\sigma^B)^\top \\ \text{(fbb-2T)} \quad & (M_{x_i}^A)^\top \cdot U(t) = U(t) \cdot (M_{x_i}^B)^\top \quad (\forall x_i \in X) \\ \text{(fbb-3T)} \quad & (\tau^A)^\top \cdot U(t) = (\tau^B)^\top. \end{aligned} \quad (21)$$

Comparing (7) and (21), clear conclusion is that the fbb between $\mathcal{A} \equiv (m, \sigma^A, \{M_{x_i}^A\}_{x_i \in X}, \tau^A)$ and $\mathcal{B} \equiv (n, \sigma^B, \{M_{x_i}^B\}_{x_i \in X}, \tau^B)$ is equivalent to bfb between $\mathcal{A} \equiv (m, (\tau^A)^\top, \{(M_{x_i}^A)^\top\}_{x_i \in X}, (\sigma^A)^\top)$ and $\mathcal{B} \equiv (n, (\tau^B)^\top, \{(M_{x_i}^B)^\top\}_{x_i \in X}, (\sigma^B)^\top)$. Symbolically, we will write

$$\begin{aligned} \text{ZNNbfb} \div \mathcal{A} \equiv (m, \sigma^A, \{M_{x_i}^A\}_{x_i \in X}, \tau^A) & \rightleftharpoons \mathcal{B} \equiv (n, \sigma^B, \{M_{x_i}^B\}_{x_i \in X}, \tau^B) \iff \\ \text{ZNNbfb} \div \mathcal{A}^\top \equiv (m, (\tau^A)^\top, \{(M_{x_i}^A)^\top\}_{x_i \in X}, (\sigma^A)^\top) & \rightleftharpoons \mathcal{B}^\top \equiv (n, (\tau^B)^\top, \{(M_{x_i}^B)^\top\}_{x_i \in X}, (\sigma^B)^\top). \end{aligned} \quad (22)$$

Therefore, ZNNfbb dynamics can be expressed as ZNNbfb dynamics on transposed systems, inheriting the same stability and convergence properties.

5.1. Exact and Approximate Bisimulations

The asymptotic behavior of the proposed ZNN dynamics depends critically on whether the coupled bisimulation system is consistent.

Approximate bisimulations are characterized by the residual of the coupled bisimulation equations. Recall that the exact bfb or fbb conditions are assembled into the vectorized linear system (9), where the matrix L contains all terminal, transition, and initial-state constraints. The corresponding residual vector is defined by $R(U) = L \text{vec}(U) - r_1$.

If the system is consistent, then there exists an exact bisimulation satisfying $R(U) = 0$.

If the system is inconsistent, no exact bisimulation exists. In this case, the best attainable approximation is obtained by solving the least-squares problem

$$U_{LS} = \arg \min_U \|L \text{vec}(U) - r_1\|_2,$$

whose minimum residual is

$$R^* = L \text{vec}(U_{LS}) - r_1.$$

The quantity

$$\varepsilon^* = \|R^*\|_2 = \min_U \|L \text{vec}(U) - r_1\|_2$$

represents the approximation error of the coupled bisimulation system. Consequently, ε^* provides a mathematically rigorous measure of the degree of behavioral similarity between the two automata.

For practical applications, one may additionally prescribe an admissible tolerance $\delta > 0$. A matrix U is then regarded as an acceptable approximate bisimulation whenever

$$\|L \text{vec}(U) - r_1\|_2 \leq \delta.$$

The threshold δ is application dependent and may be selected according to numerical precision, measurement uncertainty, noise level, or engineering accuracy requirements.

Equivalently, the residual may be expressed in terms of the original bisimulation constraints,

$$r_1 = \sigma^A - \sigma^B U^\top, \quad r_{2,x} = U^\top M_x^A - M_x^B U^\top \quad (x \in X), \quad r_3 = U^\top \tau^A - \tau^B,$$

leading to the global error measure

$$\varepsilon(U) = \|r_1\|_2^2 + \sum_{x \in X} \|r_{2,x}\|_F^2 + \|r_3\|_2^2.$$

Exact and least-squares approximate bisimulations under ZNN dynamics are investigated in Theorem 3.

Theorem 3. Let L and r_1 be defined by the vectorized coupled bisimulation system (9). The asymptotic behavior of the proposed ZNN dynamics depends on whether the coupled system is consistent.

(i) **Exact bisimulation for consistent system.** Assume that the system (9) is consistent and define the residual

$$E(t) = L \text{vec}(U(t)) - r_1.$$

The ZNN dynamics $\dot{E}(t) = -\lambda E(t)$, $\lambda > 0$ leads to $E(t) = e^{-\lambda t} E(0)$, which implies $\lim_{t \rightarrow \infty} E(t) = 0$. Hence, $L \text{vec}(U(t)) \rightarrow r_1$. Moreover,

- (a) if the exact bisimulation is unique, then $\text{vec}(U(t)) \rightarrow \text{vec}(U^*)$, where U^* is the unique exact bisimulation;
- (b) if multiple exact bisimulations exist, every accumulation point of $\text{vec}(U(t))$ belongs to the solution set

$$\mathcal{S} = \left\{ \text{vec}(U) : L \text{vec}(U) = r_1 \right\},$$

and the limiting solution depends on the initial condition of the neural trajectory.

- (ii) **Approximate bisimulation for inconsistent system.** Assume that (9) is inconsistent. Then $\min_U \|L \text{vec}(U) - r_1\|_2 > 0$ and no exact bisimulation exists. The corresponding least-squares approximate bisimulation is defined as the solution of

$$\min_U \|L \text{vec}(U) - r_1\|_2,$$

whose first-order optimality conditions are normal equations $L^\top L \text{vec}(U) = L^\top r_1$. Define the least-squares residual

$$E_{LS}(t) = L^\top (L \text{vec}(U(t)) - r_1).$$

The least-squares ZNN dynamics $\dot{E}_{LS}(t) = -\lambda E_{LS}(t)$, $\lambda > 0$ leads to $E_{LS}(t) = e^{-\lambda t} E_{LS}(0)$, and therefore $\lim_{t \rightarrow \infty} E_{LS}(t) = 0$. Consequently, every accumulation point of $\text{vec}(U(t))$ is a least-squares minimizer satisfying $L^\top L \text{vec}(U) = L^\top r_1$.

- (a) If the least-squares minimizer is unique, then

$$\text{vec}(U(t)) \rightarrow \text{vec}(U_{LS}),$$

where

$$U_{LS} = \arg \min_U \|L \text{vec}(U) - r_1\|_2.$$

- (b) If multiple least-squares solutions exist, then $L^\dagger$ determines the unique minimum-norm least-squares solution $\text{vec}(U_{LS}) = L^\dagger r_1$. The limiting residual $R^* = L \text{vec}(U_{LS}) - r_1$ satisfies $R^* \in \mathcal{N}(L^\top) = \mathcal{R}(L)^\perp$, and

$$\|R^*\|_2 = \min_U \|L \text{vec}(U) - r_1\|_2 > 0.$$

Consequently,

$$L \text{vec}(U(t)) - r_1 \rightarrow R^*,$$

while $E_{LS}(t) \rightarrow 0$. Thus, the least-squares optimality conditions are satisfied asymptotically even though the original coupled bisimulation equations remain incompatible.

Proof. (i) Exact bisimulation. According to ZNN dynamics, the residual satisfies $E(t) = e^{-\lambda t} E(0)$, which implies $\lim_{t \rightarrow \infty} E(t) = 0$ and further $L \text{vec}(U(t)) \rightarrow r_1$.

If the solution is unique, convergence of the ZNN trajectory immediately yields $\text{vec}(U(t)) \rightarrow \text{vec}(U^*)$.

When the solution set is non-singleton, every accumulation point satisfies (9) and therefore belongs to the affine solution manifold $\mathcal{S} = \{\text{vec}(U(t)) : L \text{vec}(U(t)) = r_1\}$.

(ii) Approximate bisimulations. If the coupled bisimulation equations in (9) are inconsistent, no exact solution exists. The least-squares problem

$$\min_U \|L \text{vec}(U) - r_1\|_2^2,$$

has stationary points characterized by the normal equations

$$L^T L \operatorname{vec}(U) = L^T r_1.$$

Define the least-squares residual

$$E_{LS} = L^T (L \operatorname{vec}(U) - r_1).$$

Corresponding ZNN dynamical system $\dot{E}_{LS} = -\lambda E_{LS}$ satisfies $E_{LS}(t) = e^{-\lambda t} E_{LS}(0)$, and further $E_{LS}(t) \rightarrow 0$, which implies that every accumulation point satisfies the normal equations.

If this minimizer is unique, then $\operatorname{vec}(U(t)) \rightarrow \operatorname{vec}(U_{LS})$, where U_{LS} is the unique least-squares minimizer.

If the least-squares solution is nonunique, the Moore–Penrose inverse selects the unique minimum-norm solution $\operatorname{vec}(U_{LS}) = L^\dagger r_1$. Finally,

$$R^* = L \operatorname{vec}(U_{LS}) - r_1 = -(I - LL^\dagger)r_1$$

belongs to $\mathcal{N}(L^T) = \mathcal{R}(L)^\perp$ and satisfies

$$\|R^*\|_2 = \min_U \|L \operatorname{vec}(U) - r_1\|_2 > 0.$$

Therefore, although the least-squares optimality residual converges to zero, the original bisimulation residual converges to the smallest attainable residual, completing the proof. $\square$

Interpretation. The distinction between these two cases is fundamental:

1. If an exact bisimulation exists, ZNN dynamics recover it with exponentially vanishing residual.
2. If the equations are inconsistent, ZNN dynamics cannot drive the residual to zero, but converge to the least-squares (minimum-norm) approximate bisimulation.
3. Hence, the same neurodynamic system unifies exact and approximate bisimulation computation: exact solutions are tracked when they exist, and approximate solutions are generated otherwise.

Remark 3. Although the proposed ZNN approach and the classical normal-equation method are based on different computational principles, they are closely related through their common least-squares objective. Both aim to determine a matrix U that minimizes the residual norm $\|L \operatorname{vec}(U) - r_1\|_2$, but they achieve this objective in fundamentally different ways.

- The normal-equation approach is a static algebraic method. It computes the least-squares solution by solving the normal equation $L^T L \operatorname{vec}(U) = L^T r_1$.
- The ZNN approach is a continuous-time dynamical method. Rather than solving an algebraic system directly, it drives the residual $E(t) = L \operatorname{vec}(U(t)) - r_1$ according to the ZNN evolution law $\dot{E}(t) = -\lambda E(t)$, $\lambda > 0$, so that the neural trajectory converges asymptotically to the corresponding equilibrium.
- When the bisimulation system is consistent, both approaches recover an exact bisimulation satisfying (9).
- When the system is inconsistent, both methods seek a least-squares approximation. Under the assumptions of Theorem 3, the ZNN dynamics converge to the same minimum residual equilibrium that is obtained by the Moore–Penrose least-squares solution.
- Perhaps the most significant distinction is that the normal-equation method is intrinsically designed for static matrix systems, whereas ZNN naturally extends to time-varying problems. When the coefficient matrix and the right-hand side become time dependent, i.e., $L = L(t)$ and $r_1 = r_1(t)$, and the same neural dynamics continuously track the evolving solution without

restarting the computation, making ZNN particularly suitable for time-varying and real-time weighted automata.

Theorem 4 provides criteria for deciding whether two WFAs admit an exact behavioral correspondence or only approximate.

Theorem 4. Let L and r_1 define the vectorized coupled system (9) Define the minimum attainable residual

$$\varepsilon^* = \min_{u \in \mathbb{R}^m} \|Lu - r_1\|_2, \quad u = \text{vec}(U).$$

Then the following statements are valid.

1. An exact bisimulation exists if and only if $\varepsilon^* = 0 \iff r_1 \in \mathcal{R}(L)$, where $\mathcal{R}(L)$ denotes the column space of L .
2. If $\varepsilon^* > 0$, the coupled system (9) is inconsistent, and no exact bisimulation exists. In this case, the least-squares problem $\min_{u \in \mathbb{R}^m} \|Lu - r_1\|_2$ admits at least one minimizer, each defining an optimal approximate bisimulation $U = \text{unvec}(u)$, whose residual norm is equal to ε^* .
3. If the least-squares minimizer is not unique (i.e., L is rank-deficient), the Moore–Penrose inverse defines the unique best-approximate bisimulation

$$u^\dagger = L^\dagger r_1, \quad U^\dagger = \text{unvec}(u^\dagger),$$

with corresponding approximation error

$$\varepsilon^* = \|Lu^\dagger - r_1\|_2.$$

Proof. (i) If the system is consistent, there exists u^* with $Lu^* = r_1$, satisfying $\varepsilon^* = \|Lu^* - r_1\|_2 = 0$. Conversely, if $\varepsilon^* = 0$, then there exists a vector u^* satisfying $Lu^* = r_1$, so an exact bisimulation exists. The equivalence $Lu = r_1 \iff r_1 \in \mathcal{R}(L)$ follows from the definition of the column space.

(ii) If the system is inconsistent, no exact solution of (9) exists and $\varepsilon^* > 0$. The function $f(u) = \|Lu - r_1\|_2^2$ is convex, continuous and quadratic, and bounded below by zero, so minimizers of this norm exist. Moreover,

$$\nabla f(u) = 2L^\top(Lu - r_1),$$

so every minimizer satisfies the normal equation $L^\top Lu = L^\top r_1$ characterizes all minimizers. Therefore, each minimizer determines an optimal approximate bisimulation with residual norm equal to ε^* .

(iii) If multiple minimizers exist ($\mathcal{N}(L) \neq \{0\}$), the Moore–Penrose pseudoinverse yields the unique minimum-norm solution $u^\dagger = L^\dagger r_1$. Consequently, $U^\dagger = \text{unvec}(u^\dagger)$ is the unique minimum-norm approximate bisimulation, and its residual satisfies $\varepsilon^* = \|Lu^\dagger - r_1\|_2$. $\square$

Corollary 1 (Intrinsic approximation error). The quantity

$$\varepsilon^* = \min_u \|Lu - r_1\|_2$$

is an intrinsic property of the coupled bisimulation system. Geometrically, ε^* represents the Euclidean distance from r_1 to the column space of L

$$\varepsilon^* = \text{dist}(r_1, \mathcal{R}(L)).$$

Consequently:

1. $\epsilon^* = 0$ if and only if an exact bisimulation exists;
2. $\epsilon^* > 0$ if and only if only approximate bisimulations exist;
3. ϵ^* is independent of the numerical method used and thus provides a rigorous, solver-independent measure of behavioral similarity between two weighted finite automata.

5.2. Relation Between the Original ZNN System and the Normal-Equation-Based System

The proposed ZNN model demonstrates a significant connection to the classical least-squares formulation, which is derived from the normal equations. Both approaches aim to solve the coupled bisimulation system $L \text{vec}(U) = r_1$, but they differ fundamentally in their mathematical formulation, numerical properties, and computational objectives.

Consistent case. The original ZNN model is constructed directly from the residual

$$E(t) = L \text{vec}(U(t)) - r_1.$$

The zeroing principle prescribes exponential decay, represented by $\dot{E}(t) = -\lambda E(t)$ for $\lambda > 0$. This formulation results in $E(t) = e^{-\lambda t} E(0)$ and indicates that $\lim_{t \rightarrow \infty} E(t) = 0$, provided the system remains consistent. Given the definition of $E(t)$, the neural trajectory satisfies $L \text{vec}(U(t)) \rightarrow r_1$ and therefore converges to an exact bisimulation when such a solution exists. As a result, the original ZNN directly minimizes the residual associated with the coupled bisimulation equations.

Inconsistent case. If the equation $L \text{vec}(U) = r_1$ is inconsistent, then there does not exist any vector that satisfies $L \text{vec}(U) = r_1$. In this scenario, $\min_U \|L \text{vec}(U) - r_1\|_2 > 0$, which implies that $E(t)$ does not converge to zero. Consequently, the zeroing objective cannot be achieved. Therefore, the original ZNN model is appropriate for consistent coupled bisimulation systems; however, it is unable to asymptotically eliminate the residual in inconsistent systems.

Normal-equation formulation. For inconsistent systems, the problem is reformulated as a least-squares optimization problem $\min_U \|L \text{vec}(U) - r_1\|_2^2$. The first-order optimality condition for this problem is expressed as follows.

$$L^T (L \text{vec}(U) - r_1) = 0.$$

Instead of minimizing the original residual, the ZNN flow is formulated using the normal-equation residual $E_{LS}(t) = L^T (L \text{vec}(U(t)) - r_1)$, with its dynamics defined by $\dot{E}_{LS}(t) = -\lambda E_{LS}(t)$. Since $E_{LS}(t) = e^{-\lambda t} E_{LS}(0)$, it follows that $E_{LS}(t) \rightarrow 0$. As a result, the neural trajectory converges to a stationary point of the least-squares functional

$$\text{vec}(U_{LS}) = \arg \min_U \|L \text{vec}(U) - r_1\|_2.$$

Geometric interpretation. The distinction between the two formulations is illustrated by their equilibrium conditions.

The original ZNN enforces $L \text{vec}(U) = r_1$, which requires $r_1 \in \mathcal{R}(L)$.

In contrast, the least-squares ZNN formulation imposes only the following requirement:

$$L^T (L \text{vec}(U) - r_1) = 0.$$

This condition implies that the limiting residual $R^* = L \text{vec}(U_{LS}) - r_1$ is orthogonal to the column space of L : $R^* \in \mathcal{N}(L^T) = \mathcal{R}(L)^\perp$. Consequently, the least-squares solution represents the orthogonal projection of r_1 onto $\mathcal{R}(L)$.

Relationship between the standard and least-squares ZNN models. The least-squares ZNN is not a different algorithm but a natural extension of the original ZNN.

- If the coupled bisimulation system is consistent, then $r_1 \in \mathcal{R}(L)$, the optimal residual is $R^* = 0$, and the least-squares ZNN becomes equivalent exactly to the original ZNN. In this scenario, both models converge to the same exact bisimulation.
- If the coupled system is inconsistent, then $R^* \neq 0$, and the original ZNN cannot enforce $L \text{vec}(U) - r_1 = 0$. In contrast, the least-squares ZNN drives only the projected residual $L^\top (L \text{vec}(U) - r_1)$ to zero, thereby yielding the optimal approximate bisimulation.
- Consequently, the original ZNN addresses the exact coupled bisimulation problem, whereas the least-squares ZNN provides an optimal approximation when it is not possible to satisfy all constraints exactly.

Remark 4. Standard and least-squares ZNN evolutions yield equivalent results when the coupled bisimulation equations are consistent. Under these conditions,

$$L^\top (L \text{vec}(U) - r_1) = 0 \iff L \text{vec}(U) = r_1,$$

since the residual is identically zero in this scenario.

6. Behavior of Approximate Bisimulations

Consider arbitrary elements $x_1, \dots, x_s$ of the alphabet X and the WFA defined by $\mathcal{A} \equiv (m, \sigma^A, \{M_{x_i}^A\}_{x_i \in X}, \tau^A)$ and $\mathcal{B} \equiv (n, \sigma^B, \{M_{x_i}^B\}_{x_i \in X}, \tau^B)$. Theorem 5 investigates properties of approximations generated by the bfb and fbb dynamics.

Theorem 5. Let $\mathfrak{U}(t)$ be a TV backward–forward bisimulation (bfb) between $\mathcal{A}$ and $\mathcal{B}$ generated by the ZNNbfb dynamics (17) or (19) for solving (7).

(a) Assume that the best approximate solution $\mathfrak{U}(t)$ generated by (17) or (19) for chosen initial state $U(0)$ of $U(t)$ satisfies

$$\begin{aligned} \mathfrak{U}(t) \cdot \tau^B &= \tau^A + \vec{\delta}_\tau(t), \\ M_{x_i}^A \cdot \mathfrak{U}(t) &= \mathfrak{U}(t) \cdot M_{x_i}^B + \Delta_{x_i}^A(t), \quad i = 1, \dots, s, \\ \sigma^A \cdot \mathfrak{U}(t) &= \sigma^B + \vec{\delta}_\sigma(t), \end{aligned} \quad (23)$$

for appropriate vectors $\vec{\delta}_\tau(t) \in \mathbb{R}^{m \times 1}$ and $\vec{\delta}_\sigma(t) \in \mathbb{R}^{1 \times n}$, and appropriate matrices $\Delta_{x_i}^A(t) \in \mathbb{R}^{m \times m}$, $x_i \in X$, $i = 1, \dots, s$.

Then ZNNbfb dynamics (19) satisfies

$$\llbracket \mathcal{A} \rrbracket(v) = \llbracket \mathcal{B} \rrbracket(v) + \Delta(U(0)), \quad \text{for each word } v = x_{i_1} \cdots x_{i_p} \in X^*, \quad (24)$$

where

$$\Delta(U(0)) = \vec{\delta}_\sigma \cdot M_v^B \cdot \tau^B + \sigma^A \cdot \Delta_v^A \cdot \tau^B - \sigma^A \cdot M_v^A \cdot \vec{\delta}_\tau \quad \text{and} \quad M_v = M_{x_{i_1}} \cdots M_{x_{i_p}}. \quad (25)$$

(b) Assume that the best approximate solution $\mathfrak{U}(t)$ generated by (17) or (19) for chosen initial state $U(0)$ of $U(t)$ satisfies

$$\begin{aligned} \mathfrak{U}(t) \cdot (\sigma^B)^\top &= (\sigma^A)^\top + \vec{\delta}_\sigma^\top, \\ (M_{x_i}^A)^\top \cdot \mathfrak{U}(t) &= \mathfrak{U}(t) \cdot (M_{x_i}^B)^\top + \Delta_{x_i}'^A, \quad i = 1, \dots, s, \\ (\tau^A)^\top \cdot \mathfrak{U}(t) &= (\tau^B)^\top + \vec{\delta}_\tau^\top, \end{aligned} \quad (26)$$

for appropriate vectors $\vec{\delta}_\sigma^\top \in \mathbb{R}^{m \times 1}$ and $\vec{\delta}_\tau^\top \in \mathbb{R}^{1 \times n}$, and appropriate matrices $\Delta_{x_i}'^A \in \mathbb{R}^{m \times m}$, $x_i \in X$, $i = 1, \dots, s$.

Then corresponding ZNNfbb dynamics satisfies

$$\llbracket \mathcal{A} \rrbracket(v) \rightarrow \llbracket \mathcal{B} \rrbracket(v) + \Delta'(U(0)), \text{ for each } v = x_{i_1} \cdots x_{i_p} \in X^*, \quad (27)$$

where

$$\Delta'(U(0)) = \vec{\delta}_\tau \cdot M'_v \cdot (\sigma^B)^\top + (\tau^A)^\top \cdot \Delta'_v \cdot (\sigma^B)^\top - (\tau^A)^\top \cdot M'_v \cdot \vec{\delta}_\tau \quad \text{and} \quad (28)$$

$$M'_v = (M_{x_{i_1}})^\top \cdots (M_{x_{i_p}})^\top. \quad (29)$$

Proof. (a) Based on (23), it is derived

$$\tau^A = \mathfrak{U}(t) \cdot \tau^B - \vec{\delta}_\tau, \quad M_{x_i}^A \cdot \mathfrak{U}(t) = \mathfrak{U}(t) \cdot M_{x_i}^B + \Delta_{x_i}^A, \quad x_1, \dots, x_s \in X, \quad (30)$$

$$\sigma^A \cdot \mathfrak{U}(t) = \sigma^B + \vec{\delta}_\sigma. \quad (31)$$

For arbitrary letter $w = x_{i_1}$ of length $|w| = 1$, based on (30), it follows

$$M_{x_{i_1}}^A \cdot \mathfrak{U}(t) = \mathfrak{U}(t) \cdot M_{x_{i_1}}^B + \Delta_{x_{i_1}}^A = \mathfrak{U}(t) \cdot M_{x_{i_1}}^B + M_\varepsilon^A \Delta_{x_{i_1}}^A \quad (32)$$

Now, let us assume $M_w^A \cdot \mathfrak{U}(t) = \mathfrak{U}(t) \cdot M_w^B + \Delta_w^A$, for arbitrary word $w = x_{i_1} \cdots x_{i_q} \in X^+$. Then for arbitrary letter $x \in X$ it can be concluded

$$\begin{aligned} M_{wx}^A \cdot \mathfrak{U}(t) &= M_w^A \cdot M_x^A \cdot \mathfrak{U}(t) \\ &= M_w^A \cdot (\mathfrak{U}(t) \cdot M_x^B + \Delta_x^A) = M_w^A \cdot \mathfrak{U}(t) \cdot M_x^B + M_w^A \cdot \Delta_x^A \\ &= (\mathfrak{U}(t) \cdot M_w^B + \Delta_w^A) \cdot M_x^B + M_w^A \cdot \Delta_x^A \\ &= \mathfrak{U}(t) \cdot M_w^B \cdot M_x^B + \Delta_w^A \cdot M_x^B + M_w^A \cdot \Delta_x^A \\ &= \mathfrak{U}(t) \cdot M_{wx}^B + \Delta_{wx}^A, \quad \Delta_{wx}^A = \Delta_w^A \cdot M_x^B + M_w^A \cdot \Delta_x^A. \end{aligned} \quad (33)$$

According to the second approximation in (32) and (33) and the principle of mathematical induction, for arbitrary $v_q = x_{i_1} \cdots x_{i_{q-1}} x_{i_q} \in X^+$ and $v_{q-1} = x_{i_1} \cdots x_{i_{q-1}}$ it can be concluded

$$M_{v_q}^A \cdot \mathfrak{U}(t) = \mathfrak{U}(t) \cdot M_{v_q}^B + \Delta_{v_q}^A, \quad \Delta_{v_q}^A = \Delta_{v_{q-1}}^A \cdot M_{x_{i_q}}^B + M_{v_{q-1}}^A \cdot \Delta_{x_{i_q}}^A. \quad (34)$$

The overall conclusion based on (34) is

$$\begin{aligned} \llbracket \mathcal{A} \rrbracket(v_q) &= \sigma^A \cdot M_{v_q}^A \cdot \tau^A \\ &= \sigma^A \cdot M_{v_q}^A \cdot (\mathfrak{U}(t) \cdot \tau^B - \vec{\delta}_\tau) = \sigma^A \cdot (M_{v_q}^A \cdot \mathfrak{U}(t)) \cdot \tau^B - \sigma^A \cdot M_{v_q}^A \cdot \vec{\delta}_\tau \\ &= \sigma^A \cdot (\mathfrak{U}(t) \cdot M_{v_q}^B + \Delta_{v_q}^A) \cdot \tau^B - \sigma^A \cdot M_{v_q}^A \cdot \vec{\delta}_\tau \\ &= (\sigma^A \cdot \mathfrak{U}(t)) \cdot M_{v_q}^B \cdot \tau^B + \sigma^A \cdot \Delta_{v_q}^A \cdot \tau^B - \sigma^A \cdot M_{v_q}^A \cdot \vec{\delta}_\tau \\ &= (\sigma^B + \vec{\delta}_\sigma) \cdot M_{v_q}^B \cdot \tau^B + \sigma^A \cdot \Delta_{v_q}^A \cdot \tau^B - \sigma^A \cdot M_{v_q}^A \cdot \vec{\delta}_\tau \\ &= \sigma^B \cdot M_{v_q}^B \cdot \tau^B + (\vec{\delta}_\sigma \cdot M_{v_q}^B \cdot \tau^B + \sigma^A \cdot \Delta_{v_q}^A \cdot \tau^B - \sigma^A \cdot M_{v_q}^A \cdot \vec{\delta}_\tau) \end{aligned}$$

and further

$$\llbracket \mathcal{A} \rrbracket(v_q) = \llbracket \mathcal{B} \rrbracket(v_q) + \vec{\delta}_\sigma \cdot M_{v_q}^B \cdot \tau^B + \sigma^A \cdot \Delta_{v_q}^A \cdot \tau^B - \sigma^A \cdot M_{v_q}^A \cdot \vec{\delta}_\tau.$$

Therefore, Equation (24) is valid for $\Delta(U(0))$ defined in (25).

(b) This part of the proof follows from part (a) and (22). $\square$

7. Simulink Implementation and Numerical Results

Simulink circuit of the ZNNbfb dynamics (17) is represented in Figure 1.

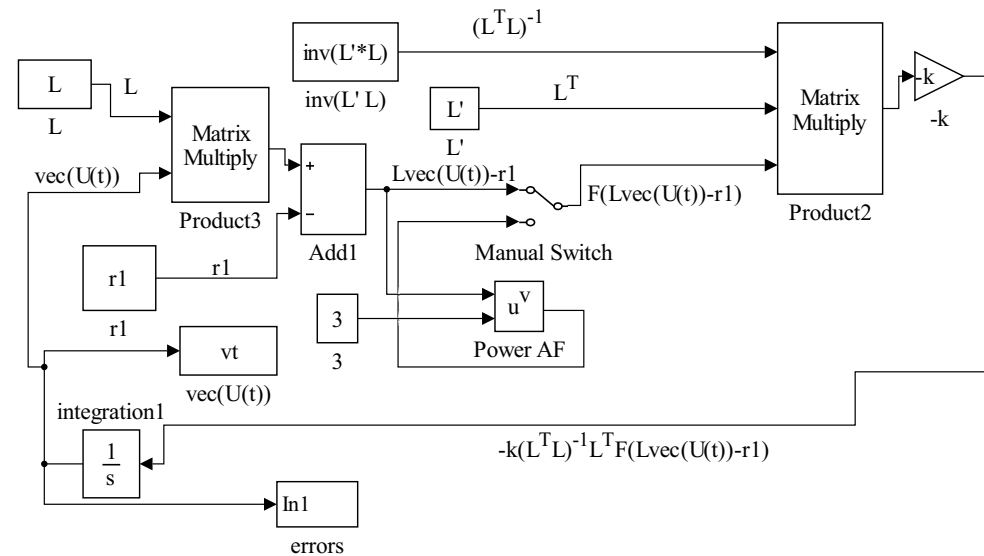


Figure 1. Simulink model of ZNNbfb dynamics (17).

Block In1 involved in Figure 1 is represented in Figure 2.

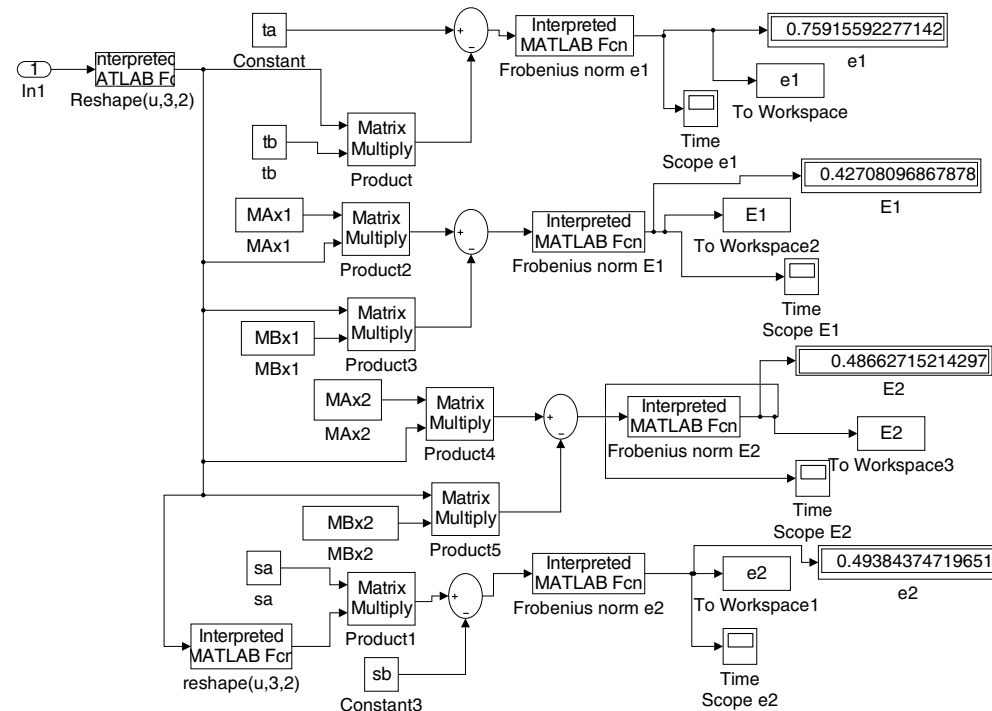


Figure 2. Errors $e1 \equiv \|\vec{\delta}_\tau(t)\|$, $E1 \equiv \|\Delta_{x_1}^A(t)\|$, $E2 \equiv \|\Delta_{x_2}^A(t)\|$, $e2 \equiv \|\vec{\delta}_\sigma(t)\|$ in the block In1 in Simulink model of (17).

All the algorithms were implemented in MATLAB R2017a and executed on a 64-bit HP ZBook laptop (Intel Core i7-4710Q CPU @ 2.50 GHz, RAM 16 GB) with the Windows 10 operating system.

Metrics used to evaluate usefulness of generated solutions will be expressed in norms of the error functions

$$\begin{aligned} \|e_1(t)\| &\equiv \|\vec{\delta}_\tau(t)\|_2 = \|\mathfrak{U}(t) \cdot \tau^B - \tau^A\|_2, \quad \|e_2(t)\| \equiv \|\vec{\delta}_\sigma(t)\|_2 = \|\sigma^A \cdot \mathfrak{U}(t) - \sigma^B\|_2, \\ \|\mathfrak{E}_i(t)\| &\equiv \|\Delta_{x_i}^A(t)\|_F = \|M_{x_i}^A \cdot \mathfrak{U}(t) - \mathfrak{U}(t) \cdot M_{x_i}^B\|_F, \quad i = 1, \dots, s. \end{aligned} \quad (35)$$

The exact ideal theoretical values for these matrix norms are equal to 0. Therefore, the obtained matrix and vector norms generated inside ZNNbfb are deviation of the obtained ZNN approximate solution $\mathfrak{U}(t)$ from the ideal solution $U(t)$ that leads to zero theoretical values.

Example 3. (a) For the input data in Example 1, values of the gain parameter $\kappa = 10, 100, 1000$ and the zero initial condition $U(0) = 0_{3 \times 2}$, the Simulink implementation of the ZNNbfb dynamics from Figure 1 gives results described as follows. For $\kappa = 10$ in the time interval $[0, 3]$, the output vector in the final time instant $t_f = 3$ is equal to

$$\text{vec}(\mathfrak{U}(3)) = [0.372029, 0.656478, 0.453665, -0.079884, -0.259506, -0.070052]^T$$

which implies

$$\mathfrak{U}(3) = \text{reshape}(\text{vec}(\mathfrak{U}(3)), 3, 2) = \begin{bmatrix} 0.372029 & -0.079884 \\ 0.656478 & -0.259506 \\ 0.453665 & -0.070052 \end{bmatrix}. \quad (36)$$

The trajectories of elements involved in $\mathfrak{U}(t)$ in the ZNNbfb design corresponding to different values of the parameter k and the Linear (resp. Power) AF are presented in Figure 3a (resp. Figure 3b). Plots represented by solid lines are generated using $\kappa = 10$, while $\kappa = 100$ is used in generating dashed plots.

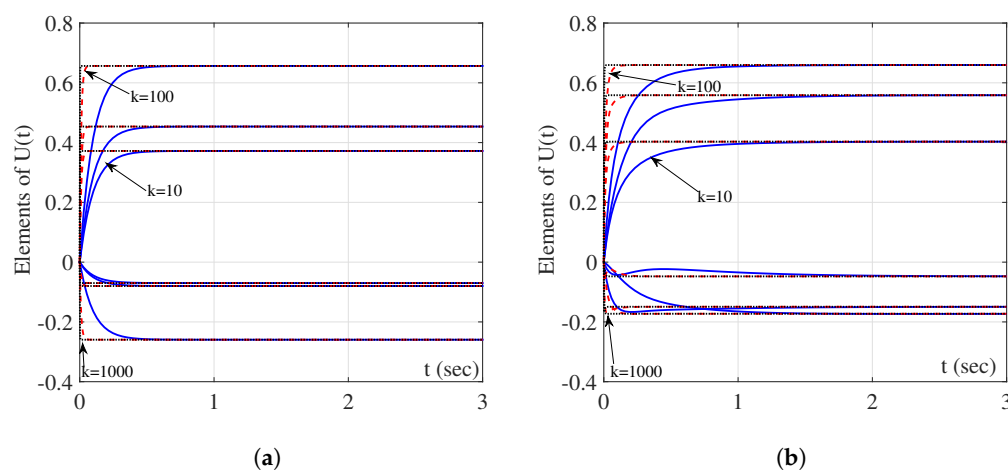


Figure 3. Elements of $\mathfrak{U}(t)$ in ZNNbfb for different values of the parameter k and different AFs. (a) Linear activation function. (b) Power activation function.

The norms of error functions in (35) for different values of k and in linear ZNNbfb are presented in Figure 4.

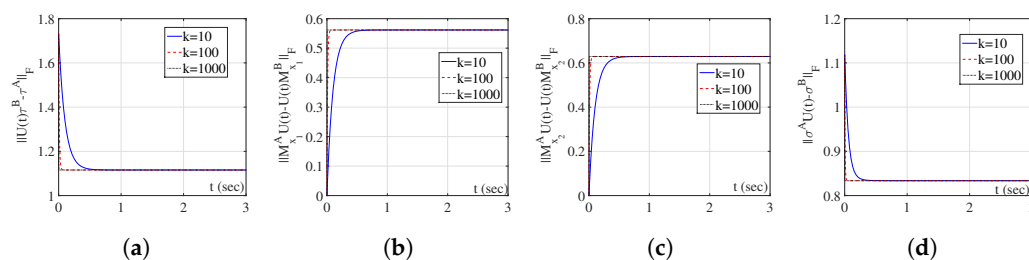


Figure 4. Errors for different values of κ in linear ZNNbfb model. (a) $\|\vec{\delta}_\tau(t)\|_2$. (b) $\|\Delta_{x_1}^A(t)\|_F$. (c) $\|\Delta_{x_2}^A(t)\|_F$. (d) $\|\vec{\sigma}_\tau(t)\|_2$.

Errors for different values of κ in ZNNbfb and with the application of Power activation function are shown in Figure 5.

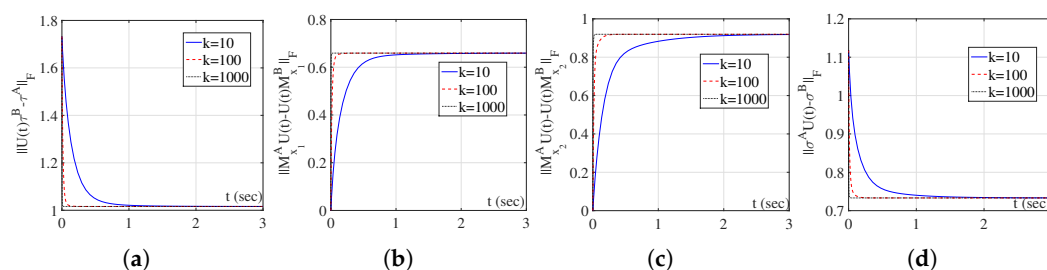


Figure 5. Errors for different values of κ and with application of Power activation function in ZNNbfb. (a) $\|\vec{\delta}_\tau(t)\|_2$. (b) $\|\Delta_{x_1}^A(t)\|_F$. (c) $\|\Delta_{x_2}^A(t)\|_F$. (d) $\|\vec{\sigma}_\tau(t)\|_2$.

Elements of the matrix $\mathfrak{U}(t)$ for different initial conditions $U(0) = \text{rand}() + 1000 \cdot \text{rand}() \cdot \text{rand}(6,1)$ and the gain parameter values $\kappa = 10, 100, 1000$ in ZNNbfb under Linear AF are arranged in Figure 6.

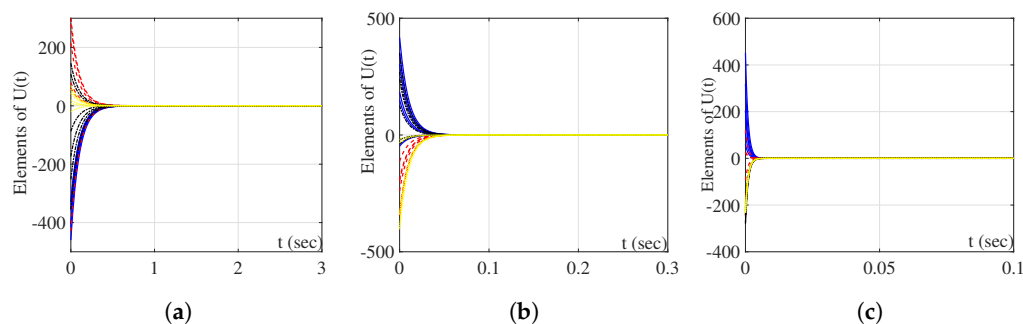


Figure 6. Elements of $\mathfrak{U}(t)$ for different initial conditions and Linear activation function. (a) $k = 10$. (b) $k = 100$. (c) $k = 1000$.

(b) Our goal in this data analysis is to continue the numerical experience on this test example analyzing the error vectors and matrices that appear in (23). The following approximations are obtained in the final stage $t = 0.7$ under $\kappa = 100$ and within the time interval $[0, 0.7]$:

$$\vec{\delta}_\tau(0.7) = \begin{bmatrix} -0.707852 & -0.603028 & -0.616387 \end{bmatrix}^T, \quad \vec{\delta}_\sigma(0.7) = \begin{bmatrix} -0.343522 & -0.759506 \end{bmatrix}$$

$$\Delta_{x_1}^A(0.7) = \begin{bmatrix} 0.153825 & 0.0424703 \\ -0.340278 & 0.326919 \\ 0.255338 & -0.045684 \end{bmatrix}, \quad \Delta_{x_2}^A(0.7) = \begin{bmatrix} -0.235030 & 0.093364 \\ -0.185166 & -0.372313 \\ -0.372313 & 0.3984175 \end{bmatrix},$$

with norms equal to

$$\|\vec{\delta}_\tau(0.7)\| = 1.115632, \|\vec{\delta}_\tau(0.7)\| = 0.833581, \|\Delta_{x_1}^A(0.7)\| = 0.561619, \|\Delta_{x_2}^A(0.7)\| = 0.628997.$$

The state variables matrix in the final stage satisfies $\mathfrak{U}(0.7) \approx \mathfrak{U}(3)$, where $\mathfrak{U}(3)$ is defined in (36).

(c) The goal in this part of example is investigation of various initial stages $\mathfrak{U}(0)$ to the final solution. To this end, we generate a series of random matrices

$$\mathfrak{U}(0) = 5000 * \text{rand}() + 10000 * \text{rand}() * \text{rand}(6, 1),$$

change time intervals $[0, t_f]$ and run the Simulink. In all test cases, $\mathfrak{U}(t_f)$ is equal to the matrix in (36). The conclusion is that the ZNN model generates the same solution for all values of parameters, regardless of the choice of the starting point $\mathfrak{U}(0)$.

Example 4. It will be interesting to examine the behavior of ZNNfbb on the same input data. For $\kappa = 10$ in the time interval $[0, 3]$, the output vector generated by ZNNfbb in the last stage $t = 3$ is equal to

$$\text{vec}(\mathfrak{U}(t)) = [-0.174144, 0.459668, 0.394089, 0.363006, -0.0079437, 0.270226]^T$$

which implies

$$\mathfrak{U}(t) = \begin{bmatrix} -0.174144 & 0.459668 \\ 0.394089 & 0.363006 \\ -0.0079437 & 0.270226 \end{bmatrix}.$$

The elements involved in the matrix $\mathfrak{U}(t)$ for different values of κ , the zero initial state $\mathfrak{U}(0) = 0$, and Linear or Power AFs are plotted in Figure 7. Further, the norms defined in (35) related to the dynamic flow with the zero initial state are plotted for different values of κ and using the Linear AF are traced in Figure 8, while in Figure 9 the same errors are shown with the application of Power AF.

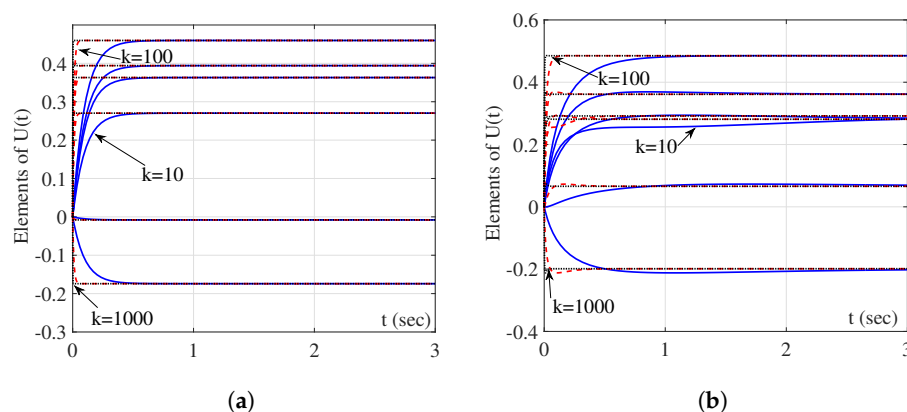


Figure 7. Elements of $\mathfrak{U}(t)$ for different values of the parameter κ and two AFs. (a) Linear activation function. (b) Power activation function.

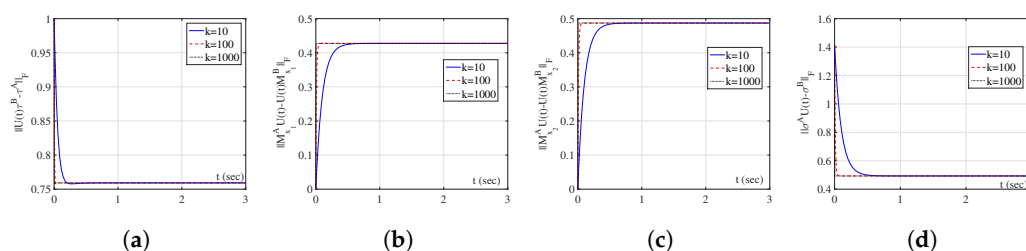


Figure 8. Errors for different values of parameter κ and with application of Linear activation function. (a) $\|\vec{\delta}_\tau(t)\|_2$. (b) $\|\Delta_{x_1}^A(t)\|_F$. (c) $\|\Delta_{x_2}^A(t)\|_F$. (d) $\|\vec{\delta}_\sigma(t)\|_2$.

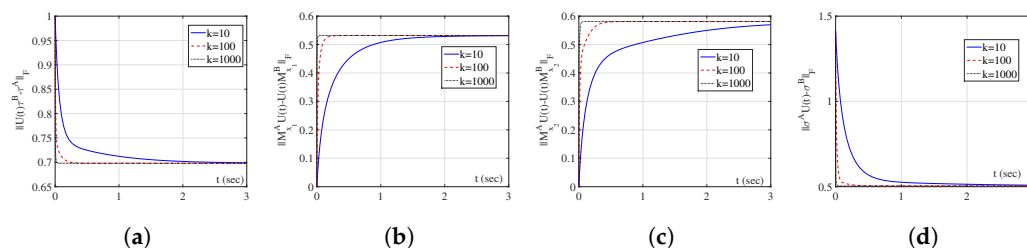


Figure 9. Errors for different values of parameter k and with application of Power activation function. (a) $\|\vec{\delta}_\tau(t)\|_2$. (b) $\|\Delta_{x_1}^A(t)\|_F$. (c) $\|\Delta_{x_2}^A(t)\|_F$. (d) $\|\vec{\delta}_\sigma(t)\|_2$.

Furthermore, the trajectories of elements in $\mathfrak{U}(t)$ for different initial conditions generated by the rule $U(0)=500*\text{rand}()+1000*\text{rand}()*\text{rand}(6,1)$, corresponding to $\kappa = 10, 100, 1000$, and under the Linear AF are arranged in Figure 10.

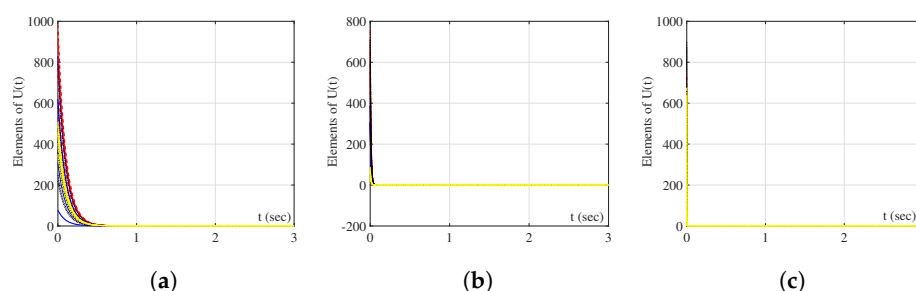


Figure 10. Elements of the matrix $U(t)$ for different initial states and Linear activation function. (a) $k = 10$. (b) $k = 100$. (c) $k = 1000$.

Figure 11 and Figure 12 show errors $\|\vec{\delta}_\tau(t)\|_2$, $\|\Delta_{x_1}^A(t)\|_F$, $\|\Delta_{x_2}^A(t)\|_F$ and $\|\vec{\delta}_\sigma(t)\|_2$ produced by ZNN_{bfb} and ZNN_{fbb} formulas with the application of Linear and Power AFs, respectively, and $\kappa = 100$. Figures 13 and 14 show the same errors produced from ZNN_{bfb} and ZNN_{fbb} with the same AFs for $\kappa = 1000$. It is observable that the corresponding norms in (35) are smaller when ZNN_{fbb} (red dashed lines) is applied compared to ZNN_{bfb} (blue solid lines). Such a conclusion reinforces the advantage of using ZNN_{fbb} against ZNN_{bfb} .

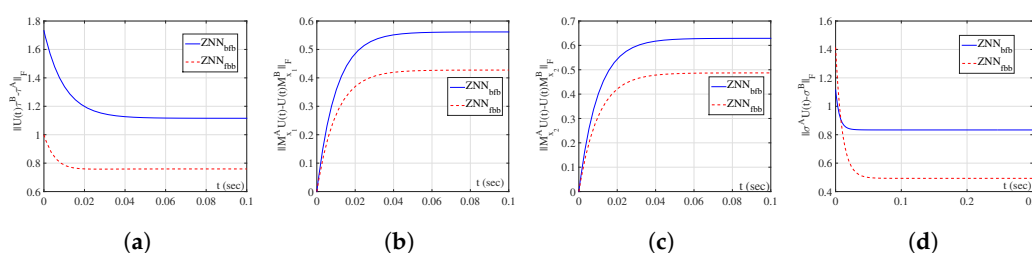


Figure 11. Comparison of the errors between ZNN_{bfb} and ZNN_{fbb} for $\kappa = 100$ and with application of the Linear activation function. (a) $\|\vec{\delta}_\tau(t)\|_2$. (b) $\|\Delta_{x_1}^A(t)\|_F$. (c) $\|\Delta_{x_2}^A(t)\|_F$. (d) $\|\vec{\delta}_\sigma(t)\|_2$.

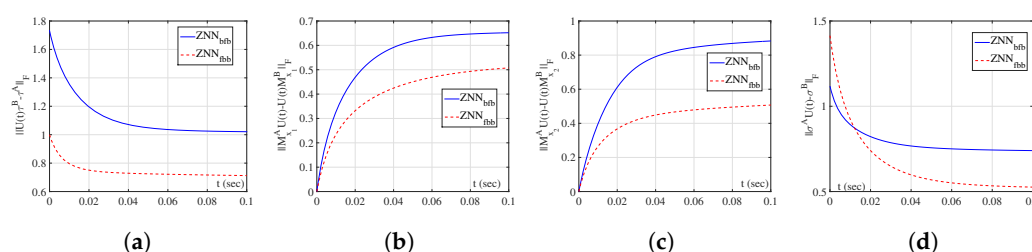


Figure 12. Comparison of the errors between ZNN_{bfb} and ZNN_{fbb} for $\kappa = 100$ and with application of the Power activation function. (a) $\|\vec{\delta}_\tau(t)\|_2$. (b) $\|\Delta_{x_1}^A(t)\|_F$. (c) $\|\Delta_{x_2}^A(t)\|_F$. (d) $\|\vec{\delta}_\sigma(t)\|_2$.

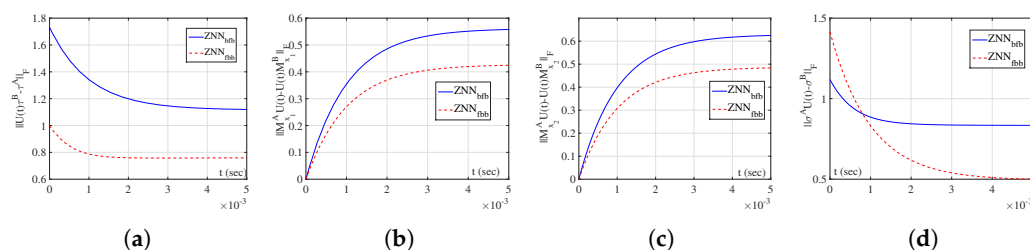


Figure 13. Comparison of the errors between ZNN_{bfb} and ZNN_{fbb} for $k = 1000$ and with application of the Linear activation function. (a) $\|\vec{\delta}_\tau(t)\|_2$. (b) $\|\Delta_{x_1}^A(t)\|_F$. (c) $\|\Delta_{x_2}^A(t)\|_F$. (d) $\|\vec{\delta}_\sigma(t)\|_2$.

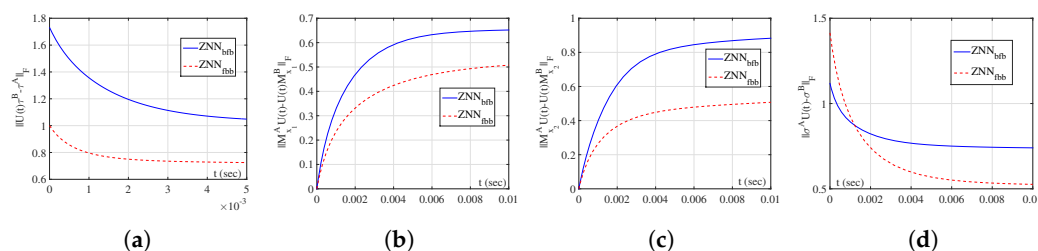


Figure 14. Comparison of the errors between ZNN_{bfb} and ZNN_{fbb} for $k = 1000$ and with application of the Power activation function. (a) $\|\vec{\delta}_\tau(t)\|_2$. (b) $\|\Delta_{x_1}^A(t)\|_F$. (c) $\|\Delta_{x_2}^A(t)\|_F$. (d) $\|\vec{\delta}_\sigma(t)\|_2$.

Example 5. This example is aimed to comparison between ZNN_{bfb} and ZNN_{fbb} on randomly generated test matrices of various dimensions. Linear (resp. Power sigmoid) activation will be denoted shortly by “/L” (resp. “/P”). The norms defined in (35) will be considered. For $\omega \in \{\text{bfb}, \text{fbb}\}$ denote by $\omega(\|e_i\|)$ the value $\|e_i\|$ generated by ZNN_i , for each $i = 1, 2$. Similarly, $\omega(\|\Delta_{x_i}^A\|)$ will denote the value $\|\Delta_{x_i}^A\|$ generated by ZNN_i , for each $i = 1, \dots, s$. The notation $\omega_1(\|e_i\|) \leq \omega_2(\|e_i\|)$, for $\omega_1, \omega_2 \in \{\text{bfb}, \text{fbb}\}$ means the situation $\text{ZNN}_{i_1}(\|e_i\|) \leq \text{ZNN}_{i_2}(\|e_i\|)$ in the majority of test cases or in all cases, for $i = 1, 2$, whereas $\omega_1(\|\Delta_{x_i}^A\|) \leq \omega_2(\|\Delta_{x_i}^A\|)$ denotes the situation $\text{ZNN}_{i_1}(\|\Delta_{x_i}^A\|) \leq \text{ZNN}_{i_2}(\|\Delta_{x_i}^A\|)$ in the majority of test cases, for $i = 1, \dots, s$. Minimal values between values $\omega_1(\|e_i\|)$, $i = 1, 2$ and $\omega(\|\Delta_{x_i}^A\|)$, $i = 1, \dots, s$ generated after the completion of ZNN_{bfb} and ZNN_{fbb} are marked in bold.

Results of bisimulations on randomly generated test matrices in the case $m, n \in \{20, 30\}$, $\kappa = 1000$, $s = 2$ are arranged in Table 3.

Table 3. Results on random test matrices for $m, n \in \{20, 30\}$ and $s = 2$.

Model/AF	(n, m)	$\ \vec{\delta}_\tau(t)\ _2$	$\ \Delta_{x_1}^A(t)\ _F$	$\ \Delta_{x_2}^A(t)\ _F$	$\ \vec{\delta}_\sigma(t)\ _2$
$\text{ZNN}_{\text{bfb}}/\text{L}$	(20, 30)	2.504	1.182	1.407	2.478
$\text{ZNN}_{\text{fbb}}/\text{L}$	(20, 30)	2.21	1.154	1.32	2.066
$\text{ZNN}_{\text{bfb}}/\text{P}$	(20, 30)	1.494	3.165	3.57	1.484
$\text{ZNN}_{\text{fbb}}/\text{P}$	(20, 30)	1.33	2.793	3.088	1.161
$\text{ZNN}_{\text{bfb}}/\text{L}$	(30, 30)	1.315	1.351	1.293	1.136
$\text{ZNN}_{\text{fbb}}/\text{L}$	(30, 30)	1.242	1.222	1.225	1.063
$\text{ZNN}_{\text{bfb}}/\text{P}$	(30, 30)	0.7616	2.018	1.963	0.7223
$\text{ZNN}_{\text{fbb}}/\text{P}$	(30, 30)	0.7479	1.876	1.835	0.6402
$\text{ZNN}_{\text{bfb}}/\text{L}$	(30, 20)	1.536	1.113	1.045	2.006
$\text{ZNN}_{\text{fbb}}/\text{L}$	(30, 20)	2.061	1.194	1.129	2.794
$\text{ZNN}_{\text{bfb}}/\text{P}$	(30, 20)	0.7906	2.55	2.463	1.248
$\text{ZNN}_{\text{fbb}}/\text{P}$	(30, 20)	1.158	3.284	3.187	1.896

According to data arranged in Table 3 it can be concluded $\text{fbb}(\|e_i\|) \leq \text{bfb}(\|e_i\|)$, $i = 1, 2$ if $n \leq m$ and $\text{bfb}(\|e_i\|) \leq \text{fbb}(\|e_i\|)$, $i = 1, 2$ if $n > m$ as well as $\text{fbb}(\|\Delta_{x_i}^A\|) \leq \text{bfb}(\|\Delta_{x_i}^A\|)$ if $n \leq m$ and $\text{bfb}(\|\Delta_{x_i}^A\|) \leq \text{fbb}(\|\Delta_{x_i}^A\|)$ if $n > m$, for each $i = 1, 2$.

Results of bisimulations on randomly generated test matrices in the case $m, n \in \{40, 50\}$, $\kappa = 1000$, $s = 2$ are organized in Table 4.

Table 4. Results on random test matrices for $m, n \in \{40, 50\}$ and $s = 2$.

Model/AF	(n, m)	$\ \vec{\delta}_\tau(t)\ _2$	$\ \Delta_{x_1}^A(t)\ _F$	$\ \Delta_{x_2}^A(t)\ _F$	$\ \vec{\delta}_\sigma(t)\ _2$
ZNN _{bfb} /L	(40, 50)	2.844	1.646	1.656	2.696
ZNN _{fbb} /L	(40, 50)	2.988	1.627	1.624	2.429
ZNN _{bfb} /P	(40, 50)	1.508	4.104	4.133	1.36
ZNN _{fbb} /P	(40, 50)	1.555	4.088	4.029	1.253
ZNN _{bfb} /L	(50, 50)	1.702	1.706	1.698	1.372
ZNN _{fbb} /L	(50, 50)	1.838	1.614	1.716	1.297
ZNN _{bfb} /P	(50, 50)	0.887	2.7	2.656	0.732
ZNN _{fbb} /P	(50, 50)	0.9326	2.643	2.712	0.7774
ZNN _{bfb} /L	(50, 40)	2.083	1.761	1.613	3.174
ZNN _{fbb} /L	(50, 40)	2.364	1.75	1.55	2.919
ZNN _{bfb} /P	(50, 40)	0.9549	4.168	3.899	1.757
ZNN _{fbb} /P	(50, 40)	1.088	4.144	3.854	1.664

Data arranged in Table 4 lead to two opposite conclusions: $\text{bfb}(\|\vec{\delta}_\tau\|) \leq \text{fbb}(\|\vec{\delta}_\tau\|)$ and $\text{fbb} \leq \text{bfb}$ for other norms $\|\vec{\delta}_\sigma\|$, $\|\Delta_{x_1}^A\|$, $\|\Delta_{x_2}^A\|$.

Results of bisimulations on randomly generated test matrices in the case $m, n \in \{60, 70\}$, $\kappa = 1000$ are arranged for $s = 4$ and $s = 6$ transition matrices in Table 5 and Table 6, respectively.

Table 5. Results on random test matrices for $m, n \in \{60, 70\}$ and $s = 4$.

Model/AF	(n, m)	$\ \vec{\delta}_\tau(t)\ _2$	$\ \Delta_{x_1}^A(t)\ _F$	$\ \Delta_{x_2}^A(t)\ _F$	$\ \Delta_{x_3}^A(t)\ _F$	$\ \Delta_{x_4}^A(t)\ _F$	$\ \vec{\delta}_\sigma(t)\ _2$
ZNN _{bfb} /L	(60, 70)	4.527	1.444	1.325	1.332	1.343	3.387
ZNN _{fbb} /L	(60, 70)	4.21	1.411	1.287	1.262	1.309	3.307
ZNN _{bfb} /P	(60, 70)	2.275	5.015	4.676	4.69	4.786	1.474
ZNN _{fbb} /P	(60, 70)	2.123	4.792	4.492	4.411	4.576	1.35
ZNN _{bfb} /L	(70, 70)	2.737	1.447	1.447	1.657	1.454	2.738
ZNN _{fbb} /L	(70, 70)	2.798	1.582	1.608	1.583	1.526	3.014
ZNN _{bfb} /P	(70, 70)	1.226	3.312	3.245	3.457	3.3	1.24
ZNN _{fbb} /P	(70, 70)	1.24	3.539	3.537	3.561	3.466	1.376
ZNN _{bfb} /L	(70, 60)	4.138	1.334	1.311	1.292	1.519	3.842
ZNN _{fbb} /L	(70, 60)	3.68	1.288	1.309	1.264	1.483	1.483
ZNN _{bfb} /P	(70, 60)	1.887	4.647	4.66	4.601	5.183	1.791
ZNN _{fbb} /P	(70, 60)	1.687	4.335	4.412	4.29	4.813	1.693

Error norms contained in Table 5 lead to the conclusion $\text{fbb} \leq \text{bfb}$ if $n \neq m$ and $\text{bfb} \leq \text{fbb}$ if $n = m$.

Table 6. Results on random test matrices for $m, n \in \{60, 70\}$ and $s = 6$.

Model/AF	(n, m)	$\ \vec{\delta}_\tau(t)\ _2$	$\ \Delta_{x_1}^A(t)\ _F$	$\ \Delta_{x_2}^A(t)\ _F$	$\ \Delta_{x_3}^A(t)\ _F$	$\ \Delta_{x_4}^A(t)\ _F$	$\ \Delta_{x_5}^A(t)\ _F$	$\ \Delta_{x_6}^A(t)\ _F$	$\ \vec{\delta}_\sigma(t)\ _2$
ZNN _{bfb} /L	(60, 70)	4.429	0.813	0.8174	0.8295	0.8925	0.8608	0.8358	3.86
ZNN _{fb} /L	(60, 70)	4.167	0.8265	0.8233	0.8098	0.8771	0.823	0.8273	3.236
ZNN _{bfb} /P	(60, 70)	2.431	4.393	4.445	4.427	4.729	4.606	3.992	1.686
ZNN _{fb} /P	(60, 70)	2.104	4.04	4.096	3.962	4.192	4.018	3.995	1.369
ZNN _{bfb} /L	(70, 60)	3.459	0.9036	0.8988	0.9066	0.8697	1.002	0.9155	4.707
ZNN _{fb} /L	(70, 60)	3.827	0.883	0.9389	0.9323	0.8418	1	0.908	4.365
ZNN _{bfb} /P	(70, 60)	1.562	4.341	4.326	4.36	4.242	4.733	4.407	2.432
ZNN _{fb} /P	(70, 60)	1.663	4.223	4.404	4.391	3.996	4.57	4.248	2.365

Data arranged in Table 6 confirm the relationship $\text{fbf}(\|\vec{\delta}_\tau\|) \leq \text{bfb}(\|\vec{\delta}_\tau\|)$ in most cases.

Based on numerical data involved in all tables, the clear conclusion is $\text{bfb}(\|\vec{\delta}_\tau\|) \leq 4.527$, $\text{fbf}(\|\vec{\delta}_\tau\|) \leq 4.167$; $\text{bfb}(\|\Delta_{x_1}^A\|) \leq 5.015$, $\text{fbf}(\|\Delta_{x_1}^A\|) \leq 4.792$; $\text{bfb}(\|\Delta_{x_2}^A\|) \leq 4.676$, $\text{fbf}(\|\Delta_{x_2}^A\|) \leq 4.492$; $\text{bfb}(\|\vec{\delta}_\sigma\|) \leq 4.707$, $\text{fbf}(\|\vec{\delta}_\sigma\|) \leq 4.365$.

Tables 5 and 6 give $\text{bfb}(\|\Delta_{x_3}^A\|) \leq 4.69$, $\text{fbf}(\|\Delta_{x_3}^A\|) \leq 4.411$; $\text{bfb}(\|\Delta_{x_4}^A\|) \leq 5.183$, $\text{fbf}(\|\Delta_{x_4}^A\|) \leq 4.813$.

Finally, Table 6 confirms $\text{bfb}(\|\Delta_{x_5}^A\|) \leq 4.733$, $\text{fbf}(\|\Delta_{x_5}^A\|) \leq 4.57$; $\text{bfb}(\|\Delta_{x_6}^A\|) \leq 4.407$, $\text{fbf}(\|\Delta_{x_6}^A\|) \leq 4.248$.

Accordingly, overall conclusion is that ZNN_{bfb} and ZNN_{fb} give approximate solutions to error functions included in (13).

7.1. Computational Performance and CPU Execution Times

Numerical simulations were conducted in a standardized environment to assess the practical efficiency of the proposed continuous-time ZNN models (ZNN_{fb} and ZNN_{bfb}) for approximating heterotypic bisimulations. All algorithms were implemented in MATLAB R2017a and Simulink and executed on an HP ZBook laptop with an Intel Core i7-4710Q CPU at 2.50GHz, 16GB RAM, and a 64-bit Windows 10 operating system.

This computational assessment aims to compare the online tracking efficiency of the neurodynamic approach with traditional offline matrix decomposition techniques. The evaluation encompasses the three principal numerical test setups investigated in this study:

- (1) Tests 7.1 and 7.2 (Small-Scale Base Case) employ the structured parameters defined in Example 1. In these cases, the target WFAs $\mathcal{A}$ and $\mathcal{B}$ are of dimensions $m = 3$ and $n = 2$, respectively, and operate over a binary alphabet ($s = 2$).
- (2) Test 7.3 (Medium-Scale Synthetic Benchmarks): This example is formulated using randomly generated transition matrices with dimensions $m, n \in \{20, 30\}$ to systematically compare error thresholds and performance boundaries between the ZNN_{fb} and ZNN_{bfb} frameworks.
- (3) Large-scale tests: Theoretical scaling boundaries are assessed for systems where $m, n \geq 100$. In these cases, explicit pseudoinverse formulations result in substantial computational bottlenecks. Therefore, low-rank algebraic methods, such as Truncated Randomized SVD, are favored.

Obtained results are arranged in Table 7.

Table 7. CPU execution times for approximating heterotypic WFA bisimulations.

Examples	$\mathcal{A}$ Dim (m)	$\mathcal{B}$ Dim (n)	κ	AF&Model Variant	Avg. CPU Time Range
Tests 7.1,7.2	3	2	10	Linear(ZNN_{bfb}/ZNN_{fbb})	0.05–0.12s
Tests 7.1,7.2	3	2	100	Power AF (ZNN_{bfb}/ZNN_{fbb})	0.08–0.15s
Tests 7.1,7.2	3	2	100	Linear (ZNN_{bfb}/ZNN_{fbb})	0.15–0.25s
Test 7.3 (Random)	20	20	1000	Linear/Power AF (ZNN_{bfb})	1.20–2.50s
Test 7.3 (Random)	30	20	1000	Linear/Power AF (ZNN_{fbb})	2.10–4.30s
Test 7.3 (Random)	30	30	1000	Power AF (ZNN_{fbb})	3.50–6.80s
Large-Scale Test	100	100	1000	Power AF (Standard Block Matrix)	45.0–75.0s

7.2. Discussion

Standard algebraic methods, such as least-squares, static pseudoinverse, and regularized least-squares, yield optimal solutions for static problems; however, they lack the continuous-time, online adaptation required for time-varying scenarios. The proposed ZNN model operates as a continuous dynamic system and addresses this limitation by ensuring global exponential convergence in real-time environments. This approach offers a significant computational advantage over traditional offline optimization techniques.

In numerical linear algebra, constructing the normal system increases the condition number of the underlying matrix, thereby amplifying numerical instability in conventional discrete solvers. In contrast, the proposed ZNN framework employs a solver that operates as a continuous-time dynamical system. The continuous integration and global exponential convergence characteristics of ZNN neurodynamics reduce the impact of ill-conditioning and ensure stable trajectories toward the least-squares solution.

8. Concluding Remarks

The existence and fundamental attributes of bisimulations for WFA over real numbers are investigated. Forward-backward bisimulation (fbb) and backward-forward bisimulation (bfb) between WFA $\mathcal{A}$ and $\mathcal{B}$ are defined as solutions to two vector equations and a certain system of homogeneous Sylvester equations whose coefficient matrices in these equations are determined by the transition matrices of $\mathcal{A}$ and $\mathcal{B}$. Thus, the original problem is reformulated as a matrix algebra problem. The proposed approach frames the study as a transition from symbolic structural analysis (bisimulations) to numerical dynamical systems (ZNN). The principle based on matrix/vector equalities have successfully mapped the algebraic properties of WFA (containment and equivalence) to a system of Sylvester and vector equations. This makes the problem accessible to numerical optimization. The core challenge identified is that these systems are “generally inconsistent.” Standard algebraic methods (like Gaussian elimination) would simply fail. By using ZNN to find an approximate solution, you are essentially seeking a “least-squares-like” equilibrium in a dynamical system. This research aims to apply the continuous-time dynamics of the zeroing neural network (ZNN) to obtain approximate solutions for these vector-matrix systems. The general problem studied in this research is specific because there is no solution that leads all observed error functions to a common zero. Consequently, the ZNN framework appears promising for addressing this type of problem. Empirical results confirm that ZNN provides approximate solutions to the observed matrix-vector equations, consistent with initial expectations. A detailed convergence analysis is conducted, focusing on the distribution of non-zero values in individual error functions resulting from collective convergence. Numerical experiments are conducted using various values and numbers

of transition matrices, as well as different initial state matrices. The application of ZNN dynamics yields approximate solutions in a time-varying form.

Numerical results indicate that outcomes produced by multi-objective optimization (MOO) methods are highly sensitive to the selection of weights or the specified priority order by the decision maker. In contrast, the ZNN dynamics are stable and produce a single solution. The distinction between ZNN and multi-objective optimization (MOO) is vital. While MOO requires subjective weighting (the “Scalarization Problem”), ZNN offers a deterministic, stable trajectory toward a single solution by integrating all Error Functions (EFs) into a composite block matrix.

Future research may consider the application of models characterized by finite and predefined convergence times or focus on developing algorithms that leverage the sparsity of matrices present in defined ZNN dynamics, as noted in Remark 2. Another avenue for future research involves addressing simulation problems formulated with inequality constraints. This study represents the initial application of ZNN design to the resolution of intractable problems. Previous research has demonstrated that ZNN can generate approximate solutions through the use of the matrix pseudoinverse. This finding suggests the potential for applying ZNN dynamics to additional linear algebra problems that lack exact solutions. Another potential direction for future research is to conceptualize the bisimulation problem as a multicriteria optimization problem.

In addition, future work will be oriented to integrating low-rank matrix approximation techniques, such as the efficient truncated randomized SVD [55], into the ZNN error function design to bypass direct pseudoinverse formulations and significantly reduce the computational footprint when dealing with massive, ill-conditioned coupled systems related to WFAs.

Author Contributions: Conceptualization, P.S., M.Ć. and D.G.; methodology, P.S., M.Ć. and D.K.; software, D.G. and P.S.; validation, M.Ć. and P.S.; formal analysis, M.Ć., P.S., D.G. and D.K.; investigation, P.S., M.Ć. and D.K.; resources, D.G.; data curation, D.G. and P.S.; writing—original draft preparation, P.S. and M.Ć.; writing—review and editing, P.S., M.Ć. and D.K.; visualization, D.G.; supervision, M.Ć. and P.S.; project administration, P.S. and M.Ć.; funding acquisition, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science, Technological Development and Innovation, Serbia, Contract No. 451-03-34/2026-03/200124.

Data Availability Statement: Data are contained within the article. Simulink implementation of the proposed ZNN models is presented in Figures 1 and 2. Input data are defined in presented numerical simulations. The data that support the findings of this study are available on request to the authors.

Acknowledgments: Predrag Stanimirović and Miroslav Ćirić are supported by the Ministry of Education, Science and Technological Development, Serbia, Contract No. 451-03-34/2026-03/200124. Predrag Stanimirović is also supported by the Ministry of Science and Technology of China under the grant H20240841. Dimitrios Gerontitis is supported by financial support of the “Savas Parastatidis” named scholarship. The authors thank the reviewers for their valuable comments, which have significantly improved the quality of the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ćirić, M.; Ignjatović, J.; Damljanović, N.; Bašić, M. Bisimulations for fuzzy automata. *Fuzzy Sets Syst.* **2012**, *186*, 100–139.
2. Ćirić, M.; Ignjatović, J.; Jančić, I.; Damljanović, N. Computation of the greatest simulations and bisimulations between fuzzy automata. *Fuzzy Sets Syst.* **2012**, *208*, 22–42.

3. Ćirić, M.; Ignjatović, J.; Bašić, M.; Jančić, I. Nondeterministic automata: Equivalence, bisimulations, and uniform relations. *Inf. Sci.* **2014**, *261*, 185–218.
4. Damjanović, N.; Ćirić, M.; Ignjatović, J. Bisimulations for weighted automata over an additively idempotent semiring. *Theor. Comput. Sci.* **2014**, *534*, 86–100.
5. Ćirić, M.; Micić, I.; Matejić, J.; Stamenković, A. Simulations and bisimulations for max-plus automata. *Discret. Event Dyn. Syst.* **2024**, *34*, 269–295.
6. Stanković, I.; Ćirić, M.; Ignjatović, J. Bisimulations for weighted networks with weights in a quantale. *Filomat* **2023**, *37*, 3335–3355.
7. Ćirić, M.; Ignjatović, J.; Stanimirović, P.S. Bisimulations for weighted finite automata over semirings. *Res. Sq.* **2022**, preprint. <https://doi.org/10.21203/rs.3.rs-2386298/v1>.
8. Urabe, N.; Hasuo, I. Quantitative simulations by matrices. *Inf. Comput.* **2017**, *252*, 110–137.
9. Matejić, J.; Ćirić, M.; Ignjatović, J.; Micić, I. Truncating and shifting weights for max-plus automata. *Axioms* **2026**, *15*, 79.
10. Stanković, M.; Ćirić, M.; Ignjatović, J. Hennessy-Milner type theorems for fuzzy multimodal logics over Heyting algebras. *J. Mult.-Valued Log. Soft Comput.* **2022**, *39*, 341–379.
11. Stanković, M.; Ćirić, M.; Ignjatović, J. Simulations and bisimulations for fuzzy multimodal logics over Heyting algebras. *Filomat* **2023**, *37*, 711–743.
12. Doyen, L.; Henzinger, T.A.; Raskin, J.-F. Equivalence of labeled Markov chains. *Int. J. Found. Comput. Sci.* **2008**, *19*, 549–563.
13. Balle, B.; Gourdeau, P.; Panangaden, P. Bisimulation metrics and norms for real weighted automata. *Inf. Comput.* **2022**, *282*, 104649.
14. Boreale, M. Weighted bisimulation in linear algebraic form. In *CONCUR 2009—Concurrency Theory*; Bravetti, M., Zavattaro, G., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5710, pp. 163–177.
15. Wang, T.; Zhang, Z.; Huang, Y.; Liao, B.; Li, S. Applications of zeroing neural networks: A survey. *IEEE Access* **2024**, *12*, 51346–51363.
16. Zhang, Y.; Yi, C. *Zhang Neural Networks and Neural-Dynamic Method*; Nova Science Publishers, Inc.: New York, NY, USA, 2011.
17. Han, L.; He, Y.; Liao, B.; Hua, C. An accelerated double-integral ZNN with resisting linear noise for dynamic Sylvester equation solving and its application to the control of the SFM chaotic system. *Axioms* **2023**, *12*, 287.
18. Jin, J.; Zhu, J.; Gong, J.; Chen, W. Novel activation functions-based ZNN models for fixed-time solving dynamic Sylvester equation. *Neural Comput. Appl.* **2022**, *34*, 14297–14315.
19. Kong, Y.; Jiang, Y.; Lou, J. Terminal computing for Sylvester equations solving with application to intelligent control of redundant manipulators. *Neurocomputing* **2019**, *335*, 119–130.
20. Li, S.; Chen, S.; Liu, B. Accelerating a recurrent neural network to finite-time convergence for solving time-varying Sylvester equation by using a sign-Bi-power activation function. *Neural Process. Lett.* **2013**, *37*, 189–205.
21. Xiao, L. A finite-time recurrent neural network for solving online time-varying Sylvester matrix equation based on a new evolution formula. *Nonlinear Dyn.* **2017**, *90*, 1581–1591.
22. Xiao, L.; Tao, J.; Li, W. An arctan-type varying-parameter ZNN for solving time-varying complex Sylvester equations in finite time. *IEEE Trans. Ind. Inf.* **2022**, *18*, 3651–3660.
23. Xiao, L.; He, Y. A noise-suppression ZNN model with new variable parameter for dynamic Sylvester equation. *IEEE Trans. Ind. Inf.* **2021**, *17*, 7513–7522.
24. Xiao, L.; Zhang, Z.; Zhang, Z.; Li, W.; Li, S. Design, verification and robotic application of a novel recurrent neural network for computing dynamic Sylvester equation. *Neural Netw.* **2018**, *105*, 185–196.
25. Xiao, L.; Zhang, Y.; Dai, J.; Li, J.; Li, W. New noise-tolerant ZNN models with predefined-time convergence for time-variant Sylvester equation solving. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 3629–3640.
26. Zhang, Y.; Jiang, D.; Wang, J. A recurrent neural network for solving Sylvester equation with time-varying coefficients. *IEEE Trans. Neural Netw.* **2002**, *13*, 1053–1063.
27. Zhang, Z.; Zheng, L.; Weng, J.; Mao, Y.; Lu, W.; Xiao, L. A new varying-parameter recurrent neural-network for online solution of time-varying Sylvester equation. *IEEE Trans. Cybern.* **2018**, *48*, 3135–3148.
28. Gong, J.; Jin, J. A faster and better robustness zeroing neural network for solving dynamic Sylvester equation. *Neural Process. Lett.* **2021**, *53*, 3591–3606.
29. Li, S.; Li, Y. Nonlinearly Activated neural network for solving time-varying complex Sylvester equation. *IEEE Trans. Cybern.* **2014**, *44*, 1397–1407.
30. Wu, W.; Zhang, Y. Zeroing neural network with coefficient functions and adjustable parameters for solving time-variant Sylvester equation. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 6757–6766.
31. Zhao, J.; Zhang, Y. A novel zeroing neural network with integral upper-bound function for solving time-varying Sylvester equation. *Comput. Appl. Math.* **2025**, *44*, 374.
32. Hua, C.; Cao, X.; Xu, Q.; Liao, B.; Li, S. Dynamic neural network models for time-varying problem solving: A survey on model structures. *IEEE Access* **2023**, *11*, 65991–66008.
33. Micić, I.; Nguen, L.A.; Stanimirović, S. Characterization and computation of approximate bisimulations for fuzzy automata. *Fuzzy Sets Syst.* **2022**, *442*, 331–350.

34. Micić, I.; Ćirić, M.; Matejić, J.; Stanimirović, S.; Nguyen, L.A. Approximate weak simulations and bisimulations for fuzzy automata over the product structure. *Fuzzy Sets Syst.* **2024**, *485*, 108959.
35. Stanimirović, S.; Micić, I.; Ćirić, M. Approximate bisimulations for fuzzy automata over complete Heyting algebras. *IEEE Trans. Fuzzy Syst.* **2022**, *30*, 437–447.
36. Yang, C.; Li, Y. ϵ -bisimulation relations for fuzzy automata. *IEEE Trans. Fuzzy Syst.* **2018**, *26*, 2017–2029.
37. Yang, C.; Li, Y. Approximate bisimulation relations for fuzzy automata. *Soft Comput.* **2018**, *22*, 4535–4547.
38. Yang, C.; Li, Y. Approximate bisimulations and state reduction of fuzzy automata under fuzzy similarity measures. *Fuzzy Sets Syst.* **2020**, *39*, 72–95.
39. Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S.; Zhang, Y. Solving complex-valued time-varying linear matrix equations via QR decomposition with applications to robotic motion tracking and on angle-of-arrival localization. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 3415–3424.
40. Li, X.; Lin, C.-L.; Simos, T.E.; Mourtas, S.D.; Katsikis, V.N. Computation of time-varying $\{2,3\}$ - and $\{2,4\}$ -inverses through zeroing neural networks. *Mathematics* **2022**, *10*, 4759.
41. Simos, T.E.; Katsikis, V.N.; Mourtas, S.D.; Stanimirović, P.S. Unique non-negative definite solution of the time-varying algebraic Riccati equations with applications to stabilization of LTV system. *Math. Comput. Simul.* **2022**, *202*, 164–180.
42. Stanimirović, P.S.; Mourtas, S.D.; Mosić, D.; Katsikis, V.N.; Cao, X.; Li, S. Zeroing Neural Network approaches for computing time-varying minimal rank outer inverse. *Appl. Math. Comput.* **2024**, *465*, 128412.
43. Stanimirović, P.S.; Ćirić, M.; Mourtas, S.D.; Milovanović, G.V.; Petrović, M.J. Simultaneous method for solving certain systems of matrix equations with two unknowns. *Axioms* **2024**, *13*, 838.
44. Stanimirović, P.S.; Ćirić, M.; Mourtas, S.D.; Brzaković, P.; Karabašević, D. Simulations and bisimulations between weighted finite automata based on time-varying models over real numbers. *Mathematics* **2024**, *12*, 2110.
45. Schützenberger, M.-P. On the definition of a family of automata. *Inf. Control* **1961**, *4*, 245–270.
46. Zhang, Y.; Guo, D. *Zhang Functions and Various Models*; Springer: Berlin/Heidelberg, Germany, 2015.
47. Golan, J. *Semirings and Their Applications*; Springer: Dordrecht, The Netherlands, 1999.
48. Damjanović, N.; Ćirić, M.; Ignjatović, J. Weighted automata over vector spaces. *EPTCS* **2023**, *386*, 67–81.
49. Graham, A. *Kronecker Products and Matrix Calculus with Applications*; Courier Dover Publications: Mineola, NY, USA, 2018.
50. Stanimirović, P.S.; Stanimirović, I. Implementation of polynomial multi-objective optimization in Mathematica. *Struct. Multidiscip. Optim.* **2008**, *36*, 411–428.
51. Gunantara, N. A review of multi-objective optimization: Methods and its applications. *Cogent Eng.* **2018**, *5*, 1502242. <https://doi.org/10.1080/23311916.2018.1502242>.
52. Jin, L.; Li, S.; Liao, B.; Zhang, Z. Zeroing neural networks: A survey. *Neurocomputing* **2017**, *267*, 597–604.
53. Zuo, K.; Chen, Y.; Yuan, L. Further representations and computations of the generalized Moore-Penrose inverse. *AIMS Math.* **2023**, *8*, 23442–23458.
54. Buchholz, P. Bisimulation relations for weighted automata. *Theor. Comput. Sci.* **2008**, *393*, 109–123.
55. Noorizadegan, A.; Chen, C.-S.; Cavoretto, R.; Rossi, A.D. Efficient truncated randomized SVD for mesh-free kernel methods. *Comput. Math. Appl.* **2024**, *164*, 12–20.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.