axioms

MDPI

Article

Simultaneous Method for Solving Certain Systems of Matrix Equations with Two Unknowns

Predrag S. Stanimirović [1,2], Miroslav Ćirić [1], Spyridon D. Mourtas [2,3], Gradimir V. Milovanović [1,4] and Milena J. Petrović [5,*]

[1] Faculty of Sciences and Mathematics, University of Niš, Višegradska 33, 18108 Niš, Serbia; pecko@pmf.ni.ac.rs (P.S.S.); miroslav.ciric@pmf.edu.rs (M.Ć.); gvm@mi.sanu.ac.rs (G.V.M.)
[2] Laboratory "Hybrid Methods of Modelling and Optimization in Complex Systems", Siberian Federal University, Prosp. Svobodny 79, Krasnoyarsk 660041, Russia
[3] Department of Economics, Division of Mathematics-Informatics and Statistics-Econometrics, National and Kapodistrian University of Athens, Sofokleous 1 Street, 10559 Athens, Greece; spirmour@econ.uoa.gr
[4] Serbian Academy of Sciences and Arts, Kneza Mihaila 35, 11000 Belgrade, Serbia
[5] Faculty of Sciences and Mathematics, University of Priština in Kosovska Mitrovica, Lole Ribara 29, 38220 Kosovska Mitrovica, Serbia
* Correspondence: milena.petrovic@pr.ac.rs

Abstract: Quantitative bisimulations between weighted finite automata are defined as solutions of certain systems of matrix-vector inequalities and equations. In the context of fuzzy automata and max-plus automata, testing the existence of bisimulations and their computing are performed through a sequence of matrices that is built member by member, whereby the next member of the sequence is obtained by solving a particular system of linear matrix-vector inequalities and equations in which the previously computed member appears. By modifying the systems that define bisimulations, systems of matrix-vector inequalities and equations with k unknowns are obtained. Solutions of such systems, in the case of existence, witness to the existence of a certain type of partial equivalence, where it is not required that the word functions computed by two WFAs match on all input words, but only on all input words whose lengths do not exceed k. Solutions of these new systems represent finite sequences of matrices which, in the context of fuzzy automata and max-plus automata, are also computed sequentially, member by member. Here we deal with those systems in the context of WFAs over the field of real numbers and propose a different approach, where all members of the sequence are computed simultaneously. More precisely, we apply a simultaneous approach in solving the corresponding systems of matrix-vector equations with two unknowns. Zeroing neural network (ZNN) neuro-dynamical systems for approximating solutions of heterotypic bisimulations are proposed. Numerical simulations are performed for various random initial states and comparison with the *Matlab*, linear programming solver `linprog`, and the pseudoinverse solution generated by the standard function `pinv` is given.

Keywords: weighted finite automata; zhang neural network; bisimulation; pseudoinverse

MSC: 15A24; 65F20; 68T05

check for updates

Citation: 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. https://doi.org/10.3390/axioms13120838

Academic Editor: Simeon Reich

Received: 14 October 2024
Revised: 13 November 2024
Accepted: 25 November 2024
Published: 28 November 2024

Copyright: © 2024 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) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction, Motivation and Methodology

One of the main issues in the theory of weighted automata is the *equivalence problem*, which determines whether two weighted automata are equivalent, that is, whether they compute the same word function. In the case of the most general class of weighted automata over a semiring, as well as in the case of most of its subclasses, that problem is undecidable or computationally hard (cf. [1,2]). Only in rare cases it is solvable in polynomial time. This fact has created the need to find methods of determining equivalence that may not work in all cases, but in cases where they are applicable, they can

be efficiently realized. The most powerful tools used for these purposes are *bisimulations*. This notion was introduced by Milner [3] and Park [4], in the context of classical nondeterministic automata, as binary relations that can recognize and relate the states of two automata with similar roles, and thus testify the equivalence between them. Around the same time, bisimulations also appeared in mathematics, in modal logic and set theory (cf. [5–11]). It is worth noting that propositional modal logic is the fragment of first-order logic invariant under bisimulation (cf. [5,7]). Bisimulations are employed today in many areas of computer science, such as functional or object-oriented languages, types, data types, domains, databases, compilers optimizations, program analysis, and verification tools. For more information about bisimulations and their applications we refer to [7–16].

With the transition from traditional Boolean-valued systems to quantitative ones, which are more suitable for modeling numerous properties of real-world systems, there was a need for bisimulations to be quantitative as well. Quantitative bisimulations would be modeled by matrices whose entries would measure the similarity of the roles played by the states of considered systems. Such bisimulations were first introduced and studied in [17,18], in the framework of fuzzy finite automata. The approach to bisimulations initiated in this research consists in defining bisimulations as solutions of certain systems of mixed matrix-vector inequalities and equations. That way, the proposed approach reduces the issues of the existence of bisimulations and their computing to the problems of solving the aforementioned matrix-vector inequalities and equations. Subsequently, fundamentally equal approach was used in the contexts of weighted finite automata (WFAs) over additively idempotent semirings [19], max-plus automata [20], and WFAs over the field of real numbers $\mathbb{R}$ [21], as well as in the most general context of WFAs over a semiring [22]. A similar approach to bisimulations was used in [23–27] (see also [28,29]), while various extensions of quantitative bisimulations were introduced in [30–35].

Procedures for testing existence and computing bisimulations developed for fuzzy finite automata in [18], max-plus automata in [20], and WFAs over additively idempotent semirings in [19], consist of building non-increasing sequences of matrices whose infima, if they satisfy certain conditions, represent the greatest bisimulations. The sequences are built member by member, where each member is derived from the previous one, as a solution of a certain system of matrix inequalities in which the previously computed member also appears. On the other hand, the methodology used in computing bisimulations for WFAs over $\mathbb{R}$ is different from the methodology used in computing bisimulations for fuzzy automata, max-plus automata or WFAs over an additively idempotent semiring.

In practical applications of WFAs, it is often not important whether the word functions of two automata have the same values on all input words, but it is enough to test equality on all input words whose length does not exceed a given natural number k. Such kind of partial equivalence, known as k-equivalence, is often more expedient than the classical equivalence. As shown in [20] (see also [35]), systems of matrix-vector inequalities and equations defining bisimulations can be transformed into simpler systems of matrix-vector inequalities and equations with k unknown matrices whose solutions witness to the existence of k-equivalence between two WFAs. In the mentioned papers, an approach similar to the one used in the calculation of bisimulations was used, where a finite sequence of matrices which represents a solution of a new system with k unknowns, is derived by constructing the next member in the sequence from the previous one. However, such a sequential approach has certain drawbacks. Namely, the next member of a sequence depends on the previous one, which does not have to be unique. Inappropriate choices among candidates for previous members can lead to unwanted situation in which the sequence cannot be continued, while for a different choice the sequence can be continued. This means that although our system has a solution, it may happen that this approach fails to give a solution. For this reason, we propose a different, simultaneous approach, where all members of the sequence of the solution are built simultaneously. Particularly, further on in this paper we deal with solving systems with two unknowns, while systems with more unknowns will be the subject of study in our further research.

Systems of matrix-vector equations required in heterotypic bisimulations in this paper are considered over $\mathbb{R}$, and their solutions are obtained as numerical approximations of solutions to systems of linear matrix-vector equations. A special difficulty is the fact that the required matrix-vector equations are not consistent in the general case. In this way, solving those systems is considered as numerical linear algebra problem.

The proposed algorithm for solving the problem is based on the zeroing neural network (ZNN) dynamic models. It should be noted that ZNN dynamic models were originally created for tracking the time-varying matrix inverse [36]. Later iterations of these models were dynamic models for figuring out the time-varying Moore–Penrose [37]. These days, they are also used to solve generalized inversion problems, such as time-varying outer inverse [38], time-varying Drazin inverse [39], and time-varying ML-weighted pseudoinverse [40]. Additionally, real-world ZNN dynamic model applications include image restoration [41], mobile manipulator control [42,43], chaotic systems synchronization [44], and solving time-variant quadratic programming [45]. A comprehensive survey regarding the application of the ZNN model is available at [46].

Designing a ZNN model is the algorithm consisting of two generic stages. In the initial step, it is necessary to declare a proper matrix or vector ZEF, denoted by $E(t)$. The ZEF $E(t)$ is properly defined if its zero point $E(t) = 0$ coincides with the theoretical solution (TSOL) of the problem. Zhang and Guo in the monograph [47] presented an extensive overview of diverse Zhang functions on different domains. Secondly, the dynamic system based on the time-derivative of $E(t)$

$$\dot{E}(t) = -\lambda E(t) \tag{1}$$

needs to be applied. The convergence speed of the dynamics (1) is controlled by the quantity $\lambda \in \mathbb{R}^+$. It is known that (1) converges faster proportionally with increasing values of λ [47]. The principal outcome of the continuous learning principle in (1) is to force the convergence $E(t) \to 0$ as $t \to \infty$ at an exponential rate λ [47,48]. A feasible ZEF is therefore considered as a tracking indicator during the development of ZNN learning in (1). The essence in defining the ZNN dynamical evolution is an efficient control over the underlying system through appropriate ZEF $E(t)$ and the error dynamics (1).

The models developed in [21] for bisimulations between WFAs over $\mathbb{R}$ are established using a ZNN dynamics in resolving systems of vector-matrix inequalities. Our goal in current research is to solve the system consisting of two vector equations and a variable number of linear matrix equations required in heterotypic bisimulations between WFAs. A mixed system of vector-matrix equations is obtained as a particular k-equivalence problem between two WFAs, resulting in a system with two unknown matrices U_1 and U_2. Such system is inconsistent in the vast majority of cases. Starting from the useful property of the ZNN model in generating approximate solutions to matrix-vector inequalities, confirmed in [21], it was a logical decision to define and implement ZNN neuro-dynamical systems for approximating matrix-vector systems arising from the 2-equivalence between WFAs. Since the considered linear matrix-vector system is not inconsistent in general, the ZNN dynamical system is defined utilizing the induced normal system and its best approximate solution generated in terms of the Moore–Penrose inverse. Numerical simulations are performed to verify effectiveness of the proposed ZNN models and comparison with the *Matlab* linear programming solver `linprog` and the pseudoinverse solution generated by the standard function `pinv`.

In this work, given that underlying linear vector-matrix systems are not solvable in the general case, our proposed action is to use the normal system that generates the best approximate solution, based on the utilization of the Moore–Penrose inverse. Finally, ZNN dynamics is applied as the tool for finding the best approximate solution.

Main results derived in this paper are emphasized as follows.

- A specific approach, based on the k-equivalence of two WFAs and simultaneous approach with two unknown matrices, is applied for solving matrix-vector equations required in heterotypic bisimulations between WFAs.

- ZNN neuro-dynamical systems for approximating the k-equivalence problem based on heterotypic bisimulations are proposed.
- Numerical simulation is presented for various random initial states and comparison with the *Matlab* linear programming solver `linprog` and the pseudoinverse solution generated by the standard function `pinv` is given.

Overall structure of our presentation is as follows. After the introduction section, the problem statement, motivation as well as justification of proposed methodology are presented in Section 2. ZNN design for solving matrix-vector systems corresponding to heterotypic bisimulations based on the k-equivalence of two WFAs and simultaneous approach with two unknown matrices is presented in Section 3. Numerical experiments on the k-equivalence problem arising from heterotypic bisimulations with two unknowns are presented in Section 4. The closing section extracts some terminate comments and describes possibilities for further research on this topic.

2. Preliminaries and Problem Formulation

In the sequel, $\mathbb{N}$ will denote the set of natural numbers (without zero), and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. For any pair $i, j \in \mathbb{N}_0$ satisfying $i < j$, it will be denoted $[i..j] = \{t \in \mathbb{N}_0 \mid i \leqslant t \leqslant j\}$. Moreover, $\mathfrak{X}$ will be a non-empty and finite set with $r \in \mathbb{N}$ elements, known as *alphabet*, while $\mathfrak{X}^+ = \{x_1 \cdots x_t \mid t \in \mathbb{N}, x_1, \ldots, x_t \in \mathfrak{X}\}$ will denote all finite sequences of entries from $\mathfrak{X}$, which are termed as *words* over $\mathfrak{X}$, and $\mathfrak{X}^* = \mathfrak{X}^+ \cup \{\varepsilon\}$, such that $\varepsilon \notin \mathfrak{X}^+$ is a symbol standing for the empty word.

A *weighted finite automaton* (WFA) over $\mathbb{R}$ and an input alphabet $\mathfrak{X}$ is defined as a tuple $\mathcal{A} = (m, \sigma^A, \{M_x^A\}_{x\in\mathfrak{X}}, \tau^A)$, in which

– m is a natural number, called the *dimension* of $\mathcal{A}$,
– $\sigma^A \in \mathbb{R}^{1\times m}$ is the *initial weights vector*,
– $\{M_x^A\}_{x\in\mathfrak{X}} \subset \mathbb{R}^{m\times m}$ is the family of *transition matrices*, each of which corresponds to one input letter $x \in \mathfrak{X}$, and
– $\tau^A \in \mathbb{R}^{m\times 1}$ is the *terminal weights vector*.

The numbers from the set $\{1, \ldots, m\}$ can be interpreted as *states* of the automaton $\mathcal{A}$, and for any state i, the i-th entries of the vectors σ^A and τ^A can be understood as measures of certainty that the automaton will start working from that state or finish working in that state, respectively, while for the states i and j, the (i, j)-th entry of the matrix M_x^A can be understood as a measure of certainty that the automaton will move from the state i to state j under the influence of the input signal represented by the letter x. Inputs of the vector σ^A are called *initial weights*, the entries of τ^A are called *terminal weights*, while the entries of the matrix M_x^A are called *transition weights*.

The *behavior* of the WFA $\mathcal{A}$ is defined as the *word function* $[\![\mathcal{A}]\!] : \mathfrak{X}^* \to \mathbb{R}$ which to any word $u = x_1 \ldots x_s \in \mathfrak{X}^+$, $x_1, \ldots, x_s \in \mathfrak{X}$ assigns the weight $[\![\mathcal{A}]\!](u)$ which is computed as

$$[\![\mathcal{A}]\!](u) = \sigma^A M_{x_1}^A \cdots M_{x_s}^A \tau^A = \sigma^A M_u^A \tau^A, \tag{2}$$

where $M_u^A = M_{x_1}^A \cdots M_{x_s}^A$, and

$$[\![\mathcal{A}]\!](\varepsilon) = \sigma^A \tau^A. \tag{3}$$

It is said that the automaton $\mathcal{A}$ computes the function $[\![\mathcal{A}]\!]$.

Consider two WFAs $\mathcal{A} = (m, \sigma^A, \{M_x^A\}_{x\in\mathfrak{X}}, \tau^A)$ and $\mathcal{B} = (n, \sigma^B, \{M_x^B\}_{x\in\mathfrak{X}}, \tau^B)$ over $\mathbb{R}$ and an alphabet $\mathfrak{X} = \{x_1, \ldots, x_r\}$. Here, we are interested in systems of matrix and vector equations that define the so-called heterotypic bisimulations between $\mathcal{A}$ and $\mathcal{B}$, as proposed in [22]. These are the following two systems:

$$\begin{aligned}
&\text{(hfbb-1)} \quad \sigma^A = \sigma^B U^{\mathrm{T}} \\
&\text{(hfbb-2)} \quad U^{\mathrm{T}} M_x^A = M_x^B U^{\mathrm{T}} \quad (x \in \mathfrak{X} = \{x_1, \ldots, x_r\}) \\
&\text{(hfbb-3)} \quad U^{\mathrm{T}} \tau^A = \tau^B
\end{aligned} \tag{4}$$

and

$$\begin{aligned} &\text{(hbfb-1)} \quad \tau^A = U\,\tau^B \\ &\text{(hbfb-2)} \quad M_x^A\,U = U\,M_x^B \quad (x \in \mathfrak{X} = \{x_1, \ldots, x_r\}) \\ &\text{(hbfb-3)} \quad \sigma^A\,U = \sigma^B \end{aligned} \tag{5}$$

where U is an unknown matrix of dimension $m \times n$. Matrices which are solutions of (4) are called *forward-backward heterotypic bisimulations* (*fbb* for short) between $\mathcal{A}$ and $\mathcal{B}$, and those which are solutions of (5) are *backward-forward heterotypic bisimulations* (*bfb* for short) between $\mathcal{A}$ and $\mathcal{B}$ [22].

Recall that the automata $\mathcal{A}$ and $\mathcal{B}$ are equivalent if $[\![\mathcal{A}]\!](u) = [\![\mathcal{B}]\!](u)$, for every word $u \in \mathfrak{X}^*$. However, in real-word applications it is not always necessary that this equality holds for all words $u \in \mathfrak{X}^*$; it is often enough that it holds for all words of length $|u| \leqslant k$, for a given natural number k. If this relaxed condition holds, then we say that the automata $\mathcal{A}$ and $\mathcal{B}$ are *k-equivalent*, and the equivalence problem can be transformed into the *k-equivalence problem*, which decides whether the automata $\mathcal{A}$ and $\mathcal{B}$ are k-equivalent.

The problem of k-equivalence will be the subject of a separate study, and here we will only present without proofs the way in which the existence of k-equivalence between automata can be witnessed, similar to the way bisimulations testify to the existence of equivalence.

Consider WFAs $\mathcal{A} = (m, \sigma^A, \{M_x^A\}_{x\in\mathfrak{X}}, \tau^A)$ and $\mathcal{B} = (n, \sigma^B, \{M_x^B\}_{x\in\mathfrak{X}}, \tau^B)$ and a sequence of matrices $\{U_i\}_{i\in[0..k]} \subset \mathbb{R}^{m\times n}$ that satisfies the following conditions obtained from the system (4):

$$\begin{aligned} &\text{(fbb-1}^k\text{)} \quad \sigma^A = \sigma^B U_0^{\mathrm{T}}, \\ &\text{(fbb-2}^k\text{)} \quad U_{i-1}^{\mathrm{T}} M_x^A = M_x^B U_i^{\mathrm{T}}, \qquad \text{for all } x \in \mathfrak{X} \text{ and } i \in [1 \ldots k], \\ &\text{(fbb-3}^k\text{)} \quad U_i^{\mathrm{T}} \tau^A = \tau^B, \qquad \text{for each } i \in [0 \ldots k]. \end{aligned} \tag{6}$$

If such a sequence exists, then the automata $\mathcal{A}$ and $\mathcal{B}$ are k-equivalent. Therefore, our task is to determine existence of matrices $U_0, U_1, \ldots, U_k$ satisfying (6). Note that (6) can be understood as a system of equations with $k+1$ unknown matrices $U_0, U_1, \ldots, U_k$, so our task is to find a solution to that system.

Sequences of matrices (possibly infinite), defined in a similar way as in (6), were studied in [35], in the circumstances of fuzzy finite automata, and in [20], in the circumstances of max-plus automata. Sequences defined in [35] were called *depth-bounded bisimulations*. The *sequential approach* used in [20], applied to solving the system (6), consists of building a sequence $U_0, U_1, \ldots, U_k$ member by member, starting from the zero member U_0, which is computed by solving the equation (fbb-1^k), while the i-th member is computed after the $(i-1)$st member, by solving the system of equations $U_{i-1}^{\mathrm{T}} M_x^A = M_x^B U_i^{\mathrm{T}}$ and $U_i^{\mathrm{T}} \tau^A = \tau^B$, with one unknown U_i.

Since U_{i-1} is not unique in the general case, the disadvantage of the sequential approach is that finding a solution for the unknown U_i depends on the choice of the particular solution for the unknown U_{i-1}. Therefore, it may happen that for some choice of a solution for U_{i-1} there is no solution for the unknown U_i, in which case formation of the sequence interrupts and we cannot find solutions for all the unknowns, although such solutions may exist.

Consequently, the question arises whether it is possible to apply the *simultaneous approach*, where solutions for all unknowns are sought at the same time. Here, we will consider the application of the simultaneous approach for the instance of the system (6) with two unknowns. For the sake of simplicity, we will denote those unknowns by U_1 and U_2, instead of U_0 and U_1, as was done in (6). In other words, we will deal with the following system of matrix-vector equations:

$$\begin{cases} \sigma^B U_1^{\mathrm{T}} = \sigma^A, \\ U_2^{\mathrm{T}} \tau^A = \tau^B, \\ U_1^{\mathrm{T}} M_{x_i}^A = M_{x_i}^B U_2^{\mathrm{T}}, \qquad \text{for each } i \in [1 \dots r], \end{cases} \tag{7}$$

where U_1 and U_2 are unknown matrices of dimension $m \times n$. In the dual case, we deal with the following system of matrix-vector equations based on (5):

$$\begin{cases} \sigma^A U_2 = \sigma^B, \\ U_1 \tau^B = \tau^A, \\ U_1 M_{x_i}^B = M_{x_i}^A U_2, \qquad \text{for each } i \in [1 \dots r], \end{cases} \tag{8}$$

where U_1 and U_2 stand for $m \times n$ unknown matrices.

The ZNN evolution is a validated matrix equations solver, confirmed in survey papers [48–50] and in a number of research papers [51–54]. Based on the initial step in the construction of ZNN dynamics, it is necessary to define a proper error function for each matrix and vector equation that is included in the system which is being resolved. The development strategy of ZNN dynamics arising from multiple Zhang error functions (ZEFs) has been utilized in several research articles, of which the most important are [38,55,56]. The ZNN models studied so far with common error functions enabled the convergence of each error function to approximation of its zero. The main idea is to generate an appropriate composite error block matrix which involves individual error functions.

The problem considered in current research is more complex, since systems of matrix and vector equations required in (7) and (8) are not solvable in the general case. The ZNN design is known as a confirmed tool for forcing the underlying error function to zero with global exponential convergence. So, it is expectable that the error functions corresponding to (7) and (8) can be forced to zero, which will lead to approximate solutions of these matrix-vector system.

Global convergence of ZNN design for arbitrary initial state can be used as a confirmation of its efficiency in solving (7) and (8). Details are described in subsequent section.

3. ZNN for Solving the Proposed Systems

This section develops, investigates, and tests two novel ZNN models aimed to solving the matrix-vector systems (4) and (5). Let $\mathcal{A} = \left(m, \sigma^A, \{M_{x_i}^A\}_{x_i \in \mathfrak{X}}, \tau^A\right)$ and $\mathcal{B} = \left(n, \sigma^B, \{M_{x_i}^B\}_{x_i \in \mathfrak{X}}, \tau^B\right)$ be WFAs over $\mathbb{R}$ and the alphabet $\mathfrak{X} = \{x_1, \dots, x_r\}$, 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 $M_{x_i}^B \in \mathbb{R}^{n \times n}$, $\sigma^B \in \mathbb{R}^{1 \times n}$, $\tau^B \in \mathbb{R}^{n \times 1}$, $i \in [1..r]$.

The $p \times 1$ vectors with all inputs equal 1 (resp. 0) will be termed as $\mathbf{1}_p$ (resp. $\mathbf{0}_p$), whereas the $p \times r$ matrix with all entries equal to 1 (resp. 0) will be termed as $\mathbf{1}_{p,r}$ and $\mathbf{0}_{p,r}$. Following the conventional notation, the $q \times q$ identity matrix will be marked by I_q, whereas $\mathrm{vec}(), \otimes, (\,)^{\dagger}$ and $\|\|_{\mathrm{F}}$ will mean the vectorization, the Kronecker product product, pseudoinversion, and the Frobenius norm, in that order.

3.1. The ZNNL-hfbb Model

Following the priority (hfbb-3) and (hfbb-1) and then (hfbb-2), let us consider the system (7) as a model for solving (4). In line with the adopted order in solving (4), the next equations must be satisfied:

$$\begin{cases} \sigma^B U_1^{\mathrm{T}}(t) - \sigma^A = \mathbf{0}_m^{\mathrm{T}}, \\ U_2^{\mathrm{T}}(t) \tau^A - \tau^B = \mathbf{0}_n, \\ U_1^{\mathrm{T}}(t) M_{x_i}^A - M_{x_i}^B U_2^{\mathrm{T}}(t) = \mathbf{0}_{n,m}, \end{cases} \tag{9}$$

where $U_1(t), U_2(t) \in \mathbb{R}^{m\times n}$ denote unknown matrices. Exploiting vectorization and the Kronecker product, (9) is reformulated in the equivalent form

$$\begin{cases} (I_m \otimes \sigma^B)\mathrm{vec}(U_1^{\mathrm{T}}(t)) - (\sigma^A)^{\mathrm{T}} = \mathbf{0}_m, \\ ((\tau^A)^{\mathrm{T}} \otimes I_n)\mathrm{vec}(U_2^{\mathrm{T}}(t)) - \tau^B = \mathbf{0}_n, \\ ((M_{x_i}^A)^{\mathrm{T}} \otimes I_n)\mathrm{vec}(U_1^{\mathrm{T}}(t)) - (I_m \otimes M_{x_i}^B)\mathrm{vec}(U_2^{\mathrm{T}}(t)) = \mathbf{0}_{mn}. \end{cases} \tag{10}$$

To calculate solutions $U_1(t), U_2(t)$ in a more efficient manner, (10) must be made simpler. Lemma 1 is restated from [57].

Lemma 1. *For $W \in \mathbb{R}^{m\times n}$, let $\mathrm{vec}(W) \in \mathbb{R}^{mn}$ denote the matrix W vectorization. What is stated below is true:*

$$\mathrm{vec}(W^{\mathrm{T}}) = P\,\mathrm{vec}(W), \tag{11}$$

where $P \in \mathbb{R}^{mn\times mn}$ is an appropriate permutation matrix depended from the number of columns n and rows m of matrix W.

The procedure for generating the permutation matrix P used in (11) is demonstrated in Algorithm 1 from [21]. Using P in generating $\mathrm{vec}(U^{\mathrm{T}}(t))$, (10) can be rewritten as

$$\begin{cases} (I_m \otimes \sigma^B)P\,\mathrm{vec}(U_1(t)) - (\sigma^A)^{\mathrm{T}} = \mathbf{0}_m, \\ ((\tau^A)^{\mathrm{T}} \otimes I_n)P\,\mathrm{vec}(U_2(t)) - \tau^B = \mathbf{0}_n, \\ ((M_{x_i}^A)^{\mathrm{T}} \otimes I_n)P\,\mathrm{vec}(U_1(t)) - (I_m \otimes M_{x_i}^B)P\,\mathrm{vec}(U_2(t)) = \mathbf{0}_{mn}, \end{cases} \tag{12}$$

while its corresponding matrix form is

$$L_{fbb}\begin{bmatrix}\mathrm{vec}(U_1(t))\\ \mathrm{vec}(U_2(t))\end{bmatrix} - \begin{bmatrix}(\sigma^A)^{\mathrm{T}}\\ \tau^B\\ \mathbf{0}_{rmn}\end{bmatrix} = \mathbf{0}_z, \tag{13}$$

in which $z = rmn + m + n$ and

$$\begin{aligned} L_{fbb} &= \begin{bmatrix}(I_m \otimes \sigma^B)P & \mathbf{0}_{m,mn}\\ \mathbf{0}_{n,mn} & ((\tau^A)^{\mathrm{T}} \otimes I_n)P\\ W_1 & W_2\end{bmatrix} \in \mathbb{R}^{z\times 2mn}, \\ W_1 &= \begin{bmatrix}((M_{x_1}^A)^{\mathrm{T}} \otimes I_n)P\\ ((M_{x_2}^A)^{\mathrm{T}} \otimes I_n)P\\ \vdots\\ ((M_{x_r}^A)^{\mathrm{T}} \otimes I_n)P\end{bmatrix} \in \mathbb{R}^{rmn\times mn}, \quad W_2 = \begin{bmatrix}(-I_m \otimes M_{x_1}^B)P\\ (-I_m \otimes M_{x_2}^B)P\\ \vdots\\ (-I_m \otimes M_{x_r}^B)P\end{bmatrix} \in \mathbb{R}^{rmn\times mn}. \end{aligned} \tag{14}$$

Based on considered transformations, the ZNN learning exploits the following ZEF, which is based on (13), for satisfying simultaneously all the equations in (9):

$$E_{fbb}(t) = L_{fbb}\begin{bmatrix}\mathrm{vec}(U_1(t))\\ \mathrm{vec}(U_2(t))\end{bmatrix} - \begin{bmatrix}(\sigma^A)^{\mathrm{T}}\\ \tau^B\\ \mathbf{0}_{rmn}\end{bmatrix}, \tag{15}$$

where $U_1(t)$ and $U_2(t)$ are unknown matrices. The time-derivative of (15) is the following:

$$\dot{E}_{fbb}(t) = L_{fbb}\begin{bmatrix}\mathrm{vec}(\dot{U}_1(t))\\ \mathrm{vec}(\dot{U}_2(t))\end{bmatrix}. \tag{16}$$

Then, combining Equations (15) and (16) with the ZNN design (1), the following can be obtained:

$$L_{fbb}\begin{bmatrix}\mathrm{vec}(\dot{U}_1(t))\\ \mathrm{vec}(\dot{U}_2(t))\end{bmatrix} = -\lambda E_{fbb}(t). \tag{17}$$

As a result, setting

$$\mathbf{x}(t)=\begin{bmatrix}\mathrm{vec}(U_1(t))\\ \mathrm{vec}(U_2(t))\end{bmatrix}\in\mathbb{R}^{2mn}, \qquad \dot{\mathbf{x}}(t)=\begin{bmatrix}\mathrm{vec}(\dot{U}_1(t))\\ \mathrm{vec}(\dot{U}_2(t))\end{bmatrix}\in\mathbb{R}^{2mn}, \tag{18}$$

the following dynamics are developed

$$L_{fbb}\,\dot{\mathbf{x}} = -\lambda E_{fbb}(t). \tag{19}$$

The normal equation corresponding to (19) is given in the form

$$\Big(L_{fbb}\Big)^{\mathrm{T}} L_{fbb}\,\dot{\mathbf{x}} = -\lambda\Big(L_{fbb}\Big)^{\mathrm{T}} E_{fbb}(t),$$

which leads to the Moore–Penrose best approximate solution

$$\dot{\mathbf{x}} = L^{\dagger}_{fbb}\Big(-\lambda E_{fbb}(t)\Big). \tag{20}$$

Appropriately defined *Matlab*'s `ode` solver is utilized to solve the ZNN design based on (20), and marked as ZNNL-hfbb. The ZNNL-hfbb's convergence and stability is considered in Theorem 1.

Theorem 1. *Let* $\mathcal{A} = \Big(m, \sigma^A, \{M^A_{x_i}\}_{x_i\in\mathfrak{X}}, \tau^A\Big)$ *and* $\mathcal{B} = \Big(n, \sigma^B, \{M^B_{x_i}\}_{x_i\in\mathfrak{X}}, \tau^B\Big)$ *be WFAs over* $\mathbb{R}$, *such that* $M^A_{x_i} \in \mathbb{R}^{m\times m}$, $\sigma^A \in \mathbb{R}^{1\times m}$, $\tau^A \in \mathbb{R}^{m\times 1}$ *and* $M^B_{x_i} \in \mathbb{R}^{n\times n}$, $\sigma^B \in \mathbb{R}^{1\times n}$, $\tau^B \in \mathbb{R}^{n\times 1}$, $i \in [1..r]$. *The dynamical system* (17) *inline with the ZNN* (1) *generate the TSOL*

$$\mathbf{x}_{\mathcal{S}}(t) = \begin{bmatrix}\mathrm{vec}(U_{1,\mathcal{S}}(t))^{\mathrm{T}} & \mathrm{vec}(U_{2,\mathcal{S}}(t))^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}},$$

which is stable in view of the theory of Lyapunov.

Proof. Let

$$\begin{cases}\sigma^B U^{\mathrm{T}}_{1,\mathcal{S}}(t) - \sigma^A = \mathbf{0}^{\mathrm{T}}_m,\\ U^{\mathrm{T}}_{2,\mathcal{S}}(t)\tau^A - \tau^B = \mathbf{0}_n,\\ U^{\mathrm{T}}_{1,\mathcal{S}}(t)M^A_{x_i} - M^B_{x_i}U^{\mathrm{T}}_{2,\mathcal{S}}(t) = \mathbf{0}_{n,m}.\end{cases} \tag{21}$$

Using vectorization, the Kronecker product, and the permutation matrix P for generating $\mathrm{vec}(U^{\mathrm{T}}_{1,\mathcal{S}}(t)(t))$ and $\mathrm{vec}(U^{\mathrm{T}}_{2,\mathcal{S}}(t)(t))$, the aforementioned system is reformulated as follows:

$$\begin{cases}(I_m \otimes \sigma^B)P\,\mathrm{vec}(U_{1,\mathcal{S}}(t)) - (\sigma^A)^{\mathrm{T}} = \mathbf{0}_m,\\ ((\tau^A)^{\mathrm{T}} \otimes I_n)P\,\mathrm{vec}(U_{2,\mathcal{S}}(t)) - \tau^B = \mathbf{0}_n,\\ ((M^A_{x_i})^{\mathrm{T}} \otimes I_n)P\,\mathrm{vec}(U_{1,\mathcal{S}}(t)) - (I_m \otimes M^B_{x_i})P\mathrm{vec}(U_{2,\mathcal{S}}(t)) = \mathbf{0}_{mn},\end{cases} \tag{22}$$

or in equivalent form

$$L_{fbb}\begin{bmatrix}\mathrm{vec}(U_{1,\mathcal{S}}(t))\\ \mathrm{vec}(U_{2,\mathcal{S}}(t))\end{bmatrix} - \begin{bmatrix}(\sigma^A)^{\mathrm{T}}\\ \tau^B\\ \mathbf{0}_{rmn}\end{bmatrix} = \mathbf{0}_z \tag{23}$$

where L_{fbb} is declared in (14).

Further, the substitution

$$\mathbf{x}_{\mathcal{O}}(t) := -\mathbf{x}(t) + \mathbf{x}_{\mathcal{S}}(t) = \begin{bmatrix} -\mathrm{vec}(U_1(t)) + \mathrm{vec}(U_{1,\mathcal{S}}(t)) \\ -\mathrm{vec}(U_2(t)) + \mathrm{vec}(U_{2,\mathcal{S}}(t)) \end{bmatrix}$$

gives

$$\mathbf{x}(t) = \mathbf{x}_{\mathcal{S}}(t) - \mathbf{x}_{\mathcal{O}}(t) = \begin{bmatrix} \mathrm{vec}(U_{1,\mathcal{S}}(t)) - \mathrm{vec}(U_{1,\mathcal{O}}(t)) \\ \mathrm{vec}(U_{2,\mathcal{S}}(t)) - \mathrm{vec}(U_{2,\mathcal{O}}(t)) \end{bmatrix},$$

which leads to the first derivative of $\mathbf{x}(t)$

$$\dot{\mathbf{x}}(t) = \dot{\mathbf{x}}_{\mathcal{S}}(t) - \dot{\mathbf{x}}_{\mathcal{O}}(t) = \begin{bmatrix} \mathrm{vec}(\dot{U}_{1,\mathcal{S}}(t)) - \mathrm{vec}(\dot{U}_{1,\mathcal{O}}(t)) \\ \mathrm{vec}(\dot{U}_{2,\mathcal{S}}(t)) - \mathrm{vec}(\dot{U}_{2,\mathcal{O}}(t)) \end{bmatrix}.$$

As a consequence, the substitution $\mathbf{x}(t) = \mathbf{x}_{\mathcal{S}}(t) - \mathbf{x}_{\mathcal{O}}(t)$ in (13) for leads to

$$E_{\mathcal{S}}(t) = L_{fbb} \begin{bmatrix} \mathrm{vec}(U_{1,\mathcal{S}}(t)) - \mathrm{vec}(U_{1,\mathcal{O}}(t)) \\ \mathrm{vec}(U_{2,\mathcal{S}}(t)) - \mathrm{vec}(U_{2,\mathcal{O}}(t)) \end{bmatrix} - \begin{bmatrix} (\sigma^A)^{\mathrm{T}} \\ \tau^B \\ \mathbf{0}_{rmn} \end{bmatrix}, \tag{24}$$

or equivalently

$$E_{\mathcal{S}}(t) = L_{fbb}(\mathbf{x}_{\mathcal{S}}(t) - \mathbf{x}_{\mathcal{O}}(t)) - \begin{bmatrix} (\sigma^A)^{\mathrm{T}} \\ \tau^B \\ \mathbf{0}_{rmn} \end{bmatrix}. \tag{25}$$

The subsequent dynamics arise from (1):

$$\dot{E}_{\mathcal{S}}(t) = L_{fbb} \begin{bmatrix} \mathrm{vec}(\dot{U}_{1,\mathcal{S}}(t)) - \mathrm{vec}(\dot{U}_{1,\mathcal{O}}(t)) \\ \mathrm{vec}(\dot{U}_{2,\mathcal{S}}(t)) - \mathrm{vec}(\dot{U}_{2,\mathcal{O}}(t)) \end{bmatrix} = -\lambda E_{\mathcal{S}}(t), \tag{26}$$

with equivalent form

$$\dot{E}_{\mathcal{S}}(t) = L_{fbb}(\dot{\mathbf{x}}_{\mathcal{S}}(t) - \dot{\mathbf{x}}_{\mathcal{O}}(t)) = -\lambda E_{\mathcal{S}}(t). \tag{27}$$

The Lyapunov function chosen to confirm the convergence is defined by

$$\mathcal{Z}(t) = \frac{1}{2}\|E_{\mathcal{S}}(t)\|_{\mathrm{F}}^2 = \frac{1}{2}\mathrm{tr}\Big(E_{\mathcal{S}}(t)(E_{\mathcal{S}}(t))^{\mathrm{T}}\Big). \tag{28}$$

The following could be concluded in this case:

$$\dot{\mathcal{Z}}(t) = \frac{2\mathrm{tr}\Big((E_{\mathcal{S}}(t))^{\mathrm{T}}\dot{E}_{\mathcal{S}}(t)\Big)}{2} = \mathrm{tr}\Big((E_{\mathcal{S}}(t))^{\mathrm{T}}\dot{E}_{\mathcal{S}}(t)\Big) = -\lambda\mathrm{tr}\Big((E_{\mathcal{S}}(t))^{\mathrm{T}}E_{\mathcal{S}}(t)\Big). \tag{29}$$

Based on (29), it can be concluded

$$\begin{aligned}
&\dot{\mathcal{Z}}(t)\begin{cases}<0, E_{\mathcal{S}}(t)\neq 0,\\ =0, E_{\mathcal{S}}(t)=0,\end{cases}\\
\Leftrightarrow&\dot{\mathcal{Z}}(t)\begin{cases}<0, L_{fbb}(\mathbf{x}_{\mathcal{S}}(t)-\mathbf{x}_{\mathcal{O}}(t))-\mathbf{b}_{fbb}\neq 0,\\ =0, L_{fbb}(\mathbf{x}_{\mathcal{S}}(t)-\mathbf{x}_{\mathcal{O}}(t))-\mathbf{b}_{fbb}=0,\end{cases}\\
\Leftrightarrow&\dot{\mathcal{Z}}(t)\begin{cases}<0, L_{fbb}\begin{bmatrix}\mathrm{vec}(U_{1,\mathcal{S}}(t))-\mathrm{vec}(U_{1,\mathcal{O}}(t))\\ \mathrm{vec}(U_{2,\mathcal{S}}(t))-\mathrm{vec}(U_{2,\mathcal{O}}(t))\end{bmatrix}-\begin{bmatrix}(\sigma^A)^{\mathrm{T}}\\ \tau^B\\ \mathbf{0}_{rmn}\end{bmatrix}\neq 0,\\ =0, L_{fbb}\begin{bmatrix}\mathrm{vec}(U_{1,\mathcal{S}}(t))-\mathrm{vec}(U_{1,\mathcal{O}}(t))\\ \mathrm{vec}(U_{2,\mathcal{S}}(t))-\mathrm{vec}(U_{2,\mathcal{O}}(t))\end{bmatrix}-\begin{bmatrix}(\sigma^A)^{\mathrm{T}}\\ \tau^B\\ \mathbf{0}_{rmn}\end{bmatrix}=0,\end{cases}\\
\Leftrightarrow&\dot{\mathcal{Z}}(t)\begin{cases}<0, & \begin{bmatrix}\mathrm{vec}(U_{1,\mathcal{O}}(t))\\ \mathrm{vec}(U_{2,\mathcal{O}}(t))\end{bmatrix}\neq 0,\\ =0, & \begin{bmatrix}\mathrm{vec}(U_{1,\mathcal{O}}(t))\\ \mathrm{vec}(U_{2,\mathcal{O}}(t))\end{bmatrix}=0.\end{cases}\\
\Leftrightarrow&\dot{\mathcal{Z}}(t)\begin{cases}<0, & \mathbf{x}_{\mathcal{O}}(t)\neq 0,\\ =0, & \mathbf{x}_{\mathcal{O}}(t)=0.\end{cases}
\end{aligned} \tag{30}$$

Furthermore, because $E_{\mathcal{S}}(0) = 0$ and $\mathbf{x}_{\mathcal{O}}(t)$ are the equilibrium points of (27), the following holds:

$$\forall\, \mathbf{x}_{\mathcal{O}}(t) \neq 0, \quad \dot{\mathcal{Z}}(t) \leq 0. \tag{31}$$

It becomes visible that the equilibrium state

$$\mathbf{x}_{\mathcal{O}}(t) = -\mathbf{x}(t) + \mathbf{x}_{\mathcal{S}}(t) = \begin{bmatrix}-\mathrm{vec}(U_1(t)) + \mathrm{vec}(U_{1,\mathcal{S}}(t))\\ -\mathrm{vec}(U_2(t)) + \mathrm{vec}(U_{2,\mathcal{S}}(t))\end{bmatrix} = 0$$

is stable in the sense of Lyapunov. After all is considered, as $t \to \infty$, the following holds

$$\mathbf{x}(t) = \begin{bmatrix}\mathrm{vec}(U_1(t))\\ \mathrm{vec}(U_2(t))\end{bmatrix} \to \mathbf{x}_{\mathcal{S}}(t) = \begin{bmatrix}\mathrm{vec}(U_{1,\mathcal{S}}(t))\\ \mathrm{vec}(U_{2,\mathcal{S}}(t))\end{bmatrix},$$

which was our original intention. □

Theorem 2. *Let* $\mathcal{A} = \left(m, \sigma^A, \{M^A_{x_i}\}_{x_i\in\mathfrak{X}}, \tau^A\right)$ *and* $\mathcal{B} = \left(n, \sigma^B, \{M^B_{x_i}\}_{x_i\in\mathfrak{X}}, \tau^B\right)$ *be WFAs defined by* $M^A_{x_i} \in \mathbb{R}^{m\times m}$, $\sigma^A \in \mathbb{R}^{1\times m}$, $\tau^A \in \mathbb{R}^{m\times 1}$ *and* $M^B_{x_i} \in \mathbb{R}^{n\times n}$, $\sigma^B \in \mathbb{R}^{1\times n}$, $\tau^B \in \mathbb{R}^{n\times 1}$, $i \in [1..r]$. *Starting from an arbitrary initialization* $\mathbf{x}(0)$, *the ZNNL-hfbb design* (20) *converges exponentially to* $\mathbf{x}^*(t)$, *which coincides with the TSOL of* (4).

Proof. The system (9) defines the solution $\mathbf{x}(t) = [\mathrm{vec}(U_1(t))^{\mathrm{T}}, \mathrm{vec}(U_2(t))^{\mathrm{T}}]^{\mathrm{T}}$, which affiliates to the backward–forward bisimulation between $\mathcal{A}$ and $\mathcal{B}$. Next, the system (9) is rewritten into (10) and then into (13) for generating $\mathrm{vec}(U^{\mathrm{T}}_{1,\mathcal{S}}(t)(t))$ and $\mathrm{vec}(U^{\mathrm{T}}_{2,\mathcal{S}}(t)(t))$. Thirdly, the ZEF (15) is established to solve the system (13) and the ZNN evolution is exploited to generate the solution $\mathbf{x}(t)$ of (4). Later, (17) is generated by the ZNN design (1) aimed to zeroing (15). In accordance with Theorem 1, the $E_{fbb}(t) \to 0$ as $t \to \infty$. In consequence, the solution of the dynamical system (20) tends to $\mathbf{x}^*(t) = [\mathrm{vec}(U^*_1(t))^{\mathrm{T}}, \mathrm{vec}(U^*_2(t))^{\mathrm{T}}]^{\mathrm{T}}$ as $t \to \infty$. Moreover, it is evident that (20) is another form of (17). □

3.2. The ZNNL-hbfb Model

According to the system (8) arising from (5), the subsequent matrix-vector equations must be fulfilled:

$$\begin{cases} \tau^A - U_1(t)\tau^B = \mathbf{0}_m, \\ \sigma^A U_2(t) - \sigma^B = \mathbf{0}_n^{\mathrm{T}}, \\ M_{x_i}^A U_2(t) - U_1(t) M_{x_i}^B = \mathbf{0}_{m,n}, \end{cases} \tag{32}$$

where $U_1(t), U_2(t) \in \mathbb{R}^{m\times n}$ imply unknown matrices. Applying vectorization and the Kronecker product, the system (32) is rewritten into

$$\begin{cases} -((\tau^B)^{\mathrm{T}} \otimes I_m)\mathrm{vec}(U_1(t)) + \tau^A = \mathbf{0}_m, \\ (I_n \otimes \sigma^A)\mathrm{vec}(U_2(t)) - (\sigma^B)^{\mathrm{T}} = \mathbf{0}_n, \\ (I_n \otimes M_{x_i}^A)\mathrm{vec}(U_2(t)) - ((M_{x_i}^B)^{\mathrm{T}} \otimes I_m)\mathrm{vec}(U_1(t)) = \mathbf{0}_{mn}. \end{cases} \tag{33}$$

Then, the corresponding matrix form of (33) is the following:

$$L_{bfb}\begin{bmatrix}\mathrm{vec}(U_1(t)) \\ \mathrm{vec}(U_2(t))\end{bmatrix} - \begin{bmatrix}-\tau^A \\ (\sigma^B)^{\mathrm{T}} \\ \mathbf{0}_{rmn}\end{bmatrix} = \mathbf{0}_z, \tag{34}$$

where

$$\begin{aligned} L_{bfb} &= \begin{bmatrix} -(\tau^B)^{\mathrm{T}} \otimes I_m & \mathbf{0}_{m,mn} \\ \mathbf{0}_{n,mn} & I_n \otimes \sigma^A \\ W_1 & W_2 \end{bmatrix} \in \mathbb{R}^{z\times 2mn}, \\ W_1 &= \begin{bmatrix} -(M_{x_1}^B)^{\mathrm{T}} \otimes I_m \\ -(M_{x_2}^B)^{\mathrm{T}} \otimes I_m \\ \vdots \\ -(M_{x_r}^B)^{\mathrm{T}} \otimes I_m \end{bmatrix} \in \mathbb{R}^{rmn\times mn}, \quad W_2 = \begin{bmatrix} I_n \otimes M_{x_1}^A \\ I_n \otimes M_{x_2}^A \\ \vdots \\ I_n \otimes M_{x_r}^A \end{bmatrix} \in \mathbb{R}^{rmn\times mn}. \end{aligned} \tag{35}$$

Following that, the ZNN develops on the following ZEF based on (34), for simultaneous solving of the equations in (32):

$$E_{bfb}(t) = L_{bfb}\begin{bmatrix}\mathrm{vec}(U_1(t)) \\ \mathrm{vec}(U_2(t))\end{bmatrix} - \begin{bmatrix}-\tau^A \\ (\sigma^B)^{\mathrm{T}} \\ \mathbf{0}_{2mn}\end{bmatrix}, \tag{36}$$

in which $U_1(t)$ and $U_2(t)$ are unknowns. The derivative of (36) is equal to

$$\dot{E}_{bfb}(t) = L_{bfb}\begin{bmatrix}\mathrm{vec}(\dot{U}_1(t)) \\ \mathrm{vec}(\dot{U}_2(t))\end{bmatrix}. \tag{37}$$

Combining (36) and (37) with the ZNN (1), the following can be obtained:

$$L_{bfb}\begin{bmatrix}\mathrm{vec}(\dot{U}_1(t)) \\ \mathrm{vec}(\dot{U}_2(t))\end{bmatrix} = -\lambda E_{bfb}(t). \tag{38}$$

As a result, setting

$$\dot{\mathbf{x}}(t) = \begin{bmatrix}\mathrm{vec}(\dot{U}_1(t)) \\ \mathrm{vec}(\dot{U}_2(t))\end{bmatrix} \in \mathbb{R}^{2mn}, \quad \mathbf{x}(t) = \begin{bmatrix}\mathrm{vec}(U_1(t)) \\ \mathrm{vec}(U_2(t))\end{bmatrix} \in \mathbb{R}^{2mn}, \tag{39}$$

the next model is obtained

$$L_{bfb}\dot{\mathbf{x}} = -\lambda E_{bfb}(t), \tag{40}$$

whose best approximate solution is

$$\dot{\mathbf{x}} = L_{bfb}^{\dagger}\Big(-\lambda E_{bfb}(t)\Big). \tag{41}$$

A suitable *Matlab*'s `ode` solver can be used in the implementation of the ZNN design (41), termed as the ZNNL-hbfb. The ZNNL-hbfb convergence and stability are investigated in the Theorem 3.

Theorem 3. *Let* $\mathcal{A} = \left(m, \sigma^A, \{M_{x_i}^A\}_{x_i \in \mathfrak{X}}, \tau^A\right)$ *and* $\mathcal{B} = \left(n, \sigma^B, \{M_{x_i}^B\}_{x_i \in \mathfrak{X}}, \tau^B\right)$ *be WFAs over* $\mathbb{R}$*, determined 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* $M_{x_i}^B \in \mathbb{R}^{n\times n}$, $\sigma^B \in \mathbb{R}^{1\times n}$, $\tau^B \in \mathbb{R}^{n\times 1}$, $i \in [1..r]$*. The dynamical system* (38) *inline with the ZNN* (1) *generates the TSOL* $\mathbf{x}_{\mathcal{S}}(t)$*, which is stable in the Lyapunov sense.*

Proof. The verification is analogous to the proof of Theorem 1. □

Theorem 4. *Let* $\mathcal{A} = \left(m, \sigma^A, \{M_{x_i}^A\}_{x_i \in \mathfrak{X}}, \tau^A\right)$ *and* $\mathcal{B} = \left(n, \sigma^B, \{M_{x_i}^B\}_{x_i \in \mathfrak{X}}, \tau^B\right)$ *be WFA over* $\mathbb{R}$ *defined upon* $M_{x_i}^A \in \mathbb{R}^{m\times m}$, $\sigma^A \in \mathbb{R}^{1\times m}$, $\tau^A \in \mathbb{R}^{m\times 1}$ *and* $M_{x_i}^B \in \mathbb{R}^{n\times n}$, $\sigma^B \in \mathbb{R}^{1\times n}$, $\tau^B \in \mathbb{R}^{n\times 1}$, $i \in [1..r]$*. Starting from an arbitrary initialization* $\mathbf{x}(0)$*, the ZNNL-hbfb design* (41) *converges exponentially to* $\mathbf{x}^*(t)$*, which coincides with the TSOL of* (5)*.*

Proof. The verification is analogous to the proof of Theorem 2. □

4. Numerical Experiments on the Proposed Models

The behavior of the ZNNL-hfbb (20) and the ZNNL-hbfb (41) are examined in each of the four numerical examinations. During the computation, the *Matlab* `ode45` solver was selected inside the time span $[0, 10]$ under both relative and absolute tolerances equal to 10^{-15}. In addition, the output produced by the ZNN is compared against the results of the *Matlab* functions `linsolve` and `pinv` (with the default settings) in solving (13) in Examples 1 and 2, and solving (34) in Examples 3 and 4. All numerical experiments are performed using the *Matlab* R2022a environment.

Example 1. *Let* $m = 4$, $n = 2$, $r = 2$, $\mathfrak{X} = \{x_1, x_2\}$*, and consider WFAs*

$$\mathcal{A} = \left(4, \sigma^A, \{M_{x_i}^A\}_{x_i \in \mathfrak{X}}, \tau^A\right) \quad \text{and} \quad \mathcal{B} = \left(2, \sigma^B, \{M_{x_i}^B\}_{x_i \in \mathfrak{X}}, \tau^B\right).$$

Accordingly, $M_{x_i}^A \in \mathbb{R}^{4\times 4}$, $\sigma^A \in \mathbb{R}^{1\times 4}$, $\tau^A \in \mathbb{R}^{4\times 1}$ *and* $M_{x_i}^B \in \mathbb{R}^{2\times 2}$, $\sigma^B \in \mathbb{R}^{1\times 2}$, $\tau^B \in \mathbb{R}^{2\times 1}$*. Consider*

$$\sigma^A = \begin{bmatrix} -918/29 & -228/29 & -228/29 & 222/29 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} -1 & -1 & 1 & 1 \end{bmatrix}^{\mathrm{T}},$$

$$M_{x_1}^A = \begin{bmatrix} 3 & 6 & 9 & 12 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ -12 & -9 & -6 & -3 \end{bmatrix}, \quad M_{x_2}^A = \begin{bmatrix} -2 & -4 & -6 & -8 \\ -2 & -2 & -2 & -2 \\ -2 & -2 & -2 & -2 \\ 8 & 6 & 4 & 2 \end{bmatrix}$$

and

$$\sigma^B = \begin{bmatrix} -2 & -4 \end{bmatrix}, \quad \tau^B = \begin{bmatrix} 8 & 8 \end{bmatrix}^{\mathrm{T}}, \quad M_{x_1}^B = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}, \quad M_{x_2}^B = \begin{bmatrix} -2 & -2 \\ -2 & -2 \end{bmatrix}.$$

The gain parameter has been chosen as $\lambda = 10$ *and the initialization conditions (ICs) are equal to:* $IC_1 : \mathbf{x}(0) = \mathbf{1}_{16}$*,* $IC_2 : \mathbf{x}(0) = -\mathbf{1}_{16}$*. It is important to mention that the initialization condition refers to the value of* $\mathbf{x}(t)$ *at* $t = 0$*.*

The results generated by the ZNNL-hfbb are arranged in Figure 1.

Example 2. *Let* $m = 4$, $n = 2$, $r = 3$, $\mathfrak{X} = \{x_1, x_2, x_3\}$, *and examine WFAs over* $\mathbb{R}$ *defined by* $\mathcal{A} = \left(4, \sigma^A, \{M^A_{x_i}\}_{x_i \in \mathfrak{X}}, \tau^A\right)$ *and* $\mathcal{B} = \left(2, \sigma^B, \{M^B_{x_i}\}_{x_i \in \mathfrak{X}}, \tau^B\right)$ *satisfying* $M^A_{x_i} \in \mathbb{R}^{4\times 4}$, $\sigma^A \in \mathbb{R}^{1\times 4}$, $\tau^A \in \mathbb{R}^{4\times 1}$ *and* $M^B_{x_i} \in \mathbb{R}^{2\times 2}$, $\sigma^B \in \mathbb{R}^{1\times 2}$, $\tau^B \in \mathbb{R}^{2\times 1}$. *Let us choose*

$$\sigma^A = \begin{bmatrix} 0 & 57/2 & 57/2 & 12 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} 2 & 2 & 2 & 2 \end{bmatrix}^{\mathrm{T}}, \quad M^A_{x_1} = \begin{bmatrix} 3 & 9 & 6 & 12 \\ 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 \\ -12 & -9 & -6 & -3 \end{bmatrix},$$

$$M^A_{x_2} = \begin{bmatrix} -2 & -6 & -4 & -8 \\ -2 & -2 & -2 & -2 \\ -2 & -2 & -2 & -2 \\ 8 & 6 & 4 & 2 \end{bmatrix}, \quad M^A_{x_3} = \begin{bmatrix} -5 & -15 & -10 & -20 \\ -5 & -5 & -5 & -5 \\ -5 & -5 & -5 & -5 \\ 20 & 15 & 10 & 5 \end{bmatrix}$$

and

$$\sigma^B = \begin{bmatrix} 1 & 2 \end{bmatrix}, \quad \tau^B = \begin{bmatrix} 32 & 40 \end{bmatrix}^{\mathrm{T}}, \quad M^B_{x_1} = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}, \quad M^B_{x_2} = \begin{bmatrix} -2 & -2 \\ -2 & -2 \end{bmatrix}, \quad M^B_{x_3} = \begin{bmatrix} -5 & -5 \\ -5 & -5 \end{bmatrix}.$$

The ZNN gain parameters are chosen as $\lambda = 10$ *and* $\lambda = 100$, *while the IC has been chosen as* $\mathbf{x}(0) = \mathbf{1}_{16}$. *The outputs of ZNNL-hfbb are shown in Figure 1.*

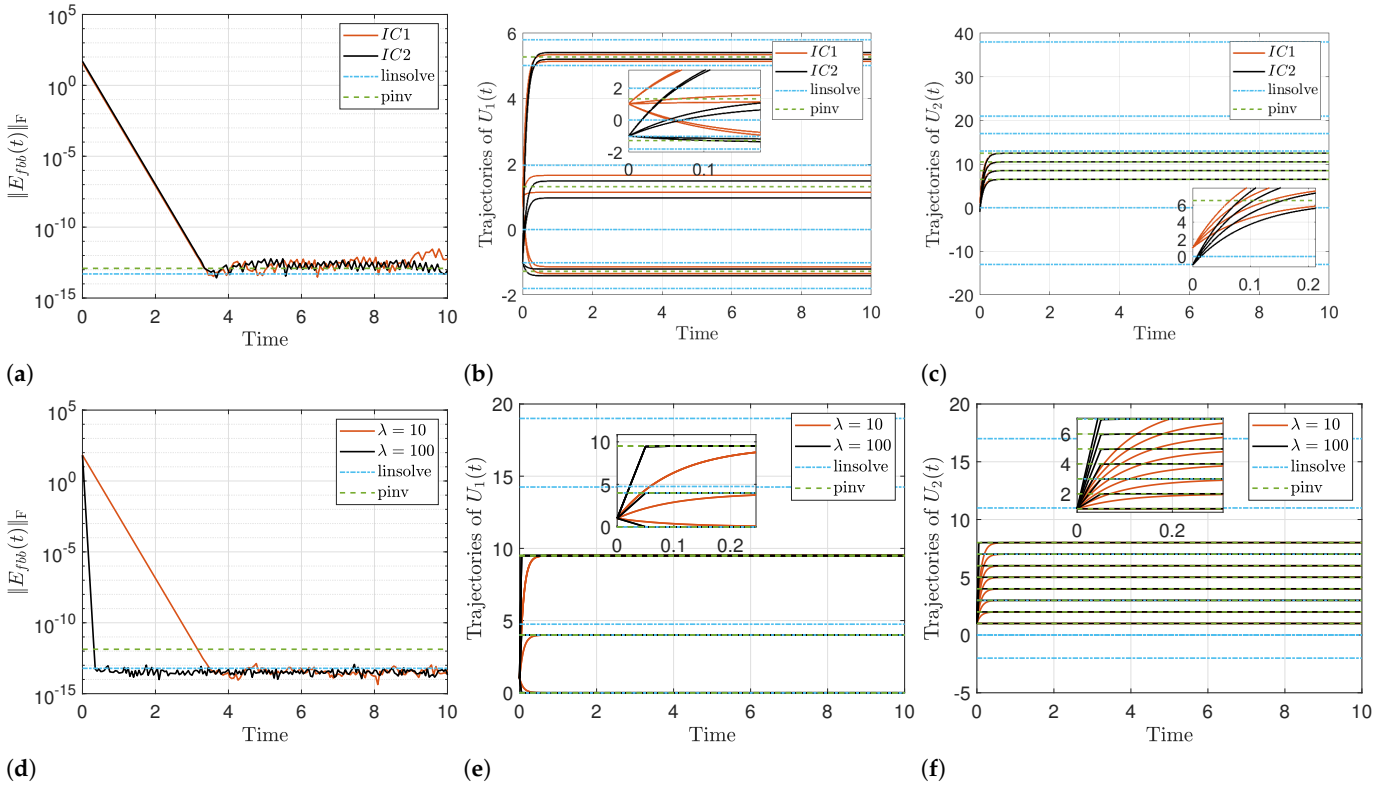

Figure 1. Errors and trajectories generated by ZNNL-hfbb in Examples 1 and 2. (**a**) Example 1: ZEF errors. (**b**) Example 1: Trajectories of $U_1(t)$. (**c**) Example 1: Trajectories of $U_2(t)$. (**d**) Example 2: ZEF errors. (**e**) Example 2: Trajectories of $U_1(t)$. (**f**) Example 2: Trajectories of $U_2(t)$.

Example 3. *Let* $m = 3$, $n = 2$, $r = 2$, $\mathfrak{X} = \{x_1, x_2\}$, *and consider WFAs*

$$\mathcal{A} = \left(3, \sigma^A, \{M^A_{x_i}\}_{x_i \in \mathfrak{X}}, \tau^A\right) \quad \text{and} \quad \mathcal{B} = \left(2, \sigma^B, \{M^B_{x_i}\}_{x_i \in \mathfrak{X}}, \tau^B\right)$$

satisfying $M_{x_i}^A \in \mathbb{R}^{3\times 3}$, $\sigma^A \in \mathbb{R}^{1\times 3}$, $\tau^A \in \mathbb{R}^{3\times 1}$ *and* $M_{x_i}^B \in \mathbb{R}^{2\times n}$, $\sigma^B \in \mathbb{R}^{1\times 2}$, $\tau^B \in \mathbb{R}^{2\times 1}$. *Let us choose*

$$\sigma^A = \begin{bmatrix} 2 & 2 & 2 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} -1/2 & -1/2 & -1/2 \end{bmatrix}^{\mathrm{T}},$$
$$M_{x_1}^A = \begin{bmatrix} -3 & -1 & 1 \\ -1 & -1 & 1 \\ -1 & -1 & 1 \end{bmatrix}, \quad M_{x_2}^A = \begin{bmatrix} -3/2 & -1/2 & 1/2 \\ -1/2 & -1/2 & 1/2 \\ -1/2 & -1/2 & 1/2 \end{bmatrix}$$

and

$$\sigma^B = \begin{bmatrix} 2 & 8 \end{bmatrix}, \quad \tau^B = \begin{bmatrix} -1/2 & -2 \end{bmatrix}^{\mathrm{T}}, \quad M_{x_1}^B = \begin{bmatrix} -1 & -3 \\ -1 & -2 \end{bmatrix}, \quad M_{x_2}^B = \begin{bmatrix} -1/2 & -3/2 \\ -1/2 & -1 \end{bmatrix}.$$

The design parameter has been selected as $\lambda = 10$ *and two ICs have been used:* $IC_1 : \mathbf{x}(0) = \mathbf{1}_{12}$, $IC_2 : \mathbf{x}(0) = -\mathbf{1}_{12}$.

The outputs of the ZNNL-hbfb are arranged in Figure 2.

Example 4. *Let* $m = 3$, $n = 2$, $r = 3$ *and* $\mathfrak{X} = \{x_1, x_2, x_3\}$, *and consider WFAs* $\mathcal{A} = \left(3, \sigma^A, \{M_{x_i}^A\}_{x_i \in \mathfrak{X}}, \tau^A\right)$ *and* $\mathcal{B} = \left(2, \sigma^B, \{M_{x_i}^B\}_{x_i \in \mathfrak{X}}, \tau^B\right)$. *Clearly* $M_{x_i}^A \in \mathbb{R}^{3\times 3}$, $\sigma^A \in \mathbb{R}^{1\times 3}$, $\tau^A \in \mathbb{R}^{3\times 1}$ *and* $M_{x_i}^B \in \mathbb{R}^{2\times 2}$, $\sigma^B \in \mathbb{R}^{1\times 2}$, $\tau^B \in \mathbb{R}^{2\times 1}$. *Consider*

$$\sigma^A = \begin{bmatrix} 2 & 2 & 2 \end{bmatrix}, \quad \tau^A = \begin{bmatrix} -1/2 & -1/2 & -1/2 \end{bmatrix}^{\mathrm{T}},$$
$$M_{x_1}^A = \begin{bmatrix} -3 & -1 & 1 \\ -1 & -1 & 1 \\ -2 & -1 & 1 \end{bmatrix}, \quad M_{x_2}^A = \begin{bmatrix} -3/2 & -1/2 & 1/2 \\ -1/2 & -1/2 & 1/2 \\ -1 & -1/2 & 1/2 \end{bmatrix}, \quad M_{x_3}^A = \begin{bmatrix} -3/4 & -1/4 & 1/4 \\ -1/4 & -1/4 & 1/4 \\ -1/2 & -1/4 & 1/4 \end{bmatrix}$$

as well as

$$\sigma^B = \begin{bmatrix} 2 & 4 \end{bmatrix}, \quad \tau^B = \begin{bmatrix} -1/2 & -1 \end{bmatrix}^{\mathrm{T}},$$
$$M_{x_1}^B = \begin{bmatrix} -1 & -3 \\ -1 & -2 \end{bmatrix}, \quad M_{x_2}^B = \begin{bmatrix} -1/2 & -3/2 \\ -1/2 & -1 \end{bmatrix}, \quad M_{x_3}^B = \begin{bmatrix} -1/4 & -3/4 \\ -1/4 & -1/2 \end{bmatrix}.$$

Gain parameters of ZNN are $\lambda = 10$ *and* $\lambda = 100$, *while the IC is* $\mathbf{x}(0) = \mathbf{1}_{12}$.

The outcomes generated by ZNNL-hbfb are arranged in Figure 2.

Results Analysis

This part presents the results from the four numerical examples that examine the performance of the ZNN models, which are shown in Figures 1 and 2. Particularly, Figures 1a,d and 2a,d show the ZEF errors in Examples 1–4, respectively, while Figures 1b,e and 2b,e show the trajectories of $U_1(t)$, and Figures 1c,f and 2c,f show the trajectories of $U_2(t)$.

According to results generated in Example 1, the next outcomes are observable for the ZNNL-hfbb initiated by IC_1 and IC_2 and forced by $\lambda = 10$. Figure 1a shows the ZNNL-hfbb model's ZEF norms. Both ZNNs are initiated by a large error cost at $t = 0$ and both ZEFs converge in the time span $[10^{-15}, 10^{-13}]$, with an insignificant error at $t = 3.5$. In this way, the ZNNL-hfbb behavior confirms Theorem 2 by the convergence to a result near to zero for two random ICs. Figure 1b,c displays the trajectories generated by $U_1(t)$ and $U_2(t)$, respectively. Included graphs indicate that $U_1(t)$ and $U_2(t)$ do not generate close trajectories initiated by IC_1 and IC_2 , but the convergence speed is similar in both cases. Therefore, the ZNNL-hfbb appears to give dissimilar solutions for a series of ICs, but the convergence behavior of its solutions is proven to match the convergence behavior of the linked ZEFs.

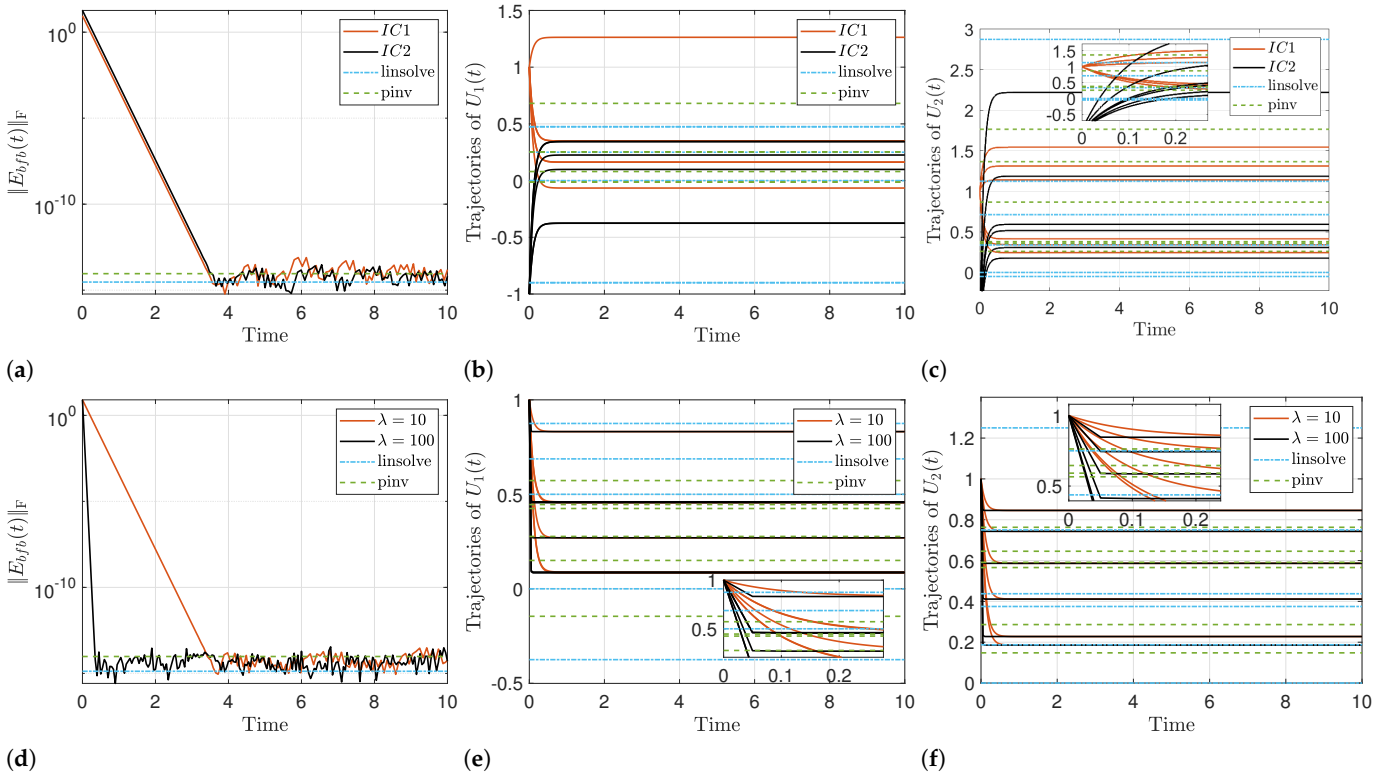

Figure 2. Errors and trajectories generated by ZNNL-hbfb in Examples 3 and 4. (**a**) Example 3: ZEF errors. (**b**) Example 3: Trajectories of $U_1(t)$. (**c**) Example 3: Trajectories of $U_2(t)$. (**d**) Example 4: ZEF errors. (**e**) Example 4: Trajectories of $U_1(t)$. (**f**) Example 4: Trajectories of $U_2(t)$.

In Example 2, under $\lambda = 10$ and $\lambda = 100$, the next observations for the ZNN-hfbb are concluded. Figure 1d shows the norm of ZNN-hfbb's ZEFs. Both evaluations in this figure start with a large error cost at $t = 0$ and converge in $[10^{-15}, 10^{-13}]$ at $t = 0.3$ for $\lambda = 100$, and at $t = 3.5$ for $\lambda = 10$, with a negligible error. Accordingly, the ZNN's convergence features are confirmed by the norm of the ZNN-hfbb's ZEF, which depends on λ. So, the results generated in ZNN-hfbb confirms Theorem 2 by the convergence to a quantity near to zero. Figure 1e and 1f, respectively, show the trajectories of the model's solutions $U_1(t)$ and $U_2(t)$. Obtained results point out that the trajectories of $U_1(t)$ and $U_2(t)$ converge much faster in the environment $\lambda = 100$ than in $\lambda = 10$. Besides that, $U_1(t)$ compared to $U_2(t)$ generates close trajectories in environments $\lambda = 10$ and $\lambda = 100$, individually. So, the ZNN-hfbb produces the same $U_1(t)$ and $U_2(t)$ solutions for different λ, and their convergence behavior coincides with the convergence of the related ZEFs.

In Example 3, the next conclusions for ZNNL-hbfb, beginning with IC_1 and IC_2 for $\lambda = 10$, are observable. Figure 2a shows the ZNNL-hbfb's ZEFs. Both evolutions are started from a large error at $t = 0$ and both ZEFs converge inside the time span $[10^{-16}, 10^{-13}]$ with an insignificant error at $t = 3.5$. Accordingly, the ZNNL-hbfb model confirms Theorem 4 by the convergence to a value near to zero for two distinctive ICs. Figure 2b and 2c, respectively, present the trajectories of $U_1(t)$ and $U_2(t)$. Generated results point out that the trajectories of $U_1(t)$ and $U_2(t)$ are not close in both instances IC_1 and IC_2 , but their convergence behavior is similar. Consequently, the ZNNL-hbfb generates diverse solutions related to different ICs, and its convergence behavior matches with the convergence behavior of the associated ZEFs.

Example 4 initiates the subsequent conclusions regarding ZNN-hbfb under the accelerations $\lambda = 10$ and $\lambda = 100$. Figure 2d exhibits the ZNN-hbfb's ZEFs. Both instances of the ZNN-hbfb evaluation in this figure start with a large error cost at $t = 0$ and converge in $[10^{-16}, 10^{-13}]$ at $t = 0.3$ for $\lambda = 100$, and at $t = 3.5$ for $\lambda = 10$, with a negligible error. In other words, the ZNN's convergence is certified by the ZNN-hbfb's ZEF, which depends

on λ, and the ZNN-hbfb certifies Theorem 4 with its own convergence to a quantity near to zero. Figure 2e and 2f display the trajectories of $U_1(t)$ and $U_2(t)$, respectively. Involved graphs indicate that the trajectories of $U_1(t)$ and $U_2(t)$ converge faster via $\lambda = 100$ compared to $\lambda = 10$. Also, $U_1(t)$ and $U_2(t)$ generated close trajectories via $\lambda = 10$ and $\lambda = 100$ individually. So, the ZNN-hbfb generates the same $U_1(t)$ and $U_2(t)$ for different λ, and its convergence coincides with the convergence pattern of the associated ZEFs.

The following outcomes are obtained when the ZNN-hfbb and ZNN-hbfb models are compared to the *Matlab*'s functions `linsolve` and `pinv`. Particularly, Figures 1a,d and 2a,d demonstrate that for the ZNN-hfbb and ZNN-hbfb models, the `linsolve` and the `pinv` yield similar low error prices in all examples when compared. Additionally, Figure 1b,c demonstrates that the trajectories generated by $U_1(t)$ and $U_2(t)$ are different between the ZNN, the `linsolve`, and `pinv`. Figure 1e,f shows that the ZNN and the `linsolve` create distinct trajectories for $U_1(t)$ and $U_2(t)$, while the ZNN and the `pinv` create identical trajectories. Figure 2b,c demonstrates that the trajectories generated by $U_1(t)$ and $U_2(t)$ are different between the ZNN, compared to `linsolve` and `pinv`. Figure 2e,f shows that the ZNN, the `linsolve`, and `pinv` all create distinct trajectories for $U_1(t)$ and $U_2(t)$. Therefore, in all examples, the ZNN-hfbb and ZNN-hbfb models perform similarly with `linsolve` and `pinv`.

The following conclusions can be drawn from the previously indicated analysis of the numerical examples:

- The ZNNL-hfbb and ZNNL-hbfb models can efficiently solve the systems (4) and (5), respectively.
- All models' behaviors are conditioned by values λ and their solutions are conditioned by ICs.
- Comparing considered ZNNs against the `linsolve` and `pinv`, it is discovered that both the ZNN-hfbb and ZNN-hbfb models exhibit comparable performance to the `linsolve` and `pinv`.
- The approximation of the TSOL $\mathbf{x}^*(t)$ in the ZNNL-hfbb and ZNNL-hbfb models is achieved faster via $\lambda = 100$ than via $\lambda = 10$.

When everything is considered, the ZNNL-hfbb and ZNNL-hbfb models perform in an appropriate and efficient manner in finding the solution of (4) and (5), respectively.

In conclusion, as the ZNN design parameter λ increases, the TSOL $\mathbf{x}^*(t)$ is approximated more quickly. Therefore, it is suggested to set the parameter λ as high as the hardware will allow.

5. Concluding Remarks

Current investigation is aimed at investigating and solving the equivalence and partial k-equivalence problem between WFAs, i.e., determining whether two WFAs generate the same word function or word functions that coincide on all input words whose lengths do not exceed positive integer value k. Our approach is based on the unification of two principal scientific areas, namely the ZNN dynamical systems and the existence of (approximate) heterotypic bisimulations between WFAs over $\mathbb{R}$. Two types of quantitative heterotypic bisimulations are proposed as solutions to particular systems of matrix-vector equations over $\mathbb{R}$. As a result, presented research is aimed to the development and analysis of two original ZNN models, called as ZNNL-hbfb and ZNNL-hfbb, for finding approximate solutions of matrix-vector equations involved in considered heterotypic bisimulations. A convergence analysis is given. Simulation examples are executed under various initialization states. Comparison with the *Matlab* linear programming solver `linprog` and the pseudoinverse solution generated by the standard function `pinv` is shown and superior achievements of the ZNN dynamics are recorded. The simulation examples also revealed another significant finding for the suggested ZNN models: the TSOL is approximated faster as the ZNN design parameter λ increases. The models solved in actual research utilize the ZNN dynamics established upon a larger number of equations and initiated ZEFs and continue research from [38,55,56].

Further research can be developed in the direction of solving minimization problems aimed to finding a WFA with the minimal number of states equivalent to the given WFA. Another optimization problem could be based on finding solutions of the corresponding systems of matrix-vector inequalities and equations of minimal matrix rank. Further research could be aimed at the topic of solving the k-equivalence problems with more than two unknown matrices. Additionally, since all kinds of noise significantly affect the accuracy of the suggested ZNN techniques, it is important to note that the suggested ZNN models have the drawback of being noise intolerant. Future work might therefore concentrate on modifying these models for ZNN dynamical systems that handle noise and improve integration. Finally, the ZNN models developed in this paper give a universal principle for solving arbitrary systems of matrix-vector equations and for solving arbitrary problems arising from such systems.

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

Funding: This work was supported by the Ministry of Science and Higher Education of the Russian Federation (Grant No. 075-15-2022-1121).

Data Availability Statement: The data supporting this study will be available on request to the authors.

Acknowledgments: Predrag Stanimirović and Miroslav Ćirić acknowledge the support by the Science Fund of the Republic of Serbia, Grant No. 7750185, Quantitative Automata Models: Fundamental Problems and Applications – QUAM. Predrag Stanimirović and Miroslav Ćirić are also supported by the Ministry of Science, Technological Development and Innovation, Republic of Serbia, Contract No. 451-03-65/2024-03/200124. Research of G.V.M. was partly supported by the Serbian Academy of Sciences and Arts (Project Φ-96). The author Milena J. Petrović gratefully acknowledges support from the Project supported by Ministry of Education, Science and Technological Development of the Republic of Serbia project no. 451-03-65/2024-03/200123.

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

References

1. Daviaud, L. Containment and Equivalence of Weighted Automata: Probabilistic and Max-Plus Cases. In *Language and Automata Theory and Applications. LATA 2020*; Lect. Notes Comput. Sci; Leporati, A., Martín-Vide, C., Shapira, D., Zandron, C., Eds.; Springer: Cham, Switzerland, 2020; Volume 12038, pp. 17–32.
2. Almagor, S.; Boker, U.; Kupferman, O. What's decidable about weighted automata? *Inf. Comput.* **2022**, *282*, 104651. [CrossRef]
3. Milner, R. *A Calculus of Communicating Systems*; Lect. Notes Comput. Sci; Springer: Berlin/Heidelberg, Germany, 1980; Volume 92.
4. Park, D. Concurrency and automata on infinite sequences. In *Theoretical Computer Science*; Deussen, P., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1981; Volume 104, pp. 167–183.
5. van Benthem, J. Modal Correspondence Theory. Ph.D. Thesis, Universiteit van Amsterdam, Amsterdam, The Netherland, 1976.
6. van Benthem, J. Correspondence Theory. In *Handbook of Philosophical Logic*; Gabbay, D., Guenthner, F., Eds.; Springer: Berlin/Heidelberg, Germany, 200; Volume 3, pp. 325–408.
7. Blackburn, P.; de Rijke, M.; Venema, Y. *Modal Logic*; Cambridge University Press: Cambridge, UK, 2001.
8. Sangiorgi, D. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.* **2009**, *31*, 111–151. [CrossRef]
9. Sangiorgi, D. Origins of bisimulation and coinduction. In *Advanced Topics in Bisimulation and Coinduction*; Sangiorgi, D., Rutten, J., Eds.; Cambridge University Press: Cambridge, UK, 2012; pp. 1–37.
10. Sangiorgi, D.; Rutten, J., Eds. *Advanced Topics in Bisimulation and Coinduction*; Cambridge University Press: Cambridge, UK, 2012.
11. Pous, D.; Sangiorgi, D. Bisimulation and coinduction enhancements: A historical perspective. *Form. Asp. Comput.* **2019**, *31*, 733–749. [CrossRef]
12. Milner, R. *Communication and Concurrency*; Prentice-Hall: Upper Saddle River, NJ, USA, 1989.
13. Milner, R. *Communicating and Mobile Systems: The p-Calculus*; Cambridge University Press: Cambridge, UK, 1999.
14. Roggenbach, M.; Majster-Cederbaum, M. Towards a unified view of bisimulation: A comparative study. *Theor. Comput. Sci.* **2000**, *238*, 81–130. [CrossRef]

15. Aceto, L.; Ingolfsdottir, A.; Larsen, K.G.; Srba, J. *Reactive Systems: Modelling, Specification and Verification*; Cambridge University Press: Cambridge, UK, 2007.
16. Cassandras, C.G.; Lafortune, S. *Introduction to Discrete Event Systems*; Springer: New York, NY, USA, 2008.
17. Ćirić, M.; Ignjatović, J.; Damljanović, N.; Bašić, M. Bisimulations for fuzzy automata. *Fuzzy Sets Syst.* **2012**, *186*, 100–139. [CrossRef]
18. 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. [CrossRef]
19. Damljanović, N.; Ćirić, M.; Ignjatović, J. Bisimulations for weighted automata over an additively idempotent semiring. *Theor. Comput. Sci.* **2014**, *534*, 86–100. [CrossRef]
20. Ćirić, M.; Micić, I.; Matejić, J.; Stamenković, A. Simulations and bisimulations for max-plus automata. *Discret. Event Dyn. Syst.* **2024**, *34*, 269–295. [CrossRef]
21. Stanimirović, P.S.; Ćiri, 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. [CrossRef]
22. Ćirić, M.; Ignjatović, J.; Stanimirović, P.S. Bisimulations for weighted finite automata over semirings. *Res. Sq.* 2022, *submitted to Soft Computing*. [CrossRef]
23. Ésik, Z.; Kuich, W. A generalization of Kozen's axiomatization of the equational theory of the regular sets. In *Words, Semigroups, and Transductions*; Ito, M., Paun, G., Yu, S., Eds.; World Scientific: River Edge, NJ, USA, 2001; pp. 99–114.
24. Beal, M.P.; Lombardy, S.; Sakarovitch, J. On the equivalence of Z-automata, In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005*; Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3580, pp. 397–409.
25. Beal, M.P.; Lombardy, S.; Sakarovitch, J. Conjugacy and equivalence of weighted automata and functional transducers. In *Computer Science—Theory and Applications, First International Symposium on Computer Science in Russia, CSR 2006*; Grigoriev, D., Harrison, J., Hirsch, E.A., Eds.; Lect. Notes Comput. Sci; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3967, pp. 58–69.
26. Buchholz, P. Bisimulation relations for weighted automata. *Theor. Comput. Sci.* **2008**, *393*, 109–123. [CrossRef]
27. Ésik, Z.; Maletti, A. Simulation vs. equivalence. *arXiv* **2010**, arXiv:1004.2426. [CrossRef]
28. Sakarovitch, J. *Elements of Automata Theory*; Cambridge University Press: Cambridge, UK, 2009.
29. Sakarovitch, J. Automata and rational expressions. In *Handbook of Automata Theory*; Pin, J.É., Ed.; European Mathematical Society Publishing House: Berlin/Heidelberg, Germany, 2021; Volume 1, pp. 39–78.
30. Jančić, I., Weak bisimulations for fuzzy automata. *Fuzzy Sets Syst.* **2014**, *249*, 49–72. [CrossRef]
31. Stanimirović, S.; Micić, I. On the solvability of weakly linear systems of fuzzy relation equations. *Inf. Sci.* **2022**, *607*, 670–687. [CrossRef]
32. Micić, I.; Nguyen, L.A.; Stanimirović, S. Characterization and computation of approximate bisimulations for fuzzy automata. *Fuzzy Sets Syst.* **2022**, *442*, 331–350. [CrossRef]
33. Nguyen, L.A. Fuzzy simulations and bisimulations between fuzzy automata. *Int. J. Approx. Reason.* **2023**, *155*, 113–131. [CrossRef]
34. Nguyen, L.A.; Micić, I.; Stanimirović, S. Fuzzy minimax nets. *IEEE Trans. Fuzzy Syst.* **2023**, *31*, 2799–2808. [CrossRef]
35. Nguyen, L.A.; Micić, I.; Stanimirović, S. Depth-bounded fuzzy simulations and bisimulations between fuzzy automata. *Fuzzy Sets Syst.* **2023**, *473*, 108729. [CrossRef]
36. Zhang, Y.; Ge, S.S. Design and analysis of a general recurrent neural network model for time-varying matrix inversion. *IEEE Trans. Neural Netw.* **2005**, *16*, 1477–1490. [CrossRef]
37. Chai, Y.; Li, H.; Qiao, D.; Qin, S.; Feng, J. A neural network for Moore-Penrose inverse of time-varying complex-valued matrices. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 663–671. [CrossRef]
38. 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. [CrossRef]
39. Qiao, S.; Wang, X.Z.; Wei, Y. Two finite-time convergent Zhang neural network models for time-varying complex matrix Drazin inverse. *Linear Algebra Appl.* **2018**, *542*, 101–117. [CrossRef]
40. Qiao, S.; Wei, Y.; Zhang, X. Computing time-varying ML-weighted pseudoinverse by the Zhang neural networks. *Numer. Funct. Anal. Optim.* **2020**, *41*, 1672–1693. [CrossRef]
41. Kovalnogov, V.N.; Fedorov, R.V.; Demidov, D.A.; Malyoshina, M.A.; Simos, T.E.; Katsikis, V.N.; Mourtas, S.D.; Sahas, R.D. Zeroing neural networks for computing quaternion linear matrix equation with application to color restoration of images. *AIMS Math.* **2023**, *8*, 14321–14339. [CrossRef]
42. Abbassi, R.; Jerbi, H.; Kchaou, M.; Simos, T.E.; Mourtas, S.D.; Katsikis, V.N. Towards higher-order zeroing neural networks for calculating quaternion matrix inverse with application to robotic motion tracking. *Mathematics* **2023**, *11*, 2756. [CrossRef]
43. Cao, M.; Xiao, L.; Zuo, Q.; Tan, P.; He, Y.; Gao, X. A fixed-time robust ZNN model with adaptive parameters for redundancy resolution of manipulators. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 3886–3898. [CrossRef]
44. Xiao, L.; Cao, P.; Song, W.; Luo, L.; Tang, W. A fixed-time noise-tolerance ZNN model for time-variant inequality-constrained quaternion matrix least-squares problem. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 10503–10512. [CrossRef]
45. Jin, L.; Zhang, Y.; Li, S.; Zhang, Y. Modified ZNN for time-varying quadratic programming with inherent tolerance to noises and its application to kinematic redundancy resolution of robot manipulators. *IEEE Trans. Ind. Electron.* **2016**, *63*, 6978–6988. [CrossRef]

46. Wang, T.; Zhang, Z.; Huang, Y.; Liao, B.; Li, S. Applications of Zeroing Neural Networks: A Survey. *IEEE Access* **2024**, *12*, 51346–51363. [CrossRef]
47. Zhang, Y.; Guo, D. *Zhang Functions and Various Models*; Springer: Berlin/Heidelberg, Germany, 2015.
48. Li, L.; Xiao, L.; Wang, Z.; Zuo, Q. A survey on zeroing neural dynamics: Models, theories, and applications. *Int. J. Syst. Sci.* **2024**, 1–34. [CrossRef]
49. Jin, L.; Li, S.; Liao, B.; Zhang, Z. Zeroing neural networks: A survey. *Neurocomputing* **2017**, *267*, 597–604. [CrossRef]
50. 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. [CrossRef]
51. Guo, D.; Yi, C.; Zhang, Y. Zhang neural network versus gradient-based neural network for time-varying linear matrix equation solving. *Neurocomputing* **2011**, *74*, 3708–3712. [CrossRef]
52. Li, Z.; Zhang, Y. Improved Zhang neural network model and its solution of time-varying generalized linear matrix equations. *Expert Syst. Appl.* **2010**, *37*, 7213–7218. [CrossRef]
53. Zhang, Y.; Chen, K. Comparison on Zhang neural network and gradient neural network for time-varying linear matrix equation $AXB = C$ solving. In Proceedings of the 2008 IEEE International Conference on Industrial Technology—ICIT, Chengdu, China, 21–24 April 2008; pp. 1–6. [CrossRef]
54. Xiao, L.; Liao, B.; Li, S.; Chen, K. Nonlinear recurrent neural networks for finite-time solution of general time-varying linear matrix equations. *Neural Netw.* **2018**, *98*, 102–113. [CrossRef]
55. 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. [CrossRef]
56. 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. [CrossRef]
57. Graham, A. *Kronecker Products and Matrix Calculus with Applications*; Courier Dover Publications: Mineola, NY, USA, 2018.

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.