



UNIVERSITY OF NIŠ  
FACULTY OF SCIENCES AND MATHEMATICS



Ivan S. Živković

**RECURRENT NEURAL NETWORKS  
FOR SOLVING  
MATRIX ALGEBRA PROBLEMS**

DOCTORAL DISSERTATION

Niš, 2018.





УНИВЕРЗИТЕТ У НИШУ  
ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ



**Иван С. Живковић**

**РЕКУРЕНТНЕ НЕУРОНСКЕ МРЕЖЕ  
ЗА РЕШАВАЊЕ ПРОБЛЕМА  
ЛИНЕАРНЕ АЛГЕБРЕ**

ДОКТОРСКА ДИСЕРТАЦИЈА

Ниш, 2018.



## Data on Doctoral Dissertation

Doctoral Supervisor: PhD, Predrag S. Stanimirović, Full professor, Faculty of Science and Mathematics, University of Niš

Title: Recurrent neural networks for solving matrix algebra problems

Abstract:

The aim of this dissertation is the application of recurrent neural networks (RNNs) to solving some problems from a matrix algebra with particular reference to the computations of the generalized inverses as well as solving the matrix equations of constant (time-invariant) matrices. We examine the ability to exploit the correlation between the dynamic state equations of recurrent neural networks for computing generalized inverses and integral representations of these generalized inverses. Recurrent neural networks are composed of independent parts (sub-networks). These sub-networks can work simultaneously, so parallel and distributed processing can be accomplished. In this way, the computational advantages over the existing sequential algorithms can be attained in real-time applications. We investigate and exploit an analogy between the scaled hyperpower family (SHPI family) of iterative methods for computing the matrix inverse and the discretization of Zhang Neural Network (ZNN) models. A class of ZNN models corresponding to the family of hyperpower iterative methods for computing the generalized inverses on the basis of the discovered analogy is defined. The Matlab Simulink implementation of the introduced ZNN models is described in the case of scaled hyperpower methods of the order 2 and 3. We present the Matlab Simulink model of a hybrid recursive neural implicit dynamics and give a simulation and comparison to the existing Zhang dynamics for real-time matrix inversion. Simulation results confirm a superior convergence of the hybrid model compared to Zhang model.

Scientific Field: Computer science

Scientific Discipline: Artificial neural networks, dynamical systems, control systems

Key Words: Artificial neural networks, dynamical systems, generalized inverses

UDC: 004.832:[512.64+517.98+519.857]

CERIF Classification: P170: Computer science, numerical analysis, systems, control

Creative Commons License Type:

**CC BY-NC-ND**



## Подаци о докторској дисертацији

Ментор:	Др Предраг С. Станимировић, редовни професор, Природно-математички факултет, Универзитет у Нишу
Наслов:	Рекурентне неуронске мреже за решавање проблема линеарне алгебре
Резиме:	<p>Предмет ове дисертације јесте проучавање примена рекурентних неуронских мрежа на неке проблеме матричне алгебре, са посебним освртом на проблем израчунавања генералисаних инверза и матричних једначина. Испитује се могућност употребе корелације између једначина динамичког стања рекурентних неуронских мрежа за израчунавање уопштених инверза и интегралних репрезентација уопштених инверза. Дефинисане рекурентне неуронске мреже састављене су од независних делова (подмрежа). Ове подмреже могу радити истовремено, па се тиме може постићи паралелна и дистрибуирана обрада. На овај начин може се остварити предност у брзини израчунавања над постојећим секвенцијалним алгоритмима. Истражује се и аналогија између итеративних метода за израчунавање регуларних инверза и дискретизације модела Жангових неуронских мрежа. На основу откривене аналогије, одређује се класа Жангових неуронских мрежа која одговара породици итеративних метода за израчунавање уопштених инверза. Описана је <i>Matlab Simulink</i> имплементација уведених модела скалираних итеративних метода реда 2 и 3. Представљен је <i>Matlab Simulink</i> модел хибридне рекурентне имплицитне динамике, и дата је симулација и компарација са постојећом Жанговом динамиком за одређивање матричне инверзије у реалном времену. Резултати симулације потврђују супериорну конвергенцију хибридног модела у поређењу са Жанговим моделом.</p>
Научна област:	Рачунарске науке
Научна дисциплина:	Вештачке неуронске мреже, динамички системи, системи управљања
Кључне речи:	Вештачке неуронске мреже, динамички системи, уопштени инверзи
УДК:	004.832:[512.64+517.98+519.857]
CERIF класификација:	P170: Computer science, numerical analysis, systems, control
Тип лиценце Креативне заједнице:	CC BY-NC-ND







**ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ  
НИШ**

**КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА**

Редни број, <b>РБР:</b>	
Идентификациони број, <b>ИБР:</b>	
Тип документације, <b>ТД:</b>	монографска
Тип записа, <b>ТЗ:</b>	текстуални / графички
Врста рада, <b>ВР:</b>	докторска дисертација
Аутор, <b>АУ:</b>	Иван С. Живковић
Ментор, <b>МН:</b>	Предраг С. Станимировић
Наслов рада, <b>НР:</b>	РЕКУРЕНТНЕ НЕУРОНСКЕ МРЕЖЕ ЗА РЕШАВАЊЕ ПРОБЛЕМА ЛИНЕАРНЕ АЛГЕБРЕ
Језик публикације, <b>ЈП:</b>	енглески
Језик извода, <b>ЈИ:</b>	енглески
Земља публикавања, <b>ЗП:</b>	Србија
Уже географско подручје, <b>УГП:</b>	Србија
Година, <b>ГО:</b>	2018.
Издавач, <b>ИЗ:</b>	ауторски репринт
Место и адреса, <b>МА:</b>	Ниш, Вишеградска 33.
Физички опис рада, <b>ФО:</b> <small>(поглавља/страна/ цитата/табела/слика/графика/прилога)</small>	172 стр., граф. прикази
Научна област, <b>НО:</b>	рачунарске науке
Научна дисциплина, <b>НД:</b>	вештачке неуронске мреже, динамички системи, системи управљања
Предметна одредница/Кључне речи, <b>ПО:</b>	вештачке неуронске мреже, динамички системи, уопштени инверзи
<b>УДК</b>	004.832:[512.64+517.98+519.857]
Чува се, <b>ЧУ:</b>	библиотека
Важна напомена, <b>ВН:</b>	
Извод, <b>ИЗ:</b>	Предмет ове дисертације јесте проучавање примена рекурентних неуронских мрежа на неке проблеме матричне алгебре, са посебним освртом на проблем израчунавања генералисаних инверза и матричних једначина. Испитује се могућност употребе корелације између једначина динамичког стања рекурентних неуронских мрежа за израчунавање уопштених инверза и интегралних репрезентација уопштених инверза.
Датум прихватања теме, <b>ДП:</b>	08. 05. 2017.
Датум одбране, <b>ДО:</b>	
Чланови комисије, <b>КО:</b>	Председник:
	Члан:
	Члан, ментор:





**ПРИРОДНО - МАТЕМАТИЧКИ ФАКУЛТЕТ  
НИШ**

**KEY WORDS DOCUMENTATION**

Accession number, <b>ANO</b> :	
Identification number, <b>INO</b> :	
Document type, <b>DT</b> :	monograph
Type of record, <b>TR</b> :	textual / graphic
Contents code, <b>CC</b> :	doctoral dissertation
Author, <b>AU</b> :	Ivan S. Živković
Mentor, <b>MN</b> :	Predrag S. Stanimirović
Title, <b>TI</b> :	RECURRENT NEURAL NETWORKS FOR SOLVING MATRIX ALGEBRA PROBLEMS
Language of text, <b>LT</b> :	English
Language of abstract, <b>LA</b> :	English
Country of publication, <b>CP</b> :	Serbia
Locality of publication, <b>LP</b> :	Serbia
Publication year, <b>PY</b> :	2018
Publisher, <b>PB</b> :	author's reprint
Publication place, <b>PP</b> :	Niš, Višegradska 33.
Physical description, <b>PD</b> : <small>(chapters/pages/ref./tables/pictures/graphs/appendixes)</small>	172 p. ; graphic representations
Scientific field, <b>SF</b> :	Computer science
Scientific discipline, <b>SD</b> :	Artificial neural networks, dynamical systems, control systems
Subject/Key words, <b>S/KW</b> :	Artificial neural networks, dynamical systems, generalized inverses
<b>UC</b>	004.832:[512.64+517.98+519
Holding data, <b>HD</b> :	library
Note, <b>N</b> :	
Abstract, <b>AB</b> :	The aim of this dissertation is the application of recurrent neural networks (RNNs) to solving some problems from a matrix algebra with particular reference to the computations of the generalized inverses as well as solving the matrix equations of constant (time-invariant) matrices. We examine the ability to exploit the correlation between the dynamic state equations of recurrent neural networks for computing generalized inverses and integral representations of these generalized inverses.
Accepted by the Scientific Board on, <b>ASB</b> :	08.05.2017.
Defended on, <b>DE</b> :	
Defended Board, <b>DB</b> :	President:
	Member:
	Member, Mentor:



## **Acknowledgements**

The research presented in this PhD thesis could not be performed without the assistance, patience and support of many individuals. First and foremost, I would like to express the deepest appreciation to my thesis advisor, Professor Predrag Stanimirović for mentoring me during the course of my undergraduate and graduate studies. He helped me through extremely difficult times over the course of the analysis and the writing of the thesis and I sincerely thank him for his confidence in me.

I would additionally like to thank Professor Yimin Wei for his support in both the research and especially the revision process of our joint papers. Further, I would like to thank Professor Igor Stojanović, Dimitrios Gerontitis and Xue-Zhong Wang for the excellent cooperation.

I would also like to extend my appreciation to Professor Branimir Todorović for introducing me to the field of the artificial neural networks and inspiring me to choose this area.

This research would not be completed without the assistance of my friend Alexandar Vukojević who provided several illustrations that found their place in this dissertation. In particular, I would like to thank Nina Milić for her work on the text. Her knowledge and understanding of the written word has allowed me to fully express the concepts behind this research.

I would like to extend my deepest gratitude to all of my friends and to all colleagues from Accordia Group, LLC.

Finally, special thanks goes to my parents, my mother Sladjana and my father Srbislav, without whose love, support and understanding I could never have completed this doctoral degree.



# Contents

<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>iv</b>
<b>List of Tables</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Basic notions from matrix theory . . . . .	2
1.1.1 Full rank and Jordan decomposition . . . . .	3
1.1.2 Singular value decomposition . . . . .	4
1.1.3 Idempotent matrices and projectors . . . . .	6
1.2 Main classes of generalized inverses . . . . .	7
1.2.1 The inverse of a nonsingular matrix . . . . .	7
1.2.2 Solvability of linear systems . . . . .	7
1.2.3 Definitions and main properties of generalized inverses . . . . .	8
1.3 Motivation and organization of the dissertation . . . . .	19
1.3.1 GNN vs ZNN dynamics . . . . .	21
1.4 Review of known GNN models . . . . .	22
1.4.1 GNN for regular inverse . . . . .	23
1.4.2 GNN for computing the Moore-Penrose inverse . . . . .	24
1.4.3 GNN for computing the weighted Moore-Penrose inverse . . . . .	25
1.4.4 GNN dynamics for solving matrix equations . . . . .	26
1.5 Review of known ZNN models . . . . .	30
1.5.1 ZNN for computing regular inverse . . . . .	30
1.5.2 ZNN for computing Moore-Penrose inverse . . . . .	31
1.5.3 ZNN for computing the Drazin inverse . . . . .	31
1.5.4 ZNN for computing outer inverse . . . . .	31
1.6 Outline of the dissertation . . . . .	32

<b>2</b>	<b>GNN for computing generalized inverses with restrictions on spectrum</b>	<b>35</b>
2.1	GNN for computing the Drazin Inverse . . . . .	35
2.1.1	Preliminaries and motivation . . . . .	35
2.1.2	Neural network architecture for computing Drazin inverse . . . . .	36
2.1.3	Illustrative examples for GNND model . . . . .	48
2.1.4	Application of the GNND model . . . . .	57
2.2	GNN models for computing outer inverse . . . . .	60
2.2.1	Preliminaries and motivation . . . . .	60
2.2.2	Neural network architecture . . . . .	61
2.2.3	Particular cases of GNNGA and GNNAG model . . . . .	66
2.2.4	Illustrative examples for GNNGA model . . . . .	67
2.3	GNN for computing the $W$ -weighted Drazin inverse . . . . .	72
2.3.1	About $W$ -weighted Drazin inverse . . . . .	72
2.3.2	Specific case for $W$ -weighted Drazin inverse . . . . .	73
<b>3</b>	<b>GNN for computing generalized inverse without restriction on spectrum</b>	<b>79</b>
3.1	Globally convergent GNN for computing Drazin inverse . . . . .	79
3.1.1	Preliminaries and motivation . . . . .	79
3.1.2	Neural network architecture . . . . .	80
3.1.3	Illustrative examples . . . . .	86
3.2	Globally convergent GNN for computing outer inverse . . . . .	96
3.2.1	Preliminaries and motivation . . . . .	96
3.2.2	Neural Network Architecture . . . . .	97
3.2.3	Illustrative Examples . . . . .	98
3.3	Globally convergent GNN for computing $W$ -weighted Drazin inverse . . . . .	102
3.3.1	Dynamic equation with global convergence . . . . .	102
3.3.2	Convergence and stability analysis of GNNDW0 . . . . .	104
<b>4</b>	<b>GNN for computing outer inverses based on the full rank representation</b>	<b>107</b>
4.1	Preliminaries and motivation . . . . .	107
4.2	On the existence and representations of outer inverses . . . . .	110
4.3	Neural networks based on full rank representation of outer inverses . . . . .	112
4.3.1	Neural network RNN(4.2.4) based on (4.2.4) . . . . .	112
4.3.2	Neural network RNN(4.2.3) based on (4.2.3) . . . . .	115
4.3.3	Relationships between different RNNs . . . . .	116
4.4	Numerical experiments on GNN based on full rank representation . . . . .	117



<b>5</b>	<b>ZNN for computing matrix inverse based on hyperpower iterative methods</b>	<b>125</b>
5.1	Introduction to ZNN design and known ZNN models . . . . .	125
5.2	Correlation between iterations and ZNN models . . . . .	127
5.3	Scaled Hyperpower iterations as discretized ZNN models . . . . .	128
5.4	Neural network architecture of ZNNCM model . . . . .	131
5.5	Convergence of the ZNNCM model . . . . .	134
5.6	Simulation results and its comparison . . . . .	136
<b>6</b>	<b>Matlab simulation of the hybrid neural dynamics for online matrix inversion</b>	<b>145</b>
6.1	Preliminaries and motivation . . . . .	145
6.2	Model formulation . . . . .	146
6.2.1	Gradient-based dynamics . . . . .	146
6.2.2	Zhang dynamics . . . . .	147
6.2.3	Improved ZNN model for matrix inversion . . . . .	147
6.3	Simulation results and its comparison . . . . .	148
<b>7</b>	<b>Conclusion</b>	<b>155</b>
<b>A</b>	<b>Biography</b>	<b>169</b>
<b>B</b>	<b>Dissertation documentation</b>	<b>171</b>



# List of Figures

1.1	Behavior of the four basic types of activation functions . . . . .	24
2.1	Architecture of the RNN for computing the Drazin inverse . . . . .	47
2.2	Convergence behavior of the RNN in $10^{-6}$ seconds for Example 2.1.1 . . . . .	49
2.3	Convergence behavior of the RNN in $10^{-6}$ seconds for Example 2.1.2 . . . . .	51
2.4	Convergence behavior of the RNN in $10^{-6}$ seconds for Example 2.1.3 . . . . .	52
2.5	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 2.1.4 . . . . .	54
2.6	Convergence behavior of the RNN in $10^{-6}$ seconds for Example 2.1.5 . . . . .	55
2.7	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 2.1.9 . . . . .	59
2.8	Architecture of the RNN for computing outer inverse in case $m \geq n$ . . . . .	66
2.9	Architecture of the RNN for computing outer inverse in case $m < n$ . . . . .	67
2.10	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 2.2.1 . . . . .	69
2.11	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 2.2.2 . . . . .	70
2.12	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 2.2.3 . . . . .	71
3.1	Architecture of the RNN for computing the Drazin inverse . . . . .	85
3.2	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 3.1.1. . . . .	87
3.3	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 3.1.2. . . . .	88
3.4	Convergence of $\ V(t) - A^D\ $ for three different values $\gamma = \gamma$ . . . . .	89
3.5	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 3.1.3. . . . .	91
3.6	Convergence behavior of the RNN in $10^{-7}$ seconds for Example 3.1.4. . . . .	92
3.7	Divergence of the RNN in $10^{-7}$ seconds for Example 3.2.1 . . . . .	99
4.1	Convergence behavior of the RNN [138] in $10^{-7}$ seconds for Example 4.4.1 . .	119
4.2	Convergence behavior of the RNN(4.2.4) in $10^{-7}$ seconds for Example 4.4.1 . .	120
5.1	Simulink implementation of the ZNNNM model. . . . .	132
5.2	Simulink implementation of the ZNNCM model. . . . .	133
5.3	Trajectories in $10^{-5}$ seconds under zero initial conditions in the ZNNCM model	138
5.4	Trajectories of the residual errors of the model ZNNCM. . . . .	138
5.5	Trajectories of the residual errors of the models ZNNNM and ZNNCM. . . . .	139

6.1	Simulink implementation of EZNNNM model. . . . .	149
6.2	Trajectories of the errors $\ A^{-1} - X(t)\ $ of $ZNNNM$ and $EZNNNM$ in Example 6.3.1. . . . .	150
6.3	Trajectories of the errors $\ A_1^{-1} - X(t)\ $ of $ZNNNM$ and $EZNNNM$ in Example 6.3.1. . . . .	151
6.4	Trajectories of $X(t)$ of $ZNNNM$ and $EZNNNM$ in Example 6.3.2. . . . .	151
6.5	Trajectories of the errors $\ A_1^{-1} - X(t)\ $ of $ZNNNM$ and $EZNNNM$ in Example 6.3.4. . . . .	153

# List of Tables

2.1	Numerical comparison test with FF[14]	50
2.2	Numerical comparison test with [94]	56
2.3	Numerical comparison test with [136]	56
2.4	Comparison test with [57] in the case $b = b_{Con}$	58
2.5	Comparison test with [57] in the case $b = b_{Gen}$	58
3.1	Numerical comparison test with [84] for Example 3.1.1	96
3.2	Numerical comparison test with [84] for Example 3.1.2	96
4.1	Results for three RNNs generated using on the set of randomly generated matrices	123
4.2	Results for three RNN(4.2.4) generated using $\gamma = 10^{13}$ on the set of randomly generated matrices	123
5.1	Comparison of the models ZNNNM, ZNNCM and GNN.	139
5.1	Comparison of the models ZNNNM, ZNNCM and GNN.	140
5.2	Comparison of the models ZNNNM, ZNNCM and ZNNHM.	140
5.2	Comparison of the models ZNNNM, ZNNCM and ZNNHM.	141
5.3	Comparison of the models ZNNNM, ZNNCM and ZNNHM with Power Sigmoid activation function $\mathcal{H}$ , defined by the parameter $p = 3$ .	142
5.4	Comparison of the models ZNNNM, ZNNCM and ZNNHM with Power Sigmoid activation function defined by the parameter $p = 3$ .	142
5.4	Comparison of the models ZNNNM, ZNNCM and ZNNHM with Power Sigmoid activation function defined by the parameter $p = 3$ .	143
5.5	ZNNNM vs ZNNCM using the power sigmoid activation function.	143
6.1	Comparison of the models ZNNNM and EZNNNM	152



# Chapter 1

## Introduction

The idea of an inverse of a singular matrix seems to have been first presented by Moore in 1920. But, no efficient investigation of the subject was made by the mid 1950s when the investigation of generalized inverses progressed. The study of generalized inverses has progressed significantly since its rebirth in the early 1950s. Roger Penrose, unfamiliar with previous work, reclassified the Moore "reciprocal inverse" in 1955. He concluded that Moore inverse could be represented by four equations (known as Moore-Penrose equations). Another well known kind of generalized inverses is the Drazin inverse, named after Michael P. Drazin (1958). A major extension of this field came in the fifties, when C. R. Rao and J. Chipman exploited the relation between generalized inverses, least squares, and statistics.

Various papers have arisen both in theory of generalized inverses and its applications. The reason for developing a generalized inverse matrix is to get a matrix that can fill in as the inverse in some sense for a more extensive class of matrices than invertible matrices. Now, it is difficult to give even an approximate number of articles devoted to the theory, computation and application of generalized inverses. It is justifiably to say that the theory of generalized inverses extensively grows and becomes an important part of pure and computational mathematics as well as important part of many applicable scientific areas, such as computer science, electrical engineering, etc.

Now, generalized inverses cover an extensive variety of mathematical fields, for example, matrix theory and operator theory. They show up in various applications such as linear estimation, differential and difference equations, Markov chains, graphics, cryptography, coding theory, incomplete data recovery, and robotics. A special case of the Drazin inverse, called Group inverse, has found application in characterizing the sensitivity of the stationary probabilities to perturbations in the underlying transition probabilities. Finally, the group inverse has recently proven to be fundamental in the analysis of Google's PageRank search engine.

## 1.1 Basic notions from matrix theory

According to the standard notation,  $\mathbb{C}(\mathbb{R})$  denotes the field of complex (real) numbers. Further,  $\mathbb{C}^{m \times n}(\mathbb{R}^{m \times n})$  denote the vector-space of  $m \times n$  complex (real) matrices over  $\mathbb{C}(\mathbb{R})$ ,  $\mathbb{C}_r^{m \times n}(\mathbb{R}_r^{m \times n})$  the class of  $m \times n$  complex (real) matrices of rank  $r$  and  $\mathbb{C}^n(\mathbb{R}^n)$  the vector-space of  $n$ -tuples of complex (real) numbers over  $\mathbb{C}(\mathbb{R})$ . The identity matrix of an appropriate order is denoted by  $I$ . For any  $A \in \mathbb{C}^{m \times n}$ , the range of  $A$  is defined by  $\mathcal{R}(A) = \{y \in \mathbb{C}^m : y = Ax \text{ for some } x \in \mathbb{C}^n\}$ , and  $\mathcal{N}(A) = \{x \in \mathbb{C}^n : Ax = 0\}$  represents the null space of  $A$ . The transpose, conjugate transpose and the rank of  $A \in \mathbb{C}^{m \times n}$  are denoted by  $A^T$ ,  $A^*$ ,  $\text{rank}(A)$  respectively. A matrix  $A$  is Hermitian if its conjugate transpose,  $A^*$ , equals  $A$ . The maximal number of linearly independent columns of a matrix  $A$  is called *rank* of  $A$  and denoted by  $\text{rank}(A)$ . The trace of a square matrix  $A \in \mathbb{C}^{n \times n}$  is denoted by  $\text{Tr}(A)$  and defined as the sum of its diagonal entries:  $\text{Tr}(A) = a_{11} + \cdots + a_{nn}$ .

**Definition 1.1.1.** A square matrix  $A \in \mathbb{C}^{n \times n}$  ( $A \in \mathbb{R}^{n \times n}$ ) is said to be:

- (a) Hermitian (self-adjoint) if  $A^* = A$  ( $A^T = A$ ),
- (b) normal, if  $A^*A = AA^*$  ( $A^T A = AA^T$ ),
- (c) lower-triangular, if  $a_{ij} = 0$  for  $i > j$ ,
- (d) upper-triangular, if  $a_{ij} = 0$  for  $i < j$ ,
- (e) positive semi-definite, if  $\text{Re}(x^*Ax) \geq 0$  for all  $x \in \mathbb{C}^n$ ,
- (f) positive definite, if  $\text{Re}(x^*Ax) > 0$  for all  $x \in \mathbb{C}^n \setminus \{0\}$ .

The notion  $\text{Re}(z)$  (resp.  $\text{Im}(z)$ ) means the real (resp. imaginary) part of a complex number  $z$ .

**Definition 1.1.2.** Let  $A \in \mathbb{C}^{m \times n}$ . A real or complex scalar  $\lambda$  which satisfies the equation

$$Ax = \lambda x \iff (A - \lambda I)x = 0,$$

is the eigenvalue of  $A$ , and  $x \in \mathbb{C}^n$  is the eigenvector of  $A$  corresponding to  $\lambda$ .

**Definition 1.1.3.** Let  $A \in \mathbb{C}^{n \times n}$  and  $\lambda$  is an eigenvalue of  $A$ . A vector  $x \in \mathbb{C}^n$  is called generalized eigenvector of  $A$  of a grade  $p$  corresponding to  $\lambda$ , or  $\lambda$ -vector of  $A$  of a grade  $p$  if it satisfies the equation

$$(A - \lambda I)^p x = 0.$$

Matrix index is another important characteristics of matrices. This notion has been frequently used in numerical linear algebra.



**Proposition 1.1.1.** *For every  $A \in \mathbb{C}^{n \times n}$  there exists an integer  $k$  such that  $\text{rank}(A^{k+1}) = \text{rank}(A^k)$ .*

**Definition 1.1.4.** *Let  $A \in \mathbb{C}^{n \times n}$  be a given square matrix. The index of  $A$  is denoted by  $\text{ind}(A) = k$  and defined as the smallest integer  $k$  satisfying  $\text{rank}(A^{k+1}) = \text{rank}(A^k)$ .*

Note that  $\text{ind}(A) = 0$  if  $A$  is regular and otherwise  $\text{ind}(A) \geq 1$ .

### 1.1.1 Full rank and Jordan decomposition

The simplest matrix decompositions are LU and Cholesky factorization (decomposition) These notions are given in Proposition 1.1.2.

**Proposition 1.1.2.** (LU and Cholesky factorization) *For every regular square matrix  $A \in \mathbb{C}^{n \times n}$  there exists lower triangular matrix  $L$  and upper triangular matrix  $U$  such that  $A = LU$  and  $l_{ii} = 1$  for every  $i = 1, 2, \dots, n$ . This factorization is known as LU factorization. Moreover if  $A$  is Hermitian and positive definite, it holds that  $U = L^*$ , and the implied factorization  $A = LL^*$  is called Cholesky factorization.*

The full rank factorization is one of most important notions in the matrix theory.

**Definition 1.1.5.** (Full rank factorization) *A full rank factorization of an arbitrary matrix  $A \in \mathbb{C}^{m \times n}$  is defined as the decomposition  $A = PQ$ , where the matrix  $P \in \mathbb{C}^{m \times r}$  is full column rank matrix and  $Q \in \mathbb{C}^{r \times n}$  is full row rank matrix.*

**Proposition 1.1.3.** *Let  $A \in \mathbb{C}^{m \times n}$  such that  $A$  is of neither full column rank nor full row rank. Then, there exists at least one full rank factorization  $A = PQ$  of the matrix  $A$ .*

For each matrix there exists a basis of generalized eigenvectors with respect to which a matrix can be represented in the Jordan form. The Jordan decomposition is stated in the following proposition.

**Proposition 1.1.4.** (The Jordan decomposition) *Let the matrix  $A \in \mathbb{C}^{n \times n}$  have  $p$  distinct eigenvalues  $\{\lambda_1, \lambda_2, \dots, \lambda_p\}$ . Then  $A$  is similar to a block diagonal matrix  $J$  with Jordan blocks on its diagonal, i.e., there exists a nonsingular matrix  $P$  such that*

$$A = PJP^{-1} = P \begin{bmatrix} J_{k_1}(\lambda_1) & 0 & \dots & 0 \\ 0 & J_{k_2}(\lambda_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & J_k(\lambda_p) \end{bmatrix} P^{-1},$$

where the Jordan blocks  $J_{k_i}(\lambda_i)$  are defined by

$$J_{k_i}(\lambda_i) = \begin{bmatrix} \lambda_i & 1 & 0 & \dots & 0 \\ 0 & \lambda_i & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_i & 1 \\ 0 & 0 & \dots & 0 & \lambda_i \end{bmatrix}$$

and the matrix  $J$  is unique up to a rearrangement of its blocks.

The following definition and proposition give us an alternative way to obtain even simpler decomposition than the Jordan decomposition, but with respect to different basis of  $\mathbb{C}^n$ .

A square matrix  $T$  of the order  $n$  is symmetric and positive semidefinite (abbreviated SPSD and denoted by  $Q \succeq 0$ ) if

$$v^T T v \geq 0 \text{ for all } v \in \mathbb{R}^n.$$

The matrix  $T$  is symmetric and positive definite (abbreviated SPD and denoted by  $Q \succ 0$ ) if

$$v^T T v > 0 \text{ for all } v \in \mathbb{R}^n, v \neq 0.$$

Recall that a symmetric matrix  $T$  is positive definite if and only if all its eigenvalues are non-negative.

**Proposition 1.1.5.** *If  $T \in \mathbb{R}^{n \times n}$  is symmetric PSD matrix, then the following statements are equivalent:*

- (a)  $T = M M^T$ , for an appropriate matrix  $M$  satisfying  $M \in \mathbb{R}^{n \times k}$ ,  $k \geq 1$ .
- (b)  $v^T T v \geq 0$  for all  $v \in \mathbb{R}^n$ ,  $v \neq 0$ .
- (c) There exist vectors  $v_i$ ,  $i = 1, \dots, n \in \mathbb{R}^k$  (for some  $k \geq 1$ ) such that  $T_{ij} = v_i^T v_j$  for all  $i, j = 1, \dots, n$ . The vectors  $v_i$ ,  $i = 1, \dots, n$  are called a Gram representation of  $T$ .
- (d) All principal minors of  $T$  are non-negative.

**Proposition 1.1.6.** *Let  $T \in \mathbb{C}^{n \times n}$  is symmetric. Then  $T \succeq 0$  and it is nonsingular if and only if  $T \succ 0$ .*

## 1.1.2 Singular value decomposition

Besides the Jordan decomposition, the Singular Value Decomposition (or SVD for shortly) is another important decomposition which is based on the eigenvalues.

**Definition 1.1.6.** *Let  $A \in \mathbb{C}^{m \times n}$ ,  $u \in \mathbb{C}^m$ ,  $v \in \mathbb{C}^n$ ,  $\sigma \geq 0$  be such that*

$$A v = \sigma u, \quad A^* u = \sigma v. \tag{1.1.1}$$

Then  $\sigma$  is called the singular value of  $A$ , and the vectors  $u$  and  $v$  are left and right singular vectors of  $A$  respectively.

According to (1.1.1),

$$A^*Av = \sigma^2v, \quad AA^*u = \sigma^2u$$

which implies that  $\sigma^2$  is the eigenvalue of  $A^*A$  and  $AA^*$ .

**Proposition 1.1.7.** Let  $A \in \mathbb{C}^{m \times n}$  and  $\{\lambda_1, \lambda_2, \dots, \lambda_p\}$  be the nonzero eigenvalues of  $AA^*$  (i.e.  $A^*A$ ). There are exactly  $p$  non-zero singular values of  $A$ , denoted by  $\sigma_i(A)$ ,  $i = 1, \dots, p$ , and they are equal to:

$$\sigma_i(A) = \sqrt{\lambda_i}, \quad i = 1, \dots, p.$$

Moreover, it can be shown that the singular values are positive stationary values<sup>1</sup> of the (vector) function  $f(x) = \|Ax\|/\|x\|$ .

**Theorem 1.1.1.** (Singular Value Decomposition) Let  $A \in \mathbb{C}_r^{m \times n}$ . Then there are unitary matrices  $U \in \mathbb{C}^{m \times m}$  and  $V \in \mathbb{C}^{n \times n}$  such that

$$A = U \begin{bmatrix} \Sigma & O \\ O & O \end{bmatrix} V^*, \quad (1.1.2)$$

where

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & \sigma_r \end{bmatrix} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r), \quad \sigma_i = \sqrt{\lambda_i}$$

and  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r > 0$  are the nonzero eigenvalues of  $A^*A$ .

*Proof.* Since  $A^*A \in \mathbb{C}_r^{n \times n}$  is Hermitian positive semidefinite, its eigenvalues are nonnegative.

Let the eigenvalues of  $A^*A$  be  $\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2$ , where

$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0 = \sigma_{r+1} = \dots = \sigma_n$ . Let  $v_1, v_2, \dots, v_n$  be a set of orthonormal eigenvectors for  $\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2$ , and let

$$V_1 = [v_1, v_2, \dots, v_r], \quad V_2 = [v_{r+1}, v_{r+2}, \dots, v_n],$$

and

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r).$$

Then

$$V_1^*(A^*A)V_1 = \Sigma^2, \quad V_2^*(A^*A)V_2 = O$$

---

<sup>1</sup>Stationary point of the function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is every point  $x_0 \in \mathbb{R}^n$  which satisfies  $\nabla f(x_0) = 0$ , while  $f(x_0)$  is the corresponding stationary value.

and consequently

$$\Sigma^{-1}V_1^*A^*AV_1\Sigma^{-1} = I, \quad AV_2 = O.$$

Now let

$$U_1 = AV_1\Sigma^{-1}.$$

Then  $U_1^*U_1 = I$ , that is the columns of  $U_1$  are orthonormal. Let  $U_2$  be chosen so that  $U = [U_1 \ U_2]$  is unitary. Then

$$\begin{aligned} U^*AV &= \begin{bmatrix} U_1^*AV_1 & U_1^*AV_2 \\ U_2^*AV_1 & U_2^*AV_2 \end{bmatrix} \\ &= \begin{bmatrix} (AV_1\Sigma^{-1})^*AV_1 & O \\ U_2^*(U_1\Sigma) & O \end{bmatrix} \\ &= \begin{bmatrix} \Sigma & O \\ O & O \end{bmatrix}. \end{aligned}$$

Thus (1.1.2) holds.  $\square$

**Remark 1.1.1.** *The quantities  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$  are the nonzero singular value of  $A$  and (1.1.2) is called the singular value decomposition of  $A$ .*

**Lemma 1.1.1.** *Let  $A$  and  $B = A + E \in \mathbb{C}^{m \times n}$  have singular values  $\sigma_1(A) \geq \sigma_2(A) \geq \dots \geq \sigma_n(A)$  and  $\sigma_1(B) \geq \sigma_2(B) \geq \dots \geq \sigma_n(B)$  respectively. Then*

$$\sigma_i(A) - \|E\|_2 \leq \sigma_i(B) \leq \sigma_i(A) + \|E\|_2, \quad i = 1, 2, \dots, n. \quad (1.1.3)$$

### 1.1.3 Idempotent matrices and projectors

Idempotent matrices and projectors are very important notions and appear in numerous problems concerning various generalized inverses.

**Lemma 1.1.2.** *Let  $E \in \mathbb{C}^{n \times n}$  be idempotent. Then  $E$  possesses the following properties:*

- (a)  $E^*$  and  $I - E$  are idempotent.
- (b) The eigenvalues of  $E$  are 0 and 1. The multiplicity of the eigenvalue 1 is equal to  $\text{rank}(E)$ .
- (c)  $\text{rank}(E) = \text{Tr}(E)$ .
- (d)  $E(I - E) = (I - E)E = O$ .
- (e)  $Ex = x$  if and only if  $x \in \mathcal{R}(E)$ .
- (f)  $E \in E\{1, 2\}$ .
- (g)  $\mathcal{N}(E) = \mathcal{R}(I - E)$ .

The transformation denoted by  $P_{L,M}$  carries any  $x \in \mathbb{C}^n$  into its projection on  $L$  along

$M$ . The transformation  $P_{L,M}$  is called the projector on  $L$  along  $M$ , or, oblique projector. It is known that the projector is a linear transformation.

**Proposition 1.1.8.** *For every idempotent matrix  $E \in \mathbb{C}^{n \times n}$ , the subspaces  $\mathcal{R}(E)$  and  $\mathcal{N}(E)$  are complementary and satisfy*

$$E = P_{\mathcal{R}(E), \mathcal{N}(E)}.$$

## 1.2 Main classes of generalized inverses

### 1.2.1 The inverse of a nonsingular matrix

A matrix has an inverse only if it is square, and even then only if it is nonsingular, or, in other words, if its columns (or rows) are linearly independent. It is well known that every nonsingular matrix  $A$  has a unique inverse, denoted by  $A^{-1}$ , such that

$$AA^{-1} = A^{-1}A = I.$$

We mention a few of the numerous properties of the inverse matrix:

$$\begin{aligned} (A^{-1})^{-1} &= A, \\ (A^T)^{-1} &= (A^{-1})^T, \\ (A^*)^{-1} &= (A^{-1})^*, \\ (AB)^{-1} &= B^{-1}A^{-1}, \end{aligned}$$

It will be recalled that a real or complex number  $\lambda$  is called an eigenvalue of a square matrix  $A$ , and a nonzero vector  $x$  is called an eigenvector of  $A$  corresponding to  $\lambda$ , if

$$Ax = \lambda x.$$

Another property of the inverse  $A^{-1}$  is that its eigenvalues are the reciprocals of those of  $A$ .

### 1.2.2 Solvability of linear systems

One of the most familiar application of matrices is to the solution of systems of simultaneous linear equations. Let

$$Ax = b \tag{1.2.1}$$

be such a system, where  $b$  is a given vector and  $x$  is an unknown vector. If  $A$  is nonsingular, there is a unique solution for  $x$  given by

$$x = A^{-1}b.$$

In the general case (when  $A$  may be singular or rectangular) there may sometimes be no solutions or multiple solutions.

The consistent system of linear equations

$$Ax = b \quad (A \in \mathbb{C}^{m \times n}, m < n, b \in \mathcal{R}(A))$$

has many solutions. The inconsistent system of linear equations

$$Ax = b \quad (A \in \mathbb{C}^{m \times n}, b \notin \mathcal{R}(A))$$

has no solution.

When (1.2.1) has many solutions, we may desire not just one solution but characterization of all solutions. It has been shown [4, 62] if  $X$  is any matrix satisfying  $AXA = A$ , then  $Ax = b$  has solution if and only if

$$AXB = b,$$

in which case the most general solution is

$$x = Xb + (I - XA)y,$$

where  $y$  is arbitrary.

### 1.2.3 Definitions and main properties of generalized inverses

The problem of the generalized inverses computation is closely related with the following four equations, called the Penrose equations:

$$(1) \quad AXA = A \quad (2) \quad XAX = X \quad (3) \quad (AX)^* = AX \quad (4) \quad (XA)^* = XA.$$

For a subset  $\mathcal{S} \subseteq \{1, 2, 3, 4\}$ , the set of all matrices obeying the conditions contained in  $\mathcal{S}$  is denoted by  $A\{\mathcal{S}\}$ . Any matrix from  $A\{\mathcal{S}\}$  is called an  $\mathcal{S}$ -inverse of  $A$  and it is denoted by  $A^{(\mathcal{S})}$ . By  $A\{\mathcal{S}\}_s$  we denote the set of all  $\mathcal{S}$ -inverses of  $A$  with rank  $s$ .

In 1955 Penrose showed for every finite matrix  $A$  (square or rectangular) of real or complex elements, there is a unique matrix  $X$  satisfying the all four equations.

**Theorem 1.2.1.** [62] *For any matrix  $A \in \mathbb{C}^{m \times n}$  there exists a single element in the set*

$A\{1, 2, 3, 4\}$ , called the Moore-Penrose inverse of  $A$  and denoted by  $A^\dagger$ .

*Proof.* Let  $X, Y \in A\{1, 2, 3, 4\}$ . Then

$$\begin{aligned} X &= X(AX)^* = XX^*A^* = X(AX)^*(AY)^* \\ &= XAY = (XA)^*(YA)^*Y = A^*Y^*Y \\ &= (YA)^*Y = Y, \end{aligned}$$

which completes the proof.  $\square$

If  $A$  is nonsingular, it is clear that  $X = A^{-1}$  satisfies the four Penrose equations. Since the Moore-Penrose inverse is unique, it follows that the Moore-Penrose inverse of a nonsingular matrix is the same as the ordinary inverse.

In this way, we come to the notion of  $\{i, j, \dots, k\}$ -inverses, where  $i, j, k \in \mathcal{S}$ . For example, for a given matrix  $A \in \mathbb{C}^{m \times n}$ , if there exists a matrix such that it satisfies only the first Penrose equation, then this matrix is called an  $\{1\}$ -inverse of the matrix  $A$  and it is denoted by  $A^{(1)}$ . Similarly, if the generalized inverse satisfies the first and the third Penrose equations, it is an  $\{1, 3\}$ -inverse of  $A$ , denoted by  $A^{(1,3)}$  while the corresponding set is denoted by  $A\{1, 3\}$ .

For a given subspaces  $T$  and  $S$  from  $\mathbb{C}^n$  by  $P_{T,S}$  we denote a projector from  $\mathbb{C}^n$  on  $T$  along  $S$ . If  $S = T^\perp$ , i.e., if  $S$  is orthogonal complement of  $T$ , then  $P_T$  is orthogonal projector from  $\mathbb{C}^n$  on  $T$ . The matrix which corresponds to a linear map which is a projector, is idempotent matrix. The matrix which corresponds to a linear map which is an orthogonal projector is a Hermitian idempotent matrix.

### Properties and representations of $\{1\}$ -inverses

**Lemma 1.2.1.** Let  $A \in \mathbb{C}_r^{m \times n}$ , and let  $E \in \mathbb{C}_m^{m \times m}$  and  $P \in \mathbb{C}_n^{n \times n}$  be matrices which transform  $A$  into the form

$$EAP = \begin{bmatrix} I_r & K \\ O & O \end{bmatrix}.$$

Then the  $n \times m$  matrix

$$X = P \begin{bmatrix} I_r & K \\ O & L \end{bmatrix} E \tag{1.2.2}$$

is an  $\{1\}$ -inverse (or inner inverse) of  $A$ , for arbitrary block  $L \in \mathbb{C}_m^{(n-r) \times (m-r)}$ .

**Lemma 1.2.2.** For a given matrix  $A \in \mathbb{C}_r^{m \times n}$  the following statement are valid:

$$(a) (\lambda A)^{(1)} = \lambda^\dagger A^{(1)}, \text{ where } \lambda \in \mathbb{C} \text{ and } \lambda^\dagger = \begin{cases} \frac{1}{\lambda}, & \lambda \neq 0 \\ 0, & \lambda = 0 \end{cases};$$

- (b)  $AA^{(1)}$  is a projection from  $\mathbb{C}^m$  on  $\mathcal{R}(A)$ , i.e.,  $AA^{(1)} = P_{\mathcal{R}(A),S}$  where  $S \in \mathbb{C}^m$  is such that  $\mathcal{R}(A) + S = \mathbb{C}^m$ ;
- (c)  $I - A^{(1)}A$  is a projection from  $\mathbb{C}^n$  on  $\mathcal{N}(A)$ , i.e.,  $I - A^{(1)}A = P_{\mathcal{N}(A),T}$  where  $T \in \mathbb{C}^n$  is such that  $T + \mathcal{N}(A) = \mathbb{C}^n$ ;
- (d)  $\text{rank}(A^{(1)}) \geq \text{rank}(A)$ ;
- (e)  $A^{(1)}A = I_n$  if and only if  $r = n$ ;
- (f)  $AA^{(1)} = I_m$  if and only if  $r = m$ ;
- (g) If  $X \in A\{1\}$ , then  $X \in A\{1, 2\}$  if and only if  $\text{rank}(A) = \text{rank}(X)$ ;
- (h)  $(A^*A)^{(1)}A^* \in A\{1, 2, 3\}$ ;
- (i)  $A^*(AA^*)^{(1)} \in A\{1, 2, 4\}$ ;
- (j)  $A^{(1,4)}AA^{(1,3)} = A^\dagger$ .

The next results establish the extremely important relationship between  $\{i, j, \dots, k\}$ -inverses and the solutions of a linear matrix equation [2, 89].

**Lemma 1.2.3.** *Let  $A \in \mathbb{C}^{m \times n}$ ,  $B \in \mathbb{C}^{p \times q}$ ,  $D \in \mathbb{C}^{m \times q}$ . Then the matrix equation*

$$AXB = D$$

*is consistent if and only if it holds*

$$AA^{(1)}DB^{(1)}B = D,$$

*for some  $A^{(1)}, B^{(1)}$ . In this case, the general solution is*

$$X = A^{(1)}DB^{(1)} + Y - A^{(1)}AYBB^{(1)}$$

*for arbitrary  $Y \in \mathbb{C}^{n \times p}$ .*

**Corollary 1.2.1.** *Let  $A \in \mathbb{C}^{m \times n}$  and  $A^{(1)} \in A\{1\}$ . Then*

$$A\{1\} = \{A^{(1)} + Z - A^{(1)}AZAA^{(1)} \mid Z \in \mathbb{C}^{n \times m}\}.$$

**Corollary 1.2.2.** *Let  $A \in \mathbb{C}^{m \times n}$  and*

$$Ax = b, \tag{1.2.3}$$



where  $A \in \mathbb{C}^{m \times n}$  and  $b \in \mathbb{C}^m$ . Then the system (1.2.3) is consistent if and only if for some  $A^{(1)}$  it holds

$$AA^{(1)}b = b,$$

in which case the general solution of the system (1.2.3) is

$$x = A^{(1)}b + (I - A^{(1)}A)y,$$

for arbitrary  $y \in \mathbb{C}^n$ .

**Lemma 1.2.4.** *The matrix equations*

$$AX = B, \quad XD = E$$

have a common solution if and only if each equation separately has a solution, i.e.,

$$AA^{(1)}B = B, \quad ED^{(1)}D = E, \tag{1.2.4}$$

and

$$AE = BD.$$

In this case,

$$X = A^{(1)}B + ED^{(1)} - A^{(1)}AED^{(1)}$$

is a common solution of both equations, for arbitrary  $A^{(1)}$  and  $D^{(1)}$ .

**Lemma 1.2.5.** *Let the equations given in (1.2.4) have a common solution  $X_0 \in \mathbb{C}^{m \times n}$ . Then the general solution of these equations is given by*

$$X = X_0 + (I - A^{(1)}A)Y(I - DD^{(1)}),$$

for arbitrary  $A^{(1)} \in A\{1\}$ ,  $D^{(1)} \in D\{1\}$ ,  $Y \in \mathbb{C}^{m \times n}$ .

**Proposition 1.2.1.** *Let  $A \in \mathbb{C}^{m \times n}$ ,  $X \in \mathbb{C}^{n \times m}$ . Then  $X \in A\{1\}$  if and only if, for all  $b \in \mathcal{R}(A)$ ,  $x = Xb$  is a solution of the system (1.2.3).*

**Proposition 1.2.2.** *The identity  $AB(AB)^{(1)}A = A$  holds if and only if  $\text{rank}(AB) = \text{rank}(A)$ . Similarly,  $B(AB)^{(1)}AB = B$  is valid if and only if  $\text{rank}(AB) = \text{rank}(B)$ .*

**Proposition 1.2.3.** *Let  $A \in \mathbb{C}^{m \times n}$  and let  $A^{(1)}$  be an arbitrary element of  $A\{1\}$ . Further, denote by  $\mathcal{R}(A) = L$  and  $\mathcal{N}(A) = M$ . Then  $AA^{(1)}$  and  $A^{(1)}A$  are idempotent and*

$$AA^{(1)} = P_{L,S}, \quad A^{(1)}A = P_{T,M},$$

where  $S$  is a subspace of  $\mathbb{C}^m$  complementary to  $L$ , and  $T$  is a subspace of  $\mathbb{C}^n$  complementary to  $M$ .

### Properties and representations of $\{1, 2\}$ -inverses

It is known that the existence of a  $\{1\}$ -inverse of a matrix  $A$  implies the existence of its  $\{1, 2\}$ -inverse. This fact is verified in Lemma 1.2.6.

**Lemma 1.2.6.** *Let  $Y, Z \in A\{1\}$ . Then  $X = YAZ \in A\{1, 2\}$ .*

According to Lemma 1.2.6, for any  $L \in \mathbb{C}^{(n-r) \times (m-r)}$ , the  $n \times m$  matrix  $X$  defined in (1.2.2) belongs to  $A\{1, 2\}$  if and only if  $X$  is given in the form (1.2.2).

**Lemma 1.2.7.** (Bjerrhammar 1958) [4] *For a given  $A$  and  $X \in A\{1\}$ , it follows that  $X \in A\{1, 2\}$  if and only if  $\text{rank}(X) = \text{rank}(A)$ .*

**Lemma 1.2.8.** *Any two of the following three statements imply the third:*

$$\begin{aligned} X &\in A\{1\}, \\ X &\in A\{2\}, \\ \text{rank}(X) &= \text{rank}(A). \end{aligned}$$

**Proposition 1.2.4.** *If  $A$  and  $X$  are  $\{1, 2\}$ -inverses of each other, then*

$$AX = P_{\mathcal{R}(A), \mathcal{N}(X)}, \quad XA = P_{\mathcal{R}(X), \mathcal{N}(A)}.$$

### Basic properties of the Moore-Penrose inverse

The most important result related to the Penrose equations is the statement that there always exists a unique matrix which satisfies the four Penrose equations. This result was shown by Penrose [61] in 1955. This matrix is called the Moore-Penrose inverse and denoted by  $A^\dagger$ .

The concept of a generalized inverses of an arbitrary matrix  $A \in \mathbb{C}^{m \times n}$  is originally due to Moore, in 1920, (called by him the "general reciprocal"). His definition was essentially as follows.

**Definition 1.2.1.** *If  $A \in \mathbb{C}^{m \times n}$ , then the generalized inverse of  $A$  is the matrix  $X \in \mathbb{C}^{n \times m}$  such that*

$$1. \quad AX = P_{\mathcal{R}(A)}; \quad 2. \quad XA = P_{\mathcal{R}(X)}. \quad (1.2.5)$$

Moore proved the existence and the uniqueness of the solution of such defined generalized inverse by proving the following result.

**Proposition 1.2.5.** *For every  $A \in \mathbb{C}^{m \times n}$  there exists a unique matrix  $X \in \mathbb{C}^{n \times m}$  satisfying (1.2.5).*

Rado proved the equivalence of Moore's and Penrose's definitions of the generalized inverse, and today this inverse is known as Moore-Penrose pseudoinverse (shortly M-P inverse or pseudoinverse).

Although  $\{1\}$ -inverses and  $\{1, 3\}$ -inverses provide a solution of a given matrix equation, the Moore-Penrose inverse most resemble to the ordinary inverse. This statement is justified by its uniqueness and the properties listed in the following two lemmas. Also, since the Moore-Penrose inverse is  $\{1\}$ -inverse, we should take into account that the properties from Lemma 1.2.2 are also valid for the Moore-Penrose inverse.

**Proposition 1.2.6.** (Penrose 1955) [61] *Let  $A \in \mathbb{C}^{m \times n}$  and  $b \in \mathbb{C}^{m \times 1}$ . The minimal-norm least-squares solution of the system  $Ax = b$  is given by  $x^* = A^\dagger b$ . All other least-squares solutions are given by*

$$x = A^\dagger b + (I_n - A^\dagger A)z, \quad z \in \mathbb{C}^n.$$

**Lemma 1.2.9.** *Let  $A \in \mathbb{C}^{m \times n}$  be an arbitrary matrix. The Moore-Penrose inverse  $A^\dagger$  possesses the following properties:*

- (a)  $(A^\dagger)^\dagger = A$ ,  $(A^\dagger)^* = (A^*)^\dagger$ ;
- (b)  $(AA^*)^\dagger = (A^*)^\dagger A^\dagger$ ,  $(A^*A)^\dagger = A^\dagger (A^*)^\dagger$ ;
- (c)  $A^\dagger AA^* = A^* = A^* AA^\dagger$ ;
- (d)  $A^\dagger = (A^*A)^\dagger A^* = A^*(AA^*)^\dagger$ ;
- (e)  $\mathcal{N}(AA^\dagger) = \mathcal{N}(A^\dagger) = \mathcal{N}(A^*) = \mathcal{R}(A)$
- (f)  $\mathcal{R}(AA^*) = \mathcal{R}(AA^{(1)}) = \mathcal{R}(A)$ ,  $\text{rank}(AA^{(1)}) = \text{rank}(A^{(1)}A) = \text{rank}(A)$ ;
- (g)  $AA^\dagger = P_{\mathcal{R}(A^*), \mathcal{N}(A)}$  and  $A^\dagger A = P_{\mathcal{R}(A), \mathcal{N}(A^*)}$ .

**Lemma 1.2.10.** *Let  $A \in \mathbb{C}^{m \times n}$  be an arbitrary matrix. Then the matrix  $A$  can be written in the form*

$$A \sim \begin{bmatrix} A_1 & O \\ O & O \end{bmatrix} : \begin{bmatrix} \mathcal{R}(A^*) \\ \mathcal{N}(A) \end{bmatrix} \rightarrow \begin{bmatrix} \mathcal{R}(A) \\ \mathcal{N}(A^*) \end{bmatrix}, \quad (1.2.6)$$

where  $A_1$  is invertible. Hence,

$$A^\dagger \sim \begin{bmatrix} A_1^{-1} & O \\ O & O \end{bmatrix} : \begin{bmatrix} \mathcal{R}(A) \\ \mathcal{N}(A^*) \end{bmatrix} \rightarrow \begin{bmatrix} \mathcal{R}(A^*) \\ \mathcal{N}(A) \end{bmatrix}.$$

The representation (1.2.6) of can be easily obtained from the Singular value decomposition (SVD) of  $A$ . More precisely, the SVD decomposition of  $A$  assumes that the matrix  $A_1$  is a diagonal matrix whose entries are the singular values of  $A$ .

If the vector  $b$  in the system (1.2.3) satisfies  $b \notin \mathcal{R}(A)$ , then it is necessary to search for an approximate solution by trying to find a vector  $x$  which minimizes the norm of the vector  $Ax - b$ .

**Definition 1.2.2.** Let  $A \in \mathbb{C}^{m \times n}$  and  $b \in \mathbb{C}^m$ . A vector  $\hat{x} \in \mathbb{C}^n$  which satisfies the minimization problem

$$\|A\hat{x} - b\|^2 = \min_{x \in \mathbb{C}^n} \|Ax - b\|^2. \quad (1.2.7)$$

is called a least-squares solution of the system (1.2.3).

The next lemma gives a characterization of all least-squares solutions of the system (1.2.3).

**Lemma 1.2.11.** The vector  $x$  is a least-squares solution of the system (1.2.3) if and only if  $x$  is a solution of the normal equation, defined by

$$A^*Ax = A^*b. \quad (1.2.8)$$

The following proposition, restated from [2], shows that  $\|Ax - b\|$  is minimized by the vector  $x = A^{(1,3)}b$ . This statement establishes very important relation between the set of  $\{1, 3\}$ -inverses and the least-squares solutions of the system (1.2.3).

**Proposition 1.2.7.** Let  $A \in \mathbb{C}^{m \times n}$ ,  $b \in \mathbb{C}^m$ . Then  $\|Ax - b\|$  is smallest when  $x = A^{(1,3)}b$ , where  $A^{(1,3)} \in A\{1, 3\}$ . Conversely, if  $X \in \mathbb{C}^{n \times m}$  has the property that, for all  $b$ ,  $\|Ax - b\|$  is smallest when  $x = Xb$ , then  $X \in A\{1, 3\}$ .

Since  $A^{(1,3)}$  inverse of a matrix is not unique, as a consequence, a system of linear equations has many least-squares solutions in general. However, among all least-squares solutions of a given system of linear equations, there exists only one such solution of minimum norm.

**Definition 1.2.3.** Let  $A \in \mathbb{C}^{m \times n}$  and  $b \in \mathbb{C}^m$ . A vector  $\hat{x}$ , which satisfies the minimization problem

$$\|\hat{x}\|^2 = \min_{x \in \mathbb{C}^n} \|x\|^2 \quad (1.2.9)$$

is called a minimal-norm solution of the system (1.2.3).

The next proposition, restated from [2], establishes a relation between  $\{1, 4\}$ -inverses and the minimum-norm solutions of the system (1.2.3).

**Proposition 1.2.8.** Let  $A \in \mathbb{C}^{m \times n}$ ,  $b \in \mathbb{C}^m$ . If  $Ax = b$  has a solution for  $x$ , the unique solution  $x$  for which  $\|x\|$  is smallest is given by  $x = A^{(1,4)}b$ , where  $A^{(1,4)} \in A\{1, 4\}$ . Conversely, if

$X \in \mathbb{C}^{n \times m}$  is such that, whenever  $Ax = b$  has a solution,  $x = Xb$  is the solution of minimal-norm, then  $X \in A\{1, 4\}$ .

Joining the results from Proposition 1.2.7 and Proposition 1.2.8 we are coming to the most important property of the Moore-Penrose inverse.

**Corollary 1.2.3.** (Penrose 1955) [61] *Let  $A \in \mathbb{C}^{m \times n}$ ,  $b \in \mathbb{C}^m$ . Then, among the least-squares solutions of  $Ax = b$ ,  $A^\dagger b$  is the one of minimum-norm. Conversely, if  $X \in \mathbb{C}^{n \times m}$  has the property that, for all  $b$ ,  $Xb$  is the minimal-norm least-squares solution of  $Ax = b$ , then  $X = A^\dagger$ .*

In the essence, Lemma 1.2.3 shows that  $A^\dagger b$  is the minimal-norm least-squares solution of the linear system  $Ax = b$ . This fact caused a dramatic increase of the interest in the generalized inverses theory.

Further, the next proposition characterizes the set of all least-squares solutions of a given system of linear equations.

**Proposition 1.2.9.** (Nashed 1970, 1976) [59] *If  $A \in \mathbb{C}^{m \times n}$  has a closed range  $\mathcal{R}(A)$ , then the set  $S$  of all least-squares solutions of the system  $Ax = b$  is given by*

$$S = A^\dagger b \oplus \mathcal{N}(A) = \{A^\dagger b + (I - A^\dagger A)y \mid y \in \mathbb{C}^n\},$$

where  $\mathcal{N}(A)$  denotes the null space of  $A$ .

Some additional properties of  $A^\dagger$  and  $A^{(1)}$  can be found for example in [2, 89].

The Moore-Penrose inverse can be computed using arbitrary  $\{1\}$ -inverse.

**Proposition 1.2.10.** (Yanai, Takeuchi, Takane 2011) [119] *The Moore-Penrose inverse  $A^\dagger$  can be expressed by an arbitrary inner inverse, as*

$$\begin{aligned} A^\dagger &= A^T A (A^T A A^T A)^{(1)} A^T \\ &= A^T (A A^T)^{(1)} A (A^T A)^{(1)} A^T. \end{aligned}$$

The singular value decomposition is very important notion in developing RNN models and even in numerical linear algebra.

**Lemma 1.2.12.** *Let  $A \in \mathbb{C}_r^{m \times n}$  and let  $A = U \Sigma V^*$  be the singular value decomposition of  $A$  where  $U \in \mathbb{C}^{m \times m}$  and  $V \in \mathbb{C}^{n \times n}$  are unitary matrices and  $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$ ,  $\sigma_i = \sqrt{\lambda_i}$  and  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r > 0$  are the nonzero eigenvalues of  $A^* A$ . If*

$$A = U \begin{bmatrix} \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r) & O \\ O & O \end{bmatrix} V^* = U \begin{bmatrix} \Sigma & O \\ O & O \end{bmatrix} V^* \in \mathbb{C}^{m \times n},$$

then

$$A^\dagger = V \begin{bmatrix} \text{diag}(1/\sigma_1, 1/\sigma_2, \dots, 1/\sigma_r) & O \\ O & O \end{bmatrix} U^* = V \begin{bmatrix} \Sigma^{-1} & O \\ O & O \end{bmatrix} U^* \in \mathbb{C}^{n \times m}.$$

Moreover, let  $S_1, S_2, S_3$  be arbitrary  $r \times m - r$ ,  $n - r \times r$ , and  $m - r \times n - r$  matrices, respectively. Then an inner inverse of  $A$  is given by

$$A^{(1)} = V \begin{bmatrix} \text{diag}(1/\sigma_1, 1/\sigma_2, \dots, 1/\sigma_r) & S_1 \\ S_2 & S_3 \end{bmatrix} U^* \in \mathbb{C}^{n \times m}.$$

**Theorem 1.2.2.** Let  $A \in \mathbb{C}_r^{m \times n}$ , let

$$A = U \begin{bmatrix} \Sigma & O \\ O & O \end{bmatrix} V^* \quad (1.2.10)$$

be the singular value decomposition (SVD decomposition) of  $A$ , where  $U \in \mathbb{C}^{m \times m}$  and  $V \in \mathbb{C}^{n \times n}$  are unitary matrices and  $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$ ,  $\sigma_i = \sqrt{\lambda_i}$  and  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r > 0$  are the nonzero eigenvalues of  $A^*A$ . Then  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$  are the nonzero singular value of  $A$  and

$$\|A\|_2 = \sigma_1, \quad \|A^\dagger\|_2 = \frac{1}{\sigma_r}. \quad (1.2.11)$$

*Proof.* From (1.2.10), we have

$$A^*A = V \begin{bmatrix} \Sigma^2 & O \\ O & O \end{bmatrix} V^*.$$

Thus the eigenvalues of  $A^*A$  are  $\sigma_i^2 = \lambda_i(A^*A)$ ,  $i = 1, 2, \dots, n$  and

$$\|A\|_2^2 = \|A^*A\|_2 = |\lambda_1(A^*A)| = \sigma_1^2.$$

So  $\|A\|_2 = \sigma_1$  holds. It is easy to verify that

$$A^\dagger = V \begin{bmatrix} \Sigma^{-1} & O \\ O & O \end{bmatrix} U^*. \quad (1.2.12)$$

Hence the non-zero singular values of  $A^\dagger$  are

$$\frac{1}{\sigma_r} \geq \frac{1}{\sigma_{r-1}} \geq \dots \geq \frac{1}{\sigma_1} > 0.$$

Thus  $\|A^\dagger\|_2 = \frac{1}{\sigma_r}$  holds.  $\square$

Next lemma shows that the full rank factorization of a matrix  $A$  leads to an explicit formula for its Moore-Penrose inverse  $A^\dagger$ . This formula is known as the full rank representation of the Moore-Penrose inverse. As usual, by  $A_R^{-1}$  and  $A_L^{-1}$  we denote a right and a left inverse of  $A$ , respectively.

**Lemma 1.2.13.** (MacDuffe, 1956) [55] *Let  $A \in \mathbb{C}_r^{m \times n}$  and  $A = PQ$ ,  $P \in \mathbb{C}_r^{m \times r}$ ,  $Q \in \mathbb{C}_r^{r \times n}$  be its full rank factorization. Then it holds*

$$A^\dagger = Q^*(P^*AQ^*)^{-1}P^* = Q^*(QQ^*)^{-1}(P^*P)^{-1}P^*.$$

*In addition,  $Q$  is right invertible and  $P$  is left invertible:*

$$Q_R^{-1} = Q^*(QQ^*)^{-1}, \quad P_L^{-1} = (P^*P)^{-1}P^*.$$

**Definition 1.2.1.** *For a given matrix  $A \in \mathbb{C}^{m \times n}$ , its weighted Moore–Penrose inverse  $A_{M,N}^\dagger \in \mathbb{C}^{n \times m}$  satisfies the following matrix equations:*

$$\begin{aligned} A_{M,N}^\dagger A &= A, \quad A_{M,N}^\dagger A A_{M,N}^\dagger = A_{M,N}^\dagger, \\ (A A_{M,N}^\dagger)^\# &= A A_{M,N}^\dagger, \quad (A_{M,N}^\dagger A)^\# = A_{M,N}^\dagger A \end{aligned}$$

*where  $M$  and  $N$  are the symmetric positive definite matrices of order  $m$  and  $n$ , respectively, and  $A^\# = N^{-1}A^T M$ . In particular, when  $M$  and  $N$  are the identity matrix  $I$ ,  $A_{M,N}^\dagger$  reduces to the Moore–Penrose inverse  $A^\dagger$ .*

A non-zero matrix that is of neither full column rank nor full row rank can be expressed as the product of a matrix of full column rank and a matrix of full row rank. We call this a full rank factorization.

**Theorem 1.2.3.** [89] *Let  $A \in \mathbb{C}_r^{m \times n}$ ,  $r > 0$ . Then there exist matrices  $P \in \mathbb{C}_r^{m \times r}$  and  $Q \in \mathbb{C}_r^{r \times n}$  such that*

$$A = PQ. \tag{1.2.13}$$

**Definition 1.2.2.** *Let  $A \in \mathbb{C}^{n \times n}$  be a given matrix. The smallest nonnegative integer  $k$  for which the condition*

$$\text{rank}(A^{k+1}) = \text{rank}(A^k) \tag{1.2.14}$$

*holds is called the index of  $A$ , and it is denoted by  $\text{ind}(A) = k$ .*

Further we will supplement the four Penrose equations by the following equations applicable only to square matrices:

$$(1^k) \quad A^{l+1}X = A^l, \quad l \geq \text{ind}(A), \quad (5) \quad AX = XA.$$

**Definition 1.2.3.** *The Drazin inverse of a square matrix  $A \in \mathbb{C}^{n \times n}$  is the unique matrix  $X \in \mathbb{C}^{n \times n}$  which fulfills the matrix equation (2) in conjunction with (1<sup>k</sup>) and (5) and it is denoted by  $X = A^D$  (for more details see [89]).*

**Definition 1.2.4.** *The Drazin inverse in the case  $\text{ind}(A) = 1$  is called the group inverse and is denoted by  $X = A^\#$ .*

**Definition 1.2.5.** *The outer inverse of  $A \in \mathbb{C}_r^{m \times n}$  with prescribed range  $T$  and null space  $S$  is the matrix  $X$  which satisfies the equation (2) and two additional properties:  $\mathcal{R}(X) = T$  and  $\mathcal{N}(X) = S$ , and is denoted by  $A_{T,S}^{(2)}$ ,*

It is well known that the Moore-Penrose inverse  $A^\dagger$  and the weighted Moore-Penrose inverse  $A_{M,N}^\dagger$ , the Drazin inverse  $A^D$  and the group inverse  $A^\#$  can be presented as particular generalized inverses  $A_{T,S}^{(2)}$  for appropriate choice of the matrices  $T = \mathcal{R}(G)$  and  $S = \mathcal{N}(G)$ . For example, the next statements are valid for a rectangular matrix  $A$  (see [2]):

$$A^\dagger = A_{\mathcal{R}(A^*), \mathcal{N}(A^*)}^{(2)}, \quad A_{M,N}^\dagger = A_{\mathcal{R}(A^\#), \mathcal{N}(A^\#)}^{(2)}, \quad (1.2.15)$$

where  $M, N$  are positive definite matrices of appropriate orders and  $A^\# = N^{-1}A^*M$ . For a given square matrix  $A$  the next identities (see [2, 89]) are satisfied:

$$A^D = A_{\mathcal{R}(A^k), \mathcal{N}(A^k)}^{(2)}, \quad A^\# = A_{\mathcal{R}(A), \mathcal{N}(A)}^{(2)}, \quad (1.2.16)$$

where  $k = \text{ind}(A)$ .

The outer generalized inverses with prescribed range and null-space are very important in matrix theory. The  $\{2\}$ -inverses have application in the iterative methods for solving the nonlinear equations [2] as well as in statistics [20, 29]. In particular, outer inverses play an important role in stable approximations of ill-posed problems and in linear and nonlinear problems involving rank-deficient generalized inverses [59, 134].

Outer inverses with prescribed range and null space are useful in solving the restricted system of linear equations. This application is based on the following essential result from [12]:

**Proposition 1.2.11.** [12] *Let  $A \in \mathbb{C}^{m \times n}$  be of rank  $r$ , let  $T$  be a subspace of  $\mathbb{C}^n$ , and let the condition*

$$b \in AT, \quad \dim(AT) = \dim(T)$$



be satisfied. Then the unique solution of

$$Ax = b, \quad x \in T$$

is given by

$$x = A_{T,S}^{(2)}b,$$

for any subspace  $S$  of  $\mathbb{C}^m$  satisfying  $AT \oplus S = \mathbb{C}^m$ .

For other important properties of generalized inverses see [2, 89].

### 1.3 Motivation and organization of the dissertation

The aim of this dissertation is the application of recurrent neural networks (RNN shortly) to solving some problems from a matrix algebra with particular reference to the computations of the generalized inverses as well as solving the matrix equations of constant (time-invariant) matrices.

The main efforts in the generalized inverse computation can be divided into two main types: numerical algorithms and continuous-time algorithms. The numerical algorithms can be divided into two categories: direct and iterative methods. The singular value decomposition (SVD) algorithm is the most known between the direct methods [2]. Also, other types of matrix factorizations has been exploited in computation of generalized inverses, such as the QR decomposition [36, 78], LU factorization [83]. Methods based on the application of the Gauss-Jordan elimination process to an appropriate augmented matrix were investigated [27, 79]. The SVD algorithm is more accurate and is thus the most commonly used method, but it requires a large amount of computational resources. The iterative methods, such as the orthogonal projection algorithms, the Newton iterative algorithm, and the higher-order convergent iterative methods [74, 47, 64, 19, 76] are more suitable for implementation. The Newton iterative method is developed for block matrices in [58]. This algorithm has a fast convergence rate, but it requires an initial condition for its convergence. All iterative methods, in general, require initial conditions which are ultimate, rigorous and sometimes cannot be fulfilled easily. A lot of iterative methods whose main objective is a numerical iterative computation of outer inverses with prescribed range and null space have been developed. An overview of these methods can be found in [19, 31, 51, 64, 76, 115, 121]. Iterative methods for computing generalized inverses assume certain conditions for their convergence and the convergence is theoretically established in the real variable case only. To overcome this disadvantage, an iterative algorithm was proposed in [118].

Lately, neural networks have shown a huge potential as parallel distributed computational models for solving many computationally challenging problems, such as various types of con-

strained optimization problems [30, 49], or the WTA (Winner-Take-All) competition phenomena [43]. A quite a number of results related to the application of neural networks in solving a variety of matrix algebra problems have been published recently. Different types of neural networks have been introduced to solve systems of linear algebraic equations. The authors of the papers [16, 97] designed recurrent neural networks for solving simultaneous linear algebraic equations. Wang in [92, 93, 95] proposed a gradient neural network to solve simultaneous linear equations. In [95], it was verified that proposed recurrent neural networks are asymptotically stable in the large and capable of computing inverse matrices and solving Lyapunov matrix equations. Two three-dimensional structured networks for solving linear equations and the Lyapunov equation were developed in [99]. Neuron-like network architectures for computing eigenvalues and eigenvectors of real matrices were investigated in [15, 70]. Two recurrent neural networks for computing LU decomposition and Cholesky factorization were presented in [98]. A variety of other matrix algebra problems have been solved by using neural networks, see for example [5, 13, 100].

In many real-time systems, real-time solutions of generalized inverses are usually imperative. An example of such applications in robotics is the solution to the manipulator inverse kinematics problem in real-time motion control of kinematically redundant robots. Reported results show that neural network architectures are more suitable for real-time applications than the conventional numerical algorithms. Thereinto, the neural approach is now regarded as a powerful alternative for scientific computing because of its parallel distributed nature and convenience of hardware implementation. A number of nonlinear and linear recurrent neural network models have been developed for the inversion and generalized inversion of square and full-rank rectangular matrices (for more details, see e.g. [34, 54, 94, 95]). Various recurrent neural networks for computing generalized inverses of rank-deficient matrices were designed in [96, 108]. Three recurrent neural networks for computing the weighted Moore–Penrose inverse of rank-deficient matrices are presented in [108]. A neural network approach to compute the Drazin inverse  $A^D$  was developed in [14]. The approach from [14] is based on a feed-forward multi-layer neural network, the gradient optimization technique and the standard back propagation learning algorithm. A new type of complex-valued Zhang neural network (ZNN), based on a complex Zhang function, is proposed and investigated in [48]. ZNN models for online time-varying full-rank matrix Moore–Penrose inversion are generalized, investigated and analyzed in [127].

We will briefly present some of the latest results that were published at the time of writing this dissertation. In [75] the authors proposed conditions for the existence and representations of  $\{2\}$ ,  $\{1, 2\}$ , and  $\{1\}$ -inverses along with a new computational framework for these generalized inverses. Proposed representations are applicable to the complex constant matrices. Four nonlinear gradient-based recurrent neural networks for computation of the Drazin inverse for

real constant matrices, based on the limiting representations of the Drazin inverse, are investigated in [101]. The global convergence performance of defined neural networks is ensured by any monotonically increasing odd activation function. In [102] two gradient-based recurrent neural networks for computing the  $W$ -weighted Drazin inverse of a real constant matrix are presented. Complex neural network models for time-varying Drazin Inverse computation are considered in [104]. The proposed network models are based on limit representations of the Drazin inverse and error-monitoring functions, which exploit Tikhnov regularization method. In [81] two gradient-based recurrent neural networks for generating various inner inverses, including the Moore-Penrose and the Drazin inverse are investigated in detail. Extension of the ZNN algorithmic conceptual framework, which was used for the computation of the regular matrix inverse, pseudoinverse, and the Drazin inverse, to the class of time-varying complex outer inverses with prescribed range and null space can be found in [103]. Moreover, the same paper investigates a hybrid combination of ZNN and GNN models for computing outer inverses of real constant matrices. In [66] the authors give a general scheme of discretization for transforming continuous-time ZNN models for matrix inversion and pseudoinversion into corresponding discrete-time iterative methods. This iterative scheme arises from the 4th order Adams-Bashforth method and is systematically examined.

However, numerical calculation of outer generalized inverses for the real constant matrices using the gradient-based recurrent neural network approach is insufficiently studied. Wei in [109] derived integral representation for the generalized inverse  $A_{T,S}^{(2)}$  in an efficient way and defined corresponding dynamic equation. But, the established dynamic equation is defined intuitively, starting from defined integral representation and using an analogy with the previous considerations.

### 1.3.1 GNN vs ZNN dynamics

According to the number of results related to the computation of inverses, generalized inverses and other matrix algebra problems, we could split neural network models into two main types: more traditional gradient-based neural networks (or termed gradient neural networks, GNN) usually aimed at constant (i.e. time-invariant) matrices and recently proposed new types of neural networks called Zhang neural networks (i.e. ZNN) introduced to generalize to the solution of online time-varying problems. The ZNN differs from conventional gradient-based neural networks designed intrinsically for static problems solving in several important things.

- The design of GNN models are based on the elimination of the norm-based scalar-valued error function which could only be positive or at least lower-bounded. In contrast, the design of ZNN models are based on the elimination of every entry of the matrix-valued error function, which could be positive, negative, bounded or even unbounded. The

matrix-valued error function could make the resultant ZNN models monitor and force every entry of the error to zero. Thus, more information are used for network learning and better performance can be achieved for the ZNN model, as compared to the GNN model.

- The ZNN models exploit the time-derivative information of problem matrix during real-time solving process. This is the reason why ZNN models could globally exponentially converge to the theoretical solution of the time-varying problem. In contrast, GNN models have not exploited such important information, and thus may not be effective in solving such a time-varying problem. In essence, the ZNN method is based on a prediction thought, while the GNN method belongs to a conventional tracking approach.
- The GNN models are depicted in explicit dynamics, i.e., in the general form  $\dot{X}(t) = \dots$ , which are usually associated with classic Hopfield-type recurrent neural networks. In contrast, the ZNN models are depicted in implicit dynamics (e.g.,  $\dot{X}(t)A(t)A^T(t) = \dots$ ), which frequently arise in analog electronic circuits and systems due to Kirchhoff's rules [124]. In addition, the implicit dynamic equations could preserve physical parameters in the coefficient matrices. They could describe the usual and unusual parts of a dynamic system in the same form. Thus, implicit systems have higher abilities in representing dynamic systems, as compared to explicit systems. If needed, the implicit dynamic systems could be transformed to explicit dynamic systems.

## 1.4 Review of known GNN models

In order to complete our motivation, we survey necessary known results of other authors regarding regular and generalized inverses computation using both GNN and ZNN types of recurrent neural networks. These results served as the starting point for the research presented in this dissertation. In addition to the results, the related sources are listed.

When it comes to creating GNN models, the dynamics of neural network is based on the usage of the scalar-valued norm-based error function  $\varepsilon(t) = \varepsilon(V(t)) = \frac{1}{2}\|E(t)\|_F^2$  where  $E(t)$  is an appropriate error matrix,  $\|A\|_F := \sqrt{\text{Tr}(A^T A)}$  denotes the Frobenius norm of the matrix  $A$  and  $\text{Tr}(\cdot)$  denotes the trace of a matrix (see [21]). The general design formula is usually defined along the negative gradient  $-\partial\varepsilon(V(t))/\partial V$  of  $\varepsilon(V(t))$ , until the minimum is reached:

$$\frac{dV(t)}{dt} = -\gamma\mathcal{F}\left(\frac{\partial\varepsilon(V(t))}{\partial V}\right). \quad (1.4.1)$$

Here,  $V(t)$  is the matrix of activation state variables,  $t \in [0, \infty)$  is a time parameter and  $\gamma$  is a positive scaling constant whose values must be harmonized with the chosen time interval.

Further,  $\mathcal{F}(R)$  is an odd and monotonically increasing function array, element-wise applicable to elements of a real matrix  $R = (r_{ij}) \in \mathbb{R}^{m \times n}$ , i.e.,  $\mathcal{F}(R) = (f(r_{ij}))$ ,  $i = 1, \dots, m$ ,  $j = 1, \dots, n$ , wherein  $f(\cdot)$  is an odd and monotonically increasing function. One of the main motivation for investigating GNNs for generalized inverses computation is based on the ability to exploit the correlation between the dynamic state equations of recurrent neural networks for computing generalized inverses and integral representations of these generalized inverses.

**Remark 1.4.1.** *Theorem 5.5.1 needs an activation function  $\mathcal{H}_1(\cdot)$  which involves a monotonically increasing odd function  $\mathcal{F}(\cdot)$ . The following widely used real-valued linear and nonlinear functions satisfy this requirement.*

*Linear function*

$$f(x) = x. \quad (1.4.2)$$

*Bipolar-sigmoid function*

$$f(x) = \frac{1 + \exp(-q)}{1 - \exp(-q)} \cdot \frac{1 - \exp(-qx)}{1 + \exp(-qx)}, \quad q > 2. \quad (1.4.3)$$

*Power-sigmoid function*

$$f(x) = \begin{cases} x^p, & \text{if } |x| \geq 1 \\ \frac{1 + \exp(-q)}{1 - \exp(-q)} \cdot \frac{1 - \exp(-qx)}{1 + \exp(-qx)}, & \text{otherwise} \end{cases}, \quad q > 2, \quad p \geq 3. \quad (1.4.4)$$

*Smooth power-sigmoid function*

$$f(x) = \frac{1}{2}x^p + \frac{1 + \exp(-q)}{1 - \exp(-q)} \cdot \frac{1 - \exp(-qx)}{1 + \exp(-qx)}, \quad p \geq 3, \quad q > 2. \quad (1.4.5)$$

For illustration and comparison, the four types of activation functions  $f$  are illustrated in Figure 1.1. Note that other new activation functions can be generated readily based on these basic types.

### 1.4.1 GNN for regular inverse

Wang in [94] proposed the dynamic equation of the linear recurrent neural network for computing the inverse of a nonsingular matrix  $A$ . This dynamics is initiated by the error matrix  $E(t) = AV(t) - I$ . Since

$$\frac{\partial \varepsilon(V(t))}{\partial V} = \frac{1}{2} \frac{\partial \|AV(t) - I\|_F^2}{\partial V} = A^T (AV(t) - I),$$

and using the general design rule (1.4.1) with linear activation function  $\mathcal{F}$ , dynamics can be

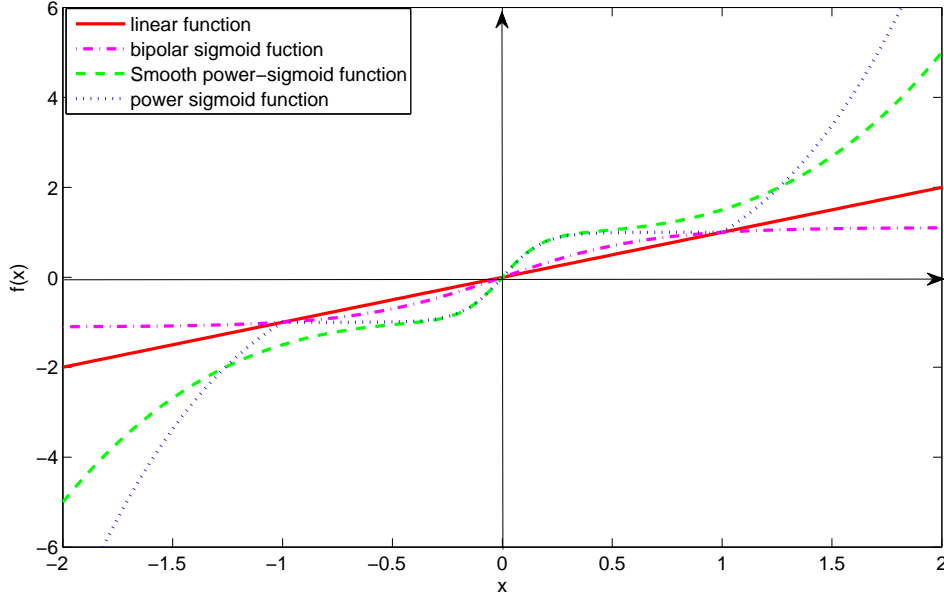


Figure 1.1: Behavior of the four basic types of activation functions

described as follows:

$$\frac{dV(t)}{dt} = -\gamma A^T A V(t) + \gamma A^T, \quad V(0) = V_0, \quad (1.4.6)$$

It is proven in [94] that the GNN model (1.4.6) is asymptotically stable in the large and the steady-state matrix of the recurrent neural network is equal to the inverse matrix of  $A$ , i.e.,  $\lim_{t \rightarrow \infty} V(t) = A^{-1}$ , for arbitrary  $V(0)$ .

## 1.4.2 GNN for computing the Moore-Penrose inverse

Recurrent neural network defined in (1.4.6) can be used for computing the Moore–Penrose inverse of a full-column rank rectangular matrix  $A \in \mathbb{R}_n^{m \times n}$ , by simply allowing the activation state matrix to be rectangular. In the full-row rank case,  $A \in \mathbb{R}_m^{m \times n}$ , the recurrent neural network

$$\frac{dV(t)}{dt} = -\gamma V(t) A A^T + \gamma A^T, \quad V(0) = V_0, \quad (1.4.7)$$

which is dual with respect to (1.4.6), can be also used to compute the Moore–Penrose inverse of  $A$ . The closed-form solution of the state matrix can be described as follows ([96]):

$$V(t) = \begin{cases} \exp(-\gamma A^T A t) V(0) + \gamma \exp(-\gamma A^T A t) \int_0^t \exp(\gamma A^T A \tau) A^T d\tau, & m \geq n, \\ V(0) \exp(-\gamma A A^T t) + \gamma A^T \exp(-\gamma A A^T t) \int_0^t \exp(\gamma A A^T \tau) d\tau, & m < n. \end{cases}$$

In the case of a full-rank rectangular  $A$ , Wang in [96] derived the following representation

of  $A^\dagger$ , which is independent of  $V(0)$ :

$$A^\dagger = \begin{cases} \lim_{t \rightarrow \infty} \gamma \exp(-\gamma A^T A t) \int_0^t \exp(\gamma A^T A \tau) A^T d\tau, & m \geq n, \\ \lim_{t \rightarrow \infty} \gamma A^T \exp(-\gamma A A^T t) \int_0^t \exp(\gamma A A^T \tau) d\tau, & m < n. \end{cases} \quad (1.4.8)$$

Moreover, Wang in [96] proposed three recurrent neural networks for computing the Moore–Penrose inverse of rank-deficient matrices. The first recurrent neural network has the dynamic equation

$$\frac{dV(t)}{dt} = \begin{cases} -MV(t)A^T A + MA^T, & V(0) = 0, \quad m \geq n, \\ -V(t)AA^T M + A^T M, & V(0) = 0, \quad m < n, \end{cases} \quad (1.4.9)$$

where  $M$  is a positive diagonal matrix satisfying  $M \in \mathbb{R}^{n \times n}$  if  $m \geq n$  and  $M \in \mathbb{R}^{m \times m}$  if  $m < n$ .

Representation of the Moore–Penrose inverse given in (1.4.8) corresponds to the following integral representation of the Moore–Penrose inverse for bounded linear operators, introduced in [26]:

$$A^\dagger = \int_0^\infty \exp(-A^T A \tau) A^T d\tau, \quad (1.4.10)$$

The global exponential convergence of Gradient neural network (1.4.7) in the case when  $A$  is nonsingular as well as its global stability when  $A$  is singular is verified in [126].

### 1.4.3 GNN for computing the weighted Moore-Penrose inverse

Wei in [108] introduced the following dynamic state equation of the first recurrent neural network (called  $NN1$ ) for computing the weighted Moore–Penrose inverse of a rank-deficient matrix:

$$\frac{dV(t)}{dt} = \begin{cases} -DA^\# AV(t) + DA^\#, & V(0) = 0, \quad m \geq n, \\ -V(t)AA^\# D + A^\# D, & V(0) = 0, \quad m < n, \end{cases} \quad (1.4.11)$$

where  $D$  is a positive diagonal matrix of proper dimensions and  $A^\# = N^{-1}A^T M$  ( $M$  and  $N$  are chosen positive definite matrices). The simplest choice for  $D$  is  $D = \gamma I$ , where  $\gamma > 0$  [108].

Corresponding integral representation of the weighted Moore–Penrose inverse of a linear operator between Hilbert spaces was introduced in [110]:

$$A_{M,N}^\dagger = \int_0^\infty \exp(-A^\# A \tau) A^\# d\tau. \quad (1.4.12)$$

### 1.4.4 GNN dynamics for solving matrix equations

#### GNN for solving the matrix equation $AXB = D$

A gradient-based neural dynamical design corresponding to the matrix equation  $AXB = D$  was investigated in [80]. The model is based on the matrix-valued error function  $E(t) = D - AV(t)B$ . The model defined in [80] is termed as  $GNN(A, B, D)$  and defined as follows:

$$\frac{dV(t)}{dt} = \dot{V}(t) = \gamma A^T \mathcal{F}(D - AV(t)B) B^T. \quad (1.4.13)$$

Convergence properties of  $GNN(A, B, D)$  design are considered in [80].

**Theorem 1.4.1.** [80] *Assume that real matrices  $A \in \mathbb{R}^{m \times n}$ ,  $B \in \mathbb{R}^{p \times q}$  and  $D \in \mathbb{R}^{m \times q}$  satisfy*

$$AA^{(1)}DB^{(1)}B = D, \quad (1.4.14)$$

*for some inner inverses  $A^{(1)}$  and  $B^{(1)}$ . If an odd and monotonically increasing array activation function  $\mathcal{F}(\cdot)$  based on an elementwise function  $f(\cdot)$  is used, then the neural state matrix  $V(t) \in \mathbb{R}^{n \times p}$  of the  $GNN(A, B, D)$  model (1.4.13) asymptotically converges to the solution of the matrix equation  $AXB = D$ , i.e.,  $AV(t)B \rightarrow D$  as  $t \rightarrow +\infty$ , for an arbitrary initial state matrix  $V(0)$ .*

**Theorem 1.4.2.** [80] *Assume that the real matrices  $A \in \mathbb{R}^{m \times n}$ ,  $B \in \mathbb{R}^{p \times q}$  and  $D \in \mathbb{R}^{m \times q}$  satisfy*

$$AA^\dagger DB^\dagger B = D. \quad (1.4.15)$$

*Then the activation state variables matrix  $V(t)$  of the model  $GNN(A, B, D)$ , defined by (1.4.13), is convergent as  $t \rightarrow +\infty$  and has the equilibrium state*

$$V(t) \rightarrow \tilde{V} = A^\dagger DB^\dagger + V(0) - A^\dagger AV(0)BB^\dagger \quad (1.4.16)$$

*for every initial state matrix  $V(0) \in \mathbb{R}^{n \times p}$ .*

#### GNN for solving the matrix equation $AXA = A$

GNN model for solving the matrix equation  $AV(t)A = A$ , arising from the error function  $\varepsilon(t) = \frac{1}{2}\|A - AV(t)A\|$ , was investigated in [81]. Then the nonlinear  $GNN(A, B, D)$  model becomes the  $GNN(A, A, A)$  model, defined in [81] as the following GNN-MP model:

$$\frac{dV(t)}{dt} = \gamma A^T \mathcal{F}(A - AV(t)A) A^T = \gamma A^T \mathcal{F}(E(t)) A^T, \quad V(0) \text{ arbitrary.} \quad (1.4.17)$$



**GNN for solving the matrix equation  $BXCAB = B$** 

Solution  $\tilde{V}_{V(0)}$  of the matrix equation

$$BXCAB = B$$

which is derived by the  $GNN(B, CAB, B)$  model

$$\dot{V}(t) = B^T \mathcal{F}(B - BV(t)CAB)(CAB)^T \quad (1.4.18)$$

gives  $\tilde{V}_{V(0)} \in (CAB)\{1\}$ . Then  $X = B\tilde{V}C$  gives various representations of outer inverses according to Urquhart formula. The Urquhart formula was originated in [88] and later extended in [89, Theorem 1.3.3] and [2, Theorem 13, P. 72].

**Proposition 1.4.1.** *Let  $A \in \mathbb{R}_r^{m \times n}$ ,  $B \in \mathbb{R}^{n \times p}$ ,  $C \in \mathbb{R}^{q \times m}$  and  $X := BVC = B(CAB)^{(1)}C$ , where  $(CAB)^{(1)}$  is a fixed but arbitrary element of  $(CAB)\{1\}$ . Then*

- (1)  $X \in A\{1\}$  if and only if  $\text{rank}(CAB) = r$ ;
- (2)  $X \in A\{2\}$  and  $\mathcal{R}(X) = \mathcal{R}(B)$  if and only if  $\text{rank}(CAB) = \text{rank}(B)$ ;
- (3)  $X \in A\{2\}$  and  $\mathcal{N}(X) = \mathcal{N}(C)$  if and only if  $\text{rank}(CAB) = \text{rank}(C)$ ;
- (4)  $X = A_{\mathcal{R}(B), \mathcal{N}(C)}^{(2)}$  if and only if  $\text{rank}(CAB) = \text{rank}(B) = \text{rank}(C)$ ;
- (5)  $X = A_{\mathcal{R}(B), \mathcal{N}(C)}^{(1,2)}$  if and only if  $\text{rank}(CAB) = \text{rank}(B) = \text{rank}(C) = r$ .

The following application of the  $GNN(A, B, D)$  model is motivated by the results from [82] and the Urquhart formula.

**Corollary 1.4.1.** *Assume that  $A \in \mathbb{R}_r^{m \times n}$ ,  $B \in \mathbb{R}^{n \times k}$ , and  $C \in \mathbb{R}^{l \times m}$  satisfy  $B(CAB)^\dagger CAB = B$ .*

(i) *If an odd and monotonically increasing function  $f(\cdot)$  is used to define the array activation function  $\mathcal{F}(\cdot)$ , then the state matrix  $V(t) \in \mathbb{R}^{n \times s}$  of the  $GNN(B, CAB, B)$  model (1.4.18) satisfies  $BV(t)CAB \rightarrow B$  as  $t \rightarrow +\infty$ , for an arbitrary initial state matrix  $V(0)$ . When  $t \rightarrow +\infty$ , the matrix  $V(t)$  is convergent and its limiting value  $\tilde{V}_{V(0)}$  satisfies*

$$\tilde{V}_{V(0)} = B^\dagger B(CAB)^\dagger + V(0) - B^\dagger BV(0)CAB(CAB)^\dagger \in (CAB)^{(1)}, \quad (1.4.19)$$

for every initial matrix  $V(0) \in \mathbb{R}^{k \times l}$ .

(ii) *The following statements are valid:*

- (1) *The condition  $\text{rank}(CAB) = r$  initiates*

$$X := B\tilde{V}_{V(0)}C \in A\{1\}.$$

(2) The condition  $\text{rank}(CAB) = \text{rank}(B)$  implies

$$X := B\tilde{V}_{V(0)}C \in A\{2\}_{\mathcal{R}(B),*}.$$

(3) The condition  $\text{rank}(CAB) = \text{rank}(C)$  implies

$$X := B\tilde{V}_{V(0)}C \in A\{2\}_{*,\mathcal{N}(C)}.$$

(4) If the conditions  $\text{rank}(CAB) = \text{rank}(B) = \text{rank}(C)$  are satisfied, then

$$X := B\tilde{V}_{V(0)}C = A_{\mathcal{R}(B),\mathcal{N}(C)}^{(2)}.$$

(5) If the conditions  $\text{rank}(CAB) = \text{rank}(B) = \text{rank}(C) = r$  are satisfied, then

$$X := B\tilde{V}_{V(0)}C = A_{\mathcal{R}(B),\mathcal{N}(C)}^{(1,2)}.$$

### GNN for solving the matrix equation $GAX = G$ or $XAG = G$

The  $GNN(GA, I, G)$  model is defined for solving the matrix equation  $GAX = G$  is based on the error matrix  $E(t) = GAV(t) - G$ , where  $A \in \mathbb{R}_r^{m \times n}$  is given and  $G \in \mathbb{R}_s^{n \times m}$ ,  $0 < s \leq r$ , is appropriately chosen matrix. This model is applicable in generating the outer inverse  $A_{\mathcal{R}(G),\mathcal{N}(G)}^{(2)}$ . Corollary 1.4.2 can be derived as a consequence of Theorem 1.4.2.

**Corollary 1.4.2.** Assume that the real matrices  $A \in \mathbb{R}_r^{m \times n}$ ,  $G \in \mathbb{R}_s^{n \times m}$  satisfy  $0 < s \leq r$  and  $\text{rank}(GA) = \text{rank}(G)$ . Then the following statements hold.

(i) The unknown matrix  $V(t)$  of the model  $GNN(GA, I, G)$

$$\frac{dV(t)}{dt} = -\gamma (GA)^T (GAV(t) - G), \quad V(0) \text{ arbitrary} \quad (1.4.20)$$

is convergent when  $t \rightarrow +\infty$  and has the limit value

$$\tilde{V}_{V(0)} = (GA)^\dagger G + V(0) - (GA)^\dagger GAV(0), \quad (1.4.21)$$

for every initial matrix  $V(0) \in \mathbb{R}^{n \times m}$ .

(ii) In particular,  $V(0) = 0$  initiates  $\tilde{V}_0 = (GA)^\dagger G = A_{\mathcal{N}(GA)^\perp, \mathcal{N}(G)}^{(2,4)}$ .

**Corollary 1.4.3.** Assume that the real matrices  $A \in \mathbb{R}_r^{m \times n}$ ,  $G \in \mathbb{R}_s^{n \times m}$  satisfy  $0 < s \leq r$  and  $\text{rank}(AG) = \text{rank}(G)$ . Then the following statements hold.

(i) The unknown matrix  $V(t)$  of the dynamical model  $GNN(I, AG, G)$  satisfies

$$\frac{dV(t)}{dt} = -\gamma (V(t)AG - G) (AG)^T, \quad V(0) \text{ arbitrary}, \quad (1.4.22)$$

it is convergent when  $t \rightarrow +\infty$  and has the limit value

$$\tilde{V}_{V(0)} = G(AG)^\dagger + V(0) - V(0)AG(AG)^\dagger, \quad (1.4.23)$$

for every initial matrix  $V(0) \in \mathbb{R}^{n \times m}$ .

(ii) In particular,  $\tilde{V}_0 = G(AG)^\dagger = A_{\mathcal{N}(G), \mathcal{N}(AG)^\perp}^{(2,3)}$ .

The authors of [138] omitted the constant term  $(GA)^\top$  from (1.4.20) and the constant term  $(AG)^\top$  from (1.4.22), and considered two dual linear GNN models, defined as follows:

$$\begin{cases} \frac{dV(t)}{dt} = -\gamma (GAV(t) - G), & V(0) = 0, \text{ if } m \geq n, \\ \frac{dV(t)}{dt} = -\gamma (V(t)AG - G), & V(0) = 0, \text{ if } m < n. \end{cases} \quad (1.4.24)$$

The application of the dynamic equation (4.1.1) is conditioned by the properties of the spectrum of the matrix  $GA$  or  $AG$ :

$$\begin{cases} \sigma(GA) \subset \{z : \operatorname{Re}(z) \geq 0\}, & m \geq n, \\ \sigma(AG) \subset \{z : \operatorname{Re}(z) \geq 0\}, & m < n. \end{cases} \quad (1.4.25)$$

More precisely, the first GNN approach used in [138] fails in the case when  $\operatorname{Re}(\sigma(GA))$  contains negative values. Clearly, the model (4.1.1) is simpler than the models (1.4.20) or (1.4.22), but it loses global stability. An approach to resolve the requirement (4.1.7) and recover global stability was proposed in [138], and it is based on the replacement of  $G$  by  $G_0 = G(GAG)^\top G$  in (4.1.1). But, this approach requires additional matrix multiplications during the computation of the matrix  $G_0$  in (4.1.1) instead of the matrix  $G$ .

#### GNN for solving the matrix equation $A^k AX = A^k$

Particularly, the GNN model for computing the Drazin inverse  $A^D$  was proposed in [84]. This model can be derived removing the first constant term in  $GNN(A^k, A, A^k)$ ,  $k \geq \operatorname{ind}(A)$ , and it is defined as

$$\frac{dV(t)}{dt} = -\gamma (A^{k+1}V(t) - A^k), \quad k \geq \operatorname{ind}(A), \quad V(0) = 0. \quad (1.4.26)$$

Accordingly, an application of the model (1.4.26) is conditioned by

$$\operatorname{Re}(\lambda_j^{k+1}) \geq 0, \quad j = 1, \dots, n, \quad (1.4.27)$$

where  $\sigma(A) = \{\lambda_1, \dots, \lambda_n\}$  is the spectrum of  $A$ , and  $m \geq \operatorname{ind}(A)$  [84]. One method to resolve the limitation (4.3.11) is proposed in [84], and it is based on the possibility to find an appropriate power  $k$  such that (4.3.11) holds. Another possibility to ensure the nonnegativity

of the spectrum of the form (4.3.11) was proposed in [85], and it is based on the usage of the matrix  $A^k \left( A^{2k+1} \right)^T A^k$ ,  $k = \text{ind}(A)$ .

### GNN for computing $W$ -weighted Drazin inverse

. The  $W$ -weighted Drazin inverse of  $A \in \mathbb{R}^{m \times n}$  corresponding to  $W \in \mathbb{R}^{n \times m}$ , denoted by  $A_{d,w}$ , satisfies the matrix equation

$$(AW)^{l+2} A_{d,w} = (AW)^l A, \quad (1.4.28)$$

which initiates the dynamical system  $GNN((AW)^{l+2}, I, (AW)^l A)$  defined by

$$\frac{dV(t)}{dt} = -\gamma \left( (AW)^{l+2} \right)^T \left( (AW)^{l+2} V(t) - (AW)^l A \right), \quad l \geq k, \quad V(0) \text{ arbitrary}, \quad (1.4.29)$$

where  $k = \max\{\text{ind}(AW), \text{ind}(WA)\}$ . The dynamical system (1.4.29) is usable in approximating  $A_{d,w}$ , since Namely, the limit value  $\tilde{V}_{V(0)}$  of the matrix of activation state variables  $V(t)$  in (1.4.29) satisfies  $\tilde{V}_{V(0)} = A_{d,w}$ .

After omitting the constant term  $\left( (AW)^{l+2} \right)^T$ , the dynamic equation (1.4.29) becomes the GNN model defined in [102]:

$$\frac{dV(t)}{dt} = -\gamma \left( (AW)^{l+2} V(t) - (AW)^l A \right), \quad l \geq k, \quad V(0) \text{ arbitrary}. \quad (1.4.30)$$

## 1.5 Review of known ZNN models

On the other hand, the ZNN model for online time-invariant matrix inversion is based upon the matrix-formed error function  $E(t)$  instead of a scalar valued function  $\varepsilon(t)$ . The time derivative of error function  $E(t)$ , should be selected in such a way that each element  $e_{ij}(t)$  of  $E(t)$  converges to zero,  $\forall i = 1, \dots, n$ . A general design rule of  $\dot{E}(t)$  is defined as follows

$$\dot{E}(t) = \frac{dE(t)}{dt} = -\gamma \mathcal{F}(E(t)). \quad (1.5.1)$$

Again,  $\mathcal{F}(C)$  is an odd and monotonically increasing function array, and  $\gamma$  is a positive scaling constant.

### 1.5.1 ZNN for computing regular inverse

Matrix error function can be defined as  $E(t) = AX(t) - I$ . Substituting it into dynamic system (1.5.1) and choosing  $\mathcal{F}$  to be the linear function, the following Zhang dynamics for

computing regular inverse of a non singular real constant matrix can be can be obtained:

$$A\dot{X} = -\gamma AX(t) + \gamma I, \quad (1.5.2)$$

where  $X(t)$  is the matrix of activation state variables. The implicit dynamics were originally proposed for online inversion of a time-varying matrix  $A(t)$  in [122]. It was shown in [122] that the Zhang dynamics globally exponentially converges to the theoretical inverse  $A^{-1}$ , starting from any initial state  $X(0)$ , with the exponential convergence rate  $\gamma$ .

### 1.5.2 ZNN for computing Moore-Penrose inverse

The starting point in [48, 132] was the fact that the left Moore-Penrose inverse  $A(t)^\dagger$  satisfies  $A(t)^* A(t) A(t)^\dagger$ . Further, on the basis of the assumption that  $A(t)^* A(t)$  is invertible, the following matrix-based error function, called ZF(5), is considered

$$E(t) := A(t)^* A(t) X(t) - A(t)^*,$$

where  $X(t)$  corresponds to  $A(t)^\dagger$ . An elegant way to avoid the assumption of the invertibility of  $A(t)^* A(t)$  was presented in [48]. Namely, the authors of [48] defined the complex ZF which arises from the ZF defined in (5.1.2), and the Tikhonov regularization:

$$E(t) = (A(t)^* A(t) + \lambda I) X(t) - A(t)^*, \quad \lambda > 0.$$

The resulting ZNN model (5.1.3) is termed as complex ZNN-II Model.

### 1.5.3 ZNN for computing the Drazin inverse

In addition, the following complex function was used as the fundamental error-monitoring function (called ZFL2) in [104]:

$$E(t) = (A(t)^{l+1} + \lambda I) X(t) - A(t)^l \quad l \geq k = \text{ind}(A), \quad \lambda > 0.$$

The matrix  $X(t)$  in (5.1.4) corresponds to the Drazin inverse  $A(t)^D$ . Let us mention that the ZNN-II model in [67] is defined on the basis of the ZF (5.1.4) and upon the Li activation function.

### 1.5.4 ZNN for computing outer inverse

The starting point in generating the ZNNATS2-I model arises from Lemma 1.5.1 from [105].

**Lemma 1.5.1.** *Let  $A(t) \in \mathbb{C}_r^{m \times n}$  be given and  $G(t) \in \mathbb{C}_s^{n \times m}$  be an arbitrary matrix whose rank satisfies  $0 < s \leq r$ . Assume that  $X(t) := A(t)_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  exists. Then both the matrix identities*

$$G(t)A(t)X(t) = G(t), \quad X(t)A(t)G(t) = G(t)$$

*are satisfied.*

The leading idea of [105] was to comprise so far known ZNN models for computing generalized inverses into a unique comprehensive model corresponding to outer inverses in the time-varying complex matrix case. The ZNNATS2-I model defined in [105] requires two matrices  $A(t) \in \mathbb{C}_r^{m \times n}$ ,  $G(t) \in \mathbb{C}_s^{n \times m}$ ,  $0 < s \leq r$ , and it is aimed to numerical computations of the outer inverse  $A(t)_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$ . The model is developed using the following two dual fundamental error-monitoring ZFs, proposed in [105]:

$$E_G(t) = \begin{cases} (G(t)A(t) + \lambda I) X(t) - G(t), & n \leq m, \lambda > 0 \\ X(t) (A(t)G(t) + \lambda I) - G(t), & n > m, \lambda > 0. \end{cases}$$

## 1.6 Outline of the dissertation

In the following three chapters we introduce and study gradient-based recurrent neural networks (GNNs) for computing the generalized inverses of a constant real matrix in real-time. Recurrent neural networks are composed of independent parts (sub-networks). These sub-networks can work simultaneously, so parallel and distributed processing can be accomplished. In this way, the computational advantages over the existing sequential algorithms can be attained in real-time applications. The proposed GNNs can be easily implemented in an electronic circuit. The number of neurons in the neural network is the same as the number of elements in the output matrix, which represents the requested solution. We consider the conditions that guarantee the stability of the defined GNNs as well as its convergence toward the inverse. We show the efficacy of the proposed neural network models through illustrative computer simulation and examples of application to the practical engineering problems.

Chapter 2 presents GNNs with dynamics conditioned by the properties of the spectrum of a certain matrix. The chapter is based on the results published in papers:

[84] Predrag S. Stanimirović, Ivan S. Živković, and Yimin Wei. "Recurrent neural network for computing the Drazin inverse." *IEEE transactions on neural networks and learning systems* 26.11 (2015): 2830-2843.

[138] Ivan S. Živković, Predrag S. Stanimirović, and Yimin Wei. "Recurrent neural network for computing outer inverse." *Neural computation* 28.5 (2016): 970-998.

Chapter 3 resolves the drawback of the GNN models from Chapter 2, at the cost of increasing the number of matrix operations. The results presented are based on the papers:

[85] Predrag S. Stanimirović, Ivan S. Živković, and Yimin Wei. "Recurrent neural network approach based on the integral representation of the Drazin inverse." *Neural computation* 27.10 (2015): 2107-2131.

[138] Ivan S. Živković, Predrag S. Stanimirović, and Yimin Wei. "Recurrent neural network for computing outer inverse." *Neural computation* 28.5 (2016): 970-998.

Chapter 4 emphasizes the equivalence between two well known general representations of outer inverses with prescribed range and null space of a given matrix. Two dynamic state equations, corresponding to particular expressions related to these representations, are defined. In this way, two gradient based neural networks, initiated by introduced dynamic equations, are exploited in generating the class of outer inverses. The results corresponding to the most common generalized inverses are obtained in particular cases. Simulation results are presented at the end of the chapter. The chapter is based on the results from:

[86] Predrag S. Stanimirović, Ivan S. Živković, and Yimin Wei. "Neural network approach to computing outer inverses based on the full rank representation." *Linear Algebra and Its Applications* 501 (2016): 344-362.

In Chapter 5 we investigate and exploit an analogy between the scaled hyperpower family (SHPI family) of iterative methods for computing the matrix inverse and the discretization of Zhang Neural Network (ZNN) models. We define a class of ZNN models corresponding to the family of hyperpower iterative methods for computing generalized inverses on the basis of the discovered analogy. The Matlab Simulink implementation of the introduced ZNN models is described in the case of scaled hyperpower methods of the order 2 and 3. Convergence properties of the proposed ZNN models are investigated as well as their numerical behavior. This chapter is based on the paper:

[87] Igor Stojanović, Predrag S. Stanimirović, Ivan S. Živković, Dimitrios Gerontitis, Xue-Zhong Wang, ZNN models for computing matrix inverse based on hyperpower iterative methods. *Filomat*, 31(10) (2017), 2999-3014.

In Chapter 6 our goal is to compare a novel kind of a hybrid recursive neural model with implicit dynamics and a conventional neural model with explicit dynamics. Through the simulation results we show the hybrid model can coincide better with systems in practice and has higher abilities in representing dynamic systems. More importantly, hybrid model can achieve superior convergence performance in comparison with the existing dynamic systems, specifically ZNN dynamics. We present the Matlab Simulink model of a hybrid recursive neural

implicit dynamics and give a simulation and comparison to the existing Zhang dynamics for real-time matrix inversion. Simulation results confirm a superior convergence of the hybrid model compared to the classical Zhang model. The chapter is based on the following paper:

[137] Ivan S. Živković, and Predrag S. Stanimirović. "Matlab simulation of the hybrid of recursive neural dynamics for online matrix inversion." *Facta Universitatis, Series: Mathematics and Informatics* (2018): 799-809.



# Chapter 2

## GNN for computing generalized inverses with restrictions on spectrum

*This chapter presents a gradient-based recurrent neural networks (GNNs) for computing the generalized inverses of a real constant matrix in a real time. These models are created of a certain number of independent sub-networks, which can operate in parallel. In this way, the computational advantages over the existing sequential algorithms can be attained in real-time applications. The GNNs introduced are convenient for an implementation in an electronic circuit. The conditions that guarantee the stability of the defined GNNs as well as its convergence toward the inverse are considered. In addition, illustrative examples and examples of application to the practical engineering problems are discussed in order to show the efficacy of the proposed neural networks.*

### 2.1 GNN for computing the Drazin Inverse

#### 2.1.1 Preliminaries and motivation

We repeat the definition of the Drazin inverse, for the sake of the completeness (for more details see [2, 89]). Let  $A \in \mathbb{C}^{n \times n}$  and  $\text{ind}(A) = k$ . Then the matrix  $X \in \mathbb{C}^{n \times n}$  satisfying

$$A^{l+1}X = A^l, \quad l \geq \text{ind}(A), \quad (2.1.1)$$

$$XAX = X, \quad (2.1.2)$$

$$AX = XA \quad (2.1.3)$$

is called the Drazin inverse of  $A$ , and it is denoted by  $X = A^{\text{D}}$ .

If  $A$  is nonsingular, then  $\text{ind}(A) = 0$  and  $A^{\text{D}} = A^{-1}$ . Otherwise, if  $A$  is singular, then  $\text{ind}(A) \geq 1$ . Also, in the case  $\text{ind}(A) = 1$  the Drazin inverse becomes the group inverse

$X = A^\#$  [69].

In [24, 115] Castro, Koliha and Wei derived a simple integral representation of the Drazin inverse  $A^D$  for matrices  $A \in \mathbb{C}^{n \times n}$  (and more generally elements of a Banach algebra) for which the nonzero eigenvalues of  $A^{m+1}$  lie in the open right half plane for some  $m \geq \text{ind}(A)$ :

$$A^D = \int_0^\infty \exp(-A^{m+1}\tau)A^m d\tau. \quad (2.1.4)$$

In the current section we present the main results from [84]. Developed GNN design represent a continuation of the results derived in [94, 96, 108] from the usual inverse and the pseudoinverse to the Drazin inverse. A dynamic state equation of the neural network corresponds to the integral representation (2.1.4). We also provide the conditions which must be imposed to the spectrum of  $A$  which ensure the stability of the neural network. A specific neural network approach to compute the Drazin inverse  $A^D$  was developed in [14]. The approach from [14] is based on a multi-layer neural network arising from the system of matrix equations (2.1.1)-(2.1.3).

## 2.1.2 Neural network architecture for computing Drazin inverse

### Dynamic state equation for computing the Drazin inverse

We will assume  $A \in \mathbb{R}^{n \times n}$ , i.e., our investigation will be limited to the real square matrices.

Let's observe the matrix equation (2.1.1). We can rewrite it as follows

$$A^{m+1}V - A^m = 0, \quad (2.1.5)$$

where  $m \geq \text{ind}(A)$ , and  $V \in \mathbb{R}^{n \times n}$  denotes the unknown matrix corresponding to  $A^D$ . To solve (2.1.5) for  $V$  via dynamic-system approach, we can define a scalar-valued norm based error function:

$$E(t) = \frac{\|A^{m+1}V(t) - A^m\|_F^2}{2}, \quad m \geq \text{ind}(A).$$

Note that the minimal value  $E(\bar{t}) = 0$  of the residual-error function  $E(t)$  is achieved in a minimizer  $\bar{V} = V(\bar{t})$  if and only if  $V(\bar{t})$  is the exact solution of (2.1.5). A computational scheme could be designed to evolve along a descent direction of this error function  $E(t)$ , until the minimum  $E(\bar{t})$  is reached. The typical descent direction of  $E(t)$  is defined by the negative gradient  $-\partial E(t)/\partial V$  of  $E(t)$ . The gradient of  $E$  with respect to  $V \in \mathbb{R}^{n \times n}$  could simply be derived as (see, for example, [25, Chapter 5])

$$\frac{\partial E(t)}{\partial V} = (A^{m+1})^T (A^{m+1}V(t) - A^m). \quad (2.1.6)$$

As a consequence, the GNN model for computing the Drazin inverse is given by

$$\begin{aligned} \frac{dV(t)}{dt} &= -\gamma (A^{m+1})^T (A^{m+1}V(t) - A^m), \\ m &\geq \text{ind}(A), V(0) = V_0. \end{aligned} \quad (2.1.7)$$

The GNN dynamics (2.1.7) will be termed as GNNAD.

The first term  $(A^{m+1})^T$  in the right hand side of (2.1.6) can be considered as a constant factor. Therefore, the gradient direction of  $E(t)$  is also defined by  $(A^{m+1}V(t) - A^m)$ , so  $(A^{m+1})^T$  can be removed.

According to design formula  $dV(t)/dt = -\gamma \partial E(t)/\partial V$ , and by excluding the constant term from (2.1.6), the authors in [84] defined the dynamic equation of a gradient recurrent neural network as follows

$$\begin{aligned} \frac{dV(t)}{dt} &= -\gamma (A^{m+1}V(t) - A^m), \\ m &\geq \text{ind}(A), V(0) = V_0. \end{aligned} \quad (2.1.8)$$

The reason for this definition could be the fact that if we find the equilibrium state  $\bar{V} = V(\bar{t})$  for dynamic system (2.1.8), then it is clear that the minimum for the residual-norm function  $E(t)$  is also achieved, because the following is satisfied

$$\frac{d\bar{V}}{dt} = 0 \quad (2.1.9)$$

at the equilibrium state  $\bar{V}$ . Thus,

$$-\gamma (A^{m+1}\bar{V} - A^m) = 0, \quad (2.1.10)$$

and, therefore (2.1.5) holds.

Here,  $V(t)$  is a matrix of activation state variables,  $t \in [0, +\infty)$ ,  $\gamma$  is a positive scaling constant which should be established as large as the hardware permits, or selected appropriately for simulative and/or experimental purposes [94, 97]).

The recurrent neural network defined in (2.1.8) is a linear dynamic system in a matrix form. According to the linear systems theory [35], the closed-form solution of the state matrix can be described as follows:

$$V(t) = \exp(-\gamma A^{m+1}t)V(0) + \gamma \exp(-\gamma A^{m+1}t) \int_0^t \exp(\gamma A^{m+1}\tau) A^m d\tau. \quad (2.1.11)$$

To analyze the convergence and stability of a neural network it is necessary to know the eigenvalues of  $A^m$ .

**Proposition 2.1.1.** [1, Page 164] *If  $x$  is an eigenvector corresponding to an eigenvalue  $\lambda \in \mathbb{C}$  of the matrix  $A \in \mathbb{R}^{n \times n}$ , then  $x$  is an eigenvector of  $A^m$  associated with the eigenvalue  $\lambda^m$ , for any integer  $m \geq 0$ .*

According to Proposition 2.1.1 the following useful result is obtained in [84].

**Lemma 2.1.1.** [84] *Let  $A \in \mathbb{R}^{n \times n}$  be given singular matrix and  $\sigma(A) = \{\lambda_1, \dots, \lambda_n\}$  be the spectrum of  $A$ , and  $m \geq 0$ . Suppose that the condition*

$$\operatorname{Re}(\lambda_j^{m+1}) \geq 0, \quad j = 1, 2, \dots, n \quad (2.1.12)$$

*is satisfied. Then the following holds:*

$$\lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1}t) = 0. \quad (2.1.13)$$

*Proof.* The Jordan normal form of the matrix  $A^{m+1}$  is defined by [2, 7]

$$A^{m+1} = P \operatorname{diag} [J_1, J_2, \dots, J_p] P^{-1},$$

where  $P$  is invertible matrix and *diag* operator denotes a block diagonal matrix whose Jordan blocks  $J_k$ ,  $k = 1, 2, \dots, p$  lie on the main diagonal and are equal to

$$\begin{aligned} J_k &= J_k(\lambda_k^{m+1}) \\ &= \begin{bmatrix} \lambda_k^{m+1} & 1 & & \\ & \lambda_k^{m+1} & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k^{m+1} \end{bmatrix} \in \mathbb{R}^{n_k \times n_k}. \end{aligned}$$

The matrix exponential of  $-\gamma A^{m+1}t$  is defined by (see [21, Chapter 11], [28])

$$\exp(-\gamma A^{m+1}t) = P \operatorname{diag} [\exp(-\gamma t J_1), \exp(-\gamma t J_2), \dots, \exp(-\gamma t J_p)] P^{-1}.$$

By applying known results from [28], it is not difficult to conclude

$$\exp(-\gamma t J_k) = \exp(-\gamma t \lambda_k^{m+1}) \begin{bmatrix} 1 & \frac{-\gamma t}{1!} & \cdots & \frac{(-\gamma t)^{n_k-1}}{(n_k-1)!} \\ & 1 & \ddots & \vdots \\ & & \ddots & \frac{-\gamma t}{1!} \\ & & & 1 \end{bmatrix}, \quad (2.1.14)$$

$$k = 1, 2, \dots, p.$$

The power  $\lambda_k^{m+1}$  of the eigenvalue  $\lambda_k$  is a complex number

$$\lambda_k^{m+1} = u_k + \imath v_k,$$

where  $\imath$  denotes the imaginary unit and  $u_k, v_k$  mean the real and imaginary part of  $\lambda_k^{m+1}$ , respectively. According to the assumption (2.1.12) the inequality

$$-\gamma \operatorname{Re}(\lambda_k^{m+1}) = -\gamma u_k \leq 0$$

is valid for each  $\lambda_k \in \sigma(A)$ . Since real parts of all eigenvalues contained in the spectrum  $\sigma(A^{m+1})$  are nonnegative, it follows that

$$\begin{aligned} \lim_{t \rightarrow \infty} \exp(-\gamma t \lambda_k^{m+1}) \cdot (-\gamma t)^l &= \\ \lim_{t \rightarrow \infty} \exp(t(-\gamma u_k)) (\cos(-\gamma v_k t) + \imath \sin(-\gamma v_k t)) \cdot (-\gamma t)^l &= \\ &= 0, \end{aligned}$$

for each  $l, 0 \leq l \leq n_k - 1$ . Therefore, according to (2.1.14),  $\lim_{t \rightarrow \infty} \exp(-\gamma t J_k)$  is the zero matrix for each  $k = 1, 2, \dots, p$ . This further implies that  $\lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1} t)$  is the zero matrix.  $\square$

Now, equation (2.1.11) and Lemma 2.1.1 imply the following representation for  $\lim_{t \rightarrow \infty} V(t) = \bar{V}$ :

$$\bar{V} = \lim_{t \rightarrow \infty} \gamma \exp(-\gamma A^{m+1} t) \int_0^t \exp(\gamma A^{m+1} \tau) A^m d\tau. \quad (2.1.15)$$

It can be verified, based on the definition and properties of matrix exponential, that the closed-form solution of  $\bar{V}$  in (2.1.15) represents the Drazin inverse, and it is independent of the parameter  $\gamma$ .

**Theorem 2.1.1.** [84] *Let  $A \in \mathbb{R}^{n \times n}$  be given matrix,  $\sigma(A) = \{\lambda_1, \dots, \lambda_n\}$  be the spectrum of  $A$ , and  $m \geq \text{ind}(A)$ . Suppose that the condition (2.1.12) is satisfied. Then the limiting expression (2.1.15) produces the Drazin inverse  $A^D$ , i.e.,*

$$\bar{V} = A^D.$$

*Proof.* Applying several elementary transformations and using basic properties of the Drazin inverse, one can verify

$$\begin{aligned} \bar{V} &= \lim_{t \rightarrow \infty} \gamma \exp(-\gamma A^{m+1}t) \int_0^t \exp(\gamma A^{m+1}\tau) A^m d\tau \\ &= \lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1}t) \int_0^t \exp(\gamma A^{m+1}\tau) \gamma A^m d\tau \\ &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1}t) \int_0^t \exp(\gamma A^{m+1}\tau) (\gamma A^{m+1}) d\tau \right] A^D \\ &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1}t) \int_0^t d \left( \exp(\gamma A^{m+1}\tau) \right) \right] A^D \\ &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1}t) \exp(\gamma A^{m+1}\tau) \Big|_{\tau=0}^{\tau=t} \right] A^D \\ &= \left\{ \lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1}t) \left[ \exp(\gamma A^{m+1}t) - I \right] \right\} A^D \\ &= \left[ I - \lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1}t) \right] A^D. \end{aligned}$$

According to Lemma 2.1.1, we conclude

$$\bar{V} = A^D,$$

which completes the proof.  $\square$

The equation (2.1.11) can be simplified by forcing the first matrix term in the right-hand side to be zero by setting zero initial states, i.e.  $V(0) = 0$ . In light of the above discussion, the dynamic state equation of the recurrent neural network for computing the Drazin inverse can be described as follows:

$$\begin{aligned} \frac{dV(t)}{dt} &= -\gamma \left( A^{m+1}V(t) - A^m \right), \\ m &\geq \text{ind}(A), V(0) = 0, \end{aligned} \tag{2.1.16}$$

where  $\gamma$  is a positive real constant. The model (2.1.16) will be termed as GNND model.

Now we are investigating the stability of the equilibrium state  $\bar{V}$ . Before the main result, which shows that the equilibrium state is stable in the sense of Lyapunov, we restate two auxiliary results.

**Proposition 2.1.2.** [38] *If  $C, D$  are two  $n \times n$  real symmetric positive semidefinite matrices. Then*

$$\lambda_{\min}(C) \operatorname{Tr}(D) \leq \operatorname{Tr}(CD) \leq \lambda_{\max}(C) \operatorname{Tr}(D), \quad (2.1.17)$$

where  $\lambda_{\min}(A)$  (resp.  $\lambda_{\max}(A)$ ) denote the smallest (resp. the largest) eigenvalue of  $A$ .

**Lemma 2.1.2.** [1] *Let  $M$  and  $N$  be two positive semi-definite matrices. Then the following statements hold:*

- (a)  $\operatorname{Tr}(MN) \geq 0$ .
- (b)  $\operatorname{Tr}(M) = 0 \iff M = 0$ .

In Lemma 2.1.3 we show that the Hermitian part of a square real matrix  $M$ , defined by

$$H(M) = \frac{1}{2} (M + M^T),$$

possesses a useful property with respect to the trace function.

**Lemma 2.1.3.** [84] *Let  $M$  and  $N$  be two real square matrices of the order  $n$ . If the matrix  $N$  is symmetric, then the following equality is valid:*

$$\operatorname{Tr}(MN) = \operatorname{Tr}(H(M)N). \quad (2.1.18)$$

*Proof.* Let us denote  $(ij)$ th element of  $M$  (resp.  $N$ ) by  $m_{ij}$  (resp.  $n_{ij}$ ). By applying definition of the trace function and several algebraic transformations, we obtain:

$$\begin{aligned} \operatorname{Tr}(H(M)N) &= \sum_{i=1}^n (H(M)N)_{ii} \\ &= \sum_{i=1}^n \sum_{j=1}^n H(M)_{ij} n_{ji} \\ &= \frac{1}{2} \left( \sum_{i=1}^n \sum_{j=1}^n m_{ij} n_{ji} + \sum_{i=1}^n \sum_{j=1}^n m_{ji} n_{ji} \right). \end{aligned}$$

Since the matrix  $N$  is symmetric, we further have

$$\begin{aligned} \operatorname{Tr}(H(M)N) &= \frac{1}{2} \left( \sum_{i=1}^n \sum_{j=1}^n m_{ij} n_{ji} + \sum_{j=1}^n \sum_{i=1}^n m_{ji} n_{ij} \right) \\ &= \frac{1}{2} \left( \sum_{i=1}^n (MN)_{ii} + \sum_{j=1}^n (MN)_{jj} \right) \\ &= \operatorname{Tr}(MN), \end{aligned}$$

which was our initial intention.  $\square$

**Theorem 2.1.2.** [84] *Let  $A \in \mathbb{R}^{n \times n}$  be given singular matrix and  $\sigma(A) = \{\lambda_1, \dots, \lambda_n\}$  be the spectrum of  $A$ . Suppose that (2.1.12) holds. Then the gradient-based neural network model GNND, defined in (2.1.16), is stable in the sense of Lyapunov.*

*Proof.* To prove stability we use the Lyapunov's method [63, Chapter 2.9]. We construct the following Lyapunov function candidate

$$E(t) = \frac{\|A^{m+1}V - A^m\|_F^2}{2} = \frac{\text{Tr} \left( (A^{m+1}V - A^m)^T (A^{m+1}V - A^m) \right)}{2}.$$

Clearly, the inequality  $E(t) \geq 0$  holds. Applying the matrix calculus (see, for example [25, Chapter 5]) on  $E(t)$ , we derive the following transformations:

$$\begin{aligned} \frac{\partial E(t)}{\partial V} &= \frac{1}{2} \frac{\partial \text{Tr}[V^T (A^{m+1})^T A^{m+1} V]}{\partial V} - \frac{1}{2} \frac{\partial \text{Tr}[V^T (A^{m+1})^T A^m]}{\partial V} - \frac{1}{2} \frac{\partial \text{Tr}[(A^m)^T A^{m+1} V]}{\partial V} \\ &= \frac{1}{2} \left\{ [(A^{m+1})^T A^{m+1}] + [(A^{m+1})^T A^{m+1}]^T \right\} V - (A^{m+1})^T A^m \\ &= (A^{m+1})^T A^{m+1} V - (A^{m+1})^T A^m \\ &= (A^{m+1})^T (A^{m+1} V - A^m). \end{aligned}$$

Thus, the time derivative of  $E(t)$  is equal to

$$\begin{aligned} \frac{dE(t)}{dt} &= \text{Tr} \left( \left( \frac{\partial E}{\partial V} \right)^T \frac{dV}{dt} \right) \\ &= \text{Tr} \left( (A^{m+1} V - A^m)^T A^{m+1} (-\gamma (A^{m+1} V - A^m)) \right) \\ &= -\gamma \text{Tr} \left( A^{m+1} (A^{m+1} V - A^m) (A^{m+1} V - A^m)^T \right). \end{aligned}$$

From the identity (2.1.18) of Lemma 2.1.3 we further get

$$\frac{dE(t)}{dt} = -\gamma \text{Tr} \left( H(A^{m+1}) (A^{m+1} V - A^m) (A^{m+1} V - A^m)^T \right).$$

Since the eigenvalues of a square matrix are equal to the eigenvalues of its transpose, then the assumption (2.1.12) implies that the nonnegativity  $\text{Re}(\lambda_j^{m+1}) \geq 0$  is valid for each  $\lambda_j^{m+1} \in \sigma(H(A^{m+1}))$ . Moreover, since  $H(A^{m+1})$  is symmetric, it is positive semi-definite. The matrices  $H(A^{m+1})$  and  $(A^{m+1} V - A^m) (A^{m+1} V - A^m)^T$  are both positive semi-definite, so that the trace of their product is nonnegative, according to Lemma 2.1.2, part (a). Therefore,

$$\frac{dE(t)}{dt} \leq 0,$$

so the gradient-based neural network GNND is stable.  $\square$



**Corollary 2.1.1.** [84] *Let  $A \in \mathbb{R}^{n \times n}$  be given nonsingular matrix and (2.1.12) holds. Then the GNND neural network (2.1.16) is globally asymptotic stable in the sense of Lyapunov.*

*Proof.* For the sake of simplicity, we use the following notations:

$$C = H(A^{m+1}), \quad D = (A^{m+1}V - A^m) (A^{m+1}V - A^m)^T.$$

By the Lyapunov stability theory, the gradient-based neural network (2.1.16) is asymptotically stable if the inequality  $dE(t)/dt < 0$  is satisfied for any non-equilibrium state  $V$ , and

$$\frac{dE(t)}{dt} = 0$$

only for  $dV(t)/dt = 0$  (at the equilibrium state  $V$ ). In our case, it is necessary to verify  $\text{Tr}(CD) = 0 \iff D = 0$ .

If  $D = 0$  holds, applying Lemma 2.1.2, part (b), we obtain  $\text{Tr}(D) = 0$ . Further, from (2.1.17)  $\text{Tr}(CD) = 0$  immediately follows.

On the other hand, let us assume  $\text{Tr}(CD) = 0$ . An application of Proposition 2.1.2 leads to  $\lambda_{\min}(C) \text{Tr}(D) \leq 0$ . Nonsingularity of  $C$  (i.e.,  $\lambda_{\min}(C) \neq 0$ ) in conjunction with  $\text{Tr}(D) \geq 0$  further implies  $\text{Tr}(D) = 0$ . Since  $D$  is positive semidefinite, we obtain  $D = 0$  and further  $dV(t)/dt = 0$ .

Moreover, the Lyapunov function  $E(V(t))$  is radially unbounded, therefore according to Theorem 4.2 [37, Chapter 4.1, page 124] the gradient-based neural network (2.1.16) is globally asymptotic stable.  $\square$

According to the Theorem 2.1.1, we can conclude that our goal is to find the integer  $m$  such that the matrix  $A^{m+1}$  has nonnegative real parts of eigenvalues, i.e.,

$$\sigma(A^{m+1}) \subset \{z : \text{Re}(z) \geq 0\}. \quad (2.1.19)$$

Conditions ensuring that the non-zero spectrum of  $A^{m+1}$  lies in the open left/right half of the complex plane as an equivalent to the condition that the non-zero spectrum of  $A$  lies in the union of  $m + 1$  angular regions are found in [24]. Therefore, the authors in [24] found conditions for the arguments of eigenvalues. If the required conditions are not satisfied, the Drazin inverse can not be generated. We develop a different approach: find corresponding conditions for the matrix exponent  $k$ , such that  $A^{k+1}$  satisfies (2.1.19).

There are several cases for selecting the parameter  $m$  which guarantee nonnegativity of real parts for all eigenvalues of the matrix  $A^{m+1}$ . These cases are discussed in Theorem 2.1.3 from [84]. Before the main results, we present several supporting facts and notations.

**Lemma 2.1.4.** [84] *Let  $A \in \mathbb{R}^{n \times n}$  be given matrix,  $\lambda_j \in \sigma(A)$  and  $k$  be a fixed positive integer. Then the condition  $\operatorname{Re}(\lambda_j^k) \geq 0$  is ensured in the following cases:*

C1.  $\lambda_j \in \mathbb{R}^+$ .

C2.  $\lambda_j \in \mathbb{R}^-$ ,  $k$  is even.

C3.  $\lambda_j$  and  $k$  satisfy

$$\varphi_j = \operatorname{Arg} \lambda_j = \pm \frac{\pi}{2}, \quad k \in \{4l : l \in \mathbb{N}^+\}, \quad (2.1.20)$$

where  $\operatorname{Arg}$  denotes the argument of a complex number.

C4.  $\lambda_j$  satisfies

$$\lambda_j \in \mathbb{C} \cap \{z = r_j e^{i\varphi_j} : 0 < |\varphi_j| < \pi\} \quad (2.1.21)$$

and the parameter  $k$  satisfies

$$\frac{4s-1}{2} \frac{\pi}{|\varphi_j|} < k < \frac{4s+1}{2} \frac{\pi}{|\varphi_j|}, \quad s \in \mathbb{N}_0, \quad (2.1.22)$$

*Proof.* The proof in the cases C1–C3 is evident. In the case C4 it follows that

$$\lambda_j = a_j + i b_j = r_j e^{i\varphi_j} = r_j (\cos(\varphi_j) + i \sin(\varphi_j)),$$

where

$$r_j = |\lambda_j| = \sqrt{a_j^2 + b_j^2},$$

$$\varphi_j = \begin{cases} -1, & a_j = 0, b_j = 0 \\ \frac{\pi}{2}, & a_j = 0, b_j > 0 \\ -\frac{\pi}{2}, & a_j = 0, b_j < 0 \\ \arctan\left(\frac{b_j}{a_j}\right), & a_j \neq 0. \end{cases}$$

The real part of

$$\lambda_j^k = r_j^k (\cos(k \varphi_j) + i \sin(k \varphi_j))$$

is equal to

$$\operatorname{Re}(\lambda_j^k) = r_j^k \cos(k \varphi_j).$$

Therefore,  $\operatorname{Re}(\lambda_j^k) > 0$  is satisfied in the case  $\cos(k |\varphi_j|) > 0$ , or equivalently

$$-\frac{\pi}{2} + 2s\pi < k |\varphi_j| < \frac{\pi}{2} + 2s\pi, \quad s \in \mathbb{N}_0.$$

Therefore, the condition (2.1.22) in conjunction with  $\gamma > 0$  implies  $\gamma \operatorname{Re}(\lambda_j^k) \geq 0$ . The length of the interval (2.1.22) which gives bounds for possible values of  $k$  is equal to  $\pi/|\varphi_j|$  so that

the condition

$$\frac{\pi}{|\varphi_j|} > 1 \quad (2.1.23)$$

must be satisfied. Therefore, necessity of (2.1.21) and (2.1.22) is verified.  $\square$

**Theorem 2.1.3.** [84] *Let  $A \in \mathbb{R}^{n \times n}$  be a given matrix of index  $k = \text{ind}(A)$  whose eigenvalues are  $\sigma(A) = \{\lambda_j, j = 1, 2, \dots, n\}$ . For each  $\lambda_j$  we consider the set of possible values of the exponent  $m$  which ensure  $\text{Re}(\lambda_j^m) \geq 0$ . The set  $M_1^j$  is defined by  $M_1^j = [l, +\infty]$ ,  $l \geq \text{ind}(A)$ , for each  $j = 1, 2, \dots, n$ . Let  $M_2^j$  be the set of values of the parameter  $m$  defined by applying the restrictions imposed in the case C2 from Lemma 2.1.4 on the basis of known value  $\lambda_j \in \sigma(A)$ . Similarly, by  $M_3^j, M_4^j$  we denote the sets of admissible values of the parameter  $m$  defined by applying the cases C3 and C4 from Lemma 2.1.4, respectively.*

Therefore,

$$\begin{aligned} M_2^j &= \{m = 2l : l \in \mathbb{N}^+\}, \quad M_3^j = \{m = 4l : l \in \mathbb{N}^+\}, \\ M_4^j &= \left\{ m : \frac{4l-1}{2} \frac{\pi}{|\varphi_j|} < m < \frac{4l+1}{2} \frac{\pi}{|\varphi_j|}, \quad l \in \mathbb{N}_0 \right\}. \end{aligned}$$

Later, the set of positive integers corresponding to  $\lambda_j$  we denote by  $U_j$ . It is easy to conclude  $U_j = M_2^j$  or  $U_j = M_3^j$  or  $U_j = M_4^j$ . The recurrent neural network defined by (2.1.16) is stable and the steady-state matrix of the recurrent neural network is equal to  $A^D$ , i.e.,

$$\lim_{t \rightarrow \infty} V(t) = A^D$$

in the following cases:

Case 1.  $\sigma(A) \subset \mathbb{R}^+, m \geq \text{ind}(A)$ .

Case 2.  $\sigma(A) \subset \mathbb{R}, \lambda_j < 0$  for at least one index  $j$  and  $m$  satisfies  $m \geq \text{ind}(A)$ ,  $m+1$  is even.

Case 3. The spectrum of  $A$  satisfies

$$\sigma(A) \subset \mathbb{C} \cap \{z = r e^{i\theta} : 0 < |\theta| < \pi\}, \quad (2.1.24)$$

and the parameter  $m$  satisfies (2.1.25):

$$m+1 = \min \left( \bigcap_{j=1}^n (U_j \cap M_1^j) \right), \quad (2.1.25)$$

where  $U_j = M_1^j$  in the case  $\lambda_j \geq 0$ .

*Proof.* It is necessary to ensure the existence of the condition (2.1.19). Also, to ensure the existence of  $A^D$ , the condition  $m \geq \text{ind}(A)$  must be satisfied in all cases. The proof in Case 1 and Case 2 is obvious from Lemma 2.1.4, since from Proposition 2.1.1 we have  $\sigma(A^{m+1}) = \{\lambda_j^{m+1}, j = 1, 2, \dots, n\}$ . In Case 1 and Case 2 the conditions (2.1.19) are satisfied. According

to Lemma 2.1.4, the condition (2.1.24) ensures  $\operatorname{Re}(\lambda_j^m) \geq 0$ . Finally, (2.1.24) and (2.1.25) ensure (2.1.19).  $\square$

The choice of  $m$  in accordance with the condition (2.1.25) in conjunction with  $\gamma > 0$  ensures that the nonzero spectrum of the matrix  $A^{m+1}$  from (2.1.8) lies in the open right half of the complex plane.

### Network configuration of GNND model

Similarly with the recurrent neural networks for matrix inversion [94, 96]), neural network is composed from a number of independent sub-networks. Each sub-network represents a column vector of  $V(t)$ . Let us define  $v_j(t)$  (resp.  $a_j^m$ ) as the  $j$ th column vector of  $V(t)$  (resp.  $A^m$ ), for  $j = 1, 2, \dots, n$ . The dynamics of the  $j$ th sub-network can be expressed as follows:

$$\frac{dv_j(t)}{dt} = -\gamma (A^{m+1}v_j(t) - a_j^m). \quad (2.1.26)$$

The dynamics defined in (2.1.26) indicates that each sub-network is basically similar to the recurrent neural network presented in [97]. Let  $W$  be the connection weight matrix. The connection weight matrix  $-\gamma A^{m+1}$  is identical for each sub-network (i.e.,  $W = -\gamma A^{m+1}$ ) and the biasing threshold vector for the  $j$ th sub-network is

$$\gamma a_j^m = \{\gamma a_{1j}^m, \dots, \gamma a_{nj}^m\}.$$

Elements of the matrix  $W$  (resp.  $V$ ) are denoted by  $w_{ij}$  (resp.  $v_{ij}$ ). Figure 2.1 depicts the architecture of the proposed recurrent neural network for computing the Drazin inverse.

Elements of unknown matrix  $V(t)$  are computed using

$$\dot{v}_{ij} = \frac{dv_{ij}}{dt} = \sum_{k=1}^n v_{kj} w_{ik} + \gamma a_{ji}^m, \quad i, j = 1, 2, \dots, n. \quad (2.1.27)$$

Elements of the column  $v_j = \{v_{1j}, \dots, v_{nj}\}$  are generated in the  $j$ th sub-network.

Since the connection weight matrix of the neurons in each sub-network is identical, the proposed recurrent neural network can also be realized by a single sub-network with time-sharing threshold vectors. In each time slot, sub-network biased by the corresponding threshold vector generates one column vector of the Drazin inverse. Therefore, the spatial complexity of the neural network can be reduced by a factor of  $n$ .

The main advantage of the neural network approach to matrix inversion lies in its potential of hardware realization. The state dynamics equation (2.1.16) indicates that the proposed recurrent neural network is convenient for an implementation in an electronic circuit. The proposed

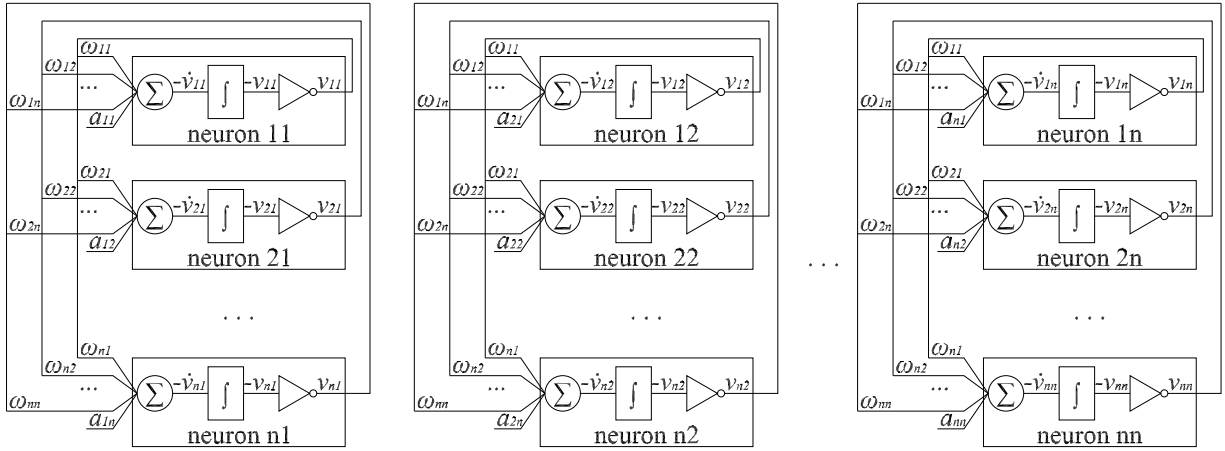


Figure 2.1: Architecture of the RNN for computing the Drazin inverse

electronic neural network consists of  $n^2$  neurons - processing elements. The neurons can be implemented by electronic devices which represent three-operational amplifiers: a summing amplifier, an integrating amplifier, and an inverting amplifier. In [94], Wang gave a detailed description of implementing a mathematical model of recurrent neural network into an electronic neural network. In the same paper, it is also shown that, since the process of finding inverse is parallel and distributed, the convergence rate of the electronic neural network is independent of the order of input matrices. In this way, the electronic neural network has an advantage over traditional sequential procedures for large-scale matrix inversion. Also, as discussed in [94], the convergence rate of the linear recurrent neural network (2.1.16) is directly proportional to the smallest nonzero eigenvalue of the connection weight matrix. Specifically, the time needed for the recurrent neural network to converge is approximately  $5/|\lambda_{\min}(W)|$ , where  $\lambda_{\min}(W)$  is the minimum nonzero eigenvalue of the connection weight matrix  $W = -\gamma A^{m+1}$ . As will be shown through illustrative examples in Section 2.1.3, the recurrent neural network defined by the dynamic equation (2.1.16) is indeed able to generate the Drazin inverse of singular matrices at the projected convergence rate.

**Remark 2.1.1.** *The positive real scaling constant  $\gamma$  should be chosen as large as possible in order to speed up the process of computation. Since  $\gamma$  multiplies the time parameter  $t$ , it is clear that the term*

$$\lim_{t \rightarrow \infty} \exp(-\gamma A^{m+1} t)$$

*from (2.1.13) will vanish faster with larger  $\gamma$ .*

**Remark 2.1.2.** *Since the computation of  $\text{ind}(A)$  requires knowledge of powers  $A^k$ , its computation could be difficult, especially for large-scale matrices. In view of its importance in the computation of the Drazin inverse, we present a speedy/simplified method to obtain integer  $l$  satisfying  $l \geq \text{ind}(A)$ . Our idea is to calculate matrix powers  $A^{2^k}$ ,  $k \geq 1$ , and find first  $l = 2^k$  satisfying  $\text{rank}(A^l) = \text{rank}(A^{2l})$ , where the rank of a matrix is computed using the built-in function `rank` in Matlab. It is clear that  $l$  selected in such a way satisfies  $l \geq \text{ind}(A)$ . Then we use  $M_1^j = [l, +\infty]$  and apply Theorem 2.1.3. More precisely, we compute the matrix power  $A^{2^k}$  using square and multiply method: calculate  $A^2 = A \cdot A$ ,  $A^4 = A^2 \cdot A^2$  etc., until the condition  $\text{rank}(A^{2^k}) = \text{rank}(A^{2^{k+1}})$  is satisfied.*

### 2.1.3 Illustrative examples for GNND model

In order to check the validity and performance of the neural network approach in the computation of the Drazin inverse some computer simulations have been done. In all examples it is assumed that  $A^D$  denotes the exact Drazin inverse of  $A$  and  $A^d$  denotes its approximation derived from our NN approach. In the case  $\text{ind}(A) = 1$ , the exact group inverse of  $A$  is denoted by  $A^\#$  and  $A^g$  denotes its approximation. All the tests are performed on AMD Athlon 3000+ machine, with 1GB RAM.

**Example 2.1.1.** *Let us consider two singular matrices from [14]:*

$$A_1 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

and

$$A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Both matrices have the same index:  $\text{ind}(A_1) = \text{ind}(A_2) = 1$ . It is easy to check that the exact group inverse of matrix  $A_1$  is

$$A_1^\# = \begin{bmatrix} 1 & -2 & -1 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}.$$

The computer simulation was done on the basis of the system of differential equations (2.1.16) and the neural network with the architecture shown in Figure 2.1. Since  $\text{ind}(A_1) = 1$  and  $\sigma(A) = \{1, 1, 0\}$ , we can choose  $m = 1$ , so the weighted matrix  $A_1^{m+1} = A_1^2$  has the same eigenvalues  $\sigma(A_1^2) = \{1, 1, 0\}$ . For the learning rate  $\gamma$  chosen as  $\gamma = 10^7$  the weighted matrix  $-\gamma A^{m+1} = -10^7 \cdot A^2$  has the eigenvalues  $\{-10000000, -10000000, 0\}$ . Because all the

eigenvalues are non-positive, neural network defined by (2.1.16) is stable. The convergence behavior of the network in  $10^{-6}$  seconds is illustrated in Figure 2.2.

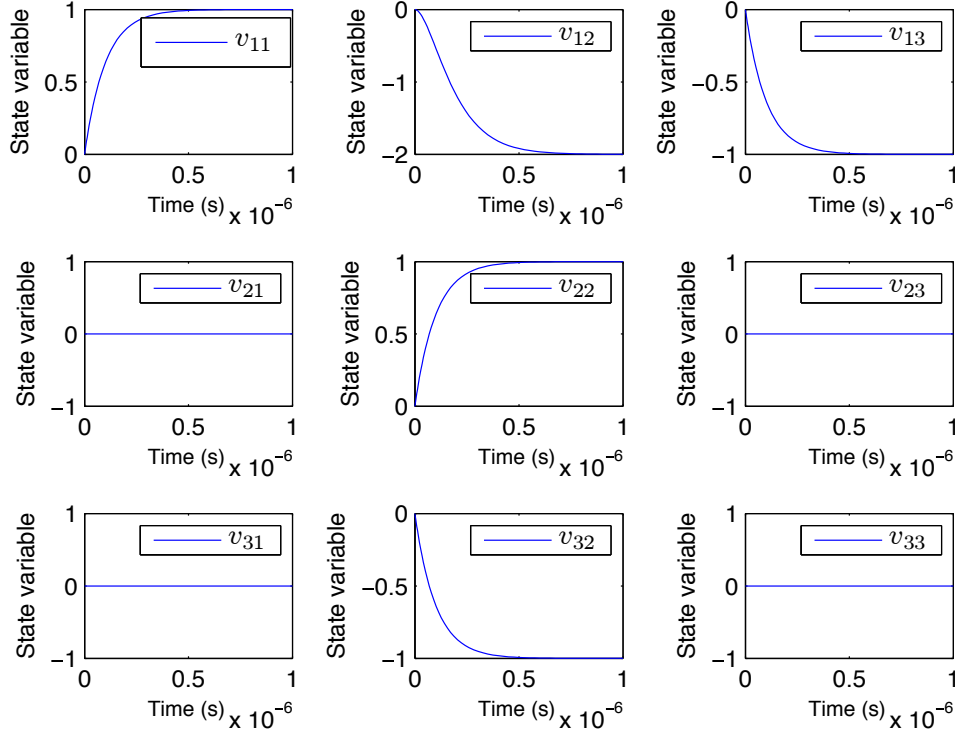


Figure 2.2: Convergence behavior of the RNN in  $10^{-6}$  seconds for Example 2.1.1

The approximation of the exact group inverse  $A_1^\#$  with the accuracy of  $10^{-4}$ , given by

$$A_1^g = \begin{bmatrix} 0.9999 & -2.0001 & -0.9999 \\ 0 & 0.9999 & 0 \\ 0 & -0.9999 & 0 \end{bmatrix},$$

was obtained after  $10^{-6}$  seconds. Figure 2.2 also shows that the states of the neural network are stabilized after approximately  $5/(10000000) = 0.0000005$  seconds.

Similarly, the approximation of the group inverse  $A_2^\#$

$$A_2^g = \begin{bmatrix} 0.9999 & 0 & 0 \\ -1.0002 & 0.9999 & 0 \\ -2.0002 & -0.9999 & 0 \end{bmatrix}$$

is generated after  $10^{-6}$  seconds. In Table 2.1 we arrange numerical results concerning the accuracy of approximations generated by our method and the feed-forward neural network approach (denoted by FF) from [14]. The matrix 2-norm  $\|\cdot\|$  is computed by the Matlab function `norm()`. We can see that more accurate results are provided using the recurrent neural

network approach compared to the feed-forward neural network approach.

Table 2.1: Numerical comparison test with FF[14]

Matrix	$\gamma$	time(s)	$\ A^g - A^G\ $	$\ FF(A) - A^G\ $
$A_1$	$10^7$	$10^{-6}$	$1.099466084 \cdot 10^{-4}$	0.022568085212
$A_2$	$10^7$	$10^{-6}$	$3.397199551 \cdot 10^{-4}$	0.042300935109

**Example 2.1.2.** Consider the matrix from [136]

$$A = \begin{bmatrix} 2 & -1.6 & 5.6 & -5.6 & 0 & 5.6 \\ 0 & 1 & 6 & -6 & 0 & 6 \\ 0 & 0 & 4 & -4 & 0.1 & 3.9 \\ 0 & 0 & 0 & 0 & 0.1 & -0.1 \\ 0 & 0 & 0 & 0 & 0.1 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

with  $\text{ind}(A) = 3$ . The exact Drazin inverse of  $A$  is equal to

$$A^D = \begin{bmatrix} 0.5000 & 0.8000 & -1.9000 & 1.9000 & 0 & -1.9000 \\ 0 & 1.0000 & -1.5001 & 1.5001 & 0 & -1.5001 \\ 0 & 0 & 0.2500 & -0.2500 & 0 & 0.2500 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The eigenvalues of  $A$  are  $\{2, 1, 4, 0, 0.1, 0\}$ . Since the matrix  $A^4$  has all eigenvalues nonnegative, we can select  $k = \text{ind}(A) = 3$ . We choose  $\gamma = 10^7$ , so the connection weight matrix  $W = -10^7 \cdot A^4$  has all non-positive eigenvalues. Trajectories for zero initial conditions are depicted in Figure 2.3 and they show convergence behavior in  $10^{-6}$  seconds.

The matrix produced by our algorithm is

$$A^d = \begin{bmatrix} 0.5000 & 0.8000 & -1.9000 & 1.9000 & 0 & -1.9000 \\ 0 & 1.0000 & -1.5000 & 1.5000 & 0 & -1.5000 \\ 0 & 0 & 0.2500 & -0.2500 & 0 & 0.2500 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$



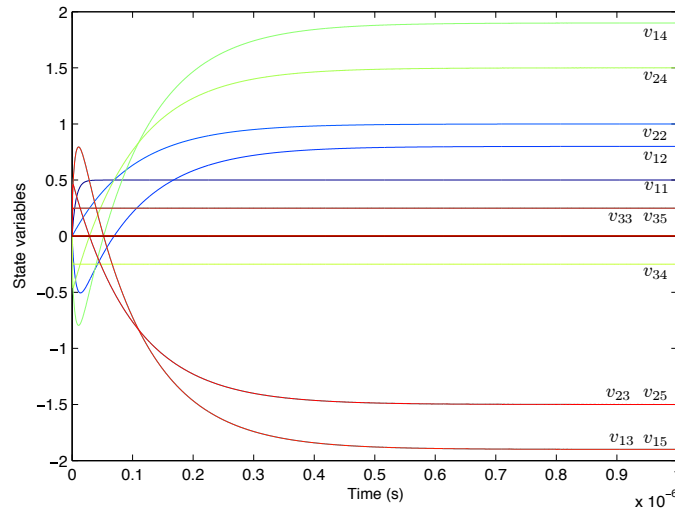


Figure 2.3: Convergence behavior of the RNN in  $10^{-6}$  seconds for Example 2.1.2

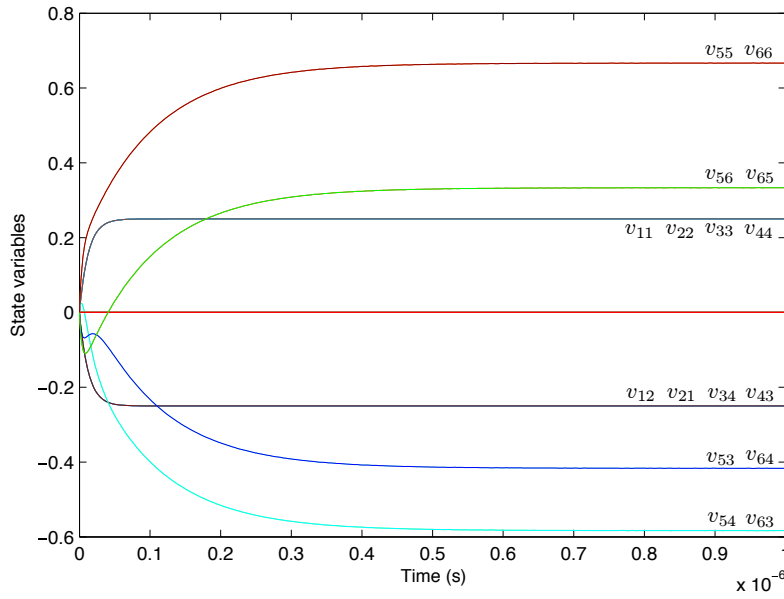
**Example 2.1.3.** Consider the matrix from [115], given by

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & -1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 0 & 2 & -1 \\ -1 & -1 & 0 & -1 & -1 & 2 \end{bmatrix}.$$

The spectrum of  $A$  is equal to  $\sigma(A) = \{3, 2, 2, 1, 0, 0\}$ . Using  $m = \text{ind}(A) = 2$  and  $\gamma = 10^7$ , the method produces the following approximation of the Drazin inverse after  $10^{-7}$  seconds;

$$A^d = \begin{bmatrix} 0.2500 & -0.2500 & 0 & 0 & 0 & 0 \\ -0.2500 & 0.2500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2500 & -0.2500 & 0 & 0 \\ 0 & 0 & -0.2500 & 0.2500 & 0 & 0 \\ 0 & 0 & -0.4166 & -0.5833 & 0.6666 & 0.3333 \\ 0 & 0 & -0.5833 & -0.4166 & 0.3333 & 0.6666 \end{bmatrix}.$$

The convergence behavior of the network in  $10^{-7}$  seconds is graphically illustrated in Figure 2.4.


 Figure 2.4: Convergence behavior of the RNN in  $10^{-6}$  seconds for Example 2.1.3

The exact Drazin inverse of  $A$  is equal to

$$A^D = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & -\frac{1}{4} & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & -\frac{5}{12} & -\frac{7}{12} & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & -\frac{7}{12} & -\frac{5}{12} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

**Example 2.1.4.** The following randomly generated matrix

$$A = \begin{bmatrix} 0.4447 & 0.4057 & 0.3529 & 0.6038 & 0.9318 & 0.6721 \\ 0.6154 & 0.9355 & 0.8132 & 0.2722 & 0.4660 & 0.8381 \\ 0.7919 & 0.9169 & 0.0099 & 0.1988 & 0.4186 & 0.0196 \\ 0.9218 & 0.4103 & 0.1389 & 0.0153 & 0.8462 & 0.6813 \\ 0.7382 & 0.8936 & 0.2028 & 0.7468 & 0.5252 & 0.3795 \\ 0.1763 & 0.0579 & 0.1987 & 0.4451 & 0.2026 & 0.8318 \end{bmatrix}$$

is nonsingular. Therefore, it satisfies  $\text{ind}(A) = 0$  and  $A^{-1} = A^D$ . We want to show that for appropriate choice of  $m$ , our neural network produces a good approximation of the inverse  $A^{-1}$ . The eigenvalues of  $A$  generate the set

$$\{3.0601, 0.6701, -0.6206 + 0.1081i, -0.6206 - 0.1081i, 0.3905, -0.1170\}.$$

We need to find such  $m \geq \text{ind}(A)$  which ensures that all eigenvalues of  $A^{m+1}$  has nonnegative

real parts. Firstly, we have  $M_1^j = [0, +\infty]$ ,  $j = 1, \dots, 6$ . The eigenvalues  $\lambda_1, \lambda_2$  and  $\lambda_5$  are positive, so that  $U_1 = M_1^1 = U_2 = M_1^2 = U_5 = M_1^5 = [0, +\infty]$ . Since we have one eigenvalue ( $\lambda_6$ ) that is negative, this means that our  $m$  corresponding to  $\lambda_6$  need to be even. Therefore,  $U_6 = M_2^6 = \{2, 4, 6, \dots\}$ . Further, we have two complex conjugate eigenvalues ( $\lambda_3, \lambda_4$ ). Since it holds  $\cos \alpha = \cos(-\alpha)$  and  $\cos(\pi - \alpha) = \cos(\pi + \alpha)$ , we have  $\operatorname{Re}(\bar{z}^{m+1}) \geq 0$  for the same  $m$  such  $\operatorname{Re}(z^{m+1}) \geq 0$ . In our case, this means that  $M_4^3 = M_4^4$ . The real part of  $\lambda_3$  is  $a = -0.6206 < 0$ , so this eigenvalue lies in the second quadrant of the complex plane. We have

$$\varphi = \arctan\left(\frac{b}{a}\right) = -9.88100^\circ = 180^\circ - 9.88100^\circ = 170.119^\circ.$$

So,  $\frac{\pi}{\varphi} = 1.05805$ . Therefore, the sets  $U_3 = M_4^3 = U_4 = M_4^4$  are defined as:

$$\frac{4s-1}{2} \frac{\pi}{|\varphi|} < m+1 < \frac{4s+1}{2} \frac{\pi}{|\varphi|}, \quad s \in \mathbb{N}_0. \quad (2.1.28)$$

For  $s = 1$ , we have

$$\frac{3}{2} \cdot 1.05805 < m+1 < \frac{5}{2} \cdot 1.05805,$$

i.e.,  $1.5870 < m+1 < 2.645125$ . So,  $m+1=2$ , and  $m=1$ . For such choice of  $m$ , the matrix  $A^{m+1} = A^2$  has all eigenvalues with positive real parts. Indeed, the eigenvalues of the matrix  $A^2$  are

$$\{9.3643, 0.4490, 0.3735 + 0.1341i, \\ 0.3735 - 0.1341i, 0.1525, 0.0137\}.$$

The neural network gives the following solution after  $10^{-7}$  seconds, for  $\gamma = 10^{10}$ :

$$A^d = \begin{bmatrix} -3.0746 & 0.1593 & -5.2799 & 3.2284 & 5.3347 & -2.6300 \\ 1.7626 & 0.0864 & 5.4113 & -2.5822 & -4.2963 & 2.4363 \\ -1.4431 & 1.4461 & -5.4049 & 1.7459 & 4.1102 & -3.4690 \\ -0.5694 & -0.3461 & -2.4723 & 0.3365 & 3.0550 & -0.8025 \\ 2.2305 & -0.3552 & 1.6884 & -0.6080 & -2.0921 & -0.0317 \\ 0.6351 & -0.1135 & 2.9452 & -0.9536 & -2.9387 & 2.8558 \end{bmatrix}$$

The exact inverse of the matrix  $A$  is

$$A^D = A^{-1} = \begin{bmatrix} -3.0746 & 0.1594 & -5.2799 & 3.2284 & 5.3347 & -2.6300 \\ 1.7626 & 0.0864 & 5.4113 & -2.5822 & -4.2963 & 2.4364 \\ -1.4430 & 1.4461 & -5.4049 & 1.7459 & 4.1102 & -3.4690 \\ -0.5694 & -0.3461 & -2.4723 & 0.3365 & 3.0550 & -0.8024 \\ 2.2305 & -0.3552 & 1.6884 & -0.6080 & -2.0921 & -0.0317 \\ 0.6351 & -0.1135 & 2.9452 & -0.9536 & -2.9387 & 2.8558 \end{bmatrix}.$$

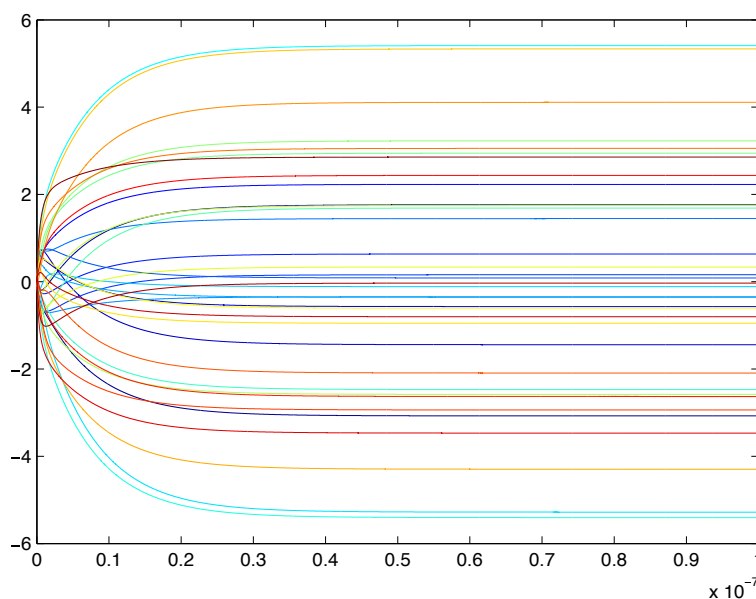


Figure 2.5: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 2.1.4

The convergence behavior of the network in  $10^{-7}$  seconds is graphically illustrated in Figure 2.5.

**Example 2.1.5.** Consider another singular matrix  $A$  from [115], with index  $\text{ind}(A) = 4$ :

$$A = \begin{bmatrix} 3 & -1 & -1 & -1 & 1 & -3 & 1 & 2 \\ 3 & -1 & -1 & -1 & 1 & -3 & 1 & 2 \\ 2 & 0 & -1 & -1 & 1 & -3 & 1 & 2 \\ 1 & 0 & 0 & -1 & 1 & -3 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & -3 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & -2 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here  $\sigma(A) = \{0, 0, 0, 0, 1, -2, -1, 1\}$ . Since  $A$  has negative eigenvalues, we can take  $m = 5$ , because the matrix  $A^{m+1} = A^{5+1} = A^6$  has all nonnegative eigenvalues. Using  $\gamma = 10^7$ , neural network produces the solution

$$A^d = \begin{bmatrix} 0 & 0 & 0 & 0 & 1.0000 & -1.4999 & -0.5000 & 1.9999 \\ 0 & 0 & 0 & 0 & 1.0000 & -1.4999 & -0.5000 & 1.9999 \\ 0 & 0 & 0 & 0 & 1.0000 & -1.4999 & -0.5000 & 1.9999 \\ 0 & 0 & 0 & 0 & 1.0000 & -1.4999 & -0.5000 & 1.9999 \\ 0 & 0 & 0 & 0 & 1.0000 & -1.4999 & -0.5000 & 1.9999 \\ 0 & 0 & 0 & 0 & 0 & -0.5000 & -0.5000 & 1.9999 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix}.$$

The exact Drazin inverse of  $A$  is

$$A^D = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & -\frac{3}{2} & -\frac{1}{2} & 2 \\ 0 & 0 & 0 & 0 & 1 & -\frac{3}{2} & -\frac{1}{2} & 2 \\ 0 & 0 & 0 & 0 & 1 & -\frac{3}{2} & -\frac{1}{2} & 2 \\ 0 & 0 & 0 & 0 & 1 & -\frac{3}{2} & -\frac{1}{2} & 2 \\ 0 & 0 & 0 & 0 & 1 & -\frac{3}{2} & -\frac{1}{2} & 2 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The convergence behavior of the network in  $10^{-6}$  seconds is graphically illustrated in Figure 2.6.

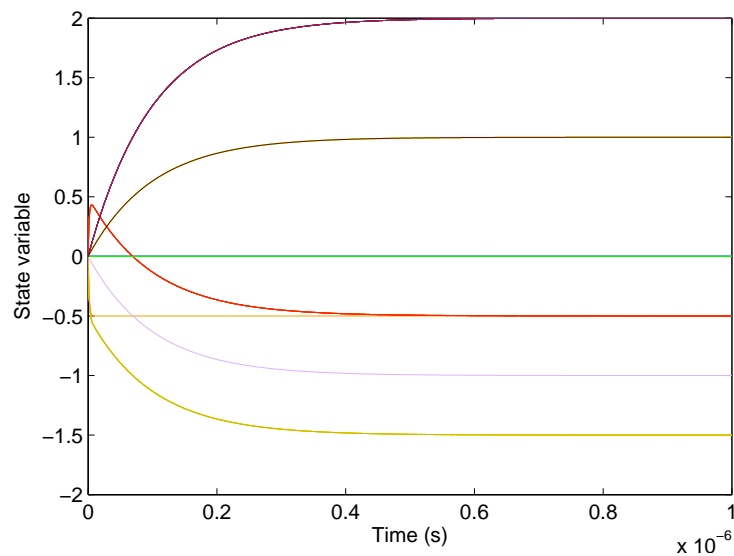


Figure 2.6: Convergence behavior of the RNN in  $10^{-6}$  seconds for Example 2.1.5

Our model is intended for computing the Drazin inverse of singular matrices, but it can also deal with the inverses of regular matrices. More precisely, if the matrix  $A$  is nonsingular, then our method produces  $A^{-1}$ ; but if  $A$  is singular, the method gets  $A^D$ . Compared to previous works, where neural networks for computing the inverse matrix are introduced (for example, [94, 96, 108]), our neural network has the same neural network architecture as discussed in [94], but different neurodynamic.

**Example 2.1.6.** *Several nonsingular matrices with different orders are randomly created in this example. We compare the performance of our neural network against the performance of the neural network for calculating the matrix inverse, presented in [94].*

*The output of our neural network is denoted by  $A^d$ ; by  $G_N(A)$  we denote the output of neural network presented in [94], while  $A^{-1}$  denotes the matrix inverse calculated by the Matlab function `inv()`. Finally, the matrix norm  $\|\cdot\|$  is computed by the Matlab function `norm()`. The test shows that our neural network computes the inverse of a given matrix with very good precision, similar to the precision achieved by the network presented in [94].*

Table 2.2: Numerical comparison test with [94]

Matrix	$n$	$\gamma$	time(s)	$\ A^d - A^{-1}\ $	$\ G_N(A) - A^{-1}\ $
$A_1$	5	$10^9$	$10^{-7}$	$1.34 \cdot 10^{-4}$	$1.06 \cdot 10^{-4}$
$A_2$	10	$10^9$	$10^{-7}$	$4.97 \cdot 10^{-5}$	$1.47 \cdot 10^{-5}$
$A_3$	15	$10^9$	$10^{-7}$	$2.97 \cdot 10^{-5}$	$9.39 \cdot 10^{-6}$
$A_4$	20	$10^9$	$10^{-7}$	$7.88 \cdot 10^{-5}$	$1.93 \cdot 10^{-6}$
$A_5$	25	$10^9$	$10^{-7}$	$1.79 \cdot 10^{-4}$	$2.30 \cdot 10^{-4}$

**Example 2.1.7.** *In this example we compare the performance of our method against the performance of an iterative method for computing the Drazin inverse proposed in [136], which we denote by  $IM$ . The Matrix  $A$  takes the values  $A_1, \dots, A_5$  reused from examples 1-5. The comparison in Table 2.3 is given in the sense of the accuracy of the solution and the CPU time required for the convergence. Previous testing has shown that both methods produce solutions with the similar accuracy. So in this table, we fixed the accuracy ( $\sim 10^{-4}$ ) and we measure the CPU time required for each method to generate the solution with that accuracy.*

Table 2.3: Numerical comparison test with [136]

Matrix	$\ A^D - A^d\ $		time(s)	
	$IM$	$NN$	$IM$	$NN$
$A_1$	$\sim 10^{-4}$	$\sim 10^{-4}$	0.029391	$10^{-6}$
$A_2$	$\sim 10^{-4}$	$\sim 10^{-4}$	0.050967	$10^{-6}$
$A_3$	$\sim 10^{-4}$	$\sim 10^{-4}$	0.006505	$10^{-6}$
$A_4$	$\sim 10^{-4}$	$\sim 10^{-4}$	0.232515	$10^{-7}$
$A_5$	$\sim 10^{-4}$	$\sim 10^{-4}$	0.541346	$10^{-6}$

Results from Table 2.3 show that RNN approach gives results with similar accuracy but about three orders of magnitude faster than [136].

### 2.1.4 Application of the GNND model

The Drazin inverse and the group inverse have been applied to various fields, for instance, singular linear systems, finite Markov chains, singular differential and difference equations, multibody system dynamics (see, [7, 22, 57, 71]), cryptography [41], etc. In the monograph [7, Page 123] it is shown that the Drazin inverse solution  $A^D b$  solves the singular linear system  $Ax = b$  if and only if  $b \in \mathcal{R}(A^k)$ . Also,  $A^D b$  is the unique solution of  $Ax = b$  provided that  $x \in \mathcal{R}(A^k)$  [7, Page 123]. It is also known result that the Drazin inverse solution represents the minimal  $P$ -norm solution of the linear system  $Ax = b$ , where  $P$  is invertible matrix such that  $P^{-1}AP$  is the Jordan canonical form of  $A$  and  $\|x\|_P = \|P^{-1}x\|_2$  [107]. Moreover, the group inverse is of fundamental importance in the analysis of Google's PageRank system [39, 40]. The restricted matrix equation

$$\begin{aligned} AXB = D, \mathcal{R}(X) \subset \mathcal{R}(A^k), \mathcal{N}(X) \supset \mathcal{N}(A^k), \\ k = \max\{\text{ind}(A), \text{ind}(B)\} \end{aligned}$$

has the unique solution  $X = A^D D B^D$  [90].

The Drazin inverse is applicable in the study of linear systems of differential equations with singular coefficients, which can occur in electrical circuits if, for example, there are dependent sources [7].

**Example 2.1.8.** *In this example we consider solutions of the linear system  $Ax = b$ . The matrix  $A$  takes, in succession, the values  $A_1, A_2, A_3, A_4$  reused from [57] and the vector  $b$  takes two values  $b_{Con} \in \mathcal{R}(A^k)$  and  $b_{Gen} \notin \mathcal{R}(A^k)$ , both defined in [57]. The results contained in Table 2.4 (resp. Table 2.5) are corresponding to the case  $b = b_{Con}$  (resp.  $b = b_{Gen}$ ). Using  $\gamma = 10^8$ , after  $10^{-7}$  seconds, RNN approach generates the result  $x = A^d b$  according to numerical data arranged in Table 2.4 and Table 2.5. By MMSW are denoted accuracy and the CPU time obtained applying the iterations from [57]; corresponding results generated by our RNN approach are denoted by NN.*

Table 2.4: Comparison test with [57] in the case  $b = b_{Con}$

Matrix	$\ A^D b - x\ $		time(s)	
	MMSW	NN	MMSW	NN
$A_1$	0.00163624	0.000880501	1.41	0.000001
$A_2$	0.00354824	0.015209950	3.47	0.000001
$A_3$	1.06348014	0.013472465	9.41	0.000001
$A_4$	0.000000197	0.002730538	0.8	0.000001

Table 2.5: Comparison test with [57] in the case  $b = b_{Gen}$

Matrix	$\ A^D b - x\ $		time(s)	
	MMSW	NN	MMSW	NN
$A_1$	0.001108242	0.0016349695464	0.73	0.000001
$A_2$	0.002457201	0.0000760248121	0.7	0.000001
$A_3$	0.486144760	0.0004332707662	6.72	0.000001
$A_4$	0.000000432	0.0026274614443	0.44	0.000001

Comparing data arranged in both Table 2.4 and Table 2.5 it can be easily observed that RNN approach produces more accurate results in significantly smaller CPU time.

**Example 2.1.9.** The group inverse plays a central role in the theory of finite Markov chains [7, 56]. Let  $T$  be the one-step transition matrix of a finite homogeneous Markov chain and  $A = I - T$ . According to known result [7, Theorem 8.2.1] the condition  $\text{ind}(A) = 1$  holds, i.e., the existence of  $A^G$  is ensured for any stochastic matrix  $T$ . It is well known fact that the answer to every important question concerning the chain can be obtained from  $A^G$  [56].

Let us consider the regular chain whose transition matrix is given by

$$T = \frac{1}{4} \begin{bmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 2 & 0 \\ 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

For the description of the Markov chain it is necessary to compute the group inverse of

$$A = I - T = \frac{1}{4} \begin{bmatrix} 4 & -2 & -2 & 0 \\ -2 & 4 & -2 & 0 \\ -2 & -1 & 4 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}.$$



The exact group inverse of  $A$  is known from [56]:

$$A^\# = \frac{2}{1083} \begin{bmatrix} 265 & -61 & -96 & -108 \\ -96 & 300 & -96 & -108 \\ -115 & -137 & 246 & 6 \\ -210 & -156 & -210 & 576 \end{bmatrix}.$$

Setting  $\gamma = 10^9$ , we obtain the resulting group inverse  $A^g$  after  $10^{-7}$  seconds. The convergence behavior is presented in Figure 2.7.

$$A^g = \begin{bmatrix} 0.4893 & -0.1126 & -0.1773 & -0.1994 \\ -0.1773 & 0.5540 & -0.1773 & -0.1994 \\ -0.2123 & -0.2530 & 0.4543 & 0.0111 \\ -0.3878 & -0.2881 & -0.3878 & 1.0637 \end{bmatrix}.$$

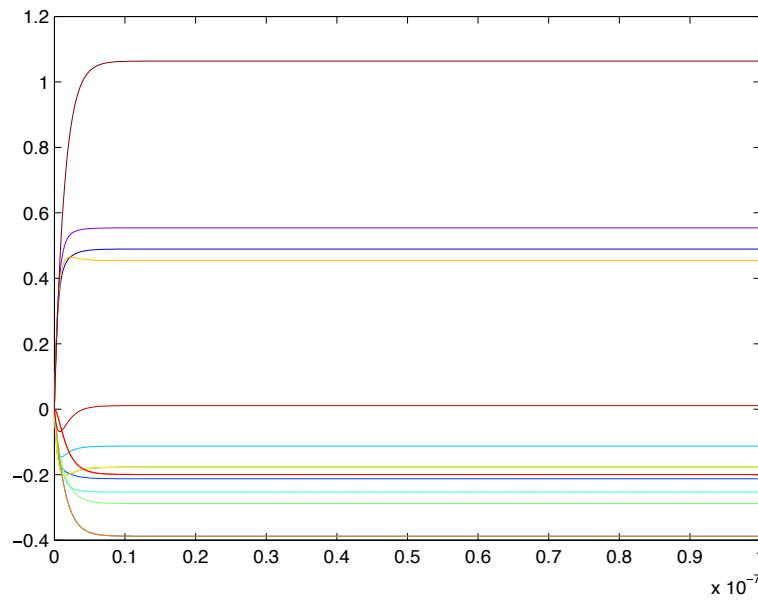


Figure 2.7: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 2.1.9

## 2.2 GNN models for computing outer inverse

Next, we present the results from [138]. We are expanding investigation applied to the computation of the Drazin inverse ([84]), based on appropriate dynamic equation and induced gradient based RNN, to the more generalized problem of computing the outer inverses. Similarly to the RNNs developed for the usual matrix inversion as well as to the RNNs for computing the matrix pseudoinverse and the Drazin inverse, developed earlier, the proposed neural network are also composed of a number of independent sub-networks which can operate concurrently. Due to its simplicity in the implementation a linear activation function is chosen. The stability of the proposed recurrent neural networks as well as their performance is demonstrated by means of a few numerical examples.

### 2.2.1 Preliminaries and motivation

Recall that the generalized inverse  $A_{T,S}^{(2)}$  of  $A \in \mathbb{C}^{m \times n}$  is the matrix  $X \in \mathbb{C}^{n \times m}$  which satisfies

$$XAX = X, \quad \mathcal{R}(X) = T, \quad \mathcal{N}(X) = S. \quad (2.2.1)$$

If  $A \in \mathbb{C}_r^{m \times n}$ ,  $T$  is a subspace of  $\mathbb{C}^n$  of dimension  $t \leq r$  and  $S$  is a subspace of  $\mathbb{C}^m$  of dimension  $m - t$ , then  $A$  has a  $\{2\}$ -inverse  $X$  such that  $\mathcal{R}(X) = T$  and  $\mathcal{N}(X) = S$  if and only if  $AT \oplus S = \mathbb{C}^m$ , in which case  $X$  is unique and it is denoted by  $A_{T,S}^{(2)}$  (see [89]).

Our aim is to develop a general RNN approach for computing the outer inverse with prescribed range and null space, continuing dynamic state equations and gradient RNN approach exploited in [84, 94, 96, 108].

It is reasonable to expect that the starting point should be the integral representation of outer inverses. In [109, 111], Wei introduced the integral representation of the generalized inverse  $A_{T,S}^{(2)}$  of a complex matrix  $A \in \mathbb{C}^{m \times n}$ , where  $T$  and  $S$  are subspaces of  $\mathbb{C}^n$  and  $\mathbb{C}^m$ , respectively. For the sake of simplicity,  $\operatorname{Re}(\sigma(GA)) \geq 0$  stands for  $\operatorname{Re} \lambda \geq 0, \forall \lambda \in \sigma(GA)$ . Under the assumptions  $\mathcal{R}(G) = T, \mathcal{N}(G) = S$ , in the case when the spectrum of  $GA$  satisfies  $\operatorname{Re}(\sigma(GA)) \geq 0$ , the following integral representation of  $A_{T,S}^{(2)}$  is valid:

$$A_{T,S}^{(2)} = \int_0^\infty \exp(-GA\tau)G \, d\tau. \quad (2.2.2)$$

In the dual case,  $\operatorname{Re}(\sigma(AG)) \geq 0$  guarantees the existence of the following integral representation:

$$A_{T,S}^{(2)} = \int_0^\infty G \exp(-AG\tau) \, d\tau. \quad (2.2.3)$$

We define an appropriate dynamic equation and the corresponding RNN, both initiated by (2.2.2) or (2.2.3). The RNN is capable of generating outer inverses with prescribed range  $\mathcal{R}(G)$

and null space  $\mathcal{N}(G)$  under the assumption of zero initial states. The particular case  $G = A^m$ , where  $m \geq \text{ind}(A)$  is appropriately defined nonnegative integer, includes all the results we presented in section 2.1.

## 2.2.2 Neural network architecture

Our general observation is that the problem of computing various generalized inverses can be resolved as a solution of appropriate dynamic state equation of a generic recurrent neural network. Furthermore, the dynamic state equation is stated by virtue of appropriately defined constraint satisfaction problem. In order to define the constraint satisfaction problem which initiates the dynamic state equation underlying in the corresponding recurrent neural network, it is necessary to introduce Lemma 2.2.1 and several observations.

**Lemma 2.2.1.** *Let  $A \in \mathbb{C}_r^{m \times n}$  be given and  $G \in \mathbb{C}_s^{n \times m}$  be arbitrarily chosen matrix satisfying  $0 < s \leq r$ . Assume that  $X := A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  exists. Then the matrix equations*

$$GAX = G, \quad XAG = G \quad (2.2.4)$$

are satisfied.

*Proof.* The following known facts are used to verify the statements (cf. [89, 120]):

$$AX = P_{A\mathcal{R}(G), \mathcal{N}(G)}, \quad XA = P_{\mathcal{R}(G), (A^*\mathcal{N}(G)^\perp)}.$$

Now, the first and the second statement in (2.2.4) follows, respectively, from the next known results (see, for example [2]):  $P_{L,M}G = G$  if and only if  $\mathcal{R}(G) \subseteq L$  and  $GP_{L,M} = G$  if and only if  $\mathcal{N}(G) \supseteq M$ .  $\square$

**Remark 2.2.1.** *The constraint satisfaction problem (2.2.4) defined in Lemma 2.2.1 for the usual matrix inversion as well as for the Moore-Penrose inverse is  $AX = I$  [94, 96]; for the Drazin inverse it is defined by  $A^{m+1}X = A^mAX = A^m$ ,  $m \geq \text{ind}(A)$  [84]. This certainly means that (2.2.4) in the case  $G = I$  leads to the results corresponding to the usual inverse and the Moore-Penrose inverse; similarly, the case  $G = A^m$ ,  $m \geq \text{ind}(A)$  produces the results corresponding to the Drazin inverse. Finally, the choice  $G = A^k (A^{2k+1})^\top A^k$ ,  $k = \text{ind}(A)$  leads to the RNN defined in [85].*

The number of floating point operations in the case  $m \geq n$  can be reduced by applying the first matrix equation in (2.2.4) in defining the dynamic equation. The first equation in (2.2.4) can be rewritten as

$$GAV_G(t) - G = 0, \quad (2.2.5)$$

where  $V_G(t) \in \mathbb{R}^{n \times m}$  denotes the unknown matrix to be solved and which corresponds to the outer inverse with prescribed range and null space  $X := A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$ . Our intention is to solve (2.2.5) for the unknown matrix  $V_G$  using dynamic-system approach. Therefore, we define a scalar-valued norm based error function

$$E(t) = \frac{\|GAV_G(t) - G\|_F^2}{2}.$$

The minimal value  $E(\bar{t}) = 0$  of the residual-error function  $E(t)$  is achieved in a minimum point  $\bar{V}_G = V_G(\bar{t})$  if and only if  $V_G(\bar{t})$  is the exact solution of (2.2.5). Therefore, a computational scheme for computing the minimum point  $\bar{V}_G$  could be defined along the gradient descent direction of  $E(t)$ . The derivative of  $E(t)$  with respect to  $V_G \in \mathbb{R}^{n \times m}$  could be derived by applying the principles of the matrix calculus described in [25, Chapter 5]:

$$\frac{\partial E(t)}{\partial V_G} = (GA)^T (GAV_G(t) - G). \quad (2.2.6)$$

Therefore, the GNN model for computing outer inverses  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  possesses the form

$$\frac{dV_G(t)}{dt} = \dot{V}_G(t) = -\gamma (GA)^T (GAV_G(t) - G), \quad V_G(0) = V_0, \quad m \geq n, \quad (2.2.7)$$

The GNN design (2.2.7) will be termed as GNNATS2 model.

Clearly, the term  $(GA)^T$  in the right-hand side of (2.2.6) is constant matrix. Therefore, following the same reason as in [84], it can be omitted without loss of generality. In this way, the dynamic equation of the initiated recurrent neural network can be given in the form

$$\frac{dV_G(t)}{dt} = \dot{V}_G(t) = -\gamma (GAV_G(t) - G), \quad V_G(0) = V_0, \quad m \geq n, \quad (2.2.8)$$

where  $V_G(t)$  is a matrix of activation state variables and  $\gamma$  is a positive scaling constant. The GNN design (2.2.8) was originated in [138] and will be termed as GNNGA model.

The equilibrium state  $\bar{V}_G = V_G(\bar{t})$  of the dynamic system (2.2.8), satisfying

$$\frac{d\bar{V}_G}{dt} = 0 \quad (2.2.9)$$

is also the minimum of the residual-norm function  $E(t)$ . Namely, (2.2.9) implies

$$-\gamma (G\bar{A}\bar{V}_G - G) = 0, \quad (2.2.10)$$

which means that  $\bar{V}_G$  satisfies (2.2.5).

**Remark 2.2.2.** *The second matrix equation in (2.2.4) is more efficient for the dual case,  $m < n$ .*

In accordance to that, it is completely justified to define following dynamic system, whose equilibrium state  $\overline{V}_G$  satisfies the second equation in (2.2.4):

$$\frac{dV_G(t)}{dt} = -\gamma (V_G(t)AG - G), \quad V_G(0) = V_0, \quad m < n. \quad (2.2.11)$$

The GNN design (2.2.11) will be termed as GNNAG model.

The recurrent neural network defined above is a linear dynamic system in matrix form. According to the linear systems theory [35], the closed-form solution of the state matrix was given in [138]:

$$V_G(t) = \begin{cases} \exp(-\gamma GA t) V_G(0) + \gamma \exp(-\gamma GA t) \int_0^t \exp(\gamma GA \tau) G d\tau, & m \geq n, \\ V_G(0) \exp(-\gamma AG t) + \gamma G \exp(-\gamma AG t) \int_0^t \exp(\gamma AG \tau) d\tau, & m < n. \end{cases} \quad (2.2.12)$$

To analyze the convergence and stability of a neural network, it is of major interest to know the eigenvalues of the matrix  $GA$  (or  $AG$  for the dual case). Using the principles from [84], it can be easily shown that the term  $\lim_{t \rightarrow \infty} \exp(-\gamma GA t)$  vanishes if the matrix  $GA$  has nonnegative eigenvalues. That fact and equation (2.2.12) imply the following representation for  $\lim_{t \rightarrow \infty} V_G(t) = \overline{V}_G$ :

$$\overline{V}_G = \begin{cases} \lim_{t \rightarrow \infty} \gamma \exp(-\gamma GA t) \int_0^t \exp(\gamma GA \tau) G d\tau, & m \geq n, \\ \lim_{t \rightarrow \infty} \gamma G \exp(-\gamma AG t) \int_0^t \exp(\gamma AG \tau) d\tau, & m < n, \end{cases} \quad (2.2.13)$$

It can be verified, based on the definition and properties of matrix exponential, that the closed-form solution of  $\overline{V}_G$  in (2.2.13) satisfies (2.2.1) and it is independent of  $\gamma$ . This means that (2.2.13) and (2.2.2) produce the same result, equal to  $A_{T,S}^{(2)}$ . The proof is presented in Theorem 2.2.1 from [138].

**Theorem 2.2.1.** [138] *Let  $A \in \mathbb{R}^{m \times n}$  be given matrix,  $G \in \mathbb{R}_s^{n \times m}$  be arbitrary matrix satisfying  $0 < s \leq r$ , and  $\sigma(GA) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  be the spectrum of  $GA$ . Suppose that the condition*

$$\operatorname{Re}(\lambda_j) \geq 0, \quad j = 1, 2, \dots, n \quad (2.2.14)$$

*is satisfied. Then the limiting expression (2.2.13) produces the outer inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$ , i.e.,*

$$\overline{V}_G = A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}. \quad (2.2.15)$$

*Proof.* Applying several elementary transformations and basic properties (2.2.4) of the outer inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$ , we conclude

$$\begin{aligned}
 \overline{V}_G &= \lim_{t \rightarrow \infty} \gamma \exp(-\gamma G A t) \int_0^t \exp(\gamma G A \tau) G \, d\tau \\
 &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma G A t) \int_0^t \exp(\gamma G A \tau) (\gamma G A) \, d\tau \right] A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} \\
 &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma G A t) \int_0^t d(\exp(\gamma G A \tau)) \right] A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} \\
 &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma G A t) \exp(\gamma G A \tau) \Big|_{\tau=0}^{\tau=t} \right] A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} \\
 &= \left\{ \lim_{t \rightarrow \infty} \exp(-\gamma G A t) [\exp(\gamma G A t) - I] \right\} A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} \\
 &= \left[ I - \lim_{t \rightarrow \infty} \exp(-\gamma G A t) \right] A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}.
 \end{aligned}$$

Using the principles from [84] one can verify that the term  $\lim_{t \rightarrow \infty} \exp(-\gamma G A t)$  vanishes. Therefore, (2.2.15) is verified.  $\square$

**Remark 2.2.3.** Analogous statement can be verified when the outer inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  is generated using dual equation  $XAG = G$  in (2.2.4). This situation is more appropriate in the case  $m < n$ , due to the reduced complexity required for matrix multiplications.

In light of the above discussion, the dynamic state equation of the RNN for computing outer inverse can be described as follows:

$$\begin{cases} \frac{dV_G(t)}{dt} = -\gamma (GAV_G(t) - G), & V(0) = 0, \text{ if } m \geq n, \\ \frac{dV_G(t)}{dt} = -\gamma (V_G(t)AG - G), & V(0) = 0, \text{ if } m < n \end{cases} \quad (2.2.16)$$

Now we investigate stability of the equilibrium state  $\overline{V}_G$ . The main result shows that the equilibrium state is stable.

**Theorem 2.2.2.** [138] Let  $A \in \mathbb{R}^{m \times n}$  be given matrix,  $G \in \mathbb{R}_s^{n \times m}$  be arbitrary matrix satisfying  $0 < s \leq r$  and  $\sigma(GA) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  be the spectrum of  $GA$ . Suppose that (2.2.14) holds. Then the gradient-based neural network (2.2.16) is stable in the sense of Lyapunov.

*Proof.* It is only worth to mention that the Lyapunov candidate function is defined by

$$E(t) = \frac{\|GAX - G\|_F^2}{2} = \frac{\text{Tr} \left( (GAX - G)^T (GAX - G) \right)}{2}. \quad (2.2.17)$$

Then the rest of the proof is analogous to the proof of Theorem 2 from [84].  $\square$

According to the Theorem 2.2.1, the matrix  $G$  must be chosen such that the matrices  $GA$  (in the case  $m \geq n$ ) or  $AG$  (in the case  $m < n$ ), have nonnegative real parts of all their eigenvalues, i.e.

$$\sigma(GA) \subset \{z : \operatorname{Re}(z) \geq 0\}, \quad m \geq n, \quad (2.2.18)$$

$$\sigma(AG) \subset \{z : \operatorname{Re}(z) \geq 0\}, \quad m < n. \quad (2.2.19)$$

The positive real scaling constant  $\gamma$  should be chosen as large as possible in order to speedup the process of computation. Since  $\gamma$  multiplies the time parameter  $t$ , it is clear that vanishing of the term  $\lim_{t \rightarrow \infty} \exp(-\gamma GA t)$  from (2.2.12) is faster with for larger  $\gamma$ .

The neural network used in our implementation is composed from a number of independent sub-networks, in the similar way as it has already been discussed in [94, 96]. Specifically, the number of sub-networks is  $m$  if  $m \geq n$  or it is equal to  $n$  if  $m < n$ . The connection weight matrix  $W$  of the neurons is identical in each sub-network and defined as

$$W = \begin{cases} -\gamma GA, & m \geq n \\ -\gamma AG, & m < n. \end{cases}$$

Note that the size of the connection weight matrix is  $\min\{m, n\} \times \min\{m, n\}$  which usually has fewer connections than being defined otherwise. The threshold (input) matrix of the neuron array is  $-\gamma G$ . Figures 2.8 and 2.9 present the architecture of the recurrent neural network for computing outer inverses in both cases.

Consider the case  $m \geq n$ . Let us denote by  $v_j(t)$  (resp.  $g_j$ ) the  $j$ th column vector of  $V_G(t)$  (resp.  $G$ ), for  $j = 1, 2, \dots, m$ . The dynamics of the  $j$ th sub-network can be expressed in the general form which is first time presented in [94]:

$$\frac{dv_j(t)}{dt} = -\gamma (GA v_j(t) - g_j). \quad (2.2.20)$$

Each sub-network exploits the same connection weight matrix  $W = -\gamma GA$  and

$$\gamma g_j = \{\gamma g_{1j}, \gamma g_{2j}, \dots, \gamma g_{nj}\}$$

is the biasing threshold vector for the  $j$ th sub-network.

Elements  $v_{ij}$  of unknown matrix  $V_G(t)$  are computed using

$$\dot{v}_{ij} = \frac{dv_{ij}}{dt} = \sum_{k=1}^n w_{ik} v_{kj} + \gamma g_{ji}, \quad i = 1, 2, \dots, n; j = 1, 2, \dots, m, \quad (2.2.21)$$

where  $w_{ij}$  are elements of  $W$ . It is important to mention that the column vector

$$v_j = \{v_{1j}, v_{2j}, \dots, v_{nj}\}$$

is output in the  $j$ th sub-network.

Because the connection weight matrix of the neurons in each sub-network is identical, the proposed recurrent neural network can also be realized by a single sub-network with time-sharing threshold vectors. In each time slot, sub-network biased by the corresponding threshold vector generates one column vector of the outer inverse. Each neuron can be implemented by cascading a summer, an integrator, and an inverter, in the same way as was done in [94].

Analogous equations can be derived in the case  $m < n$ .

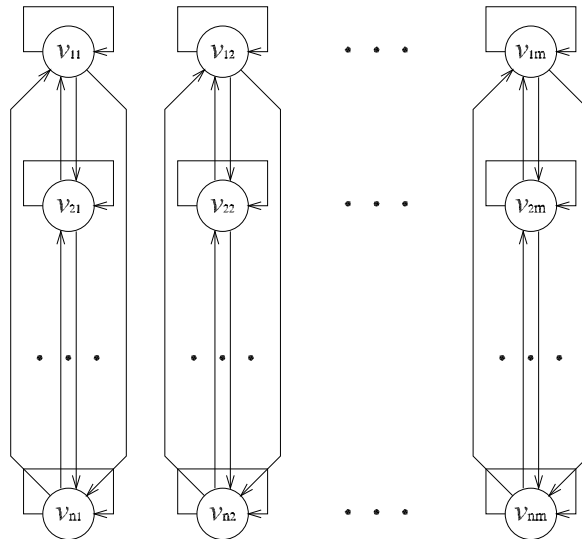


Figure 2.8: Architecture of the RNN for computing outer inverse in case  $m \geq n$

### 2.2.3 Particular cases of GNNGA and GNNAG model

According to (1.2.15), the following particular choices of the matrix  $G$  could be pointed out.

2.a) In the particular case  $G = A^T$  of of GNNGA model we immediately derive known results concerning the usual inverse [94] when  $A$  is nonsingular, as well as the Moore–Penrose inverse when matrix  $A$  is rectangular or rank-deficient [96].

2.b) The choice  $G = A^\sharp = N^{-1}A^*M$  produces the results corresponding to the weighted Moore–Penrose inverse  $A_{M,N}^\dagger$  [108].

The following particular choices of the matrix  $G$  in GNNGA could be pointed out according to



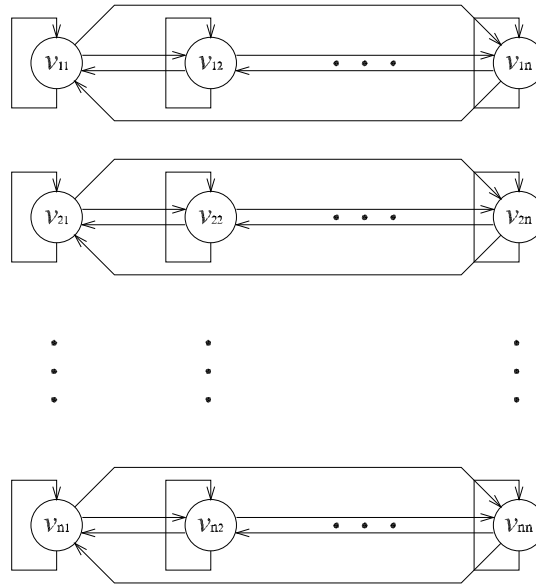


Figure 2.9: Architecture of the RNN for computing outer inverse in case  $m < n$

(1.2.16).

3.a) In the case  $G = A^m$ ,  $m \geq \text{ind}(A)$  we get corresponding RNN approach in computation of the Drazin inverse. But, in this case we have additional possibilities to generate the Drazin inverse using appropriate choice of the exponent  $m$ . Two different approaches in resolving this problem are described in [85, 84].

3.b) For a square matrix  $A$  of index  $\text{ind}(A) = 1$ , the limiting expression (4.3.9) produces the group inverse  $A^\#$  of  $A$  in the case when  $G$  is equal to a square matrix  $A$  of index  $\text{ind}(A) = 1$ .

3.c) The choice  $\text{rank}(G) = r = \text{rank}(A)$  implies  $\overline{V_G} \in A\{1, 2\}$ .

According to Proposition 1.2.11, it is possible to use defined RNN approach to compute the outer inverse  $A_{T,S}^{(2)}$  and subsequently the vector  $x = A_{T,S}^{(2)}b$ , which is a solution of the linear system  $Ax = b$  under certain conditions presented in Proposition 1.2.11.

## 2.2.4 Illustrative examples for GNNGA model

**Example 2.2.1.** Consider the matrix  $A$  equal to

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 4 & 6 & 2 \\ 2 & 3 & 4 & 5 & 3 \\ 3 & 4 & 5 & 6 & 4 \\ 4 & 5 & 6 & 7 & 6 \\ 6 & 6 & 7 & 7 & 8 \end{bmatrix}, \quad (2.2.22)$$

choose the following matrices  $P \in \mathbb{R}^{5 \times 2}$  and  $Q \in \mathbb{R}^{2 \times 6}$

$$P = \begin{bmatrix} 0 & 0 \\ 2 & 1 \\ 3 & 2 \\ 5 & 3 \\ 1 & 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (2.2.23)$$

and compute  $G = PQ$ . The spectrum of  $GA$  is

$$\sigma(GA) = \{266.346716180717, 0.653283819282910, 0, 0, 0\}.$$

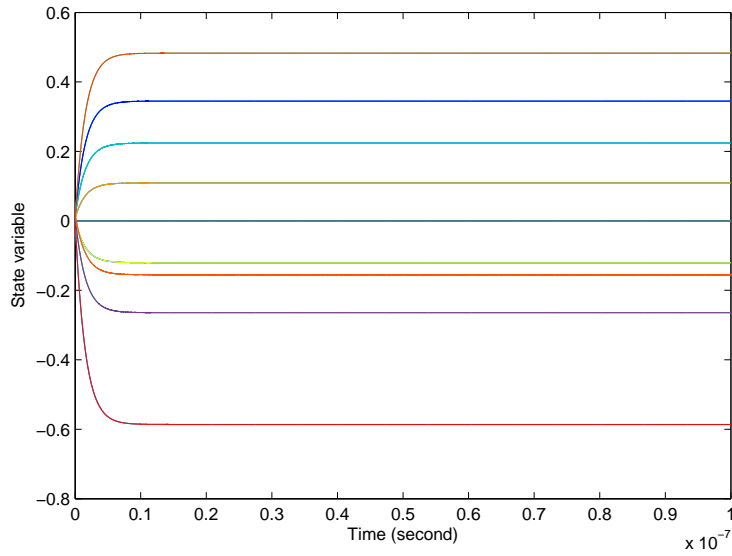
The following approximation of  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$  can be obtained using  $G = PQ$  and  $\gamma = 10^8$  in the RNN (2.2.16):

$$X = \begin{bmatrix} 0 & 0. & 0 & 0 & 0 & 0 \\ -0.1207 & 0.1092 & -0.1207 & 0.1092 & -0.1207 & 0.1092 \\ 0.3448 & 0.2644 & 0.3448 & -0.2644 & 0.3448 & -0.2644 \\ 0.2241 & -0.1552 & 0.2241 & -0.1552 & 0.2241 & -0.1552 \\ -0.5862 & 0.4828 & -0.5862 & 0.4828 & -0.5862 & 0.4828 \end{bmatrix}.$$

Corresponding exact  $\{2\}$ -inverse of  $A$  corresponding to  $G$  is defined by  $P(QAP)^{-1}Q$  (see [12]) and it is equal to

$$A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)} = \frac{1}{174} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -21 & 19 & -21 & 19 & -21 & 19 \\ 60 & -46 & 60 & -46 & 60 & -46 \\ 39 & -27 & 39 & -27 & 39 & -27 \\ -102 & 84 & -102 & 84 & -102 & 84 \end{bmatrix}.$$

Convergence behavior of the network in time  $10^{-7}$  seconds is graphically illustrated in Figure 2.10.


 Figure 2.10: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 2.2.1

**Example 2.2.2.** *The goal of this example is to compute the Drazin inverse of the matrix*

$$A = \begin{bmatrix} 2 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -2 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4 & 2 \end{bmatrix}$$

satisfying  $\text{ind}(A) = 3$ . Therefore, our first attempt is to use  $G = A^3$ . Since the eigenvalues of  $GA$  are equal to

$$\begin{aligned} \sigma(GA) = \sigma(A^4) = \{ & 16.0000 + 0.0000i, 16.0000 + 0.0000i, 4.3866, 0.7168 + 2.4576i, \\ & 0.7168 - 2.4576i, 0.0934 + 0.0000i, 0, 0, 0, 0, 0.7168 + 2.4576i, \\ & 0.7168 - 2.4576i\}, \end{aligned}$$

whose real parts are greater than zero, the choice  $G = A^3$  is appropriate. Using  $\gamma = 10^{10}$ , we get the following approximation of the Drazin inverse with the precision  $10^{-9}$  after  $10^{-7}$

seconds:

$A^d =$

$$\begin{bmatrix} 0.25 & -0.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 1.25 & 1.25 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ -1.6640 & -0.9922 & 0.25 & -0.25 & 0. & 0. & 0. & 0. & -0.0625 & -0.0625 & 0. & 0.1563 \\ -1.1953 & -0.6797 & -0.25 & 0.25 & 0. & 0. & 0. & 0. & -0.0625 & 0.1875 & 0.6875 & 1.3437 \\ -2.7637 & -1.0449 & -1.875 & -1.25 & -1.25 & 1.25 & 1.25 & 1.25 & 1.4844 & 2.5781 & 3.3203 & 6.6406 \\ -2.7637 & -1.0449 & -1.875 & -1.25 & -1.25 & 1.25 & 1.25 & 1.25 & 1.4844 & 2.5781 & 4.5703 & 8.5156 \\ 14.1094 & 6.3008 & 6.625 & 3.375 & 5. & -3. & -5. & -5. & -4.1875 & -8.5 & -10.5078 & -22.4609 \\ -19.3242 & -8.5078 & -9.75 & -5.25 & -7.5 & 4.5 & 7.5 & 7.5 & 6.375 & 12.5625 & 15.9766 & 33.7891 \\ -0.625 & -0.3125 & 0. & 0. & 0. & 0. & 0. & 0. & 0.25 & -0.25 & -0.875 & -1.625 \\ -1.25 & -0.9375 & 0. & 0. & 0. & 0. & 0. & 0. & -0.25 & 0.25 & -0.875 & -1.625 \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.25 & 1.25 \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & -0.25 & 0.25 \end{bmatrix}.$$

Convergence behavior of the network in time  $10^{-7}$  seconds is graphically illustrated in Figure 2.11.

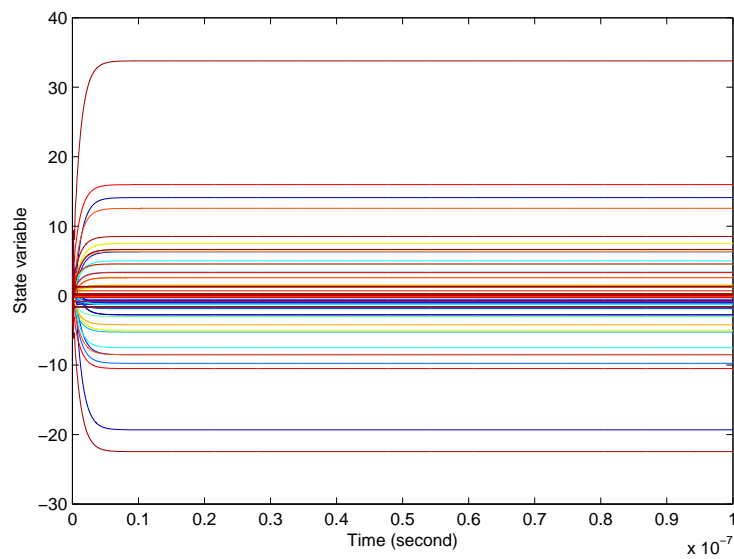


Figure 2.11: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 2.2.2

**Example 2.2.3.** Consider the Hessenberg matrix of the order 6:

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}.$$

The index of  $A$  is equal to  $\text{ind}(A) = 3$ , so that the exact Drazin inverse of  $A$  is equal to

$$A^d = A^3 (A^{2*3+1})^\dagger A^3 = \begin{bmatrix} -\frac{7}{8} & \frac{7}{8} & \frac{7}{16} & -\frac{7}{4} & \frac{29}{32} & \frac{67}{32} \\ -\frac{1}{8} & \frac{1}{8} & \frac{5}{16} & -\frac{3}{4} & \frac{11}{32} & \frac{29}{32} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{3}{2} & -\frac{3}{4} & -\frac{7}{4} \\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{8} & -\frac{1}{2} & \frac{5}{16} & \frac{7}{16} \\ 0 & 0 & \frac{1}{4} & -\frac{1}{2} & \frac{1}{8} & \frac{7}{8} \\ 0 & 0 & -\frac{1}{4} & \frac{1}{2} & -\frac{1}{8} & -\frac{7}{8} \end{bmatrix}.$$

Eigenvalues of the matrix  $GA = A^4$  form the set  $\{135.88225, 16.0, 0.1177, 0., 0., 0.\}$ . Using  $G = A^3$ , and  $\gamma = 10^8$ , RNN produces the following approximation of the Drazin Inverse:

$$\overline{V}_G = A^D = \begin{bmatrix} -0.8750 & 0.8750 & 0.4375 & -1.7500 & 0.9062 & 2.0937 \\ -0.1250 & 0.1250 & 0.3125 & -0.7500 & 0.3437 & 0.9062 \\ 0.5000 & -0.5000 & -0.5000 & 1.5000 & -0.7500 & -1.7500 \\ -0.2500 & 0.2500 & 0.1250 & -0.5000 & 0.3125 & 0.4375 \\ 0. & 0. & 0.2500 & -0.5000 & 0.1250 & 0.8750 \\ 0. & 0. & -0.2500 & 0.5000 & -0.1250 & -0.8750 \end{bmatrix}.$$

Convergence behavior of the network in  $10^{-7}$  seconds is illustrated in Figure 2.12.

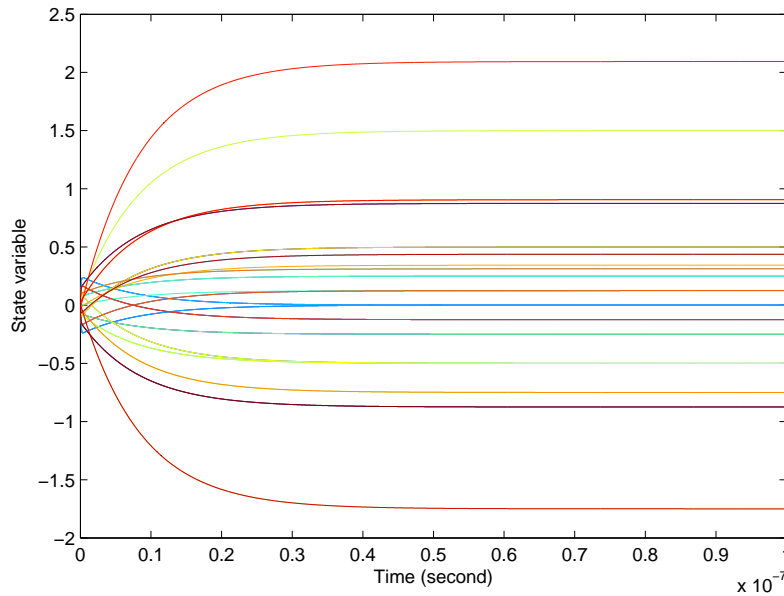


Figure 2.12: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 2.2.3

## 2.3 GNN for computing the $W$ -weighted Drazin inverse

This section is written on the basis of the results from the recent paper [102].

### 2.3.1 About $W$ -weighted Drazin inverse

A rectangular matrix  $X$  is called a  $W$ -weighted Drazin inverse of  $A \in \mathbb{C}^{m \times n}$  if it satisfies three matrix equations

$$(AW)^{k+1}XW = (AW)^k, \quad XWAWX = X, \quad AWX = XWA, \quad (2.3.1)$$

where  $W \in \mathbb{C}^{n \times m}$  (see [18]). Usually, the notation  $X = A_{d,w}$  has been used to denote the  $W$ -weighted Drazin inverse of  $A$ .

In particular, when  $A$  is square and  $W = I$ , the  $W$ -weighted Drazin inverse becomes the Drazin inverse  $X = A^D$  of  $A$ . Further, in the case  $k = 1$  the Drazin inverse reduces to the group inverse  $A^\#$ . The  $W$ -weighted Drazin inverse possesses the following representations (see [18, 113, 114])

$$A_{d,w} = A[(WA)^D]^2 = [(AW)^D]^2 A,$$

and the following properties

$$\mathcal{R}(A_{d,w}) = \mathcal{R}[(AW)^k], \quad \mathcal{N}(A_{d,w}) = \mathcal{N}[(WA)^k].$$

The following several lemmas are needed in what follows to analyze the convergence and stability of proposed neural networks.

**Lemma 2.3.1. ([1])** *Let  $M$  and  $N$  be two positive semi-definite matrices. Then the following statements on the trace hold:*

- (a)  $\text{Tr}(MN) \geq 0$ .
- (b)  $\text{Tr}(M) = 0 \iff M = 0$ .

**Lemma 2.3.2. ([84])** *Let  $M$  and  $N$  be two real square matrices of the order  $n$ . If the matrix  $N$  is symmetric, then the following relation is valid:*

$$\text{Tr}(MN) = \text{Tr}[H(M)N], \quad (2.3.2)$$

where  $H(M) = \frac{1}{2}(M + M^T)$ .

The following result from [68] will be useful in the rest of the section.

**Lemma 2.3.3. ([68])** *If  $A$  is  $W$ -Drazin invertible, then the  $W$ -Drazin inverse  $A_{d,w}$  is a  $\{2\}$ -inverse of  $WAW$  with the range  $\mathcal{R}(A(WA)^k)$  and the null space  $\mathcal{N}(A(WA)^k)$ , where  $k =$*

$\max \{\text{Ind}(AW),$   
 $\text{Ind}(WA)\},$  *i.e.*,

$$A_{d,w} = (WAW)_{\mathcal{R}(A(WA)^k), \mathcal{N}(A(WA)^k)}^{(2)}. \quad (2.3.3)$$

Since  $A(WA)^k = (AW)^k A$ , Lemma 2.3.3 immediately implies

$$A_{d,w} = (WAW)_{\mathcal{R}((AW)^k A), \mathcal{N}((AW)^k A)}^{(2)}. \quad (2.3.4)$$

Let  $\sigma(A)$  be the spectrum of  $A$  and  $s(A) = \text{Re}(\sigma(A)) = \{\text{Re}(\lambda) : \lambda \in \sigma(A)\}$ , then  $s(A) \geq 0$  denotes  $\text{Re}(\lambda_i) \geq 0, i = 1, 2, \dots, n$ .

### 2.3.2 Specific case for $W$ -weighted Drazin inverse

We consider the case when the nonzero eigenvalues of  $(AW)^{l+2}$  lie in the open right-half plane, *i.e.*, the condition  $s((AW)^{l+2}) \geq 0$  holds, for some positive integer  $l$  satisfying  $l \geq k = \max\{\text{ind}(AW), \text{ind}(WA)\}$ . Value of  $l$  could be determined according to rules given in Lemma 4 and Theorem 3 from [84]. Because of the assumed constraint, we use the term ‘specific case’ to denote this RNN approach.

#### Dynamic equation in the specific case

We restrict our investigation to real matrices, *i.e.*, it will be assumed that  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$ .

**Lemma 2.3.4.** [102] *The  $W$ -weighted Drazin inverse of  $A \in \mathbb{R}^{m \times n}$  satisfies the following matrix equation:*

$$(AW)^{l+2} A_{d,w} = (AW)^l A. \quad (2.3.5)$$

*Proof.* Let  $G = (AW)^l A$ . When the existence of  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  is ensured, it is possible to verify that the matrix equations

$$G A A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} = G, \quad A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} A G = G \quad (2.3.6)$$

are satisfied. According to the result representation (2.3.4), in conjunction with (2.3.6), immediately follows the identity

$$(AW)^l A (WAW) A_{d,w} = (AW)^l A, \quad (2.3.7)$$

which is just another appearance of (2.3.5).  $\square$

Let  $V \in \mathbb{R}^{m \times m}$  be the unknown matrix corresponding to  $A_{d,w}$ . According to (2.3.7), the following matrix equation holds:

$$(AW)^{l+2}V - (AW)^lA = 0. \quad (2.3.8)$$

In order to solve (2.3.8) for  $V(t)$  via dynamic-system approach, we can define a scalar-valued norm based error function:

$$E(t) = \frac{\|(AW)^{l+2}V(t) - (AW)^lA\|_F^2}{2}, \quad l \geq k.$$

Note that the minimal value  $E(\bar{t}) = 0$  of the residual-error function  $E(t)$  is achieved in a minimizer  $\bar{V} = V(\bar{t})$  if and only if  $V(\bar{t})$  is the exact solution of (2.3.8). A computational scheme could be designed to evolve along a descent direction of this error function  $E(t)$ , until the minimum  $E(\bar{t})$  is reached. The typical descent direction of  $E(t)$  is defined by the negative gradient  $-\partial E(t)/\partial V$  of  $E(t)$ . The gradient of  $E$  with respect to  $V \in \mathbb{R}^{m \times m}$  could simply be derived as

$$\frac{\partial E(t)}{\partial V} = \left( (AW)^{l+2} \right)^T \left( (AW)^{l+2}V(t) - (AW)^lA \right). \quad (2.3.9)$$

Therefore, the GNN neural dynamics for computing the  $W$ -weighted Drazin inverse is defined by

$$\frac{dV(t)}{dt} = -\gamma \left( (AW)^{l+2}V(t) - (AW)^lA \right), \quad l \geq k, \quad V(0) = V_0 \quad (2.3.10)$$

where  $V(t)$  is a matrix of activation state variables,  $t \in [0, +\infty)$ , and  $\gamma$  is a positive scaling constant. The model (2.3.10) will be referred as GNNADW.

The first term,  $\left( (AW)^{l+2} \right)^T$ , in the right-hand side of (2.3.10) is a constant factor. Therefore, the gradient direction of  $E(t)$  is also defined by  $(AW)^{l+2}V(t) - (AW)^lA$ .

According to design formula

$$\frac{dV(t)}{dt} = -\gamma \frac{\partial E(t)}{\partial V},$$

and by omitting the constant term, the dynamic equation of a gradient recurrent neural network can be defined as

$$\frac{dV(t)}{dt} = -\gamma \left( (AW)^{l+2}V(t) - (AW)^lA \right), \quad l \geq k, \quad V(0) = V_0. \quad (2.3.11)$$

Here,  $V(t)$  is a matrix of activation state variables,  $t \in [0, +\infty)$ , and  $\gamma$  is a positive scaling constant. The model (2.3.11) was proposed in [102] and will be referred as GNNDW.

The reason for this definition could be the fact that if we find the equilibrium state  $\bar{V} = V(\bar{t})$  for dynamical system (2.3.11), then the minimum for the residual-norm function  $E(t)$  is also



achieved, because it satisfies

$$\frac{d\bar{V}}{dt} = 0 \quad (2.3.12)$$

at the equilibrium state  $\bar{V}$ . Thus,

$$-\gamma \left( (AW)^{l+2}V - (AW)^l A \right) = 0, \quad (2.3.13)$$

and, therefore (2.3.8) holds.

The RNN defined in this section is determined by the linear dynamical system (2.3.11) in a matrix form. The closed-form solution of the state matrix can be derived using known results of the linear systems theory (see [35]), and it is equal to

$$\begin{aligned} V(t) = & \exp[-\gamma(AW)^{l+2}t]V(0) + \\ & + \gamma \exp[-\gamma(AW)^{l+2}t] \int_0^t \exp[\gamma(AW)^{l+2}\tau](AW)^l A \, d\tau. \end{aligned} \quad (2.3.14)$$

### Convergence and stability analysis in the specific case of GNNDW

To analyze the convergence and stability of a neural network, the following lemmas are needed in what follows.

**Lemma 2.3.5.** [102] *Let  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$ . Suppose that the condition  $s\left((AW)^{l+2}\right) \geq 0$  is satisfied for some  $l \geq k = \max\{\text{ind}(AW), \text{ind}(WA)\}$  and  $t \in [0, +\infty]$ . Then it holds,*

$$\lim_{t \rightarrow \infty} \exp\left[-\gamma(AW)^{l+2}t\right] = 0.$$

*Proof.* For each singular matrix  $AW \in \mathbb{R}^{m \times m}$  which is not nilpotent, there exists a nonsingular matrix  $P$  such that

$$AW = P \begin{pmatrix} B & O \\ O & N \end{pmatrix} P^{-1},$$

where  $B$  is a nonsingular matrix and  $N$  is nilpotent. It is known that the nilpotency index of  $N$  is equal to the index of  $AW$ .

It is easy to check that

$$(AW)^{l+2} = P \begin{pmatrix} B^{l+2} & O \\ O & O \end{pmatrix} P^{-1},$$

where  $s(B^{l+2}) \geq 0$ . Since

$$\exp\left[-\gamma(AW)^{l+2}t\right] = P \begin{pmatrix} \exp(-\gamma B^{l+2}t) & O \\ O & O \end{pmatrix} P^{-1},$$

from Lemma 1 in [84], it follows that

$$\lim_{t \rightarrow \infty} \exp(-\gamma B^{l+2}t) = 0,$$

and later

$$\lim_{t \rightarrow \infty} \exp[-\gamma(AW)^{l+2}t] = 0.$$

This completes the proof.  $\square$

Now, (2.3.14) and Lemma 2.3.5 imply the following representation for  $\lim_{t \rightarrow \infty} V(t) = \bar{V}$ :

$$\lim_{t \rightarrow \infty} V(t) = \bar{V} = \lim_{t \rightarrow \infty} \gamma \exp[-\gamma(AW)^{l+2}t] \int_0^t \exp[\gamma(AW)^{l+2}\tau] (AW)^l A \, d\tau. \quad (2.3.15)$$

Based on the definition and properties of the matrix exponential, it can be verified that the closed-form solution of  $\bar{V}$  in (2.3.15) represents the  $W$ -weighted Drazin inverse  $A_{d,w}$ , and it is independent of the gain parameter  $\gamma$ .

**Theorem 2.3.1.** [102] *Let  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$ . Let the eigenvalues of  $(AW)^{l+2}$  satisfy  $s((AW)^{l+2}) \geq 0$ , for some  $l \geq k = \max\{\text{ind}(AW), \text{ind}(WA)\}$ . Then the limiting expression (2.3.15) satisfies  $\bar{V} = A_{d,w}$ .*

*Proof.* The following can be derived employing several basic properties of the  $W$ -weighted Drazin inverse:

$$\begin{aligned} \bar{V} &= \lim_{t \rightarrow \infty} \gamma \exp(-\gamma(AW)^{l+2}t) \int_0^t \exp[\gamma(AW)^{l+2}\tau] (AW)^l A \, d\tau \\ &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma(AW)^{l+2}t) \int_0^t \exp(\gamma(AW)^{l+2}\tau) (\gamma(AW)^{l+2}) \, d\tau \right] A_{d,w} \\ &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma(AW)^{l+2}t) \exp(\gamma(AW)^{l+2}\tau) \Big|_{\tau=0}^{\tau=t} \right] A_{d,w} \\ &= \left[ I - \lim_{t \rightarrow \infty} \exp(-\gamma(AW)^{l+2}t) \right] A_{d,w}. \end{aligned}$$

Now, the result of Lemma 2.3.5 implies

$$\bar{V} = A_{d,w},$$

which was our initial intention.  $\square$

If  $A$  is a square and  $W = I$ , then the  $W$ -weighted Drazin inverse  $A_{d,w}$  reduces to Drazin inverse  $A^D$ . The corresponding convergence result is stated in Corollary 2.3.1.

**Corollary 2.3.1.** *Let  $A \in \mathbb{R}^{n \times n}$  and the nonzero eigenvalue  $\lambda_i$  of  $A^{l+1}$ ,  $l \geq \text{Ind}(A)$ , satisfy  $s(A^{l+1}) \geq 0$ . Then the limiting expression (2.3.15) produces the Drazin inverse  $A^D$ , i.e.,  $\bar{V} = A^D$ .*

Note that the equation (2.3.14) can be simplified by forcing the first matrix term in the right-hand side to be zero by setting zero initial states. In light of the above discussion, the dynamical state equation (2.3.11) of the recurrent neural network for computing the  $W$ -weighted Drazin inverse can be stated in the form

$$\frac{dV(t)}{dt} = -\gamma \left( (AW)^{l+2}V(t) - (AW)^l A \right), \quad l \geq \text{Ind}(AW), \quad V(0) = 0. \quad (2.3.16)$$

Now we investigate the stability of the equilibrium state  $\bar{V}$ .

**Theorem 2.3.2.** [102] *Let  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$  with  $l \geq k = \max\{\text{ind}(AW), \text{ind}(WA)\}$ . If  $s((AW)^{l+2}) \geq 0$ , then the gradient-based neural network (2.3.16) is stable in the sense of Lyapunov.*

*Proof.* We consider the following Lyapunov function candidate

$$\begin{aligned} E(t) &= \frac{\|(AW)^{l+2}V - (AW)^l A\|_F^2}{2} \\ &= \frac{\text{Tr} \left( \left( (AW)^{l+2}V - (AW)^l A \right)^T \left( (AW)^{l+2}V - (AW)^l A \right) \right)}{2}. \end{aligned}$$

Clearly, the inequality  $E(t) \geq 0$  holds. Applying the matrix derivative on  $E(t)$ , we derive the following statements:

$$\begin{aligned} \frac{\partial E(t)}{\partial V} &= \frac{1}{2} \frac{\partial \text{Tr} \left( \left( (AW)^{l+2}V - (AW)^l A \right)^T \left( (AW)^{l+2}V - (AW)^l A \right) \right)}{\partial V} \\ &= \frac{1}{2} \left\{ \left[ \left( (AW)^{l+2} \right)^T \left( (AW)^{l+2} \right) \right] + \left[ \left( (AW)^{l+2} \right)^T \left( (AW)^{l+2} \right) \right]^T \right\} V - \left( (AW)^{l+2} \right)^T \left( (AW)^l A \right) \\ &= \left( (AW)^{l+2} \right)^T \left( (AW)^{l+2}V - (AW)^l A \right). \end{aligned}$$

Thus, the time derivative of  $E(t)$  is equal to

$$\begin{aligned} \frac{dE(t)}{dt} &= \text{Tr} \left( \left( \frac{\partial E}{\partial V} \right)^T \frac{dV}{dt} \right) \\ &= \text{Tr} \left( \left( (AW)^{l+2}V - (AW)^l A \right)^T (AW)^{l+2} \left( -\gamma \left( (AW)^{l+2}V - (AW)^l A \right) \right) \right) \\ &= -\gamma \text{Tr} \left( (AW)^{l+2} \left( (AW)^{l+2}V - (AW)^l A \right) \left( (AW)^{l+2}V - (AW)^l A \right)^T \right). \end{aligned}$$

It follows from the identity (5.1.5) of Lemma 2.3.2 that

$$\frac{dE(t)}{dt} = -\gamma \text{Tr} \left( H((AW)^{l+2}) \left( (AW)^{l+2}V - (AW)^lA \right) \left( (AW)^{l+2}V - (AW)^lA \right)^T \right).$$

Since the eigenvalues of a square matrix are equal to the eigenvalues of its transpose, and using the assumption that the nonzero eigenvalues  $\lambda_i$  of  $(AW)^{l+2}$  satisfy  $\text{Re}(\lambda_i) > 0$ , together with the fact that  $H((AW)^{l+2})$  is symmetric, thus,  $H((AW)^{l+2})$  is positive semi-definite. The matrices  $H((AW)^{l+2})$  and

$$\left( (AW)^{l+2}V - (AW)^lA \right) \left( (AW)^{l+2}V - (AW)^lA \right)^T$$

are both positive semi-definite. According to Lemma 2.3.1, the trace of their product is non-negative. Therefore

$$\frac{dE(t)}{dt} \leq 0,$$

so the gradient-based neural network is stable and the proof is thus complete.  $\square$

Particular, if  $A \in \mathbb{R}^{n \times n}$  and take  $W = I$ , then the  $W$ -weighted Drazin inverse  $A_{d,w}$  reduces to the Drazin inverse  $A^D$ . We also obtain the stability of the equilibrium state  $\bar{V}$  as follows.

**Corollary 2.3.2.** *Let  $A \in \mathbb{R}^{n \times n}$ . If  $s(A^{l+1}) \geq 0$ , for some  $l \geq \text{ind}(A)$ , then the gradient-based neural network (2.3.16) is stable in the sense of Lyapunov.*

# Chapter 3

## GNN for computing generalized inverse without restriction on spectrum

*All considered models in Chapter 2 have a drawback concerning the convergence. Namely, the convergence of these GNNs is conditioned by the nonnegativity of real parts of eigenvalues of certain matrices. In this chapter, we present the dynamic equations and corresponding recurrent neural networks for computing the generalized inverses for arbitrary real matrix, convergent without any restriction on its eigenvalues. These neural networks resolve the drawback of considered GNN models from the previous chapter, at the cost of increasing the number of matrix operations. The structure of the neural networks introduced in this chapter is the same as the structure discussed in Chapter 2. We discuss conditions which ensure the stability of the defined recurrent neural networks as well as its convergence. Several illustrative examples present the results of computer simulations.*

### 3.1 Globally convergent GNN for computing Drazin inverse

#### 3.1.1 Preliminaries and motivation

The stability of the gradient-based neural networks for computing the usual inverse and the Moore-Penrose inverse [94, 96] is warranted, since the matrix  $M = A^T A \in \mathbb{R}^{n \times n}$  possesses the spectrum  $\sigma(M) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  which satisfies

$$\lambda_j \geq 0, \quad j = 1, \dots, n. \quad (3.1.1)$$

In the case of the weighted Moore-Penrose inverse [109] the stability is ensured since the spectrum  $\sigma(M) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  of the matrix  $M = A^\# A \in \mathbb{R}^{n \times n}$  also satisfies (3.1.1).

In addition, the conditions that must be imposed to the spectrum of  $A^m$  ensure the stability

of the gradient-based neural network initiated in [84, Lemma 1]. In the case of the Drazin inverse, the stability is ensured in the case when the spectrum  $\sigma(M) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  of the matrix  $M = A^m A = A^{m+1} \in \mathbb{R}^{n \times n}$  satisfies

$$\operatorname{Re}(\lambda_j) \geq 0, \quad j = 1, \dots, n. \quad (3.1.2)$$

One of the main results of [84] was the algorithm, proposed in Lemma 4, for estimating an appropriate exponent  $m$  such that the power  $A^{m+1}$  satisfies (3.1.2). But, that algorithm may request an unbounded in advance exponent  $m$ . Now, we present an alternative approach developed by Stanimirović, Živković and Wei in [85]. Basically, instead of the matrix  $A^m$ , where the exponent  $m$  is defined as in [84, Lemma 4], our current intention is to use the matrix  $A^k (A^{2k+1})^T A^k$ ,  $k = \operatorname{ind}(A)$ . That matrix requires relatively great exponent  $A^{4k+1}$ ,  $k = \operatorname{ind}(A)$ . But, this exponent is fixed and predefined in advance. Necessary details will be explained in the next section.

### 3.1.2 Neural network architecture

We follow the main leading idea from [23]. Namely, the goal here is to drop the restriction (2.1.4) imposed on the spectrum of  $A^{m+1}$ . Guided by this idea, the authors of [23] introduced an additional representation of the Drazin inverse  $A^D$ , which is applicable without any restriction on the eigenvalues of  $A$ .

**Proposition 3.1.1.** [23] *Suppose that  $A \in \mathbb{C}^{n \times n}$  and  $k = \operatorname{ind}(A)$ . Then*

$$A^D = \int_0^\infty \exp\left(-A^k (A^{2k+1})^T A^{k+1} \tau\right) A^k (A^{2k+1})^T A^k d\tau. \quad (3.1.3)$$

As we have seen in the previous chapter, the general property of GNNs for computing generalized inverses is based on a main principle which requires solving one representative matrix equation via dynamic-system approach. Further, the dynamic-system approach is defined as a scalar-valued norm based error function  $E(t)$ . Then the proposed approaches try to find the minimum for the residual-norm function  $E(t)$  using the design formula  $dV(t)/dt = -\gamma \partial E(t)/\partial V$ . Therefore, the main idea used in [84, 96, 108] is the same as the leading principle from [118]: convert the generalized inverse problem into a matrix norm optimization problem.

Here we continue the described approach. The leading notions will be the integral representations of the Drazin inverse in conjunction with the dynamic-system equation which is defined as an appropriately defined norm based error function  $E(t)$ . To simplify notation, we use the

substitution  $G = A^k (A^{2k+1})^T A^k$ ,  $k = \text{ind}(A)$ . Clearly that

$$GAA^D = G$$

is satisfied. Then the following matrix equation with respect to unknown matrix  $V$  can be considered:

$$GAV - G = 0. \quad (3.1.4)$$

The scalar-valued norm based error function corresponding to (3.1.4) is defined as

$$E(t) = \frac{\|GAV(t) - G\|_F^2}{2}. \quad (3.1.5)$$

Note that the minimal value  $E(\bar{t}) = 0$  of  $E(t)$  is achieved if and only if  $V(\bar{t})$  is the exact solution of (3.1.4). The gradient descent direction of  $E(t)$  is defined by  $-\partial E(t)/\partial V$ . The gradient of  $E$  with respect to  $V \in \mathbb{R}^{n \times n}$  could simply be derived as (see, for example, [25, Chapter 5])

$$\frac{\partial E(t)}{\partial V} = (GA)^T (GAV(t) - G). \quad (3.1.6)$$

To make the equation simpler, the authors of [85] omitted the constant factor  $(GA)^T$  and the dynamics corresponding to the integral representation (3.1.3) can be defined as

$$\frac{dV(t)}{dt} = -\gamma G (AV(t) - I), \quad V(0) = V_0, \quad G = A^k (A^{2k+1})^T A^k, \quad k = \text{ind}(A). \quad (3.1.7)$$

Here,  $V(t)$  is a matrix of activation state variables,  $t \in [0, +\infty)$ ,  $\gamma$  is a positive scaling constant [94, 97].

The recurrent neural network (3.1.7) is a linear dynamic system in matrix form. The closed-form solution of the state matrix can be described as follows (see [35]):

$$V(t) = \exp(-\gamma GAt) V(0) + \gamma \exp(-\gamma GAt) \int_0^t \exp(\gamma GA\tau) G d\tau. \quad (3.1.8)$$

Lemma 3.1.1 along with Proposition 3.1.2 provide necessary conditions that the first matrix term in the right-hand side of (3.1.8) approaches the zero matrix of the same size as time approaches infinity, regardless of the initial states; i.e., for arbitrary  $V(0)$

$$\lim_{t \rightarrow \infty} \exp(-\gamma GAt) V(0) = 0. \quad (3.1.9)$$

**Lemma 3.1.1.** *Let  $M \in \mathbb{R}^{n \times n}$  be given matrix,  $\sigma(M) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  be the spectrum of*

$M$ , and  $t \in [0, +\infty)$ . Suppose that the condition (3.1.2) is satisfied in the case of  $\sigma(M)$ . Then the following limiting expression holds:

$$\lim_{t \rightarrow \infty} \exp(-Mt) = 0. \quad (3.1.10)$$

*Proof.* This result is a generalization of Lemma 1 from [84] and can be proved in the same way.  $\square$

**Proposition 3.1.2.** *The matrix  $GA = A^k (A^{2k+1})^T A^{k+1}$ ,  $k = \text{ind}(A)$ , from (3.1.8) has a nonnegative spectrum, i.e.  $\sigma(GA) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$  satisfies  $\lambda_j \geq 0$ ,  $j = 1, \dots, n$ , for arbitrary given matrix  $A \in \mathbb{R}^{n \times n}$ .*

*Proof.* The proof of this statement has been shown in the proof of Theorem 2.1 in [23].  $\square$

**Corollary 3.1.1.** *The limit expression (3.1.9) is valid for arbitrary matrix  $A \in \mathbb{R}^{n \times n}$ .*

The result of Corollary 3.1.1 is very important, and shows that (3.1.9) holds for arbitrary  $V(0)$  as well as for arbitrary matrix  $A$ , in both nonsingular and singular cases. Now, equations (3.1.8) and (3.1.9) imply the following representation for  $\lim_{t \rightarrow \infty} V(t) = \bar{V}$ :

$$\bar{V} = \lim_{t \rightarrow \infty} \gamma \exp(-\gamma GA t) \int_0^t \exp(\gamma GA \tau) G d\tau. \quad (3.1.11)$$

Next we verify, based on the definition and properties of matrix exponential, that the closed-form solution of  $\bar{V}$  in (3.1.11) satisfies (1<sup>k</sup>), (2), (5) and it is independent of the parameter  $\gamma$ . This means that (3.1.11) and (2.1.4) produce the same result,  $A^D$ .

**Theorem 3.1.1.** [85] *Let  $A \in \mathbb{R}^{n \times n}$  be given matrix, and  $k = \text{ind}(A)$ . Then the limiting expression (3.1.11) produces the Drazin inverse  $A^D$ , i.e.,*

$$\bar{V} = A^D.$$

*Proof.* According to the basic property (1<sup>k</sup>) of the Drazin inverse, immediately follows

$$G = A^k (A^{2k+1})^T A^k = A^k (A^{2k+1})^T A^{k+1} A^D = G A A^D,$$

which, in conjunction with (3.1.11), implies



$$\begin{aligned}
 \bar{V} &= \lim_{t \rightarrow \infty} \gamma \exp\left(-\gamma A^k \left(A^{2k+1}\right)^T A^{k+1} t\right) \int_0^t \exp\left(\gamma A^k \left(A^{2k+1}\right)^T A^{k+1} \tau\right) A^k \left(A^{2k+1}\right)^T A^k d\tau \\
 &= \lim_{t \rightarrow \infty} \exp(-\gamma G A t) \int_0^t \exp(\gamma G A \tau) \gamma G A A^D d\tau \\
 &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma G A t) \int_0^t d(\exp(\gamma G A \tau)) \right] A^D \\
 &= \left[ \lim_{t \rightarrow \infty} \exp(-\gamma G A t) \exp(\gamma G A \tau) \Big|_{\tau=0}^{\tau=t} \right] A^D \\
 &= \left\{ \lim_{t \rightarrow \infty} \exp(-\gamma G A t) [\exp(\gamma G A t) - I] \right\} A^D \\
 &= \left[ I - \lim_{t \rightarrow \infty} \exp(-\gamma G A t) \right] A^D.
 \end{aligned}$$

An application of Corollary 3.1.1 gives  $\lim_{t \rightarrow \infty} \exp(-\gamma G A t) = 0$  and  $\bar{V} = A^D$ .  $\square$

The equation (3.1.8) can be simplified by forcing the first matrix term in the right-hand side to be zero by setting zero initial states. For this purpose, it is useful to appoint  $V(0) = 0$  in the first matrix term in the right-hand of (3.1.8), which will also speed up the convergence. Therefore the dynamic state equation of the recurrent neural network for computing the Drazin inverse can be presented as follows:

$$\frac{dV}{dt} = -\gamma G (AV(t) - I), \quad V(0) = 0, \quad G = A^k \left(A^{2k+1}\right)^T A^k, \quad k = \text{ind}(A). \quad (3.1.12)$$

Now we are investigating the stability of the equilibrium state  $\bar{V}$ . Before the main result, which shows that the equilibrium state is stable in the sense of Lyapunov, we restate the following Lemma 3.1.2 from [1], for the sake of completeness.

**Lemma 3.1.2.** *Let  $M$  and  $N$  be two positive semi-definite matrices. Then*

$$\text{Tr}(MN) \geq 0. \quad (3.1.13)$$

**Theorem 3.1.2.** [85] *The recurrent neural network described in (3.1.12) is stable in the sense of Lyapunov.*

*Proof.* The Lyapunov's method (see [63, Chapter 2.9]) is used to prove the stability. The Lyapunov function candidate can be instinctively defined as

$$E(t) = \frac{\|GAV - G\|_F^2}{2} = \frac{\text{Tr} \left( (GAV - G)^T (GAV - G) \right)}{2}. \quad (3.1.14)$$

It is clear that the inequality  $E(t) \geq 0$  is true.

Applying known matrix calculus (see, for example [25, Chapter 5]) on  $E(t)$ , it is not difficult to verify

$$\frac{\partial E(t)}{\partial V} = (GA)^T(GAV - G).$$

Thus, the time derivative of  $E(t)$  is equal to

$$\begin{aligned} \frac{dE(t)}{dt} &= \text{Tr} \left( \left( \frac{\partial E}{\partial V} \right)^T \frac{dV}{dt} \right) \\ &= \text{Tr} \left( (GAV - G)^T GA (-\gamma (GAV - G)) \right) \\ &= -\gamma \text{Tr} \left( GA (GAV - G) (GAV - G)^T \right). \end{aligned} \tag{3.1.15}$$

In the rest of the proof we consider the Hermitian part of a square real matrix  $GA$ , defined by

$$H(GA) = \frac{1}{2} (GA + (GA)^T).$$

Since  $(GAV - G)(GAV - G)^T$  and  $H(GA)$  are both symmetric, after some algebraic transformation one can verify

$$\frac{dE(t)}{dt} = -\gamma \text{Tr} \left( H(GA) (GAV - G) (GAV - G)^T \right).$$

In accordance with the known fact that all eigenvalues of a square matrix are equal to the eigenvalues of its transpose, an application of Proposition 3.1.2 implies the nonnegativity of  $\lambda_j \geq 0$ , for each  $\lambda_j \in \sigma(H(GA))$ . Moreover, since  $H(GA)$  is symmetric, it is positive semi-definite. Finally, since both the matrices  $H(GA)$  and  $(GAV - G)(GAV - G)^T$  are positive semi-definite, another application of Proposition 3.1.2 immediately implies

$$\frac{dE(t)}{dt} \leq 0.$$

Thus, by Lyapunov stability theory, neural network (3.1.12) is stable, therefore, the proof is complete.  $\square$

**Remark 3.1.1.** *The positive real scaling constant  $\gamma$  could be set as large as possible. This will speed up the process of computation, because it is clear that the term*

$$\lim_{t \rightarrow \infty} \exp \left( -\gamma A^k \left( A^{2k+1} \right)^T A^{k+1} t \right), \quad k = \text{ind}(A)$$

*will vanish faster for larger  $\gamma$ , because  $\gamma$  multiplies the time parameter  $t$ . Usually, values of  $\gamma$  should be as large as the hardware permits, or selected appropriately for simulative and/or*

*experimental purposes.*

The configuration of a neural network is similar to what it has already been discussed in [94, 96]. It is composed from a number of independent sub-networks where each sub-network consists of  $n$  massively connected linear neurons in a single layer, and each layer represents a column vector of  $V(t)$ . Simple linear activation functions can be used instead of nonlinear ones. We denote  $v_j(t)$  (resp.  $a_j$ ) as the  $j$ th column vector of  $V(t)$  (resp.  $A^k (A^{2k+1})^T A^k$ ), for  $j = 1, 2, \dots, n$ . The dynamics of the  $j$ th sub-network can be expressed as follows:

$$\frac{dv_j(t)}{dt} = -\gamma \left( A^k (A^{2k+1})^T A^{k+1} v_j(t) - a_j \right). \quad (3.1.16)$$

The dynamics defined in (3.1.16) indicates that each sub-network is essentially the same as the recurrent neural network presented in [94]. The connection weight matrix  $W$  is identical for each sub-network,  $W = -\gamma A^k (A^{2k+1})^T A^{k+1}$ , and the biasing threshold vector for the  $j$ th sub-network is  $\gamma a_j = \{\gamma a_{1j}, \dots, \gamma a_{nj}\}$ . Elements of the matrix  $W$  (resp.  $V$ ) are denoted by  $w_{ij}$  (resp.  $v_{ij}$ ). Figure 3.1 depicts the architecture of the proposed recurrent neural network for computing the Drazin inverse.

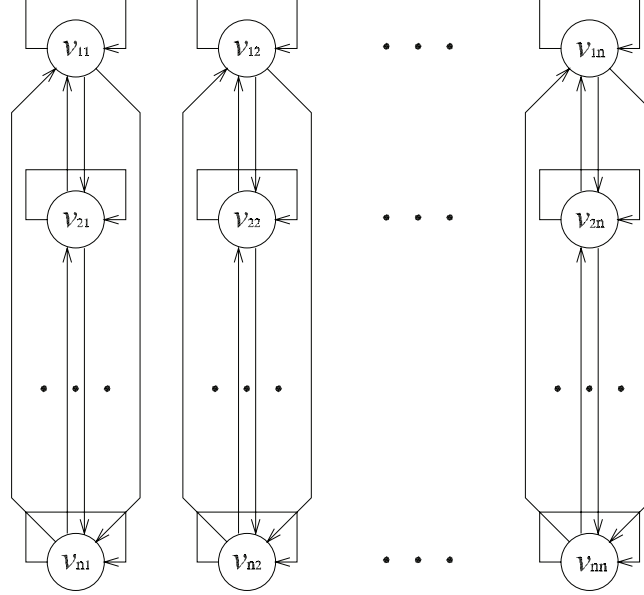


Figure 3.1: Architecture of the RNN for computing the Drazin inverse

Elements of unknown matrix  $V(t)$  are computed using the relation (34) from [84]. This essentially means that the elements of the column  $v_j = \{v_{1j}, \dots, v_{nj}\}$  are generated in the  $j$ th sub-network.

### 3.1.3 Illustrative examples

Several computer simulations have been done in order to check the validity and performance of the neural network approach in the computation of the Drazin inverse. In all examples it is assumed that  $A^D$  denotes the exact Drazin inverse of  $A$  and  $A^d$  denotes its approximation derived from RNN approach. All the tests are performed on Intel Core i3-4130 CPU machine, with 8GB RAM.

**Example 3.1.1.** Consider the following singular matrix from [115]

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

Here  $\sigma(A) = \{0, 0, 0, 0, 2, 2, 2, 2\}$ . The following approximation of the Drazin inverse  $A^D$  is obtained using  $k = \text{ind}(A) = 4$  and  $\gamma = 10^4$  after  $10^{-7}$  seconds:

$$A^d = \begin{bmatrix} 0.2500 & -0.2500 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.2500 & 0.2500 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2500 & -0.2500 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.2500 & 0.2500 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.0625 & 0.0625 & 0.2500 & -0.2500 & 0 & 0 \\ 0 & 0 & -0.0625 & 0.0625 & -0.2500 & 0.2500 & 0 & 0 \\ 0.0625 & 0.0625 & -0.0625 & -0.0625 & 0 & 0 & 0.2500 & -0.2500 \\ -0.0625 & -0.0625 & 0.1875 & -0.0625 & 0 & 0 & -0.2500 & 0.2500 \end{bmatrix}.$$

Elementwise convergence behavior of the network in  $10^{-7}$  seconds is graphically illustrated in Figure 3.2.

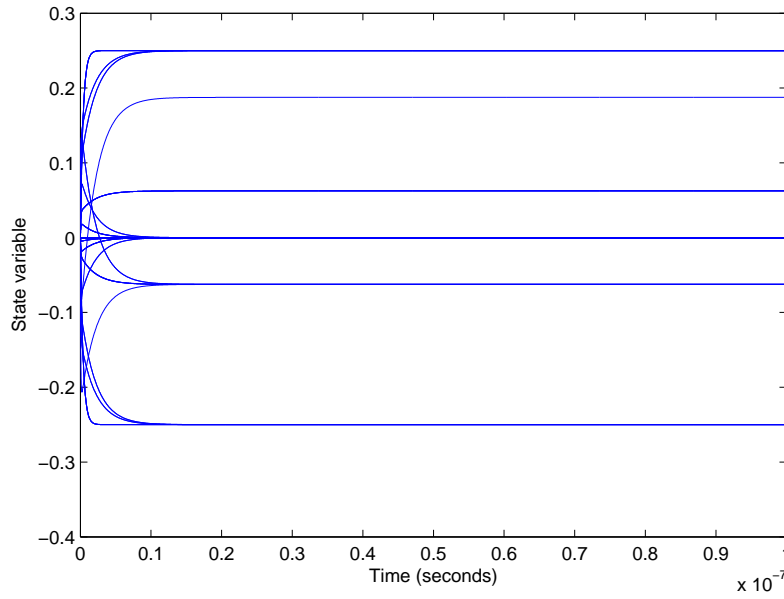


Figure 3.2: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 3.1.1.

Let us mention that the exact Drazin inverse of  $A$  is equal to

$$A^D = \begin{bmatrix} \frac{1}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & -\frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{16} & \frac{1}{16} & \frac{1}{4} & -\frac{1}{4} & 0 & 0 \\ 0 & 0 & -\frac{1}{16} & \frac{1}{16} & -\frac{1}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{16} & \frac{1}{16} & -\frac{1}{16} & -\frac{1}{16} & 0 & 0 & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{16} & -\frac{1}{16} & \frac{3}{16} & -\frac{1}{16} & 0 & 0 & -\frac{1}{4} & \frac{1}{4} \end{bmatrix}.$$

**Example 3.1.2.** Consider the following singular matrix

$$A = \begin{bmatrix} \frac{3}{2} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{4} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & \frac{3}{4} & -\frac{3}{4} & 0 & 0 & 0 & 0 \\ -1 & -1 & -\frac{3}{4} & \frac{3}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{3}{4} & -\frac{3}{4} & -1 & -1 \\ 0 & 0 & -1 & 0 & -\frac{3}{4} & \frac{3}{4} & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\frac{1}{4} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{3}{2} \end{bmatrix}.$$

The eigenvalues of matrix  $A$  are included in the set

$$\sigma(A) = \{1.5, 1.5, 1.25 + 0.1443i, 1.25 - 0.1443i, 1.25 - 0.1443i, 1.25 + 0.1443i\}.$$

Using  $k = \text{ind}(A) = 2$ , and  $\gamma = 10^8$  we get after  $10^{-7}$  seconds the following approximation of the Drazin inverse:

$$A^d = \begin{bmatrix} 0.6316 & -0.2105 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1579 & 0.9474 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.6150 & -0.5319 & 0.3333 & -0.3333 & 0 & 0 & 0 & 0 \\ -0.6150 & -0.5319 & -0.3333 & 0.3333 & 0 & 0 & 0 & 0 \\ 0.6166 & 0.5282 & -0.1111 & 0.3333 & 0.3333 & -0.3333 & -0.5319 & -0.6150 \\ -0.1442 & -0.1538 & -0.1111 & -0.1111 & -0.3333 & 0.3333 & -0.5319 & -0.6150 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9474 & 0.1579 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.2105 & 0.6316 \end{bmatrix}.$$

Convergence behavior of the network used in Example 3.1.2 in  $10^{-7}$  seconds is illustrated in Figure 3.3.

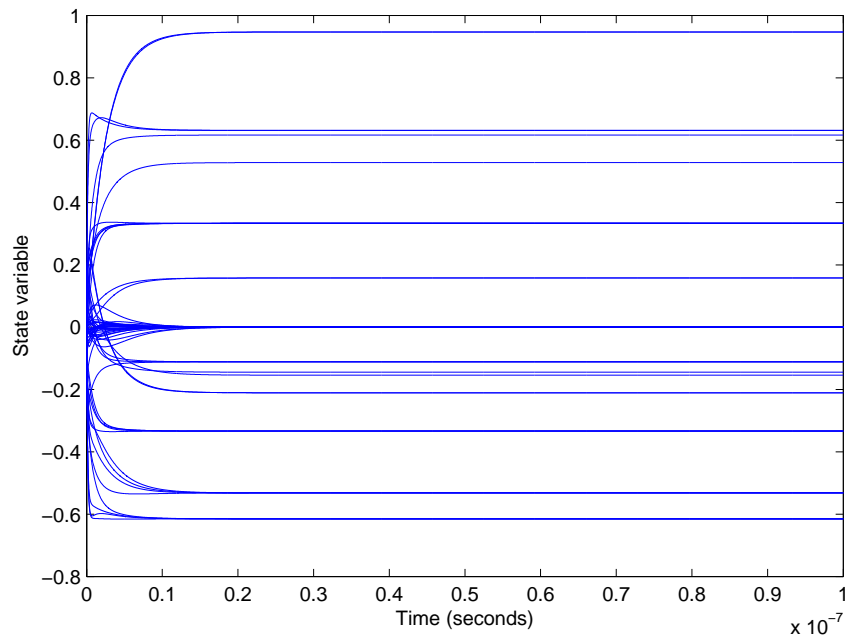


Figure 3.3: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 3.1.2.

The exact Drazin inverse is

$$A^D = \begin{bmatrix} \frac{12}{19} & -\frac{4}{19} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{19} & \frac{18}{19} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{222}{361} & -\frac{192}{361} & \frac{1}{3} & -\frac{1}{3} & 0 & 0 & 0 & 0 \\ -\frac{222}{361} & -\frac{192}{361} & -\frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{12688}{20577} & \frac{32608}{61731} & -\frac{1}{9} & \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} & -\frac{192}{361} & -\frac{222}{361} \\ -\frac{2968}{20577} & -\frac{9496}{61731} & -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{3} & \frac{1}{3} & -\frac{192}{361} & -\frac{222}{361} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{18}{19} & \frac{3}{19} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{4}{19} & \frac{12}{19} \end{bmatrix}.$$

To monitor the network convergence, it is appropriate to use the norm of the computational error  $\|V(t) - A^D(t)\|$ . Figure 3.4 shows that the state matrices of presented neural network all converge to the theoretical inverse  $A^D$  and computational errors  $\|V(t) - A^D(t)\|$  all converge to zero, for three different values of  $\gamma$ . Also, it is observable that the convergence can be accelerated by increasing  $\gamma = \gamma$ .

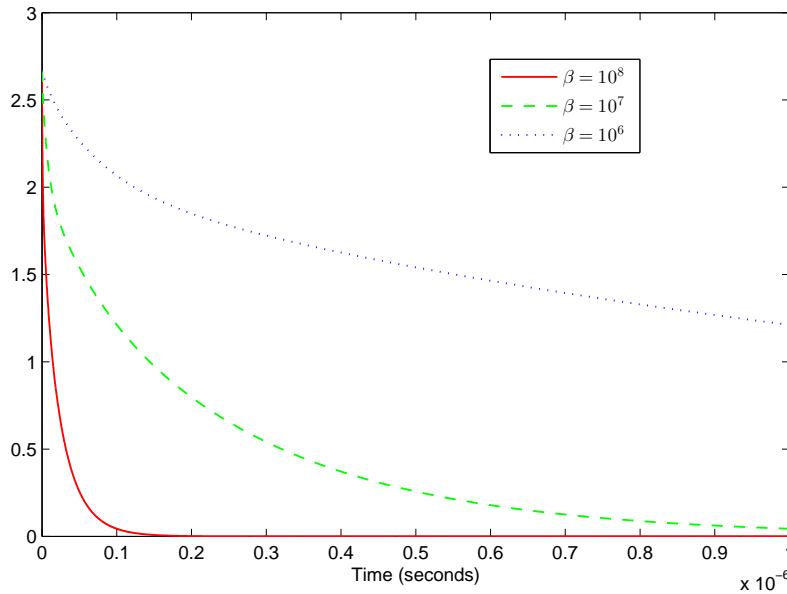


Figure 3.4: Convergence of  $\|V(t) - A^D\|$  for three different values  $\gamma = \gamma$ .

**Example 3.1.3.** *In this example we will show that our neural network produces the regular inverse in the case when the input matrix is regular. Consider the following randomly generated matrix*

$$A = \begin{bmatrix} 0.0046 & 0.8001 & 0.8693 & 0.5132 & 0.4173 & 0.9001 & 0.0965 \\ 0.7749 & 0.4314 & 0.5797 & 0.4018 & 0.04965 & 0.3692 & 0.1320 \\ 0.8173 & 0.9106 & 0.5499 & 0.0760 & 0.9027 & 0.1112 & 0.9421 \\ 0.8687 & 0.1818 & 0.1450 & 0.2399 & 0.9448 & 0.7803 & 0.9561 \\ 0.0844 & 0.2638 & 0.8530 & 0.1233 & 0.4909 & 0.3897 & 0.5752 \\ 0.3998 & 0.1455 & 0.6221 & 0.1839 & 0.4893 & 0.2417 & 0.05978 \\ 0.2599 & 0.1361 & 0.3510 & 0.2400 & 0.3377 & 0.4039 & 0.2348 \end{bmatrix}.$$

*The matrix  $A$  is regular, so it satisfies  $\text{ind}(A) = 0$ . The spectrum of  $A$  is equal to  $\sigma(A) = \{3.0953, -0.9778, -0.5571, 0.2904 + 0.3749i, 0.2904 - 0.3749i, 0.2354, -0.1833\}$ . Using  $k = \text{ind}(A) = 0$  and  $\gamma = 10^{11}$ , the neural network produces the following approximation of the Drazin inverse after  $10^{-7}$  seconds:*

$$A^d = \begin{bmatrix} 0.0439 & 1.2022 & -0.4308 & 1.1517 & 0.0312 & 1.1600 & -4.0275 \\ 1.0869 & 0.1738 & 0.5595 & 0.2298 & -0.4034 & -0.1539 & -2.6975 \\ -0.0877 & 0.5401 & -0.4067 & -0.0575 & 1.3195 & 0.8319 & -1.8463 \\ -2.3972 & -1.4086 & 2.0950 & -4.6892 & -2.6342 & -3.6889 & 19.8601 \\ -0.0888 & -1.6265 & 0.6749 & -0.4814 & -1.2210 & 1.1998 & 2.8891 \\ 1.7409 & 0.8704 & -1.6648 & 2.8154 & 1.0611 & 1.1242 & -8.8760 \\ -0.9648 & 0.0431 & 0.5126 & -0.6807 & 0.8499 & -2.3282 & 3.8572 \end{bmatrix}.$$

*The convergence behavior of the network in  $10^{-7}$  seconds is graphically illustrated in Figure 3.5.*

*The exact Drazin inverse of  $A$  is equal to*

$$A^D = A^{-1} = \begin{bmatrix} 0.0440 & 1.2022 & -0.4308 & 1.1517 & 0.0312 & 1.1600 & -4.0275 \\ 1.0869 & 0.1738 & 0.5595 & 0.2298 & -0.4034 & -0.1539 & -2.6975 \\ -0.0877 & 0.5401 & -0.4067 & -0.0575 & 1.3195 & 0.8319 & -1.8463 \\ -2.3972 & -1.4086 & 2.0950 & -4.6892 & -2.6342 & -3.6889 & 19.8601 \\ -0.0888 & -1.6265 & 0.6749 & -0.4814 & -1.2210 & 1.1998 & 2.8891 \\ 1.7409 & 0.8704 & -1.6648 & 2.8154 & 1.0611 & 1.1242 & -8.8760 \\ -0.9648 & 0.0431 & 0.5126 & -0.6807 & 0.8499 & -2.3282 & 3.8572 \end{bmatrix}.$$



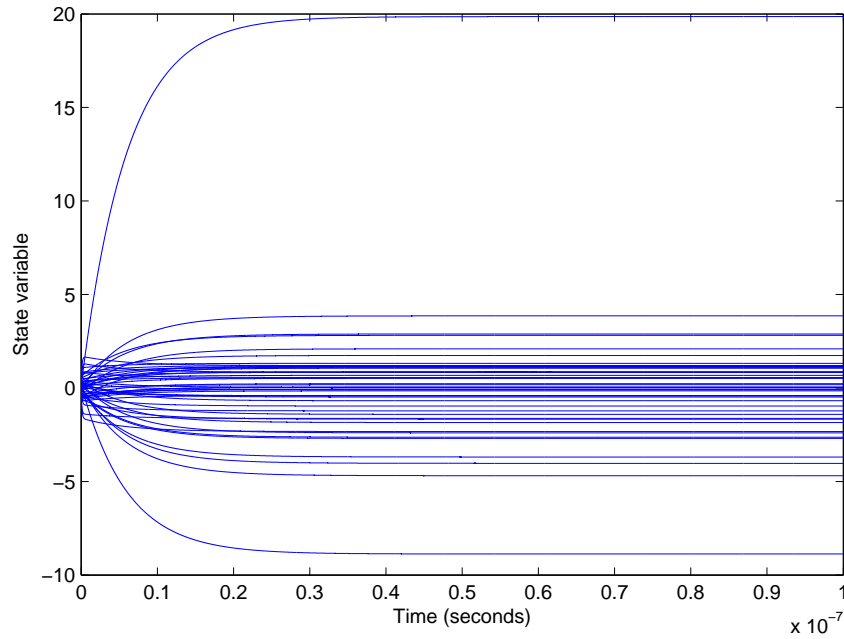


Figure 3.5: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 3.1.3.

**Example 3.1.4.** Consider the following Hessenberg matrix of the order  $n = 4$ :

$$A = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 4 & -2 & 1 & 0 \\ -8 & 4 & -2 & 1 \\ 16 & -8 & 4 & -2 \end{bmatrix}.$$

Here, the spectrum of  $A$  is equal to  $\sigma(A) = \{-6, -2, 0, 0\}$ . Again, using  $\gamma = 10^7$ , we obtain the following approximation of the Drazin inverse, after  $10^{-7}$  seconds:

$$A^D = \begin{bmatrix} -0.2778 & 0.1389 & -0.00463 & -0.03009 \\ 0.1111 & -0.05555 & 0.01852 & -0.004628 \\ 0.6667 & -0.3333 & -0.05556 & 0.1389 \\ -1.333 & 0.6667 & 0.1111 & -0.2778 \end{bmatrix}.$$

The exact Drazin inverse of  $A$  is equal to

$$A^d = \begin{bmatrix} -0.2778 & 0.1389 & -0.00463 & -0.03009 \\ 0.1111 & -0.05555 & 0.01852 & -0.004628 \\ 0.6667 & -0.3333 & -0.05556 & 0.1389 \\ -1.333 & 0.6667 & 0.1111 & -0.2778 \end{bmatrix}.$$

The convergence behavior of the network in  $10^{-7}$  seconds is graphically illustrated in Figure 3.6.

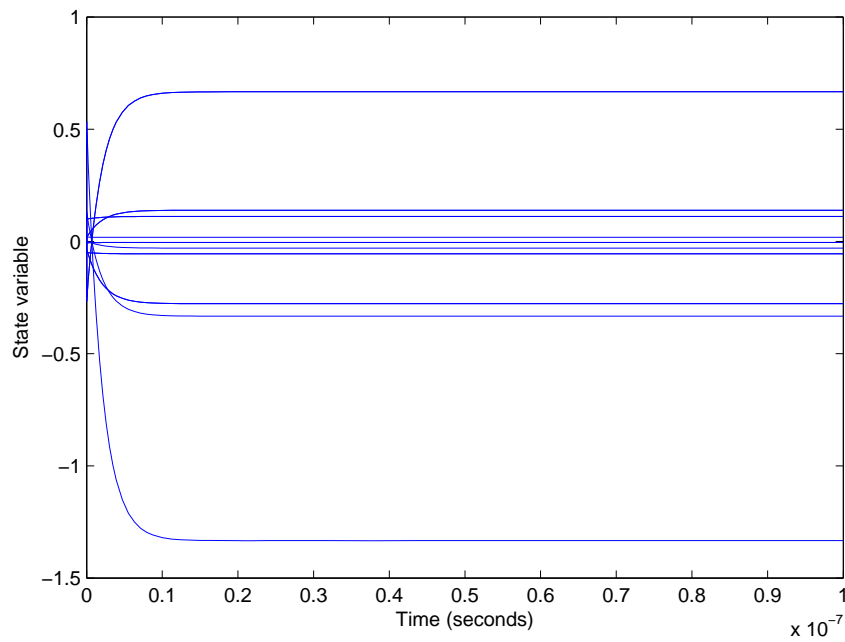


Figure 3.6: Convergence behavior of the RNN in  $10^{-7}$  seconds for Example 3.1.4.

**Example 3.1.5.** Consider the following well known Rosser matrix:

$$A = \begin{bmatrix} 611 & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\ 196 & 899 & 113 & -192 & -71 & -43 & -8 & -44 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 52 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ -52 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ -49 & -8 & 8 & 59 & 208 & 208 & 99 & -911 \\ 29 & -44 & 52 & -23 & 208 & 208 & -911 & 99 \end{bmatrix}.$$

The matrix is  $8 \times 8$  with integer elements. It has a double eigenvalue, three nearly equal eigenvalues, dominant eigenvalues of the opposite sign, a zero eigenvalue and a small nonzero eigenvalue:

$$\sigma(A) = \begin{bmatrix} -1.020049018429998e + 03 \\ 0.0000000000000001e + 03 \\ 0.000098048640722e + 03 \\ 1.000000000000000e + 03 \\ 1.000000000000000e + 03 \\ 1.019901951359279e + 03 \\ 1.020000000000000e + 03 \\ 1.020049018429996e + 03 \end{bmatrix}.$$

The exact Drazin inverse of  $A$  is equal to

$$A^D = \begin{bmatrix} 4.0406 & -2.0198 & 2.0198 & -4.0396 & -0.2000 & 0.2000 & 0.4000 & -0.4000 \\ -2.0198 & 1.0109 & -1.0099 & 2.0198 & 0.0999 & -0.1001 & -0.2000 & 0.2000 \\ 2.0198 & -1.0099 & 1.0109 & -2.0198 & -0.0999 & 0.1001 & 0.2000 & -0.2000 \\ -4.0396 & 2.0198 & -2.0198 & 4.0406 & 0.2000 & -0.2000 & -0.4000 & 0.4000 \\ -0.2000 & 0.0999 & -0.0999 & 0.2000 & 0.0103 & -0.0105 & -0.0196 & 0.0200 \\ 0.2000 & -0.1001 & 0.1001 & -0.2000 & -0.0105 & 0.0103 & 0.0200 & -0.0196 \\ 0.4000 & -0.2000 & 0.2000 & -0.4000 & -0.0196 & 0.0200 & 0.0397 & -0.0405 \\ -0.4000 & 0.2000 & -0.2000 & 0.4000 & 0.0200 & -0.0196 & -0.0405 & 0.0397 \end{bmatrix}.$$

Neural network produces the following approximation of the Drazin inverse:

$$A^d = \begin{bmatrix} 4.0406 & -2.0198 & 2.0198 & -4.0396 & -0.2000 & 0.1999 & 0.4000 & -0.4000 \\ -2.0198 & 1.0109 & -1.0099 & 2.0198 & 0.1000 & -0.1001 & -0.2000 & 0.2000 \\ 2.0198 & -1.0099 & 1.0109 & -2.0198 & -0.1000 & 0.1001 & 0.2000 & -0.2000 \\ -4.0396 & 2.0198 & -2.0198 & 4.0406 & 0.2000 & -0.2000 & -0.4000 & 0.4000 \\ -0.2000 & 0.0999 & -0.0999 & 0.2000 & 0.0104 & -0.0105 & -0.0196 & 0.0200 \\ 0.2000 & -0.1001 & 0.1000 & -0.2000 & -0.0104 & 0.0103 & 0.0200 & -0.0196 \\ 0.4000 & -0.2000 & 0.2000 & -0.4000 & -0.0196 & 0.0200 & 0.0397 & -0.0405 \\ -0.4000 & 0.2000 & -0.2000 & 0.4000 & 0.0201 & -0.0196 & -0.0405 & 0.0397 \end{bmatrix}.$$

**Example 3.1.6.** In this example we want to show the application of the method in finding the solution of the singular linear system  $Ax = b$ , where  $b \in \mathcal{R}(A^k)$ ,  $k = \text{ind}(A)$ . In the monograph [7], it is shown that the Drazin inverse solution  $A^D b$  solves the singular linear system  $Ax = b$  if and only if  $b \in \mathcal{R}(A^k)$ . In addition,  $A^D b$  is the unique solution of  $Ax = b$  provided that  $x \in \mathcal{R}(A^k)$  [7]. By  $x^d$  and  $x^D$  we denote the approximation of the Drazin-inverse solution and the exact theoretical solution, respectively.

Consider

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & -1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 0 & 2 & -1 \\ -1 & -1 & 0 & -1 & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} -14 \\ 14 \\ -22 \\ 22 \\ 81 \\ -28 \end{bmatrix}.$$

After setting  $\gamma = 10^8$ ,  $k = \text{ind}(A) = 2$ , the neural network produces the following Drazin-

inverse solution in  $10^{-7}$  seconds:

$$x^d = A^d b = \begin{bmatrix} -7.0000 \\ 7.0000 \\ -10.9993 \\ 10.9993 \\ 40.9974 \\ 12.0026 \end{bmatrix}.$$

Theoretical solution is

$$x^D = A^D b = \begin{bmatrix} -7.0000 \\ 7.0000 \\ -11.0000 \\ 11.0000 \\ 41.0000 \\ 12.0000 \end{bmatrix}.$$

The estimation error given in terms of the norm  $\| \cdot \|$ , and computed in Matlab with built-in function `norm()` is  $\|A^D b - x^d\| = 0.0038$ .

**Example 3.1.7.** Consider the homogeneous differential equation from [8]:

$$Ax' + Bx = 0,$$

where it is assumed

$$A = \begin{bmatrix} 1 & 0 & -2 \\ -1 & 0 & 2 \\ 2 & 3 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 2 \\ -27 & -22 & -17 \\ 18 & 14 & 10 \end{bmatrix}.$$

Following calculations described in [8] and introducing  $\hat{A}$  and  $\hat{B}$ , where

$$\hat{A} = \frac{1}{3} \begin{bmatrix} -3 & -5 & -4 \\ 6 & 5 & -2 \\ -3 & 2 & 10 \end{bmatrix}, \quad \hat{B} = \frac{1}{3} \begin{bmatrix} 6 & 5 & 4 \\ -6 & -2 & 2 \\ 3 & -2 & 7 \end{bmatrix},$$

one can get the solution of the homogeneous differential equation in the form

$$x(t) = \exp(-\hat{A}^D \hat{B} t) x(0),$$

where  $x(0)$  is the initial vector of the differential equation.

For  $\gamma = 10^6$ ,  $k = \text{ind}(A) = 0$ , our neural network produces the following approximation

of the Drazin inverse of the matrix  $\hat{A}$ :

$$\hat{A}^d = \begin{bmatrix} -0.1068 & 0.1885 & -0.0717 \\ -0.1513 & 0.1663 & 0.0394 \\ -0.0889 & -0.0444 & 0.2222 \end{bmatrix},$$

so, the final result can be obtained:

$$x(t) = \exp(-\hat{A}^d \hat{B}t) x(0) = \exp\left(\begin{bmatrix} 0.6624 & 0.2559 & 0.1840 \\ 0.5957 & 0.3892 & -0.0012 \\ -0.1333 & 0.2667 & -0.3704 \end{bmatrix} t\right) x(0).$$

**Example 3.1.8.** The group inverse plays an important role in the theory of finite Markov chains [7, 56]. Let  $T$  be the one-step transition matrix of a finite homogeneous Markov chain reused from [84, Example 9]:

$$T = \frac{1}{4} \begin{bmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 2 & 0 \\ 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

To characterize the Markov chain [56] it is of importance to compute the group inverse of

$$A = I - T = \frac{1}{4} \begin{bmatrix} 4 & -2 & -2 & 0 \\ -2 & 4 & -2 & 0 \\ -2 & -1 & 4 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix},$$

wherein  $I$  denotes the identity matrix. The exact group inverse of  $A$  is given in [56]:

$$A^\# = \frac{2}{1083} \begin{bmatrix} 265 & -61 & -96 & -108 \\ -96 & 300 & -96 & -108 \\ -115 & -137 & 246 & 6 \\ -210 & -156 & -210 & 576 \end{bmatrix}.$$

Setting  $\gamma = 10^{10}$ , we obtain the resulting approximation of the group inverse after  $10^{-7}$  seconds.

$$A^g = \begin{bmatrix} 0.4893 & -0.1127 & -0.1773 & -0.1994 \\ -0.1773 & 0.5540 & -0.1773 & -0.1994 \\ -0.2123 & -0.2530 & 0.4543 & 0.0111 \\ -0.3878 & -0.2881 & -0.3878 & 1.0637 \end{bmatrix}$$

**Example 3.1.9.** *In order to compare the performance of the architecture proposed in the current section with the one presented in previous chapter [84], we firstly reuse the matrix  $A$  from Example 3.1.1. The output of the neural network from [84] is denoted by  $A_{[84]}^d$ ; by  $A^d$  we denote the output of the neural network produced in this paper; while  $A^D$  denotes the theoretical Drazin inverse. Finally, the matrix norm  $\|\cdot\|$  is computed by the Matlab function `norm()`. Both the neural networks give the solution after approximately  $10^{-7}$  seconds. The comparison in Table 3.1 is given in the sense of the accuracy of the solutions  $A^d$  and  $A_{[84]}^d$ , generated for the different values of the parameter  $\gamma$ . It is clear that more accurate solutions  $A^d$  with respect to  $A_{[84]}^d$  are reached for much smaller values of  $\gamma$ . This observation indicates a less computational complexity of the neural network used in [84] during the convergence of the neural network.*

Table 3.1: Numerical comparison test with [84] for Example 3.1.1

	$\gamma = 10^3$	$\gamma = 10^4$	$\gamma = 10^5$	$\gamma = 10^6$	$\gamma = 10^7$	$\gamma = 10^8$
$\ A_{[84]}^d - A^D\ $	0.5957	0.5886	0.5208	0.0917	$8.9562 \times 10^{-6}$	$1.4079 \times 10^{-5}$
$\ A^d - A^D\ $	0.0060	$1.0387 \times 10^{-5}$	$1.9609 \times 10^{-5}$	$3.7774 \times 10^{-13}$	$2.2236 \times 10^{-5}$	$4.7100 \times 10^{-6}$

Similarly, let us reuse the matrix  $A$  from the Example 3.1.2, and compare the previous two architectures in the same way. Table 3.2 shows the accuracy of the approximations  $A^d$  and  $A_{[84]}^d$  reached after  $10^{-7}$  seconds, for several different values of  $\gamma$ .

Table 3.2: Numerical comparison test with [84] for Example 3.1.2

	$\gamma = 10^5$	$\gamma = 10^6$	$\gamma = 10^7$	$\gamma = 10^8$	$\gamma = 10^9$	$\gamma = 10^{10}$
$\ A_{[84]}^d - A^D\ $	1.7357	1.5397	0.4122	$2.9217 \times 10^{-5}$	$8.1804 \times 10^{-5}$	$2.2366 \times 10^{-5}$
$\ A^d - A^D\ $	1.4463	0.9121	0.0419	$3.4623 \times 10^{-7}$	$4.9231 \times 10^{-7}$	$3.8940 \times 10^{-7}$

The data arranged in Table 3.2 lead to similar conclusions as the data arranged in Table 3.1. The network from [84] converges little bit slower, so that the corresponding architecture requires larger  $\gamma$  in order to produce a more accurate solution.

## 3.2 Globally convergent GNN for computing outer inverse

### 3.2.1 Preliminaries and motivation

We follow the same principles we applied to the Drazin inverse computation in the manner presented in the previous section [85]. We expand the main idea to the set of outer inverses with prescribed range and null space. Guided by this idea, our intention is to exploit the integral representation of outer inverses which is restated in Proposition 3.2.1. All the results presented here are derived in [138].

**Proposition 3.2.1.** ([52, 112]) *Suppose that  $A \in \mathbb{C}^{m \times n}$  and  $T$  and  $S$  are subspaces of  $\mathbb{C}^n$  and  $\mathbb{C}^m$ , respectively. Under the assumptions  $\mathcal{R}(G) = T, \mathcal{N}(G) = S$ , If  $A$  has the outer inverse  $A_{T,S}^{(2)}$ , then*

$$\begin{aligned} A_{T,S}^{(2)} &= \int_0^\infty \exp(-G(GAG)^T G A \tau) G(GAG)^T G d\tau \\ &= \int_0^\infty \exp(-G_0 A \tau) G_0 d\tau, \quad G_0 = G(GAG)^T G. \end{aligned} \quad (3.2.1)$$

Generally speaking, we follow the main leading idea which initiated the results from [112, 135]. As it was stated in [135], most of methods for computing outer inverses start from an appropriate matrix  $G$  which satisfies  $\mathcal{R}(G) = T, \mathcal{N}(G) = S$  and assume the condition  $\text{Re}(GA) \subset (0, +\infty)$  or  $\text{Re}(\sigma(GA)) > 0$ . The authors of the paper [135] established a unified representation theorem for the generalized inverse  $A_{T,S}^{(2)}$  which avoids the restriction on the nonnegativity of the spectrum of  $GA$ . Also, the integral representation given in (3.2.1) assumes  $\text{Re}(\sigma(GA)) > 0$ . Our intention is to avoid this restriction imposed on the spectrum of  $GA$ .

This goal can be achieved by replacing the appropriate matrix  $G$  from (2.2.2) or (2.2.3) by the matrix  $G_0 = G(GAG)^T G$ , used in (3.2.1). The drawback of this approach is the necessity to perform additional matrix multiplications in order to derive  $G_0$  from  $G$ . But, there is no other alternative to overcome very restrictive and ultimate conditions on the spectrum of  $GA$  or  $AG$ .

### 3.2.2 Neural Network Architecture

In the general case, it is possible that the set  $\sigma(GA)$  contains elements with negative real parts. Then the steady-state matrix of the recurrent neural network is generally not equal to  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$ . In spite of this, there is the possibility that the outer inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  exists in this case. In order to investigate the existence of outer inverse with prescribed range and null space we use the result derived in Proposition 3.2.2.

**Proposition 3.2.2.** *Let  $A \in \mathbb{C}_r^{m \times n}$  be given and  $G \in \mathbb{C}_s^{n \times m}$  be arbitrary matrix satisfying  $0 < s \leq r$ . The condition  $\text{ind}(AG) = \text{ind}(GA) = 1$  implies the existence of the outer inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}$ .*

*Proof.* Indeed, the assumption  $\text{ind}(AG) = \text{ind}(GA) = 1$  implies the existence of  $(AG)^\#$  and  $(GA)^\#$ . Further, using known result from [106], the existence of the outer inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} = G(AG)^\# = (GA)^\# G$  is ensured.  $\square$

Certainly, in the case when (2.2.14) is not satisfied and  $\text{ind}(AG) = \text{ind}(GA) = 1$  the generalized inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  can not be generated using RNN (2.2.16), in spite of its existence. However,  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  can be generated using the dynamic equation of the form

$$\begin{cases} \frac{dV_G(t)}{dt} = -\gamma (G_0 A V_G(t) - G_0), & V(0) = 0, \text{ if } m \geq n, \\ \frac{dV_G(t)}{dt} = -\gamma (V_G(t) A G_0 - G_0), & V(0) = 0, \text{ if } m < n, \end{cases} \quad (3.2.2)$$

where  $G_0 = G(GAG)^T G$ .

The connection weight matrix of the neurons in RNN is identical in each sub-network and defined by

$$W = \begin{cases} -\gamma G_0 A, & m \geq n \\ -\gamma A G_0, & m < n. \end{cases}$$

**Corollary 3.2.1.** *Let  $A \in \mathbb{R}^{m \times n}$  be given matrix,  $G \in \mathbb{R}_s^{n \times m}$  be arbitrary matrix satisfying  $0 < s \leq r$ . Then the limiting expression*

$$\overline{V}_G = \begin{cases} \lim_{t \rightarrow \infty} \gamma \exp(-\gamma G_0 A t) \int_0^t \exp(\gamma G_0 A \tau) G_0 d\tau, & m \geq n, \\ \lim_{t \rightarrow \infty} \gamma G \exp(-\gamma A G_0 t) \int_0^t \exp(\gamma A G_0 \tau) d\tau, & m < n \end{cases} \quad (3.2.3)$$

satisfies

$$\overline{V}_G = A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}. \quad (3.2.4)$$

*Proof.* Since  $\text{Re}(\sigma(G_0 A)) \geq 0$  is satisfied (see [52, 112]), an application of Theorem 2.2.1 gives  $\overline{V}_G = A_{\mathcal{R}(G_0), \mathcal{N}(G_0)}^{(2)}$ . The proof can be completed using  $\mathcal{R}(G_0) = \mathcal{R}(G)$  and  $\mathcal{N}(G_0) = \mathcal{N}(G)$  (see [135]).  $\square$

### 3.2.3 Illustrative Examples

**Example 3.2.1.** *Consider the initial matrix  $A$  equal to*

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & -1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 0 & 2 & -1 \\ -1 & -1 & 0 & -1 & -1 & 2 \end{bmatrix}.$$

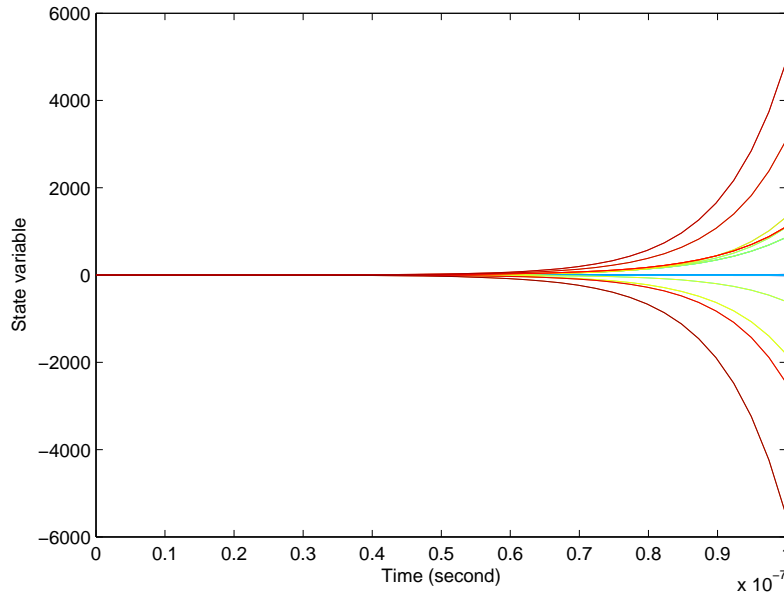
*In the case  $G = -A^T$  the RNN (2.2.16) does not provide a solution, since the matrix  $GA$  is negative semidefinite. Its eigenvalues are included in the set*

$$\sigma(GA) = \{-10.6056, -9.1231, -4.0000, -3.3944, -0.8769, 0.0000\}.$$

*The divergence of the neural network RNN (2.2.16) is illustrated in Figure 3.7.*

*Since  $\text{ind}(GA) = 1$ , according to Proposition 3.2.2,  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  exists. But, in spite of its existence,  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  cannot be computed using RNN (2.2.16). A possible solution is usage of the matrix  $G = (-A^T) \left( (-A^T) A (-A^T) \right)^T (-A^T) = A^T \left( A^T A A^T \right)^T A^T$  instead of the matrix  $G = -A^T$ . Such a choice initiates that the spectrum of  $GA$  contains elements with nonnegative*




 Figure 3.7: Divergence of the RNN in  $10^{-7}$  seconds for Example 3.2.1

*real parts:*

$$\sigma(GA) = 10^3 \{1.1929, 0.7593, 0.0640, 0.0391, 0.0007, 0.0000\}.$$

Using  $\gamma = 1000000000$  and  $G = A^T (A^T A A^T)^T A^T$  the RNN (3.2.2) approach produces the approximation of the Moore-Penrose inverse of  $A$ :

$$A^\dagger = \begin{bmatrix} 0.2500 & -0.2500 & -0.2500 & -0.2500 & 0.0000 & 0.0000 \\ -0.2500 & 0.2500 & -0.2500 & -0.2500 & 0.0000 & 0.0000 \\ 0 & 0 & 0.5000 & 0.0000 & -0.2500 & -0.2500 \\ 0 & 0 & 0.0000 & 0.5000 & -0.2500 & -0.2500 \\ 0 & 0 & -0.1667 & -0.3333 & 0.4167 & 0.0833 \\ 0 & 0 & -0.3333 & -0.1667 & 0.0833 & 0.4167 \end{bmatrix}.$$

**Example 3.2.2.** Consider following randomly generated matrix of rank 2:

$$A = \begin{bmatrix} -0.3891 & -0.5719 & -1.0486 & 0.0906 \\ 0.0507 & 0.0440 & -0.7345 & -0.5092 \\ -0.2421 & -0.3543 & -0.6068 & 0.0825 \\ -0.6085 & -0.8380 & -0.0271 & 1.0628 \\ 0.3589 & 0.4896 & -0.1159 & -0.7021 \end{bmatrix}.$$

The matrix  $G$  is generated using the full rank factorization  $G = PQ$ , where  $P$  and  $Q$  are

randomly generated matrices of rank 2:

$$P = \begin{bmatrix} 0.6948 & 0.4387 \\ 0.3171 & 0.3816 \\ 0.9502 & 0.7655 \\ 0.0344 & 0.7952 \end{bmatrix}, Q = \begin{bmatrix} 0.1869 & 0.4456 & 0.7094 & 0.2760 & 0.6551 \\ 0.4898 & 0.6463 & 0.7547 & 0.6797 & 0.1626 \end{bmatrix}.$$

The RNN (2.2.16) does not converge to the outer inverse, because there is one negative eigenvalue in the spectrum  $\sigma(GA)$ :

$$\sigma(GA) = \{-2.9910, 0.2889, 0.0, 0.0\}.$$

Again,  $\text{ind}(GA) = 1$  confirms the existence of the outer inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  initiated by the matrix  $G$ .

But, if we take  $G_0 = G(GAG)^T G$  and  $\gamma = 10^{11}$  the RNN (3.2.2) approach produces the following outer inverse of  $A$ :

$$X = \begin{bmatrix} -0.162712466533 & -0.193869486921 & -0.206592043421 & -0.224055760876 & 0.008074234858 \\ -0.038944020388 & -0.133747729506 & -0.234434903367 & -0.060929232708 & -0.258398656300 \\ -0.190291830114 & -0.306447329758 & -0.410440140316 & -0.268697936905 & -0.228150378169 \\ 0.142546400482 & -0.202636669798 & -0.607873904204 & 0.165143602730 & -1.117224217594 \end{bmatrix}.$$

The exact outer inverse corresponding to  $G = PQ$  is equal to

$$\begin{aligned} A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} &= P(QAP)^{-1}Q \\ &= \begin{bmatrix} -0.162712645811 & -0.193869773308 & -0.206592425100 & -0.224056013867 & 0.008074026949 \\ -0.038944133693 & -0.133747910475 & -0.234435144552 & -0.060929392570 & -0.258398787674 \\ -0.190292104036 & -0.306447767247 & -0.410440723476 & -0.268698323377 & -0.228150695786 \\ 0.142546257394 & -0.202636898362 & -0.607874208808 & 0.165143400823 & -1.117224383521 \end{bmatrix}. \end{aligned}$$

**Example 3.2.3.** Consider following randomly generated matrix of rank 4:

$$A = \begin{bmatrix} 1.2510 & -1.4505 & 1.6163 & -0.3286 & -0.7332 & -0.2914 \\ -1.2983 & -0.1638 & -1.3136 & -0.2843 & -0.2447 & 1.3777 \\ -1.7206 & 0.1765 & -1.5799 & 0.3005 & 0.7192 & 0.5433 \\ 0.7153 & -0.1860 & 0.9885 & 0.7135 & -0.4896 & -0.2466 \\ -0.7785 & -1.1098 & -0.1949 & 0.3593 & -0.0978 & 0.5137 \\ -0.9282 & -0.8967 & -0.2734 & 0.6571 & 0.5276 & -0.1082 \\ 0.2117 & -0.1662 & 0.3192 & 0.0994 & 0.1811 & -0.4174 \end{bmatrix}.$$

Now, randomly generated matrices  $P$  and  $Q$  of rank 4 are used:

$$P = \begin{bmatrix} 0.3102 & 0.1705 & 0.7818 & 0.6814 \\ -0.6748 & -0.5524 & 0.9186 & -0.4914 \\ -0.7620 & 0.5025 & 0.0944 & 0.6286 \\ -0.0033 & -0.4898 & -0.7228 & -0.5130 \\ 0.9195 & 0.0119 & -0.7014 & 0.8585 \\ -0.3192 & 0.3982 & -0.4850 & -0.3000 \end{bmatrix},$$

$$Q = \begin{bmatrix} -0.6068 & -0.2967 & 0.8344 & -0.2391 & 0.0616 & 0.1376 & -0.6756 \\ -0.4978 & 0.6617 & -0.4283 & 0.1356 & 0.5583 & -0.0612 & 0.5886 \\ 0.2321 & 0.1705 & 0.5144 & -0.8483 & 0.8680 & -0.9762 & -0.3776 \\ -0.0534 & 0.0994 & 0.5075 & -0.8921 & -0.7402 & -0.3258 & 0.0571 \end{bmatrix}.$$

For the choice of  $G = PQ$ , the RNN (2.2.16) does not converge to the outer inverse, because there are three eigenvalues in the spectrum  $\sigma(GA)$  with negative real parts:

$$\sigma(GA) = \{-0.6491 - 1.2262i, -0.6491 + 1.2262i, -1.1737, 1.1509, 0.0, 0.0\}.$$

On the other hand, usage of the matrix  $G_0 = G(GAG)^T G$  and the scaling parameter  $\gamma = 10^{11}$  in the RNN (3.2.2) approach produces the following approximation of  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$ :

$$X = \begin{bmatrix} 0.7418226 & -0.8130738 & 0.9380724 & -0.6689539 & -0.179301 & -0.473387 & -1.080997 \\ -1.200862 & 1.709335 & -1.782507 & 0.6594245 & -0.8578323 & 0.7906579 & 2.5548255 \\ -0.5638703 & 0.7777778 & -1.2338862 & 0.9416353 & 0.0575493 & 0.6281124 & 1.2348809 \\ -0.5744556 & 0.3232869 & -0.4726930 & 0.5609873 & -0.0340408 & 0.5260483 & 0.5112868 \\ 1.1415376 & -2.1082814 & 1.9963191 & -0.4710172 & 0.3702859 & -0.3844626 & -2.8415981 \\ -0.2089968 & 0.4880962 & -0.4971972 & 0.2283032 & 0.4464316 & -0.0373900 & 0.5130587 \end{bmatrix}.$$

The exact outer inverse corresponding to  $G = PQ$  is equal to

$$A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)} = P(QAP)^{-1}Q$$

$$= \begin{bmatrix} 0.7418274 & -0.8130737 & 0.9380642 & -0.6689465 & -0.1792986 & -0.4733854 & -1.0809940 \\ -1.2008608 & 1.7093355 & -1.7825098 & 0.6594270 & -0.8578313 & 0.7906585 & 2.5548266 \\ -0.5638670 & 0.7777779 & -1.2338919 & 0.9416405 & 0.0575511 & 0.6281137 & 1.2348833 \\ -0.5744604 & 0.3232867 & -0.4726849 & 0.5609799 & -0.0340436 & 0.5260464 & 0.5112835 \\ 1.1415368 & -2.1082815 & 1.9963204 & -0.4710184 & 0.3702855 & -0.3844629 & -2.8415986 \\ -0.2089987 & 0.4880962 & -0.4971941 & 0.2283003 & 0.4464306 & -0.0373907 & 0.5130574 \end{bmatrix}.$$

**Remark 3.2.1.** Usage of the matrix product  $G_0 = G(GAG)^T G$  sometimes causes numerical instability. But, despite this disadvantage, the gradient based RNN (3.2.2) and the dynamic state equation based on  $G_0$  offer a solution of the problem and is capable to generate  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$ . The solution has its own drawbacks, it is charged by an increased number of ma-

trix multiplications, but in some way overcomes the problem and generates an approximation of  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$ .

### 3.3 Globally convergent GNN for computing $W$ -weighted Drazin inverse

In this section, we consider the case of  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$  without the necessity to find positive integer  $l$  such that  $s((AW)^{l+2}) \geq 0$ .

#### 3.3.1 Dynamic equation with global convergence

If  $A$  is a rectangular matrix, Wei [113], Liu and Zhong [53] derived an integral representation of the  $W$ -weighted Drazin inverse  $A_{d,w}$  of the matrix  $A \in \mathbb{C}^{m \times n}$ , without any restriction on the eigenvalues of the matrix  $(AW)^{l+2}$ ,  $l \geq k = \max\{\text{ind}(AW), \text{ind}(WA)\}$ .

**Theorem 3.3.1.** ([53]) *Assume that  $A \in \mathbb{C}^{m \times n}$ ,  $W \in \mathbb{C}^{n \times m}$  and  $k = \max\{\text{Ind}(AW), \text{Ind}(WA)\}$ . Then*

$$A_{d,w} = \int_0^\infty \exp\left(- (AW)^l A [(AW)^{2l+2} A]^* (AW)^{l+2} \tau\right) (AW)^l A [(AW)^{2l+2} A]^* (AW)^l A d\tau, \quad (3.3.1)$$

where  $l \geq k$ .

However, the dynamical state equation of a recurrent neural network which corresponds to the integral representation (3.3.1), to the best of our knowledge, has not been developed. In the following, we prove that the  $W$ -weighted Drazin inverse of a rectangular matrix can be determined by using a recurrent neural network that presented in [85]. Our investigation is restricted to real matrices, i.e., it is assumed that  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$ . For the sake of simplicity, we use the notations

$$G = (AW)^l A, \quad G_0 = G(GWAWG)^T G = (AW)^l A [(AW)^{2l+2} A]^T (AW)^l A.$$

**Lemma 3.3.1.** *The  $W$ -weighted Drazin inverse of  $A \in \mathbb{R}^{m \times n}$  satisfies*

$$(AW)^l A [(AW)^{2l+2} A]^T (AW)^{l+2} A_{d,w} - (AW)^l A [(AW)^{2l+2} A]^T (AW)^l A = 0 \quad (3.3.2)$$

*Proof.* On the basis of  $\mathcal{R}(G_0) = \mathcal{R}(G)$  and  $\mathcal{N}(G_0) = \mathcal{N}(G)$ , the following matrix equation is valid

$$G_0 W A W A_{d,w} - G_0 = 0, \quad (3.3.3)$$

which implies (3.3.2).  $\square$

According to (3.3.3), the following matrix equation with respect to unknown matrix  $V$  can be considered:

$$G_0 W A W V - G_0 = 0. \quad (3.3.4)$$

The scalar-valued norm-based error function corresponding to equation (3.3.4) is defined as

$$E(t) = \frac{\|G_0 W A W V(t) - G_0\|_F^2}{2}, \quad l \geq k.$$

Note that the minimal value  $E(\bar{t}) = 0$  of the residual-error function  $E(t)$  is achieved in a minimizer  $\bar{V} = V(\bar{t})$  if and only if  $V(\bar{t})$  is the exact solution of (3.3.4). A computational scheme could be designed to evolve along a descent direction of this error function  $E(t)$ , until the minimum  $E(\bar{t})$  is reached. The typical descent direction of  $E(t)$  is defined by the negative gradient  $-\partial E(t)/\partial V$  of  $E(t)$ . The gradient of  $E$  with respect to  $V \in \mathbb{R}^{m \times m}$  is equal to

$$\frac{\partial E(t)}{\partial V} = (G_0 W A W)^T (G_0 W A W V(t) - G_0). \quad (3.3.5)$$

According to the design formula

$$\frac{dV(t)}{dt} = -\gamma \frac{\partial E(t)}{\partial V},$$

and by omitting the constant term  $(G_0 W A W)^T$ , it is reasonable to define the dynamical equation of a gradient recurrent neural network as follows:

$$\begin{aligned} \frac{dV(t)}{dt} &= -\gamma (G_0 W A W V(t) - G_0), \\ G_0 &= (A W)^l A [(A W)^{2l+2} A]^T (A W)^l A, \quad l \geq k, \quad V(0) = V_0. \end{aligned} \quad (3.3.6)$$

The model (3.3.6) will be denoted by GNNDW0. On the other hand, the original GNN design for computing  $W$ -weighted Drazin inverse can be defined as

$$\begin{aligned} \frac{dV(t)}{dt} &= -\gamma (G_0)^T (G_0 W A W V(t) - G_0), \\ G_0 &= (A W)^l A [(A W)^{2l+2} A]^T (A W)^l A, \quad l \geq k, \quad V(0) = V_0. \end{aligned} \quad (3.3.7)$$

The recurrent neural network defined above is a linear dynamical system in a matrix form. According to the linear systems theory [35], the closed-form solution of the state matrix can be defined as

$$\begin{aligned} V(t) &= \exp(-\gamma G_0 W A W t) V(0) + \\ &+ \gamma \exp(-\gamma G_0 W A W t) \int_0^t \exp(\gamma G_0 W A W \tau) G_0 d\tau. \end{aligned} \quad (3.3.8)$$

### 3.3.2 Convergence and stability analysis of GNNDW0

To analyze the convergence and stability of a neural network, the following lemma is needed in what follows.

**Lemma 3.3.2.** *For arbitrary rectangular complex matrices  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$ , the matrix  $G_0 W A W = (A W)^l A [(A W)^{2l+2} A]^T (A W)^{l+2}$ ,  $l \geq k$ , possesses the eigenvalues  $\lambda_i$  satisfying  $s(G_0 W A W) \geq 0$ .*

*Proof.* See the proof of Theorem 2.3 in [53].  $\square$

**Corollary 3.3.1.** *For any  $A \in \mathbb{R}^{m \times n}$ ,  $W \in \mathbb{R}^{n \times m}$ ,  $l \geq k$  and  $t \in [0, +\infty]$ , the following holds:*

$$\lim_{t \rightarrow \infty} \exp(-\gamma G_0 W A W t) = 0. \quad (3.3.9)$$

The result of Corollary 3.3.1 is very important and shows that (3.3.9) holds for arbitrary  $V(0)$  as well as for arbitrary matrices  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$ .

Now, (3.3.8) and (3.3.9) imply the following representation for  $\lim_{t \rightarrow \infty} V(t) = \bar{V}$ ,

$$\lim_{t \rightarrow \infty} V(t) = \bar{V} = \lim_{t \rightarrow \infty} \gamma \exp(-\gamma G_0 W A W t) \int_0^t \exp(\gamma G_0 W A W \tau) G_0 d\tau. \quad (3.3.10)$$

Based on the definition and properties of the matrix exponential, it can be verified that the closed-form solution of  $\bar{V}$  in (3.3.10) represents the  $W$ -weighted Drazin inverse  $A_{d,w}$ , and it is independent of  $\gamma$ . Corresponding statement is given in Theorem 5.5.2.

**Theorem 3.3.2.** *Let  $A \in \mathbb{R}^{m \times n}$  and  $W \in \mathbb{R}^{n \times m}$  with  $l \geq k$  and  $t \in [0, +\infty]$ . Then the limiting expression (3.3.10) produces the  $W$ -weighted Drazin inverse  $A_{d,w}$ , i.e.,  $\bar{V} = A_{d,w}$ .*

*Proof.* The proof is similar as the proof of Theorem 2.3.1 and based on the usage of the matrix

$$G_0 = (A W)^l A [(A W)^{2l+2} A]^T (A W)^l A$$

instead of the matrix  $G = (A W)^l A$ .  $\square$

If  $A$  is a square and  $W = I$ , then the results corresponding to the  $W$ -weighted Drazin inverse  $A_{d,w}$  reduces to analogous results initiating the Drazin inverse  $A^D$ . Indeed, the dynamic equation of a gradient RNN (3.3.6) recasts to the dynamic equation

$$\frac{dV(t)}{dt} = -\gamma (G_0 A V(t) - G_0), \quad G_0 = A^{l+1} (A^{2(l+1)+1})^T A^{l+1}, \quad l \geq k, \quad V(0) = V_0,$$

which is of the same effect as the dynamic equation (3.5) proposed in [85]. Simply, a replacement of  $k = \text{Ind}(A)$  by  $l + 1, l \geq k$  is done, which is allowed. Corresponding convergence result can be stated immediately in Corollary 3.3.2.

**Corollary 3.3.2.** *Let  $A \in \mathbb{R}^{n \times n}$ ,  $W = I$  and  $l \geq \text{Ind}(A)$ . Then the limiting expression (3.3.10) produces the Drazin inverse  $A^D$ , i.e.,  $\bar{V} = A^D$ .*

Clearly, (3.3.8) can be simplified by forcing the first matrix term in its right-hand side to be the zero matrix, i.e., by establishing zero initial states. In light of the above discussion and according to (3.3.6), the dynamic state equation of the RNN for computing the  $W$ -weighted Drazin inverse can be stated using the zero initialization in (3.3.6):

$$\begin{aligned} \frac{dV(t)}{dt} &= -\gamma (G_0 W A W V(t) - G_0), \\ G_0 &= (AW)^l A [(AW)^{2l+2} A]^T (AW)^l A, \quad l \geq k, \quad V(0) = 0. \end{aligned} \quad (3.3.11)$$

The stability of the equilibrium state  $\bar{V}$  is investigated in Theorem 3.3.3.

**Theorem 3.3.3.** *The gradient-based neural network given in equation (3.3.11) is stable in the sense of Lyapunov.*

*Proof.* The proof is similar to the proof of Theorem 2.3.2. It is necessary to replace  $G = (AW)^l A$  by  $G_0 = G(GAG)^T G$  and consider the following Lyapunov function candidate:

$$\begin{aligned} E(t) &= \frac{\|G_0 W A W V - G_0\|_F^2}{2} \\ &= \frac{\text{Tr}[(G_0 W A W V - G_0)^T (G_0 W A W V - G_0)]}{2}. \end{aligned}$$

□

If  $A \in \mathbb{R}^{n \times n}$  and  $W = I$ , then the  $W$ -weighted Drazin inverse  $A_{d,w}$  reduces to the Drazin inverse  $A^D$ . We also obtain the stability of the equilibrium state  $\bar{V}$  as follows.

**Corollary 3.3.3.** *Let  $A \in \mathbb{R}^{n \times n}$ ,  $W = I$  and  $l \geq \text{Ind}(A)$ . Then the gradient-based neural network (3.3.11) is stable in the sense of Lyapunov.*

### Neural networks architecture of GNNDW0

Since the positive real scaling constant  $\gamma$  multiplies the time parameter  $t$ , its value could be as large as possible. Greater values of  $\gamma$  will speed up the computation, because the terms

$$\lim_{t \rightarrow \infty} \exp[-\gamma (AW)^{l+2} t] = \lim_{t \rightarrow \infty} \exp[-\gamma G W A W t], \quad G = (AW)^l A, \quad l \geq k$$

or

$$\lim_{t \rightarrow \infty} \exp[-\gamma G_0 W A W t], \quad G_0 = (AW)^l A [(AW)^{2l+2} A]^T (AW)^l A, \quad l \geq k$$

vanish faster for larger  $\gamma$ .

Similarly with the RNNs for computing the Drazin inverse which are defined in [85, 84], our neural network is composed from a number of independent subnetworks. Each subnetwork represents a column vector of  $V(t)$ . Denote by  $v_j(t)$  (respectively, by  $h_j, g_j, (G_0)_j$ ) the  $j$ th column vector of  $V(t)$  (respectively,  $(AW)^l A, G, G_0$ ), for  $j = 1, 2, \dots, n$ . The dynamics of the  $j$ th subnetwork corresponding specific case can be expressed as

$$\frac{dv_j(t)}{dt} = -\gamma \left( (AW)^{l+2} v_j(t) - h_j \right). \quad (3.3.1)$$

In the general case, the  $j$ th subnetwork can be stated as

$$\frac{dv_j(t)}{dt} = -\gamma (G_0 W A W v_j(t) - (G_0)_j), \quad G_0 = (AW)^l A [(AW)^{2l+2} A]^T (AW)^l A. \quad (3.3.2)$$



# Chapter 4

## GNN for computing outer inverses based on the full rank representation

*In this chapter, we study numerical computation of outer inverses with prescribed range and null space using dynamic equations and corresponding gradient based recurrent neural networks. The defined neural networks are aimed to computation of the usual inverse or the group inverse which are involved in matrix products representing two general representations of outer inverses with known range and null space. The explicit closed-form solutions of defined dynamic equations are derived. Similarly with the recurrent neural networks for matrix inversion developed earlier, proposed neural networks are composed of a number of independent sub-networks. Each sub-network corresponds to the column or row of the inverse matrix. Numerical performances and stability of the proposed neural networks are demonstrated and compared by means of several numerical examples.*

### 4.1 Preliminaries and motivation

The starting point of our investigations is the dynamic state equation of the RNN for computing outer inverse which was introduced in [138]:

$$\begin{cases} \frac{dV(t)}{dt} = -\gamma (GAV(t) - G), & V(0) = 0, \text{ if } m \geq n, \\ \frac{dV(t)}{dt} = -\gamma (V(t)AG - G), & V(0) = 0, \text{ if } m < n. \end{cases} \quad (4.1.1)$$

The GNN evolution (4.1.1) is termed as GNNGA or GNNAG. The main advantage of the dynamic equation (4.1.1) is its universality. More precisely, (4.1.1) comprises all dynamic equations proposed for numerical computation of generalized inverses.

**A)** In the case  $G = A^T$ , the dynamic equation (4.1.1) reduces to the dynamic equation of the linear recurrent neural network for the usual matrix inversion, proposed in [94]:

$$\frac{dV(t)}{dt} = -\gamma A^T A V(t) + \gamma A^T, \quad V(0) = V_0, \quad (4.1.2)$$

wherein  $V(t)$  is a matrix of activation state variables corresponding to the inverse matrix of  $A$  and  $\gamma$  is a positive scaling constant. As it is proved in [94], the RNN defined on the basis of (4.1.2) is asymptotically stable in the sense of Lyapunov and its steady-state matrix is equal to  $A^{-1}$ .

**B)** By allowing the activation state matrix to be rectangular, recurrent neural network defined in (4.1.2) can be used for computing the Moore-Penrose inverse of a full-column rectangular matrix  $A \in \mathbb{R}_n^{m \times n}$ . In the dual full-row case  $A \in \mathbb{R}_m^{m \times n}$ , the dual GNN model

$$\frac{dV(t)}{dt} = -\gamma V(t) A A^T + \gamma A^T, \quad V(0) = V_0 \quad (4.1.3)$$

can also be used to compute the Moore-Penrose inverse of  $A$ .

**C)** Three recurrent neural networks for computing the Moore-Penrose inverse of rank-deficient matrices were proposed in [96]. The first RNN is most similar to our approach, and it exploits the dynamic equation

$$\frac{dV(t)}{dt} = \begin{cases} -M A^T A V(t) + M A^T, & V(0) = 0, \quad m \geq n, \\ -V(t) A A^T M + A^T M, & V(0) = 0, \quad m < n, \end{cases} \quad (4.1.4)$$

where  $M$  is a positive diagonal matrix satisfying  $M \in \mathbb{R}^{n \times n}$  in the case  $m \geq n$  and  $M \in \mathbb{R}^{m \times m}$  in the case  $m < n$ . The dynamic equation presented in (4.1.4) can be derived from (4.1.1) in particular cases  $G = M A^T$  or  $G = A^T M$ .

**D)** Another particular case of (4.1.1) leads to RNN approach in computing the weighted Moore-Penrose inverse. Wei in [108] introduced the following dynamic state equation of the first recurrent neural network (called *NN1*) for computing the weighted Moore-Penrose inverses of a rank-deficient matrix  $A$ :

$$\frac{dV(t)}{dt} = \begin{cases} -D A^\# A V(t) + D A^\#, & V(0) = 0, \quad m \geq n, \\ -V(t) A A^\# D + A^\# D, & V(0) = 0, \quad m < n, \end{cases} \quad (4.1.5)$$

where  $D$  is a positive diagonal matrix of appropriate dimensions and  $A^\# = N^{-1} A^T M$  ( $M$  and  $N$  are chosen positive definite matrices). The simplest choice for  $D$  is  $D = \gamma I$ , where  $\gamma > 0$  [108].

**E)** The isolated case  $G = A^m, m \geq \text{ind}(A)$  in (4.1.1) leads to the dynamic equation of a gradient recurrent neural network for computing the Drazin inverse of a square matrix  $A$ , which was defined in [84]:

$$\frac{dV(t)}{dt} = -\gamma (A^{m+1}V(t) - A^m), \quad m \geq \text{ind}(A), \quad V(0) = V_0. \quad (4.1.6)$$

The exponent  $m$  in (4.1.6) is the smallest integer which assures nonnegative real parts for all eigenvalues of the matrix  $A^{m+1}$ . An alternative dynamic, which avoids the necessity to avoid the computation of an appropriate exponent  $m$ , was defined in [85]. This dynamic is based on the usage of the matrix  $G_0 = A^k(A^k A A^k)^T A^k, k = \text{ind}(A)$ , instead of the matrix  $G = A^m, m \geq \text{ind}(A)$ .

But, the dynamics defined in (4.1.1) has a serious drawback. Namely, the matrix  $G$  must be chosen such that exactly one of the next two conditions is satisfied:

$$\sigma(GA) \subset \{z : \text{Re}(z) \geq 0\}, \quad m \geq n, \quad (4.1.7)$$

$$\sigma(AG) \subset \{z : \text{Re}(z) \geq 0\}, \quad m < n. \quad (4.1.8)$$

The evolution rule (4.1.1), used in [138], fails in the case when some of real parts in the spectrum  $\sigma(GA)$  are negative. If one of the conditions (4.1.7) or (4.1.8) is not satisfied, the generalized inverse  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$  could not be generated on the basis of (4.1.1), despite of his existence. One possible solution is proposed in [138]; it is based on the replacement of the matrix  $G$  by  $G_0 = G(GAG)^T G$  in (4.1.1). The replacement of  $G$  by  $G_0$  eliminates the necessity to fulfill the constraints (4.1.7) or (4.1.8), since  $\sigma(G_0 A)$  satisfies (4.1.7) and  $\sigma(A G_0)$  satisfies (4.1.8). But, the goal is achieved by a relatively large increase of the number of expensive matrix operations. Additionally, the numbers in  $G_0$  grow, in some cases, which could cause numerical instability. Finally, our numerical experience shows that real parts of eigenvalues contained in  $\sigma(G_0 A)$  or  $\sigma(A G_0)$  are sometimes very small negative integers, which later causes numerical instability, or even divergence.

Two additional possibilities to overcome the problem were proposed in [86]. In this chapter we present these possibilities. Derived simulations of RNN are based on two different representations of outer inverses. Equivalence between these representations is investigated in Section 4.2. Two dynamic equations arising from these representations and initiated RNNs are considered in Section 4.3.

## 4.2 On the existence and representations of outer inverses

We firstly show that Proposition 4.2.2 and Proposition 4.2.3 produce the same outer inverse with prescribed range and null space. The next auxiliary result from [2] is used: if in the matrix product  $A = FHG$  the matrix  $F$  is of full-column rank and  $G$  is of full-row rank, then  $\text{rank}(A) = \text{rank}(H)$ . Also, the representation of the group inverse, proposed in Proposition 4.2.1, is very important.

**Proposition 4.2.1.** [18] *A square matrix  $A$  determined by the full-rank representation  $A = BC$  has a group inverse if and only if  $CB$  is nonsingular, in which case*

$$A^\# = B(CB)^{-2}C.$$

A useful representation of  $A_{T,S}^{(2)}$ , based on the usage of the group inverse, is presented in [106]. This representation, restated in the next proposition, gave a new computational aspect in relationship with the  $A_{T,S}^{(2)}$  inverse.

**Proposition 4.2.2.** [106] *Let  $A \in \mathbb{C}^{m \times n}$  be of rank  $r$ , let  $T$  be a subspace of  $\mathbb{C}^n$  of dimension  $s \leq r$ , and let  $S$  be a subspace of  $\mathbb{C}^m$  of dimension  $m - s$ . In addition, suppose that  $G \in \mathbb{C}^{n \times m}$  satisfies  $\mathcal{R}(G) = T$  and  $\mathcal{N}(G) = S$ . If  $A$  has  $A_{T,S}^{(2)}$  then  $\text{ind}(AG) = \text{ind}(GA) = 1$  and*

$$A_{T,S}^{(2)} = G(AG)^\# = (GA)^\#G.$$

Full-rank representation of  $\{2\}$ -inverses with prescribed range and null space is determined in the next proposition, which originated in [12].

**Proposition 4.2.3.** [12] *Let  $A \in \mathbb{C}_r^{m \times n}$ ,  $T$  be a subspace of  $\mathbb{C}^n$  of dimension  $s \leq r$  and let  $S$  be a subspace of  $\mathbb{C}^m$  of dimensions  $m - s$ . In addition, suppose that  $G \in \mathbb{C}^{n \times m}$  satisfies  $\mathcal{R}(G) = T, \mathcal{N}(G) = S$ . Let  $G$  has an arbitrary full-rank decomposition, that is  $G = PQ$ . If  $A$  has a  $\{2\}$ -inverse  $A_{T,S}^{(2)}$ , then:*

- (1)  $QAP$  is an invertible matrix;
- (2)  $A_{T,S}^{(2)} = P(QAP)^{-1}Q$ .

Representation of outer inverses in the general form  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} = G(AG)^\# = (GA)^\#G$  has been widely exploited in scientific literature. This representation is used in deriving the determinantal representation of generalized inverses [6, 121] as well as in the construction of iterative methods for their computation [47, 115, 116, 121]. Also, the full rank representation  $A_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)} = P(QAP)^{-1}Q$  has been frequently applied in many numerical calculations. For example, such a representation has been exploited to define the determinantal representation of  $A_{T,S}^{(2)}$  inverse in [12]. Also, this representation has been used in the construction of the general

successive matrix squaring algorithm for computing  $A_{T,S}^{(2)}$  [76] or in the block representation of the set  $A\{2\}_s$  [73].

Our first observation is that Proposition 4.2.2 should be stated as an "if and only if" statement. Similar situation appears in Proposition 4.2.3. These results are verified in Lemma 4.2.1. Also, Lemma 4.2.1 gives an unified aspect to these, so far, separate results.

**Lemma 4.2.1.** *Let  $A \in \mathbb{C}^{m \times n}$  be of rank  $r$ , let  $T$  be a subspace of  $\mathbb{C}^n$  of dimension  $s \leq r$ , and let  $S$  be a subspace of  $\mathbb{C}^m$  of dimension  $m - s$ . In addition, suppose that  $G \in \mathbb{C}_s^{n \times m}$  satisfies  $\mathcal{R}(G) = T$  and  $\mathcal{N}(G) = S$ . Assume that  $G = PQ$  is a full rank factorization of  $G$ . Then  $A$  has  $A_{T,S}^{(2)}$  if and only if one of the following conditions hold:*

$$\text{ind}(AG) = \text{ind}(GA) = 1, \quad (4.2.1)$$

$$QAP \text{ is invertible.} \quad (4.2.2)$$

*In this case*

$$A_{T,S}^{(2)} = G(AG)^\# = (GA)^\#G \quad (4.2.3)$$

$$= P(QAP)^{-1}Q. \quad (4.2.4)$$

*Proof.* Proposition 4.2.2 claims the following: If  $A$  has  $A_{T,S}^{(2)}$  then  $\text{ind}(AG) = \text{ind}(GA) = 1$  and  $A_{T,S}^{(2)} = G(AG)^\# = (GA)^\#G$ .

But, the opposite case also holds. Namely, does  $\text{ind}(AG) = \text{ind}(GA) = 1$  ensures the existence of  $A_{\mathcal{R}(G),\mathcal{N}(G)}^{(2)}$ . Indeed,  $\text{ind}(AG) = \text{ind}(GA) = 1$  implies the existence of  $(AG)^\#$  and  $(GA)^\#$ . Further, the existence of  $A_{\mathcal{R}(G),\mathcal{N}(G)}^{(2)} = G(AG)^\# = (GA)^\#G$  is ensured. Therefore, we just verified that  $A_{\mathcal{R}(G),\mathcal{N}(G)}^{(2)}$  exists if and only if (4.2.1) is satisfied.

Now, it is necessary to verify equivalence between (4.2.1) and (4.2.2). We use that  $AG = (AP)Q$  is a full rank factorization of  $AG$  [12]. Since the  $(AG)^\#$  exists, according to full-rank representation of the group inverse from [18], immediately follows that  $Q(AP)$  is invertible (see, [12]).

It suffices to prove identity between the expressions (4.2.3) and (4.2.4). Concerning invertibility of  $QAP$ , the following transformation is allowed:

$$P(QAP)^{-1}Q = PQAP(QAP)^{-2}Q.$$

Using the general representation of the group inverse from Proposition 4.2.1 we have

$$(AG)^\# = AP(QAP)^{-2}Q$$

which implies

$$P(QAP)^{-1}Q = G(AG)^{\#}.$$

Similarly, using  $GA = P(QA)$  as a full rank factorization of  $GA$ , one can verify

$$P(QAP)^{-1}Q = P(QAP)^{-2}QAPQ = (GA)^{\#}G,$$

which completes the proof.  $\square$

### 4.3 Neural networks based on full rank representation of outer inverses

In order to simplify notations, by RNN(4.2.3) and RNN(4.2.4) we denote the RNNs which can be used in the computation of the representation (4.2.3) and (4.2.4), respectively. RNN(4.2.3) is designed for calculating the group inverse  $(AG)^{\#}$  or  $(GA)^{\#}$  in accordance with the results proposed in [84]. RNN(4.2.4) is defined in order to calculate  $(QAP)^{-1}$ . The main reason for this choice are the relatively small dimensions  $s \times s$  of the matrix  $QAP$  with respect to dimensions  $m \times m$  or  $n \times n$  of matrices  $AG$  or  $GA$ , respectively. Application of RNN(4.2.4) to the invertible matrix of small dimensions indicates fast numerical computation and global stability.

#### 4.3.1 Neural network RNN(4.2.4) based on (4.2.4)

The following facts are assumed in the rest of this chapter. The matrix  $A \in \mathbb{R}_r^{m \times n}$  is given;  $G = PQ$  is a full-rank factorization of a selected matrix  $G \in \mathbb{R}_s^{n \times m}$ , where the integer  $s$  satisfying  $0 < s \leq r$  is the rank of  $G$ . Under the assumption that  $QAP$  is invertible, the following full rank representation holds:

$$A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)} = P(QAP)^{-1}Q. \quad (4.3.1)$$

The RNN for computing  $(QAP)^{-1}$  is generated using the initial matrix equation

$$QAPV - I = 0, \quad (4.3.2)$$

where  $V \in \mathbb{R}^{n \times m}$  represents the unknown matrix which should generate  $(QAP)^{-1}$ . A scalar-valued norm based error function

$$E(t) = \frac{\|QAPV(t) - I_s\|_F^2}{2} \quad (4.3.3)$$

can be used to solve (4.3.2) via dynamic-system approach. The derivative of  $E(t)$  with respect

to  $V \in \mathbb{G}^{n \times m}$  can be derived using [25, Chapter 5]:

$$\frac{\partial E(t)}{\partial V} = (QAP)^T (QAPV(t) - I_s). \quad (4.3.4)$$

This approach leads to the dynamic equation of the form which was used in the complex-valued gradient neural network from [122, 125, 129]:

$$\dot{V}(t) = -\gamma \frac{\partial E(t)}{\partial V}. \quad (4.3.5)$$

Therefore, it is reasonable to define the dynamic equation of a recurrent neural network as follows

$$\frac{dV(t)}{dt} = -\gamma \left( (QAP)^T QAPV(t) - (QAP)^T \right), \quad V(0) = V_0, \quad (4.3.6)$$

where  $V(t) \in \mathbb{R}^{s \times m}$  is a matrix of activation state variables and  $\gamma$  is a positive scaling constant.

The closed-form solution of (4.3.6) can be derived using the linear systems theory (see, for example [35]):

$$\begin{aligned} V(t) = & \exp\left(-\gamma(QAP)^T QAPt\right) V(0) + \\ & + \gamma \exp\left(-\gamma(QAP)^T QAPt\right) \int_0^t \exp(\gamma QAP\tau) d\tau (QAP)^T. \end{aligned} \quad (4.3.7)$$

The matrix  $QAP$  is positive definite (according to assumptions), which means it has all eigenvalues real and positive. Also, since  $\gamma > 0$ , all eigenvalues of  $-\gamma QAP$  are real and negative, and according to the linear dynamic system theory, the recurrent neural network is globally asymptotically stable. Therefore, the term  $\exp(-\gamma QAPt)V(0)$  in the right-hand side of (4.3.7) vanishes to the zero matrix of the same size as time approaches infinity, i.e.,

$$\lim_{t \rightarrow \infty} \exp\left(-\gamma(QAP)^T QAPt\right) V(0) = 0, \quad (4.3.8)$$

regardless of the initial states  $V(0)$ . Now, equations (4.3.7) and (4.3.8) imply the following representation of  $\lim_{t \rightarrow \infty} V(t) = \bar{V}$ , for arbitrary  $V(0)$ :

$$\bar{V} = \lim_{t \rightarrow \infty} \gamma \exp(-\gamma(QAP)^T QAPt) \int_0^t \exp(\gamma(QAP)^T QAP\tau) d\tau (QAP)^T. \quad (4.3.9)$$

According to known results from [96], it can be verified the closed-form solution  $\bar{V} = (QAP)^{-1}$  of  $\bar{V}$  in (4.3.9), and it is independent of  $\gamma$ .

**Theorem 4.3.1.** [86] *Let  $A \in \mathbb{R}^{m \times n}$  be a given matrix and  $G \in \mathbb{R}_s^{n \times m}$  be an arbitrary matrix whose rank satisfies  $0 < s \leq r$ . Further, let  $G = PQ$  be a full rank factorization of  $G$  and  $\sigma(QAP) = \{\lambda_1, \dots, \lambda_n\}$  be the spectrum of  $QAP$ . Suppose that  $QAP$  is regular. Then the*

outer inverse  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$  can be generated using the limiting value  $\bar{V}$  of (4.3.9), as follows:

$$P\bar{V}Q = A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}. \quad (4.3.10)$$

Also, the RNN initiated by (4.3.6) is globally asymptotically stable.

*Proof.* Since  $QAP$  is regular, the condition

$$\operatorname{Re}(\lambda_j) > 0, \quad j = 1, \dots, n \quad (4.3.11)$$

is satisfied. Then the proof can be verified using the full-rank representation (4.3.1) and the results from [94].  $\square$

Following the recurrent neural networks for the matrix inversion [122, 129], our neural network is composed from  $s$  independent sub-networks, where each sub-network generates a column of  $V(t)$ . Let us denote by  $v_j(t)$  (resp.  $g_j$ ) the  $j$ th column vector of  $V(t)$  (resp.  $G = (QAP)^T$ ), for  $j = 1, 2, \dots, m$ . The dynamics of the  $j$ th sub-network can be expressed as follows:

$$\frac{dv_j(t)}{dt} = -\gamma \left( (QAP)^T QAP v_j(t) - g_j \right). \quad (4.3.12)$$

The dynamics defined in (4.3.12) indicates that each sub-network is essentially the same as the recurrent neural network presented in [94]. The connection weight matrix  $W = -\gamma(QAP)^T QAP$  is identical for each sub-network and the biasing threshold vector for the  $j$ th sub-network is  $\gamma g_j = \{\gamma g_{1j}, \dots, \gamma g_{nj}\}$ . Elements of the matrix  $W$  (resp.  $V$ ) are denoted by  $w_{ij}$  (resp.  $v_{ij}$ ).

Elements of unknown matrix  $V(t)$  are computed using

$$\dot{v}_{ij} = \frac{dv_{ij}}{dt} = \sum_{k=1}^n w_{ik} v_{kj} + \gamma g_{ij}, \quad i, j = 1, \dots, n. \quad (4.3.13)$$

Elements of the column  $v_j = \{v_{1j}, \dots, v_{nj}\}$  are generated in the  $j$ th sub-network.

Two advantages of the proposed RNN(4.2.4) approach are immediately observable:

1. Dimensions of  $QAP$  are equal to  $s \times s$ ; they are potentially much smaller than the dimensions  $m$  and  $n$  of  $AG$  or  $GA$ .

2. RNN(4.2.4) is globally asymptotically stable in the case when  $QAP$  is invertible. Therefore, RNN(4.2.4) is an elegant way to avoid the requirements on the spectrum of the matrix  $AG$  or  $GA$ .



### 4.3.2 Neural network RNN(4.2.3) based on (4.2.3)

Let us consider the case  $n \leq m$  in more details. The second type of RNN, called RNN(4.2.3), integrates  $n$  independent sub-networks, where each sub-network generates a column vector of  $V(t)$ . Under the assumptions that  $v_j(t)$  denotes the  $j$ th column vector of  $V(t)$  and  $g_j$  stands for the  $j$ th column vector of  $G_1 = (GA)^k$ ,  $k \geq 1$ , the dynamics of the  $j$ th sub-network can be expressed as follows:

$$\frac{dv_j(t)}{dt} = -\gamma \left( (G_1)^{k+1} v_j(t) - g_j \right), \quad j = 1, 2, \dots, m. \quad (4.3.14)$$

The integer  $k$  in (4.3.14) is the first positive integer satisfying

$$\operatorname{Re}(\lambda_j^{k+1}) \geq 0, \quad j = 1, \dots, n, \quad (4.3.15)$$

where  $\sigma((GA)^{k+1}) = \{\lambda_1, \dots, \lambda_n\}$  is the spectrum of  $(GA)^{k+1}$ . The integer  $k$  can be defined using the results from [84].

The connection weight matrix  $W - \gamma(G_1)^{k+1}$  is identical for each sub-network and the biasing threshold vector for the  $j$ th sub-network is  $\gamma g_j = \{\gamma g_{1j}, \dots, \gamma g_{nj}\}$ . Elements of the matrix  $W$  (resp.  $V$ ) are denoted by  $w_{ij}$  (resp.  $v_{ij}$ ). Then

$$\dot{v}_{ij} = \frac{dv_{ij}}{dt} = \sum_{k=1}^n v_{kj} w_{ik} + \gamma g_{ji}, \quad i, j = 1, \dots, n. \quad (4.3.16)$$

Elements of the column  $v_j = \{v_{1j}, \dots, v_{nj}\}$  are generated in the  $j$ th sub-network.

**Corollary 4.3.1.** *Let  $A \in \mathbb{R}^{m \times n}$  be a given matrix,  $G \in \mathbb{R}_s^{n \times m}$  be an arbitrary matrix and  $\sigma(GA) = \{\lambda_1, \dots, \lambda_n\}$  is the spectrum of  $GA$ . Assume that the integer  $k$  is the first positive integer satisfying (4.3.15). Then the outer inverse  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$  can be generated using*

$$A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)} = \bar{V} G = \lim_{t \rightarrow \infty} \gamma \exp(-\gamma(GA)^{k+1}t) \int_0^t \exp(\gamma(GA)^{k+1}\tau) d\tau \cdot G. \quad (4.3.17)$$

*Proof.* According to [84, Theorem 1],  $\bar{V} = (GA)^\#$ . Then the proof follows from Proposition 4.2.2.  $\square$

**Remark 4.3.1.** *The results stated in Corollary 4.3.1 are applicable in the case  $n \leq m$ . In the opposite case,  $m \leq n$ , similar result can be stated using the matrix  $AG$  instead of  $GA$ .*

### 4.3.3 Relationships between different RNNs

For the sake of simplicity, by  $RNN(AT S2)$  is denoted the RNN proposed in [138]. Outputs generated by defined RNNs for input parameters  $\gamma, A, G$  are denoted by  $RNN(AT S2)[\gamma, A, G]$ ,  $RNN(4.2.3)[\gamma, A, G]$  and  $RNN(4.2.4)[\gamma, A, G]$ .

In the particular case  $G = A^T$  all three approaches,  $RNN(AT S2)$ ,  $RNN(4.2.3)$  and  $RNN(4.2.4)$ , derive known results concerning the usual inverse, originated in [94], as well as known results concerning the Moore–Penrose inverse [96]:

$$RNN(AT S2)[\gamma, A, A^T] = RNN(4.2.3)[\gamma, A, A^T] = RNN(4.2.4)[\gamma, A, A^T] = A^\dagger.$$

The case  $G = A^\# = N^{-1}A^*M$  produces the results corresponding to the weighted Moore–Penrose inverse  $A_{M,N}^\dagger$  from [108]:

$$RNN(AT S2)[\gamma, A, A^\#] = RNN(4.2.3)[\gamma, A, A^\#] = A_{M,N}^\dagger.$$

In the case  $G = A^k$ ,  $k \geq \text{ind}(A)$ ,  $RNN(AT S2)[\gamma, A, A^k]$  generates the Drazin inverse  $A^D$ . This case was devised in [84]. An algorithm for appropriate choice of the exponent  $k$  is given in [84].

For a square matrix  $A$  of index  $\text{ind}(A) = 1$ ,  $k \geq 1$ ,  $RNN(AT S2)[\gamma, A, A^k]$  produces the group inverse  $A^\#$  of  $A$ .

The choice  $\text{rank}(G) = r = \text{rank}(A)$  it is not difficult to verify  $\overline{V}_G \in A\{1, 2\}$ .

Clearly, the following is valid under the assumption that  $G = PQ$  is a full-rank factorization of  $G$ :

$$RNN(4.2.3)[\gamma, A, G] = \begin{cases} G * RNN(AT S2)[\gamma, AG, (AG)^k], & m \leq n \\ RNN(AT S2)[\gamma, GA, (GA)^k] * G, & n \leq m, \end{cases}$$

where  $k$  is the first integer satisfying (4.3.15). Also, the following general statement is valid:

$$RNN(4.2.4)[\gamma, A, G = PQ] = P * RNN(AT S2)[\gamma, QAP, (QAP)^T] * Q.$$

It is not difficult to verify the following in the case  $G = A^k$ ,  $k \geq \text{ind}(A)$ :

$$RNN(AT S2)[\gamma, A, A^k] = RNN(4.2.3)[\gamma, A, A^k] = A^D.$$

Also, the full-rank representations of the sets  $A\{2, 4\}_s$  and  $A\{2, 3\}_s$  as particular cases of the full-rank representation of the set  $A\{2\}_s$  are derived in [77]. Introduced full-rank representations enable adaptation of well-known algorithms for computing outer inverses with

prescribed range and null space into corresponding algorithms for computing  $\{2, 4\}$  and  $\{2, 3\}$ -inverses.

In the case  $G = (QA)^*Q$ , where  $Q \in \mathbb{C}^{s \times m}$  is an appropriate matrix,  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$  reduces to

$$A_{\mathcal{N}(QA)^\perp, \mathcal{N}(Q)}^{(2,4)} = (QA)^*(QA(QA)^*)^{-1}Q. \quad (4.3.18)$$

In the case  $G = P(AP)^*$ , where  $P \in \mathbb{C}^{n \times s}$  is an appropriate matrix,  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$  produces

$$A_{\mathcal{R}(P), \mathcal{R}(AP)^\perp}^{(2,3)} = P((AP)^*AP)^{-1}(AP)^*. \quad (4.3.19)$$

According to (4.3.18), the following is satisfied:

$$\begin{aligned} A_{\mathcal{N}(QA)^\perp, \mathcal{N}(Q)}^{(2,4)} &= (QA)^\top(QA(QA)^\top)^{-1}Q = \mathbf{RNN}(ATS2)[\gamma, A, (QA)^\top Q] \\ &= (QA)^\top * \mathbf{RNN}(ATS2)[\gamma, QA(QA)^\top, (QA(QA)^\top)^\top] * Q \\ &= \mathbf{RNN}(4.2.4)[\gamma, A, (QA)^\top Q] \\ &= (QA)^\top Q * \mathbf{RNN}(ATS2)[\gamma, (QA)^\top QA, (QA)^\top QA] \\ &= \mathbf{RNN}(4.2.3)[\gamma, A, (QA)^\top Q]. \end{aligned}$$

Similarly, according to (4.3.19), the following holds:

$$\begin{aligned} A_{\mathcal{R}(P), \mathcal{R}(AP)^\perp}^{(2,3)} &= P((AP)^\top AP)^{-1}(AP)^\top = \mathbf{RNN}(ATS2)[\gamma, A, P(AP)^\top] \\ &= P * \mathbf{RNN}(ATS2)[\gamma, (AP)^\top AP, ((AP)^\top AP)^\top] * (AP)^\top \\ &= \mathbf{RNN}(4.2.4)[\gamma, A, P(AP)^\top] \\ &= P(AP)^\top * \mathbf{RNN}(ATS2)[\gamma, AP(AP)^\top, (AP(AP)^\top)^\top] \\ &= \mathbf{RNN}(4.2.3)[\gamma, A, P(AP)^\top]. \end{aligned}$$

## 4.4 Numerical experiments on GNN based on full rank representation

**Example 4.4.1.** Consider the matrix

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 \\ 1 & 3 & 4 & 6 & 2 \\ 2 & 3 & 4 & 5 & 3 \\ 3 & 4 & 5 & 6 & 4 \\ 4 & 5 & 6 & 7 & 6 \\ 6 & 6 & 7 & 7 & 8 \end{bmatrix}, \quad (4.4.1)$$

and choose the following matrices  $P \in \mathbb{R}^{5 \times 2}$  and  $Q \in \mathbb{R}^{2 \times 6}$

$$P = \begin{bmatrix} 0 & 0 \\ 2 & 1 \\ 3 & 2 \\ 5 & 3 \\ 1 & 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}. \quad (4.4.2)$$

According to (4.2.4), exact  $\{2\}$ -inverse of  $A$  corresponding to  $G = PQ$  is defined by

$$A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)} = P(QAP)^{-1}Q = \frac{1}{174} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -21 & 19 & -21 & 19 & -21 & 19 \\ 60 & -46 & 60 & -46 & 60 & -46 \\ 39 & -27 & 39 & -27 & 39 & -27 \\ -102 & 84 & -102 & 84 & -102 & 84 \end{bmatrix}.$$

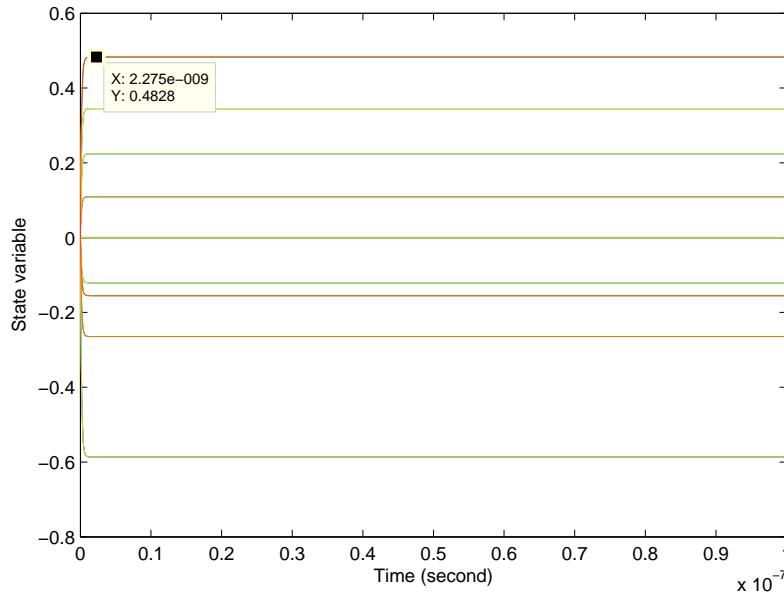
**A)** The spectrum of  $GA$  is included in the set

$$\sigma(GA) = \{266.346716180717, 0.653283819282910, 0, 0, 0\}$$

and provides the condition (4.1.7). Applying  $\text{RNN}(\text{ATS2})[10^{10}, A, G = PQ]$ , we obtain the following approximation of  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$ :

$$X = \begin{bmatrix} 0 & 0. & 0 & 0 & 0 & 0 \\ -0.1207 & 0.1092 & -0.1207 & 0.1092 & -0.1207 & 0.1092 \\ 0.3448 & 0.2644 & 0.3448 & -0.2644 & 0.3448 & -0.2644 \\ 0.2241 & -0.1552 & 0.2241 & -0.1552 & 0.2241 & -0.1552 \\ -0.5862 & 0.4828 & -0.5862 & 0.4828 & -0.5862 & 0.4828 \end{bmatrix}. \quad (4.4.3)$$

Convergence properties are illustrated in Figure 4.1


 Figure 4.1: Convergence behavior of the RNN [138] in  $10^{-7}$  seconds for Example 4.4.1

**B)** Since  $\text{rank}(GA) = \text{rank}((GA)^2) = 2$ , the existence of  $(GA)^\#$  is ensured as well as the existence of the outer inverse  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)} = (GA)^\#G$ . Also  $s((GA)^2) \geq 0$  ensures stability of RNN(4.2.3). Using  $G_1 = GA$  and applying the RNN defined according to (4.3.16) with  $\gamma = 10^{10}$ , one can obtain the following approximation of the group inverse  $(GA)^\#$ :

$$\text{RNN}(4.2.3)[10^{10}, GA, GA] = (GA)_g = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.2240 & 0.1194 & 0.0148 & -0.0898 & 0.2755 \\ -0.6056 & -0.3215 & -0.0373 & 0.2470 & -0.7442 \\ -0.3816 & -0.2020 & -0.0225 & 0.1570 & -0.4690 \\ 1.0537 & 0.5602 & 0.0668 & -0.4267 & 1.2952 \end{bmatrix}.$$

Now, the approximation  $(GA)_gG$  of  $(GA)^\#G = A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$  coincides with the approximation  $X$  in (4.4.3).

**C)** An approximation  $(QAP)^{(-1)}$  of  $(QAP)^{-1}$  can be produced using RNN(ATS2)[ $10^9$ ,  $QAP$ ,  $(QAP)^T$ ]:

$$(QAP)^{(-1)} = \begin{bmatrix} 0.4828 & -0.5862 \\ -0.8563 & 1.0517 \end{bmatrix}.$$

The behavior of the method RNN(ATS2)[ $10^9$ ,  $QAP$ ,  $(QAP)^T$ ] in the process of its convergence is illustrated in Figure 4.2.

Now, the approximation

$$Y = P(QAP)^{(-1)}Q = P * \text{RNN}(\text{ATS2})[10^9, QAP, (QAP)^T] * Q = \text{RNN}(\text{ATS2})[10^9, A, PQ]$$

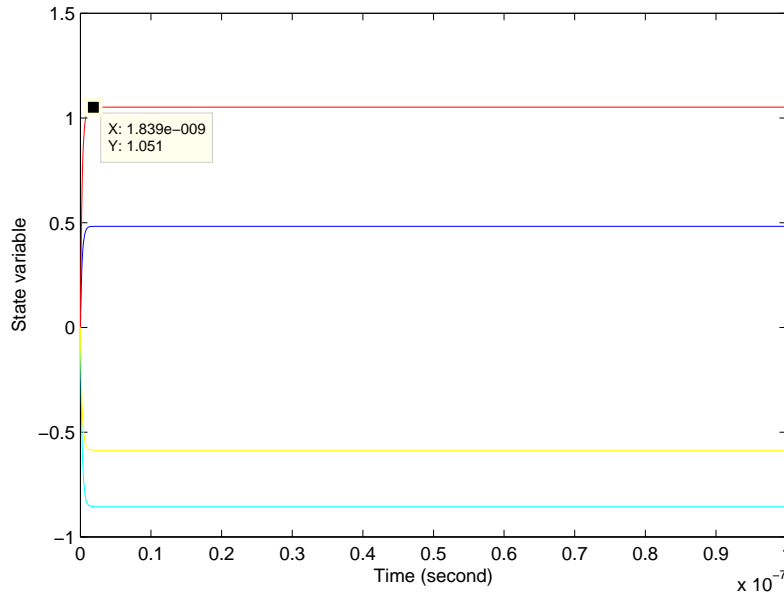


Figure 4.2: Convergence behavior of the RNN(4.2.4) in  $10^{-7}$  seconds for Example 4.4.1

of the outer inverse coincides with the resulting matrix (4.4.3).

**Example 4.4.2.** Consider the matrices  $A$  as in Example 4.4.1 and randomly generated  $3 \times 6$  matrix

$$Q = \begin{bmatrix} 0.8147 & 0.9134 & 0.2785 & 0.9649 & 0.9572 & 0.1419 \\ 0.9058 & 0.6324 & 0.5469 & 0.1576 & 0.4854 & 0.4218 \\ 0.1270 & 0.0975 & 0.9575 & 0.9706 & 0.8003 & 0.9157 \end{bmatrix}.$$

Now, let us consider the matrix

$$G_Q = (QA)^T Q = \begin{bmatrix} 16.6417 & 15.1378 & 20.0523 & 24.0504 & 24.1164 & 17.1831 \\ 24.0178 & 21.9829 & 26.1593 & 32.2128 & 32.8233 & 22.0808 \\ 30.6784 & 28.0700 & 32.7198 & 40.3917 & 41.3443 & 27.5294 \\ 38.0545 & 34.9151 & 38.8268 & 48.5541 & 50.0512 & 32.4271 \\ 23.7370 & 21.6404 & 28.2141 & 34.0462 & 34.1819 & 24.1278 \end{bmatrix}.$$

Clearly,  $\sigma(G_Q A) \geq 0$ , and the output generated by applying  $\text{RNN}(\text{ATS2})[10^{10}, A, G = G_Q]$  is equal to

$$X = \begin{bmatrix} 0.9990 & -0.0844 & 1.0245 & -1.7595 & -0.6968 & 0.9428 \\ -0.8849 & -0.0042 & -0.7783 & 1.4540 & 0.5631 & -0.7026 \\ 1.5164 & 0.0354 & 1.3197 & -2.4205 & -0.9194 & 1.1878 \\ -0.3676 & 0.1157 & -0.4832 & 0.7931 & 0.3405 & -0.4576 \\ -1.1124 & -0.1118 & -0.8261 & 1.6635 & 0.6168 & -0.7257 \end{bmatrix}. \quad (4.4.4)$$

The  $\{2, 4\}$  inverse of  $A$  is defined by

$$A_{\mathcal{N}(QA)^\perp, \mathcal{N}(Q)}^{(2,4)} = (QA)^\top (QA(QA)^\top)^{-1} Q,$$

which coincides with  $X$  defined in (4.4.4).

Now, our intention is to solve the same problem using RNN(4.2.4). For this purpose, it suffices to consider the matrices  $P_1 = (QA)^\top$ ,  $Q_1 = Q$  and later  $A_1 = Q_1 A P_1$ ,  $G_1 = A_1^\top$ . Applying RNN(ATS2)[ $10^{12}$ ,  $A_1$ ,  $G_1$ ], we obtain the following approximation

$$(Q_1 A P_1)^{(-1)} = \begin{bmatrix} 21.9358 & -31.9662 & 1.7207 \\ -31.9663 & 46.6893 & -2.5770 \\ 1.7207 & -2.5770 & 0.1811 \end{bmatrix}.$$

of  $(Q_1 A P_1)^{-1} = (QA(QA)^\top)^{-1}$ . Now, the matrix  $P_1 (Q_1 A P_1)^{(-1)} Q_1$  is equal to the matrix  $X$  which is defined in (4.4.4).

**Example 4.4.3.** The goal of this example is to show global asymptotic stability of RNN(4.2.3). For this purpose, we use randomly generated initialization  $V_0$  of the matrix of activation state variables  $V(t)$ . The matrices  $A$ ,  $G$ ,  $P$  and  $Q$  are chosen as in Example 4.4.1.

The RNN(ATS2)[ $\gamma$ ,  $A$ ,  $G$ ] is not asymptotically globally stable, which means that the solution, in general, does not converge to the outer inverse  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)}$  and depends on choice of the initial matrix  $V_0$ . The function RNN(ATS2)[ $10^{10}$ ,  $A$ ,  $G$ ] converges, but generates the incorrect result:

$$\begin{bmatrix} 1.2571 & -0.9957 & 0.3411 & 1.0149 & 0.1232 & -0.1224 \\ -1.0682 & 0.3991 & -0.5537 & 0.2515 & 0.5429 & -1.4664 \\ 1.9182 & 0.0495 & 2.3513 & 0.5645 & 2.4586 & 0.2087 \\ -0.4413 & -0.5021 & -0.9267 & -0.6737 & -1.3316 & -0.0051 \\ -1.4994 & 1.0369 & -1.1589 & -0.6180 & -1.5949 & 1.2889 \end{bmatrix}.$$

On the other hand, the RNN(ATS2)[ $10^{10}$ ,  $QAP$ ,  $(QAP)^\top$ ] converges successfully starting from the generated initialization  $V_0$ , and RNN(4.2.3)[ $10^{10}$ ,  $A$ ,  $G$ ] =  $P * \text{RNN(ATS2)}[10^{10}, QAP, (QAP)^\top] * Q$  produces the outer inverse generated in Example 4.4.1 using  $\gamma = 10^{10}$ .

**Example 4.4.4.** In this example, the matrix  $A$  is reused from Example 4.4.1 and the matrices  $P$ ,  $Q$  are randomly generated:

$$P = \text{rand}(5, 3) = \begin{bmatrix} 0.9421 & 0.3532 & 0.6491 \\ 0.9561 & 0.8212 & 0.7317 \\ 0.5752 & 0.0154 & 0.6477 \\ 0.0598 & 0.0430 & 0.4509 \\ 0.2348 & 0.1690 & 0.5470 \end{bmatrix},$$

$$Q = rand(3, 6) = \begin{bmatrix} 0.2963 & 0.6868 & 0.6256 & 0.9294 & 0.4359 & 0.5085 \\ 0.7447 & 0.1835 & 0.7802 & 0.7757 & 0.4468 & 0.5108 \\ 0.1890 & 0.3685 & 0.0811 & 0.4868 & 0.3063 & 0.8176 \end{bmatrix}.$$

The matrix  $G$  is defined by  $G = PQ$ . The spectrum of  $GA$  contains negative values:

$$\sigma(GA) = \{87.5644, -0.3869, -0.0621, 0.0000, 0.0000\}.$$

This causes divergence of  $RNN(ATS2)$ , with the incorrect result

$$1.0e^{+168} \begin{bmatrix} -0.8558 & -2.3809 & 1.1049 & 4.6212 & -2.3984 & 2.9684 \\ -1.1025 & -3.0671 & 1.4234 & 5.9531 & -3.0896 & 3.8240 \\ 0.1336 & 0.3717 & -0.1725 & -0.7215 & 0.3744 & -0.4634 \\ 0.7326 & 2.0380 & -0.9458 & -3.9557 & 2.0530 & -2.5409 \\ 0.4764 & 1.3253 & -0.6150 & -2.5723 & 1.3350 & -1.6523 \end{bmatrix}.$$

Using  $RNN(ATS2)[10^{12}, QAP, (QAP)^T]$ , it is possible to generate an approximation  $(QAP)^{(-1)}$  of  $(QAP)^{-1}$ . Later,  $\gamma = 10^{12}$  in

$RNN(4.2.3)[10^{12}, A, PQ] = P * RNN(ATS2)[10^{12}, QAP, (QAP)^T] * Q$  leads to the following approximation of  $A_{\mathcal{R}(P), \mathcal{N}(Q)}^{(2)} = P(QAP)^{-1}Q$ :

$$P(QAP)^{(-1)}Q = \begin{bmatrix} 0.7174 & -0.7519 & 0.2254 & -0.1981 & 0.0500 & 0.1226 \\ -1.8441 & 1.5217 & -1.4851 & 0.0060 & -0.1595 & 0.7803 \\ 1.5050 & -1.3240 & 1.0054 & -0.1087 & 0.1292 & -0.3699 \\ -0.1352 & 0.2318 & 0.1193 & 0.1430 & 0.0049 & -0.1943 \\ -0.3319 & 0.3753 & -0.0776 & 0.1210 & -0.0142 & -0.0667 \end{bmatrix}.$$

**Example 4.4.5.** In this example,  $A, P$  and  $Q$  are randomly generated matrices of the order  $m \times n, n \times s$  and  $s \times m$ , respectively. The matrix  $G$  is defined as  $G = PQ$ . IEEE arithmetic representation for Not-a-Number (NaN) denotes the divergence of the method. The symbol '-' means a time interval greater than  $5e+003$ . The results arranged in Table 4.1 confirm that  $RNN(4.2.4)$  shows the best performances. The results in Table 4.2 show that increase in dimensions  $m$  and  $n$  does not affect the CPU time and values of the residual error in the case when  $s$  remains unchanged.



Table 4.1: Results for three RNNs generated using on the set of randomly generated matrices

<i>RNN</i>	<i>Size <math>m, n, s</math></i>	<i>CPU Time</i>	$\ XAX - X\ _2$
RNN(ATS2)[ $10^{11}, A, G$ ]	10, 15, 2	1.5600	3.4837e-006
RNN(4.2.3)[ $10^{11}, A, G$ ]	10, 15, 2	0.6552	1.4050e-014
RNN(4.2.4)[ $10^{13}, A, G$ ]	10, 15, 2	0.1716	3.6337e-014
RNN(ATS2)[ $10^7, A, G$ ]	10, 15, 5	0.5928	1.7208
RNN(ATS2)[ $10^{11}, A, G$ ]	10, 15, 5	9.4693	NaN
RNN(4.2.3)[ $10^7, A, G$ ]	10, 15, 5	0.5772	0.5497
RNN(4.2.3)[ $10^{11}, A, G$ ]	10, 15, 5	5.7876	NaN
RNN(4.2.4)[ $10^{13}, A, G$ ]	10, 15, 5	0.1092	4.7661e-013
RNN(ATS2)[ $10^7, A, G$ ]	10, 15, 10	0.7644	4.1438
RNN(ATS2)[ $10^{11}, A, G$ ]	10, 15, 10	17.2381	NaN
RNN(4.2.3)[ $10^7, A, G$ ]	10, 15, 10	0.5772	1.3629
RNN(4.2.3)[ $10^{11}, A, G$ ]	10, 15, 10	17.0353	NaN
RNN(4.2.4)[ $10^{15}, A, G$ ]	10, 15, 10	1.1856	5.6842e-006
RNN(4.2.4)[ $10^{17}, A, G$ ]	10, 15, 10	1.2324	3.9636e-009
RNN(4.2.4)[ $10^{19}, A, G$ ]	10, 15, 10	1.2948	3.5628e-007
RNN(ATS2)[ $10^{11}, A, G$ ]	30, 45, 5	2.6530e + 003	NaN
RNN(4.2.3)[ $10^{11}, A, G$ ]	30, 45, 5	–	NaN
RNN(4.2.4)[ $10^{13}, A, G$ ]	30, 45, 5	0.1872	3.3026e-011

 Table 4.2: Results for three RNN(4.2.4) generated using  $\gamma = 10^{13}$  on the set of randomly generated matrices

<i>RNN</i>	<i>Size <math>m, n, s</math></i>	<i>CPU Time</i>	$\ XAX - X\ _2$
RNN(4.2.4)[ $10^{13}, A, G$ ]	90, 135, 15	3.2292	2.7609e-011
RNN(4.2.4)[ $10^{13}, A, G$ ]	180, 270, 15	2.5116	3.3336e-010
RNN(4.2.4)[ $10^{13}, A, G$ ]	360, 540, 15	2.5428	2.8120e-011
RNN(4.2.4)[ $10^{13}, A, G$ ]	720, 1080, 15	2.5116	7.2166e-006
RNN(4.2.4)[ $10^{13}, A, G$ ]	1040, 2160, 15	1.9500	3.7591e-011
RNN(4.2.4)[ $10^{13}, A, G$ ]	2080, 4320, 15	3.1512	1.2941e-009



## Chapter 5

# ZNN for computing matrix inverse based on hyperpower iterative methods

*This chapter investigates and exploits an analogy between the scaled hyperpower family (SHPI family) of iterative methods for computing the matrix inverse and the discretization of Zhang Neural Network (ZNN) models. A class of ZNN models corresponding to the family of hyperpower iterative methods for computing generalized inverses is defined on the basis of the discovered analogy. The Simulink implementation in Matlab of the introduced ZNN models is described in the case of scaled hyperpower methods of the order 2 and 3. Convergence properties of the proposed ZNN models are investigated as well as their numerical behavior.*

### 5.1 Introduction to ZNN design and known ZNN models

The GNN approach uses the Frobenius norm of the error matrix as the performance criterion and defines a neural network evolving along the negative gradient-descent direction. In the time-varying case, the underlying Frobenius norm of the error matrix cannot converge to zero even after infinite time [48]. Based on this fact, Zhang neural networks (or zeroing neural networks) (ZNNs) were developed for solving online time-varying problems. ZNN is originally invented by Zhang in his seminal work [130]. Their dynamics is designed based on an indefinite error-monitoring function instead of a usual norm-based energy function [131]. In addition, Zhang neural dynamic is, in general, implicit, whereas dynamic of GNNs is explicit [131].

Five complex-valued ZNN models which are aimed to computation of time-varying complex matrix generalized inverses were proposed and investigated in [48]. ZNN models for online time-varying full-rank matrix pseudoinversion were introduced and analyzed in [127]. The relationship between the Zhang matrix inverse and the Drazin inverse, discovered in [128], leads to the same dynamic state equation which was considered in [84] in the time invariant matrix case. The dynamical equation and corresponding artificial recurrent neural network

for computing the Drazin inverse of an arbitrary square real matrix, without any restriction on eigenvalues of its rank invariant powers, were proposed in [85]. A discrete-time model of ZNN for matrix inversion, which is depicted by a system of difference equations, was investigated in [125]. A general recurrent neural network model for online inversion of time-varying matrices was presented in [122]. The simulation and verification of such a ZNN were investigated in [129]. ZNN models for computing online time-varying Moore-Penrose inverse of a full-rank matrix were generalized, investigated and analyzed in [133]. Two complex Zhang neural network (ZNN) models for computing the Drazin inverse of arbitrary time-varying complex square matrix were presented in [104]. The design of the ZNNs defined in [104] is based on corresponding matrix-valued error functions arising from the limit representations of the Drazin inverse. As a continuation, the paper [67] investigated the computation of the Drazin inverse of a complex time-varying matrix by means of two ZNN models derived based on two Zhang functions constructed on the basis of two limit representations of the Drazin inverse. The finite-time convergence is ensured by means of the Li activation function.

Integration-enhanced noise-tolerant ZNN models (shortly IENTZNN) have been studied extensively last year. An IENTZNN model for time-varying matrix inversion process was defined in [32]. Zeroing neural networks which are able to eliminate various kinds of noise and resolve the redundancy in kinematic control have attracted a great attention last years. Zhang et al. in [33] designed a noise-tolerant zeroing neural network (NTZNN) design formula from the viewpoint of control. Corresponding discrete-time models were also defined in [33]. Various ZNN models applicable in resolving the redundancy of robotic manipulators for kinematic control in the presence of polynomial type noises were proposed in in [44, 45, 46].

The ZNN model for approximating the time-varying matrix inverse is defined using the matrix-valued indefinite error function

$$E(t) = A(t)X(t) - I. \quad (5.1.1)$$

The starting point in [48, 132] was the fact that the left Moore-Penrose inverse  $A(t)^\dagger$  satisfies  $A(t)^* A(t) A(t)^\dagger$ . Further, on the basis of the assumption that  $A(t)^* A(t)$  is invertible, the following matrix-based error function, called ZF(5), is considered

$$E(t) := A(t)^* A(t) X(t) - A(t)^*, \quad (5.1.2)$$

where  $X(t)$  corresponds to  $A(t)^\dagger$ . An elegant way to avoid the assumption of the invertibility of  $A(t)^* A(t)$  was presented in [48]. Namely, the authors of [48] defined the complex ZF which arises from the ZF defined in (5.1.2), and the Tikhonov regularization:

$$E(t) = (A(t)^* A(t) + \lambda I) X(t) - A(t)^*, \quad \lambda > 0. \quad (5.1.3)$$

The resulting ZNN model (5.1.3) is termed as complex ZNN-II Model.

In addition, the following complex function was used as the fundamental error-monitoring function (called ZFL2) in [104]:

$$E(t) = \left( A(t)^{l+1} + \lambda I \right) X(t) - A(t)^l \quad l \geq k = \text{ind}(A), \quad \lambda > 0. \quad (5.1.4)$$

The matrix  $X(t)$  in (5.1.4) corresponds to the Drazin inverse  $A(t)^D$ . Let us mention that the ZNN-II model in [67] is defined on the basis of the ZF defined in (5.1.4) and upon the Li activation function.

The leading idea of [105] was to comprise so far known ZNN models for computing generalized inverses into a unique comprehensive model corresponding to outer inverses in the time-varying complex matrix case. The ZNNATS2-I model defined in [105] requires two matrices  $A(t) \in \mathbb{C}_r^{m \times n}$ ,  $G(t) \in \mathbb{C}_s^{n \times m}$ ,  $0 < s \leq r$ , and it is aimed to numerical computations of the outer inverse  $A(t)_{\mathcal{R}(G), \mathcal{N}(G)}^{(2)}$ . The model is developed using the following two dual fundamental error-monitoring ZFs, proposed in [105]:

$$E_G(t) = \begin{cases} (G(t)A(t) + \lambda I) X(t) - G(t), & n \leq m, \quad \lambda > 0 \\ X(t) (A(t)G(t) + \lambda I) - G(t), & n > m, \quad \lambda > 0. \end{cases} \quad (5.1.5)$$

## 5.2 Correlation between iterations and ZNN models

There exist two categories of the numerical algorithms: direct and iterative methods. The direct method means that the accurate solutions for the problem are computed in finite steps. An iterative method for computing  $A^\dagger$  is a set of instructions for generating a sequence  $\{X_k\}$  converging to  $A^\dagger$ . The instructions specify how to select the initial approximation  $X_0$ , how to proceed from  $X_k$  to  $X_{k+1}$  for each  $k$ , and when to stop, having obtained a reasonable approximation. Main results can be found at [12, 91, 106].

One of the most important methods for computing the matrix inverse and various generalized inverses is the family of hyperpower iterations. These iterations possess an arbitrary order of the convergence  $p \geq 2$ , and are given by the standard form

$$X_{k+1} = X_k \left( I + R_k + \cdots + R_k^{p-1} \right) = X_k \sum_{i=0}^{p-1} R_k^i, \quad R_k = I - AX_k. \quad (5.2.1)$$

The hyperpower iterative family has been investigated extensively in a number of papers [17, 47, 50, 117].

The basic motivation in [87] was the fact that the scaled Newton method for the usual matrix inversion appears after the discretization of the Zhang Neural Network (ZNN) designed for the matrix inversion introduced in [125]. More precisely, we generalize the significant result

*From Zhang neural network to Newton iteration for matrix inversion,*

derived in [125], into the more general goal

*From Zhang neural network to scaled hyperpower iterations for matrix inversion and vice versa.*

Main goals from [87], reused in this section, can be summarized as follows.

(1) Generalize the discretization from [125] and consequently define the scaled hyperpower iterative methods (SHPI shortly) of an arbitrary order  $p \geq 2$ .

(2) In addition, our intention is to define a ZNN model (called ZNNCM) whose discretization produces the scaled Chebyshev iterative method introduced in [72].

(3) Numerical behavior as well as the convergence properties of the ZNNCM model are investigated.

(4) A combination of the ZNNCM and the ZNNM model, called the ZNNHM model, is also defined and considered in numerical testing.

These goals are fulfilled in [87] and in the further discussion we present the main results.

### 5.3 Scaled Hyperpower iterations as discretized ZNN models

It is assumed that the matrix  $A$  is a constant  $n \times n$  nonsingular matrix. For the sake of completeness, we restate main steps of the discretization which was defined in [125]. The matrix-valued error-monitoring function (ZF) of the form

$$E(X(t), t) := AX(t) - I \quad (5.3.1)$$

was used to derive the dynamic equation determined by the general pattern

$$\frac{dE(X(t), t)}{dt} = -\Gamma \mathcal{H}(E(X(t), t)), \quad (5.3.2)$$

where  $\Gamma \in \mathbb{R}^{n \times n}$  is a positive-definite matrix used to scale the inversion process and  $\mathcal{H}(\cdot) : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times n}$  denotes an appropriate matrix-valued activation-function mapping. An application of the general pattern (5.3.2) on the Zhang error-monitoring function (5.3.1) in the case  $\mathcal{H} = \mathcal{I}$  and  $\Gamma = \gamma I$ , where  $\gamma > 0$  is a scalar-valued design parameter, leads to the following implicit dynamic equation of ZNN:

$$A\dot{X}(t) = -\gamma (AX(t) - I). \quad (5.3.3)$$

Further, assume that the linear activation function  $\mathcal{H} = \mathcal{I}$  is used and the discretization of the continuous-time model (5.3.3) is performed by using the Euler forward-difference rule

$$\dot{X}(t) \approx (X_{k+1} - X_k)/\tau,$$

where  $\tau$  denotes the sampling time and  $X_k = X(t = k\tau)$ ,  $k = 1, 2, \dots$ . Then the discrete-time model of (5.3.3) is defined by

$$AX_{k+1} = AX_k - \gamma (AX_k - I), \quad (5.3.4)$$

where  $\gamma = \tau \gamma > 0$  is the step size that should appropriately be selected for the convergence to the theoretical inverse  $A^{-1}$ . Since  $A$  is nonsingular, the implicit discrete-time ZNN model can be rewritten as

$$X_{k+1} = X_k - \gamma A^{-1} (AX_k - I). \quad (5.3.5)$$

According to [122], the state matrix  $X(t)$  converges to  $A^{-1}$  in the continuous-time ZNN model (5.3.3). Hence, it is justifiably to replace  $A^{-1}$  by its approximation  $X_k$ . This replacement yields the following explicit difference equation of the discrete-time ZNN for the nonsingular matrix inversion:

$$X_{k+1} = X_k - \gamma X_k (AX_k - I) = X_k (I + \gamma (I - AX_k)). \quad (5.3.6)$$

The iterative rule (5.3.6) is exactly of the form of the scaled Newton iteration for computing outer inverses with prescribed range and null space, introduced in [64, 65]:

$$X_{k+1} = (1 + \gamma)X_k - \gamma X_k AX_k, \quad X_0 = \alpha G, \quad \gamma \in (0, 1], \quad (5.3.7)$$

where  $G \in \mathbb{C}_r^{m \times n}$  is a given matrix,  $\alpha, \gamma$  are real constants and  $G \in \mathbb{C}_s^{n \times m}$  is a chosen matrix and  $0 < s \leq r$ . In the case  $\gamma = 1$  the iterative process (5.3.7) produces well known generalization of the Schultz iterative method, intended for computing outer inverses [19, 115].

Charif et al. in [9] developed a new fast online algorithms for motion estimation which is based on the Horn & Schunck algorithm with the Discrete Zhang Neural Networks (DZNN) defined by (5.3.6) and Simoncelli's matched-pair 5 tap filters. A novel implementation of the multi-dimensional Capon spectral estimator was proposed in [3]. The algorithm is derived using the discrete Zhang neural network for the online covariance matrix inversion.

In order to extend defined discretization, we start from the continuous-time model which is based on the error-monitoring function defined by the second and the third term of the hyperpower iterative process:

$$E_C(X(t), t) := I - AX(t) + (I - AX(t))^2 = 2I - 3AX(t) + (AX(t))^2. \quad (5.3.8)$$

In respect of the general ZNN pattern (5.3.2), the Zhang error-monitoring function (5.3.8) leads

to the following implicit dynamic equation

$$\begin{aligned}
 \dot{E}_C(X(t), t) &= -A\dot{X}(t) - A\dot{X}(t)(I - AX(t)) + (I - AX(t))(-A\dot{X}(t)) \\
 &= -3A\dot{X}(t) + A\dot{X}(t)AX(t) + AX(t)A\dot{X}(t) \\
 &= -\Gamma\mathcal{H}(2I - AX(t)(3I - AX(t))).
 \end{aligned} \tag{5.3.9}$$

The expected convergence of  $X(t)$  to  $A^{-1}$  approves the substitution  $AX(t) = I$  in the left hand side of (5.3.9), which in the case  $\Gamma = \gamma I$ ,  $\mathcal{H} = I$  leads to

$$\begin{aligned}
 A\dot{X}(t) &= \gamma(2I - AX(t)(3I - AX(t))) \\
 &= -\gamma(- (AX(t))^2 + 3AX(t) - 2I),
 \end{aligned} \tag{5.3.10}$$

where  $X(0)$  is appropriately defined initial point. Further, the discrete-time model of (5.3.10) based on the Euler forward-difference rule is defined by

$$AX_{k+1} = AX_k + \gamma(2I - AX_k(3I - AX_k)),$$

where  $\gamma = \tau\gamma > 0$  is the step size. After the replacement of  $A^{-1}$  by  $X_k$ , the implicit discrete-time ZNN model for the usual matrix inversion can be stated as

$$X_{k+1} = X_k \left( I + \gamma \left( 2I - 3AX_k + (AX_k)^2 \right) \right), \tag{5.3.11}$$

i.e. in the form of scaled hyperpower iterative method of the order 3. This method was proposed by Srivastava and Gupta in [72] for estimating the Moore-Penrose inverse. The scaled hyperpower iterative method (5.3.11) of the order 3 is developed by extending the scaled hyperpower iterative method (5.3.7) of the order 2.

As a consequence, it is reasonable to investigate the ZNN model defined in (5.3.10), initiated by the ZF defined in (5.3.8).

Our intention is to extend just defined principle in the widest sense, which assumes an arbitrary hyperpower method of the order  $p \geq 2$ . In view of the previously exploited principle, the corresponding continuous-time model starts from the error-monitoring function defined by

$$E_H(X(t), t) := \sum_{i=1}^{p-1} R(t)^i = \sum_{i=1}^{p-1} (I - AX(t))^i.$$

The principle of mathematical induction reveals

$$R(t)^i = - \sum_{l=0}^{i-1} (I - AX(t))^l A\dot{X}(t) (I - AX(t))^{i-1-l}.$$



Then the general ZNN design model (5.3.2) leads to the following implicit dynamic equation in the case  $\mathcal{H} = I$ :

$$-\sum_{i=1}^{p-1} \sum_{l=0}^{i-1} (I - AX(t))^l A\dot{X}(t) (I - AX(t))^{i-1-l} = -\gamma \sum_{i=1}^{p-1} (I - AX(t))^i. \quad (5.3.12)$$

After the substitution  $AX(t) = I$  in the left hand side of the implicit dynamics (5.3.12), one can verify

$$A\dot{X}(t) = \gamma \sum_{i=1}^{p-1} (I - AX(t))^i. \quad (5.3.13)$$

The discretization of the ZNN model (5.3.13) corresponding to the Euler forward-difference rule is given as

$$AX_{k+1} = AX_k + \gamma \sum_{i=1}^{p-1} (I - AX_k)^i, \quad \gamma = \tau \gamma > 0.$$

The inverse  $A^{-1}$  can be approximated by  $X_k$ , so that the implicit discrete-time ZNN model of (5.3.13), aimed for the matrix inversion, is given as

$$X_{k+1} = X_k \left( I + \gamma \sum_{i=1}^{p-1} (I - AX_k)^i \right). \quad (5.3.14)$$

The iterative rule (5.3.14) is referred as the scaled hyperpower iterative methods (SHPI shortly) of an arbitrary order  $p \geq 2$ .

In conclusion, it is reasonable to define the ZNN model (5.3.3) as the continuous-time version of the scaled Chebyshev iterative method, in the same way as the ZNN model (5.3.10) represents the continuous-time version of the scaled Newton iterative method. A comparison between these two concurrent ZNN models will be investigated.

## 5.4 Neural network architecture of ZNNCM model

The graphical editor, customizable block libraries and solvers available in Matlab Simulink are used for modeling and simulating the proposed dynamic systems. As it was mentioned in [123], the ZNN modeling could be readily developed, expanded and finally realized by using Matlab Simulink tool. This fact was our motivation to use the Matlab Simulink tool in the implementation of defined ZNN models. The ZNN model (5.3.3) will be denoted by ZNNM. Also, the ZNN model (5.3.10) is termed as ZNNCM. In addition, we define a hybrid method which starts from the ZNN model (5.3.3) and finishes with (5.3.10). Finally, GNN denotes the gradient based neural network from [138] in the nonsingular case, corresponding to the case  $G = A^T$  in the RNN1 model.

The Simulink implementation of the ZNNM model (5.3.3), restated in the equivalent form

$$\dot{X}(t) = (I - A)\dot{X}(t) - \gamma \mathcal{H}(AX(t) - I). \quad (5.4.1)$$

is presented in Figure 5.1.

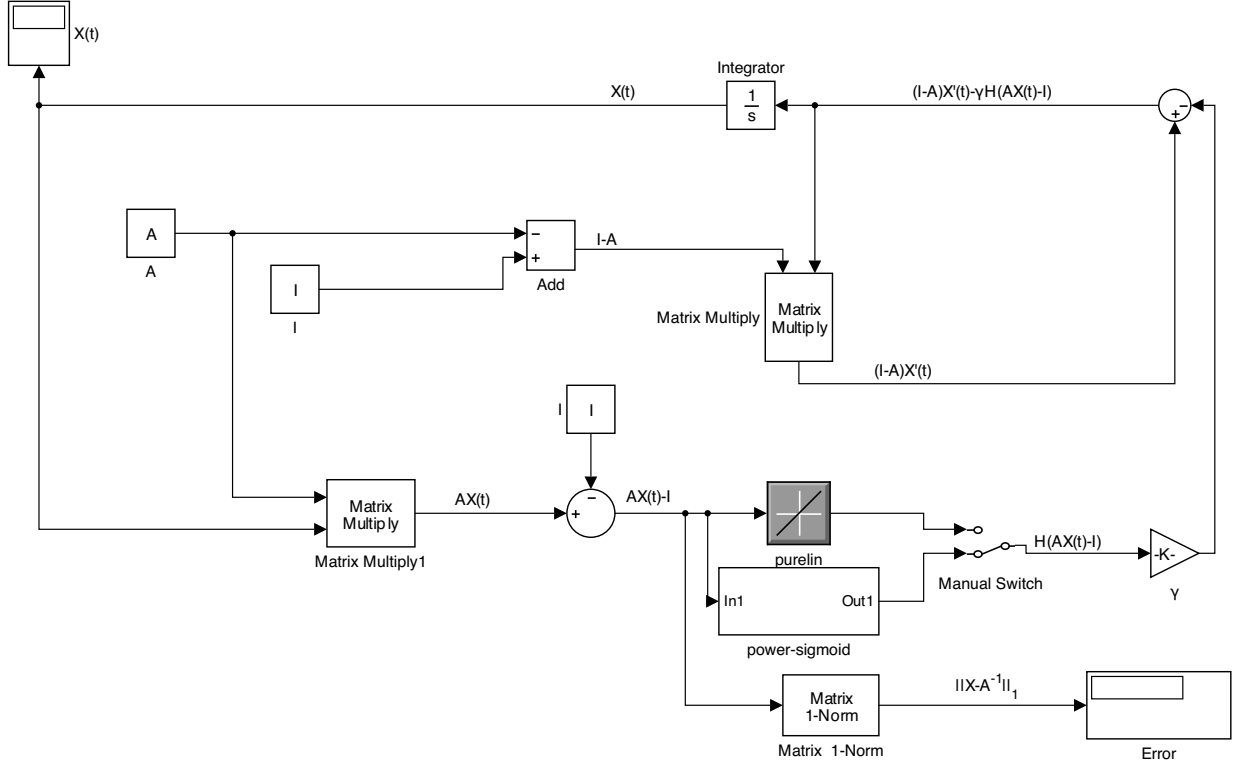


Figure 5.1: Simulink implementation of the ZNNM model.

In order to ensure the implementation, (5.3.10) is transformed into the following equivalent form:

$$\dot{X}(t) = (I - A)\dot{X}(t) - \gamma \mathcal{H}\left(- (AX(t))^2 + 3AX(t) - 2I\right). \quad (5.4.2)$$

The Matlab Simulink implementation of the ZNNCM model, based on (5.4.2), is presented in Figure 5.2.

Two appropriate activation functions, introduced in [42], will be exploited in nodes of two developed ZNNs. Their definitions will be restated here in order to complete the presentation. It is assumed that  $A \in \mathbb{C}^{n \times n}$  is written as  $B + \iota C$ , where  $\iota = \sqrt{-1}$  denotes the imaginary unit and  $B \in \mathbb{R}^{n \times n}$ ,  $C \in \mathbb{R}^{n \times n}$  are two real matrices. The matrices  $B, C$  correspond to real and imaginary part of the complex entries of  $A$ , respectively. Additionally, let  $\mathcal{F}(D)$  be an odd and monotonically increasing function element-wise applicable to elements of  $D = (d_{kj}) \in \mathbb{R}^{n \times n}$  according to the rule  $\mathcal{F}(D) = (f(d_{kj}))$ , where  $f(\cdot)$  is an odd and monotonically increasing function.

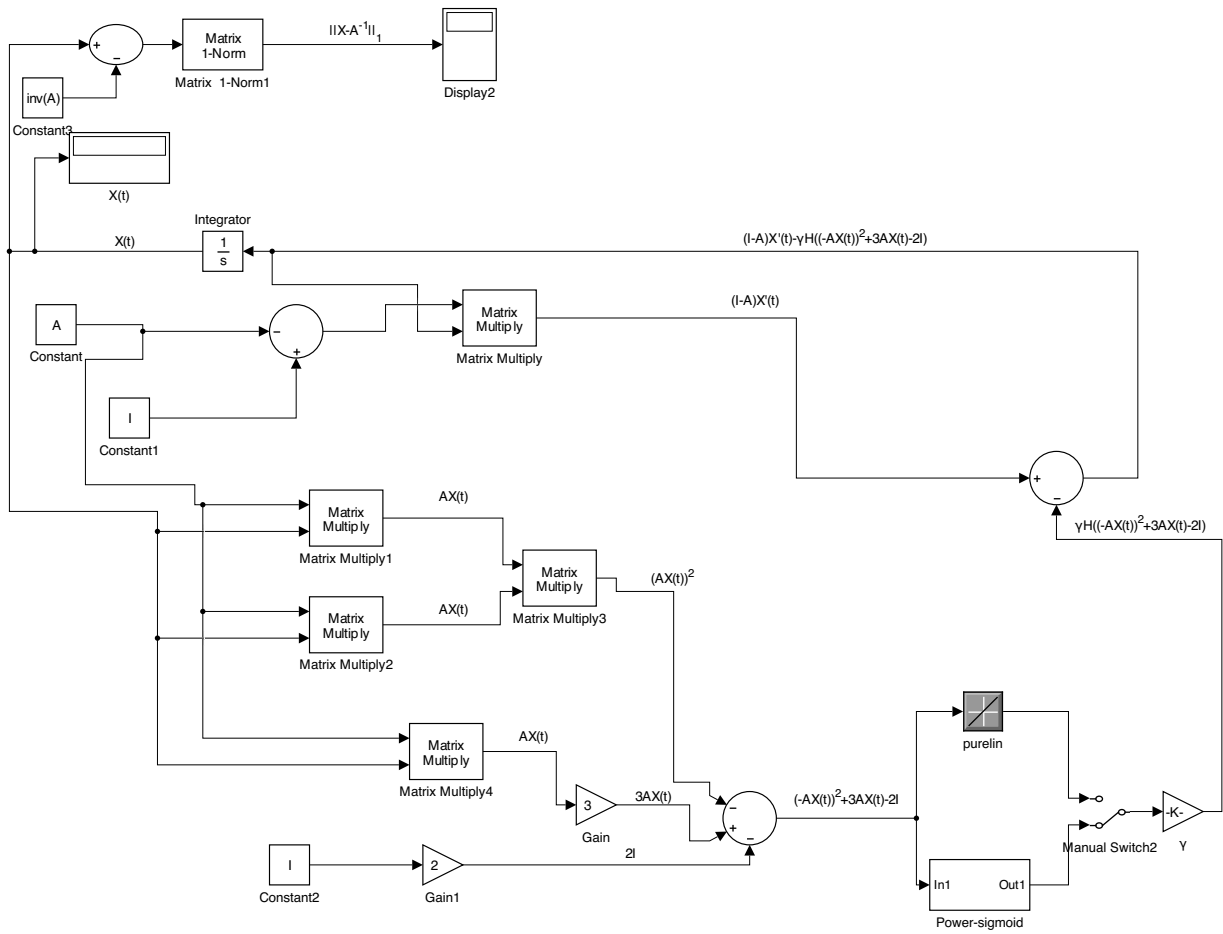


Figure 5.2: Simulink implementation of the ZNNCM model.

The type I activation function is defined by

$$\mathcal{H}_1(A) = \mathcal{H}_1(B + \iota C) = \mathcal{F}(B) + \iota \mathcal{F}(C). \quad (5.4.3)$$

Similarly, the type II activation function exploits the Hadamard product  $U \circ V = (u_{kj}v_{kj})$  of matrices  $U = (u_{kj})$  and  $V = (v_{kj})$ , and it is defined as

$$\mathcal{H}_2(A) = \mathcal{H}_2(B + \iota C) = \mathcal{F}(\Gamma) \circ \exp(\iota \Theta), \quad (5.4.4)$$

where  $\Gamma = |B + \iota C| \in \mathbb{R}^{n \times n}$  and  $\Theta = \Theta(B + \iota C) \in (-\pi, \pi]^{n \times n}$  denote element-wise modulus and the element-wise arguments, respectively, of the complex matrix  $B + \iota C$ . In sequel, we use the notation  $\mathcal{H}_k$  as a universal replacement for  $\mathcal{H}_1$  or  $\mathcal{H}_2$ .

The hybrid method starts using the ZNNCM method and then continues with the ZNNNM method. The starting point  $x_0$  of the ZNNNM method is just the output of the ZNNCM method and the finishing time of the ZNNCM method is the initial time of the ZNNNM method. The hybrid method will be denoted by  $\text{ZNNHM}(t_0)$ , where  $t_0$  denotes the time when the ZNNCM

stops and ZNNNM continues. More precisely, the ZNNCM method evaluates in the time interval  $[0, t_0]$ , while the ZNNNM method evaluates in the time interval  $[t_0, t]$ , where  $[0, t]$  denotes the considered time interval of the hybrid method. Since the ZNNNM model is globally exponentially convergent to the exact time-varying inverse  $A(t)^{-1}$ , the output  $x_0$  of the ZNNCM method could be submitted as the initial point of the ZNNNM method.

## 5.5 Convergence of the ZNNCM model

In this section, it is proven the convergence of the complex neural network model (5.3.10) based on both the activation functions  $\mathcal{H}_1$  and  $\mathcal{H}_2$ .

**Theorem 5.5.1.** [87] *Let the invertible complex matrix  $A \in \mathbb{C}^{n \times n}$  be given. Then the state matrix  $X(t) \in \mathbb{C}^{n \times n}$  of the complex neural network model (5.3.10) based on the activation function  $\mathcal{H}_1$  converges to the matrix inverse  $A^{-1}$ , and the solution is stable in the sense of Lyapunov.*

*Proof.* Let  $\tilde{X}(t) := A^{-1} - X(t)$ . Then  $X(t) = A^{-1} - \tilde{X}(t)$  and  $\dot{\tilde{X}}(t) = -\dot{X}(t)$ . Substituting the above two equations into (5.3.10) yields

$$A\dot{\tilde{X}}(t) = \gamma\mathcal{H}_1\left(-\left(I - A\tilde{X}(t)\right)^2 + 3\left(I - A\tilde{X}(t)\right) - 2I\right). \quad (5.5.1)$$

After substituting  $X(t) = A^{-1} - \tilde{X}(t)$  in the ZF defined by (5.3.8), one can verify

$$E_C(\tilde{X}(t), t) = A\tilde{X}(t) + (A\tilde{X}(t))^2. \quad (5.5.2)$$

In the view of the definition of activation function  $\mathcal{H}_1(\cdot)$ , taking into account  $E_C(t) = \mathbf{Re}(E_C(t)) + \iota\mathbf{Im}(E_C(t))$ , the general model  $\dot{E}(t) = -\gamma\mathcal{H}_1(E(t))$  splits into the following two equations in the real domain:

$$\mathbf{Re}(\dot{E}_C(t)) = -\gamma\mathcal{F}(\mathbf{Re}(E_C(t)))$$

and

$$\mathbf{Im}(\dot{E}_C(t)) = -\gamma\mathcal{F}(\mathbf{Im}(E_C(t))).$$

In order to verify the convergence, the Lyapunov function candidate is defined as

$$L(\tilde{X}(t), t) = L(t) = \frac{\|E_C(t)\|_F^2}{2} = \frac{\text{Tr}(E_C(t)^H E_C(t))}{2}. \quad (5.5.3)$$

Then the following identities can be verified:

$$\begin{aligned}
 \frac{dL(t)}{dt} &= \frac{\text{Tr} \left( \dot{E}_C(t)^H E_C(t) + E_C(t)^H \dot{E}_C(t) \right)}{2} \\
 &= -\frac{1}{2} \gamma \text{Tr} \left\{ \left( \mathcal{F}(\mathbf{Re}(E_C(t)))^T - \iota \mathcal{F}(\mathbf{Im}(E_C(t)))^T \right) (\mathbf{Re}(E(t)) + \iota \mathbf{Im}(E_C(t))) \right. \\
 &\quad \left. + \left( \mathbf{Re}(E_C(t))^T - \iota \mathbf{Im}(E_C(t))^T \right) (\mathcal{F}(\mathbf{Re}(E_C(t))))^T + \iota \mathcal{F}(\mathbf{Im}(E_C(t))) \right\} \\
 &= -\gamma \text{Tr} \left\{ \mathbf{Re}(E_C(t))^T \mathcal{F}(\mathbf{Re}(E_C(t))) + \mathbf{Im}(E_C(t))^T \mathcal{F}(\mathbf{Im}(E_C(t))) \right\}.
 \end{aligned}$$

Since  $\mathcal{F}(C) = (f(c_{kj}))$  and  $f(\cdot)$  is an odd and monotonically increasing function, it follows that

$$\begin{aligned}
 &\text{Tr} \left\{ \mathbf{Re}(E_C(t))^T \mathcal{F}(\mathbf{Re}(E_C(t))) + \mathbf{Im}(E_C(t))^T \mathcal{F}(\mathbf{Im}(E_C(t))) \right\} \\
 &= \text{Tr} \left\{ \mathbf{Re}(E_C(t))^T \mathcal{F}(\mathbf{Re}(E_C(t))) \right\} + \text{Tr} \left\{ \mathbf{Im}(E_C(t))^T \mathcal{F}(\mathbf{Im}(E_C(t))) \right\} \geq 0.
 \end{aligned}$$

To simplify notation, let us denote  $(i, j)$ th element of  $\mathbf{Re}(E(t))$  by  $e_{ij}$  and  $(i, j)$ th element of  $\mathbf{Im}(E(t))$  by  $e'_{ij}$ . Then

$$\begin{aligned}
 &\text{Tr} \left\{ \mathbf{Re}(E_C(t))^T \mathcal{F}(\mathbf{Re}(E_C(t))) + \mathbf{Im}(E_C(t))^T \mathcal{F}(\mathbf{Im}(E_C(t))) \right\} \\
 &= \sum_j e_{ij} f(e_{ij}) + \sum_j e'_{ij} f(e'_{ij}) \geq 0
 \end{aligned}$$

and finally

$$\frac{dL(\tilde{X}(t), t)}{dt} \begin{cases} < 0 & \text{if } E_C(\tilde{X}(t), t) \neq 0, \\ = 0 & \text{if } E_C(\tilde{X}(t), t) = 0. \end{cases}$$

Since  $\tilde{X}(t) = 0$  is an equilibrium point of the system (5.5.1), and  $E(0) = 0$  it follows that

$$\frac{dL(\tilde{X}(t), t)}{dt} \leq 0, \quad \forall \tilde{X}(t) \neq 0.$$

As a consequence of the Lyapunov stability theory, the equilibrium state  $\tilde{X}(t) = 0$  is stable. Since  $\tilde{X}(t) := A^{-1} - X(t)$ , we have  $X(t) \rightarrow A^{-1}, t \rightarrow \infty$ .

□

**Theorem 5.5.2.** [87] *Let the invertible complex matrix  $A \in \mathbb{C}^{n \times n}$  be given. Then the state matrix  $X(t) \in \mathbb{C}^{n \times n}$  of the complex neural network model (5.3.10) based on the activation function  $\mathcal{H}_2$  converges to the matrix inverse  $A^{-1}$ , and the solution is stable in the sense of the Lyapunov.*

*Proof.* Analogically as in the proof of Theorem 5.5.1, the general model is given by

$$\dot{E}_C(t) = -\gamma \mathcal{H}_2(E(t)),$$

where  $E(\tilde{X}(t), t) = E_C(t)$  is defined in (5.5.2). The definition of  $\mathcal{H}_2(\cdot)$  implies

$$\mathcal{H}_2(E_C(t)) = \mathcal{F}(|E_C(t)|) \circ \exp(\iota\Theta(E_C(t))).$$

The time derivative of the Lyapunov function candidate (5.5.3) is equal to

$$\begin{aligned} \frac{dL(t)}{dt} &= \frac{\text{Tr} \left( E_C(t)^H \dot{E}(t) + \dot{E}_C(t)^H E_C(t) \right)}{2} \\ &= -\frac{1}{2}\gamma \text{Tr} \left( E(t)^H \mathcal{H}_2(E_C(t)) + E(t) \mathcal{H}_2(E_C(t))^H \right) \\ &= -\frac{1}{2}\gamma \text{Tr} \left( E_C(t)^H \mathcal{H}_2(E_C(t)) + \left( E_C(t)^H \mathcal{H}_2(E_C(t)) \right)^H \right) \\ &= -\gamma \text{Tr} \left( \mathbf{Re} \left( E_C(t)^H \mathcal{H}_2(E_C(t)) \right) \right) \\ &= -\gamma \text{Tr} \left\{ \mathbf{Re} \left[ E_C(t)^H \mathcal{F}(|E_C(t)|) \circ \exp(\iota\Theta(E_C(t))) \right] \right\}. \end{aligned}$$

Since  $E_C(t) = |E_C(t)| \circ \exp(\iota\Theta(E_C(t)))$ , it follows that

$$\frac{dL(t)}{dt} = -\gamma \text{Tr} \left\{ \mathbf{Re} \left[ \exp(-\iota\Theta(E_C(t)^H)) \circ |E_C(t)^H| \left( \mathcal{F}(|E_C(t)|) \circ \exp(\iota\Theta(E_C(t))) \right) \right] \right\}.$$

Again, using that  $\mathcal{F}(\cdot)$  is monotonically increasing, it follows the inequality  $\mathcal{F}(|E_C(t)|) > 0$ , for  $E(t) \neq 0$ , and  $\mathcal{F}(|E_C(t)|) = 0$ , for  $E_C(t) = 0$  which implies that

$$\frac{dL(\tilde{X}(t), t)}{dt} \leq 0, \quad \forall \tilde{X}(t) \neq 0.$$

According to the Lyapunov stability theory, the equilibrium state  $\tilde{X}(t) = 0$  is stable and,  $X(t) \rightarrow A^{-1}, t \rightarrow \infty$ .  $\square$

## 5.6 Simulation results and its comparison

**Example 5.6.1.** As it was observed in [60], the GNN models are not appropriate for calculating the inverse of a matrix with a big condition number. So this is a reason to apply the ZNNCM model to a matrix with a big condition number. The following matrix  $A$  is considered for this purpose:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 \\ 1 & 3 & 9 & 27 & 81 & 243 \\ 1 & 4 & 16 & 64 & 256 & 1024 \\ 1 & 5 & 25 & 125 & 625 & 3125 \end{bmatrix}$$

with the condition number  $\text{cond}(A) = 5.7689e+04$ . The theoretical inverse of  $A$  is equal to

$$A^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{137}{60} & 5 & -5 & \frac{10}{3} & -\frac{5}{4} & \frac{1}{5} \\ \frac{15}{8} & -\frac{77}{12} & \frac{107}{12} & -\frac{13}{2} & \frac{61}{24} & -\frac{5}{12} \\ -\frac{17}{24} & \frac{71}{24} & -\frac{59}{12} & \frac{49}{12} & -\frac{41}{24} & \frac{7}{24} \\ \frac{1}{8} & -\frac{7}{12} & \frac{13}{12} & -1 & \frac{11}{24} & -\frac{1}{12} \\ -\frac{1}{120} & \frac{1}{24} & -\frac{1}{12} & \frac{1}{12} & -\frac{1}{24} & \frac{1}{120} \end{bmatrix}.$$

Using the scaling parameter  $\gamma = 10^6$ , the Power-Sigmoid activation function and *ode45* solver, after  $t = 10^{-5}$  s sec, the ZNNCM model gives the results  $\text{ZNNCM}(A)$  which is equal to

$$\begin{bmatrix} 0.99999999294662 & -0.000000000000000 & -0.000000000000000 \\ -2.283333331722813 & 4.999999996473304 & -4.999999996473306 \\ 1.874999998677495 & -6.416666662140734 & 8.916666660377395 \\ -0.708333332833721 & 2.958333331246700 & -4.916666663198751 \\ 0.12499999911833 & -0.583333332921884 & 1.083333332569217 \\ -0.00833333327456 & 0.04166666637277 & -0.083333333274555 \\ 0.000000000000000 & -0.000000000000000 & -0.000000000000000 \\ 3.333333330982192 & -1.24999999118328 & 0.19999999858933 \\ -6.499999995415274 & 2.541666664873933 & -0.416666666372777 \\ 4.083333330453185 & -1.708333332128381 & 0.291666666460944 \\ -0.99999999294657 & 0.458333333010054 & -0.083333333274555 \\ 0.083333333274555 & -0.04166666637278 & 0.008333333327456 \end{bmatrix}$$

with the absolute error  $\|X(t) - A^{-1}\|_1 = 1.4106399215397e-08$ . Trajectories of convergence behavior in  $10^{-5}$  s under zero initial conditions in the ZNNCM model are shown in Figure 5.3.

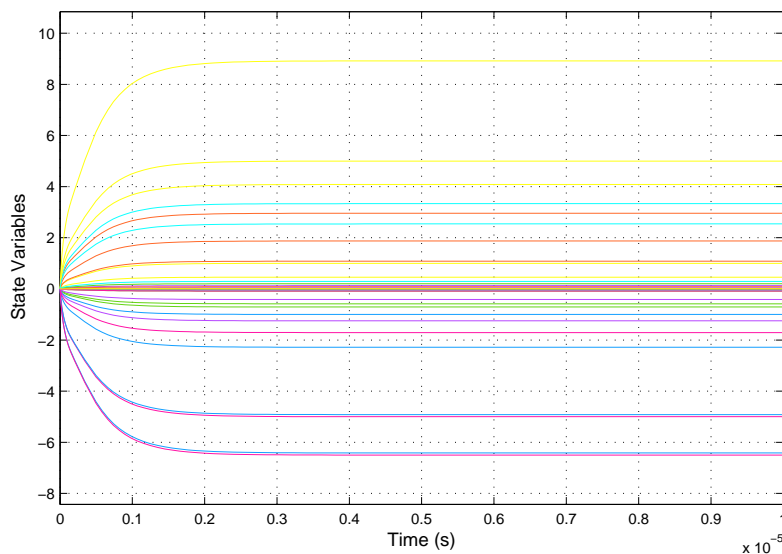


Figure 5.3: Trajectories in  $10^{-5}$  seconds under zero initial conditions in the ZNNCM model

Trajectories of the residual errors  $\|X(t) - A^{-1}\|_1$  of the model ZNNCM are illustrated in Figure 5.4.

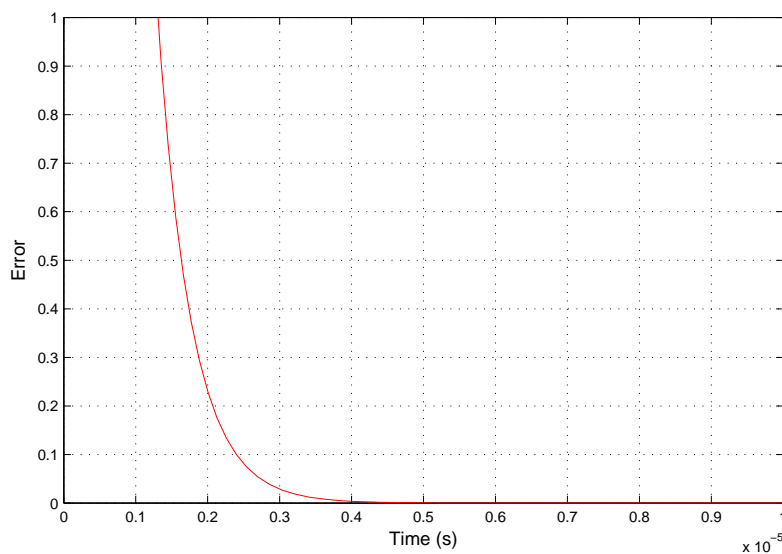


Figure 5.4: Trajectories of the residual errors of the model ZNNCM.

Trajectories of the residual errors  $\|X(t) - A^{-1}\|_1$  of both the ZNNCM and ZNNNM models are illustrated in Figure 5.5.

The ZNNNM method produces the result with the absolute error  $\|X(t) - A^{-1}\|_1 = 3.5446017745966e-08$  while the GNN model corresponding to the usual matrix inversion, from [94] does not achieve the convergence and stops with the absolute error equal to  $\|X(t) - A^{-1}\|_1 = 19.509081114657$ . So, the ZNNCM model can be used to compute the inverses of ill-conditioned matrices. This is one more advantage for the ZNNCM model over the GNN model.



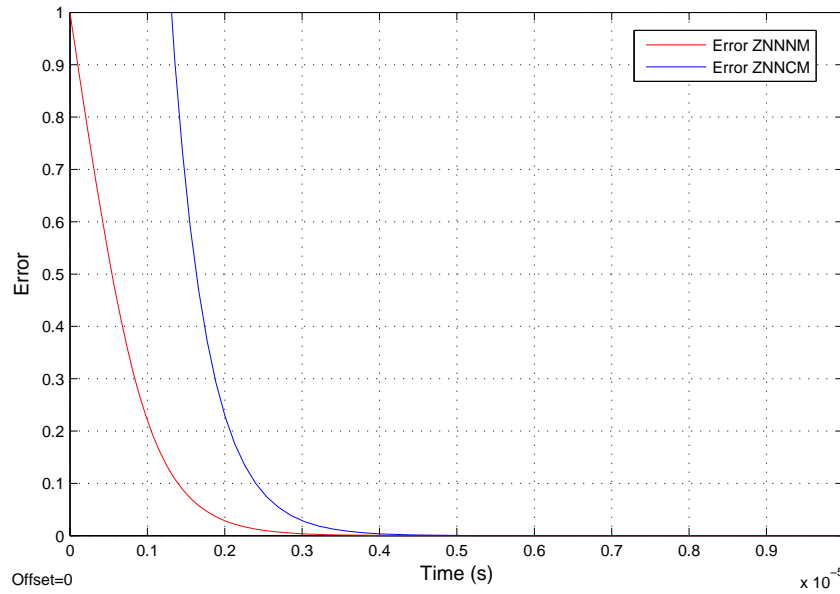


Figure 5.5: Trajectories of the residual errors of the models ZNNNM and ZNNCM.

In the subsequent examples, the matrix  $A$  is a randomly generated  $n \times n$  matrix and  $x(0)$  is a vectorization of a given  $n \times n$  matrix  $X(0)$ . It is assumed that  $x(0)$  is the same for all models ZNNNM, ZNNCM and GNN in the actual table. The ordered triple  $(t, n, solver)$  in headings of subsequent tables will include the time  $t$ , the dimension  $n$  of the input matrix and the used *Matlab* solver. Let us mention that the best results in all tables are marked in bold.

**Example 5.6.2.** According to Theorem 5.5.1 and Theorem 5.5.2, the solution of the complex neural network model (5.3.10) is stable in the sense of the Lyapunov. Therefore, it is desirable to choose the zero initial state  $X(0)$ . In this example,  $X(0)$  is randomly generated  $n \times n$  matrix in order to test behavior of the ZNNCM model. The activation function  $\mathcal{H}$  is linear.

Table 5.1: Comparison of the models ZNNNM, ZNNCM and GNN.

$(10^{-9}, 10, ode45)$			$(10^{-11}, 10, ode45)$		
Method	$\gamma$	$\ X - A^{-1}\ _1$	Method	$\gamma$	$\ X - A^{-1}\ _1$
ZNNNM	$10^8$	9.6055	ZNNNM	$10^8$	10.6051
<b>ZNNCM</b>	$10^8$	8.6367	<b>ZNNCM</b>	$10^8$	10.5938
GNN	$10^8$	10.4382	GNN	$10^8$	10.6126
<b>ZNNNM</b>	$10^9$	3.9053	ZNNNM	$10^9$	10.5101
ZNNCM	$10^9$	8.7346	<b>ZNNCM</b>	$10^9$	10.3980
GNN	$10^9$	9.8657	GNN	$10^9$	10.5877
<b>ZNNNM</b>	$10^{10}$	$4.8369e-04$	ZNNNM	$10^{10}$	9.6055
ZNNCM	$10^{10}$	0.0013	<b>ZNNCM</b>	$10^{10}$	8.6367
GNN	$10^{10}$	8.4838	GNN	$10^{10}$	10.4382

Table 5.1: Comparison of the models ZNNNM, ZNNCM and GNN.

$(10^{-9}, 10, ode45)$			$(10^{-11}, 10, ode45)$		
Method	$\gamma$	$\ X - A^{-1}\ _1$	Method	$\gamma$	$\ X - A^{-1}\ _1$
<b>ZNNNM</b>	$10^{11}$	$2.8613e-06$	<b>ZNNNM</b>	$10^{11}$	$3.9053$
ZNNCM	$10^{11}$	$7.9558e-05$	ZNNCM	$10^{11}$	$8.7346$
GNN	$10^{11}$	$2.1187$	GNN	$10^{11}$	$9.8657$
ZNNNM	$10^{12}$	$1.0572e-05$	<b>ZNNNM</b>	$10^{12}$	$4.8369e-04$
ZNNCM	$10^{12}$	$3.4264e-05$	ZNNCM	$10^{12}$	$0.0013$
<b>GNN</b>	$10^{12}$	$9.8238e-06$	GNN	$10^{12}$	$8.4838$
ZNNNM	$10^{13}$	$2.1264e-05$	<b>ZNNNM</b>	$10^{13}$	$2.8613e-06$
ZNNCM	$10^{13}$	$0.0010$	ZNNCM	$10^{13}$	$7.9558e-05$
<b>GNN</b>	$10^{13}$	$5.7854e-06$	GNN	$10^{13}$	$2.1187$
<b>ZNNNM</b>	$10^{14}$	$2.0846e-06$	ZNNNM	$10^{14}$	$1.0572e-05$
ZNNCM	$10^{14}$	$0.0015$	ZNNCM	$10^{14}$	$3.4264e-05$
GNN	$10^{14}$	$3.3838e-06$	<b>GNN</b>	$10^{14}$	$9.8238e-06$

According to the results arranged in Table 5.1, the following observations could be emphasized:

- The property "as large as possible" of the scaling parameter  $\gamma$  is valid for ZNNNM and GNN methods, especially for the GNN method, and it is not applicable in the case of the ZNNCM method,
- The ZNNCM method produces better results within the smaller time period  $[0, 10^{-11}]$  than in the time period  $[0, 10^{-9}]$ ,
- The ZNNCM method produces the best results during the time  $[0, 10^{-11}]$  and smaller values of  $\gamma$ :  $\gamma = 10^8, 10^9, 10^{10}$ ,
- In the case of a nonzero initial state  $X(0)$ , the ZNNCM method should be used in a short time  $[0, 10^{-11}]$  and with smaller values of  $\gamma \leq 10^{10}$ .

**Example 5.6.3.** In this example,  $X(0)$  is randomly generated  $n \times n$  matrix and the activation function  $\mathcal{H}$  is linear.

Table 5.2: Comparison of the models ZNNNM, ZNNCM and ZNNHM.

$(10^{-11}, 30, ode45)$			$(10^{-11}, 30, ode15s)$		
Method	$\gamma$	$\ X - A^{-1}\ _1$	Method	$\gamma$	$\ X - A^{-1}\ _1$
ZNNNM	$10^{10}$	$12.6304$	<b>ZNNNM</b>	$10^{10}$	$16.3735$
ZNNCM	$10^{10}$	$11.9629$	ZNNCM	$10^{10}$	$1.8502e + 03$
<b>ZNNHM</b> ( $10^{-12}$ )	$10^{10}$	$11.2585$	ZNNHM( $10^{-12}$ )	$10^{10}$	$19.4103$

Table 5.2: Comparison of the models ZNNNM, ZNNCM and ZNNHM.

$(10^{-11}, 30, ode45)$			$(10^{-11}, 30, ode15s)$		
Method	$\gamma$	$\ X - A^{-1}\ _1$	Method	$\gamma$	$\ X - A^{-1}\ _1$
ZNNHM( $10^{-14}$ )	$10^{10}$	11.4939	ZNNHM( $10^{-14}$ )	$10^{10}$	18.2060
ZNNNM	$10^{11}$	5.1351	ZNNNM	$10^{11}$	6.6561
ZNNCM	$10^{11}$	$5.1555e + 15$	ZNNCM	$10^{11}$	7.6935
ZNNHM( $10^{-12}$ )	$10^{11}$	4.7521	ZNNHM( $10^{-12}$ )	$10^{11}$	680.4448
ZNNHM( $10^{-14}$ )	$10^{11}$	4.6613	ZNNHM( $10^{-14}$ )	$10^{11}$	7.4102
ZNNNM	$10^{12}$	$6.3565e - 04$	ZNNNM	$10^{12}$	$8.2654e - 04$
ZNNCM	$10^{12}$	$6.1391e + 15$	ZNNCM	$10^{12}$	$4.7134e - 04$
ZNNHM( $10^{-12}$ )	$10^{12}$	$2.7946e + 11$	ZNNHM( $10^{-12}$ )	$10^{12}$	$7.7231e - 04$
ZNNHM( $10^{-14}$ )	$10^{12}$	$5.6655e - 04$	ZNNHM( $10^{-14}$ )	$10^{12}$	$9.6796e - 04$
ZNNNM	$10^{13}$	$6.7814e - 06$	ZNNNM	$10^{13}$	$5.4011e - 12$
ZNNCM	$10^{13}$	$1.0634e + 16$	ZNNCM	$10^{13}$	$1.7082e - 13$
ZNNHM( $10^{-12}$ )	$10^{13}$	$2.7107e - 05$	ZNNHM( $10^{-12}$ )	$10^{13}$	$3.7240e - 11$
ZNNHM( $10^{-14}$ )	$10^{13}$	$6.2916e - 06$	ZNNHM( $10^{-14}$ )	$10^{13}$	$1.9452e - 10$
ZNNHM( $10^{-16}$ )	$10^{13}$	$3.5050e - 06$	ZNNHM( $10^{-16}$ )	$10^{13}$	$3.0346e - 12$
ZNNNM	$10^{14}$	$1.9529e - 05$	ZNNNM	$10^{14}$	$3.2919e - 14$
ZNNCM	$10^{14}$	$5.1555e + 15$	ZNNCM	$10^{14}$	$8.8654e - 14$
ZNNHM( $10^{-12}$ )	$10^{14}$	$7.3399e - 05$	ZNNHM( $10^{-12}$ )	$10^{14}$	$3.8212e - 14$
ZNNHM( $10^{-14}$ )	$10^{14}$	$7.3399e - 05$	ZNNHM( $10^{-14}$ )	$10^{14}$	$3.6538e - 14$
ZNNHM( $10^{-16}$ )	$10^{14}$	$6.1610e - 06$	ZNNHM( $10^{-16}$ )	$10^{14}$	$4.2286e - 14$

The following observations rise from the numerical results arranged in Table 5.2:

- The hybrid method ZNNHM produces the best results in the case when the underlying solver is *ode45*,
- the ZNNNM or ZNNCM give the best results in the case when the underlying solver is *ode15s*,
- The solver *ode15s* is more appropriate than *ode45* with respect to the ZNNCM method,
- ZNNCM model is sensitive on the choice of the initial point  $X_0$  and the best choice is  $X_0 = 0$ .

**Example 5.6.4.** In the left column of Table 5.3 this example,  $X(0)$  is randomly generated  $n \times n$  matrix and  $X(0)$   $n \times n$  zero matrix in the right column. The activation function  $\mathcal{H}$  is Power Sigmoid activation function.

Table 5.3: Comparison of the models ZNNNM, ZNNCM and ZNNHM with Power Sigmoid activation function  $\mathcal{H}$ , defined by the parameter  $p = 3$ .

$(10^{-7}, 20, ode15s), X_0$ is arbitrary			$(10^{-7}, 30, ode15s), X_0 = 0$		
Method	$\gamma$	$\ X - A^{-1}\ _1$	Method	$\gamma$	$\ X - A^{-1}\ _1$
ZNNNM	$10^{12}$	$2.7645e - 14$	<b>ZNNNM</b>	$10^{12}$	$1.4903e - 12$
ZNNCM	$10^{12}$	$4.3998e - 14$	ZNNCM	$10^{12}$	$6.1530e - 12$
<b>ZNNHM</b> ( $10^{-12}$ )	$10^{12}$	$2.3267e - 14$	ZNNHM( $10^{-12}$ )	$10^{12}$	$1.6522e - 12$
ZNNHM( $10^{-14}$ )	$10^{12}$	$2.7345e - 14$	ZNNHM( $10^{-14}$ )	$10^{12}$	$1.7446e - 12$
ZNNHM( $10^{-16}$ )	$10^{12}$	$2.9818e - 14$	ZNNHM( $10^{-16}$ )	$10^{12}$	$1.6834e - 12$
<b>ZNNNM</b>	$10^{13}$	$2.6716e - 14$	ZNNNM	$10^{13}$	$1.7769e - 12$
ZNNCM	$10^{13}$	$4.5023e - 14$	ZNNCM	$10^{13}$	$4.1402e - 12$
ZNNHM( $10^{-12}$ )	$10^{13}$	$3.1053e - 14$	ZNNHM( $10^{-12}$ )	$10^{13}$	$1.7335e - 12$
ZNNHM( $10^{-14}$ )	$10^{13}$	$2.7970e - 14$	<b>ZNNHM</b> ( $10^{-14}$ )	$10^{13}$	$1.6732e - 12$
ZNNHM( $10^{-16}$ )	$10^{13}$	$2.8306e - 14$	ZNNHM( $10^{-16}$ )	$10^{13}$	$2.0054e - 12$
ZNNNM	$10^{14}$	$2.7645e - 14$	ZNNNM	$10^{14}$	$5.3491e - 12$
ZNNCM	$10^{14}$	$4.3998e - 14$	ZNNCM	$10^{14}$	$5.4503e - 12$
<b>ZNNHM</b> ( $10^{-12}$ )	$10^{14}$	$2.3267e - 14$	<b>ZNNHM</b> ( $10^{-12}$ )	$10^{14}$	$1.9956e - 12$
ZNNHM( $10^{-14}$ )	$10^{14}$	$2.7345e - 14$	ZNNHM( $10^{-14}$ )	$10^{14}$	$2.0058e - 12$
ZNNHM( $10^{-16}$ )	$10^{14}$	$2.9818e - 14$	ZNNHM( $10^{-16}$ )	$10^{14}$	$2.4750e - 12$

 Table 5.4: Comparison of the models ZNNNM, ZNNCM and ZNNHM with Power Sigmoid activation function defined by the parameter  $p = 3$ .

$(10^{-10}, 30, ode15s), X_0 = 0$			$(10^{-5}, 30, ode15s), X_0 = 0$		
Method	$\gamma$	$\ X - A^{-1}\ _1$	Method	$\gamma$	$\ X - A^{-1}\ _1$
<b>ZNNNM</b>	$10^{12}$	$1.2136e - 12$	ZNNNM	$10^{12}$	$1.6829e - 12$
ZNNCM	$10^{12}$	$1.8007e - 09$	ZNNCM	$10^{12}$	$3.7154e - 12$
ZNNHM( $10^{-12}$ )	$10^{12}$	$1.2410e - 11$	ZNNHM( $10^{-12}$ )	$10^{12}$	$2.3492e - 12$
ZNNHM( $10^{-14}$ )	$10^{12}$	$1.8332e - 12$	<b>ZNNHM</b> ( $10^{-14}$ )	$10^{12}$	$1.3554e - 12$
ZNNHM( $10^{-16}$ )	$10^{12}$	$1.8565e - 12$	ZNNHM( $10^{-16}$ )	$10^{12}$	$1.8953e - 12$
ZNNNM	$10^{13}$	$1.6428e - 12$	ZNNNM	$10^{13}$	$1.9011e - 12$
ZNNCM	$10^{13}$	$4.4561e - 11$	ZNNCM	$10^{13}$	$5.9375e - 12$
ZNNHM( $10^{-12}$ )	$10^{13}$	$1.7544e - 12$	ZNNHM( $10^{-12}$ )	$10^{13}$	$1.7805e - 12$
<b>ZNNHM</b> ( $10^{-14}$ )	$10^{13}$	$1.5256e - 12$	<b>ZNNHM</b> ( $10^{-14}$ )	$10^{13}$	$1.7413e - 12$
ZNNHM( $10^{-16}$ )	$10^{13}$	$1.6547e - 12$	ZNNHM( $10^{-16}$ )	$10^{13}$	$2.0168e - 12$
ZNNNM	$10^{14}$	$1.7526e - 12$	ZNNNM	$10^{14}$	$1.9145e - 12$
ZNNCM	$10^{14}$	$4.4716e - 12$	ZNNCM	$10^{14}$	$4.6704e - 12$

Table 5.4: Comparison of the models ZNNM, ZNNCM and ZNNHM with Power Sigmoid activation function defined by the parameter  $p = 3$ .

$(10^{-10}, 30, ode15s), X_0 = 0$			$(10^{-5}, 30, ode15s), X_0 = 0$		
Method	$\gamma$	$\ X - A^{-1}\ _1$	Method	$\gamma$	$\ X - A^{-1}\ _1$
ZNNHM( $10^{-12}$ )	$10^{14}$	$1.7646e - 12$	ZNNHM( $10^{-12}$ )	$10^{14}$	$1.9448e - 12$
<b>ZNNHM</b> ( $10^{-14}$ )	$10^{14}$	$1.2410e - 12$	<b>ZNNHM</b> ( $10^{-14}$ )	$10^{14}$	$1.5412e - 12$
ZNNHM( $10^{-16}$ )	$10^{14}$	$2.0978e - 12$	ZNNHM( $10^{-16}$ )	$10^{14}$	$1.9367e - 12$

The following conclusion arises from the results presented in Table 5.3 and Table 5.4:

(a) The ZNNHM method gives best results for appropriately selected intermediate time  $t_0$ . This value is, in most cases, equal to  $t_0 = 10^{-14}$ .

**Example 5.6.5.** The results produced by the Simulink and based on based on the power sigmoid activation function are arranged in Table 5.5.

Table 5.5: ZNNM vs ZNNCM using the power sigmoid activation function.

Method	$\gamma$	$\ X - A^{-1}\ _1$	Method	$\gamma$	$\ X - A^{-1}\ _1$
$(10^{-8}, 10, ode45), X_0 = 0$			$(10^{-8}, 10, ode15s), X_0 = 0$		
ZNNM	$10^6$	0.99000742007452	ZNNM	$10^6$	0.99000742007452
ZNNCM	$10^6$	11.796704121711	ZNNCM	$10^6$	11.796704121712
<b>ZNNHM</b> ( $10^{-10}$ )	$10^6$	0.98921005121423	<b>ZNNHM</b> ( $10^{-10}$ )	$10^6$	0.98921005121423
$(10^{-5}, 10, ode45), X_0 = 0$			$(10^{-5}, 10, ode15s), X_0 = 0$		
ZNNM	$10^6$	$1.7721525005302e - 09$	ZNNM	$10^6$	$1.7721525005302e - 09$
ZNNCM	$10^6$	$8.9282757020914e - 09$	ZNNCM	$10^6$	$8.9286161242264e - 09$
ZNNHM( $10^{-8}$ )	$10^6$	$1.5373390357755e - 09$	ZNNHM( $10^{-8}$ )	$10^6$	$1.5373790308806e - 09$
$(10^{-3}, 10, ode45), X_0 = 0$			$(10^{-3}, 10, ode15s), X_0 = 0$		
ZNNM	$10^6$	$2.7877256580469e - 15$	ZNNM	$10^6$	$2.776852376599e - 15^*$
ZNNCM	$10^6$	$9.048317650695e - 15$	ZNNCM	$10^6$	$7.549516567451e - 15$
<b>ZNNHM</b> ( $10^{-5}$ )	$10^6$	$2.7947546454768e - 15$	<b>ZNNHM</b> ( $10^{-5}$ )	$10^6$	$2.140363747233e - 15$

The star sign in Table 5.5 means that the ZNNM model stopped the computation with the message:

"Relative tolerance of 1.0E-15 is too small, setting relative tolerance to 2.8421709430404007E-14".



## Chapter 6

# Matlab simulation of the hybrid neural dynamics for online matrix inversion

*A novel kind of a hybrid recursive neural implicit dynamics for real-time matrix inversion has been recently proposed and exploited. It was shown that, comparing a hybrid recursive neural implicit dynamics on the one hand, and conventional explicit neural dynamics on the other hand, a hybrid model can coincide better with systems in practice and has higher abilities in representing dynamic systems. More importantly, hybrid model can achieve superior convergence performance in comparison with the existing dynamic systems, specifically recently-proposed Zhang dynamics. This chapter presents the Simulink model of a hybrid recursive neural implicit dynamics and gives a simulation and comparison to the existing Zhang dynamics for real-time matrix inversion. A simulation results confirm a superior convergence of the hybrid model compared to Zhang model.*

### 6.1 Preliminaries and motivation

A new type of complex-valued recurrent neural networks, called Zhang neural network (ZNN), was proposed in 2001 and has been extensively exploited in solving various time-varying complex generalized inverse problems. The design of complex ZNN models arises from the choice of a complex matrix-valued error-monitoring function, called the Zhang function (ZF). Computation of the Moore-Penrose inverse of time-varying full-rank matrix by means of different ZNN models were investigated in [127]. Liao and Zhang in [48] proposed five different complex ZFs and, accordingly developed and investigated five different complex ZNN models for computing the time-varying complex pseudoinverse.

In this chapter we present a Matlab Simulink model based on the hybrid recurrent neural networks for computing inverse of nonsingular matrix [10]. This Simulink model was proposed in [137].

The following defining equation of matrix inverse  $A^{-1} \in \mathbb{R}^{n \times n}$  can be given:

$$AX(t) - I = 0 \quad (6.1.1)$$

or

$$X(t)A - I = 0, \quad (6.1.2)$$

where  $I \in \mathbb{R}^{n \times n}$  is the identity matrix, and  $X(t) \in \mathbb{R}^{n \times n}$  denotes the unknown matrix to be inverted which corresponds to the theoretical inverse  $A^{-1}$ .

Some known results related to the gradient-based dynamics and implicit Zhang dynamics as well as improved hybrid model are restated in Section 6.2. In Section 6.3 we present the corresponding Matlab Simulink model of improved hybrid dynamics along with simulation examples and comparison.

## 6.2 Model formulation

We assume invertible condition for matrix inversion : Equation (6.1.1) (or (6.1.2) in dual case) has a unique solution if the minimal eigenvalue  $\lambda$  of matrix  $A^T A$  is larger than 0.

### 6.2.1 Gradient-based dynamics

The dynamics of the gradient neural network (GNN) models for computing inverses are based on the usage of the scalar-valued norm-based error function

$$\varepsilon(t) = \varepsilon(X(t)) = \frac{1}{2} \|E(t)\|_F^2, \quad (6.2.1)$$

where  $E(t)$  is an appropriate error matrix and  $\|A\|_F := \sqrt{\text{Tr}(A^T A)}$  denotes the Frobenius norm of the matrix  $A$  and  $\text{Tr}(\cdot)$  denotes the trace of a matrix. The general design formula is typically defined along the negative gradient  $-\partial\varepsilon(X(t))/\partial X$  of  $\varepsilon(X(t))$  until the minimum is reached. Using the above negative gradient to construct the neural dynamics, we could have the gradient-based dynamics as follows

$$\frac{dX(t)}{dt} = -\gamma \mathcal{F} \left( \frac{\partial\varepsilon(X(t))}{\partial X} \right). \quad (6.2.2)$$

The scaling real parameter  $\gamma$  in (6.2.2) is used to adjust the convergence rate and could be chosen as large as possible in order to accelerate the convergence. Further,  $\mathcal{F}(C)$  is an odd and monotonically increasing function array, element-wise applicable to elements of a real matrix  $C = (c_{ij}) \in \mathbb{R}^{n \times n}$ , i.e.  $\mathcal{F}(C) = (f(c_{ij}))$ ,  $i = 1, \dots, m$ ,  $j = 1, \dots, n$ , wherein  $f(\cdot)$  is an odd



and monotonically increasing function.

The dynamic equation of the linear recurrent neural network for the inversion of a real nonsingular matrix is initiated by the error matrix  $E(t) = AX(t) - I$ , and it was proposed in [94]:

$$\frac{dX(t)}{dt} = -\gamma A^T AX(t) + \gamma A^T = -\gamma A^T (AX(t) - I). \quad (6.2.3)$$

The same principle was extended for computing the Moore–Penrose inverse of a full-column rectangular matrix  $A \in \mathbb{R}_n^{m \times n}$  or a full-row rectangular matrix  $A \in \mathbb{R}_m^{m \times n}$ . Wang showed in [96] that the model can be used for computing the Moore–Penrose inverse of rank-deficient matrices under the zero initial condition,  $V(0) = 0$ .

### 6.2.2 Zhang dynamics

On the other hand, the ZNN model for online time-invariant matrix inversion is based upon the matrix-formed error function  $E(t)$ , instead of a scalar valued function. The time derivative of error function  $E(t)$ , should be chosen such that each element  $e_{ij}(t)$  of  $E(t)$  converges to zero,  $\forall i = 1, \dots, n$ . A general design rule of  $\dot{E}(t)$  is defined

$$\dot{E}(t) = \frac{dE(t)}{dt} = -\gamma \mathcal{F}(E(t)). \quad (6.2.4)$$

Substituting  $E(t)$  into dynamic system (6.2.4) and choosing  $\mathcal{F}$  to be linear function, the following Zhang dynamics for online matrix inversion can be obtained:

$$A\dot{X} = -\gamma AX(t) + \gamma I \quad (6.2.5)$$

The implicit dynamics were originally proposed for online inversion of a time-varying matrix  $A(t)$  in [122]. It was shown in [122] that the Zhang dynamics (6.2.6) globally exponentially converges to the theoretical inverse  $A^{-1}$ , starting from any initial state  $X(0)$ , with the exponential convergence rate  $\gamma$ .

### 6.2.3 Improved ZNN model for matrix inversion

The ZNN model for online time-invariant matrix inversion is based upon the error matrix  $E(t) = AX(t) - I$ , and it is defined using the general design rule

$$\dot{E}(t) = -\gamma E(t) = -\gamma(AX(t) - I),$$

which initiates the implicit dynamics

$$A\dot{X}(t) = -\gamma(AX(t) - I). \quad (6.2.6)$$

A gradient-based recurrent neural dynamics for real-time inverse of a time-invariant matrix was proposed by Wang [94] in the form of an explicit dynamic system

$$\dot{X}(t) = -\gamma A^T AX(t) + \gamma A^T. \quad (6.2.7)$$

The above explicit gradient-based recursive dynamics (6.2.7) can be transformed into an implicit form

$$A\dot{X}(t) = -\gamma AA^T(AX(t) - I). \quad (6.2.8)$$

Recently, a novel kind of recurrent implicit dynamics for real-time matrix inversion was proposed and investigated in [10, 11]. This hybrid model can be obtain by combining the right hand side of both Zhang dynamics (6.2.6) and gradient dynamics (6.2.8)

$$A\dot{X}(t) = -\gamma(AA^T + I)(AX(t) - I). \quad (6.2.9)$$

Global exponential convergence rate of the implicit dynamics (6.2.9) was investigated in [10, 11].

**Theorem 6.2.1.** [10] *Given nonsingular matrix  $A \in \mathbb{R}^{n \times n}$ , the state matrix  $A \in \mathbb{R}^{n \times n}$  of the model (6.2.9), starting from any initial state  $X(0) \in \mathbb{R}^{n \times n}$ , achieves global exponential convergence to theoretical inverse  $X^* = A^{-1}$ . In addition, the exponential convergence rate is the product of  $\gamma$  and the minimum eigenvalue  $\gamma > 1$  of  $A^T A + I$ .*

### 6.3 Simulation results and its comparison

The graphical editor and customizable block libraries available in Matlab Simulink tool were used in [137] for simulating and comparing the Zhang dynamic system (6.2.6) and recently proposed hybrid dunamic system (6.2.9). Simulink implementation of (6.2.6) was described in [87]. The models (6.2.6) and (6.2.9) will be termed as *ZNNNM* and *EZNNNM*.

The Simulink implementation of the hybrid model (6.2.9) is based on the equivalent form given by

$$\dot{X}(t) = (I - A)\dot{X}(t) - \gamma(AA^T + I)(AX(t) - I). \quad (6.3.1)$$

and is presented in Figure 6.1. For solving differential equations in the models we used *ode15s* solver.

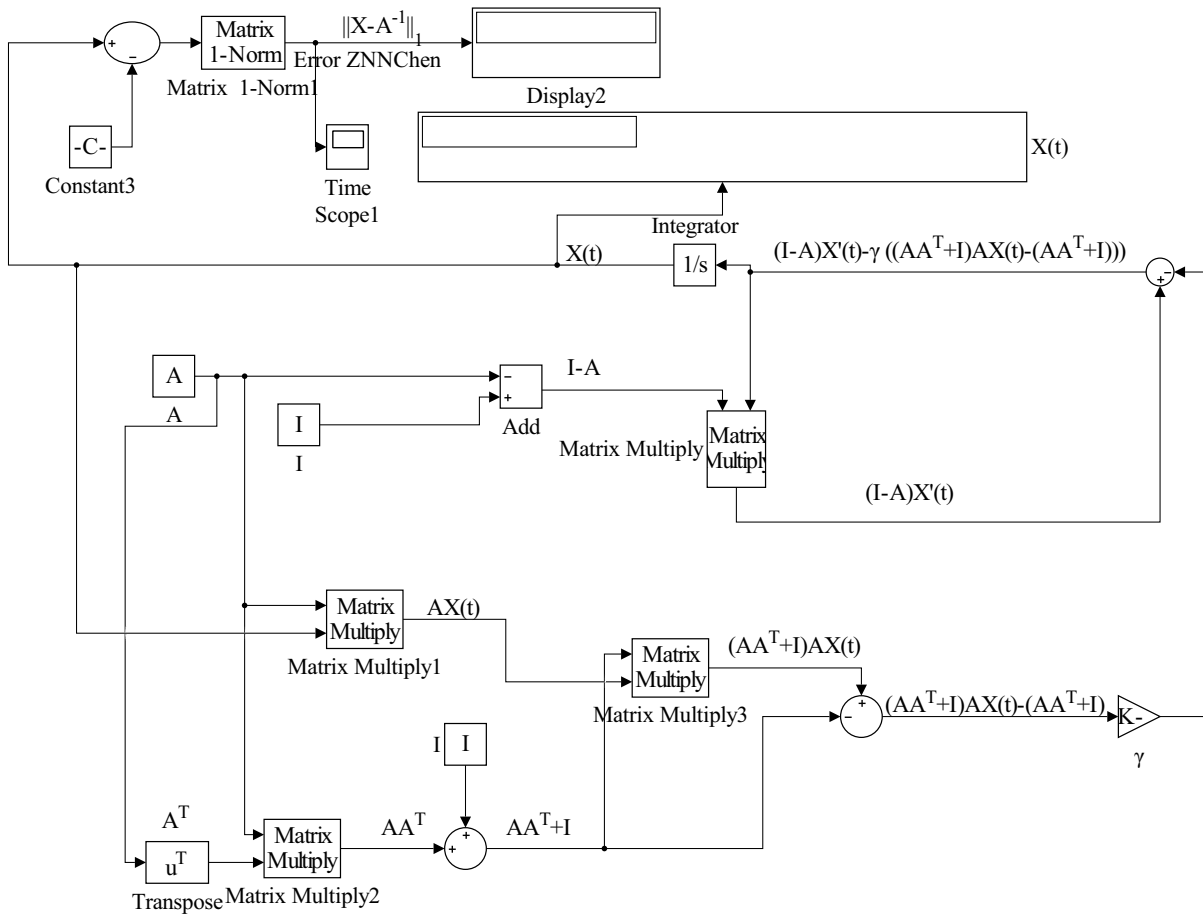


Figure 6.1: Simulink implementation of EZNNNM model.

The next examples will compare performances of both ZNNNM and EZNNNM models.

**Example 6.3.1. (a)** Consider the following matrix

$$A = \begin{bmatrix} 0.8147 & 0.1576 & 0.6557 & 0.7060 & 0.4387 & 0.2760 & 0.7513 & 0.8407 & 0.3517 & 0.0759 \\ 0.9058 & 0.9706 & 0.0357 & 0.0318 & 0.3816 & 0.6797 & 0.2551 & 0.2543 & 0.8308 & 0.0540 \\ 0.1270 & 0.9572 & 0.8491 & 0.2769 & 0.7655 & 0.6551 & 0.5060 & 0.8143 & 0.5853 & 0.5308 \\ 0.9134 & 0.4854 & 0.9340 & 0.0462 & 0.7952 & 0.1626 & 0.6991 & 0.2435 & 0.5497 & 0.7792 \\ 0.6324 & 0.8003 & 0.6787 & 0.0971 & 0.1869 & 0.1190 & 0.8909 & 0.9293 & 0.9172 & 0.9340 \\ 0.0975 & 0.1419 & 0.7577 & 0.8235 & 0.4898 & 0.4984 & 0.9593 & 0.3500 & 0.2858 & 0.1299 \\ 0.2785 & 0.4218 & 0.7431 & 0.6948 & 0.4456 & 0.9597 & 0.5472 & 0.1966 & 0.7572 & 0.5688 \\ 0.5469 & 0.9157 & 0.3922 & 0.3171 & 0.6463 & 0.3404 & 0.1386 & 0.2511 & 0.7537 & 0.4694 \\ 0.9575 & 0.7922 & 0.6555 & 0.9502 & 0.7094 & 0.5853 & 0.1493 & 0.6160 & 0.3804 & 0.0119 \\ 0.9649 & 0.9595 & 0.1712 & 0.0344 & 0.7547 & 0.2238 & 0.2575 & 0.4733 & 0.5678 & 0.3371 \end{bmatrix}$$

This matrix has minimum eigenvalue  $\alpha = 0.0154$  of  $A^T A$ . We compare the linear ZNNNM

*EZNNNM* model with the gain parameter  $\gamma = 10^6$ . The initial matrix is chosen by  $V(0) = 0$ . Figure 6.2 (right) shows the trajectories of the error norms  $\|A^{-1} - X(t)\|$  in the total simulation time  $t_{tot} = 10^{-5}$ . Figure 6.2 (left) shows the trajectories of the error norms  $\|A^{-1} - X(t)\|$  in the total simulation time  $t_{tot} = 10^{-6}$ .

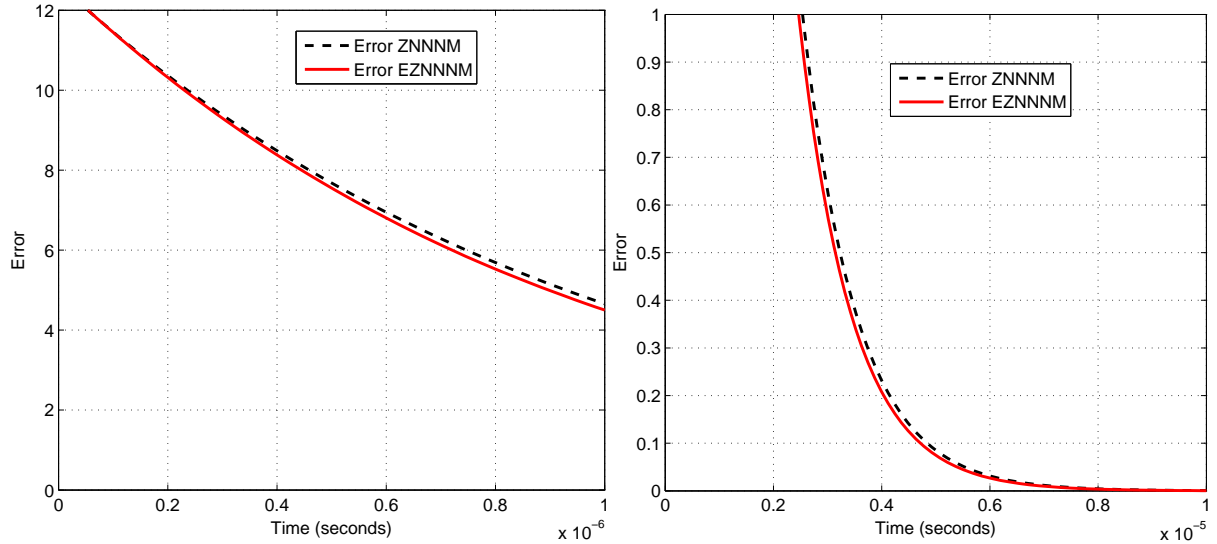


Figure 6.2: Trajectories of the errors  $\|A^{-1} - X(t)\|$  of *ZNNNM* and *EZNNNM* in Example 6.3.1.

In general, Figure 6.2 shows that *EZNNNM* model slightly outperforms the *ZNNNM* model. Both models generate almost identical residual norms in the initial phase and then *EZNNNM* generates a bit smaller residual norms. According to Figure 6.2, the *EZNNNM* model possesses a bit faster convergence.

**(b)** Now, consider matrix  $A_1 = 5I + A$ . This matrix has  $\alpha = 16.6813$  which is quite larger than  $\alpha$  in the previous example. We apply the linear *EZNNNM* model with the gain parameter  $\gamma = 10^6$ .

Figure 6.3 (right) shows the trajectories of the error norm  $\|A^{-1} - X(t)\|$  in the total simulation time  $t_{tot} = 10^{-5}$ . Figure 6.3 (left) shows the trajectories of the error norm  $\|A^{-1} - X(t)\|$  in the total simulation time  $t_{tot} = 10^{-6}$ .

According to Figure 6.3, the *EZNNNM* model possesses faster convergence.

**Example 6.3.2.** Consider the matrix

$$A = \begin{bmatrix} 20 & 6 \\ -1 & 30 \end{bmatrix}$$

which satisfies  $\alpha = 386.2656$ . Elements of the matrix  $X(t)$  generated by the model *EZNNNM* are denoted by  $x_{ij}^{EZNNNM}$ . Similarly, elements of the matrix  $X(t)$  generated by the model

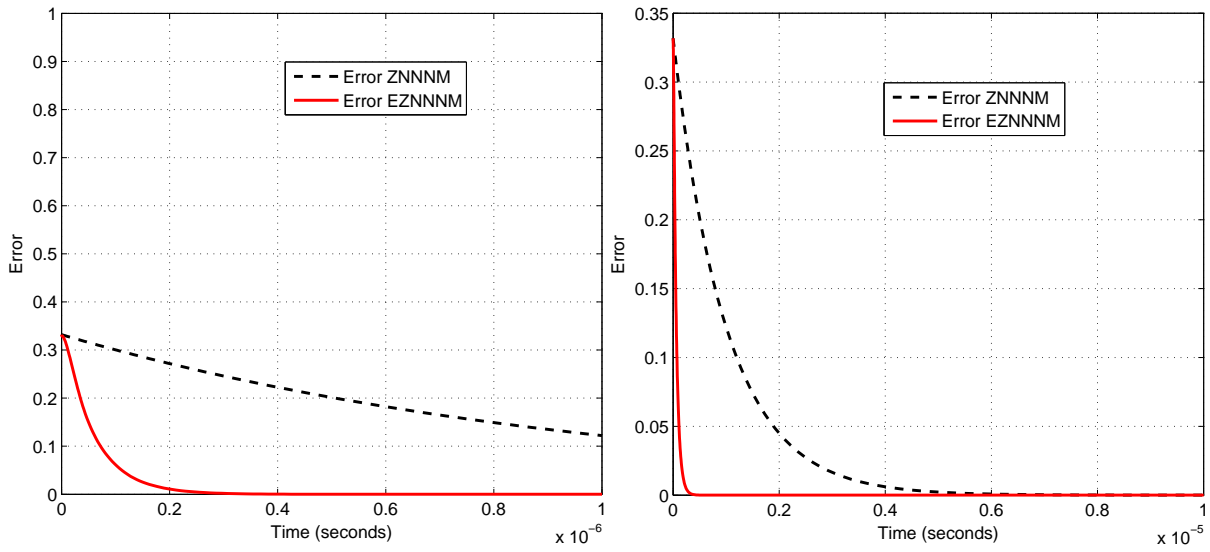


Figure 6.3: Trajectories of the errors  $\|A_1^{-1} - X(t)\|$  of  $ZNNM$  and  $EZNNM$  in Example 6.3.1.

$ZNNM$  are denoted by  $x_{ij}^{ZNNM}$ . Trajectories of the elements of the matrices are presented in Figure 6.4.

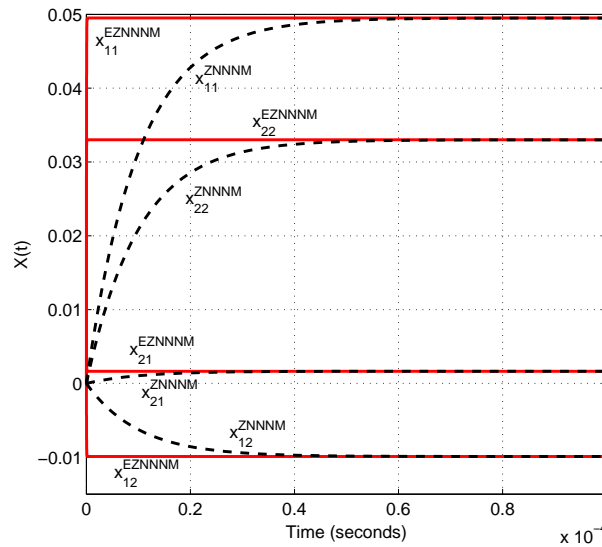


Figure 6.4: Trajectories of  $X(t)$  of  $ZNNM$  and  $EZNNM$  in Example 6.3.2.

Greater value  $\alpha$  initiates significantly faster convergence of  $EZNNM$  with respect to  $ZNNM$ .

**Example 6.3.3.** This example shows the influence of  $\alpha$  on the convergence of the  $ZNN$  models. The gain parameters in the simulation is  $\gamma = 10^6$  and the time period is  $[0, 10^{-5}]$  s. First, consider the following randomly generated matrix  $B$ :

$$B = \begin{bmatrix} 0.9631 & 0.6241 & 0.0377 & 0.2619 & 0.1068 \\ 0.5468 & 0.6791 & 0.8852 & 0.3354 & 0.6538 \\ 0.5211 & 0.3955 & 0.9133 & 0.6797 & 0.4942 \\ 0.2316 & 0.3674 & 0.7962 & 0.1366 & 0.7791 \\ 0.4889 & 0.9880 & 0.0987 & 0.7212 & 0.7150 \end{bmatrix}.$$

Then we are varying matrix  $A$  in the way that value  $\alpha$  becomes larger in every step, and testing both models on such matrix  $A$  in the order to find the error norm of the model results.

Table 6.1: Comparison of the models ZNNNM and EZNNNM

$A$	$\alpha$	$\ X^{ZNNNM} - A^{-1}\ _1$	$\ X^{EZNNNM} - A^{-1}\ _1$
$B$	0.0149	0.000594	0.000494
$B + I$	0.1729	0.000143	$2.7866e-05$
$B + 2I$	1.9360	$3.8564e-05$	$2.1079e-08$
$B + 3I$	5.6776	$2.2102e-05$	$1.4172e-09$
$B + 4I$	11.4133	$1.5414e-05$	$3.6347-10$
$B + 5I$	19.1467	$1.1757-05$	$1.5278-09$
$B + 10I$	87.8023	$5.2761e-06$	$1.0706e-11$
$B + 15I$	206.4530	$3.3656e-06$	$2.7424e-11$
$B + 20I$	375.1024	$2.4667e-06$	$5.7302e-12$
$B + 50I$	2437	$9.4123e-07$	$8.6352e-14$

From the Table 6.1 we can see that when the value of  $\alpha$  is grater EZNNNM model gives better accuracy of the solution related to ZNNNM model in the same given period of time.

**Example 6.3.4.** In this example we ask the answer for the question: is it possible to compensate advantage of the EZNNNM model using greater values  $\gamma$  in ZNNNM. For this purpose, we tested these models on the matrix

$$A = \begin{bmatrix} 5.8147 & 0.0975 & 0.1576 & 0.1419 & 0.6557 \\ 0.9058 & 5.2785 & 0.9706 & 0.4218 & 0.0357 \\ 0.1270 & 0.5469 & 5.9572 & 0.9157 & 0.8491 \\ 0.9134 & 0.9575 & 0.4854 & 5.7922 & 0.9340 \\ 0.6324 & 0.9649 & 0.8003 & 0.9595 & 5.6787 \end{bmatrix}$$

with the minimal eigenvalue of  $AA^T$  equal to 20.8356. The maximal possible value in Simulink model ZNNNM is  $10^7$  and in EZNNNM is  $10^6$ . Figure 6.5 (right) shows the trajectories of the error norm  $\|A^{-1} - X(t)\|$  in the total simulation time  $t_{tot} = 10^{-5}$ . Figure 6.5 (left) shows the trajectories of the error norm  $\|A^{-1} - X(t)\|$  in the total simulation time  $t_{tot} = 10^{-6}$ .

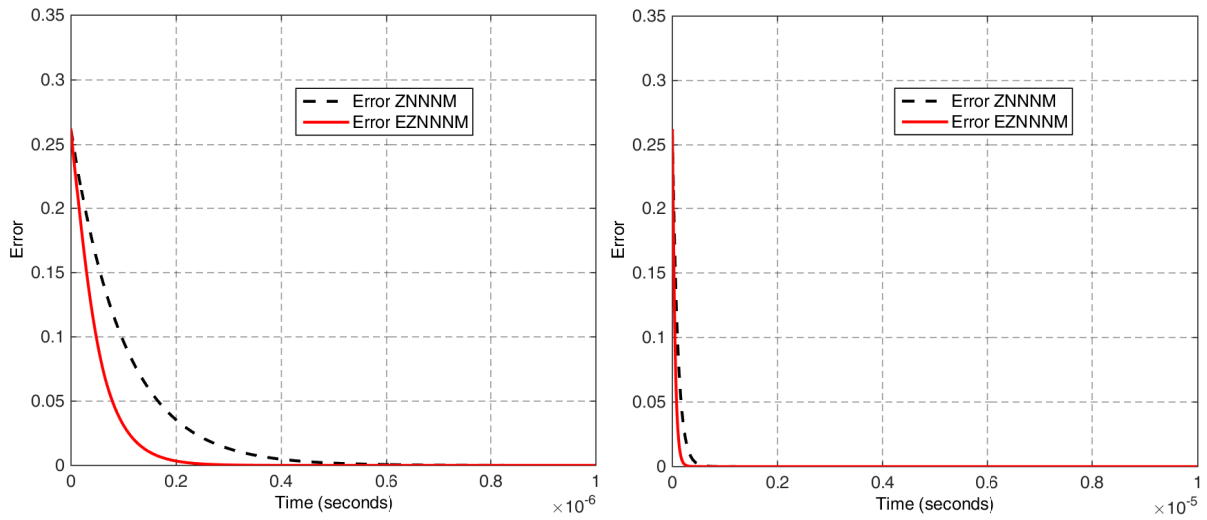


Figure 6.5: Trajectories of the errors  $\|A_1^{-1} - X(t)\|$  of  $ZNNM$  and  $EZNNM$  in Example 6.3.4.

According to Figure 6.3, the  $EZNNM$  model still possesses faster convergence. This means that greater values  $\gamma$  can not compensate faster convergence rate of the  $EZNNM$  model with respect to  $ZNNM$  model.





# Chapter 7

## Conclusion

Generalized inverses arise in various applications such as statistics, linear estimation, least squares approximation, singular differential and difference equations, singular control, Markov chains, ill-posed problems, graphics, cryptography, coding theory, incomplete data recovery, and robotics. One of the main motivation of this dissertation is based on the ability to exploit the correlation between the dynamic state equations of recurrent neural networks for computing generalized inverses and integral representations of these generalized inverses. We have focused particularly on the problem of generalized inverses computation as well as computation of regular inverses and matrix equations using artificial recurrent neural network approach. Here we summarize most important contributions.

We have introduced artificial recurrent neural network as a parallel distributed computational model for computing the generalized inverses and have followed the main principle which requires solving one representative matrix equation related to the considered generalized inverse via dynamic-system approach, defined over a certain norm based error function  $E(t)$ . It is necessary to find the minimum for the residual-norm function  $E(t)$  and  $dV(t)/dt = -\gamma \partial E(t)/\partial V$ . The stability of the neural networks is conditioned by the requirement of the given matrix spectrum. According to the spectrum requirement, we have studied two types of the GNNs: GNN for computing generalized inverses with restrictions on a spectrum and GNN for computing generalized inverses without restrictions on a spectrum. First, we have defined the recurrent neural network for the Drazin inverse of a singular and the inverse of a nonsingular matrix and then we have introduced more generalized neural network model for computing outer inverses with prescribed range and null space. Main generalized inverses, such as the Moore-Penrose and the weighted Moore-Penrose inverse, the Drazin inverse and the group inverse can be derived after appropriate particular choices. In addition, we have analyzed and proved the stability of such a solutions with regard to Lyapunov stability theory. These neural network models has proven to be capable of computing the considered generalized inverse. By reducing the number of interconnections required for the network, the models reduce the

complexity for computing the generalized inverse of a given matrix. These time-consuming computations occur in large-scale applications. The proposed neural network models are stable in the large and they are able to converge to the generalized inverse with a high accuracy and in a short period of time. Through computer-simulated illustrative examples, we have shown the proposed recurrent neural networks are able to generate generalized inverses of singular matrices at the projected convergence rate. Computer simulations have been done using Matlab environment.

We have investigated the conditions for the existence of outer inverse with prescribed range and null space as well as different representations of outer inverses. In the essence, these investigations showed equivalence between two important representations of outer inverse. Then we have applied obtained representations and introduced two dynamic state equations and initiated neural networks: RNN(4.2.3) and RNN(4.2.4). These RNNs are based on the full rank representation of the outer inverse.

We have observed an analogy between the scaled hyperpower family (SHPI family) of iterative methods for computing the matrix inverse and the discretization of Zhang Neural Network (ZNN) models. On the basis of the discovered analogy, we have defined a family of ZNN models corresponding to the family of hyperpower iterative methods. The ZNN model corresponding to the hyperpower method of the order 2 (resp. of the order 3) is denoted as ZNNNM (resp. ZNNCM). We have described the implementation of the introduced ZNN models in the case of the scaled hyperpower methods of the order 2 and 3 using Matlab Simulink toolbox.

Derived simulation results indicate that the results derived by the ZNNCM method are not favorable. But, the ZNNCM model becomes useful in the initialization of the ZNNNM method. For the time being, it is very difficult to determine or estimate the optimal value of the decisive time moment  $t_0$ . These investigations should be interesting topic for further research. In the current research, we recommend only heuristics and verification. Additionally, it is observable that ZNNCM is most sensible to the choice of the initial approximation  $X_0$ .

Also, general conclusion is that an approach to avoid usage of matrix iterations is defined. The proposed alternative is based on the ZNN model and its Simulink implementation.

We have proposed the Matlab Simulink model of a novel implicit dynamic system (6.2.9) for online matrix inversion. Compared to Zhang implicit dynamic system (6.2.6), superior global exponential convergence to the theoretical inverse by hybrid implicit dynamic system (6.2.9) has been confirmed and justified by several computer simulation results. Tests showed that, with a greater value of  $\alpha$  (i.e.  $\gamma = 1 + \alpha$ ), faster convergence and better accuracy of the solution can be obtained with the hybrid implicit dynamic system related to the Zhang implicit dynamic system.

# Bibliography

- [1] K. M. Abadir and J. R. Magnus. *Matrix Algebra*, volume 1. Cambridge University Press, 2005.
- [2] A. Ben-Israel and T. N. E. Greville. *Generalized Inverses*. Springer, second edition, 2003.
- [3] A. Benchabane, A. Bennis, F. Charif, and A. Taleb-Ahmed. Multi-dimensional Capon spectral estimation using discrete Zhang neural networks. *Multidimensional Systems and Signal Processing*, 24(3):583–598, 2013.
- [4] A. Bjerhammar. *A generalized matrix algebra*. Lindståhl, 1958.
- [5] R. W. Brockett. Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear Algebra and its Applications*, 146:79–91, 1991.
- [6] J. Cai and G. Chen. On determinantal representation for the generalized inverse  $A_{T,S}^{(2)}$  and its applications. *Numerical Linear Algebra with Applications*, 14(3):169–182, 2007.
- [7] S. L. Campbell and C. D. Meyer. *Generalized Inverses of Linear Transformations*, volume 56. SIAM, Philadelphia, 2009.
- [8] S. L. Campbell, C. D. Meyer, Jr, and N. J. Rose. Applications of the drazin inverse to linear systems of differential equations with singular constant coefficients. *SIAM Journal on Applied Mathematics*, 31(3):411–425, 1976.
- [9] F. Charif, A. Benchabane, N. Djedi, and A. Taleb-Ahmed. Horn & Schunck meets a discrete Zhang neural networks for computing 2D optical flow. *matrix*, 2:2, 2009.
- [10] K. Chen. Recurrent implicit dynamics for online matrix inversion. *Applied Mathematics and Computation*, 219(20):10218–10224, 2013.
- [11] K. Chen and C. Yi. Robustness analysis of a hybrid of recursive neural dynamics for online matrix inversion. *Applied Mathematics and Computation*, 273:969–975, 2016.

- [12] Y. Chen. A cramer rule for solution of the general restricted linear equation. *Linear and Multilinear Algebra*, 34(2):177–186, 1993.
- [13] A. Cichocki. Neural network for singular value decomposition. *Electronics Letters*, 28(8):784–786, 1992.
- [14] A. Cichocki, T. Kaczorek, and A. Stajniak. Computation of the Drazin inverse of a singular matrix making use of neural networks. *Bulletin of the Polish Academy of Sciences. Technical Sciences*, 40(4):387–394, 1992.
- [15] A. Cichocki and R. Unbehauen. Neural networks for computing eigenvalues and eigenvectors. *Biological Cybernetics*, 68(2):155–164, 1992.
- [16] A. Cichocki and R. Unbehauen. Neural networks for solving systems of linear equations and related problems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(2):124–138, 1992.
- [17] J.-J. Climent, N. Thome, and Y. Wei. A geometrical approach on generalized inverses by Neumann-type series. *Linear algebra and its applications*, 332:533–540, 2001.
- [18] R. E. Cline and T. Greville. A Drazin inverse for rectangular matrices. *Linear Algebra and its Applications*, 29:53–62, 1980.
- [19] D. Djordjević, P. Stanimirović, and Y. Wei. The representation and approximations of outer generalized inverses. *Acta Mathematica Hungarica*, 104(1-2):1–26, 2004.
- [20] A. J. Getson and F. C. Hsuan.  $\{2\}$ -inverses and their statistical application, volume 47. Springer Science & Business Media, 2012.
- [21] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Baltimore and London, The Johns Hopkins, fourth edition, 2013.
- [22] N. C. González. On the convergence of semiiterative methods to the Drazin inverse solution of linear equations in Banach spaces. *Collect. Math.*, 46(3):303–314, 1995.
- [23] N. C. Gonzalez, J. Koliha, and Y. Wei. Integral representation of the Drazin inverse. *Electron. J. Linear Algebra*, 9:129–131, 2002.
- [24] N. C. González, J. Koliha, and Y. Wei. On integral representations of the Drazin inverse in Banach algebras. *Proceedings of the Edinburgh Mathematical Society (Series 2)*, 45(02):327–331, 2002.
- [25] A. Graham. *Kronecker Products and Matrix Calculus: with Applications*, volume 108. Horwood Chichester, 1981.

- 
- [26] C. W. Groetsch. *Generalized Inverses of Linear Operators: Representation and Approximation*. Marcel Dekker, New York-Basel, 1977.
- [27] W. Guo and T. Huang. Method of elementary transformation to compute moore–penrose inverse. *Applied Mathematics and Computation*, 216(5):1614–1617, 2010.
- [28] N. J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, 2008.
- [29] F. Hsuan, P. Langenberg, and A. Getson. The  $\{2\}$ -inverse with applications in statistics. *Linear algebra and its applications*, 70:241–248, 1985.
- [30] X. Hu, C. Sun, and B. Zhang. Design of recurrent neural networks for solving constrained least absolute deviation problems. *IEEE Transactions on Neural Networks*, 21(7):1073–1086, 2010.
- [31] Y. Hua, T. Chen, and W.-Y. Yan. Global convergence of oja’s subspace algorithm for principal component extraction. *IEEE Transactions on Neural Networks*, 9(1), 1998.
- [32] L. Jin, Y. Zhang, S. Li, *Integration-Enhanced Zhang Neural Network for real-time-varying matrix inversion in the presence of various kinds of noises*, IEEE Transactions on Neural Networks And Learning Systems **27** (2016), 2615–2627.
- [33] L. Jin, Y. Zhang, S. Li, Y. Zhang, *Noise-tolerant ZNN models for solving time-varying zero-finding problems: a control-theoretic approach*, IEEE Transactions on Automatic Control, **62**(2) (2017), 992–997.
- [34] J. Jang, S. Lee, and S. Shin. An optimization network for matrix inversion. In *Neural Information Processing Systems*, pages 397–401. College Park, MD: AIP, 1988.
- [35] T. Kailath. *Linear Systems*, volume 1. Prentice-Hall Englewood Cliffs, NJ, 1980.
- [36] V. N. Katsikis, D. Pappas, and A. Petralias. An improved method for the computation of the moore–penrose inverse matrix. *Applied Mathematics and Computation*, 217(23):9828–9834, 2011.
- [37] H. K. Khalil. *Nonlinear Systems*. Prentice Hall (Upper Saddle River, NJ), 1996.
- [38] D. L. Kleinman and M. Athans. The design of suboptimal linear time-varying systems. *IEEE Trans. Automat. Contr.*, AC-13:150–159, 1968.
- [39] A. N. Langville and C. D. Meyer. *Google’s Pagerank and Beyond: the Science of Search Engine Rankings*. Princeton University Press, Princeton, NJ, 2006.

- [40] A. N. Langville and C. D. Meyer. *Updating Markov chains with an eye on Google's PageRank*, volume 27. SIAM, 2006.
- [41] J. Levine and R. E. Hartwig. Applications of Drazin inverse to the Hill cryptographic systems, part i. *Cryptologia*, 4(2):71–85, 1980.
- [42] S. Li and Y. Li. Nonlinearly activated neural network for solving time-varying complex sylvester equation. *IEEE transactions on cybernetics*, 44(8):1397–1407, 2014.
- [43] S. Li, B. Liu, and Y. Li. Selective positive–negative feedback produces the winner-take-all competition in recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 24(2):301–309, 2013.
- [44] S. Li, H. Wang, M.U. Rafique, *A novel recurrent neural network for manipulator control with improved noise tolerance*, IEEE Transactions On Neural Networks And Learning Systems (2018), **29**:5 (2018), 1908–1918.
- [45] S. Li, Y. Zhang, L. Jin, *Kinematic Control of Redundant Manipulators Using Neural Networks*, IEEE Transactions On Neural Networks And Learning Systems **28**(10) (2017), 2243–2254.
- [46] S. Li, M.C. Zhou, X. Luo, *Modified primal-dual neural networks for motion control of redundant manipulators with dynamic rejection of harmonic noises*, IEEE Transactions On Neural Networks and Learning Systems (2017), DOI: 10.1109/TNNLS.2017.2770172.
- [47] W. Li and Z. Li. A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix. *Applied Mathematics and Computation*, 215(9):3433–3442, 2010.
- [48] B. Liao and Y. Zhang. Different complex ZFs leading to different complex ZNN models for time-varying complex generalized inverse matrices. *IEEE Transactions on Neural Networks and Learning Systems*, 25(9):1621–1631, 2014.
- [49] Q. Liu and J. Wang. Finite-time convergent recurrent neural network with a hard-limiting activation function for constrained optimization with piecewise-linear objective functions. *IEEE Transactions on Neural Networks*, 22(4):601–613, 2011.
- [50] X. Liu, H. Jin, and Y. Yu. Higher-order convergent iterative method for computing the generalized inverse and its application to Toeplitz matrices. *Linear Algebra and Its Applications*, 439(6):1635–1650, 2013.

- 
- [51] X. Liu, Y. Yu, and C. Hu. The iterative methods for computing the generalized inverse  $A_{T,S}^{(2)}$  of the bounded linear operator between banach spaces. *Applied Mathematics and Computation*, 214(2):391–410, 2009.
- [52] X. Liu, Y. Yu, J. Zhong, and Y. Wei. Integral and limit representations of the outer inverse in banach space. *Linear and Multilinear Algebra*, 60(3):333–347, 2012.
- [53] X. Liu, J. Zhong, *Integral representation of the  $W$ -weighted Drazin inverse for Hilbert space operators*, Appl. Math. Comput. **216** (2010), 3228–3233.
- [54] F.-L. Luo and Z. Bao. Neural network approach to computing matrix inversion. *Applied Mathematics and Computation*, 47(2):109–120, 1992.
- [55] MacDuffee, Đạ. Đạ. *The Theory of Matrices*, 1933. Chelsea, New York (1956).
- [56] C. D. Meyer, Jr. The role of the group generalized inverse in the theory of finite Markov chains. *SIAM Review*, 17(3):443–464, 1975.
- [57] S. Miljković, M. Miladinović, P. S. Stanimirović, and Y. Wei. Gradient methods for computing the Drazin-inverse solution. *Journal of Computational and Applied Mathematics*, 253:255–263, 2013.
- [58] H. S. Najafi and M. S. Solary. Computational algorithms for computing the inverse of a square matrix, quasi-inverse of a non-square matrix and block matrices. *Applied mathematics and computation*, 183(1):539–550, 2006.
- [59] M. Z. Nashed. *Generalized Inverse and Applications*. Academic Press, New York, 1976.
- [60] S. Osowski. Neural networks in interpolation problems. *Neurocomputing*, 5(2-3):105–118, 1993.
- [61] R. Penrose, *On a best approximate solutions to linear matrix equations*, Proc. Cambridge Philos. Soc. **52** (1956), 17–19.
- [62] R. Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.
- [63] L. Perko. *Differential Equations and Dynamical Systems*. Springer-Verlag, New York, 1991.
- [64] M. D. Petković and P. S. Stanimirović. Iterative method for computing the Moore–Penrose inverse based on Penrose equations. *Journal of Computational and applied Mathematics*, 235(6):1604–1613, 2011.

- [65] M. D. Petković and P. S. Stanimirović. Two improvements of the iterative method for computing Moore–Penrose inverse based on Penrose equations. *Journal of Computational and Applied Mathematics*, 267:61–71, 2014.
- [66] M. D. Petković, P. S. Stanimirović, and V. N. Katsikis. Modified discrete iterations for computing the inverse and pseudoinverse of the time-varying matrix. *Neurocomputing*, 289:155–165, 2018.
- [67] S. Qiao, X.-Z. Wang, Y. Wei, *Two finite-time convergent Zhang neural network models for time-varying complex matrix Drazin inverse*, *Linear Algebra Appl.* (2017), <https://doi.org/10.1016/j.laa.2017.03.014>.
- [68] V. Rakočević and Y. Wei, The representation and approximation of the W-weighted Drazin inverse of linear operators in Hilbert space. *Appl. Math. Comput.*, 141:455–470, 2003.
- [69] P. Robert. On the group-inverse of a linear transformation. *Journal of Mathematical Analysis and Applications*, 22(3):658–669, 1968.
- [70] N. Samardžija and R. Waterland. A neural network for computing eigenvectors and eigenvalues. *Biological Cybernetics*, 65(4):211–214, 1991.
- [71] B. Simeon, C. Führer, and P. Rentrop. The Drazin inverse in multibody system dynamics. *Numerische Mathematik*, 64(1):521–539, 1993.
- [72] S. Srivastava and D. Gupta. A third order iterative method for  $A^\dagger$ . *International Journal of Computing Science and Mathematics*, 4(2):140–151, 2013.
- [73] P. S. Stanimirović. Block representations of  $\{2\}, \{1, 2\}$  inverses and the Drazin inverse of matrices. *Indian Journal of Pure and Applied Mathematics*, 29:1159–1176, 1998.
- [74] P. S. Stanimirović. Limit representations of generalized inverses and related methods. *Applied mathematics and computation*, 103(1):51–68, 1999.
- [75] P. S. Stanimirović, M. Ćirić, I. Stojanović, and D. Gerontitis. Conditions for existence, representations, and computation of matrix generalized inverses. *Complexity*, 2017, 2017.
- [76] P. S. Stanimirović and D. S. Cvetković-Ilić. Successive matrix squaring algorithm for computing outer inverses. *Applied Mathematics and Computation*, 203(1):19–29, 2008.
- [77] P. S. Stanimirović, D. S. Cvetković-Ilić, S. Miljković, and M. Miladinović. Full-rank representations of  $\{2, 4\}, \{2, 3\}$ -inverses and successive matrix squaring algorithm. *Applied Mathematics and Computation*, 217(22):9358–9367, 2011.



- 
- [78] P. S. Stanimirović, D. Pappas, V. N. Katsikis, and I. P. Stanimirović. Full-rank representations of outer inverses based on the QR decomposition. *Applied Mathematics and Computation*, 218(20):10321–10333, 2012.
- [79] P. S. Stanimirović and M. D. Petković. Gauss-jordan elimination method for computing outer inverses. *Applied Mathematics and Computation*, 219(9):4667–4679, 2013.
- [80] P.S. Stanimirović, M. Petković, Gradient neural dynamics for solving matrix equations and their applications, *Neurocomputing* 306 (2018), 200–212.
- [81] P. S. Stanimirović, M. D. Petković, and D. Gerontitis. Gradient neural network with non-linear activation for computing inner inverses and the Drazin inverse. *Neural Processing Letters*, 1–25, 2017.
- [82] P.S. Stanimirović, M. Ćirić, I. Stojanović, D. Gerontitis, *Conditions for existence, representations and computation of matrix generalized inverses*, Complexity, Volume 2017, Article ID 6429725, 27 pages, <https://doi.org/10.1155/2017/6429725>.
- [83] P. S. Stanimirović and M. B. Tasić. Computing generalized inverses using LU factorization of matrix product. *International Journal of Computer Mathematics*, 85(12):1865–1878, 2008.
- [84] P. S. Stanimirović, I. Živković, and Y. Wei. Recurrent neural network for computing the Drazin inverse. *IEEE Transactions on Neural Networks and Learning Systems*, 26(11) (2015), 2830–2843.
- [85] P. S. Stanimirović, I. S. Živković, and Y. Wei. Recurrent neural network approach based on the integral representation of the Drazin inverse. *Neural Computation*, 27 (2015), 2107–2131.
- [86] P. S. Stanimirović, I. S. Živković, and Y. Wei. Neural network approach to computing outer inverses based on the full rank representation. *Linear Algebra and its Applications*, 501:344–362, 2016.
- [87] I. Stojanović, P.S. Stanimirović, I. Živković, D. Gerontitis, X.-Z. Wang, *ZNN models for computing matrix inverse based on hyperpower iterative methods*, *Filomat* **31**(10) (2017), 2999–3014.
- [88] N.S. Urquhart, *Computation of generalized inverse matrices which satisfy specified conditions*, *SIAM Review*, **10** (1968), 216–218.
- [89] G. Wang, Y. Wei, and S. Qiao. *Generalized Inverses: Theory and Computations*. Developments in Mathematics 53. Springer, Singapore and Beijing, 2018.

- [90] G. Wang and Z. Xu. Solving a kind of restricted matrix equations and Cramer rule. *Applied Mathematics and Computation*, 162(1):329–338, 2005.
- [91] H. Wang, M. Wei, and X. Liu. Several representations of  $\{2\}$ -inverses. *Arabian Journal for Science and Engineering*, 36(6):1161, 2011.
- [92] J. Wang. Electronic realisation of recurrent neural network for solving simultaneous linear equations. *Electronics Letters*, 28(5):493–495, 1992.
- [93] J. Wang. Recurrent neural networks for solving systems of complex-valued linear equations. *Electronics Letters*, 28(18):1751–1753, 1992.
- [94] J. Wang. A recurrent neural network for real-time matrix inversion. *Applied Mathematics and Computation*, 55(1):89–100, 1993.
- [95] J. Wang. Recurrent neural networks for solving linear matrix equations. *Computers and Mathematics with Applications*, 26(9):23–34, 1993.
- [96] J. Wang. Recurrent neural networks for computing pseudoinverses of rank-deficient matrices. *SIAM Journal on Scientific Computing*, 18(5):1479–1493, 1997.
- [97] J. Wang and H. Li. Solving simultaneous linear equations using recurrent neural networks. *Information Sciences*, 76(3):255–277, 1994.
- [98] J. Wang and G. Wu. Recurrent neural networks for LU decomposition and Cholesky factorization. *Mathematical and Computer Modelling*, 18(6):1–8, 1993.
- [99] L.-X. Wang and J. M. Mendel. Three-dimensional structured networks for matrix equation solving. *IEEE Transactions on Computers*, 40(12):1337–1346, 1991.
- [100] L.-X. Wang and J. M. Mendel. Parallel structured networks for solving a wide variety of matrix algebra problems. *Journal of Parallel and Distributed Computing*, 14(3):236–247, 1992.
- [101] X.-Z. Wang, H. Ma, and P. S. Stanimirović. Nonlinearly activated recurrent neural network for computing the drazin inverse. *Neural Processing Letters*, 46(1):195–217, 2017.
- [102] X.-Z. Wang, H. Ma, and P. S. Stanimirović. Recurrent neural network for computing the W-weighted Drazin inverse. *Applied Mathematics and Computation*, 300:1–20, 2017.
- [103] X.-Z. Wang, P. S. Stanimirović, and Y. Wei. Complex Zfs for computing time-varying complex outer inverses. *Neurocomputing*, 275:983–1001, 2018.

- 
- [104] X.-Z. Wang, Y. Wei, and P. S. Stanimirović. Complex neural network models for time-varying drazin inverse. *Neural Computation*, 28(12):2790–2824, 2016.
- [105] X.-Z. Wang, P.S. Stanimirović, Y. Wei, *Complex ZFs for computing time-varying complex outer inverses*, *Neurocomputing* **275** (2018), 983–1001.
- [106] Y. Wei. A characterization and representation of the generalized inverse  $A_{T,S}^{(2)}$  and its applications. *Linear Algebra and its Applications*, 280(2):87–96, 1998.
- [107] Y. Wei. Index splitting for the Drazin inverse and the singular linear system. *Applied Mathematics and Computation*, 95(2):115–124, 1998.
- [108] Y. Wei. Recurrent neural networks for computing weighted Moore-Penrose inverse. *Applied Mathematics and Computation*, 116(3):279–287, 2000.
- [109] Y. Wei. Integral representation of the generalized inverse  $A_{T,S}^{(2)}$  and its applications. *In Recent Research on Pure and Applied Algebra*, pages 59–65, 2003.
- [110] Y. Wei. The representation and approximation for the weighted Moore–Penrose inverse in Hilbert space. *Applied Mathematics and Computation*, 136(2):475–486, 2003.
- [111] Y. Wei. Recent results on the generalized inverse  $A_{T,S}^{(2)}$ . *In Linear Algebra Research Advances*, pages 231–250, 2007.
- [112] Y. Wei and D. S. Djordjević. On integral representation of the generalized inverse  $A_{T,S}^{(2)}$ . *Applied mathematics and computation*, 142(1):189–194, 2003.
- [113] Y. Wei, *Integral representation of the W-weighted Drazin inverse*, *Appl. Math. Comput.* **144** (2003), 3–10.
- [114] Y. Wei and G. Wang, *The perturbation theory for the Drazin inverse and its applications*, *Linear Algebra Appl.* **258** (1997), 179–186.
- [115] Y. Wei and H. Wu. The representation and approximation for Drazin inverse. *Journal of Computational and Applied Mathematics*, 126(1):417–432, 2000.
- [116] Y. Wei and N. Zhang. A note on the representation and approximation of the outer inverse  $A_{T,S}^{(2)}$  of a matrix A. *Applied mathematics and computation*, 147(3):837–841, 2004.
- [117] L. Weiguo, L. Juan, and Q. Tiantian. A family of iterative methods for computing Moore–Penrose inverse of a matrix. *Linear Algebra and Its Applications*, 438(1):47–56, 2013.

- [118] Y. Xia, T. Chen, and J. Shan. A novel iterative method for computing generalized inverse. *Neural computation*, 26(2):449–465, 2014.
- [119] H. Yanai, K. Takeuchi, Y. Takane, *Projection Matrices, Generalized Inverse Matrices, and Singular Value Decomposition*, Springer, New York, Dordrecht, Heidelberg, London, 2011.
- [120] C. Yonglin and Z. Bingjun. On g-inverses and the nonsingularity of a bordered matrix  $\begin{pmatrix} A & B \\ C & 0 \end{pmatrix}$ . *Linear Algebra and its Applications*, 133:133–151, 1990.
- [121] Y. Yu and Y. Wei. The representation and computational procedures for the generalized inverse  $A_{T,S}^{(2)}$  of an operator A in Hilbert spaces. *Numerical Functional Analysis and Optimization*, 30(1-2):168–182, 2009.
- [122] Y. Zhang and S. S. Ge. Design and analysis of a general recurrent neural network model for time-varying matrix inversion. *IEEE Transactions on Neural Networks*, 16(6):1477–1490, 2005.
- [123] Y. Zhang, X. Guo, W. Ma, K. Chen, and B. Cai. Matlab Simulink modeling and simulation of Zhang neural network for online time-varying matrix inversion. In *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*, pages 1480–1485. IEEE, 2008.
- [124] Y. Zhang, D. Jiang, and J. Wang. A recurrent neural network for solving sylvester equation with time-varying coefficients. *IEEE Transactions on Neural Networks*, 13(5):1053–1063, 2002.
- [125] Y. Zhang, W. Ma, and B. Cai. From Zhang neural network to Newton iteration for matrix inversion. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 56(7):1405–1415, 2009.
- [126] Y. Zhang, Y. Shi, K. Chen, and C. Wang. Global exponential convergence and stability of gradient-based neural network for online matrix inversion. *Applied Mathematics and Computation*, 215(3):1301–1306, 2009.
- [127] Y. Zhang, Y. Yang, N. Tan, and B. Cai. Zhang neural network solving for time-varying full-rank matrix Moore-Penrose inverse. *Computing*, 97(1):97–121, 2011.
- [128] Y. Zhang, B. Qiu, L. Jin, D. Guo, *Infinitely many Zhang functions resulting in various ZNN models for time-varying matrix inversion with link to Drazin inverse*, Information Processing Letters **115** (2015), 703–706.

- 
- [129] Y. Zhang, C. Yi, and W. Ma. Simulation and verification of Zhang neural network for online time-varying matrix inversion. *Simulation Modelling Practice and Theory*, 17(10):1603–1617, 2009.
- [130] Y.Zhang, B.Mu, H.Zheng, *Link between and comparison and combination of Zhang neural network and quasi-Newton BFGS method for time-varying quadratic minimization*, IEEE Trans. Cybern. **43** (2013), 490–503.
- [131] Y. Zhang, C. Yi, D. Guo, J. Zheng, *Comparison on Zhang neural dynamics and gradient-based neural dynamics for online solution of nonlinear time-varying equation*, Neural Comput. & Applic. **20** (2011), 1–7.
- [132] Y. Zhang, Z. Li, K. Li, *Complex-valued Zhang neural network for online complex-valued time-varying matrix inversion*, Appl. Math. Comput. **217** (2011), 10066–10073.
- [133] Y. Zhang, Y. Yang, N. Tan, B. Cai, *Zhang neural network solving for time-varying full-rank matrix Moore-Penrose inverse*, Computing **92** (2011), 97–121.
- [134] B. Zheng and R. Bapat. Generalized inverse  $A_{T,S}^{(2)}$  and a rank equation. *Applied mathematics and computation*, 155(2):407–415, 2004.
- [135] B. Zheng and G. Wang. Representation and approximation for generalized inverse  $A_{T,S}^{(2)}$  Revisited. *Journal of Applied Mathematics and Computing*, 22(3):225–240, 2006.
- [136] J. Zhong, X. Liu, G. Zhou, and Y. Yu. A new method for computing the Drazin inverse. *Filomat*, 26(3):597–606, 2012.
- [137] I. S. Živković and P. S. Stanimirović. Matlab simulation of the hybrid of recursive neural dynamics for online matrix inversion. *Facta Universitatis-Series Mathematics And Informatics*, 32(5):799–809, 2017.
- [138] I. S. Živković, P. S. Stanimirović, and Y. Wei. Recurrent neural network for computing outer inverse. *Neural computation*, 28(5):970–998, 2016.



# Appendix A

## Biography

Ivan Živković was born on March 20, 1983 in Majdanpek. He completed *12. September* elementary school and *Mile Arsenijević - Bandera* general-education high school in Majdanpek.

In the academic year 2002/2003 he entered the Faculty of Sciences and Mathematics in Niš, Department of Mathematics and Informatics. He received the M.Sc. degree from this faculty in 2009 with the grade point average of 9.11/10 and grade 10/10 on his diploma thesis *Object-oriented implementation of NARX neural network training using Kalman Filter*.

He enrolled for his Doctoral Academic Studies in the academic year 2009/2010 at the Department of Computer Science, the Faculty of Sciences and Mathematics in Niš. He has passed all the foreseen exams with the highest grade and has authored or co-authored five scientific papers published in international journals with IF and one paper in a domestic journal.

He was a Research Associate with the Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, Serbia, from 2011 to 2017, working on the interdisciplinary project *Development of new information and communication technologies, based on advanced mathematical methods, with applications in medicine, telecommunications, power systems, protection of national heritage and education* (No. III 044006), funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia.

He is currently a software developer at Accordia Group, LLC, New York-based company.

## List of papers

- Stanimirović, Predrag S., Ivan S. Živković, and Yimin Wei. "Recurrent neural network for computing the Drazin inverse." *IEEE transactions on neural networks and learning systems* 26.11 (2015): 2830-2843.
- Stanimirović, Predrag S., Ivan S. Živković, and Yimin Wei. "Recurrent neural network approach based on the integral representation of the Drazin inverse." *Neural computation*

27.10 (2015): 2107-2131.

- Živković, Ivan S., Predrag S. Stanimirović, and Yimin Wei. "Recurrent neural network for computing outer inverse." *Neural computation* 28.5 (2016): 970-998.
- Stanimirović, Predrag S., Ivan S. Živković, and Yimin Wei. "Neural network approach to computing outer inverses based on the full rank representation." *Linear Algebra and Its Applications* 501 (2016): 344-362.
- Stojanović, I., Stanimirović, P. S., Živković, I. S., Gerontitis, D., & Wang, X. Z. (2017). ZNN models for computing matrix inverse based on hyperpower iterative methods. *Filomat*, 31(10), 2999-3014.
- Živković, Ivan S., and Predrag S. Stanimirović. "MATLAB SIMULATION OF THE HYBRID OF RECURSIVE NEURAL DYNAMICS FOR ONLINE MATRIX INVERSION." *Facta Universitatis, Series: Mathematics and Informatics* (2018): 799-809.



# **Appendix B**

## **Dissertation documentation**



## ИЗЈАВА О АУТОРСТВУ

Изјављујем да је докторска дисертација, под насловом

### РЕКУРЕНТНЕ НЕУРОНСКЕ МРЕЖЕ ЗА РЕШАВАЊЕ ПРОБЛЕМА ЛИНЕАРНЕ АЛГЕБРЕ

која је одбрањена на Природно-математичком факултету Универзитета у Нишу:

- резултат сопственог истраживачког рада;
- да ову дисертацију, ни у целини, нити у деловима, нисам пријављивао на другим факултетима, нити универзитетима;
- да нисам повредио ауторска права, нити злоупотребио интелектуалну својину других лица.

Дозвољавам да се објаве моји лични подаци, који су у вези са ауторством и добијањем академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада, и то у каталогу Библиотеке, Дигиталном репозиторијуму Универзитета у Нишу, као и у публикацијама Универзитета у Нишу.

У Нишу, 08. 06. 2018.

Потпис аутора дисертације:

Иван Живковић  
Др Иван С. Живковић





**ИЗЈАВА О ИСТОВЕТНОСТИ ШТАМПАНОГ И ЕЛЕКТРОНСКОГ ОБЛИКА  
ДОКТОРСКЕ ДИСЕРТАЦИЈЕ**

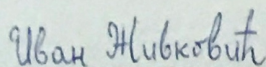
Наслов дисертације:

**РЕКУРЕНТНЕ НЕУРОНСКЕ МРЕЖЕ  
ЗА РЕШАВАЊЕ ПРОБЛЕМА  
ЛИНЕАРНЕ АЛГЕБРЕ**

Изјављујем да је електронски облик моје докторске дисертације, коју сам предао за уношење у Дигитални репозиторијум Универзитета у Нишу, истоветан штампаном облику.

У Нишу, 08. 06. 2018.

Потпис аутора дисертације:

  
Др Иван С. Живковић





## ИЗЈАВА О КОРИШЋЕЊУ

Овлашћујем Универзитетску библиотеку „Никола Тесла“ да у Дигитални репозиторијум Универзитета у Нишу унесе моју докторску дисертацију, под насловом:

### РЕКУРЕНТНЕ НЕУРОНСКЕ МРЕЖЕ ЗА РЕШАВАЊЕ ПРОБЛЕМА ЛИНЕАРНЕ АЛГЕБРЕ

Дисертацију са свим прилозима предао сам у електронском облику, погодном за трајно архивирање.

Моју докторску дисертацију, унету у Дигитални репозиторијум Универзитета у Нишу, могу користити сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons), за коју сам се одлучио/ла.

1. Ауторство (CC BY)

2. Ауторство – некомерцијално (CC BY-NC)

3. Ауторство – некомерцијално – без прераде (CC BY-NC-ND)

4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)

5. Ауторство – без прераде (CC BY-ND)

6. Ауторство – делити под истим условима (CC BY-SA)

У Нишу, 08. 06. 2018.

Потпис аутора дисертације:

*Иван Живковић*

Др Иван С. Живковић

