CĂLIN-ADRIAN POPA

Complex- and hypercomplex-valued neural networks
Habilitation thesis

Timișoara, 2021

To my family

Contents

Abstract

This habilitation thesis presents the most important scientific, professional, and academic achievements of its author, starting from May 29th, 2015, when the author defended his PhD thesis entitled "Learning algorithms for Clifford neural networks", at the Politehnica University of Timișoara.

In this period, the author elaborated 29 papers, all as first author, out of which 6 were journal papers, published in high impact academic journals, and 23 were conference papers, published at premiere conferences in the field of artificial intelligence, more precisely of neural networks. The author was also the director of two research grants won by competition.

The author began his academic career in October 2015 as a Teaching Assistant, then promoted to Assistant Professor in October 2016, and from October 2019, he is an Associate Professor at the Department of Computers and Information Technology, at the Politehnica University of Timișoara. He updated the course and/or the laboratory for 4 courses, and introduced 4 new courses at the Master of Machine Learning, for which he is the program coordinator. He was also the coordinator for over 70 Bachelor's theses, over 20 Master's theses, and is a member of the guidance commission for 7 doctoral students.

All these achievements are presented in detail in Chapter 1 of the thesis.

The scientific research of the author continued in the domain of complex- and hypercomplex-valued neural networks, extending from the feedforward neural networks discussed in the PhD thesis, to more general neural networks models belonging to the deep learning paradigm, like convolutional neural networks, stacked denoising autoencoders, deep belief networks, deep Boltzmann machines, and especially recurrent neural networks, more specifically Hopfield networks and bidirectional associative memories, for which different dynamic properties were studied.

Chapter 2 presents a somewhat direct continuation of the work done in the PhD thesis. It presents learning algorithms for quaternion-valued neural networks, but deduced in a different way than that in the PhD thesis of the author, using the recently introduced $\mathbb{HR}$ calculus. The algorithms discussed are: enhanced gradient descent algorithms, conjugate gradient algorithms, scaled conjugate gradient method, quasi-Newton learning methods, and Levenberg-Marquardt learning algorithm. The chapter summarizes 5 conference papers and 1 journal paper of the author.

Then, Chapter 3 is dedicated to summarizing 6 conference papers of the author, and aims to extend deep learning-specific algorithms to the complex domain. As such, complex-valued convolutional neural networks are introduced, a variant based on the Fourier transform, and a hybrid real–complex-valued variant of these networks. The classical deep learning models stacked denoising autoencoders, deep belief networks, and deep Boltzmann machines were extended to the complex-valued domain next.

Chapters 4 and 5 are each based on 1 journal paper of the author, and mark the transition to the study of dynamic properties for complex- and hypercomplex-valued neural networks (more precisely, quaternion-valued neural networks in this case). The analysis of the dynamic prop-

erties of neural networks is a research field in its own right, with hundreds of papers appearing each year in this domain. The extension to multidimensional neural networks of this field is rather recent, and has gained increasing interest in the last few years.

Dynamic properties of octonion-valued neural networks are discussed in Chapter 6, which is based on 5 conference papers and 2 journal papers of the author. The domain of octonion-valued neural networks were introduced by the author in a paper summarized in the first section of Chapter 6. Then, the author introduced Hopfield networks and bidirectional associative memories with octonion values, which are the subject of the next two sections. The asymptotic and exponential stability properties of octonion-valued Hopfield neural networks with different types of delays are discussed in the rest of the chapter.

Matrix-valued neural networks were also introduced by the author in his PhD thesis. Chapter 7 presents the Hopfield and bidirectional associative memory variants of these networks, for which the asymptotic and exponential stability and dissipativity properties were analyzed. The contents of the chapter is based on 7 conference papers and 1 journal paper of the author. A special type of matrix-valued neural networks, Lie algebra-valued neural networks, were also first formulated by the author. As such, the last section of Chapter 7 introduces Hopfield networks and bidirectional associative memories with Lie algebraic values.

Lastly, the thesis ends with Chapter 8, which sketches the scientific, professional, and academic future work plans of the author.

Rezumat

Această teză de abilitare prezintă cele mai importante realizări științifice, profesionale și academice ale autorului ei, începând cu 29 mai 2015, când autorul și-a susținut teza de doctorat intitulată „Algoritmi de învățare pentru rețele neuronale Clifford", la Universitatea Politehnica Timișoara.

În această perioadă, autorul a elaborat 29 de articole, toate în calitate de prim autor, dintre care 6 au fost articole de jurnal, publicate în jurnale academice cu un factor de impact mare și 23 au fost articole de conferință, publicate la conferințe de prim rang în domeniul inteligenței artificiale, mai precis al rețelelor neuronale. Autorul a fost, de asemenea, directorul a două proiecte de cercetare câștigate prin competiție.

Autorul și-a început cariera academică în octombrie 2015 ca asistent, apoi a promovat ca șef de lucrări în octombrie 2016, iar din octombrie 2019 este conferențiar la Departamentul de Calculatoare și Tehnologia Informației, la Universitatea Politehnica Timișoara. El a actualizat cursul și/sau laboratorul pentru 4 cursuri și a introdus 4 noi cursuri la Master of Machine Learning, pentru care este coordonatorul programului. De asemenea, a fost coordonatorul a peste 70 de teze de licență, peste 20 de teze de masterat și este membru al comisiei de îndrumare pentru 7 doctoranzi.

Toate aceste realizări sunt prezentate în detaliu în Capitolul 1 al tezei.

Cercetarea științifică a autorului a continuat în domeniul rețelelor neuronale cu valori complexe și hipercomplexe, extinzându-se de la rețelele neuronale feedforward discutate în teza de doctorat, la modele mai generale de rețele neuronale aparținând paradigmei de învățare profundă, cum ar fi rețelele neuronale convoluționale, stacked denoising autoencoders, deep belief networks, deep Boltzmann machines, și în special rețele neuronale recurente, mai precis rețele Hopfield și bidirectional associative memories, pentru care au fost studiate diferite proprietăți dinamice.

Capitolul 2 prezintă o continuare oarecum directă a muncii efectuate în teza de doctorat. Prezintă algoritmi de învățare pentru rețele neuronale cu valori cuaternionice, dar deduse într-un mod diferit de cel din teza de doctorat a autorului, utilizând calculul $\mathbb{HR}$, recent introdus. Algoritmii discutați sunt: algoritmi gradient descent îmbunătățiți, algoritmi bazați pe gradienți conjugați, metoda gradientului conjugat scalat, metode de învățare cvasi-Newton și algoritmul de învățare Levenberg-Marquardt. Capitolul rezumă 5 articole de conferință și 1 articol de jurnal ale autorului.

Apoi, capitolul 3 este dedicat rezumării a 6 articole de conferință ale autorului și își propune să extindă algoritmii specifici învățării profunde la domeniul complex. Ca atare, sunt introduse rețele neuronale convoluționale cu valori complexe, o variantă bazată pe transformata Fourier și o variantă hibridă cu valori complexe și valori reale ale acestor rețele. Modelele clasice de învățare profundă stacked denoising autoencoders, deep belief networks și deep Boltzmann machines au fost extinse la domeniul complex.

Capitolele 4 și 5 se bazează fiecare pe câte 1 articol de jurnal al autorului și marchează tranziția către studiul proprietăților dinamice pentru rețele neuronale cu valori complexe și

hipercomplexe (mai exact, rețele neuronale cu valori cuaternionice în acest caz). Analiza proprietăților dinamice ale rețelelor neuronale este un domeniu de cercetare în sine, cu sute de articole care apar în fiecare an în acest domeniu. Extinderea la rețelele neuronale multidimensionale în acest domeniu este destul de recentă și a câștigat un interes crescând în ultimii ani.

Proprietățile dinamice ale rețelelor neuronale cu valori octonionice sunt discutate în capitolul 6, care se bazează pe 5 articole de conferință și 2 articole de jurnal ale autorului. Domeniul rețelelor neuronale cu valori octonionice a fost introdus de autor într-o lucrare rezumată în prima secțiune a capitolului 6. Apoi, autorul a introdus rețele Hopfield și bidirectional associative memories cu valori octonionice, care fac obiectul următoarelor două secțiuni. Proprietățile de stabilitate asimptotică și exponențială ale rețelelor neuronale Hopfield cu valori octonionice cu diferite tipuri de întârzieri sunt discutate în restul capitolului.

Rețelele neuronale cu valori matriciale au fost, de asemenea, introduse de autor în teza sa de doctorat. Capitolul 7 prezintă variantele Hopfield și bidirectional associative memory ale acestor rețele, pentru care au fost analizate proprietățile de stabilitate asimptotică și exponențială și de disipativitate. Conținutul capitolului se bazează pe 7 articole de conferință și 1 articol de jurnal ale autorului. Un tip special de rețele neuronale cu valori matriciale, rețelele neuronale cu valori în algebre Lie, au fost, de asemenea, formulate pentru prima dată de către autor. Astfel, ultima secțiune a capitolului introduce rețele Hopfield și bidirectional associative memories cu valori în algebre Lie.

În cele din urmă, teza se încheie cu capitolul 8, care schițează planurile de lucru științifice, profesionale și academice viitoare ale autorului.

Chapter 1

An overview of scientific, professional, and academic results

1.1 Scientific and professional results

This thesis presents the scientific achievements of its author, starting from the spring of 2015, when the author presented his PhD thesis entitled "Learning algorithms for Clifford neural networks", which was defended in May 29th, 2015, at the Politehnica University of Timișoara.

The scientific research of the author continued in the domain of complex- and hypercomplex-valued neural networks, extending from the feedforward neural networks discussed in the PhD thesis, to more general neural networks models belonging to the deep learning paradigm, like convolutional neural networks, stacked denoising autoencoders, deep belief networks, deep Boltzmann machines, and especially recurrent neural networks, more specifically Hopfield networks and bidirectional associative memories, for which different dynamic properties were studied.

The journal papers on which this thesis is based are, in order of appearance:

1. C.-A. Popa. Learning Algorithms for Quaternion-Valued Neural Networks. *Neural Processing Letters*. September 2017. (Impact factor 2.891, Q2) [164]

2. C.-A. Popa and E. Kaslik. Multistability and multiperiodicity in impulsive hybrid quaternion-valued neural networks with mixed delays. *Neural Networks*. December 2017. (Impact factor 5.535, Q1) [176]

3. C.-A. Popa. Global Exponential Stability of Neutral-Type Octonion-Valued Neural Networks with Time-Varying Delays. *Neurocomputing*. May 2018. (Impact factor 4.438, Q1) [166]

4. C.-A. Popa. Global Exponential Stability of Octonion-Valued Neural Networks with Leakage Delay and Mixed Delays. *Neural Networks*. June 2018. (Impact factor 5.535, Q1) [167]

5. C.-A. Popa. Global μ-Stability of Neutral-Type Impulsive Complex-Valued BAM Neural Networks with Leakage Delay and Unbounded Time-Varying Delays. *Neurocomputing*. September 2019. (Impact factor 4.438, Q1) [173]

6. C.-A. Popa. Dissipativity of impulsive matrix-valued neural networks with leakage delay and mixed delays. *Neurocomputing*. March 2020. (Impact factor 4.438, Q1) [174]

The impact factor of each journal is given according to Journal Citation Reports, published by Clarivate Analytics (former ISI) in 2020. Also given is the quartile of each journal in the domain of Computer Science, Artificial Intelligence. As can be seen, the contributions of the author were published at premiere neural networks journals, having high impact factors:

- 2 papers in Neural Networks (Impact factor 5.535, Q1), which is the official journal of the International Neural Network Society (INNS),
- 3 papers in Neurocomputing (Impact factor 4.438, Q1),
- 1 paper in Neural Processing Letters (Impact factor 2.891, Q2).

The conference papers on which the thesis is based are, with two exceptions, all indexed in Clarivate Analytics Web of Science (former ISI Web of Science). In order of appearance, the conference papers are:

1. C.-A. Popa. Lie Algebra-Valued Hopfield Neural Networks. *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. September 2015, Timișoara, Romania. (Rank C, ISI) [148]
2. C.-A. Popa. Conjugate Gradient Algorithms for Quaternion-Valued Neural Networks. *International Conference on Soft Computing (MENDEL)*. June 2016, Brno, Czech Republic. (Unranked, pending ISI indexing, previous editions were indexed) [154]
3. C.-A. Popa. Lie Algebra-Valued Bidirectional Associative Memories. *International Conference on Soft Computing (MENDEL)*. June 2016, Brno, Czech Republic. (Unranked, pending ISI indexing, previous editions were indexed) [163]
4. C.-A. Popa. Matrix-Valued Hopfield Neural Networks. *International Symposium on Neural Networks (ISNN)*. July 2016, Saint Petersburg, Russia. (Rank C, ISI) [152]
5. C.-A. Popa. Enhanced Gradient Descent Algorithms for Quaternion-Valued Neural Networks. *International Workshop on Soft Computing Applications (SOFA)*. August 2016, Arad, Romania. (Rank C, ISI) [165]
6. C.-A. Popa. Matrix-Valued Bidirectional Associative Memories. *International Workshop on Soft Computing Applications (SOFA)*. August 2016, Arad, Romania. (Rank C, ISI) [172]
7. C.-A. Popa. Octonion-Valued Neural Networks. *International Conference on Artificial Neural Networks (ICANN)*. September 2016, Barcelona, Spain. (Rank B, ISI) [149]
8. C.-A. Popa. Levenberg-Marquardt Learning Algorithm for Quaternion-Valued Neural Networks. *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. September 2016, Timișoara, Romania. (Rank C, ISI) [150]
9. C.-A. Popa. Scaled Conjugate Gradient Learning for Quaternion-Valued Neural Networks. *International Conference on Neural Information Processing (ICONIP)*. October 2016, Kyoto, Japan. (Rank A, ISI) [151]
10. C.-A. Popa. Global Asymptotic Stability for Matrix-Valued Recurrent Neural Networks with Time Delays. *International Joint Conference on Neural Networks (IJCNN)*. May 2017, Anchorage, Alaska, USA. (Rank A, ISI) [160]

11. C.-A. Popa. Complex-Valued Convolutional Neural Networks for Real-Valued Image Classification. *International Joint Conference on Neural Networks (IJCNN)*. May 2017, Anchorage, Alaska, USA. (Rank A, ISI) [153]

12. C.-A. Popa. Octonion-Valued Bidirectional Associative Memories. *International Joint Conference on Neural Networks (IJCNN)*. May 2017, Anchorage, Alaska, USA. (Rank A, ISI) [156]

13. C.-A. Popa. Exponential Stability for Delayed Octonion-Valued Recurrent Neural Networks. *International Work-Conference on Artificial Neural Networks (IWANN)*. June 2017, Cadiz, Spain. (Rank B, ISI) [158]

14. C.-A. Popa. Quasi-Newton Learning Methods for Quaternion-Valued Neural Networks. *International Work-Conference on Artificial Neural Networks (IWANN)*. June 2017, Cadiz, Spain. (Rank B, ISI) [155]

15. C.-A. Popa. Global Asymptotic Stability for Octonion-Valued Neural Networks with Delay. *International Symposium on Neural Networks (ISNN)*. June 2017, Sapporo, Hokkaido, Japan. (Rank C, ISI) [157]

16. C.-A. Popa. Global Exponential Stability for Matrix-Valued Neural Networks with Time Delay. *International Symposium on Neural Networks (ISNN)*. June 2017, Sapporo, Hokkaido, Japan. (Rank C, ISI) [161]

17. C.-A. Popa. Asymptotic Stability of Delayed Octonion-Valued Neural Networks with Leakage Delay. *International Conference on Neural Information Processing (ICONIP)*. November 2017, Guangzhou, China. (Rank A, ISI) [159]

18. C.-A. Popa. Exponential Stability of Matrix-Valued BAM Neural Networks with Time-Varying Delays. *International Conference on Neural Information Processing (ICONIP)*. November 2017, Guangzhou, China. (Rank A, ISI) [162]

19. C.-A. Popa and C. Cernăzanu-Glăvan. Fourier Transform-Based Image Classification Using Complex-Valued Convolutional Neural Networks. *International Symposium on Neural Networks (ISNN)*. June 2018, Minsk, Belarus. (Rank A, ISI) [175]

20. C.-A. Popa. Complex-Valued Deep Belief Networks. *International Symposium on Neural Networks (ISNN)*. June 2018, Minsk, Belarus. (Rank C, ISI) [170]

21. C.-A. Popa. Complex-Valued Stacked Denoising Autoencoders. *International Symposium on Neural Networks (ISNN)*. June 2018, Minsk, Belarus. (Rank C, ISI) [169]

22. C.-A. Popa. Complex-Valued Deep Boltzmann Machines. *International Joint Conference on Neural Networks (IJCNN)*. July 2018, Rio de Janeiro, Brazil, Brazil. (Rank A, ISI) [171]

23. C.-A. Popa. Deep Hybrid Real-Complex-Valued Convolutional Neural Networks for Image Classification. *International Joint Conference on Neural Networks (IJCNN)*. July 2018, Rio de Janeiro, Brazil, Brazil. (Rank A, ISI) [168]

The rank of each conference is given according to the Australian Research Council Conference Rankings. As can be seen, the contributions of the author were published at premiere neural networks conferences:

- 5 papers at International Joint Conference on Neural Networks – IJCNN (Rank A, ISI), which is the official conference of the International Neural Network Society (INNS),
- 3 papers at International Conference on Neural Information Processing – ICONIP (Rank A, ISI), which is the official conference of the Asia Pacific Neural Network Society (APNNS),
- 1 paper at International Conference on Artificial Neural Networks – ICANN (Rank B, ISI), which is the official conference of the European Neural Network Society (ENNS),
- 2 papers at International Work-Conference on Artificial Neural Networks – IWANN (Rank B, ISI),
- 6 papers at International Symposium on Neural Networks – ISNN (Rank C, ISI),
- 2 papers at International Symposium on Symbolic and Numeric Algorithms for Scientific Computing – SYNASC (Rank C, ISI),
- 2 papers at International Workshop on Soft Computing Applications – SOFA (Rank C, ISI),
- 2 papers at International Conference on Soft Computing (MENDEL).

Out of the 29 journal and conference papers, the author was **single author** for **27** of them, and the **first author** of **all** of them.

The author, due to the participation at the above-mentioned conferences, is a member of International Neural Network Society (INNS), Asia Pacific Neural Network Society (APNNS), and European Neural Network Society (ENNS).

The venues which accepted the papers of the author show that the ideas proposed in the respective papers were well regarded by the academic community in the domain of neural networks.

Also as a recognition of the scientific activity of the author, he was invited to **review** for the following journals:

1. IEEE Transactions on Cybernetics (IEEE, ISI Impact factor 11.079, Q1)
2. IEEE Transactions on Systems, Man, and Cybernetics: Systems (IEEE, ISI Impact factor 9.309, Q1)
3. IEEE Transactions on Neural Networks and Learning Systems (IEEE, ISI Impact factor 8.793, Q1)
4. Automation in Construction (ScienceDirect, ISI Impact factor 5.669, Q1)
5. Neural Newtorks (ScienceDirect, ISI Impact factor 5.535, Q1)
6. Expert Systems with Applications (ScienceDirect, ISI Impact factor 5.452, Q1)
7. IEEE Transactions on Network Science and Engineering (ScienceDirect, ISI Impact factor 5.213, Q1)
8. Nonlinear Dynamics (SpringerLink, ISI Impact factor 4.867, Q1)
9. Neurocomputing (ScienceDirect, ISI Impact factor 4.438, Q1)

10. IEEE Transactions on Biomedical Engineering (IEEE, ISI Impact factor 4.424, Q1)
11. Engineering Applications of Artificial Intelligence (ScienceDirect, ISI Impact factor 4.201, Q1)
12. Journal of The Franklin Institute (ScienceDirect, ISI Impact factor 4.036, Q1)
13. International Journal of Machine Learning and Cybernetics (SpringerLink, ISI Impact factor 3.753, Q1)
14. IEEE Access (IEEE, ISI Impact factor 3.745 , Q1)
15. Applied Mathematical Modelling (ScienceDirect, ISI Impact factor 3.633, Q1)
16. Computer Methods and Programs in Biomedicine (ScienceDirect, ISI Impact factor 3.632, Q1)
17. International Journal of Electrical Power and Energy Systems (ScienceDirect, ISI Impact factor 3.588, Q1)
18. Applied Mathematics and Computation (ScienceDirect, ISI Impact factor 3.472, Q1)
19. Fuzzy Sets and Systems (ScienceDirect, ISI Impact factor 3.305, Q1)
20. Sensors (MDPI, ISI Impact factor 3.275, Q1)
21. Neural Processing Letters (SpringerLink, ISI Impact factor 2.891, Q2)
22. Asian Journal of Control (Wiley Online Library, ISI Impact factor 2.779, Q2)
23. International Journal of Control, Automation and Systems (SpringerLink, ISI Impact factor 2.733, Q2)
24. Symmetry (MDPI, ISI Impact factor 2.645, Q2)
25. Neural Computation (MIT Press Journals, ISI Impact factor 2.505, Q2)
26. Complexity (Hindawi, ISI Impact factor 2.462, Q2)
27. Metals (MDPI, ISI Impact factor 2.117, Q1)
28. Mathematics (MDPI, ISI Impact factor 1.747, Q1)
29. Arabian Journal for Science and Engineering (SpringerLink, ISI Impact factor 1.711, Q3)
30. Mathematical Methods in the Applied Sciences (Wiley Online Library, ISI Impact factor 1.626, Q2)
31. Mathematics and Computers in Simulation (ScienceDirect, ISI Impact factor 1.620, Q2)
32. Journal of Difference Equations and Applications (Taylor & Francis Online, ISI Impact factor 1.162, Q3)
33. Acta Applicandae Mathematicae (SpringerLink, ISI Impact factor 0.974, Q3)

34. Numerical Functional Analysis and Optimization (Taylor & Francis Online, ISI Impact factor 0.896, Q3)

35. AIMS Mathematics (AIMS Press, ISI Impact factor 0.882, Q3)

and for the conference

1. International Joint Conference on Neural Networks (IJCNN) (IEEE, ISI).

The impact factor of each journal is given according to Journal Citation Reports, published by Clarivate Analytics (former ISI) in 2020. Also, the publisher of the journal is indicated, and the most favorable quartile placement according to Journal Citation Reports. As can be seen, the author is reviewer to very high impact factor journals, out of which 22 are in the Q1 quartile, 8 are in the Q2 quartile and 5 are in the Q3 quartile.

Although the domain of the research is rather new, and the majority of the papers appeared starting late 2015, they have **111 citations** in Clarivate Analytics Web of Science (former ISI) journals and conferences, and **17 citations** in other international databases, not counting the citations by the author or coauthors. The **Hirsch Index** of the author according to Web of Science is **6**, according to Scopus is **8**, and according to Google Scholar is **9**. Based on his research results and publication record, the author has recently been promoted to the position of **Associate Professor** at the Politehnica University of Timișoara (October 2019), after being a Teaching Assistant for just one year (between October 2015 and October 2016), and an Assistant Professor (Lecturer) for three years (between October 2016 and October 2019).

The author received the **„Excellence in Research" Prize** for outstanding results obtained in the academic year 2017–2018, awarded by the Politehnica University of Timișoara in 2018. He also received the **„Eminent Researcher" Prize**, awarded by the „Orizonturi Universitare" Association in 2019.

The author was also director for **two research projects** won by competition:

1. **Project PCD-TC-2017-41**, „Complex-valued deep neural networks", Director C.-A. Popa, awarded by the Politehnica University of Timișoara, Budget 46500 RON (10000 EUR). Members: Cosmin Cernăzanu-Glăvan. The results of the project are:
   - 5 ISI conference papers — **2 Rank A, 3 Rank C conferences**
   - 1 ISI journal paper — **Q1 quartile**
   - buying 2 high-performance computers with 2 GPUs for training neural networks.
2. **Project** „Bayesian Deep Learning for Depth Estimation with Application to 3D Object Detection", Director C.-A. Popa, awarded by Continental Automotive România S.R.L., Budget 48089.49 RON (10010.88 EUR). Members: Daniela-Ana Rusu, Petra Csereoka. The results of the project are to be included in one paper which is under development.

Both projects involved one or more team members besides the author, which proves the capacity of the author to **coordinate research teams**.

1.2 Academic results

The author introduced the following **courses** and **laboratories** at the Computers and Information Technology Department from the Politehnica University of Timișoara:

1. *Computer Assisted Mathematics*, 1st Year Bachelor's Computers – in Romanian

   - C.-A. Popa. Computer Assisted Mathematics Course Notes (in Romanian), 259 pages, 2017
   - C.-A. Popa. Computer Assisted Mathematics Laboratory in MATLAB (in Romanian), 65 pages, 2017

2. *Modeling and Simulation*, 4th Year Bachelor's Computers – in English

   - C.-A. Popa. Modeling and Simulation Course Notes (in English), 219 pages, 2015
   - C.-A. Popa. Modeling and Simulation Laboratory in AnyLogic (in English), 305 pages, 2015

3. *Image Processing and Recognition*, 1st Year Master's Computer Engineering (discontinued from the Master's and moved to 4th Year Bachelor's Computers – in English)

   - C.-A. Popa. Image Processing and Recognition Course Notes (in English), 184 pages, 2016
   - C.-A. Popa. Image Processing and Recognition Laboratory in MATLAB (in English), 151 pages, 2016

4. *Graphics and Human-Computer Interfaces*, 3rd Year Bachelor's Computers – in English

   - C.-A. Popa. Computer Graphics Laboratory in OpenGL ES 1.x (in English), 135 pages, 2016

5. *Deep Learning*, 1st Year Master's Machine Learning

   - C.-A. Popa. Deep Learning Course Notes (in English), 2020

6. *Research Topics in Machine Learning*, 1st Year Master's Machine Learning

   - C.-A. Popa. Research Topics in Machine Learning Course Notes (in English), 2020

7. *Reinforcement Learning*, 1st Year Master's Machine Learning

   - C.-A. Popa. Reinforcement Learning Course Notes (in English), 2021

8. *Autonomous Driving*, 2nd Year Master's Machine Learning

   - C.-A. Popa. Autonomous Driving Course Notes (in English), pending fall 2021

The recognition of the fact that the author is an expert in the domain of machine learning by his colleagues is manifested in that he was named **program coordinator** for the **Master's of Machine Learning**, which was introduced for the first time at the Politehnica University of Timișoara starting from the fall of 2020. The main attributions were deciding upon a curriculum for the new master's program, and developing the file needed for the accreditation of the new master's program.

The author was the **coordinator** of an increasing number of students for the development of their **Bachelor's** and **Master's theses**, as follows:

- 5 students for Bachelor's thesis in 2016,

- 5 students for Bachelor's thesis in 2017,
- 1 student for Bachelor's thesis, 8 students for Master's thesis in 2018,
- 15 students for Bachelor's thesis, 6 students for Master's thesis in 2019,
- 21 students for Bachelor's thesis, 3 students for Master's thesis in 2020,
- 26 students for Bachelor's thesis, 8 students for Master's thesis in 2021 (pending).

Also, the author is in the Guidance Commission for **7 Doctorate students**.

These facts are proof of the capacity of the author to coordinate Bachelor's, Master's, and Doctoral students into doing research in the domain of machine learning, and highlight again the author's capacity to **coordinate research teams**.

As a recognition of his good relation with the students and as a token of their appreciation, the author received the **„Profesor Bologna" Distinction** in 2017, awarded by the National Association of Student Organizations in Romania (ANOSR), at the recommendation of the Liga AC Student Association from the Faculty of Automation and Computers at the Politehnica University of Timișoara.

Chapter 2

Learning algorithms for quaternion-valued neural networks

The domain of quaternion-valued neural networks has received an increasing interest over the last few years. Some popular applications of these networks include chaotic time-series prediction [7], color image compression [89], color night vision [104], polarized signal classification [22], and 3D wind forecasting [92, 213, 214].

Some signals in the 3D and 4D domains can be more naturally expressed in quaternion-valued form. Thus, these networks appear as a natural choice for solving problems such as time series prediction. Several methods have been proposed to increase the efficiency of learning in quaternion-valued neural networks. These methods include different network architectures and different learning algorithms, some of which are specially designed for this type of networks, while others are extended from the real-valued case.

One of the simplest classes of algorithms that perform better than the classical gradient descent in the real and complex [142] domains is the so-called enhanced gradient descent algorithms. By doing a series of ad hoc modifications to the gradient descent algorithm, these methods have managed to increase its performance. Out of the vast literature presenting such methods, we chose some of the most known, robust, and theoretically well-founded optimization methods to extend to the quaternion domain.

The quickprop algorithm approximates the error function by a quadratic polynomial and finds its minimum. The resilient backpropagation algorithm replaces the partial derivatives that appear in the expression of the update rule of the gradient descent with other weight update quantities, which are computed taking into account only the signs of the partial derivatives. Lastly, delta-bar-delta and SuperSAB, which differ only slightly, make the learning rate hyperparameter specific to each weight of the network and to each training epoch, which means that they also provide a method to update these specific learning rates.

First proposed, among others, by [94, 29], the conjugate gradient learning algorithm has also been proven to be very efficient in the real-valued and complex-valued [147] domains. The scaled conjugate gradient learning method, proposed by [122], has also become a popular algorithm for training the real-valued and complex-valued [143] feedforward neural networks. They belong to the class of first order methods, more specifically line search methods, which replace the gradient in the gradient descent method with a number of conjugate directions.

One of the most effective from the class of second order methods used to minimize a cost function, is, in theory, the Newton method, see [135]. But because it needs the explicit calculation of the Hessian matrix of the cost function, more precisely its inverse, which is a computationally expensive task, quasi-Newton methods have been developed. They replace the explicit

calculation of the Hessian with an approximation of it, which is positive definite by construction, to avoid the convergence problems in the Newton method when the Hessian is not positive definite. First applied, among others, by [226, 11] to the training of neural networks, the quasi-Newton learning method has proved to be very efficient in the real-valued and complex-valued [145] cases.

On the other hand, to solve the same problem, the Levenberg-Marquardt algorithm uses an approximation of the Hessian matrix which only requires first order derivatives to be computed, more precisely the Jacobian matrix. This approximation, together with the trust region method, avoid the convergence problems in the Newton method when the Hessian is not positive definite or is ill-conditioned (i.e., almost singular). Introduced by [121] and first applied to training neural networks by [70], the Levenberg-Marquardt algorithm represents one of the most popular algorithms to train real-valued and complex-valued [2] feedforward neural networks.

Taking all the above facts into consideration, a natural idea was to extend these learning algorithms to training quaternion-valued feedforward neural networks, also. The presentation of the enhanced gradient descent algorithms follows that of [180], and of the classical book [18], with the obvious adaptation to the quaternion domain. Using the framework of the $\mathbb{HR}$ calculus, we deduce the quaternion-valued conjugate gradient, scaled conjugate gradient, quasi-Newton, and Levenberg-Marquardt algorithms starting from the real-valued case, also following the classical book [18]. We test all the proposed algorithms on linear and chaotic time series prediction problems.

The presentation in this chapter follows the one in the author's paper [164], which in turn summarizes the author's papers [165], [154], [151], [155], and [150].

2.1 The $\mathbb{HR}$ calculus

First, we will present the basic ideas of the $\mathbb{HR}$ calculus [231], which will be later used to deduce the different algorithms for optimizing a quaternion domain error function.

Let $\mathbb{H} = \{q_a + \imath q_b + \jmath q_c + \kappa q_d | q_a, q_b, q_c, q_d \in \mathbb{R}\}$ be the algebra of quaternions, where $\imath$, $\jmath$, κ are the imaginary units which satisfy $\imath^2 = \jmath^2 = \kappa^2 = \imath\jmath\kappa = -1$. We define the operation $q^\mu := \mu q \mu^{-1}$, for any $\mu \in \mathbb{H}$. Using this operation, for any $q = q_a + \imath q_b + \jmath q_c + \kappa q_d \in \mathbb{H}$, we have $q^\imath = \imath q \imath^{-1} = q_a + \imath q_b - \jmath q_c - \kappa q_d$, $q^\jmath = \jmath q \jmath^{-1} = q_a - \imath q_b + \jmath q_c - \kappa q_d$, $q^\kappa = \kappa q \kappa^{-1} = q_a - \imath q_b - \jmath q_c + \kappa q_d$. For a function $f : \mathbb{H} \to \mathbb{R}$, we can define the $\mathbb{HR}$ derivatives of f by

$$\begin{pmatrix} \frac{\partial f}{\partial q} \\ \frac{\partial f}{\partial q^\imath} \\ \frac{\partial f}{\partial q^\jmath} \\ \frac{\partial f}{\partial q^\kappa} \end{pmatrix} := \frac{1}{4} \begin{pmatrix} 1 & -\imath & -\jmath & -\kappa \\ 1 & -\imath & \jmath & \kappa \\ 1 & \imath & -\jmath & \kappa \\ 1 & \imath & \jmath & -\kappa \end{pmatrix} \begin{pmatrix} \frac{\partial f}{\partial q_a} \\ \frac{\partial f}{\partial q_b} \\ \frac{\partial f}{\partial q_c} \\ \frac{\partial f}{\partial q_d} \end{pmatrix}.$$

Consider now a quaternion vector $\mathbf{q} = (q_1, q_2, \ldots, q_N)^T \in \mathbb{H}^N$, which can be written as $\mathbf{q} = \mathbf{q}_a + \imath \mathbf{q}_b + \jmath \mathbf{q}_c + \kappa \mathbf{q}_d \in \mathbb{H}^N$, where $\mathbf{q}_a, \mathbf{q}_b, \mathbf{q}_c, \mathbf{q}_d \in \mathbb{R}^N$. We have $\mathbf{q}^\imath = \imath \mathbf{q} \imath^{-1} = \mathbf{q}_a + \imath \mathbf{q}_b - \jmath \mathbf{q}_c - \kappa \mathbf{q}_d$, $\mathbf{q}^\jmath = \jmath \mathbf{q} \jmath^{-1} = \mathbf{q}_a - \imath \mathbf{q}_b + \jmath \mathbf{q}_c - \kappa \mathbf{q}_d$, $\mathbf{q}^\kappa = \kappa \mathbf{q} \kappa^{-1} = \mathbf{q}_a - \imath \mathbf{q}_b - \jmath \mathbf{q}_c + \kappa \mathbf{q}_d \in \mathbb{H}^N$, or, equivalently,

$$\begin{pmatrix} \mathbf{q} \\ \mathbf{q}^\imath \\ \mathbf{q}^\jmath \\ \mathbf{q}^\kappa \end{pmatrix} = \begin{pmatrix} \mathbb{I}_N & \imath \mathbb{I}_N & \jmath \mathbb{I}_N & \kappa \mathbb{I}_N \\ \mathbb{I}_N & \imath \mathbb{I}_N & -\jmath \mathbb{I}_N & -\kappa \mathbb{I}_N \\ \mathbb{I}_N & -\imath \mathbb{I}_N & \jmath \mathbb{I}_N & -\kappa \mathbb{I}_N \\ \mathbb{I}_N & -\imath \mathbb{I}_N & -\jmath \mathbb{I}_N & \kappa \mathbb{I}_N \end{pmatrix} \begin{pmatrix} \mathbf{q}_a \\ \mathbf{q}_b \\ \mathbf{q}_c \\ \mathbf{q}_d \end{pmatrix},$$

where $\mathbb{I}_N$ is the $N \times N$ identity matrix. By denoting

$$\overset{\mathcal{H}}{\mathbf{q}} := \begin{pmatrix} \mathbf{q} \\ \mathbf{q}^{\imath} \\ \mathbf{q}^{\jmath} \\ \mathbf{q}^{\kappa} \end{pmatrix} \in \mathbb{H}^{4N}, \ \overset{\mathcal{R}}{\mathbf{q}} := \begin{pmatrix} \mathbf{q}_a \\ \mathbf{q}_b \\ \mathbf{q}_c \\ \mathbf{q}_d \end{pmatrix} \in \mathbb{R}^{4N}, \ \mathbb{J}_N := \begin{pmatrix} \mathbb{I}_N & \imath\mathbb{I}_N & \jmath\mathbb{I}_N & \kappa\mathbb{I}_N \\ \mathbb{I}_N & \imath\mathbb{I}_N & -\jmath\mathbb{I}_N & -\kappa\mathbb{I}_N \\ \mathbb{I}_N & -\imath\mathbb{I}_N & \jmath\mathbb{I}_N & -\kappa\mathbb{I}_N \\ \mathbb{I}_N & -\imath\mathbb{I}_N & -\jmath\mathbb{I}_N & \kappa\mathbb{I}_N \end{pmatrix},$$

the above relation becomes

$$\overset{\mathcal{H}}{\mathbf{q}} = \mathbb{J}_N \overset{\mathcal{R}}{\mathbf{q}}.$$

It can be checked that $\mathbb{J}_N^H \mathbb{J}_N = \mathbb{J}_N \mathbb{J}_N^H = 4\mathbb{I}_{4N}$, and so we deduce that

$$\overset{\mathcal{R}}{\mathbf{q}} = \frac{1}{4} \mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{q}}. \tag{2.1.1}$$

A function $f : \mathbb{H}^N \to \mathbb{R}$ can now be seen in three equivalent forms

$$f(\mathbf{q}) \Leftrightarrow f(\overset{\mathcal{H}}{\mathbf{q}}) := f(\mathbf{q}, \mathbf{q}^{\imath}, \mathbf{q}^{\jmath}, \mathbf{q}^{\kappa}) \Leftrightarrow f(\overset{\mathcal{R}}{\mathbf{q}}) := f(\mathbf{q}_a, \mathbf{q}_b, \mathbf{q}_c, \mathbf{q}_d).$$

If we define

$$\frac{\partial f}{\partial \mathbf{q}} := \left(\frac{\partial f}{\partial q_1}, \ldots, \frac{\partial f}{\partial q_N} \right),$$
$$\frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}} := \left(\frac{\partial f}{\partial \mathbf{q}}, \frac{\partial f}{\partial \mathbf{q}^{\imath}}, \frac{\partial f}{\partial \mathbf{q}^{\jmath}}, \frac{\partial f}{\partial \mathbf{q}^{\kappa}} \right),$$
$$\frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}} := \left(\frac{\partial f}{\partial \mathbf{q}_a}, \frac{\partial f}{\partial \mathbf{q}_b}, \frac{\partial f}{\partial \mathbf{q}_c}, \frac{\partial f}{\partial \mathbf{q}_d} \right),$$

we have, from the chain rule, that

$$\frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}} = \frac{1}{4} \frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}} \mathbb{J}_N^H \Leftrightarrow \frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}} = \frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}} \mathbb{J}_N.$$

If we now define $\nabla_{\mathbf{q}} f := \left(\frac{\partial f}{\partial \mathbf{q}} \right)^H$, $\nabla_{\overset{\mathcal{H}}{\mathbf{q}}} f := \left(\frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}} \right)^H$, $\nabla_{\overset{\mathcal{R}}{\mathbf{q}}} f := \left(\frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}} \right)^T$, where $(\cdot)^T$ and $(\cdot)^H$ represent the transpose and the Hermitian transpose, respectively, the above relations can be written as

$$\nabla_{\overset{\mathcal{H}}{\mathbf{q}}} f = \frac{1}{4} \mathbb{J}_N \nabla_{\overset{\mathcal{R}}{\mathbf{q}}} f \Leftrightarrow \nabla_{\overset{\mathcal{R}}{\mathbf{q}}} f = \mathbb{J}_N^H \nabla_{\overset{\mathcal{H}}{\mathbf{q}}} f. \tag{2.1.2}$$

Similarly, by defining $\nabla^2_{\mathbf{q}} f := \frac{\partial}{\partial \mathbf{q}} \left(\frac{\partial f}{\partial \mathbf{q}} \right)^H$, $\nabla^2_{\overset{\mathcal{H}}{\mathbf{q}}} f := \frac{\partial}{\partial \overset{\mathcal{H}}{\mathbf{q}}} \left(\frac{\partial f}{\partial \overset{\mathcal{H}}{\mathbf{q}}} \right)^H$, $\nabla_{\overset{\mathcal{R}}{\mathbf{q}}} f := \frac{\partial}{\partial \overset{\mathcal{R}}{\mathbf{q}}} \left(\frac{\partial f}{\partial \overset{\mathcal{R}}{\mathbf{q}}} \right)^T$, we obtain that

$$\nabla^2_{\overset{\mathcal{H}}{\mathbf{q}}} f = \frac{1}{16} \mathbb{J}_N (\nabla^2_{\overset{\mathcal{R}}{\mathbf{q}}} f) \mathbb{J}_N^H \Leftrightarrow \nabla^2_{\overset{\mathcal{R}}{\mathbf{q}}} f = \mathbb{J}_N^H (\nabla^2_{\overset{\mathcal{H}}{\mathbf{q}}} f) \mathbb{J}_N. \tag{2.1.3}$$

2.2 Enhanced gradient descent algorithms

In this and the following sections, assume that we have a quaternion-valued feedforward neural network with an error function $E : \mathbb{H}^N \to \mathbb{R}$, and an N-dimensional weight vector denoted by $\mathbf{w} \in \mathbb{H}^N$, whose elements are $w_i \in \mathbb{H}$, $1 \leq i \leq N$. We can define the weight step at epoch k by

$$\Delta w_i(k) = w_i(k+1) - w_i(k).$$

Then, for the gradient descent algorithm, the update rule for the weight w_i is given by

$$\Delta w_i(k) = -\varepsilon \left(\frac{\partial E}{\partial [w_i]_a}(k) + \imath \frac{\partial E}{\partial [w_i]_b}(k) + \jmath \frac{\partial E}{\partial [w_i]_c}(k) + \kappa \frac{\partial E}{\partial [w_i]_d}(k) \right),$$

where $\varepsilon \in \mathbb{R}$ is the learning rate.

The update rules for the enhanced gradient descent algorithms do not depend on the quaternion numbers as a whole, but rather are formulated separately for each of the four real components of quaternions. The main reason for this is that these algorithms make assumptions on the value of the partial derivative of the error function with respect to the weight that is being updated, assumptions which involve comparisons. It is well known that, unlike real numbers, quaternion numbers do not have a natural ordering, so there are no quaternion-valued inequalities. As a consequence, we had to give separate update rules for the four components of each weight, which depend on assumptions made about the partial derivatives of the error function with respect to each of the four quaternion components. Because of this, unlike for the following algorithms, the $\mathbb{HR}$ calculus will not be used for the deduction of these learning methods.

2.2.1 Quickprop

The *quickprop* algorithm was first proposed by Fahlman in [53]. Its update expression for the quaternion-valued domain is: [165]

$$[\Delta w_i(k)]_x = \frac{\frac{\partial E}{\partial [w_i]_x}(k)}{\frac{\partial E}{\partial [w_i]_x}(k-1) - \frac{\partial E}{\partial [w_i]_x}(k)} [\Delta w_i(k-1)]_x, \ \forall x \in \{a, b, c, d\}.$$

The heuristic behind the algorithm is that it approximates the error function with respect to each weight by a quadratic polynomial. This assumes that each weight is independent of all the others and that the quadratic polynomial has a minimum, and not a maximum.

2.2.2 Resilient backpropagation

Riedmiller and Braun [181] devised the *resilient backpropagation* algorithm, or *RPROP* for short, in order to eliminate the potential harmful influence of the size of the partial derivative on the weight step. The algorithm replaces the partial derivative from the weight update formula with another quantity denoted by Δ_i, which will then be used for updating the weight w_i. The update formulas for computing the quantities Δ_i are: [165]

$$[\Delta_i(k)]_x = \begin{cases} \eta^+ [\Delta_i(k-1)]_x, & \text{if } \frac{\partial E}{\partial [w_i]_x}(k-1) \times \frac{\partial E}{\partial [w_i]_x}(k) > 0 \\ \eta^- [\Delta_i(k-1)]_x, & \text{if } \frac{\partial E}{\partial [w_i]_x}(k-1) \times \frac{\partial E}{\partial [w_i]_x}(k) < 0 \\ [\Delta_i(k-1)]_x, & \text{else} \end{cases}, \ \forall x \in \{a, b, c, d\},$$

which are calculated taking into account only the signs of the partial derivatives: each time the partial derivatives $\frac{\partial E}{\partial [w_i]_x}$ change their sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the quantities $[\Delta_i]_x$ are decreased by a factor of η^-, and each time those partial derivatives have the same sign, the quantities $[\Delta_i]_x$ are increased by a factor of η^+, in order to accelerate the convergence. The real hyperparameters η^+ and η^- must satisfy $0 < \eta^- < 1 < \eta^+$, and are usually taken to be $\eta^- = 0.5$ and $\eta^+ = 1.2$.

We can now write the formulas for updating the weight w_i based on the quantities Δ_i:

$$[\Delta w_i(k)]_x = \begin{cases} -[\Delta_i(k)]_x, & \text{if } \frac{\partial E}{\partial [w_i]_x}(k) > 0 \\ [\Delta_i(k)]_x, & \text{if } \frac{\partial E}{\partial [w_i]_x}(k) < 0 \\ 0, & \text{else} \end{cases}, \ \forall x \in \{a, b, c, d\}.$$

This update rule is simple: if the derivative is positive, which means an increasing error, the weight is decreased, and if the derivative is negative, the weight is increased.

2.2.3 Delta-bar-delta

The *delta-bar-delta* algorithm was proposed by Jacobs in [91], who had the idea to introduce a different learning rate for each weight in the network, and to devise a procedure for updating these learning rates while learning. The gradient descent rule becomes in this case

$$\begin{aligned} \Delta w_i(k) &= -\Bigg([\varepsilon_i(k)]_a \frac{\partial E}{\partial [w_i]_a}(k) + \imath [\varepsilon_i(k)]_b \frac{\partial E}{\partial [w_i]_b}(k) \\ &\quad + \jmath [\varepsilon_i(k)]_c \frac{\partial E}{\partial [w_i]_c}(k) + \kappa [\varepsilon_i(k)]_d \frac{\partial E}{\partial [w_i]_d}(k) \Bigg). \end{aligned}$$

We can see that each learning rate $\varepsilon_i(k)$ depends on the weight w_i and on the epoch k, and is a quaternion, unlike in the quaternion gradient descent, where the learning rate ε is a real number. The update rules for the learning rates ε_i are: [165]

$$[\varepsilon_i(k)]_x = \begin{cases} \kappa + [\varepsilon_i(k-1)]_x, & \text{if } \frac{\partial \bar{E}}{\partial [w_i]_x}(k-1) \times \frac{\partial E}{\partial [w_i]_x}(k) > 0 \\ \eta^- [\varepsilon_i(k-1)]_x, & \text{if } \frac{\partial \bar{E}}{\partial [w_i]_x}(k-1) \times \frac{\partial E}{\partial [w_i]_x}(k) < 0 \\ [\varepsilon_i(k-1)]_x, & \text{else} \end{cases}, \ \forall x \in \{a, b, c, d\},$$

where we denoted

$$\frac{\partial \bar{E}}{\partial [w_i]_x}(k) := (1-\theta) \frac{\partial E}{\partial [w_i]_x}(k) + \theta \frac{\partial \bar{E}}{\partial [w_i]_x}(k-1), \ \forall x \in \{a, b, c, d\}.$$

The hyperparameter η^- must satisfy $0 < \eta^- < 1$, and is usually taken to be, like in the RPROP algorithm, equal to 0.5. We also took $\kappa = 10^{-6}$ and $\theta = 0.7$ in our experiments, but these values are usually determined empirically.

In this learning algorithm, if two consecutive partial derivatives $\frac{\partial E}{\partial [w_i]_x}$ have the same sign, then the learning rate is increased by a small constant κ to accelerate learning. If the two partial derivatives have different signs, this indicates that the local minimum was missed, which means that the weight step was too big. Thus, the learning rate is decreased by multiplying it with a positive factor η^-. To ensure convergence, the previous partial derivative $\frac{\partial E}{\partial [w_i]_x}(k-1)$ was replaced by an exponentially weighted average of the current and past partial derivatives with θ as the base and the epoch as the exponent. This weighted average was denoted by $\frac{\partial \bar{E}}{\partial [w_i]_x}(k)$, and was defined above.

2.2.4 SuperSAB

The last algorithm in this category that we present is called *SuperSAB* and was proposed by Tollenaere in [208]. It is very similar to the delta-bar-delta algorithm, in that a different learning

rate is used for each weight in the network. The update rule is the same as in the previous case:

$$\begin{aligned}\Delta w_i(k) &= -\left([\varepsilon_i(k)]_a \frac{\partial E}{\partial [w_i]_a}(k) + \imath[\varepsilon_i(k)]_b \frac{\partial E}{\partial [w_i]_b}(k)\right. \\ &\quad \left. +\jmath[\varepsilon_i(k)]_c \frac{\partial E}{\partial [w_i]_c}(k) + \kappa[\varepsilon_i(k)]_d \frac{\partial E}{\partial [w_i]_d}(k)\right),\end{aligned}$$

with the exception that here the learning rate $\varepsilon_i(k)$ corresponding to weight w_i and epoch k has a different update formula: [165]

$$[\varepsilon_i(k)]_x = \begin{cases} \eta^+[\varepsilon_i(k-1)]_x, & \text{if } \frac{\partial E}{\partial [w_i]_x}(k-1) \times \frac{\partial E}{\partial [w_i]_x}(k) > 0 \\ \eta^-[\varepsilon_i(k-1)]_x, & \text{if } \frac{\partial E}{\partial [w_i]_x}(k-1) \times \frac{\partial E}{\partial [w_i]_x}(k) < 0 \\ [\varepsilon_i(k-1)]_x, & \text{else} \end{cases}, \ \forall x \in \{a,b,c,d\}.$$

Thus, SuperSAB can be seen as a combination of the RPROP and delta-bar-delta algorithms. Exactly like in the resilient backpropagation algorithm, the hyperparameters also have to satisfy $0 < \eta^- < 1 < \eta^+$. Because of this resemblance, we took in our experiments the same values for these hyperparameters as above, namely $\eta^- = 0.5$ and $\eta^+ = 1.2$.

Instead of adding a term to the learning rate if the partial derivatives have the same sign, like in the delta-bar-delta algorithm, we multiply it by the factor η^+, like in the RPROP algorithm. Also, the last two partial derivatives were taken into account with their original values, unlike the exponentially weighted average from the delta-bar-delta algorithm.

2.3 Conjugate gradient algorithms

Conjugate gradient methods belong to the larger class of line search algorithms, which replace the negative gradient of the gradient descent method with some particular search direction, and then determine the minimum of the error function in that direction, thus yielding a real number that tells us how far we should move in the search direction, which replaces the learning rate. These methods generally perform better than the classical gradient descent. For the full deduction of the conjugate gradient algorithms in the real-valued case, see [18, 94].

We start with the conjugate gradient algorithm for the real-valued case, in which the error function $E(\mathbf{w})$ can be viewed as $E(\overset{\mathcal{R}}{\mathbf{w}})$. The iteration for calculating the value $\overset{\mathcal{R}}{\mathbf{w}}^*$, for which the minimum of the function $E(\overset{\mathcal{R}}{\mathbf{w}})$ is attained, is

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k, \tag{2.3.1}$$

where $\overset{\mathcal{R}}{\mathbf{p}}_k \in \mathbb{R}^{4N}$ represents the search direction. The value of $\alpha_k \in \mathbb{R}$ can be determined exactly using the formula

$$\alpha_k = -\frac{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{g}}_k}{\overset{\mathcal{R}}{\mathbf{p}}_k^T \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{p}}_k}, \tag{2.3.2}$$

where $\overset{\mathcal{R}}{\mathbf{g}}_k := \nabla_{\overset{\mathcal{R}}{\mathbf{w}}_k} E$ and $\overset{\mathcal{R}}{\mathbf{H}}_k := \nabla^2_{\overset{\mathcal{R}}{\mathbf{w}}_k} E$, but, because of the computational burden, we can replace the explicit calculation of α_k with an inexact line search that minimizes $E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) = E(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k)$, i.e., a line minimization along the search direction $\overset{\mathcal{R}}{\mathbf{p}}_k$, starting at the point

$\overset{\mathcal{R}}{\mathbf{w}}_k$. In our experiments, we used the golden section search, which is guaranteed to have linear convergence, see [118].

Using (2.1.1), equation (2.3.1) becomes: [154]

$$\frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{w}}_k + \alpha_k \frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{p}}_k,$$

or, equivalently,

$$\overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \overset{\mathcal{H}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{H}}{\mathbf{p}}_k. \tag{2.3.3}$$

Now, the iteration for the next search direction is given by

$$\overset{\mathcal{R}}{\mathbf{p}}_{k+1} = -\overset{\mathcal{R}}{\mathbf{g}}_{k+1} + \beta_k \overset{\mathcal{R}}{\mathbf{p}}_k, \tag{2.3.4}$$

where $\beta_k \in \mathbb{R}$ has different expressions, depending on the type of the conjugate gradient algorithm. For example, the *Hestenes-Stiefel* update expression (see [74]) for β_k is:

$$\beta_k = \frac{\overset{\mathcal{R}}{\mathbf{g}}_{k+1}^T (\overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k)}{\overset{\mathcal{R}}{\mathbf{p}}_k^T (\overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k)}. \tag{2.3.5}$$

From (2.1.1) and (2.1.2), we observe that (2.3.4) can be written as

$$\frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{p}}_{k+1} = -\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{g}}_{k+1} + \beta_k \frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{p}}_k,$$

or, equivalently, as

$$\overset{\mathcal{H}}{\mathbf{p}}_{k+1} = -\overset{\mathcal{H}}{\mathbf{g}}_k + \beta_k \overset{\mathcal{H}}{\mathbf{p}}_k, \tag{2.3.6}$$

because the $\frac{1}{4}$ factor can be absorbed into $\overset{\mathcal{H}}{\mathbf{p}}_k$.

Taking into account the fact that

$$\overset{\mathcal{R}}{\mathbf{q}}_1^T \overset{\mathcal{R}}{\mathbf{q}}_2 = \overset{\mathcal{R}}{\mathbf{q}}_1^H \overset{\mathcal{R}}{\mathbf{q}}_2 = \left(\frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{q}}_1\right)^H \frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{q}}_2 = \frac{1}{16}\overset{\mathcal{H}}{\mathbf{q}}_1^H \mathbb{J}_N \mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{q}}_2 = \frac{1}{4}\overset{\mathcal{H}}{\mathbf{q}}_1^H \overset{\mathcal{H}}{\mathbf{q}}_2,$$

the update expression (2.3.5) can be written as

$$\beta_k = \frac{\overset{\mathcal{H}}{\mathbf{g}}_{k+1}^H (\overset{\mathcal{H}}{\mathbf{g}}_{k+1} - \overset{\mathcal{H}}{\mathbf{g}}_k)}{\overset{\mathcal{H}}{\mathbf{p}}_k^H (\overset{\mathcal{H}}{\mathbf{g}}_{k+1} - \overset{\mathcal{H}}{\mathbf{g}}_k)}. \tag{2.3.7}$$

Up until now we have worked with vectors from $\mathbb{H}^{4N}$. Ideally, we would like to work with vectors directly in $\mathbb{H}^N$. Considering the definition of $\overset{\mathcal{H}}{\mathbf{q}}$ for $\mathbf{q} \in \mathbb{H}^N$, this is done by taking the first N elements of the vector $\overset{\mathcal{H}}{\mathbf{q}}$. Thus, equations (2.3.3) and (2.3.6) become, respectively,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k, \tag{2.3.8}$$

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{p}_k, \tag{2.3.9}$$

where $\mathbf{w}_k$, $\mathbf{p}_k$, $\mathbf{g}_k := \nabla_{\mathbf{w}_k} E \in \mathbb{H}^N$. A simple calculation shows that

$$\frac{1}{4}\overset{\mathcal{H}}{\mathbf{q}}_1^H \overset{\mathcal{H}}{\mathbf{q}}_2 = \operatorname{Re}\left(\mathbf{q}_1^H \mathbf{q}_2\right),$$

where $\mathrm{Re}(q)$ represents the real part of the quaternion q, i.e., $\mathrm{Re}(q) = q_a$, if $q = q_a + \imath q_b + \jmath q_c + \kappa q_d \in \mathbb{H}$, and so (2.3.7) becomes: [154]

$$\beta_k = \frac{\mathrm{Re}\left(\mathbf{g}_{k+1}^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}{\mathrm{Re}\left(\mathbf{p}_k^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}. \tag{2.3.10}$$

Relations (2.3.8), (2.3.9), and (2.3.10) now define the *quaternion-valued Hestenes-Stiefel* algorithm.

Similar calculations give the *quaternion-valued Polak-Ribiere* update expression (see [141]): [154]

$$\beta_k = \frac{\mathrm{Re}\left(\mathbf{g}_{k+1}^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}{\mathbf{g}_k^H \mathbf{g}_k},$$

and the *quaternion-valued Fletcher-Reeves* update formula (see [178]): [154]

$$\beta_k = \frac{\mathbf{g}_{k+1}^H \mathbf{g}_{k+1}}{\mathbf{g}_k^H \mathbf{g}_k}.$$

If the error function E is quadratic, then the conjugate gradient algorithm is guaranteed to find its minimum in at most N steps. But, in general, the error function may be far from quadratic, and so the algorithm will need more than N steps to approach the minimum. Over these steps, the conjugacy of the search directions tends to deteriorate, so it is a common practice to restart the algorithm with the negative gradient $\mathbf{p}_k = -\mathbf{g}_k$, after N steps.

A more sophisticated restart algorithm, proposed by Powell in [177], following an idea by Beale in [14], is to restart if there is little orthogonality left between the current gradient and the previous gradient. To test this, we verify that the following inequality holds:

$$|\mathrm{Re}\left(\mathbf{g}_k^H \mathbf{g}_{k+1}\right)| \geq 0.2 \mathbf{g}_{k+1}^H \mathbf{g}_{k+1}.$$

The update rule (2.3.9) for the search direction $\mathbf{p}_k$ is also changed to [154]

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{p}_k + \gamma_k \mathbf{p}_t,$$

where

$$\beta_k = \frac{\mathrm{Re}\left(\mathbf{g}_{k+1}^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}{\mathrm{Re}\left(\mathbf{p}_k^H(\mathbf{g}_{k+1} - \mathbf{g}_k)\right)}, \ \gamma_k = \frac{\mathrm{Re}\left(\mathbf{g}_{k+1}^H(\mathbf{g}_{t+1} - \mathbf{g}_t)\right)}{\mathrm{Re}\left(\mathbf{p}_t^H(\mathbf{g}_{t+1} - \mathbf{g}_t)\right)},$$

and $\mathbf{p}_t$ is an arbitrary downhill restarting direction. The conjugate gradient algorithm with these characteristics is called the *Powell-Beale* algorithm.

In order to apply the conjugate gradient algorithms to quaternion-valued feedforward neural networks, we only need to calculate the gradient $\mathbf{g}_k$ of the error function E at different steps, which we do by using the backpropagation algorithm.

2.4 Scaled conjugate gradient method

The scaled conjugate algorithm was proposed by Møller in [122], and brings two improvements to the conjugate gradient algorithm. The first one uses the model trust region method known from the Levenberg-Marquardt algorithm to ensure the positive definiteness of the Hessian matrix by adding to it a sufficiently large positive constant λ_k multiplied by the identity matrix. Thus, the formula for the step length given in (2.3.2), becomes in this case

$$\alpha_k = -\frac{\overset{\mathcal{R}}{\mathbf{p}_k}^T \overset{\mathcal{R}}{\mathbf{g}_k}}{\delta_k}, \tag{2.4.1}$$

where we denoted by $\delta_k := \overset{\mathcal{R}}{\mathbf{p}}_k^{T\,\mathcal{R}} \mathbf{H}_k \overset{\mathcal{R}}{\mathbf{p}}_k + \lambda_k \overset{\mathcal{R}}{\mathbf{p}}_k^{T\,\mathcal{R}} \overset{\mathcal{R}}{\mathbf{p}}_k$. If the Hessian matrix is positive definite, we have that $\delta_k > 0$. But, if $\delta_k \leq 0$, then we should increase the value of δ_k in order to make it positive. If we denote by $\overline{\delta_k}$ the new value of δ_k, and by $\overline{\lambda_k}$ the new value of λ_k, then $\overline{\delta_k}$ is given by

$$\overline{\delta_k} = \delta_k + (\overline{\lambda_k} - \lambda_k) \overset{\mathcal{R}}{\mathbf{p}}_k^{T\,\mathcal{R}} \overset{\mathcal{R}}{\mathbf{p}}_k, \tag{2.4.2}$$

where we take $\overline{\lambda_k} = 2\left(\lambda_k - \frac{\delta_k}{\overset{\mathcal{R}}{\mathbf{p}}_k^{T\,\mathcal{R}} \overset{\mathcal{R}}{\mathbf{p}}_k}\right)$, which ensures that $\overline{\delta_k} > 0$, and will be used in (2.4.1) to calculate the value of α_k.

The second improvement uses a comparison parameter to evaluate how good the quadratic approximation for the error function E really is, in the conjugate gradient algorithm. This parameter is defined by

$$\Delta_k = \frac{2(E(\overset{\mathcal{R}}{\mathbf{w}}_k) - E(\overset{\mathcal{R}}{\mathbf{w}}_k + \alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k))}{\alpha_k \overset{\mathcal{R}}{\mathbf{p}}_k^{T\,\mathcal{R}} \mathbf{g}_k}. \tag{2.4.3}$$

Based on the value of Δ_k, the parameter λ_k is then updated in the following way:

$$\lambda_{k+1} = \begin{cases} \lambda_k/2, & \text{if } \Delta_k > 0.75 \\ 4\lambda_k, & \text{if } \Delta_k < 0.25 \\ \lambda_k, & \text{else} \end{cases},$$

in order to ensure a better quadratic approximation.

Thus, there are two stages for updating λ_k: one to ensure that $\delta_k > 0$ and one according to the validity of the local quadratic approximation. The two stages are applied successively after each weight update.

Using the same ideas as in the above section, relations (2.4.1), (2.4.2), and (2.4.3), become, respectively: [151]

$$\alpha_k = -\frac{\mathrm{Re}(\mathbf{p}_k^H \mathbf{g}_k)}{\delta_k},$$

$$\delta_k = \mathrm{Re}(\mathbf{p}_k^H \mathbf{H}_k \mathbf{p}_k) + \lambda_k \mathbf{p}_k^H \mathbf{p}_k,$$

$$\overline{\delta_k} = \delta_k + (\overline{\lambda_k} - \lambda_k) \mathbf{p}_k^H \mathbf{p}_k,$$

$$\overline{\lambda_k} = 2\left(\lambda_k - \frac{\delta_k}{\mathbf{p}_k^H \mathbf{p}_k}\right),$$

$$\Delta_k = \frac{2(E(\mathbf{w}_k) - E(\mathbf{w}_k + \alpha_k \mathbf{p}_k))}{\alpha_k \mathrm{Re}(\mathbf{p}_k^H \mathbf{g}_k)},$$

which, together, give the *quaternion-valued scaled conjugate gradient* algorithm.

2.5 Quasi-Newton learning methods

Again, we start with the quasi-Newton algorithm for the real-valued case, in which the function $E(\mathbf{w})$ can be viewed as $E(\overset{\mathcal{R}}{\mathbf{w}})$. The iteration for calculating the value $\overset{\mathcal{R}}{\mathbf{w}}^*$, for which the minimum of the function $E(\overset{\mathcal{R}}{\mathbf{w}})$ is attained, is

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - \alpha_k \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{g}}_k, \tag{2.5.1}$$

where $\overset{\mathcal{R}}{\mathbf{g}}_k := \nabla_{\overset{\mathcal{R}}{\mathbf{w}}_k} E \in \mathbb{R}^{4N}$ and $\overset{\mathcal{R}}{\mathbf{H}}_k \in \mathbb{R}^{4N\times 4N}$ is an approximation of the inverse of the Hessian matrix $\nabla^2_{\overset{\mathcal{R}}{\mathbf{w}}_k} E$. The value of $\alpha_k \in \mathbb{R}$ is determined using an inexact line search that minimizes $E(\overset{\mathcal{R}}{\mathbf{w}}_{k+1}) = E(\overset{\mathcal{R}}{\mathbf{w}}_k - \alpha_k \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{g}}_k)$. In our experiments, we used the golden section search, which is guaranteed to have linear convergence, see [118].

Using (2.1.2) and (2.1.3), we have that [155]

$$\overset{\mathcal{R}}{\mathbf{g}}_k = \mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{g}}_k$$

and

$$\overset{\mathcal{R}}{\mathbf{H}}_k = \frac{1}{16}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{H}}_k \mathbb{J}_N,$$

where $\overset{\mathcal{H}}{\mathbf{g}}_k := \nabla_{\overset{\mathcal{H}}{\mathbf{w}}_k} E \in \mathbb{H}^{4N}$, and we took into account the fact that $\overset{\mathcal{R}}{\mathbf{H}}_k$ approximates $(\nabla^2_{\overset{\mathcal{R}}{\mathbf{w}}_k} E)^{-1}$ and $\overset{\mathcal{H}}{\mathbf{H}}_k$ approximates $(\nabla^2_{\overset{\mathcal{H}}{\mathbf{w}}_k} E)^{-1}$. Thus, relation (2.5.1) can be written as

$$\frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{w}}_k - \alpha_k \frac{1}{16}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{H}}_k \mathbb{J}_N \mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{g}}_k,$$

where we also used relation (2.1.1), or, equivalently,

$$\overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \overset{\mathcal{H}}{\mathbf{w}}_k - \alpha_k \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{g}}_k. \tag{2.5.2}$$

Now, the update expression of the inverse Hessian approximation for the *symmetric rank-one (SR1)* method is

$$\overset{\mathcal{R}}{\mathbf{H}}_{k+1} = \overset{\mathcal{R}}{\mathbf{H}}_k + \frac{(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k)(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k)^T}{(\overset{\mathcal{R}}{\mathbf{p}}_k - \overset{\mathcal{R}}{\mathbf{H}}_k \overset{\mathcal{R}}{\mathbf{q}}_k)^T \overset{\mathcal{R}}{\mathbf{q}}_k}, \tag{2.5.3}$$

where

$$\overset{\mathcal{R}}{\mathbf{p}}_k := \overset{\mathcal{R}}{\mathbf{w}}_{k+1} - \overset{\mathcal{R}}{\mathbf{w}}_k,\ \overset{\mathcal{R}}{\mathbf{q}}_k := \overset{\mathcal{R}}{\mathbf{g}}_{k+1} - \overset{\mathcal{R}}{\mathbf{g}}_k.$$

Again from (2.1.1) and (2.1.2), we obtain

$$\overset{\mathcal{H}}{\mathbf{p}}_k := \overset{\mathcal{H}}{\mathbf{w}}_{k+1} - \overset{\mathcal{H}}{\mathbf{w}}_k,\ \overset{\mathcal{H}}{\mathbf{q}}_k := \overset{\mathcal{H}}{\mathbf{g}}_{k+1} - \overset{\mathcal{H}}{\mathbf{g}}_k.$$

Taking the above relations into account, equation (2.5.3) becomes

$$\frac{1}{16}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{H}}_{k+1} \mathbb{J}_N = \frac{1}{16}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{H}}_k \mathbb{J}_N + \frac{(\frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{p}}_k - \frac{1}{16}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{H}}_k \mathbb{J}_N \mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{q}}_k)(\frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{p}}_k - \frac{1}{16}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{H}}_k \mathbb{J}_N \mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{q}}_k)^H}{(\frac{1}{4}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{p}}_k - \frac{1}{16}\mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{H}}_k \mathbb{J}_N \mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{q}}_k)^H \mathbb{J}_N^H \overset{\mathcal{H}}{\mathbf{q}}_k},$$

which is equivalent to

$$\overset{\mathcal{H}}{\mathbf{H}}_{k+1} = \overset{\mathcal{H}}{\mathbf{H}}_k + \frac{(\overset{\mathcal{H}}{\mathbf{p}}_k - \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k)(\overset{\mathcal{H}}{\mathbf{p}}_k - \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k)^H}{(\overset{\mathcal{H}}{\mathbf{p}}_k - \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k)^H \overset{\mathcal{H}}{\mathbf{q}}_k}.$$

Up until now we have worked with vectors from $\mathbb{H}^{4N}$. Ideally, we would like to work with vectors directly in $\mathbb{H}^N$. Considering the definition of $\overset{\mathcal{H}}{\mathbf{q}}$ for $\mathbf{q} \in \mathbb{H}^N$, this is done by taking the first N elements of the vector $\overset{\mathcal{H}}{\mathbf{q}}$. For this, we denote

$$\overset{\mathcal{H}}{\mathbf{r}}_k := \overset{\mathcal{H}}{\mathbf{p}}_k - \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k,$$

and so

$$\mathbf{r}_k = \mathbf{p}_k - (\mathbf{H}_k^1 \mathbf{q}_k + \mathbf{H}_k^2 (\mathbf{q}_k)^{\imath} + \mathbf{H}_k^3 (\mathbf{q}_k)^{\jmath} + \mathbf{H}_k^4 (\mathbf{q}_k)^{\kappa}),$$

where $\mathbf{H}_k^1 := (\overset{\mathcal{H}}{\mathbf{H}}_k)_{11}$, $\mathbf{H}_k^2 := (\overset{\mathcal{H}}{\mathbf{H}}_k)_{12}$, $\mathbf{H}_k^3 := (\overset{\mathcal{H}}{\mathbf{H}}_k)_{13}$, $\mathbf{H}_k^4 := (\overset{\mathcal{H}}{\mathbf{H}}_k)_{14} \in \mathbb{H}^{N \times N}$ are block components of the matrix $\overset{\mathcal{H}}{\mathbf{H}}_k \in \mathbb{H}^{4N \times 4N}$. We can compute

$$(\overset{\mathcal{H}}{\mathbf{p}}_k - \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k)^H \overset{\mathcal{H}}{\mathbf{q}}_k = \overset{\mathcal{H}^H}{\mathbf{r}}_k \overset{\mathcal{H}}{\mathbf{q}}_k = 4\mathrm{Re}(\mathbf{r}_k^H \mathbf{q}_k),$$

where, as before, $\mathrm{Re}(q)$ represents the real part of the quaternion q, i.e., $\mathrm{Re}(q) = q_a$, if $q = q_a + \imath q_b + \jmath q_c + \kappa q_d \in \mathbb{H}$. The iteration (2.5.2) becomes [155]

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \left(\mathbf{H}_k^1 \mathbf{g}_k + \mathbf{H}_k^2 (\mathbf{g}_k)^{\imath} + \mathbf{H}_k^3 (\mathbf{g}_k)^{\jmath} + \mathbf{H}_k^4 (\mathbf{g}_k)^{\kappa} \right), \tag{2.5.4}$$

where the iterations for calculating the matrices $\mathbf{H}_k^1, \ldots, \mathbf{H}_k^4$ are:

$$\begin{aligned} \mathbf{H}_{k+1}^1 &= \mathbf{H}_k^1 + \frac{\mathbf{r}_k \mathbf{r}_k^H}{4\mathrm{Re}(\mathbf{r}_k^H \mathbf{q}_k)}, \\ &\vdots \\ \mathbf{H}_{k+1}^4 &= \mathbf{H}_k^4 + \frac{\mathbf{r}_k ((\mathbf{r}_k)^{\kappa})^H}{4\mathrm{Re}(\mathbf{r}_k^H \mathbf{q}_k)}. \end{aligned} \tag{2.5.5}$$

Relations (2.5.4) and (2.5.5) now give the *quaternion-valued symmetric rank-one (SR1)* method.

Proceeding in the same manner, we can deduce the update rule of the inverse Hessian approximation for the *Davidon-Fletcher-Powell (DFP)* method (see [54]) in the form

$$\overset{\mathcal{H}}{\mathbf{H}}_{k+1} = \overset{\mathcal{H}}{\mathbf{H}}_k + \frac{\overset{\mathcal{H}}{\mathbf{p}}_k \overset{\mathcal{H}^H}{\mathbf{p}}_k}{\overset{\mathcal{H}^H}{\mathbf{p}}_k \overset{\mathcal{H}}{\mathbf{q}}_k} - \frac{\overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k \overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{H}}_k}{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k}.$$

Now, by denoting $\overset{\mathcal{H}}{\mathbf{r}}_k := \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k$, we have that [155]

$$\mathbf{r}_k = \mathbf{H}_k^1 \mathbf{g}_k + \mathbf{H}_k^2 (\mathbf{g}_k)^{\imath} + \mathbf{H}_k^3 (\mathbf{g}_k)^{\jmath} + \mathbf{H}_k^4 (\mathbf{g}_k)^{\kappa},$$

and the matrices $\mathbf{H}_k^1, \ldots, \mathbf{H}_k^4$ are computed using the iterations:

$$\begin{aligned} \mathbf{H}_{k+1}^1 &= \mathbf{H}_k^1 + \rho_k \mathbf{p}_k \mathbf{p}_k^H - \sigma_k \mathbf{r}_k \mathbf{r}_k^H \\ &\vdots \\ \mathbf{H}_{k+1}^4 &= \mathbf{H}_k^4 + \rho_k \mathbf{p}_k ((\mathbf{p}_k)^{\kappa})^H - \sigma_k \mathbf{r}_k ((\mathbf{r}_k)^{\kappa})^H, \end{aligned} \tag{2.5.6}$$

where

$$\rho_k := \frac{1}{4\mathrm{Re}(\mathbf{p}_k^H \mathbf{q}_k)} \text{ and } \sigma_k := \frac{1}{4\mathrm{Re}(\mathbf{q}_k^H \mathbf{r}_k)}.$$

Thus, the *quaternion-valued Davidon-Fletcher-Powell (DFP)* method is given by equations (2.5.4) and (2.5.6).

The most popular quasi-Newton algorithm is the *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* method (see [195]), for which the following inverse Hessian approximation iteration can be similarly obtained:

$$\overset{\mathcal{H}}{\mathbf{H}}_{k+1} = \overset{\mathcal{H}}{\mathbf{H}}_k - \frac{\overset{\mathcal{H}}{\mathbf{p}}_k \overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{H}}_k}{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{p}}_k} - \frac{\overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k \overset{\mathcal{H}^H}{\mathbf{p}}_k}{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{p}}_k} + \frac{\overset{\mathcal{H}}{\mathbf{p}}_k \overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{q}}_k \overset{\mathcal{H}^H}{\mathbf{p}}_k}{\left(\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{p}}_k\right)^2} + \frac{\overset{\mathcal{H}}{\mathbf{p}}_k \overset{\mathcal{H}^H}{\mathbf{p}}_k}{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{p}}_k}.$$

In this case, if we denote $\overset{\mathcal{H}}{\mathbf{r}}_k := \overset{\mathcal{H}}{\mathbf{H}}_k \overset{\mathcal{H}}{\mathbf{q}}_k$ and $\overset{\mathcal{H}}{\mathbf{s}}_k := \overset{\mathcal{H}}{\mathbf{p}}_k \overset{\mathcal{H}^H}{\mathbf{q}}_k$, then

$$\mathbf{r}_k = \mathbf{H}_k^1 \mathbf{g}_k + \mathbf{H}_k^2 (\mathbf{g}_k)^\imath + \mathbf{H}_k^3 (\mathbf{g}_k)^\jmath + \mathbf{H}_k^4 (\mathbf{g}_k)^\kappa$$

and

$$\mathbf{s}_k = \mathbf{p}_k \mathbf{q}_k^H + (\mathbf{p}_k)^\imath \left((\mathbf{q}_k)^\imath\right)^H + (\mathbf{p}_k)^\jmath \left((\mathbf{q}_k)^\jmath\right)^H + (\mathbf{p}_k)^\kappa \left((\mathbf{q}_k)^\kappa\right)^H.$$

Now, the *quaternion-valued Broyden-Fletcher-Goldfarb-Shanno (BFGS)* method is given by relation (2.5.4) and the following updates for the matrices $\mathbf{H}_k^1, \ldots, \mathbf{H}_k^4$: [155]

$$\begin{aligned}
\mathbf{H}_{k+1}^1 &= \mathbf{H}_k^1 - \rho_k \mathbf{p}_k \mathbf{r}_k^H - \rho_k \mathbf{r}_k \mathbf{p}_k^H + \rho_k^2 \mathbf{s}_k \mathbf{s}_k^H + \rho_k \mathbf{p}_k \mathbf{p}_k^H, \\
&\vdots \\
\mathbf{H}_{k+1}^4 &= \mathbf{H}_k^4 - \rho_k \mathbf{p}_k ((\mathbf{r}_k)^\kappa)^H - \rho_k \mathbf{r}_k ((\mathbf{p}_k)^\kappa)^H + \rho_k^2 \mathbf{s}_k ((\mathbf{s}_k)^\kappa)^H + \rho_k \mathbf{p}_k ((\mathbf{p}_k)^\kappa)^H,
\end{aligned}$$

where

$$\rho_k := \frac{1}{4\mathrm{Re}(\mathbf{q}_k^H \mathbf{p}_k)}.$$

Lastly, of practical importance is the *one step secant (OSS)* method (see [12]), which pertains to the class of quasi-Newton algorithms, although it does not require storing the inverse Hessian approximation, like the BFGS method from which it was derived. The update rule (2.5.2) in this case has the form

$$\overset{\mathcal{H}}{\mathbf{w}}_{k+2} - \overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \alpha_{k+1} \left(-\overset{\mathcal{H}}{\mathbf{g}}_{k+1} + A_k \overset{\mathcal{H}}{\mathbf{p}}_k + B_k \overset{\mathcal{H}}{\mathbf{q}}_k \right),$$

where

$$A_k = -\left(\frac{1}{4} + \frac{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{q}}_k}{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{p}}_k}\right) \frac{\overset{\mathcal{H}^H}{\mathbf{p}}_k \overset{\mathcal{H}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{p}}_k} + \frac{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{p}}_k}, \; B_k = \frac{\overset{\mathcal{H}^H}{\mathbf{p}}_k \overset{\mathcal{H}}{\mathbf{g}}_{k+1}}{\overset{\mathcal{H}^H}{\mathbf{q}}_k \overset{\mathcal{H}}{\mathbf{p}}_k}.$$

We can process this to yield the relations that give the *quaternion-valued one step secant (OSS)* method: [155]

$$\mathbf{w}_{k+2} - \mathbf{w}_{k+1} = \alpha_{k+1} \left(-\mathbf{g}_{k+1} + A_k \mathbf{p}_k + B_k \mathbf{q}_k \right),$$

where

$$A_k = -\left(\frac{1}{4} + \frac{\mathbf{q}_k^H \mathbf{q}_k}{\mathrm{Re}(\mathbf{q}_k^H \mathbf{p}_k)}\right) \frac{\mathrm{Re}(\mathbf{p}_k^H \mathbf{g}_{k+1})}{\mathrm{Re}(\mathbf{q}_k^H \mathbf{p}_k)} + \frac{\mathrm{Re}(\mathbf{q}_k^H \mathbf{g}_{k+1})}{\mathrm{Re}(\mathbf{q}_k^H \mathbf{p}_k)}, \; B_k = \frac{\mathrm{Re}(\mathbf{p}_k^H \mathbf{g}_{k+1})}{\mathrm{Re}(\mathbf{q}_k^H \mathbf{p}_k)}.$$

Like in the conjugate gradient case, we calculate the gradient $\mathbf{g}_k$ of the error function E by using the quaternion-valued backpropagation algorithm.

2.6 Levenberg-Marquardt learning algorithm

We will first present the Levenberg-Marquardt algorithm for the real-valued case, and then we will extend it to the quaternion domain.

We start with the error function of the quaternion-valued neural network written in the form

$$E(\mathbf{w}) = \frac{1}{2}\sum_{m=1}^{M}(\mathbf{e}^m)^H\mathbf{e}^m,$$

where $\mathbf{e}^m = \begin{pmatrix} e_1^m \\ \vdots \\ e_c^m \end{pmatrix} \in \mathbb{H}^c$ represents the error (the difference between the actual output vector and the desired output vector) for training pattern m, $1 \leq m \leq M$, and c represents the number of output neurons of the network. Now, following the above discussion, the function $E(\mathbf{w})$ can be viewed as $E(\overset{\mathcal{R}}{\mathbf{w}})$:

$$E(\overset{\mathcal{R}}{\mathbf{w}}) = \frac{1}{2}\sum_{m=1}^{M}\overset{\mathcal{R}}{\mathbf{e}^m}^T\overset{\mathcal{R}}{\mathbf{e}^m}.$$

We can form the Jacobian matrix of dimension $4cM \times 4N$, whose elements are the partial derivatives of each error of the form $(e_j^m)_x$ with respect to the weights $(w_i)_y$, $1 \leq m \leq M$, $1 \leq j \leq c$, $1 \leq i \leq N$, $x, y \in \{a, b, c, d\}$:

$$\overset{\mathcal{R}}{\mathbf{J}} = \begin{pmatrix} \frac{\overset{\mathcal{R}}{\partial \mathbf{e}^1}}{\overset{\mathcal{R}}{\partial \mathbf{w}}} \\ \vdots \\ \frac{\overset{\mathcal{R}}{\partial \mathbf{e}^M}}{\overset{\mathcal{R}}{\partial \mathbf{w}}} \end{pmatrix}.$$

The gradient of the error function $E(\overset{\mathcal{R}}{\mathbf{w}})$ can be written as

$$\begin{aligned}
\overset{\mathcal{R}}{\mathbf{g}} := \nabla_{\overset{\mathcal{R}}{\mathbf{w}}}E &= \left(\frac{\partial E}{\partial \overset{\mathcal{R}}{\mathbf{w}}}\right)^T \\
&= \frac{1}{2}\sum_{m=1}^{M}\left(\frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathcal{R}}{\mathbf{w}}}\overset{\mathcal{R}}{\mathbf{e}^m} + \overset{\mathcal{R}}{\mathbf{e}^m}^T\frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}}{\partial \overset{\mathcal{R}}{\mathbf{w}}}\right)^T \\
&= \frac{1}{2}\sum_{m=1}^{M}\left(\overset{\mathcal{R}}{\mathbf{e}^m}^T\frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}}{\partial \overset{\mathcal{R}}{\mathbf{w}}^T} + \frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathcal{R}}{\mathbf{w}}^T}\overset{\mathcal{R}}{\mathbf{e}^m}\right) \\
&= \sum_{m=1}^{M}\frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathcal{R}}{\mathbf{w}}^T}\overset{\mathcal{R}}{\mathbf{e}^m} \\
&= \overset{\mathcal{R}}{\mathbf{J}}^T\overset{\mathcal{R}}{\mathbf{e}}, \qquad (2.6.1)
\end{aligned}$$

where $\overset{\mathcal{R}}{\mathbf{e}} = \begin{pmatrix} \overset{\mathcal{R}}{\mathbf{e}^1} \\ \vdots \\ \overset{\mathcal{R}}{\mathbf{e}^M} \end{pmatrix}$.

The Hessian matrix of the error function $E(\overset{\mathcal{R}}{\mathbf{w}})$ is

$$\begin{aligned}
\overset{\mathcal{R}}{\mathbf{H}} := \frac{\partial}{\partial \overset{\mathcal{R}}{\mathbf{w}}} \left(\frac{\partial E}{\partial \overset{\mathcal{R}}{\mathbf{w}}} \right)^T &= \sum_{m=1}^{M} \frac{\partial}{\partial \overset{\mathcal{R}}{\mathbf{w}}} \left(\frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathcal{R}}{\mathbf{w}}^T} \overset{\mathcal{R}}{\mathbf{e}^m} \right) \\
&= \sum_{m=1}^{M} \left(\frac{\partial^2 \overset{\mathcal{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathcal{R}}{\mathbf{w}} \partial \overset{\mathcal{R}}{\mathbf{w}}^T} \overset{\mathcal{R}}{\mathbf{e}^m} + \frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathcal{R}}{\mathbf{w}}^T} \frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}}{\partial \overset{\mathcal{R}}{\mathbf{w}}} \right) \\
&\approx \sum_{m=1}^{M} \frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}^T}{\partial \overset{\mathcal{R}}{\mathbf{w}}^T} \frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}}{\partial \overset{\mathcal{R}}{\mathbf{w}}} \\
&= \overset{\mathcal{R}}{\mathbf{J}}^T \overset{\mathcal{R}}{\mathbf{J}}, \qquad (2.6.2)
\end{aligned}$$

where we approximated the second derivatives of the errors with respect to each training pattern with 0.

One of the most effective from the class of second order methods used to minimize the error function, is, in theory, the Newton method, see [135]. Unfortunately, it can be computationally intensive because it needs the explicit calculation of the Hessian matrix of the error function, more precisely its inverse. It can also became unstable, if the Hessian matrix is not positive definite. To address this last problem, the trust region Newton method was developed, which has the update formula

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - (\overset{\mathcal{R}}{\mathbf{H}}_k + \lambda_k \mathbb{I}_{4N})^{-1} \overset{\mathcal{R}}{\mathbf{g}}_k,$$

where $\lambda_k \in \mathbb{R}$ is chosen so that $\overset{\mathcal{R}}{\mathbf{H}}_k + \lambda_k \mathbb{I}_{4N}$ is positive definite.

To address the first problem, using (2.6.1) and (2.6.2), the update formula for the trust region Newton method becomes

$$\overset{\mathcal{R}}{\mathbf{w}}_{k+1} = \overset{\mathcal{R}}{\mathbf{w}}_k - (\overset{\mathcal{R}}{\mathbf{J}}_k^T \overset{\mathcal{R}}{\mathbf{J}}_k + \lambda_k \mathbb{I}_{4N})^{-1} \overset{\mathcal{R}}{\mathbf{J}}_k^T \overset{\mathcal{R}}{\mathbf{e}}_k,$$

which is the update rule for the *Levenberg-Marquardt* method (see [121]). This method replaces the explicit calculation of the Hessian with a calculation of the Jacobian $\overset{\mathcal{R}}{\mathbf{J}}_k$, which is less computationally intensive, because it only contains first order partial derivatives.

Next, we will deduce the quaternion-valued Levenberg-Marquardt algorithm, starting from the above formula. The expression for the Jacobian of the function $E(\overset{\mathcal{H}}{\mathbf{w}})$ is: [150]

$$\overset{\mathcal{H}}{\mathbf{J}} = \begin{pmatrix} \frac{\partial \overset{\mathcal{H}}{\mathbf{e}^1}}{\partial \overset{\mathcal{H}}{\mathbf{w}}} \\ \vdots \\ \frac{\partial \overset{\mathcal{H}}{\mathbf{e}^M}}{\partial \overset{\mathcal{H}}{\mathbf{w}}} \end{pmatrix}.$$

Taking into account relations (2.1.1) and (2.1.2), we have that

$$\frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}}{\partial \overset{\mathcal{R}}{\mathbf{w}}} = \frac{\partial \overset{\mathcal{R}}{\mathbf{e}^m}}{\partial \overset{\mathcal{H}}{\mathbf{w}}} \mathbb{J}_N = \frac{\partial \left(\frac{1}{4} \mathbb{J}_c^H \overset{\mathcal{H}}{\mathbf{e}^m} \right)}{\partial \overset{\mathcal{H}}{\mathbf{w}}} \mathbb{J}_N = \frac{1}{4} \mathbb{J}_c^H \frac{\partial \overset{\mathcal{H}}{\mathbf{e}^m}}{\partial \overset{\mathcal{H}}{\mathbf{w}}} \mathbb{J}_N,$$

$\forall 1 \leq m \leq M$, hence

$$\overset{\mathcal{R}}{\mathbf{J}} = \frac{1}{4}\mathbb{J}_{cM}^{H}\overset{\mathcal{H}}{\mathbf{J}}\mathbb{J}_{N}.$$

From (2.1.1) we have $\overset{\mathcal{R}}{\mathbf{e}} = \frac{1}{4}\mathbb{J}_{cM}^{H}\overset{\mathcal{H}}{\mathbf{e}}$, and then

$$\begin{aligned}
\overset{\mathcal{R}}{\mathbf{J}}^{T}\overset{\mathcal{R}}{\mathbf{e}} = \overset{\mathcal{R}}{\mathbf{J}}^{H}\overset{\mathcal{R}}{\mathbf{e}} &= \left(\frac{1}{4}\mathbb{J}_{cM}^{H}\overset{\mathcal{H}}{\mathbf{J}}\mathbb{J}_{N}\right)^{H}\frac{1}{4}\mathbb{J}_{cM}^{H}\overset{\mathcal{H}}{\mathbf{e}} \\
&= \frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{J}}^{H}\mathbb{J}_{cM}\frac{1}{4}\mathbb{J}_{cM}^{H}\overset{\mathcal{H}}{\mathbf{e}} \\
&= \frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{J}}^{H}\overset{\mathcal{H}}{\mathbf{e}},
\end{aligned}$$

and

$$\begin{aligned}
\overset{\mathcal{R}}{\mathbf{J}}^{T}\overset{\mathcal{R}}{\mathbf{J}} = \overset{\mathcal{R}}{\mathbf{J}}^{H}\overset{\mathcal{R}}{\mathbf{J}} &= \left(\frac{1}{4}\mathbb{J}_{cM}^{H}\overset{\mathcal{H}}{\mathbf{J}}\mathbb{J}_{N}\right)^{H}\frac{1}{4}\mathbb{J}_{cM}^{H}\overset{\mathcal{H}}{\mathbf{J}}\mathbb{J}_{N} \\
&= \frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{J}}^{H}\mathbb{J}_{cM}\frac{1}{4}\mathbb{J}_{cM}^{H}\overset{\mathcal{H}}{\mathbf{J}}\mathbb{J}_{N} \\
&= \frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{J}}^{H}\overset{\mathcal{H}}{\mathbf{J}}\mathbb{J}_{N}.
\end{aligned}$$

With these relations, the update formula for the Levenberg-Marquardt algorithm becomes: $\frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{w}}_{k} - \left(\frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{J}}_{k}^{H}\overset{\mathcal{H}}{\mathbf{J}}_{k}\mathbb{J}_{N} + \lambda_{k}\mathbb{I}_{4N}\right)^{-1}\frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{J}}_{k}^{H}\overset{\mathcal{H}}{\mathbf{e}}_{k} \Leftrightarrow \frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \frac{1}{4}\mathbb{J}_{N}^{H}\overset{\mathcal{H}}{\mathbf{w}}_{k} - \frac{1}{4}\mathbb{J}_{N}^{H}\left(\overset{\mathcal{H}}{\mathbf{J}}_{k}^{H}\overset{\mathcal{H}}{\mathbf{J}}_{k} + \lambda_{k}\mathbb{I}_{4N}\right)^{-1}\overset{\mathcal{H}}{\mathbf{J}}_{k}^{H}\overset{\mathcal{H}}{\mathbf{e}}_{k}$, or, finally,

$$\overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \overset{\mathcal{H}}{\mathbf{w}}_{k} - \left(\overset{\mathcal{H}}{\mathbf{J}}_{k}^{H}\overset{\mathcal{H}}{\mathbf{J}}_{k} + \lambda_{k}\mathbb{I}_{4N}\right)^{-1}\overset{\mathcal{H}}{\mathbf{J}}_{k}^{H}\overset{\mathcal{H}}{\mathbf{e}}_{k}.$$

If we denote $\overset{\mathcal{H}}{\mathbf{H}}_{k} := \overset{\mathcal{H}}{\mathbf{J}}_{k}^{H}\overset{\mathcal{H}}{\mathbf{J}}_{k} + \lambda_{k}\mathbb{I}_{4N}$ and $\overset{\mathcal{H}}{\mathbf{g}}_{k} := \overset{\mathcal{H}}{\mathbf{J}}_{k}^{H}\overset{\mathcal{H}}{\mathbf{e}}_{k}$, the above update formula becomes:

$$\overset{\mathcal{H}}{\mathbf{w}}_{k+1} = \overset{\mathcal{H}}{\mathbf{w}}_{k} - \overset{\mathcal{H}}{\mathbf{H}}_{k}^{-1}\overset{\mathcal{H}}{\mathbf{g}}_{k}. \tag{2.6.3}$$

Now, we will try to find an expression for computing $\overset{\mathcal{H}}{\mathbf{J}}_{k}$. By denoting $\mathbf{J}_{k}^{1} := \frac{\partial \mathbf{e}_{k}}{\partial \mathbf{w}_{k}}$, $\mathbf{J}_{k}^{2} := \frac{\partial \mathbf{e}_{k}}{\partial (\mathbf{w}_{k})^{\imath}}$, $\mathbf{J}_{k}^{3} := \frac{\partial \mathbf{e}_{k}}{\partial (\mathbf{w}_{k})^{\jmath}}$, $\mathbf{J}_{k}^{4} := \frac{\partial \mathbf{e}_{k}}{\partial (\mathbf{w}_{k})^{\kappa}}$, we can deduce that [150]

$$\overset{\mathcal{H}}{\mathbf{J}}_{k} = \begin{pmatrix} \mathbf{J}_{k}^{1} & \mathbf{J}_{k}^{2} & \mathbf{J}_{k}^{3} & \mathbf{J}_{k}^{4} \\ \imath\mathbf{J}_{k}^{2}\imath^{-1} & \imath\mathbf{J}_{k}^{1}\imath^{-1} & \imath\mathbf{J}_{k}^{4}\imath^{-1} & \imath\mathbf{J}_{k}^{3}\imath^{-1} \\ \jmath\mathbf{J}_{k}^{3}\jmath^{-1} & \jmath\mathbf{J}_{k}^{4}\jmath^{-1} & \jmath\mathbf{J}_{k}^{1}\jmath^{-1} & \jmath\mathbf{J}_{k}^{2}\jmath^{-1} \\ \kappa\mathbf{J}_{k}^{4}\kappa^{-1} & \kappa\mathbf{J}_{k}^{3}\kappa^{-1} & \kappa\mathbf{J}_{k}^{2}\kappa^{-1} & \kappa\mathbf{J}_{k}^{1}\kappa^{-1} \end{pmatrix},$$

and, consequently, that

$$\overset{\mathcal{H}}{\mathbf{J}}_{k}^{H} = \begin{pmatrix} (\mathbf{J}_{k}^{1})^{H} & \imath(\mathbf{J}_{k}^{2})^{H}\imath^{-1} & \jmath(\mathbf{J}_{k}^{3})^{H}\jmath^{-1} & \kappa(\mathbf{J}_{k}^{4})^{H}\kappa^{-1} \\ (\mathbf{J}_{k}^{2})^{H} & \imath(\mathbf{J}_{k}^{1})^{H}\imath^{-1} & \jmath(\mathbf{J}_{k}^{4})^{H}\jmath^{-1} & \kappa(\mathbf{J}_{k}^{3})^{H}\kappa^{-1} \\ (\mathbf{J}_{k}^{3})^{H} & \imath(\mathbf{J}_{k}^{4})^{H}\imath^{-1} & \jmath(\mathbf{J}_{k}^{1})^{H}\jmath^{-1} & \kappa(\mathbf{J}_{k}^{2})^{H}\kappa^{-1} \\ (\mathbf{J}_{k}^{4})^{H} & \imath(\mathbf{J}_{k}^{3})^{H}\imath^{-1} & \jmath(\mathbf{J}_{k}^{2})^{H}\jmath^{-1} & \kappa(\mathbf{J}_{k}^{1})^{H}\kappa^{-1} \end{pmatrix}.$$

This means that the matrix $\overset{\mathcal{H}}{\mathbf{H}}_k$ has the form

$$\overset{\mathcal{H}}{\mathbf{H}}_k = \begin{pmatrix} \mathbf{H}_k^1 & \mathbf{H}_k^2 & \mathbf{H}_k^3 & \mathbf{H}_k^4 \\ \imath\mathbf{H}_k^2\imath^{-1} & \imath\mathbf{H}_k^1\imath^{-1} & \imath\mathbf{H}_k^4\imath^{-1} & \imath\mathbf{H}_k^3\imath^{-1} \\ \jmath\mathbf{H}_k^3\jmath^{-1} & \jmath\mathbf{H}_k^4\jmath^{-1} & \jmath\mathbf{H}_k^1\jmath^{-1} & \jmath\mathbf{H}_k^2\jmath^{-1} \\ \kappa\mathbf{H}_k^4\kappa^{-1} & \kappa\mathbf{H}_k^3\kappa^{-1} & \kappa\mathbf{H}_k^2\kappa^{-1} & \kappa\mathbf{H}_k^1\kappa^{-1} \end{pmatrix},$$

where $\mathbf{H}_k^1 = (\mathbf{J}_k^1)^H\mathbf{J}_k^1 + \imath(\mathbf{J}_k^2)^H\mathbf{J}_k^2\imath^{-1} + \jmath(\mathbf{J}_k^3)^H\mathbf{J}_k^3\jmath^{-1} + \kappa(\mathbf{J}_k^4)^H\mathbf{J}_k^4\kappa^{-1} + \lambda_k\mathbb{I}_N$, $\mathbf{H}_k^2 = (\mathbf{J}_k^1)^H\mathbf{J}_k^2 + \imath(\mathbf{J}_k^2)^H\mathbf{J}_k^1\imath^{-1} + \jmath(\mathbf{J}_k^3)^H\mathbf{J}_k^4\jmath^{-1} + \kappa(\mathbf{J}_k^4)^H\mathbf{J}_k^3\kappa^{-1}$, $\mathbf{H}_k^3 = (\mathbf{J}_k^1)^H\mathbf{J}_k^3 + \imath(\mathbf{J}_k^2)^H\mathbf{J}_k^4\imath^{-1} + \jmath(\mathbf{J}_k^3)^H\mathbf{J}_k^1\jmath^{-1} + \kappa(\mathbf{J}_k^4)^H\mathbf{J}_k^2\kappa^{-1}$, $\mathbf{H}_k^4 = (\mathbf{J}_k^1)^H\mathbf{J}_k^4 + \imath(\mathbf{J}_k^2)^H\mathbf{J}_k^3\imath^{-1} + \jmath(\mathbf{J}_k^3)^H\mathbf{J}_k^2\jmath^{-1} + \kappa(\mathbf{J}_k^4)^H\mathbf{J}_k^1\kappa^{-1}$. Furthermore, we have that $\overset{\mathcal{H}}{\mathbf{g}}_k = \begin{pmatrix} \mathbf{g}_k \\ \imath\mathbf{g}_k\imath^{-1} \\ \jmath\mathbf{g}_k\jmath^{-1} \\ \kappa\mathbf{g}_k\kappa^{-1} \end{pmatrix}$, where $\mathbf{g}_k = (\mathbf{J}_k^1)^H\mathbf{e}_k + \imath(\mathbf{J}_k^2)^H\mathbf{e}_k\imath^{-1} + \jmath(\mathbf{J}_k^3)^H\mathbf{e}_k\jmath^{-1} + \kappa(\mathbf{J}_k^4)^H\mathbf{e}_k\kappa^{-1}$. Now we have all the necessary ingredients to compute the update rule given in (2.6.3).

Up until now we have worked with vectors from $\mathbb{H}^{4N}$. Ideally, we would like to work with vectors directly in $\mathbb{H}^N$. Considering the definition of $\overset{\mathcal{H}}{\mathbf{q}}$ for $\mathbf{q} \in \mathbb{H}^N$, this is done by taking the first N elements of the vector $\overset{\mathcal{H}}{\mathbf{q}}$. By using the Banachiewicz inversion formula [237], relation (2.6.3) thus becomes: [150]

$$\mathbf{w}_{k+1} = \mathbf{w}_k - (\mathbf{H}_k^1)^{-1}\mathbf{g}_k + (\mathbf{H}_k^1)^{-1}\begin{pmatrix}\mathbf{H}_k^2 & \mathbf{H}_k^3 & \mathbf{H}_k^4\end{pmatrix}\mathbf{T}^{-1}\begin{pmatrix} -\imath\mathbf{H}_k^2\imath^{-1}(\mathbf{H}_k^1)^{-1}\mathbf{g}_k + \imath\mathbf{g}_k\imath^{-1} \\ -\jmath\mathbf{H}_k^3\jmath^{-1}(\mathbf{H}_k^1)^{-1}\mathbf{g}_k + \jmath\mathbf{g}_k\jmath^{-1} \\ -\kappa\mathbf{H}_k^4\kappa^{-1}(\mathbf{H}_k^1)^{-1}\mathbf{g}_k + \kappa\mathbf{g}_k\kappa^{-1} \end{pmatrix},$$

where $\mathbf{T} = \begin{pmatrix} \imath\mathbf{H}_k^1\imath^{-1} & \imath\mathbf{H}_k^4\imath^{-1} & \imath\mathbf{H}_k^3\imath^{-1} \\ \jmath\mathbf{H}_k^4\jmath^{-1} & \jmath\mathbf{H}_k^1\jmath^{-1} & \jmath\mathbf{H}_k^2\jmath^{-1} \\ \kappa\mathbf{H}_k^3\kappa^{-1} & \kappa\mathbf{H}_k^2\kappa^{-1} & \kappa\mathbf{H}_k^1\kappa^{-1} \end{pmatrix} - \begin{pmatrix} \imath\mathbf{H}_k^2\imath^{-1} \\ \jmath\mathbf{H}_k^3\jmath^{-1} \\ \kappa\mathbf{H}_k^4\kappa^{-1} \end{pmatrix}(\mathbf{H}_k^1)^{-1}\begin{pmatrix}\mathbf{H}_k^2 & \mathbf{H}_k^3 & \mathbf{H}_k^4\end{pmatrix}$, which represents the *quaternion-valued Levenberg-Marquardt (LM)* algorithm.

In this case too, the gradient of the error function at different steps is computed using the well-known backpropagation scheme.

2.7 Experimental results

2.7.1 Linear autoregressive process with circular noise

An important benchmark first proposed in [120], and used in [58, 59, 60, 61, 228] for the complex-valued case, and in [210, 223, 32, 229] for the quaternion-valued case, is the prediction of the quaternion-valued circular white noise $n(k) = n_a(k) + \imath n_b(k) + \jmath n_c(k) + \kappa n_d(k)$, where n_a, n_b, n_c, $n_d \sim \mathcal{N}(0,1)$, passed through the stable autoregressive filter given by

$$y(k) = 1.79y(k-1) - 1.85y(k-2) + 1.27y(k-3) - 0.41y(k-4) + n(k).$$

In our experiments, we trained quaternion-valued feedforward neural networks using the classical gradient descent algorithm (abbreviated GD), the quickprop algorithm (QCP), the resilient backpropagation algorithm (RPR), the delta-bar-delta algorithm (DBD), the SuperSAB algorithm (SAB), the conjugate gradient algorithm with Hestenes-Stiefel updates (CGHS), the conjugate gradient algorithm with Polak-Ribiere updates (CGPR), the conjugate gradient algorithm with Fletcher-Reeves updates (CGFR), the conjugate gradient algorithm with Powell-Beale restarts (CGPB), the scaled conjugate gradient algorithm (SCG), the quasi-Newton al-

gorithm with symmetric rank-one updates (SR1), the quasi-Newton algorithm with Davidon-Fletcher-Powell updates (DFP), the quasi-Newton algorithm with Broyden-Fletcher-Goldfarb-Shanno updates (BFGS), the one step secant method (OSS), and the Levenberg-Marquardt algorithm (LM).

The tap input of the filter was 4, so the networks had 4 inputs, 4 hidden neurons on a single hidden layer, and one output. The activation function for the hidden layer was the fully quaternion hyperbolic tangent function, given by $G^2(q) = \tanh q = \frac{e^q - e^{-q}}{e^q + e^{-q}}$, and the activation function for the output layer was the identity $G^3(q) = q$. Training was done for 5000 epochs with 5000 randomly generated training samples.

To evaluate the effectiveness of the algorithms, we used a measure of performance called *prediction gain*, defined by $R_p = 10 \log_{10} \frac{\sigma_x^2}{\sigma_e^2}$, where σ_x^2 represents the variance of the input signal and σ_e^2 represents the variance of the prediction error. The prediction gain is given in dB and it is obvious that, because of the way it is defined, a bigger prediction gain means better performance. After running each algorithm 50 times, the prediction gains are given in Table 2.1.

We can see that QCP, SAB, and RPR performed approximately the same, followed by DBD, but all of them performed better than the gradient descent algorithm. Then, CGHS and CGPR gave approximately the same results, with CGFR performing better and CGPB worse. The SCG algorithm was better than the conjugate gradient algorithms. From the quasi-Newton methods, DFP and SR1 gave approximately the same results, with BFGS performing better and OSS worse. The absolute best was the LM algorithm.

Table 2.1: Experimental results for linear autoregressive process with circular noise

Algorithm	Prediction gain
GD	**4.51±6.64e-2**
QCP	6.37±1.08e-1
RPR	**6.41±1.47e-1**
DBD	5.46±1.48e-1
SAB	6.40±1.31e-1
CGHS	5.17±1.30e-1
CGPR	5.19±8.14e-2
CGFR	**6.91±2.51e-1**
CGPB	5.00±9.57e-2
SCG	**7.36±9.25e-2**
SR1	6.73±2.34e-1
DFP	6.61±2.15e-1
BFGS	**7.23±3.80e-1**
OSS	5.11±2.04e-1
LM	***8.94±3.33e-1***
QESN [229]	3.57
AQESN [229]	3.51

2.7.2 3D Lorenz system

The 3D Lorenz system is given by the ordinary differential equations

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= \alpha(y-x) \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= -xz+\rho x-y \\ \frac{\mathrm{d}z}{\mathrm{d}t} &= xy-\beta z, \end{aligned}$$

where $\alpha = 10$, $\rho = 28$, and $\beta = 2/3$. This represents a chaotic time series prediction problem, and was used to test quaternion-valued neural networks in [7, 23, 209, 212, 33, 229].

The tap input of the filter was 4, and so the networks had 4 inputs, 4 hidden neurons, and one output neuron. The networks were trained for 5000 epochs with 1337 training samples, which result from solving the 3D Lorenz system on the interval $[0, 25]$, with the initial conditions $(x, y, z) = (1, 2, 3)$.

The results after 50 runs of each algorithm are given in Table 2.2. The measure of performance was the prediction gain, like in the above experiment.

In this case, QCP and RPR performed best, SAB followed, and DBD was again last of the four, but still better than GD. Next, CGHS and CGPB performed approximately in the same way, CGFR slightly better, and the best was CGPR. In this experiment also, SCG had better results than the conjugate gradient algorithms. From the quasi-Newton methods, DFP and SR1 performed approximately in the same way, OSS slightly better, and the best was BFGS. The best overall performance was attained by the LM algorithm.

Table 2.2: Experimental results for the 3D Lorenz system

Algorithm	Prediction gain
GD	**7.56±7.42e-1**
QCP	10.59±5.29e-1
RPR	**11.07±7.08e-1**
DBD	9.35±6.17e-1
SAB	10.33±7.09e-1
CGHS	10.04±6.65e-1
CGPR	**11.31±8.34e-1**
CGFR	10.69±5.81e-1
CGPB	10.12±7.35e-1
SCG	**12.58±6.44e-1**
SR1	11.74±6.82e-1
DFP	11.27±7.76e-1
BFGS	**13.74±6.30e-1**
OSS	12.09±7.806e-1
LM	***31.45±1.21e0***
QESN [229]	17.73
AQESN [229]	18.92

2.7.3 4D Saito chaotic circuit

Lastly, we experimented on the 4D Saito chaotic circuit given by

$$\begin{bmatrix} \frac{\mathrm{d}x_1}{\mathrm{d}t} \\ \frac{\mathrm{d}y_1}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -\alpha_1 & \alpha_1\beta_1 \end{bmatrix} \begin{bmatrix} x_1 - \eta\rho_1 h(z) \\ y_1 - \eta\frac{\rho_1}{\beta_1} h(z) \end{bmatrix}$$
$$\begin{bmatrix} \frac{\mathrm{d}x_2}{\mathrm{d}t} \\ \frac{\mathrm{d}y_2}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -\alpha_2 & \alpha_2\beta_2 \end{bmatrix} \begin{bmatrix} x_2 - \eta\rho_2 h(z) \\ y_2 - \eta\frac{\rho_2}{\beta_2} h(z) \end{bmatrix},$$

where $h(z) = \begin{cases} 1, & z \geq -1 \\ -1, & z \leq 1 \end{cases}$ is the normalized hysteresis value, and $z = x_1 + x_2$, $\rho_1 = \frac{\beta_1}{1-\beta_1}$, $\rho_2 = \frac{\beta_2}{1-\beta_2}$. The parameters are given by $(\alpha_1, \beta_1, \alpha_2, \beta_2, \eta) = (7.5, 0.16, 15, 0.097, 1.3)$. This is also a chaotic time series prediction problem, and was used as a benchmark for quaternion-valued neural networks in [5, 6, 7, 31, 210, 211, 32].

The networks had the same architectures as the ones described earlier, and were trained for 5000 epochs with 5249 training samples, which result from solving the 4D Saito chaotic circuit on the interval $[0, 10]$, with the initial conditions $(x_1, y_1, x_2, y_2) = (1, 0, 1, 0)$.

The prediction gains after 50 runs of each algorithm are given in Table 2.3.

In this last experiment, the performances were similar to those in the previous experiments: QCP, SAB, and RPR had approximately the same performance, followed by DBD, and finally by GD. Also, CGPR had the best performance, followed closely by CGPB, and lastly by CGFR and CGHS. The performance of the SCG algorithm was similar to the ones in the previous experiments. Between the quasi-Newton algorithms, OSS had the best performance, followed closely by BFGS, and lastly by SR1 and DFP. The conclusion is the same: the LM algorithm had the best performance among all the tested algorithms.

Table 2.3: Experimental results for the 4D Saito chaotic circuit

Algorithm	Prediction gain
GD	**5.76±1.70e-1**
QCP	11.49±6.47e-1
RPR	**11.58±7.91e-1**
DBD	6.28±3.36e-1
SAB	11.55±4.96e-1
CGHS	11.59±4.09e-1
CGPR	**13.64±3.67e-1**
CGFR	12.08±5.30e-1
CGPB	13.02±4.93e-1
SCG	**15.32±9.35-1**
SR1	11.71±6.73e-1
DFP	11.10±6.32e-1
BFGS	16.24±5.06e-1
OSS	**16.94±7.70e-1**
LM	***25.36±9.63e-1***

2.7.4 Discussion

The problem of linear autoregressive process with circular noise deals with time series prediction directly in the quaternion domain, because of the quaternion-valued circular white noise

$n(k)$, whereas the 3D Lorenz system and the 4D Saito chaotic circuit are real-valued problems cast into the domain of quaternions. The 3D Lorenz system uses only the three quaternion imaginary units for the three variables, and the 4D Saito chaotic circuit uses a full quaternion for its four variables. It was showed, for example in [7], that quaternion-valued feedforward neural networks perform better on these problems than the real-valued ones. This gives rise to the possibility that, in the future, more high-dimensional problems from the real domain be treated using quaternion-valued neural networks.

Although we only used time series prediction problems to illustrate the effectiveness of the deduced algorithms, pattern recognition problems like color image compression [89] and color night vision [104] could also benefit from using these learning methods. The domain of quaternion-valued neural networks is just starting to gain interest, which means that the future might bring more applications in the pattern recognition domain, also.

The performances of the algorithms were similar to the ones from the real-valued case, mainly because these learning methods were derived starting from their real-valued counterparts. The main difference is the quaternion multiplication, which gives rise to the specific formulations of these algorithms in the quaternion domain. However, the performances of the algorithms in each class differ from the ones in the real-valued case, mainly due to the quaternion dynamics, giving one more reason to extend these algorithms to the quaternions.

The computational costs for these algorithms are on the order of the computational cost of the same algorithms in the real-valued domain. In all cases, the learning methods scale well, meaning that the computational cost is four times that of the real version of the same method, which is the best that can be obtained, taking into account the fact that every quaternion is formed of four real numbers.

The $\mathbb{HR}$ calculus was used to deduce the algorithms starting from the real-valued case. Two other methods could have been used for the deduction. One is by splitting the quaternions into their four real components, and giving the formulation of the algorithms separately for each of the four components. However, this would have meant that the specific relations between components in the quaternion domain would have been lost, and also the formulations would have been much more cumbersome, because of the need to split the fully quaternion activation function into components. In principle, these algorithms would have also been deducible directly in the quaternion domain, but this type of deduction would have needed quaternion-domain derivatives and would have been much more error prone. Nonetheless, all the three methods give equivalent formulations, thus the easiest and most natural method is preferred.

Chapter 3

Complex-valued deep learning

3.1 Complex-valued convolutional neural networks

Convolutional neural networks have become one of the most successful models in solving virtually any image recognition task. Proposed for the first time in [107], where they were used for handwritten digit recognition, they were later applied to handwriting recognition in [108]. By 1995, applications of this type of networks appeared in image recognition, speech recognition, and time series prediction [106]. Convolutional neural networks represent a particularization of feedforward neural networks, in which matrix multiplication is replaced by convolution and the weight matrix is replaced by many convolution kernels with much smaller dimension than that of the weight matrices. Although they had many applications in computer vision [109], convolutional networks started gaining more popularity only with the increase in the available computational resources and their implementation using parallel computing on graphics processors (GPU).

The use of these computational resources allowed a reducing of training times by a factor of 100, giving way to models with an increasing number of layers and parameters, thus inaugurating the domain of deep learning [15, 17, 193, 64]. The basis of this domain is represented by the convolutional networks, for which the increase in the number of layers gives better performance, as opposed to feedforward networks, whose performance degrades for a big number of layers. The same field includes other network models, for which it has been proved mathematically that performance is directly proportional to the size of the model.

The domain of complex-valued deep learning has appeared in the last few years. Although feedforward complex-valued neural networks have been applied to image recognition [137], and a single layer complex-valued convolutional neural network was used in [72] for object detection in Polarimetric Synthetic Aperture Radar (PolSAR) images, only in the last few years deep learning algorithms using complex numbers were derived and used. For example, in [10] complex-valued autoencoders are proposed, which are a type of model belonging to the deep learning paradigm. In [20] a wavelet scattering network is proposed, which uses complex numbers. Neuron synchrony in a complex-valued Deep Boltzmann Machine (DBM) was discussed in [179], showing superior performances to the real-valued case.

Very recent works discuss complex-valued recurrent neural network models [8], as well as learning time series representations using complex-valued recurrent networks [188], both with similar if not superior results than the real-valued ones, and the existence of certain properties of these networks that do not appear in the real-valued case, which makes them suitable for certain types of applications. One of the most important papers in this domain is [215], which gives a mathematical motivation for complex-valued convolutional neural networks, showing

that they can be seen as nonlinear multiwavelet packets, thus making the mathematical analysis from the signal processing domain available for a rigorous formulation of the properties of complex-valued convolutional networks. Following the footsteps of this paper, it is expected that research in the complex-valued deep learning domain will increase in the coming years.

Taking the above discussion into consideration, a natural idea is to use complex-valued convolutional neural networks for image recognition, also taking into account the fact that some images are given by the imaging devices in complex form [105].

Synthetic Aperture Radars (SAR) are imaging systems that produce complex-valued images of the ground [51, 187, 30]. They can be Interferometric Synthetic Aperture Radars (InSAR) [204] or Polarimetric Synthetic Aperture Radars (PolSAR) [3, 71, 72]. Complex-valued neural networks were used for noise reduction, compression, and recognition of this type of complex-valued images. To date, to the best of our knowledge, there are only two attempts of recognition of this type of images directly in the complex domain, one using complex-valued feedforward neural networks, and one using a complex-valued convolutional neural network with a single layer, both models being more suitable to this problem than real-valued neural networks, which ignore the dependencies present in the data in the complex plane [71, 72]. This fact lead to the idea that deep convolutional neural networks could bring even better performance in this area.

On the other hand, functional Magnetic Resonance Imaging (fMRI) collects the data in complex form also, but the majority of studies on this type of imaging only use the amplitude of the data in their analysis, ignoring the frequency information [86]. Complex-valued neural networks have proven their superior performances in the reconstruction and analysis of such images [86, 73]. Complex-valued independent component analysis was also successfully applied for this type of images [182, 111, 234]. Previous observations open up the possibility of applying complex-valued convolutional neural networks for the recognition of different patterns that may appear in these images, taking into account all the information provided by the imaging devices, and not ignoring the frequency information as was done until now.

In this section, we start by applying complex-valued convolutional neural networks to real-valued image recognition, leaving the complex-valued image recognition using these networks as future work. The presentation in this section follows that in the author's papers [153] and [168].

3.1.1 Model formulation

Complex-valued convolutional neural networks (CVCNNs) have been introduced to handle complex-valued data organized as a (2D) grid, for example a complex-valued image. CVCNNs are particularizations of feedforward neural networks, in which the matrix multiplication of the inputs and weights is replaced by convolution:

$$h(x,y) = (f \bigstar g)(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) g(x-m, y-n),$$

for $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$, where $M,\ N$ are the dimensions of the 2D grid on which the convolution operation is applied.

A convolutional neural network is, generally, composed of three types of layers: convolutional, pooling, and fully connected layers.

For convolutional layers of CVCNNs, the inputs of a layer are complex-valued images organized into channels. The weights of the convolutional layers are represented by relatively small convolution kernels. In order to perform complex-valued convolution using computational frameworks that can mainly handle real-valued operations, we can write the complex

convolution in terms of two real-valued convolutions, as follows:

$$k\bigstar x = (k^R + \imath k^I)\bigstar(x^R + \imath x^I) = k^R\bigstar x^R - k^I\bigstar x^I + \imath(k^R\bigstar x^I + k^I\bigstar x^R),$$

from which we have that

$$\begin{aligned}(k\bigstar x)^R &= k^R\bigstar x^R - k^I\bigstar x^I\\ (k\bigstar x)^I &= k^R\bigstar x^I + k^I\bigstar x^R,\end{aligned}$$

where x represents an input, k represents a convolution kernel, x^R is the real part, and x^I is the imaginary part of complex matrix x.

In order to reduce the covariate shift of the output of a convolution layer, batch normalization was introduced [88]. In many models it has been shown to accelerate learning, and its use is standard in state-of-the-art deep neural networks. In CVCNNs, batch normalization is formulated as follows: given a complex-valued minibatch $\mathcal{B} = \{x_{1\ldots m}\}$, we have that

$$\mu_{\mathcal{B}}^R = \frac{1}{m}\sum_{i=1}^m x_i^R, \quad \mu_{\mathcal{B}}^I = \frac{1}{m}\sum_{i=1}^m x_i^I,$$

$$(\sigma_{\mathcal{B}}^R)^2 = \frac{1}{m}\sum_{i=1}^m \left(x_i^R - \mu_{\mathcal{B}}^R\right)^2, \quad (\sigma_{\mathcal{B}}^I)^2 = \frac{1}{m}\sum_{i=1}^m \left(x_i^I - \mu_{\mathcal{B}}^I\right)^2,$$

$$\widehat{x_i^R} = \frac{x_i^R - \mu_{\mathcal{B}}^R}{\sqrt{(\sigma_{\mathcal{B}}^R)^2 + \epsilon}}, \quad \widehat{x_i^I} = \frac{x_i^I - \mu_{\mathcal{B}}^I}{\sqrt{(\sigma_{\mathcal{B}}^I)^2 + \epsilon}},$$

$$y_i^R = \gamma^R\widehat{x_i^R} + \beta^R, \quad y_i^I = \gamma^I\widehat{x_i^I} + \beta^I,$$

where γ^R, β^R, γ^I, β^I are learnable parameters, and $y_i = y_i^R + \imath y_i^I$ is the output after batch normalization.

After the convolution operation, a nonlinearity is applied. Different choices are possible for the nonlinearity in the complex domain, but preliminary experiments done with CVCNNs have shown that the best results are obtained using the split-complex $\text{ReLU}_{\mathbb{C}}$ nonlinearity, which applies the ReLU function separately on the real and imaginary components of a complex number:

$$\text{ReLU}_{\mathbb{C}}(x^R + \imath x^I) = \text{ReLU}(x^R) + \imath\text{ReLU}(x^I) = \max(0, x^R) + \imath\max(0, x^I).$$

Different variants of the ReLU nonlinearity were proposed for the complex domain, such as modReLU [8]:

$$\text{modReLU}(x) = \begin{cases}(|x| + b)\frac{x}{|x|} & \text{if } |x| + b \geq 0\\ 0 & \text{otherwise}\end{cases},$$

where $b \in \mathbb{R}$ is a learnable parameter, and zReLU [67]:

$$\text{zReLU}(x) = \begin{cases}x & \text{if } x^R,\ x^I \geq 0\\ 0 & \text{otherwise}\end{cases},$$

but, unfortunately, the performance in the context of CVCNNs of these nonlinearities is worse than that of the split-complex $\text{ReLU}_{\mathbb{C}}$ defined above.

The second type of layer in convolutional networks is the pooling or subsampling layer, which is used to reduce the dimensionality of the input and obtain a more abstract representation of it. The most known types of pooling layers are max pooling and average pooling. In the max

pooling layer, any neighborhood of small dimension in an image is replaced by the pixel with the maximum value from that neighborhood. The average pooling layer replaces the neighborhood with the average of the pixels in that neighborhood. In the context of CVCNNs, these pooling operations can only be done separately on the real and imaginary components of the complex-valued inputs.

For the max pooling operation, this could mean that the real part of the output of the max pooling operation can come from one complex-valued pixel, and the imaginary part of the output from a different complex-valued pixel, thus introducing unwanted distortions in the final subsampled image. Preliminary experiments done using max pooling have shown that this can be an issue which degrades the performance of CVCNNs. For this reason, a better choice is the average pooling, because the average of a set of complex numbers is just the complex number whose real part is the average of the real parts and whose imaginary part is the average of the imaginary parts of the numbers in the set.

Lastly, the fully connected layers are the same as the hidden layers in a feedforward neural network. In order to use real-valued operations, the complex-valued matrix multiplication can be expressed similarly as the convolution:

$$\begin{aligned}(Wx)^R &= W^R x^R - W^I x^I \\ (Wx)^I &= W^R x^I + W^I x^R,\end{aligned}$$

where x is the input to the fully connected layer, and W is the weight matrix of the fully connected layer. The nonlinearity used for the fully connected layers is the same split-complex $\mathrm{ReLU}_{\mathbb{C}}$ nonlinearity, defined above.

Usually, the output of a convolutional network is represented by class scores assigned to each class using the softmax function, in conjunction with the negative log likelihood loss function. In the complex domain, a separate softmax function is applied to the real and imaginary parts of the last fully connected layer, and the final class scores are considered to be the average of the two softmax scores given by the real and imaginary parts. Also, the negative log likelihood loss is applied to the real and imaginary parts separately. This novel formulation of CVCNNs has shown slightly better results in preliminary experiments than the one using only the real part of the last fully connected layer for defining the class scores and as input to the negative log likelihood loss.

3.1.2 Experimental results

3.1.2.1 MNIST

The MNIST dataset contains handwritten digits, and was created by [108]. It has $60,000$ training samples and $10,000$ test samples, each being a grayscale image of size 28×28 pixels.

In our experiments, we trained real-valued and complex-valued convolutional neural networks. Both types of networks contained convolutional layers immediately followed by max pooling layers. Training was done using gradient descent, with a fixed learning rate of 0.1.

All the $60,000$ images were used for training, with no training set augmentation. The number of epochs was 50 for both types of networks. The first two convolutional and max pooling layers had 16 and 64 channels, respectively, the third, when it was present, had 256 channels, and the fully connected layer had 512 neurons, all for the complex-valued networks. The real-valued networks had approximately 1.41 times more channels and neurons, to give the same number of real parameters for both types of networks and thus assure a fair comparison. The max pooling layers all had kernel size 2.

The experimental results are given in Table 3.1. In the first column of the table, the sizes of the convolution kernels are given. The results show that the complex-valued networks had better performance than the real-valued networks in all the experiments. It can be observed that the bigger the kernel sizes, the bigger the improvement. Training times differed only slightly, mainly because the real- and complex-valued networks had the same number of real parameters.

Thus, it can be said that complex-valued convolutional networks perform better than their real-valued counterparts, and their performance increases with the increase in the size of the network.

Table 3.1: Experimental results for MNIST

Convolution kernel sizes	Real-valued error	Complex-valued error
$3, 2$	0.70%	$\mathbf{0.65\%}$
$3, 4$	0.59%	$\mathbf{0.56\%}$
$5, 3$	0.52%	$\mathbf{0.50\%}$
$5, 5$	0.52%	$\mathbf{0.48\%}$
$5, 3, 2$	0.46%	$\mathbf{0.44\%}$

3.1.2.2 CIFAR-10

The CIFAR-10 dataset contains images in 10 classes, like airplane, bird, or truck, for example, and was created by [101]. There are $50,000$ training and $10,000$ test RGB color images, each of size 32×32 pixels. The experimental setting was the same as the one in the above set of experiments: training was done using gradient descent with a fixed learning rate of 0.1, convolutional layers were followed by max pooling layers, and the activation functions were real- and complex-valued ReLUs, respectively, for the convolutional and fully connected layers, and real- and complex-valued softmax, respectively, for the output layer.

The $50,000$ training set images were randomly cropped to size 24×24 pixels, which were then input to the networks. Training set augmentation was used, and included randomly flipping the images horizontally, and randomly adjusting the brightness and contrast of the images. Each network was trained for $50,000$ iterations, each iteration using a batch of 128 dynamically generated images. All the convolutional layers had 64 channels, and the two fully connected layers had 384 and 192 neurons, respectively, for the complex-valued case. In the real-valued case, the number of channels and neurons was increased approximately 1.41 times to ensure that both types of networks had the same number of real parameters, thus allowing for a fair comparison between them. Like before, the max pooling layers all had kernel size 2.

The obtained results are summarized in Table 3.2, in the same manner as in the above set of experiments, with the kernel sizes of the convolution layers in the first column, and the real- and complex-valued errors following. The conclusion is the same: in all the experiments, the complex-valued networks performed better, with the best increase in performance for the bigger kernel sizes.

Table 3.2: Experimental results for CIFAR-10

Convolution kernel sizes	Real-valued error	Complex-valued error
$3, 2$	16.95%	$\mathbf{16.71\%}$
$3, 4$	15.44%	$\mathbf{14.90\%}$
$5, 3$	15.02%	$\mathbf{14.47\%}$
$5, 5$	14.41%	$\mathbf{13.83\%}$
$5, 3, 3$	15.85%	$\mathbf{15.36\%}$
$5, 5, 2$	15.41%	$\mathbf{14.73\%}$

3.2 Fourier transform-based complex-valued convolutional neural networks

Although the applications described in Section 3.1 are interesting, they are limited to complex-valued images, whereas the majority of images of interest are real-valued. Beside these uses in complex-valued imaging, CVCNNs were rarely used for real-valued image recognition. As shown in Section 3.1, CVCNNs have better performance than their real-valued counterparts (RVCNNs) on real-valued image classification.

It is possible, however, to obtain a unique complex-valued representation of any real-valued image. This representation is given by the Discrete Fourier Transform (DFT) of an image. Based on this bijective correspondence between real-valued images and their complex-valued DFTs, it can be deduced that the problem of classifying real-valued images is equivalent with the problem of classifying their complex-valued DFTs.

Taking all the above observations into account, this section introduces complex-valued Fourier transform-based image classification using CVCNNs. The presentation follows the author's paper [175].

3.2.1 The Fourier transform

The French mathematician Jean-Baptiste Joseph Fourier proved that any periodic function can be expressed as a sum of sines and cosines of different frequencies, each multiplied by different coefficients, which is called the Fourier series. Based on it, the Fourier transform decomposes any function into its different frequencies. In the domain of image processing, the function represents a digital image, and thus we are concerned only with the Discrete Fourier Transform (DFT). The DFT has a wide range of applications in image processing, such as image analysis, image filtering, image denoising and reconstruction, and image compression. Our presentation of the main properties of the DFT mainly follows that of [63].

For an $M \times N$ image denoted by $f(x, y)$, with spatial coordinates $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$, the Discrete Fourier Transform (DFT) is defined as

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-\imath 2\pi(ux/M + vy/N)},$$

where $\imath^2 = -1$ is the complex imaginary unit, $u = 0, 1, 2, \ldots, M-1$ and $v = 0, 1, 2, \ldots, N-1$. One of the most important properties of the Fourier transform is that it is a bijective function on the space of images, which means that we can define the Inverse Discrete Fourier Transform

(IDFT) of $F(u, v)$ by:

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{\imath 2\pi(ux/M + vy/N)},$$

for $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$. This means that each image has one and only one Fourier transform, and the Fourier transform of an image can be used to retrieve the original image. The two form a discrete Fourier transform pair, and we will write

$$f(x, y) \leftrightarrow F(u, v).$$

As it can be easily seen, in general, the Fourier transform of a real-valued image is complex-valued because of the Euler formula: $e^{\imath x} = \cos x + \imath \sin x$. Thus, we can write

$$F(u, v) = R(u, v) + \imath I(u, v) = |F(u, v)| e^{\imath \phi(u,v)},$$

where

$$|F(u, v)| = \sqrt{R^2(u, v) + \imath I^2(u, v)},$$

is the Fourier (or frequency) spectrum and

$$\phi(u, v) = \arctan \left[\frac{I(u, v)}{R(u, v)} \right],$$

represents the phase angle, for $u = 0, 1, 2, \ldots, M-1$ and $v = 0, 1, 2, \ldots, N-1$.

The DFT of an image satisfies the following properties:

$$f(x, y) e^{\imath 2\pi(u_0 x/M + v_0 y/N)} \leftrightarrow F(u - u_0, v - v_0),$$

and

$$f(x - x_0, y - y_0) \leftrightarrow F(u, v) e^{-\imath 2\pi(u x_0/M + v y_0/N)}.$$

Based on these properties, the Fourier transform of an image can be shifted so that $F(0, 0)$ is at the point $(u_0, v_0) = (M/2, N/2)$, using the transformation

$$f(x, y)(-1)^{x+y} \leftrightarrow F(u - M/2, v - N/2).$$

Because of the periodicity of the DFT, which states that

$$F(u, v) = F(u + k_1 M, v + k_2 N),$$

for any integers k_1, k_2, the transform $F(u - M/2, v - N/2)$ contains the exact information that the original $F(u, v)$ contains, but organized in a different way.

Taking all of the above properties of the Fourier transform into account, the problem of classifying real-valued images is equivalent with the problem of classifying their complex-valued Fourier transforms. Because the Fourier transform offers a richer representation of an image, it can constitute the basis for classification algorithms that could potentially work better than their real-valued counterparts on the original images. This means that CVCNNs can be used to perform Fourier transform-based image classification of real-valued images.

3.2.2 Experimental results

All the experiments were done using the same network architectures for the real-valued convolutional neural networks (RVCNNs), complex-valued convolutional neural networks (CVCNNs) trained on the real-valued images (for which the imaginary part of the input was set to zero), and complex-valued convolutional networks trained on the complex-valued Discrete Fourier Transform (DFT) of the real-valued images.

The architecture consists of two (or three) convolutional layers, each followed by an average pooling layer. The number of input channels of the first convolutional layer was 1 or 3, depending on whether the input images were grayscale or RGB images. The number of input and output channels of the rest of the convolutional layers was 64 for the RVCNNs. For the CVCNNs trained on the real-valued images, this number was 45, which is approximately 1.41 times smaller than 64, thus assuring the same number of real parameters for the two networks. Because the CVCNNs trained on Fourier-transformed images have to deal with fully complex-valued inputs, the number of input and output channels for the convolutional layers was also 64, to ensure a fair comparison with the other networks. Thus, they have twice the number of real parameters, but also twice the number of real inputs. The average pooling was done over neighborhoods of 2×2 pixels.

The last average pooling layer was followed by two fully connected layers, which had 6 times, and 3 times, respectively, more units than the output channels of the convolutional layers (384, and 192, respectively, for RVCNNs and CVCNNs trained on Fourier-transformed images, and 270, and 135, respectively, for CVCNNs trained on real-valued images). The rest of the details of the CVCNNs are the ones mentioned in Section 3.1, which were similarly applied to the RVCNNs.

Training was done using stochastic gradient descent for 200 epochs, with the learning rate starting from 0.01, which was then divided by 10 two times, first after 100 epochs, and then after 150 epochs. One important aspect is that the DFTs of the input images were also shifted so that $F(0, 0)$ is at the center of the image, as mentioned in Section 3.2.1. This is usually a standard procedure in the image processing domain, and preliminary experiments have shown that training on the unshifted images has worse results than training on shifted ones.

3.2.2.1 MNIST

The full training set of $60,000$ was used for training the three types of networks described above, with no augmentations.

The experimental results are given in Table 3.3. The first column represents the dimensions of the convolution kernels for the first two (or three) convolutional layers. The second column represents the errors measured on the full $10,000$ image test set for the CVCNNs trained on the Discrete Fourier Transform (DFT) of the original images. Next, the errors on the test set for the CVCNNs trained on the original images are reported, for which the imaginary parts of the network inputs are considered to be zero. Lastly, the errors of the RVCNNs are given.

It can be seen that, although these last two types of networks have approximately the same number of parameters, CVCNNs perform better than RVCNNs. But even better performance than the CVCNNs trained on real-valued images was attained by the CVCNNs trained on the Fourier-transformed images. This shows that, in order to fully harness the power of complex-valued neural networks, they must be used with complex-valued inputs.

Table 3.3: Experimental results for MNIST

Kernel sizes	Complex DFT	Complex	Real
$3, 2$	$\mathbf{0.69\%}$	0.89%	0.98%
$3, 4$	$\mathbf{0.63\%}$	0.73%	0.79%
$5, 3$	$\mathbf{0.56\%}$	0.78%	0.96%
$5, 5$	$\mathbf{0.55\%}$	0.82%	0.88%
$5, 3, 2$	$\mathbf{0.54\%}$	0.65%	0.87%

3.2.2.2 SVHN

The Street View House Numbers (SVHN) dataset was created by [124], and consists of RGB images of digits cropped from house numbers in Google Street View images. The dataset contains $73,257$ images for training, $26,032$ images for testing, and $531,131$ additional, somewhat less difficult samples, for use as extra training data. We only use the standard training and testing images, with no data augmentation. The size of each sample in the dataset is 32×32 pixels. It is a more difficult task than MNIST, because the images are RGB, and may also contain distracting digits to the sides of the digit of interest.

The three types of networks described above were trained on this dataset, and the errors on the test set are reported in Table 3.4. The structure of the table is the same as the one in the previous experiment: first the kernel sizes of the convolutional layers are given, then the errors attained by the CVCNNs trained on the Fourier-transformed images, by the CVCNNs trained on the original images, and by the RVCNNs, respectively, are shown. The Fourier transform of an RGB image is computed separately for the three RGB channels, thus providing three complex-valued inputs for the CVCNNs.

The errors show that CVCNNs trained on the original images obtain better results than the RVCNNs trained on the same inputs. But the best performance was attained by the CVCNNs trained on the Fourier-transformed images.

Table 3.4: Experimental results for SVHN

Kernel sizes	Complex DFT	Complex	Real
$3, 2$	$\mathbf{8.66\%}$	10.97%	12.17%
$3, 4$	$\mathbf{8.42\%}$	10.61%	11.21%
$5, 3$	$\mathbf{7.94\%}$	9.40%	10.45%
$5, 5$	$\mathbf{8.16\%}$	9.10%	9.91%
$5, 3, 3$	$\mathbf{8.69\%}$	8.81%	9.27%

3.2.2.3 CIFAR-10

Training was done on the full $50,000$ image training set, which were only randomly flipped before being given to the networks. No other types of data augmentation were used.

The results of training the three types of networks are given in the same format as above, in Table 3.5. The errors are reported for the full $10,000$ image test set. The conclusion of the experiment is consistent with the ones of the above experiments: the CVCNNs perform better than their real-valued counterparts, but the CVCNNs trained on the Fourier-transformed images perform best.

Table 3.5: Experimental results for CIFAR-10

Kernel sizes	Complex DFT	Complex	Real
$3, 2$	$\mathbf{22.26\%}$	24.81%	27.13%
$3, 4$	$\mathbf{21.51\%}$	24.86%	24.97%
$5, 3$	$\mathbf{21.77\%}$	25.64%	25.99%
$5, 5$	$\mathbf{21.25\%}$	25.24%	25.68%
$5, 3, 3$	$\mathbf{21.61\%}$	21.88%	22.32%

3.3 Deep hybrid real–complex-valued convolutional neural networks

In Section 3.1, we showed that shallow CVCNN models have better performance on real-valued image recognition than their real-valued counterparts (RVCNNs). This gave rise to the idea of formulating deeper CVCNN models for use in image classification tasks. Thus, a first contribution of this section is the proposal of CVCNN architectures which are inspired by the well known VGG architecture [196], with the necessary adaptations to the complex domain.

On the other hand, because of their different parameter spaces, the structure of learning is different in CVCNNs than in RVCNNs. This means that the two types of networks learn different convolution kernels, and, as a consequence, will make systematically different errors. This fact led to the idea of a hybrid real–complex-valued ensemble, which combines the two types of networks, harnessing the advantages of both. The hybrid ensemble is expected to have significantly better performance than the individual CVCNN and RVCNN models, which we confirm by testing the proposed models on well known image classification datasets.

The presentation in this section follows that in the author's paper [168].

3.3.1 Model formulation

The architecture of the networks considered in this section is inspired by the one proposed by [196], which is a very popular deep convolutional network architecture. Taking into account the observations made in Section 3.1, we formulate real-valued and complex-valued variants of this architecture, given in Table 3.6.

In the table, the architectures are given with their established names, though there is only one fully connected layer which is used as input to the softmax function. The convolution kernels are all of dimension 3×3 pixels, which is the smallest dimension that can capture the notion of left/right, up/down, and center. The convolution is followed by batch normalization, which is not used in the original architecture of [196]. The number of channels is also given in the table, thus "Conv3BN-64" means a convolutional layer with 3×3 convolution kernels, followed by batch normalization (and the $\mathrm{ReLU}_{\mathbb{C}}$ nonlinearity which is omitted in this notation), with 64 channels.

Because of the discussion in Section 3.1, to ensure a fair comparison between the real and complex networks, average pooling is used, with the size of the neighborhood being 2×2 pixels for the first four average pooling layers, and 1×1 for the last average pooling layer, because we only work with 32×32 pixel images, and a neighborhood of 2×2 pixels would not be possible in this case. Thus, "AvgPool2" means an average pooling layer with 2×2 pixels neighborhood.

Lastly, "FC10" (or "FC100") signifies the fully connected layer with 10 channels for networks which classify images belonging to 10 classes and 100 channels for networks which classify images belonging to 100 classes.

Table 3.6: Real and complex convolutional neural network architectures

Real VGG11	**Real VGG13**	**Real VGG16**	**Real VGG19**
Conv3BN-64	Conv3BN-64	Conv3BN-64	Conv3BN-64
	Conv3BN-64	Conv3BN-64	Conv3BN-64
AvgPool2			
Conv3BN-128	Conv3BN-128	Conv3BN-128	Conv3BN-128
	Conv3BN-128	Conv3BN-128	Conv3BN-128
AvgPool2			
Conv3BN-256	Conv3BN-256	Conv3BN-256	Conv3BN-256
Conv3BN-256	Conv3BN-256	Conv3BN-256	Conv3BN-256
		Conv3BN-256	Conv3BN-256
			Conv3BN-256
AvgPool2			
Conv3BN-512	Conv3BN-512	Conv3BN-512	Conv3BN-512
Conv3BN-512	Conv3BN-512	Conv3BN-512	Conv3BN-512
		Conv3BN-512	Conv3BN-512
			Conv3BN-512
AvgPool2			
Conv3BN-512	Conv3BN-512	Conv3BN-512	Conv3BN-512
Conv3BN-512	Conv3BN-512	Conv3BN-512	Conv3BN-512
		Conv3BN-512	Conv3BN-512
			Conv3BN-512
AvgPool1			
FC10 (FC100)			

Complex VGG11	**Complex VGG13**	**Complex VGG16**	**Complex VGG19**
Conv3BN-45	Conv3BN-45	Conv3BN-45	Conv3BN-45
	Conv3BN-45	Conv3BN-45	Conv3BN-45
AvgPool2			
Conv3BN-90	Conv3BN-90	Conv3BN-90	Conv3BN-90
	Conv3BN-90	Conv3BN-90	Conv3BN-90
AvgPool2			
Conv3BN-180	Conv3BN-180	Conv3BN-180	Conv3BN-180
Conv3BN-180	Conv3BN-180	Conv3BN-180	Conv3BN-180
		Conv3BN-180	Conv3BN-180
			Conv3BN-180
AvgPool2			
Conv3BN-360	Conv3BN-360	Conv3BN-360	Conv3BN-360
Conv3BN-360	Conv3BN-360	Conv3BN-360	Conv3BN-360
		Conv3BN-360	Conv3BN-360
			Conv3BN-360
AvgPool2			
Conv3BN-360	Conv3BN-360	Conv3BN-360	Conv3BN-360
Conv3BN-360	Conv3BN-360	Conv3BN-360	Conv3BN-360
		Conv3BN-360	Conv3BN-360
			Conv3BN-360
AvgPool1			
FC10 (FC100)			

An important point that must be made is that the complex-valued architectures contain approximately 1.41 less channels for the convolutional layers than their real-valued counterparts. This is to ensure a fair comparison between the two types of architectures, which have, in this way, approximately the same number of real parameters. For example, where RVCNNs have a "Conv3BN-64" layer, the CVCNNs have a "Conv3BN-45" layer, because 45 is approximately 1.41 times smaller than 64.

However, because of the different structure of learning in the real and complex network architectures, although they are as similar as possible and have the same number of real parameters, it is expected that CVCNNs learn different convolution kernels than RVCNNs. This difference in learning means that the errors made by the CVCNNs are of different nature than those made by RVCNNs. Real-valued network ensembles give rather modest improvements in classification error. Because the structure of learning is very similar in these networks, the learned convolution kernels are also similar, which means that the networks forming the ensemble will make the same errors.

For this reason, we consider a promising idea to formulate a RVCNN–CVCNN ensemble in the form of a hybrid real–complex-valued convolutional network (RCVCNN). The output of the hybrid network is given by the average of the class scores given by the RVCNN and CVCNN components of the hybrid ensemble. This type of hybrid real–complex-valued ensemble is the simplest that can be formulated, and constitutes a first step done towards hybrid real–complex-valued networks.

3.3.2 Experimental results

The experiments were done using the RVCNN, CVCNN, and hybrid RCVCNN architectures described above. The RVCNNs and CVCNNs were trained separately, and their outputs were then averaged to give the output of the hybrid network. All inputs were real-valued RGB images of dimension 32×32 pixels, and so the networks had 3 input channels. The imaginary parts of the inputs for the CVCNNs were considered to be zero.

The networks were trained with stochastic gradient descent with minibatches of 128 images and momentum equal to 0.9. Weight decay was used with an L_2 penalty multiplier equal to $5 \cdot 10^{-4}$, like in [196]. The learning rate was set initially to 0.05, and then divided by 10 twice, after every 35 epochs. Thus the total number of epochs was 105, which is enough due to the use of batch normalization, which speeds up learning.

3.3.2.1 SVHN

The first dataset we test our networks on is the Street View House Numbers (SVHN) dataset. We only use the standard training and testing images, with no data augmentation.

The experimental results for training RVCNNs, CVCNNs, and for their hybrid ensemble are given in Table 3.7. In the first column, the name of the architecture is given, following the discussion in Section 3.3 and Table 3.6. Next, the error rate for the CVCNNs is given, followed by the error rate of the RVCNNs. It can be seen that, for the proposed type of architectures, the CVCNNs obtain better results than their real-valued counterparts. But, most importantly, the hybrid RCVCNN ensemble, formed with only two networks, has a much better performance than the best network of the ensemble, which is the CVCNN, in some cases as much as 10% in relative error improvement.

Table 3.7: Experimental results for SVHN

Architecture	Complex	Real	Real–Complex
VGG11	**4.19%**	4.68%	***4.06%***
VGG13	**3.93%**	4.10%	***3.66%***
VGG16	**3.96%**	3.98%	***3.59%***
VGG19	**3.63%**	3.75%	***3.24%***

3.3.2.2 CIFAR-10

Before being fed to the networks, the images in the dataset were padded with 4 pixels to a dimension of 40×40 pixels, which were then randomly cropped to 32×32 pixels, and then were randomly flipped. No other data augmentation was used.

The results of training the above-described convolutional architectures on this dataset are given in Table 3.8, in the same form as with the previous experiment. It can be seen from the table that CVCNNs performed better than RVCNNs, but the performance increase of the hybrid ensemble was significant compared to the CVCNNs.

Table 3.8: Experimental results for CIFAR-10

Architecture	Complex	Real	Real–Complex
VGG11	**8.49%**	10.13%	***8.17%***
VGG13	**7.10%**	7.12%	***6.27%***
VGG16	**6.50%**	6.95%	***5.96%***
VGG19	**6.51%**	8.08%	***6.40%***

3.3.2.3 CIFAR-100

The CIFAR-100 dataset contains the same images as CIFAR-10, but organized in 100 different classes, being much more specialized than the CIFAR-10 dataset [101]. Each of the 100 classes has 500 samples in the $50,000$ training set and 100 samples in the $10,000$ testing set. Because of the relatively big number of classes and the relatively small number of training samples per class, the classification errors are much bigger than the ones for the CIFAR-10 dataset.

The images were distorted in the same way as the ones in the previous experiment: random crop from the original image padded with 4 pixels, and random flip. The experimental results of RVCNNs and CVCNNs are given in Table 3.9 for the four different architectures. It can be seen that, in some cases, the performance improvements between CVCNNs and RVCNNs are important, but the biggest improvements were attained by the hybrid ensemble.

Table 3.9: Experimental results for CIFAR-100

Architecture	Complex	Real	Real–Complex
VGG11	**31.07%**	31.53%	***28.87%***
VGG13	**26.84%**	27.93%	***25.18%***
VGG16	**26.34%**	29.09%	***25.10%***
VGG19	**28.42%**	30.04%	***26.32%***

3.4 Complex-valued stacked denoising autoencoders

One of the first papers in the deep learning domain was [76], which has shown that Restricted Boltzmann Machines (RBMs) can be stacked to form a deep belief network, which could then in turn be used as an unsupervised pretraining procedure for deep neural networks. The idea was that greedy layer-wise pretraining [16] can be used to initialize deep architectures [52], which can then be fine-tuned via supervised learning to produce better results than the ones using random weight initialization. The random weight initialization prevented deep models from obtaining better performance than shallow ones.

Autoencoders are networks that learn to generate a representation of the input called a code, which can be then used to reconstruct the original input. Denoising autoencoders [220] are a special type of autoencoders which learn to reconstruct the original input from a corrupted version of it. Stacking denoising autoencoders is a similar idea to stacking RBMs to form deep belief networks, and was introduced by [221]. It has been shown in [221] that stacked denoising autoencoders match and even surpass the performance attained by deep belief networks.

In the complex domain, complex-valued linear autoencoders were proposed by [10], which also gave a training algorithm for them. But the most interesting types of autoencoders are nonlinear. Following the success of complex-valued convolutional neural networks for real-valued image classification presented in Section 3.1, and the above observations, we considered a promising idea to introduce complex-valued stacked denoising autoencoders.

The presentation in this section follows that in the author's paper [169].

3.4.1 Model formulation

A complex-valued autoencoder is composed of two parts: the encoder, which transforms the input into a representation called a code, and the decoder, which transforms this code into a reconstruction of the input. The goal of the autoencoder is thus to obtain a representation of its input, which, upon reconstruction, is as close as possible to the original. The following deduction of the complex-valued stacked denoising autoencoders follows its real-valued counterpart given in [221].

For a complex-valued encoder, we denote by $f(x)$ the mapping that transforms the input vector x of dimension d into its representation y of dimension d'. Let W denote the $d' \times d$ complex-valued weight matrix of the encoder and b its d'-dimensional bias. Then, the representation y is given by:

$$\begin{aligned} y &= y^R + \imath y^I \\ &= f(x) \\ &= \text{ReLU}_{\mathbb{C}}(Wx + b) \\ &= \text{ReLU}(W^R x^R - W^I x^I + b^R) + \imath \text{ReLU}(W^R x^I + W^I x^R + b^I), \end{aligned}$$

where $z = z^R + \imath z^I$, which means that z^R and z^I respectively denote the real and imaginary parts of complex matrix z, and $\imath$, with $\imath^2 = -1$, represents the complex imaginary unit. The above writing of the complex-valued representation y is necessary because the common computational frameworks that are used in the deep learning field mainly deal with real-valued operations, and thus we need to express all complex-valued relations using only real-valued operations.

The nonlinearity used is the split complex $\text{ReLU}_{\mathbb{C}}$ nonlinearity, which applies the ReLU

function to the real and imaginary parts of a complex number separately:

$$\begin{aligned} \text{ReLU}_{\mathbb{C}}(x^R + \imath x^I) &= \text{ReLU}(x^R) + \imath\text{ReLU}(x^I) \\ &= \max(0, x^R) + \imath \max(0, x^I). \end{aligned}$$

Because of its popularity in real-valued deep learning, we also use it in the complex domain.

For the complex-valued decoder, we denote by z the d-dimensional reconstruction of the representation y, and by $g(y)$ the mapping that does this reconstruction. We have that:

$$\begin{aligned} z &= z^R + \imath z^I \\ &= g(y) \\ &= \sigma_{\mathbb{C}}(W'y + b') \\ &= \sigma(W'^R y^R - W'^I y^I + b'^R) + \imath\sigma(W'^R y^I + W'^I y^R + b'^I), \end{aligned}$$

where W' represents the $d \times d'$ complex-valued weight matrix of the decoder, b' its d-dimensional weight, and $\sigma_{\mathbb{C}}$ represents the split complex sigmoid function:

$$\begin{aligned} \sigma_{\mathbb{C}}(x^R + \imath x^I) &= \sigma(x^R) + \imath\sigma(x^I) \\ &= \frac{1}{1 + e^{-x^R}} + \imath \frac{1}{1 + e^{-x^I}}. \end{aligned}$$

We will consider that $x \in [0, 1]^d + \imath[0, 1]^d$. We can also easily see that, because of the sigmoid nonlinearity, we also have $z \in [0, 1]^d + \imath[0, 1]^d$. The loss function which measures the reconstruction error is given by the cross-entropy loss:

$$\begin{aligned} L(x, z) &= -\sum_j \left[x_j^R \log z_j^R + (1 - x_j^R) \log(1 - z_j^R)\right] \\ &\quad -\sum_j \left[x_j^I \log z_j^I + (1 - x_j^I) \log(1 - z_j^I)\right]. \end{aligned}$$

If a good reconstruction of the input is given by the representation, then it has retained much of the information presented as input. But it is possible that the autoencoder simply learns the identity mapping, without discovering a more useful representation than the input, especially in the case when $d' > d$. Usually, autoencoders produce an under-complete representation in the case $d' < d$, and so we say that they produce a lossy compressed representation of the input. On the other hand, it is possible to avoid the learning of the identity mapping by an over-complete but sparse representation ($d' > d$).

A different approach to avoid the simple copying of the input by the autoencoder is to change the reconstruction criterion in order for it to perform a different task: to reconstruct the input from a corrupted version of it. This approach is called denoising, and was proposed by [221] for the real-valued case. The idea behind the approach is that, while learning to reconstruct the input, the autoencoder will also extract features that capture useful structure in the input.

Thus, a denoising autoencoder is trained to reconstruct the input x from a corrupted version $\widetilde{x}$ of x. As a consequence, the encoder becomes $y = f(\widetilde{x})$, and the decoder remains the same: $z = g(y)$. The observation is that, in this case, we are interested that the reconstruction z be as close as possible to the original input x, and not to the corrupted input $\widetilde{x}$. For this reason, the loss function is the same cross-entropy loss $L(x, z)$, defined above. Each time an input x is presented, the autoencoder will use a different corrupted version $\widetilde{x}$ of it, generated by the corruption process.

We consider two types of corruption processes:

- Additive Gaussian Noise (GN), in which $\widetilde{x}$ is given as $\widetilde{x} = \widetilde{x}^R + \imath\widetilde{x}^I \sim \mathcal{N}(x^R, \sigma^2/2) + \imath\mathcal{N}(x^I, \sigma^2/2)$;
- Masking Noise (MN), in which $\widetilde{x}$ is obtained by setting a fraction ν of the elements of x to $0 + 0\imath = 0$, randomly chosen for each example.

It has been shown that stacking Restricted Boltzmann Machines (RBMs) to form deep belief networks and autoencoders to form stacked autoencoders can be used as unsupervised pretraining procedures for deep neural networks [76, 75]. Likewise, denoising autoencoders can be stacked to form stacked denoising autoencoders. In such networks, each layer is trained separately, using greedy layer-wise training [16]. We must mention that the corrupted input to each individual layer is only used in the training phase, and after the mapping f has been learned, it is used on the uncorrupted layer input.

After all the mappings f have been learned for each of the layers of the stacked denoising autoencoder, a logistic regression layer can be added on top of the encoders, giving rise to a complex-valued deep neural network, which can be used for classification. The weights learned through unsupervised pretraining can then be fine-tuned using supervised training via gradient descent.

3.4.2 Experimental results

In our experiments, we train real-valued and complex-valued stacked denoising autoencoders, like the ones described above. The 28×28 pixel images are linearized as 784-dimensional vectors to be given to the networks. Each encoder is then pretrained using the greedy layer-wise procedure described above. The number of neurons for each complex-valued encoder is 1.41 times smaller then for its real-valued counterpart, to ensure approximately the same number of parameters in the two networks, and, as such, a fair comparison between them.

After the greedy layer-wise pretraining has learned the weights of each layer, a logistic regression layer which consists of the softmax function and the negative log likelihood loss function is added on top of the last encoder in the network. Learning then continues in a supervised manner, like in a deep neural network.

The Adam [98] algorithm with minibatches of 128 images was used for training. The learning rate was 0.001 and the number of epochs was 50 both for pretraining and for fine-tuning. The two types of noise used for corrupting the layer inputs were: Gaussian noise (GN) with standard deviation of $1/\sqrt{2}$ for the complex case, and of 1 for the real case; and masking noise (MN), for which a fraction of $\nu = 0.25$ (25%) of the inputs to the layers were set to 0.

3.4.2.1 MNIST

The experimental results for the MNIST dataset are given in Table 3.10. The first column in the table shows the architecture of the complex-valued network which has 1.41 times less neurons than the real-valued one. In the second and third columns, the value of the cross-entropy loss for the pretraining part computed on the test set is given for the complex-valued and real-valued autoencoders, respectively. Then, in the last two columns, the error (computed on the test set) after fine-tuning is given for the two types of networks. It can be seen from the table that the complex-valued networks have better performance than their real-valued counterparts both in terms of reconstruction loss and in terms of classification error.

Also, the reconstructed images by the real-valued and complex-valued stacked denoising autoencoders are given in Figure 3.1, along with the original images. It can be seen that the

quality of the images reconstructed by the complex-valued network is better than that of the images reconstructed by the real-valued network.

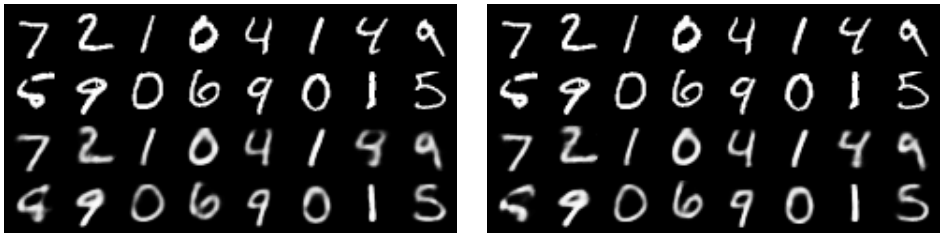

Figure 3.1: MNIST images reconstructed by the real-valued (left) and complex-valued (right) stacked denoising autoencoders, along with the original images [169]

Table 3.10: Experimental results for MNIST

Architecture	Complex Loss	Real Loss	Complex Error	Real Error
784-128-64-32 (GN)	$\mathbf{10.88e-4}$	$11.33e-4$	$\mathbf{1.44\%}$	1.49%
784-128-64-32 (MN)	$\mathbf{9.24e-4}$	$9.67e-4$	$\mathbf{1.71\%}$	1.91%
784-256-128-64 (GN)	$\mathbf{10.70e-4}$	$10.88e-4$	$\mathbf{1.18\%}$	1.26%
784-256-128-64 (MN)	$\mathbf{8.96e-4}$	$9.19e-4$	$\mathbf{1.35\%}$	1.63%
784-1024-512-256-128 (GN)	$\mathbf{10.64e-4}$	$10.83e-4$	$\mathbf{1.19\%}$	1.21%
784-1024-512-256-128 (MN)	$\mathbf{8.69e-4}$	$9.15e-4$	$\mathbf{1.18\%}$	1.24%

3.4.2.2 FashionMNIST

A dataset that appeared recently is the FashionMNIST dataset [230]. Its characteristics are very similar to the MNIST dataset, from which it was inspired, but it is more complicated. The grayscale 28×28 pixel images pertain to the following 10 classes: t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot. There are $60,000$ training samples and $10,000$ test samples.

The experimental results are given in the same way as with the above experiment in Table 3.11. It can be seen that the task is more complicated, both in terms of reconstruction error as well as in terms of classification error. Nonetheless, the conclusion of the set of experiments is the same: complex-valued networks have better reconstruction error and better classification error then the real-valued networks.

The reconstructed images for the two types of networks are given in Figure 3.2, along with the original images. In this case too, the complex-valued reconstructed images have a better quality.

3.5 Complex-valued deep belief networks

Among the first models in the paradigm of deep learning were deep belief networks. It was shown in [76] that Restricted Boltzmann Machines (RBMs) can be stacked to form a deep belief network, which, after training, can be used to initialize learning in a feedforward neural network [52]. Because deep belief networks are trained in an unsupervised manner, using the greedy layer-wise technique [16], the resulted networks can then be fine-tuned using supervised

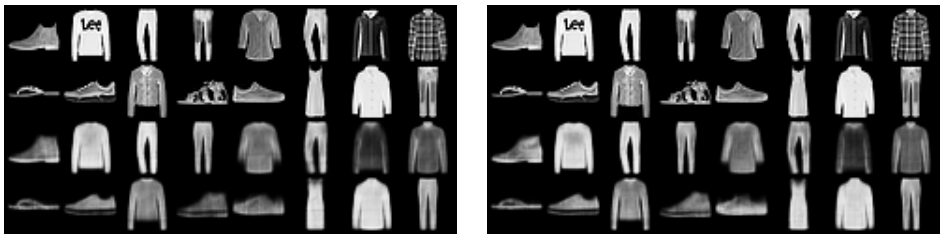

Figure 3.2: FashionMNIST images reconstructed by the real-valued (left) and complex-valued (right) stacked denoising autoencoders, along with the original images [169]

Table 3.11: Experimental results for FashionMNIST

Architecture	Complex Loss	Real Loss	Complex Error	Real Error
784-128-64-32 (GN)	$\mathbf{24.32e-4}$	$24.54e-4$	$\mathbf{10.37\%}$	10.72%
784-128-64-32 (MN)	$\mathbf{23.07e-4}$	$23.47e-4$	$\mathbf{10.15\%}$	10.82%
784-256-128-64 (GN)	$\mathbf{24.21e-4}$	$24.30e-4$	$\mathbf{10.30\%}$	10.37%
784-256-128-64 (MN)	$\mathbf{22.82e-4}$	$23.07e-4$	$\mathbf{10.21\%}$	10.45%
784-1024-512-256-128 (GN)	$\mathbf{24.23e-4}$	$24.29e-4$	$\mathbf{10.01\%}$	10.09%
784-1024-512-256-128 (MN)	$\mathbf{22.84e-4}$	$22.89e-4$	$\mathbf{9.97\%}$	10.05%

learning. As it turns out, this unsupervised pretraining of deep neural networks allows them to have better results than with random weight initialization.

On the footsteps of these papers, also taking into account the success of complex-valued convolutional neural networks for real-valued image classification demonstrated in Section 3.1, we introduce complex-valued deep belief networks. The presentation in this section follows that in the author's paper [170].

3.5.1 Model formulation

Restricted Boltzmann Machines (RBMs) are part of the larger family of energy-based models, which associate a scalar energy to each configuration of the variables of interest. Learning corresponds to modifying the energy function so that it has some desired properties. Usually, we would want the energy to be as low as possible. The following deduction of the properties of complex-valued RBMs follows that of [15] for the real-valued case.

For Boltzmann Machines (BMs), the energy function is linear in its free parameters. To increase the representational power of the Boltzmann Machines in order for them to be able to represent more complicated input distributions, some variables are considered to never be observed, and that is why they are called hidden variables. Restricted Boltzmann Machines (RBMs) restrict the BM model by not allowing visible-visible and hidden-hidden connections. If we denote the visible variables by v and the hidden variables by h, in the case of complex-valued RBMs, the energy function is defined by

$$\begin{aligned}
\mathcal{E}(v,h) &= -(b^H v)^R - (c^H h)^R - (h^H W v)^R \\
&= -(b^R)^T v^R - (b^I)^T v^I - (c^R)^T h^R - (c^I)^T h^I \\
&\quad -(h^R)^T (Wv)^R - (h^I)^T (Wv)^I \\
&= -(b^R)^T v^R - (b^I)^T v^I - (c^R)^T h^R - (c^I)^T h^I \\
&\quad -(v^R)^T (W^H h)^R - (v^I)^T (W^H h)^I, \qquad (3.5.1)
\end{aligned}$$

where z^R and z^I are the real and imaginary parts, respectively, of the complex-valued matrix z, z^H is the Hermitian (complex-conjugate) transpose of matrix z, a^T is the transpose of real-valued matrix a, and we used the property that $(x^H y)^R = (y^H x)^R$. Also, W represents the weights connecting the visible and hidden layers, b represents the bias of the visible layer, and c represents the bias of the hidden layer.

With the above notations, the probability distribution of complex-valued RBMs can be defined as

$$P(v) = \sum_h P(v,h) = \sum_h \frac{e^{-\mathcal{E}(v,h)}}{Z},$$

where Z is called the partition function by analogy with physical systems.

If we denote by

$$\mathcal{F}(v) = -\log \sum_h e^{-\mathcal{E}(v,h)},$$

which is called free energy (a notion also inspired from physics), we have that

$$P(v) = \frac{e^{-\mathcal{F}(v)}}{Z},$$

and Z is given by

$$Z = \sum_v e^{-\mathcal{F}(v)}.$$

Now, from (3.5.1), we have that

$$\begin{aligned}
\mathcal{F}(v) &= -(b^R)^T v^R - (b^I)^T v^I \\
&\quad - \sum_i \log \sum_{h_i^R} e^{h_i^R (c_i^R + (W_i v)^R)} - \sum_i \log \sum_{h_i^I} e^{h_i^I (c_i^I + (W_i v)^I)} \\
&= -(b^R)^T v^R - (b^I)^T v^I \\
&\quad - \sum_i \log \sum_{h_i^R} e^{h_i^R (c_i^R + W_i^R v^R - W_i^I v^I)} - \sum_i \log \sum_{h_i^I} e^{h_i^I (c_i^I + W_i^R v^I + W_i^I v^R)}, \quad (3.5.2)
\end{aligned}$$

where h_i is the ith element of vector h and W_i is the ith column of matrix W.

We can also obtain an expression for the conditional probabilities $P(h^R|v)$ and $P(h^I|v)$:

$$\begin{aligned}
P(h^R|v) &= \frac{e^{(b^R)^T v^R + (b^I)^T v^I + (c^R)^T h^R + (c^I)^T h^I + (h^R)^T (Wv)^R + (h^I)^T (Wv)^I}}{\sum_{\widetilde{h}^R} e^{(b^R)^T v^R + (b^I)^T v^I + (c^R)^T \widetilde{h}^R + (c^I)^T h^I + (\widetilde{h}^R)^T (Wv)^R + (h^I)^T (Wv)^I}} \\
&= \frac{\prod_i e^{c_i^R h_i^R + h_i^R (W_i v)^R}}{\prod_i \sum_{\widetilde{h_i}^R} e^{c_i^R \widetilde{h}_i^R + \widetilde{h}_i^R (W_i v)^R}} \\
&= \prod_i \frac{e^{h_i^R (c_i^R + (W_i v)^R)}}{\sum_{\widetilde{h_i}^R} e^{\widetilde{h}_i^R (c_i^R + (W_i v)^R)}} \\
&= \prod_i P(h_i^R|v),
\end{aligned}$$

$$\begin{aligned} P(h^I|v) &= \frac{e^{(b^R)^T v^R+(b^I)^T v^I+(c^R)^T h^R+(c^I)^T h^I+(h^R)^T (Wv)^R+(h^I)^T (Wv)^I}}{\sum_{\widetilde{h}^I} e^{(b^R)^T v^R+(b^I)^T v^I+(c^R)^T h^R+(c^I)^T \widetilde{h}^I+(h^R)^T (Wv)^R+(\widetilde{h}^I)^T (Wv)^I}} \\ &= \frac{\prod_i e^{c_i^I h_i^I+h_i^I (W_i v)^I}}{\prod_i \sum_{\widetilde{h_i}^I} e^{c_i^I \widetilde{h}_i^I+\widetilde{h}_i^I (W_i v)^I}} \\ &= \prod_i \frac{e^{h_i^I (c_i^I+(W_i v)^I)}}{\sum_{\widetilde{h_i}^I} e^{\widetilde{h}_i^I (c_i^I+(W_i v)^I)}} \\ &= \prod_i P(h_i^I|v). \end{aligned}$$

This means that the visible and hidden neurons are conditionally independent given one another.

If we assume that h_i^R, $h_i^I \in \{0, 1\}$, we obtain

$$P(h_i^R = 1|v) = \frac{e^{c_i^R+(W_i v)^R}}{1 + e^{c_i^R+(W_i v)^R}} = \sigma(c_i^R + (W_i v)^R) = \sigma(c_i^R + W_i^R v^R - W_i^I v^I),$$

$$P(h_i^I = 1|v) = \frac{e^{c_i^I+(W_i v)^I}}{1 + e^{c_i^I+(W_i v)^I}} = \sigma(c_i^I + (W_i v)^I) = \sigma(c_i^I + W_i^R v^I + W_i^I v^R),$$

where σ is the real-valued sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$. The above expressions, together with (3.5.2), prove that a complex-valued RBM can be implemented only using real-valued operations. This is important, because the computational frameworks used in the deep learning domain mainly deal with real-valued operations.

Because of the symmetry in the expression of the energy function between the visible and hidden neurons, assuming that v_j^R, $v_j^I \in \{0, 1\}$, the following relations can also be deduced:

$$P(v^R|h) = \prod_j P(v_j^R|h),$$

$$P(v^I|h) = \prod_j P(v_j^I|h),$$

$$P(v_j^R = 1|h) = \sigma(b_j^R + (W_j^H h)^R) = \sigma(b_j^R + (W_j^R)^T h^R + (W_j^I)^T h^I),$$

$$P(v_j^I = 1|h) = \sigma(b_j^I + (W_j^H h)^I) = \sigma(b_j^I + (W_j^R)^T h^I - (W_j^I)^T h^R).$$

The free energy for an RBM with binary neurons can be further simplified to

$$\begin{aligned} \mathcal{F}(v) &= -(b^R)^T v^R - (b^I)^T v^I \\ &\quad - \sum_i \log(1 + e^{c_i^R+W_i^R v^R-W_i^I v^I}) - \sum_i \log(1 + e^{c_i^I+W_i^R v^I+W_i^I v^R}). \end{aligned}$$

Samples from the distribution $P(x)$ can be obtained by running a Markov chain to convergence, using as transition operator the Gibbs sampling procedure. Gibbs sampling for N joint random variables $S = (S_1, \ldots, S_N)$ is done in a sequence of N sampling steps of the form $S_i \sim P(S_i|S_{-i})$, where S_{-i} denotes the other $N - 1$ variables that are not S_i. In the case of an RBM, this means that first we sample h^R, h^I from $P(h^R|v)$, $P(h^I|v)$, and then we sample v^R, v^I from $P(v^R|h)$, $P(v^I|h)$. By doing this procedure a sufficient amount of time, it is guaranteed that (v, h) is an accurate sample of $P(v, h)$. This would however be very computationally

expensive, and so different algorithms have been devised to sample from $P(v, h)$ efficiently during learning.

One such algorithm is contrastive divergence, which we use in our experiments. It is based on two ideas to speed up the sampling process. First, because we want to have $P(v) \approx P_{\text{train}}(v)$, i.e., the true distribution of the training data, we initialize the Markov chain described above with a training sample, which will speed up convergence. The second idea is that contrastive divergence doesn't wait for the Markov chain to converge, but only does k steps of Gibbs sampling. Surprisingly, $k = 1$ gives good results in practice.

Now that we have all the ingredients for constructing and training a complex-valued RBM, we can stack several complex-valued RBMs to form a complex-valued deep belief network. This type of network is trained one layer at a time, using the greedy layer-wise procedure [16]. After training the first layer to model the input, the following layers are trained as complex-valued RBMs to model the outputs of the previous layers. After learning the weights for all the RBMs in the deep belief network, a logistic regression layer is added on top of the last RBM in the deep belief network, thus forming a complex-valued deep neural network. This network can then be fine-tuned in a supervised manner, using gradient-based methods. Thus, the deep belief network is used to initialize the parameters of the deep neural network.

3.5.2 Experimental results

In the experiments, we use real-valued and complex-valued deep belief networks for the unsupervised pretraining of deep neural networks. The 28×28 images of the MNIST dataset were linearized into 784-dimensional vectors, which constitute the inputs of the networks. The number of neurons in every hidden layer of the complex-valued networks is given in the first column of Table 3.12. The real-valued networks had 1.41 times more neurons in the hidden layers, to ensure the same number of real parameters between the two types of networks, and thus a fair comparison. Each layer was trained in an unsupervised manner, and then the learned weights were used to initialize the weights of the deep networks. These networks were then fine-tuned using stochastic gradient descent.

The number of pretraining epochs was 100, and the number of fine-tuning epochs was 50. The learning rate was 0.01 for the unsupervised learning, and 0.1 for the supervised learning.

The experimental results on the MNIST dataset are given in Table 3.12. It can be seen from the table that we tested different architectures, and the results were consistent: complex-valued neural networks pretrained using complex-valued deep belief networks attained better classification results than real-valued networks using their real-valued counterparts.

Table 3.12: Experimental results for MNIST

Hidden layer sizes	**Real-valued error**	**Complex-valued error**
1000	1.42%	$\mathbf{1.40\%}$
$1000, 500$	1.36%	$\mathbf{1.32\%}$
$1000, 1000$	1.38%	$\mathbf{1.23\%}$
$1000, 1000, 1000$	1.31%	$\mathbf{1.20\%}$
$1000, 1000, 1000, 1000$	1.38%	$\mathbf{1.37\%}$
$2000, 1500, 1000, 500$	1.64%	$\mathbf{1.34\%}$
$2500, 2000, 1500, 1000, 500$	1.45%	$\mathbf{1.31\%}$

3.6 Complex-valued deep Boltzmann machines

Deep Boltzmann machines (DBMs) were among the first models pertaining to the deep neural networks paradigm. Unlike deep belief networks (DBNs), which were introduced earlier [76], DBMs are fully undirected deep generative models, and were proposed in [183]. DBMs have been applied to many tasks which require learning of features from very few examples, including document modeling [201].

The DBM model was later improved. A new approximate inference algorithm for DBMs was proposed in [184]. To improve the conditioning of the underlying optimization problem, in [123], the centering trick was introduced, which consists of rewriting the energy of the DBMs as a function of centered states. Multimodal learning with DBMs is discussed in [200]. Hierarchical deep models based on DBMs were developed in [186]. In order to improve classification done using DBM pretraining, multi-prediction DBMs are analyzed in [65], and are shown to outperform standard DBMs in this task.

Following the path laid by these papers, and also taking into account the success of complex-valued convolutional neural networks for real-valued image classification (Section 3.1), we consider a promising idea to introduce complex-valued deep Boltzmann machines. The presentation in this section follows the author's paper [171].

3.6.1 Model formulation

We begin by presenting a complex-valued variant of the Boltzmann Machines (BMs), following their real-valued presentation in [183, 186]. Boltzmann Machines are part of the energy-based models family, which associate a scalar energy to any configuration of the variables of interest. In these models, learning consists of modifying the energy function so that it has some desired properties, usually the property to be as low as possible.

To increase the representational power of BMs, some of its variables are considered not to be observed, which is the reason why they are called hidden variables. In this paper, we assume that all the variables are binary, and we denote the visible units by $\boldsymbol{v} \in \{0,1\}^D + \imath\{0,1\}^D$, and the hidden units by $\boldsymbol{h} \in \{0,1\}^P + \imath\{0,1\}^P$. The energy function for the state $\{\boldsymbol{v}, \boldsymbol{h}\}$ is given by

$$\begin{aligned}
\mathcal{E}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) &= -\frac{1}{2}(\boldsymbol{v}^H \boldsymbol{L}\boldsymbol{v})^R - \frac{1}{2}(\boldsymbol{h}^H \boldsymbol{J}\boldsymbol{h})^R - (\boldsymbol{v}^H \boldsymbol{W}\boldsymbol{h})^R \\
&= -\frac{1}{2}(\boldsymbol{v}^R)^T(\boldsymbol{L}\boldsymbol{v})^R - \frac{1}{2}(\boldsymbol{v}^I)^T(\boldsymbol{L}\boldsymbol{v})^I - \frac{1}{2}(\boldsymbol{h}^R)^T(\boldsymbol{J}\boldsymbol{h})^R - \frac{1}{2}(\boldsymbol{h}^I)^T(\boldsymbol{J}\boldsymbol{h})^I \\
&\quad -(\boldsymbol{v}^R)^T(\boldsymbol{W}\boldsymbol{h})^R - (\boldsymbol{v}^I)^T(\boldsymbol{W}\boldsymbol{h})^I \\
&= -\frac{1}{2}(\boldsymbol{v}^R)^T(\boldsymbol{L}\boldsymbol{v})^R - \frac{1}{2}(\boldsymbol{v}^I)^T(\boldsymbol{L}\boldsymbol{v})^I - \frac{1}{2}(\boldsymbol{h}^R)^T(\boldsymbol{J}\boldsymbol{h})^R - \frac{1}{2}(\boldsymbol{h}^I)^T(\boldsymbol{J}\boldsymbol{h})^I \\
&\quad -(\boldsymbol{h}^R)^T(\boldsymbol{W}^H\boldsymbol{v})^R - (\boldsymbol{h}^I)^T(\boldsymbol{W}^H\boldsymbol{v})^I, \qquad (3.6.1)
\end{aligned}$$

where $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{L}, \boldsymbol{J}\}$ represent the parameters of the model: $\boldsymbol{W}$ is the visible-hidden weight matrix, $\boldsymbol{L}$ is the visible-visible weight matrix, and $\boldsymbol{J}$ is the hidden-hidden weight matrix. The diagonal elements of $\boldsymbol{L}$ and $\boldsymbol{J}$ are 0. For the clarity of the presentation, we have omitted the bias terms in the energy function, because biases can be considered as weights of a connection to a unit whose state is always $1 + 1\imath$. We also denoted by $\boldsymbol{z}^R$ and $\boldsymbol{z}^I$ the real and imaginary parts, respectively, of the complex-valued matrix or vector $\boldsymbol{z}$, by $\imath$, with $\imath^2 = -1$, the complex imaginary unit, by $\boldsymbol{z}^T$ the transpose of real-valued matrix or vector $\boldsymbol{z}$, and by $\boldsymbol{z}^H$ the Hermitian

(complex-conjugate) transpose of complex-valued matrix or vector $\boldsymbol{z}$. In the deduction of the last equality in (3.6.1), we used the property $(x^H y)^R = (y^H x)^R$.

The probability the above-defined complex-valued BM assigns to the visible vector $\boldsymbol{v}$ is

$$p(\boldsymbol{v};\boldsymbol{\theta}) = \frac{p^*(\boldsymbol{v};\boldsymbol{\theta})}{Z(\boldsymbol{\theta})} = \frac{1}{Z(\boldsymbol{\theta})}\sum_{\boldsymbol{h}} \exp(-\mathcal{E}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})), \tag{3.6.2}$$

where $p^*(\boldsymbol{v};\boldsymbol{\theta})$ is the unnormalized probability, and

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} \exp(-\mathcal{E}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})),$$

represents the partition function.

The expressions for the conditional probabilities $p(h_j^R|\boldsymbol{v},\boldsymbol{h}_{-j})$ and $p(h_j^I|\boldsymbol{v},\boldsymbol{h}_{-j})$ in a BM with arbitrary units can be obtained as

$$\begin{aligned} p(h_j^R|\boldsymbol{v},\boldsymbol{h}_{-j}) &= \frac{\exp(h_j^R(\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^R + h_j^R(\boldsymbol{W}_j^H\boldsymbol{v})^R)}{\sum_{\widetilde{h}_j^R}\exp(\widetilde{h}_j^R(\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^R + \widetilde{h}_j^R(\boldsymbol{W}_j^H\boldsymbol{v})^R)}, \\ p(h_j^I|\boldsymbol{v},\boldsymbol{h}_{-j}) &= \frac{\exp(h_j^I(\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^I + h_j^I(\boldsymbol{W}_j^H\boldsymbol{v})^I)}{\sum_{\widetilde{h}_j^I}\exp(\widetilde{h}_j^I(\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^I + \widetilde{h}_j^I(\boldsymbol{W}_j^H\boldsymbol{v})^I)}, \end{aligned}$$

where $\boldsymbol{z}_{-j}$ represents the vector $\boldsymbol{z}$ without element z_j, and $\boldsymbol{Z}_{-j}$ represents matrix $\boldsymbol{Z}$ without column j, which is denoted by $\boldsymbol{Z}_j$. But, because we assumed that all the units are binary, we have that

$$\begin{aligned} p(h_j^R=1|\boldsymbol{v},\boldsymbol{h}_{-j}) &= \frac{\exp((\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^R + (\boldsymbol{W}_j^H\boldsymbol{v})^R)}{1+\exp((\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^R + (\boldsymbol{W}_j^H\boldsymbol{v})^R)}, \\ &= \sigma((\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^R + (\boldsymbol{W}_j^H\boldsymbol{v})^R) \\ &= \sigma\left((\boldsymbol{J}_{-j}^R)^T\boldsymbol{h}_{-j}^R + (\boldsymbol{J}_{-j}^I)^T\boldsymbol{h}_{-j}^I + (\boldsymbol{W}_j^R)^T\boldsymbol{v}^R + (\boldsymbol{W}_j^I)^T\boldsymbol{v}^I\right), \end{aligned}$$

$$\begin{aligned} p(h_j^I=1|\boldsymbol{v},\boldsymbol{h}_{-j}) &= \frac{\exp((\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^I + (\boldsymbol{W}_j^H\boldsymbol{v})^I)}{1+\exp((\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^I + (\boldsymbol{W}_j^H\boldsymbol{v})^I)}, \\ &= \sigma((\boldsymbol{J}_{-j}^H\boldsymbol{h}_{-j})^I + (\boldsymbol{W}_j^H\boldsymbol{v})^I) \\ &= \sigma\left((\boldsymbol{J}_{-j}^R)^T\boldsymbol{h}_{-j}^I - (\boldsymbol{J}_{-j}^I)^T\boldsymbol{h}_{-j}^R + (\boldsymbol{W}_j^R)^T\boldsymbol{v}^I - (\boldsymbol{W}_j^I)^T\boldsymbol{v}^R\right), \end{aligned}$$

where σ is the real-valued sigmoid function: $\sigma(x) = \frac{1}{1+\exp(-x)}$. The expression of the energy function (3.6.1) can be written as

$$\begin{aligned} \mathcal{E}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) &= -\frac{1}{2}(\boldsymbol{v}^R)^T\boldsymbol{L}^R\boldsymbol{v}^R - \frac{1}{2}(\boldsymbol{v}^R)^T\boldsymbol{L}^I\boldsymbol{v}^I - \frac{1}{2}(\boldsymbol{v}^I)^T\boldsymbol{L}^R\boldsymbol{v}^I + \frac{1}{2}(\boldsymbol{v}^I)^T\boldsymbol{L}^I\boldsymbol{v}^R \\ &\quad -\frac{1}{2}(\boldsymbol{h}^R)^T\boldsymbol{J}^R\boldsymbol{h}^R - \frac{1}{2}(\boldsymbol{h}^R)^T\boldsymbol{J}^I\boldsymbol{h}^I - \frac{1}{2}(\boldsymbol{h}^I)^T\boldsymbol{J}^R\boldsymbol{h}^I + \frac{1}{2}(\boldsymbol{h}^I)^T\boldsymbol{J}^I\boldsymbol{h}^R \\ &\quad -(\boldsymbol{v}^R)^T\boldsymbol{W}^R\boldsymbol{h}^R - (\boldsymbol{v}^R)^T\boldsymbol{W}^I\boldsymbol{h}^I - (\boldsymbol{v}^I)^T\boldsymbol{W}^R\boldsymbol{h}^I + (\boldsymbol{v}^I)^T\boldsymbol{W}^I\boldsymbol{h}^R. \end{aligned}$$

This, along with the expressions for $p(h_j^R=1|\boldsymbol{v},\boldsymbol{h}_{-j})$ and $p(h_j^I=1|\boldsymbol{v},\boldsymbol{h}_{-j})$, deduced above, show that a complex-valued BM can be implemented only using real-valued operations, which is a desirable property since the computational frameworks that are popular in the deep learning domain mainly deal with this type of operations.

Due to the symmetry in the expression of the energy function between the visible and hidden units, the following relations can also be deduced:

$$\begin{aligned} p(v_i^R = 1|\boldsymbol{h}, \boldsymbol{v}_{-i}) &= \sigma\left((\boldsymbol{L}_{-i}^R)^T \boldsymbol{v}_{-i}^R + (\boldsymbol{L}_{-i}^I)^T \boldsymbol{v}_{-i}^I + \boldsymbol{W}_i^R \boldsymbol{h}^R - \boldsymbol{W}_i^I \boldsymbol{h}^I\right), \\ p(v_i^I = 1|\boldsymbol{h}, \boldsymbol{v}_{-i}) &= \sigma\left((\boldsymbol{L}_{-i}^R)^T \boldsymbol{v}_{-i}^I - (\boldsymbol{L}_{-i}^I)^T \boldsymbol{v}_{-i}^R + \boldsymbol{W}_i^R \boldsymbol{h}^I + \boldsymbol{W}_i^I \boldsymbol{h}^R\right). \end{aligned}$$

The derivatives of the log-likelihood with respect to the model parameters $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{L}, \boldsymbol{J}\}$ can be obtained as:

$$\begin{aligned} \frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{W}^R} &= E_{P_{\text{data}}}[\boldsymbol{v}^R(\boldsymbol{h}^R)^T + \boldsymbol{v}^I(\boldsymbol{h}^I)^T] - E_{P_{\text{model}}}[\boldsymbol{v}^R(\boldsymbol{h}^R)^T + \boldsymbol{v}^I(\boldsymbol{h}^I)^T], \\ \frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{W}^I} &= E_{P_{\text{data}}}[\boldsymbol{v}^R(\boldsymbol{h}^I)^T - \boldsymbol{v}^I(\boldsymbol{h}^R)^T] - E_{P_{\text{model}}}[\boldsymbol{v}^R(\boldsymbol{h}^I)^T - \boldsymbol{v}^I(\boldsymbol{h}^R)^T], \\ \frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{L}^R} &= E_{P_{\text{data}}}[\boldsymbol{v}^R(\boldsymbol{v}^R)^T + \boldsymbol{v}^I(\boldsymbol{v}^I)^T] - E_{P_{\text{model}}}[\boldsymbol{v}^R(\boldsymbol{v}^R)^T + \boldsymbol{v}^I(\boldsymbol{v}^I)^T], \\ \frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{L}^I} &= E_{P_{\text{data}}}[\boldsymbol{v}^R(\boldsymbol{v}^I)^T - \boldsymbol{v}^I(\boldsymbol{v}^R)^T] - E_{P_{\text{model}}}[\boldsymbol{v}^R(\boldsymbol{v}^I)^T - \boldsymbol{v}^I(\boldsymbol{v}^R)^T], \\ \frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{J}^R} &= E_{P_{\text{data}}}[\boldsymbol{h}^R(\boldsymbol{h}^R)^T + \boldsymbol{h}^I(\boldsymbol{h}^I)^T] - E_{P_{\text{model}}}[\boldsymbol{h}^R(\boldsymbol{h}^R)^T + \boldsymbol{h}^I(\boldsymbol{h}^I)^T], \\ \frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{J}^I} &= E_{P_{\text{data}}}[\boldsymbol{h}^R(\boldsymbol{h}^I)^T - \boldsymbol{h}^I(\boldsymbol{h}^R)^T] - E_{P_{\text{model}}}[\boldsymbol{h}^R(\boldsymbol{h}^I)^T - \boldsymbol{h}^I(\boldsymbol{h}^R)^T], \end{aligned}$$

where $E_{P_{\text{data}}}[\cdot]$ represents the expectation of the completed data distribution

$$P_{\text{data}}(\boldsymbol{h}, \boldsymbol{v}; \boldsymbol{\theta}) = p(\boldsymbol{h}|\boldsymbol{v}; \boldsymbol{\theta}) P_{\text{data}}(\boldsymbol{v}),$$

with $P_{\text{data}}(\boldsymbol{v}) = \frac{1}{N}\sum_n \delta(\boldsymbol{v} - \boldsymbol{v}_n)$ being the empirical distribution, and $E_{P_{\text{model}}}[\cdot]$ represents the expectation of the distribution defined by the model in (3.6.2). We can see that $E_{P_{\text{data}}}[\cdot]$ is a data-dependent expectation, and $E_{P_{\text{model}}}[\cdot]$ is the model expectation.

Exact maximum likelihood learning in this model is intractable, because exact computation of the data-dependent expectations is exponential in the number of hidden units, and the exact computation of the model expectations is exponential in the number of visible and hidden units.

Therefore, approximate learning procedures are needed. Mean-field inference will be used to estimate data-dependent expectations and stochastic approximation procedure based on Markov Chain Monte Carlo (MCMC) will be used to estimate the model expectations.

In variational learning, the true posterior distribution $p(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta})$ is replaced by an approximation $q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu})$ of it, and the parameter updates follow the gradient of the lower bound on the log-likelihood:

$$\begin{aligned} \log p(\boldsymbol{v};\boldsymbol{\theta}) &\geq \sum_{\boldsymbol{h}} q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu}) \log p(\boldsymbol{v}, \boldsymbol{h};\boldsymbol{\theta}) + \mathcal{H}(q) \\ &\geq \log p(\boldsymbol{v};\boldsymbol{\theta}) - KL[q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu})||p(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta})], \end{aligned}$$

where $\mathcal{H}(q)$ represents the entropy functional and $KL[q||p]$ the Kullback-Leibler divergence of the two distributions. In addition to maximizing the log-likelihood of the data, variational learning has the property to also minimize the Kullback-Leibler divergence between the true posterior distribution and its approximation.

Following the mean-field approach, we assume that $q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu})$ is a fully factorized distribution:

$$q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu}) = \prod_j q(h_j^R|\boldsymbol{v}) \prod_j q(h_j^I|\boldsymbol{v}),$$

where $q(h_j^R = 1) = \mu_j^R$, $q(h_j^I = 1) = \mu_j^I$ are the mean-field parameters. The lower bound of the log-probability of the data becomes:

$$\begin{aligned}
\log p(\boldsymbol{v};\boldsymbol{\theta}) \;\geq\; & -\frac{1}{2}(\boldsymbol{v}^R)^T\boldsymbol{L}^R\boldsymbol{v}^R - \frac{1}{2}(\boldsymbol{v}^R)^T\boldsymbol{L}^I\boldsymbol{v}^I - \frac{1}{2}(\boldsymbol{v}^I)^T\boldsymbol{L}^R\boldsymbol{v}^I + \frac{1}{2}(\boldsymbol{v}^I)^T\boldsymbol{L}^I\boldsymbol{v}^R \\
& -\frac{1}{2}(\boldsymbol{\mu}^R)^T\boldsymbol{J}^R\boldsymbol{\mu}^R - \frac{1}{2}(\boldsymbol{\mu}^R)^T\boldsymbol{J}^I\boldsymbol{\mu}^I - \frac{1}{2}(\boldsymbol{\mu}^I)^T\boldsymbol{J}^R\boldsymbol{\mu}^I + \frac{1}{2}(\boldsymbol{\mu}^I)^T\boldsymbol{J}^I\boldsymbol{\mu}^R \\
& -(\boldsymbol{v}^R)^T\boldsymbol{W}^R\boldsymbol{\mu}^R - (\boldsymbol{v}^R)^T\boldsymbol{W}^I\boldsymbol{\mu}^I - (\boldsymbol{v}^I)^T\boldsymbol{W}^R\boldsymbol{\mu}^I + (\boldsymbol{v}^I)^T\boldsymbol{W}^I\boldsymbol{\mu}^R \\
& -\log Z(\boldsymbol{\theta}) \\
& +\sum_j \left[\mu_j^R \log \mu_j^R + (1-\mu_j^R)\log(1-\mu_j^R)\right] \\
& +\sum_j \left[\mu_j^I \log \mu_j^I + (1-\mu_j^I)\log(1-\mu_j^I)\right].
\end{aligned}$$

During learning, this lower bound is maximized with respect to the parameters $\boldsymbol{\mu}$ for fixed $\boldsymbol{\theta}$, which results in the mean-field fixed-point equations:

$$\begin{aligned}
\mu_j^R &\leftarrow \sigma\left((\boldsymbol{J}_{-j}^R)^T\boldsymbol{\mu}_{-j}^R + (\boldsymbol{J}_{-j}^I)^T\boldsymbol{\mu}_{-j}^I + (\boldsymbol{W}_j^R)^T\boldsymbol{v}^R + (\boldsymbol{W}_j^I)^T\boldsymbol{v}^I\right), \\
\mu_j^I &\leftarrow \sigma\left((\boldsymbol{J}_{-j}^R)^T\boldsymbol{\mu}_{-j}^I - (\boldsymbol{J}_{-j}^I)^T\boldsymbol{\mu}_{-j}^R + (\boldsymbol{W}_j^R)^T\boldsymbol{v}^I - (\boldsymbol{W}_j^I)^T\boldsymbol{v}^R\right).
\end{aligned}$$

Now, having the variational parameters $\boldsymbol{\mu}$, the model parameters $\boldsymbol{\theta}$ are updated to maximize the variational bound using stochastic approximation based on MCMC [207], which proceeds as follows. Let $\boldsymbol{\theta}_t$ be the current parameters and $\boldsymbol{x}_t = \{\boldsymbol{v}_t, \boldsymbol{h}_t\}$ the current state. Then $\boldsymbol{\theta}_t$ and $\boldsymbol{x}_t$ are updated as follows:

- Given $\boldsymbol{x}_t$, a new state $\boldsymbol{x}_{t+1}$ is sampled from the transition operator $T_{\boldsymbol{\theta}_t}(\boldsymbol{x}_{t+1};\boldsymbol{x}_t)$ so that $p_{\boldsymbol{\theta}_t}$ remains invariant.
- The new parameters $\boldsymbol{\theta}_{t+1}$ are obtained by replacing the model expectation, which is intractable, by the expectation with respect to $\boldsymbol{x}_{t+1}$, in the gradient step.

An average over a set of M persistent sample particles $\{\boldsymbol{x}_{t,1}, \ldots, \boldsymbol{x}_{t,M}\}$ is typically used in practice. Precise sufficient conditions for the almost sure convergence to an asymptotically stable point can be derived. One such condition is that the learning rate should decrease with time, such that $\sum_{t=0}^{\infty}\alpha_t = \infty$ and $\sum_{t=0}^{\infty}\alpha_t^2 < \infty$. One way to achieve this is to put $\alpha_t = 1/t$. On the other hand, in practice, the sequence $|\boldsymbol{\theta}_t|$ is usually bounded, and the Markov chain having the transition operator $T_{\boldsymbol{\theta}_t}(\boldsymbol{x}_{t+1};\boldsymbol{x}_t)$ is ergodic. These two conditions are sufficient to guarantee almost sure convergence.

A complex-valued Deep Boltzmann Machine (DBM) is similar to a complex-valued Boltzmann Machine, with the exception that the model contains a sequence of layers of hidden units denoted by $\boldsymbol{h}^{(1)} \in \{0,1\}^{P^{(1)}} + \imath\{0,1\}^{P^{(1)}}$, $\boldsymbol{h}^{(2)} \in \{0,1\}^{P^{(2)}} + \imath\{0,1\}^{P^{(2)}}$, …, $\boldsymbol{h}^{(L)} \in \{0,1\}^{P^{(L)}} + \imath\{0,1\}^{P^{(L)}}$, and connections exist only between hidden units in adjacent layers and between the visible units and the hidden units in the first hidden layer, i.e., there are no visible-visible or intralayer hidden-hidden connections. For clarity of presentation, we consider a DBM with three hidden layers ($L = 3$) and no biases, which can be considered as weights of a connection to a unit whose state is always $1 + 1\imath$. The deduction of the learning procedure for complex-valued DBMs is very similar to the one for complex-valued BMs, and to the one for their real-valued counterparts, given in [183, 186].

The energy function for such a network is given by:

$$
\begin{aligned}
\mathcal{E}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) &= -(\boldsymbol{v}^H\boldsymbol{W}^{(1)}\boldsymbol{h}^{(1)})^R - (\boldsymbol{h}^{(1)H}\boldsymbol{W}^{(2)}\boldsymbol{h}^{(2)})^R - (\boldsymbol{h}^{(2)H}\boldsymbol{W}^{(3)}\boldsymbol{h}^{(3)})^R \\
&= -(\boldsymbol{v}^R)^T(\boldsymbol{W}^{(1)}\boldsymbol{h}^{(1)})^R - (\boldsymbol{v}^I)^T(\boldsymbol{W}^{(1)}\boldsymbol{h}^{(1)})^I - (\boldsymbol{h}^{(1)R})^T(\boldsymbol{W}^{(2)}\boldsymbol{h}^{(2)})^R \\
&\quad -(\boldsymbol{h}^{(1)I})^T(\boldsymbol{W}^{(2)}\boldsymbol{h}^{(2)})^I - (\boldsymbol{h}^{(2)R})^T(\boldsymbol{W}^{(3)}\boldsymbol{h}^{(3)})^R - (\boldsymbol{h}^{(2)I})^T(\boldsymbol{W}^{(3)}\boldsymbol{h}^{(3)})^I \\
&= -(\boldsymbol{v}^R)^T\boldsymbol{W}^{(1)R}\boldsymbol{h}^{(1)R} - (\boldsymbol{v}^R)^T\boldsymbol{W}^{(1)I}\boldsymbol{h}^{(1)I} - (\boldsymbol{v}^I)^T\boldsymbol{W}^{(1)R}\boldsymbol{h}^{(1)I} \\
&\quad +(\boldsymbol{v}^I)^T\boldsymbol{W}^{(1)I}\boldsymbol{h}^{(1)R} - (\boldsymbol{h}^{(1)R})^T\boldsymbol{W}^{(2)R}\boldsymbol{h}^{(2)R} - (\boldsymbol{h}^{(1)R})^T\boldsymbol{W}^{(2)I}\boldsymbol{h}^{(2)I} \\
&\quad -(\boldsymbol{h}^{(1)I})^T\boldsymbol{W}^{(2)R}\boldsymbol{h}^{(2)I} + (\boldsymbol{h}^{(1)I})^T\boldsymbol{W}^{(2)I}\boldsymbol{h}^{(2)R} - (\boldsymbol{h}^{(2)R})^T\boldsymbol{W}^{(3)R}\boldsymbol{h}^{(3)R} \\
&\quad -(\boldsymbol{h}^{(2)R})^T\boldsymbol{W}^{(3)I}\boldsymbol{h}^{(3)I} - (\boldsymbol{h}^{(2)I})^T\boldsymbol{W}^{(3)R}\boldsymbol{h}^{(3)I} + (\boldsymbol{h}^{(2)I})^T\boldsymbol{W}^{(3)I}\boldsymbol{h}^{(3)R}, \quad (3.6.3)
\end{aligned}
$$

where $\boldsymbol{h} = \{\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \boldsymbol{h}^{(3)}\}$ is the set of hidden units and $\boldsymbol{\theta} = \{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)}\}$ are the parameters of the model.

The probability the above-defined complex-valued DBM assigns to the visible vector $\boldsymbol{v}$ is

$$
p(\boldsymbol{v};\boldsymbol{\theta}) = \frac{p^*(\boldsymbol{v};\boldsymbol{\theta})}{Z(\boldsymbol{\theta})} = \frac{1}{Z(\boldsymbol{\theta})}\sum_{\boldsymbol{h}}\exp(-\mathcal{E}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})), \quad (3.6.4)
$$

where $p^*(\boldsymbol{v};\boldsymbol{\theta})$ is the unnormalized probability, and

$$
Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}}\exp(-\mathcal{E}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta})),
$$

represents the partition function.

The expressions for the conditional probabilities in a complex-valued DBM are:

$$
p(h_j^{(1)R} = 1|\boldsymbol{v},\boldsymbol{h}^{(2)}) = \sigma\left((\boldsymbol{W}_j^{(1)R})^T\boldsymbol{v}^R + (\boldsymbol{W}_j^{(1)I})^T\boldsymbol{v}^I + \boldsymbol{W}_j^{(2)R}\boldsymbol{h}^{(2)R} + \boldsymbol{W}_j^{(2)I}\boldsymbol{h}^{(2)I}\right),
$$

$$
p(h_j^{(1)I} = 1|\boldsymbol{v},\boldsymbol{h}^{(2)}) = \sigma\left((\boldsymbol{W}_j^{(1)R})^T\boldsymbol{v}^I - (\boldsymbol{W}_j^{(1)I})^T\boldsymbol{v}^R + \boldsymbol{W}_j^{(2)R}\boldsymbol{h}^{(2)I} - \boldsymbol{W}_j^{(2)I}\boldsymbol{h}^{(2)R}\right),
$$

$$
p(h_l^{(2)R} = 1|\boldsymbol{h}^{(1)},\boldsymbol{h}^{(3)}) = \sigma\left((\boldsymbol{W}_l^{(2)R})^T\boldsymbol{h}^{(1)R} + (\boldsymbol{W}_l^{(2)I})^T\boldsymbol{h}^{(1)I} + \boldsymbol{W}_l^{(3)R}\boldsymbol{h}^{(3)R} + \boldsymbol{W}_l^{(3)I}\boldsymbol{h}^{(3)I}\right),
$$

$$
p(h_l^{(2)I} = 1|\boldsymbol{h}^{(1)},\boldsymbol{h}^{(3)}) = \sigma\left((\boldsymbol{W}_l^{(2)R})^T\boldsymbol{h}^{(1)I} - (\boldsymbol{W}_l^{(2)I})^T\boldsymbol{h}^{(1)R} + \boldsymbol{W}_l^{(3)R}\boldsymbol{h}^{(3)I} - \boldsymbol{W}_l^{(3)I}\boldsymbol{h}^{(3)R}\right),
$$

$$
p(h_k^{(3)R} = 1|\boldsymbol{h}^{(2)}) = \sigma\left((\boldsymbol{W}_k^{(3)R})^T\boldsymbol{h}^{(2)R} + (\boldsymbol{W}_k^{(3)I})^T\boldsymbol{h}^{(2)I}\right),
$$

$$
p(h_k^{(3)I} = 1|\boldsymbol{h}^{(2)}) = \sigma\left((\boldsymbol{W}_k^{(3)R})^T\boldsymbol{h}^{(2)I} - (\boldsymbol{W}_k^{(3)I})^T\boldsymbol{h}^{(2)R}\right),
$$

$$
p(v_i^R = 1|\boldsymbol{h}^{(1)}) = \sigma\left(\boldsymbol{W}_i^{(1)R}\boldsymbol{h}^{(1)R} + \boldsymbol{W}_i^{(1)I}\boldsymbol{h}^{(1)I}\right),
$$

$$
p(v_i^I = 1|\boldsymbol{h}^{(1)}) = \sigma\left(\boldsymbol{W}_i^{(1)R}\boldsymbol{h}^{(1)I} - (\boldsymbol{W}_i^{(1)I})^T\boldsymbol{h}^{(1)R}\right).
$$

The derivatives of the log-likelihood with respect to the model parameters $\boldsymbol{\theta} = \{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)}\}$ can be obtained as:

$$
\begin{aligned}
\frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{W}^{(1)R}} &= E_{P_{\text{data}}}[\boldsymbol{v}^R(\boldsymbol{h}^{(1)R})^T + \boldsymbol{v}^I(\boldsymbol{h}^{(1)I})^T] - E_{P_{\text{model}}}[\boldsymbol{v}^R(\boldsymbol{h}^{(1)R})^T + \boldsymbol{v}^I(\boldsymbol{h}^{(1)I})^T],\\
\frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{W}^{(1)I}} &= E_{P_{\text{data}}}[\boldsymbol{v}^R(\boldsymbol{h}^{(1)I})^T - \boldsymbol{v}^I(\boldsymbol{h}^{(1)R})^T] - E_{P_{\text{model}}}[\boldsymbol{v}^R(\boldsymbol{h}^{(1)I})^T - \boldsymbol{v}^I(\boldsymbol{h}^{(1)R})^T],\\
\frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{W}^{(2)R}} &= E_{P_{\text{data}}}[\boldsymbol{h}^{(1)R}(\boldsymbol{h}^{(2)R})^T + \boldsymbol{h}^{(1)I}(\boldsymbol{h}^{(2)I})^T] - E_{P_{\text{model}}}[\boldsymbol{h}^{(1)R}(\boldsymbol{h}^{(2)R})^T + \boldsymbol{h}^{(1)I}(\boldsymbol{h}^{(2)I})^T],\\
\frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{W}^{(2)I}} &= E_{P_{\text{data}}}[\boldsymbol{h}^{(1)R}(\boldsymbol{h}^{(2)I})^T - \boldsymbol{h}^{(1)I}(\boldsymbol{h}^{(2)R})^T] - E_{P_{\text{model}}}[\boldsymbol{h}^{(1)R}(\boldsymbol{h}^{(2)I})^T - \boldsymbol{h}^{(1)I}(\boldsymbol{h}^{(2)R})^T],\\
\frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{W}^{(3)R}} &= E_{P_{\text{data}}}[\boldsymbol{h}^{(2)R}(\boldsymbol{h}^{(3)R})^T + \boldsymbol{h}^{(2)I}(\boldsymbol{h}^{(3)I})^T] - E_{P_{\text{model}}}[\boldsymbol{h}^{(2)R}(\boldsymbol{h}^{(3)R})^T + \boldsymbol{h}^{(2)I}(\boldsymbol{h}^{(3)I})^T],\\
\frac{\partial \log p(\boldsymbol{v};\boldsymbol{\theta})}{\partial \boldsymbol{W}^{(3)I}} &= E_{P_{\text{data}}}[\boldsymbol{h}^{(2)R}(\boldsymbol{h}^{(3)I})^T - \boldsymbol{h}^{(2)I}(\boldsymbol{h}^{(3)R})^T] - E_{P_{\text{model}}}[\boldsymbol{h}^{(2)R}(\boldsymbol{h}^{(3)I})^T - \boldsymbol{h}^{(2)I}(\boldsymbol{h}^{(3)R})^T],
\end{aligned}
$$

where $E_{P_{\text{data}}}[\cdot]$ represents the expectation of the completed data distribution $P_{\text{data}}(\boldsymbol{h},\boldsymbol{v};\boldsymbol{\theta}) = p(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta})P_{\text{data}}(\boldsymbol{v})$, with $P_{\text{data}}(\boldsymbol{v}) = \frac{1}{N}\sum_n \delta(\boldsymbol{v}-\boldsymbol{v}_n)$ being the empirical distribution, and $E_{P_{\text{model}}}[\cdot]$ represents the expectation of the distribution defined by the model in (3.6.4). We can see that $E_{P_{\text{data}}}[\cdot]$ is a data-dependent expectation, and $E_{P_{\text{model}}}[\cdot]$ is the model expectation.

Like in the case of complex-valued BMs, exact maximum likelihood learning in this model is intractable. Thus, mean-field inference will be used to estimate data-dependent expectations and stochastic approximation procedure based on Markov Chain Monte Carlo (MCMC) will be used to estimate the model expectations.

If $q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu})$ represents the approximation of the true posterior distribution $p(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta})$, the parameter updates follow the gradient of the lower bound on the log-likelihood:

$$
\begin{aligned}
\log p(\boldsymbol{v};\boldsymbol{\theta}) &\geq \sum_{\boldsymbol{h}} q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu}) \log p(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\theta}) + \mathcal{H}(q) \qquad (3.6.5)\\
&\geq \log p(\boldsymbol{v};\boldsymbol{\theta}) - KL[q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu})||p(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\theta})],
\end{aligned}
$$

where the notations are the same as for complex-valued BMs.

Following the mean-field approach, we assume that $q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu})$ is a fully factorized distribution:

$$
q(\boldsymbol{h}|\boldsymbol{v};\boldsymbol{\mu}) = \prod_j q(h_j^{(1)R}|\boldsymbol{v}) \prod_j q(h_j^{(1)I}|\boldsymbol{v}) \prod_l q(h_l^{(2)R}|\boldsymbol{v}) \prod_l q(h_l^{(2)I}|\boldsymbol{v}) \prod_k q(h_k^{(3)R}|\boldsymbol{v}) \prod_k q(h_k^{(3)I}|\boldsymbol{v}),
$$

where $\boldsymbol{\mu} = \{\boldsymbol{\mu}^{(1)},\boldsymbol{\mu}^{(2)},\boldsymbol{\mu}^{(3)}\}$ are the mean-field parameters, with $q(h_i^{(l)R}=1) = \mu_i^{(l)R}$, $q(h_i^{(l)I}=1) = \mu_i^{(l)I}$, $l = 1,2,3$. The lower bound of the log-probability of the data becomes:

$$
\begin{aligned}
\log p(\boldsymbol{v};\boldsymbol{\theta}) \geq\ & -(\boldsymbol{v}^R)^T\boldsymbol{W}^{(1)R}\boldsymbol{\mu}^{(1)R} - (\boldsymbol{v}^R)^T\boldsymbol{W}^{(1)I}\boldsymbol{\mu}^{(1)I} - (\boldsymbol{v}^I)^T\boldsymbol{W}^{(1)R}\boldsymbol{\mu}^{(1)I}\\
&+(\boldsymbol{v}^I)^T\boldsymbol{W}^{(1)I}\boldsymbol{\mu}^{(1)R} - (\boldsymbol{\mu}^{(1)R})^T\boldsymbol{W}^{(2)R}\boldsymbol{\mu}^{(2)R} - (\boldsymbol{\mu}^{(1)R})^T\boldsymbol{W}^{(2)I}\boldsymbol{\mu}^{(2)I}\\
&-(\boldsymbol{\mu}^{(1)I})^T\boldsymbol{W}^{(2)R}\boldsymbol{\mu}^{(2)I} + (\boldsymbol{\mu}^{(1)I})^T\boldsymbol{W}^{(2)I}\boldsymbol{\mu}^{(2)R} - (\boldsymbol{\mu}^{(2)R})^T\boldsymbol{W}^{(3)R}\boldsymbol{\mu}^{(3)R}\\
&-(\boldsymbol{\mu}^{(2)R})^T\boldsymbol{W}^{(3)I}\boldsymbol{\mu}^{(3)I} - (\boldsymbol{\mu}^{(2)I})^T\boldsymbol{W}^{(3)R}\boldsymbol{\mu}^{(3)I} + (\boldsymbol{\mu}^{(2)I})^T\boldsymbol{W}^{(3)I}\boldsymbol{\mu}^{(3)R}\\
&-\log Z(\boldsymbol{\theta}) + \mathcal{H}(q).
\end{aligned}
$$

During learning, this lower bound is maximized with respect to the parameters $\boldsymbol{\mu}$ for fixed $\boldsymbol{\theta}$,

which results in the mean-field fixed-point equations:

$$\begin{aligned}
\mu_j^{(1)R} &\leftarrow \sigma\left((\boldsymbol{W}_j^{(1)R})^T\boldsymbol{v}^R + (\boldsymbol{W}_j^{(1)I})^T\boldsymbol{v}^I + \boldsymbol{W}_j^{(2)R}\boldsymbol{\mu}^{(2)R} + \boldsymbol{W}_j^{(2)I}\boldsymbol{\mu}^{(2)I}\right),\\
\mu_j^{(1)I} &\leftarrow \sigma\left((\boldsymbol{W}_j^{(1)R})^T\boldsymbol{v}^I - (\boldsymbol{W}_j^{(1)I})^T\boldsymbol{v}^R + \boldsymbol{W}_j^{(2)R}\boldsymbol{\mu}^{(2)I} - \boldsymbol{W}_j^{(2)I}\boldsymbol{\mu}^{(2)R}\right),\\
\mu_l^{(2)R} &\leftarrow \sigma\left((\boldsymbol{W}_l^{(2)R})^T\boldsymbol{\mu}^{(1)R} + (\boldsymbol{W}_l^{(2)I})^T\boldsymbol{\mu}^{(1)I} + \boldsymbol{W}_l^{(3)R}\boldsymbol{\mu}^{(3)R} + \boldsymbol{W}_l^{(3)I}\boldsymbol{\mu}^{(3)I}\right),\\
\mu_l^{(2)I} &\leftarrow \sigma\left((\boldsymbol{W}_l^{(2)R})^T\boldsymbol{\mu}^{(1)I} - (\boldsymbol{W}_l^{(2)I})^T\boldsymbol{\mu}^{(1)R} + \boldsymbol{W}_l^{(3)R}\boldsymbol{\mu}^{(3)I} - \boldsymbol{W}_l^{(3)I}\boldsymbol{\mu}^{(3)R}\right),\\
\mu_k^{(3)R} &\leftarrow \sigma\left((\boldsymbol{W}_k^{(3)R})^T\boldsymbol{\mu}^{(2)R} + (\boldsymbol{W}_k^{(3)I})^T\boldsymbol{\mu}^{(2)I}\right),\\
\mu_k^{(3)I} &\leftarrow \sigma\left((\boldsymbol{W}_k^{(3)R})^T\boldsymbol{\mu}^{(2)I} - (\boldsymbol{W}_k^{(3)I})^T\boldsymbol{\mu}^{(2)R}\right).
\end{aligned}$$

Now, having the variational parameters $\boldsymbol{\mu}$, the model parameters $\boldsymbol{\theta}$ are updated to maximize the variational bound using stochastic approximation based on MCMC [207], which proceeds as follows. Let $\boldsymbol{\theta}_t$ be the current parameters and $\boldsymbol{x}_t = \{\boldsymbol{v}_t, \boldsymbol{h}_t^{(1)}, \boldsymbol{h}_t^{(2)}, \boldsymbol{h}_t^{(3)}\}$ the current state. Then $\boldsymbol{\theta}_t$ and $\boldsymbol{x}_t$ are updated as follows:

- Given $\boldsymbol{x}_t$, a new state $\boldsymbol{x}_{t+1}$ is sampled from the transition operator $T_{\boldsymbol{\theta}_t}(\boldsymbol{x}_{t+1}; \boldsymbol{x}_t)$ so that $p_{\boldsymbol{\theta}_t}$ remains invariant.
- The new parameters $\boldsymbol{\theta}_{t+1}$ are obtained by replacing the model expectation, which is intractable, by the expectation with respect to $\boldsymbol{x}_{t+1}$, in the gradient step.

An average over a set of M persistent sample particles $\{\boldsymbol{x}_{t,1}, \ldots, \boldsymbol{x}_{t,M}\}$ is typically used in practice. The sufficient conditions for the almost sure convergence to an asymptotically stable point are the same as for the complex-valued BMs, discussed above.

The learning procedure for complex-valued DBMs described above can be used with random weight initializations, but better performance is attained using greedy layer-wise pretraining [183]. Each layer in a DBM can be seen as a Restricted Boltzmann Machine (RBM), which is a special case of a complex-valued Boltzmann Machine, described above. This means that each layer can be trained in isolation. The first layer is trained to model the input data, and all the other layers to model the outputs of the previous layer. After the RBMs have been trained using this procedure, they can be combined to form a complex-valued DBM.

However, the RBM parameters must be modified before inclusion in the DBM [183]. Specifically, the weights of all but the top and bottom layers must be divided by two, the bottom RBM must be trained with two copies of each visible unit, and the weights should be made equal for the two copies. Also, the top layer must be trained with two copies of the output. This is to account for the fact that an inner layer is trained only with bottom-up input, but in the DBM it will have bottom-up as well as top-down inputs.

In order to evaluate complex-valued DBMs, we must be able to efficiently estimate their partition function. This can be done using a Monte Carlo based method, namely Annealed Importance Sampling (AIS). This procedure for complex-valued DBMs is the same as the one for real-valued DBMs, which is detailed in [185, 183]. Once we have an estimation $\widehat{Z}$ of the global partition function $Z(\boldsymbol{\theta})$, we can estimate, for a test case $\boldsymbol{v}^*$, the variational lower bound in (3.6.5):

$$\begin{aligned}
\log p(\boldsymbol{v}^*; \boldsymbol{\theta}) &\geq -\sum_{\boldsymbol{h}} q(\boldsymbol{h}|\boldsymbol{v}^*; \boldsymbol{\mu})\mathcal{E}(\boldsymbol{v}^*, \boldsymbol{h}; \boldsymbol{\theta}) + \mathcal{H}(q) - \log Z(\boldsymbol{\theta})\\
&\approx -\sum_{\boldsymbol{h}} q(\boldsymbol{h}|\boldsymbol{v}^*; \boldsymbol{\mu})\mathcal{E}(\boldsymbol{v}^*, \boldsymbol{h}; \boldsymbol{\theta}) + \mathcal{H}(q) - \log \widehat{Z}. \qquad (3.6.6)
\end{aligned}$$

For each test vector, the lower bound is maximized with respect to the parameters $\boldsymbol{\mu}$ using the mean-field equations above.

3.6.2 Experimental results

In our experiments, we train real-valued and complex-valued Deep Boltzmann Machines (DBMs), first using greedy layer-wise pretraining for 50 epochs, and then the learning algorithm described above, also for 50 epochs. The 28×28 pixel images are linearized as 784-dimensional vectors to be given to the networks. The number of units in each complex-valued network is 1.41 times smaller than for its real-valued counterpart, to ensure approximately the same number of real parameters in the two networks, and, as such, a fair comparison between them.

The number of persistent Markov chains is $M = 100$ and the maximum number of mean-field updates for each parameter update is $N = 50$. For computing the estimate of the partition function, we use AIS with $200,000$ intermediate distributions.

After training the DBMs, a logistic regression layer which consists of the softmax function and the negative log likelihood loss function is added on top of the last layer in the networks. Learning then continues in a supervised manner, like in a deep neural network.

The Adam [98] algorithm with minibatches of 128 images is used for supervised training. The learning rate is 0.001 and the number of epochs is 50 for fine-tuning.

3.6.2.1 MNIST

The results for training real-valued and complex-valued DBMs on the MNIST dataset are given in Table 3.13. The average log-probability calculated using (3.6.6) is presented for the real and complex network architectures with two and three layers of different sizes. The layer sizes are given for the real-valued networks. Then, for the same networks, the classification error on the test set is given. It can be seen from the table that complex-valued DBMs outperform real-valued DBMs in terms of average log-probability, and the complex-valued networks initialized using DBMs have better classification performance than their real-valued counterparts. The samples generated by the real-valued and complex-valued DBMs after $3,000$ Gibbs steps are given in Figure 3.3.

3.6.2.2 FashionMNIST

The experimental results are given in the same way as with the above experiment in Table 3.14. It can be seen that the task is more complicated, both in terms of average log-probability as well as in terms of classification error. Nonetheless, the conclusion of the set of experiments is the same: complex-valued networks have better average log-probabilities and better classification errors than real-valued networks. Samples generated after $3,000$ Gibbs steps by the real-valued and complex-valued DBMs are given in Figure 3.4.

Figure 3.3: MNIST images generated by the real-valued (left) and complex-valued (right) DBMs, along with training images (center) [171]

Table 3.13: Experimental results for MNIST

Hidden layer sizes	Real Avg. Log-Prob.	Complex Avg. Log-Prob.
$500, 1000$	-86.59	$\mathbf{-86.37}$
$1000, 1000$	-86.31	$\mathbf{-86.16}$
$500, 1000, 1000$	-85.48	$\mathbf{-85.23}$
$1000, 1000, 1000$	-85.16	$\mathbf{-84.95}$

Hidden layer sizes	Real Error	Complex Error
$500, 1000$	1.36%	$\mathbf{1.23\%}$
$1000, 1000$	1.38%	$\mathbf{1.32\%}$
$500, 1000, 1000$	1.35%	$\mathbf{1.21\%}$
$1000, 1000, 1000$	1.31%	$\mathbf{1.20\%}$

Figure 3.4: FashionMNIST images generated by the real-valued (left) and complex-valued (right) DBMs, along with training images (center) [171]

Table 3.14: Experimental results for FashionMNIST

Hidden layer sizes	**Real Avg. Log-Prob.**	**Complex Avg. Log-Prob.**
$500, 1000$	-236.95	$\mathbf{-231.34}$
$1000, 1000$	-228.43	$\mathbf{-225.76}$
$500, 1000, 1000$	-221.76	$\mathbf{-216.85}$
$1000, 1000, 1000$	-216.52	$\mathbf{-213.63}$

Hidden layer sizes	**Real Error**	**Complex Error**
$500, 1000$	10.01%	$\mathbf{9.93\%}$
$1000, 1000$	9.86%	$\mathbf{9.65\%}$
$500, 1000, 1000$	10.13%	$\mathbf{9.96\%}$
$1000, 1000, 1000$	10.04%	$\mathbf{9.87\%}$

Chapter 4

Dynamics of complex-valued neural networks (CVNNs)

Over the last few years, there has been an increasing interest in the domain of recurrent neural networks, especially the following types of models: Hopfield [80, 81], Cohen-Grossberg [44], cellular [42, 43], and bidirectional associative memory neural networks [100], mainly because of their applications in many areas such as classification, optimization, signal and image processing, solving nonlinear algebraic equations, pattern recognition, system identification, associative memories, cryptography, and so on. These applications are highly dependent on the dynamical properties of the networks. Thus, the analysis of the dynamical behavior is an important part in the design of the recurrent neural networks used in applications.

The study of the dynamics of recurrent neural networks has become a field of study in its own right, attracting many researchers. The review paper [238] lists more than 300 references only for the stability analysis of continuous-time recurrent neural networks, not taking into account the discrete-time ones, or other dynamical properties that can be studied such as bifurcation, attractivity, dissipativity, passivity, synchronization, and so on.

On the other hand, complex-valued neural networks have been proposed by Aizenberg [1], but have caught the attention of researchers in the past years, especially due to their applications in physical systems dealing with electromagnetic, ultrasonic, quantum, and light waves, and also in filtering, imaging, optoelectronics, speech synthesis, computer vision, and so on (see, for example, [119, 77]). In these applications, the stability of the complex-valued neural networks plays a very important role. Moreover, they have more complicated properties than the real-valued neural networks because of their complex-valued states, connection weights, and activation functions. The activation functions cannot be a simple generalization of the real-valued ones, because, by Liouville's theorem, it can be deduced that a bounded entire function is a constant, which makes the choice of such functions more difficult. As a consequence, the study of the dynamic behavior of the complex-valued recurrent neural networks has received increasing interest, especially in the last few years.

4.1 μ-Stability of neutral-type impulsive BAM CVNNs with leakage delay and unbounded time-varying delays

Since they were first introduced by Kosko in [100], bidirectional associative memories, an extension of the unidirectional auto-associative Hopfield neural networks, were intensely studied, and have many applications in pattern recognition, signal and image processing, and automatic

control.

Time delays are known to appear in practical implementations of neural networks due to the finite switching speed of amplifiers, and can cause instability or chaotic behavior. Past derivative information is also considered to influence the present state in neutral-type systems. These systems more accurately describe the properties of neural reaction processes that occur in the real world. The existence of neutral-type delays makes the study of these systems more complicated than that of the usual time-delayed models. This type of delays are relevant in automatic control, population dynamics, and vibrating masses attached to an elastic bar. Also, neutral delays may appear when implementing neural networks in VLSI circuits. On the other hand, impulsive effects express instantaneous changes that naturally occur in electronic networks, caused by switching phenomena, frequency changes, or noise. Taking the above analysis into account, the neutral-type impulsive complex-valued BAM neural networks with leakage delay and unbounded time-varying delays will be studied by giving sufficient conditions for the existence and uniqueness of the equilibrium point and for its global μ-stability.

The following notations will be used in this section: $\mathbb{R}$ denotes the set of real numbers, $\mathbb{C}$ the set of complex numbers, $\mathbb{Z}^+$ the set of positive integer numbers, $\mathbb{R}^n$ denotes the n-dimensional Euclidean space, and $\mathbb{C}^n$ the n-dimensional unitary space. Real matrices of dimension $n\times m$ are denoted as $\mathbb{R}^{n\times m}$, and similarly complex matrices of dimension $n \times m$ as $\mathbb{C}^{n\times m}$. A^T denotes the transpose of matrix A, A^* denotes the Hermitian transpose of matrix A, and $\star$ denotes the symmetric or conjugate symmetric terms in a matrix. I_n denotes the identity matrix of dimension n. $\lambda_{\min}(P)$ is defined as the smallest eigenvalue of positive definite matrix P. $|z|$ stands for the norm of the complex number z. $||z||$ stands for the Euclidean norm of the complex vector z. $A > 0$ ($A < 0$) means that A is a positive definite (negative definite) matrix. A^R and A^I, denote, respectively, the real and imaginary parts of complex matrix A.

The presentation in this section follows that in the author's paper [173].

4.1.1 Main results

Consider the complex-valued bidirectional associative memories given by the following system of differential equations

$$\begin{cases} \dot{z}(t) = -D_1 z(t-\delta) + A_1 f_1(w(t)) + B_1 g_1(w(t-\tau(t))) + E_1 \dot{z}(t-\eta(t)) + u_1,\ t \neq t_k,\ t > 0, \\ \dot{w}(t) = -D_2 w(t-\delta) + A_2 f_2(z(t)) + B_2 g_2(z(t-\tau(t))) + E_2 \dot{w}(t-\eta(t)) + u_2,\ t \neq t_k,\ t > 0, \\ \Delta z(t_k) = z(t_k) - z(t_k^-) = J_k^1(z(t_k^-), z_{t_k^-}),\ k \in \mathbb{Z}^+, \\ \Delta w(t_k) = w(t_k) - w(t_k^-) = J_k^2(w(t_k^-), w_{t_k^-}),\ k \in \mathbb{Z}^+, \end{cases} \tag{4.1.1}$$

where the impulse moments t_k satisfy $0 = t_0 < t_1 < \cdots < t_k < \cdots$ and $\lim_{k\to\infty} t_k = +\infty$, $z(t) = [z_1(t), \ldots, z_n(t)]^T \in \mathbb{C}^n$ and $w(t) = [w_1(t), \ldots, w_m(t)]^T \in \mathbb{C}^m$ are the state vectors at time t, $f_1(w(t)) = [f_1^1(w_1(t)), \ldots, f_m^1(w_m(t))]^T \in \mathbb{C}^m$, $g_1(w(t-\tau(t))) = [g_1^1(w_1(t-\tau(t))), \ldots, g_m^1(w_m(t-\tau(t)))]^T \in \mathbb{C}^m$, $f_2(z(t)) = [f_1^2(z_1(t)), \ldots, f_n^2(z_n(t))]^T \in \mathbb{C}^n$, and $g_2(z(t-\tau(t))) = [g_1^2(z_1(t-\tau(t))), \ldots, g_n^2(z_n(t-\tau(t)))]^T \in \mathbb{C}^n$, are the vector activation functions without and with delays, respectively, $D_1 = \text{diag}(d_1^1, d_2^1, \ldots, d_n^1) \in \mathbb{R}^{n\times n}$, $d_i^1 > 0$, $\forall i = 1, 2, \ldots, n$, $D_2 = \text{diag}(d_1^2, d_2^2, \ldots, d_m^2) \in \mathbb{R}^{m\times m}$, $d_j^2 > 0$, $\forall j = 1, 2, \ldots, m$, are the self-feedback connection weight matrices, $A_1 \in \mathbb{C}^{n\times m}$, $A_2 \in \mathbb{C}^{m\times n}$ are the connection weight matrices, $B_1 \in \mathbb{C}^{n\times m}$, $B_2 \in \mathbb{C}^{m\times n}$ are the delayed connection weight matrices, $E_1 = \text{diag}(e_1^1, \ldots, e_n^1) \in \mathbb{C}^{n\times n}$, $E_2 = \text{diag}(e_1^2, \ldots, e_n^2) \in \mathbb{C}^{m\times m}$ are the neutral-type delay connection weight matrices, $u_1 \in \mathbb{C}^n$, $u_2 \in \mathbb{C}^m$ are the external input vectors, and J_k^1, J_k^2 are

the impulsive functions. $\delta > 0$ represents the leakage delay, $\tau(t)$ represents the unbounded time-varying delay, and $\eta(t)$ represents the neutral-type delay.

In order to study the stability of the above-defined network, we need to make a series of assumptions about the activation functions and about the delays.

Assumption 4.1. *The activation functions f_i^1, f_j^2, g_i^1, and g_j^2 satisfy the Lipschitz conditions*

$$|f_i^1(z) - f_i^1(z')| \leq l_i^{f_1}|z - z'|,$$
$$|f_j^2(z) - f_j^2(z')| \leq l_j^{f_2}|z - z'|,$$
$$|g_i^1(z) - g_i^1(z')| \leq l_i^{g_1}|z - z'|,$$
$$|g_j^2(z) - g_j^2(z')| \leq l_j^{g_2}|z - z'|,$$

$\forall z,\ z' \in \mathbb{C}$, where $l_i^{f_1},\ l_j^{f_2},\ l_i^{g_1},\ l_j^{g_2} > 0$ are the respective Lipschitz constants, $\forall i = 1, 2, \ldots, m$, $\forall j = 1, 2, \ldots, n$. Furthermore, we make the notations $L_{f_1} = \operatorname{diag}(l_1^{f_1}, l_2^{f_1}, \ldots, l_m^{f_1})$, $L_{f_2} = \operatorname{diag}(l_1^{f_2}, l_2^{f_2}, \ldots, l_n^{f_2})$, $L_{g_1} = \operatorname{diag}(l_1^{g_1}, l_2^{g_1}, \ldots, l_m^{g_1})$, $L_{g_2} = \operatorname{diag}(l_1^{g_2}, l_2^{g_2}, \ldots, l_n^{g_2})$.

Assumption 4.2. *The time-varying delays $\tau : \mathbb{R} \to \mathbb{R}$ and the neutral-type delays $\eta : \mathbb{R} \to \mathbb{R}$ are differentiable, and satisfy the conditions $\dot{\tau}(t) \leq \tau_d < 1$, $\eta(t) \leq \eta$, $\dot{\eta}(t) \leq \eta_d < 1$, $\forall t > 0$, where τ_d, η, and η_d are positive real constants.*

Assumption 4.3. *For the impulsive functions J_k^1 and J_k^2, there exist $F_k^1 \in \mathbb{C}^{n\times n}$, $F_k^2 \in \mathbb{C}^{m\times m}$, such that*

$$\begin{aligned} J_k^1(z(t_k^-), z_{t_k^-}) &= F_k^1\left(z(t_k^-) - D_1\int_{t_k-\delta}^{t_k} z(s)ds\right), \\ J_k^2(w(t_k^-), w_{t_k^-}) &= F_k^2\left(w(t_k^-) - D_2\int_{t_k-\delta}^{t_k} w(s)ds\right), \end{aligned}$$

for $\forall k \in \mathbb{Z}^+$.

Also, we will need the following definitions and lemmas:

Definition 4.1. ([34]). If $\begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}$ is an equilibrium point of system (4.1.1), $\mu(t)$ is a positive continuous function that satisfies $\mu(t) \to \infty$ as $t \to \infty$, and there exists a positive constant M such that

$$\left\|\begin{bmatrix}z(t)\\ w(t)\end{bmatrix} - \begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}\right\| \leq \frac{M}{\mu(t)},\ \forall t \geq 0,$$

then the equilibrium point $\begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}$ is said to be μ-stable.

Definition 4.2. If $\begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}$ is an equilibrium point of system (4.1.1), and there exist positive constants M and ε such that

$$\left\|\begin{bmatrix}z(t)\\ w(t)\end{bmatrix} - \begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}\right\| \leq \frac{M}{e^{\varepsilon t}},\ \forall t \geq 0,$$

then the equilibrium point $\begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}$ is said to be exponentially stable.

Definition 4.3. If $\begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}$ is an equilibrium point of system (4.1.1), and there exist positive constants M and ε such that

$$\left\| \begin{bmatrix} z(t)\\ w(t)\end{bmatrix} - \begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix} \right\| \leq \frac{M}{t^{\varepsilon}},\ \forall t \geq 0,$$

then the equilibrium point $\begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}$ is said to be power stable.

Definition 4.4. If $\begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}$ is an equilibrium point of system (4.1.1), and there exist positive constants M and ε such that

$$\left\| \begin{bmatrix} z(t)\\ w(t)\end{bmatrix} - \begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix} \right\| \leq \frac{M}{\ln(\varepsilon t+1)},\ \forall t \geq 0,$$

then the equilibrium point $\begin{bmatrix}\hat{z}\\ \hat{w}\end{bmatrix}$ is said to be log-stable.

Lemma 4.1. *([40]). If $H(z,w) : \mathbb{C}^{n+m} \to \mathbb{C}^{n+m}$ is a continuous map that satisfies the following conditions:*

(i) $H(z,w)$ is injective on $\mathbb{C}^{n+m}$,

(ii) $||H(z,w)|| \to \infty$ as $||(z,w)|| \to \infty$, where $||\cdot||$ represents the Euclidean norm on $\mathbb{C}^{n+m}$,

then $H(z,w)$ is a homeomorphism of $\mathbb{C}^{n+m}$ onto itself.

Lemma 4.2. *([40]). For any vectors $x,\ y \in \mathbb{C}^n$, positive definite Hermitian matrix $P \in \mathbb{C}^{n\times n}$, and real constant $\varepsilon > 0$, the following linear matrix inequality (LMI) holds:*

$$x^*y + y^*x \leq \varepsilon x^*Px + \frac{1}{\varepsilon}y^*P^{-1}y.$$

Lemma 4.3. *([36]) For any vector function $z : [a,b] \to \mathbb{C}^n$ and positive definite Hermitian matrix $P \in \mathbb{C}^{n\times n}$, the following linear matrix inequality (LMI) holds:*

$$\left(\int_a^b z(s)ds\right)^* P \left(\int_a^b z(s)ds\right) \leq (b-a)\int_a^b z^*(s)Pz(s)ds,$$

where the integrals are assumed to be well defined.

Lemma 4.4. *([218]) For any vector function $z : [a,b] \to \mathbb{C}^n$ and positive definite Hermitian matrix $P \in \mathbb{C}^{n\times n}$, the following linear matrix inequality (LMI) holds:*

$$\left(\int_a^b\int_\theta^b z(s)dsd\theta\right)^* P \left(\int_a^b\int_\theta^b z(s)dsd\theta\right) \leq \frac{(b-a)^2}{2}\int_a^b\int_\theta^b z^*(s)Pz(s)dsd\theta,$$

where the integrals are assumed to be well defined.

Lemma 4.5. *([37]). A Hermitian matrix P is negative definite, $P < 0$, if and only if*

$$\begin{bmatrix} P^R & -P^I\\ P^I & P^R\end{bmatrix} < 0.$$

Lemma 4.6. *For any vectors $z_1,\ z_2 \in \mathbb{C}^n$, any positive definite Hermitian matrix $P \in \mathbb{C}^{n\times n}$, any matrix $Q \in \mathbb{C}^{n\times n}$, and any $\alpha \in (0,1)$, such that $\begin{bmatrix} P & Q \\ Q^* & P \end{bmatrix} \geq 0$, the following linear matrix inequality (LMI) holds:*

$$\frac{1}{\alpha} z_1^* P z_1 + \frac{1}{1-\alpha} z_2^* P z_2 \geq \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}^* \begin{bmatrix} P & Q \\ Q^* & P \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}.$$

Proof. Let $z_1 = z_1^R + \imath z_1^I$, $z_2 = z_2^R + \imath z_2^I$, $P = P^R + \imath P^I$, where $P^* = P \Leftrightarrow (P^R)^T = P^R$, $-(P^I)^T = P^I$, $Q = Q^R + \imath Q^I$. Then, by the reciprocally convex combination lemma [138], we have that

$$\begin{aligned}
\frac{1}{\alpha} z_1^* P z_1 + \frac{1}{1-\alpha} z_2^* P z_2 &= \frac{1}{\alpha} \begin{bmatrix} z_1^R \\ z_1^I \end{bmatrix}^T \begin{bmatrix} P^R & -P^I \\ P^I & P^R \end{bmatrix} \begin{bmatrix} z_1^R \\ z_1^I \end{bmatrix} + \frac{1}{1-\alpha} \begin{bmatrix} z_2^R \\ z_2^I \end{bmatrix}^T \begin{bmatrix} P^R & -P^I \\ P^I & P^R \end{bmatrix} \begin{bmatrix} z_2^R \\ z_2^I \end{bmatrix} \\
&\geq \begin{bmatrix} z_1^R \\ z_1^I \\ z_2^R \\ z_2^I \end{bmatrix}^T \begin{bmatrix} P^R & -P^I & Q^R & -Q^I \\ P^I & P^R & Q^I & Q^R \\ (Q^R)^T & (Q^I)^T & P^R & -P^I \\ -(Q^I)^T & (Q^R)^T & P^I & P^R \end{bmatrix} \begin{bmatrix} z_1^R \\ z_1^I \\ z_2^R \\ z_2^I \end{bmatrix} \\
&= \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}^* \begin{bmatrix} P & Q \\ Q^* & P \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}.
\end{aligned}$$

□

Lemma 4.7. *For any differentiable function $z : [a,b] \to \mathbb{C}^n$ and positive definite Hermitian matrix $P \in \mathbb{C}^{n\times n}$, the following linear matrix inequality (LMI) holds:*

$$\int_a^b \dot{z}^*(s) P \dot{z}(s) ds \geq \frac{1}{b-a} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}^* \begin{bmatrix} P & 0 \\ 0 & 3P \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix},$$

where $\xi_1 = z(b) - z(a)$, $\xi_2 = z(b) + z(a) - \frac{2}{b-a}\int_a^b z(s)ds$.

Proof. Let $z(s) = z^R(s) + \imath z^I(s)$, $\forall s$, $P = P^R + \imath P^I$, where $P^* = P \Leftrightarrow (P^R)^T = P^R$, $-(P^I)^T = P^I$. Then, by the Wirtiger-based integral inequality [194], we have that

$$\begin{aligned}
\int_a^b \dot{z}^*(s) P \dot{z}(s) ds &= \int_a^b \begin{bmatrix} \dot{z}^R(s) \\ \dot{z}^I(s) \end{bmatrix}^T \begin{bmatrix} P^R & -P^I \\ P^I & P^R \end{bmatrix} \begin{bmatrix} \dot{z}^R(s) \\ \dot{z}^I(s) \end{bmatrix} ds \\
&\geq \frac{1}{b-a} \begin{bmatrix} \xi_1^R \\ \xi_1^I \\ \xi_2^R \\ \xi_2^I \end{bmatrix}^T \begin{bmatrix} P^R & -P^I & 0 & 0 \\ P^I & P^R & 0 & 0 \\ 0 & 0 & 3P^R & -3P^I \\ 0 & 0 & 3P^I & 3P^R \end{bmatrix} \begin{bmatrix} \xi_1^R \\ \xi_1^I \\ \xi_2^R \\ \xi_2^I \end{bmatrix} \\
&= \frac{1}{b-a} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}^* \begin{bmatrix} P & 0 \\ 0 & 3P \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix},
\end{aligned}$$

where $\xi_1 = z(b) - z(a)$, $\xi_2 = z(b) + z(a) - \frac{2}{b-a}\int_a^b z(s)ds$. □

We now give an LMI-based sufficient condition for the existence and uniqueness of the equilibrium point for system (4.1.1).

Theorem 4.1. *Under Assumptions 4.1 and 4.2, the system* (4.1.1) *has a unique equilibrium point if there exist positive definite Hermitian matrices P_1, P_2, and positive diagonal matrices G_1, G_2, G_3, G_4, such that the following LMIs hold:*

$$\Omega_1 = \begin{bmatrix} \Omega_{1,1}^1 & P_1A_1 & P_1B_1 \\ \star & -G_1 & 0 \\ \star & \star & -G_3 \end{bmatrix} < 0, \tag{4.1.2}$$

$$\Omega_2 = \begin{bmatrix} \Omega_{1,1}^2 & P_2A_2 & P_2B_2 \\ \star & -G_2 & 0 \\ \star & \star & -G_4 \end{bmatrix} < 0, \tag{4.1.3}$$

where $\Omega_{1,1}^1 = -P_1D_1 - D_1P_1 + L_{f_2}^* G_2 L_{f_2} + L_{g_2}^* G_4 L_{g_2}$, $\Omega_{1,1}^2 = -P_2D_2 - D_2P_2 + L_{f_1}^* G_1 L_{f_1} + L_{g_1}^* G_3 L_{g_1}$.

Proof. Define the function $H(z,w) : \mathbb{C}^{n+m} \to \mathbb{C}^{n+m}$ by

$$H(z,w) = -\begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}\begin{bmatrix} z \\ w \end{bmatrix} + \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}\begin{bmatrix} f_1(w) \\ f_2(z) \end{bmatrix} + \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix}\begin{bmatrix} g_1(w) \\ g_2(z) \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{4.1.4}$$

We begin by proving that H is injective. For this, assume by contradiction that there exist $\begin{bmatrix} z \\ w \end{bmatrix}, \begin{bmatrix} z' \\ w' \end{bmatrix} \in \mathbb{C}^{n+m}$, $\begin{bmatrix} z \\ w \end{bmatrix} \neq \begin{bmatrix} z' \\ w' \end{bmatrix}$, such that $H(z,w) = H(z',w')$, or, equivalently

$$\begin{aligned} H(z,w) - H(z',w') &= -\begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}\begin{bmatrix} z-z' \\ w-w' \end{bmatrix} + \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}\begin{bmatrix} f_1(w)-f_1(w') \\ f_2(z)-f_2(z') \end{bmatrix} \\ &\quad + \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix}\begin{bmatrix} g_1(w)-g_1(w') \\ g_2(z)-g_2(z') \end{bmatrix} \\ &= 0. \end{aligned}$$

Multiplying to the left by $\begin{bmatrix} z-z' \\ w-w' \end{bmatrix}^* \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}$, we get

$$\begin{bmatrix} z-z' \\ w-w' \end{bmatrix}^* \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix} \left(-\begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}\begin{bmatrix} z-z' \\ w-w' \end{bmatrix} + \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}\begin{bmatrix} f_1(w)-f_1(w') \\ f_2(z)-f_2(z') \end{bmatrix} \right.$$
$$\left. + \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix}\begin{bmatrix} g_1(w)-g_1(w') \\ g_2(z)-g_2(z') \end{bmatrix} \right) = 0,$$

that is

$$-(z-z')^*P_1D_1(z-z') - (w-w')^*P_2D_2(w-w') + (z-z')^*P_1A_1(f_1(w)-f_1(w'))$$
$$+(w-w')^*P_2A_2(f_2(z)-f_2(z')) + (z-z')^*P_1B_1(g_1(w)-g_1(w')) + (w-w')^*P_2B_2(g_2(z)-g_2(z')) = 0. \tag{4.1.5}$$

Taking the complex conjugate of this relation, yields

$$-(z-z')^*D_1P_1(z-z') - (w-w')^*D_2P_2(w-w') + (f_1(w)-f_1(w'))^*A_1^*P_1(z-z')$$
$$+(f_2(z)-f_2(z'))^*A_2^*P_2(w-w') + (g_1(w)-g_1(w'))^*B_1^*P_1(z-z') + (g_2(z)-g_2(z'))^*B_2^*P_2(w-w') = 0. \tag{4.1.6}$$

Now, adding up (4.1.5) and (4.1.6), we have

$$(z-z')^*(-P_1D_1-D_1P_1)(z-z') + (w-w')^*(-P_2D_2-D_2P_2)(w-w') + (z-z')^*P_1A_1(f_1(w)-f_1(w'))$$

$$+(f_1(w)-f_1(w'))^*A_1^*P_1(z-z')+(w-w')^*P_2A_2(f_2(z)-f_2(z'))+(f_2(z)-f_2(z'))^*A_2^*P_2(w-w')$$
$$+(z_1-z_2)^*P_1B_1(g_1(w)-g_1(w'))+(g_1(w)-g_1(w'))^*B_1^*P_1(z-z')+(w-w')^*P_2B_2(g_2(z)-g_2(z'))$$
$$+(g_2(z)-g_2(z'))^*B_2^*P_2(w-w')=0.$$

From Assumption 4.1 on the activation functions, we deduce that there exist positive diagonal matrices G_1, G_2, G_3, G_4, such that

$$(f_1(w)-f_1(w'))^*G_1(f_1(w)-f_1(w'))\leq(w-w')^*L_{f_1}^*G_1L_{f_1}(w-w'), \tag{4.1.7}$$

$$(f_2(z)-f_2(z'))^*G_2(f_2(z)-f_2(z'))\leq(z-z')^*L_{f_2}^*G_2L_{f_2}(z-z'), \tag{4.1.8}$$

$$(g_1(w)-g_1(w'))^*G_3(g_1(w)-g_1(w'))\leq(w-w')^*L_{g_1}^*G_3L_{g_1}(w-w'), \tag{4.1.9}$$

$$(g_2(z)-g_2(z'))^*G_4(g_2(z)-g_2(z'))\leq(z-z')^*L_{g_2}^*G_4L_{g_2}(z-z'). \tag{4.1.10}$$

Lemma 4.2 and relations (4.1.7)–(4.1.10) allow us to write the following inequalities

$$(z-z')^*(-P_1D_1-D_1P_1)(z-z')+(w-w')^*(-P_2D_2-D_2P_2)(w-w')+(z-z')^*P_1A_1(f_1(w)-f_1(w'))$$
$$+(f_1(w)-f_1(w'))^*A_1^*P_1(z-z')+(w-w')^*P_2A_2(f_2(z)-f_2(z'))+(f_2(z)-f_2(z'))^*A_2^*P_2(w-w')$$
$$+(z_1-z_2)^*P_1B_1(g_1(w)-g_1(w'))+(g_1(w)-g_1(w'))^*B_1^*P_1(z-z')+(w-w')^*P_2B_2(g_2(z)-g_2(z'))$$
$$+(g_2(z)-g_2(z'))^*B_2^*P_2(w-w')\leq$$
$$(z-z')^*(-P_1D_1-D_1P_1)(z-z')+(w-w')^*(-P_2D_2-D_2P_2)(w-w')+(z-z')^*P_1A_1G_1^{-1}A_1^*P_1(z-z')$$
$$+(f_1(w)-f_1(w'))^*G_1(f_1(w)-f_1(w'))+(w-w')^*P_2A_2G_2^{-1}A_2^*P_2(w-w')+(f_2(z)-f_2(z'))^*G_2(f_2(z)-f_2(z'))$$
$$+(z-z')^*P_1B_1G_3^{-1}B_1^*P_1(z-z')+(g_1(w)-g_1(w'))^*G_3(g_1(w)-g_1(w'))+(w-w')^*P_2B_2G_4^{-1}B_2^*P_2(w-w')$$
$$+(g_2(z)-g_2(z'))^*G_4(g_2(z)-g_2(z'))\leq$$
$$(z-z')^*(-P_1D_1-D_1P_1+P_1A_1G_1^{-1}A_1^*P_1+P_1B_1G_3^{-1}B_1^*P_1+L_{f_2}^*G_2L_{f_2}+L_{g_2}^*G_4L_{g_2})(z-z')$$
$$+(w-w')^*(-P_2D_2-D_2P_2+P_2A_2G_2^{-1}A_2^*P_2+P_2B_2G_4^{-1}B_2^*P_2+L_{f_1}^*G_1L_{f_1}+L_{g_1}^*G_3L_{g_1})(w-w')<0, \tag{4.1.11}$$

where the last inequality is based on the following relations:

$$-P_1D_1-D_1P_1+P_1A_1G_1^{-1}A_1^*P_1+P_1B_1G_3^{-1}B_1^*P_1+L_{f_2}^*G_2L_{f_2}+L_{g_2}^*G_4L_{g_2}<0, \tag{4.1.12}$$

$$-P_2D_2-D_2P_2+P_2A_2G_2^{-1}A_2^*P_2+P_2B_2G_4^{-1}B_2^*P_2+L_{f_1}^*G_1L_{f_1}+L_{g_1}^*G_3L_{g_1}<0, \tag{4.1.13}$$

which can be immediately deduced from conditions (4.1.2) and (4.1.3), using Schur's complement. We have thus obtained that

$$H(z,w)-H(z',w')<0,$$

in contradiction with our initial assumption. This means that function $H(z,w)$ is injective.

Next, we will prove that $||H(z,w)||\to\infty$ as $||(z,w)||\to\infty$. From (4.1.12) and (4.1.13), we deduce that there exists a sufficiently small constant $\varepsilon>0$, such that

$$-P_1D_1-D_1P_1+P_1A_1G_1^{-1}A_1^*P_1+P_1B_1G_3^{-1}B_1^*P_1+L_{f_2}^*G_2L_{f_2}+L_{g_2}^*G_4L_{g_2}<-\varepsilon I_n,$$

$$-P_2D_2-D_2P_2+P_2A_2G_2^{-1}A_2^*P_2+P_2B_2G_4^{-1}B_2^*P_2+L_{f_1}^*G_1L_{f_1}+L_{g_1}^*G_3L_{g_1}<-\varepsilon I_m.$$

Taking $(z',w')=(0,0)$, and using (4.1.11) and the above relations, we have

$$\begin{bmatrix}z\\w\end{bmatrix}^*(H(z,w)-H(0,0))=\begin{bmatrix}z\\w\end{bmatrix}^*\left(-\begin{bmatrix}D_1&0\\0&D_2\end{bmatrix}\begin{bmatrix}z\\w\end{bmatrix}+\begin{bmatrix}A_1&0\\0&A_2\end{bmatrix}\begin{bmatrix}f_1(w)\\f_2(z)\end{bmatrix}+\begin{bmatrix}B_1&0\\0&B_2\end{bmatrix}\begin{bmatrix}g_1(w)\\g_2(z)\end{bmatrix}\right)\leq$$

$$z^*(-P_1D_1 - D_1P_1 + P_1A_1G_1^{-1}A_1^*P_1 + P_1B_1G_3^{-1}B_1^*P_1 + L_{f_2}^*G_2L_{f_2} + L_{g_2}^*G_4L_{g_2})z$$
$$+w^*(-P_2D_2-D_2P_2+P_2A_2G_2^{-1}A_2^*P_2+P_2B_2G_4^{-1}B_2^*P_2+L_{f_1}^*G_1L_{f_1}+L_{g_1}^*G_3L_{g_1})w < -\varepsilon(||z||^2+||w||^2). \tag{4.1.14}$$

By applying the Cauchy-Schwarz inequality, relation (4.1.14) becomes:

$$\begin{aligned}\varepsilon(||z||^2 + ||w||^2) &\leq \left\| \begin{bmatrix} z \\ w \end{bmatrix}^* (H(z,w) - H(0,0)) \right\| \\ &\leq \sqrt{||z||^2 + ||w||^2}\,(||H(z,w)|| + ||H(0,0)||)\,,\end{aligned}$$

which immediately yields that $||H(z,w)|| \to \infty$ when $||(z,w)|| \to \infty$.

Now, function $H(z,w)$ satisfies both conditions in Lemma 4.1, which means that it is a homeomorphism of $\mathbb{C}^{n+m}$ onto itself. Thus, the equation $H(z,w) = 0$ has a unique solution, and so system (4.1.1) has a unique equilibrium point, which will be denoted by $\begin{bmatrix} \hat{z} \\ \hat{w} \end{bmatrix}$. □

We can now shift the equilibrium point of system (4.1.1) to the origin, yielding

$$\begin{cases} \dot{\tilde{z}}(t) = -D_1\tilde{z}(t-\delta) + A_1\tilde{f}_1(\tilde{w}(t)) + B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) + E_1\dot{\tilde{z}}(t-\eta(t)),\ t \neq t_k,\ t > 0, \\ \dot{\tilde{w}}(t) = -D_2\tilde{w}(t-\delta) + A_2\tilde{f}_2(\tilde{z}(t)) + B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) + E_2\dot{\tilde{w}}(t-\eta(t)),\ t \neq t_k,\ t > 0, \\ \Delta\tilde{z}(t_k) = \tilde{z}(t_k) - \tilde{z}(t_k^-) = J_k^1(\tilde{z}(t_k^-), \tilde{z}_{t_k^-}),\ k \in \mathbb{Z}^+, \\ \Delta\tilde{w}(t_k) = \tilde{w}(t_k) - \tilde{w}(t_k^-) = J_k^2(\tilde{w}(t_k^-), \tilde{w}_{t_k^-}),\ k \in \mathbb{Z}^+, \end{cases} \tag{4.1.15}$$

where $\tilde{z}(t) = z(t) - \hat{z}$, $\tilde{w}(t) = w(t) - \hat{w}$, $\tilde{f}_1(\tilde{w}(t)) = f_1(\tilde{w}(t) + \hat{w}) - f_1(\hat{w})$, $\tilde{g}_1(\tilde{w}(t-\tau(t))) = g_1(\tilde{w}(t-\tau(t)) + \hat{w}) - g_1(\hat{w})$, $\tilde{f}_2(\tilde{z}(t)) = f_2(\tilde{z}(t) + \hat{z}) - f_2(\hat{z})$, $\tilde{g}_2(\tilde{z}(t-\tau(t))) = g_2(\tilde{z}(t - \tau(t)) + \hat{z}) - g_2(\hat{z})$. From this point on, we will study the stability properties of the origin of system (4.1.15), which will be equivalent with the stability of the unique equilibrium point of system (4.1.1).

Theorem 4.2. *If Assumptions 4.1 and 4.2–4.3 hold, the origin of the system* (4.1.15) *is globally μ-stable if there exist positive constants γ_1, γ_2, γ_3, γ_4, such that*

$$\frac{\dot{\mu}(t)}{\mu(t)} \leq \gamma_1,\ \frac{\inf_{s\in[t-\delta,t]}\mu(s)}{\mu(t)} \geq \gamma_2,\ \frac{\mu(t-\tau(t))}{\mu(t)} \geq \gamma_3,\ \frac{\min\{\inf_{s\in[t-\eta,t]}\mu(s),\ \mu(t-\eta(t))\}}{\mu(t)} \geq \gamma_4, \tag{4.1.16}$$

$\forall t \geq 0$, *and there exist positive definite Hermitian matrices P_1, P_2, Q_1, Q_2, R_1, R_2, S_1, S_2, T_1, T_2, …, T_6, U_1, U_2, X_1, X_2, Y_1, Y_2, Z_1, Z_2, positive diagonal matrices G_1, G_2, …, G_8, and any matrices M_1, M_2, N_1, N_2, …, N_{10}, such that the following LMIs hold*

$$\Pi - \gamma_4^2\zeta_1^*\Phi_1\zeta_1 - \gamma_4^2\zeta_2^*\Phi_2\zeta_2 < 0,\ \Phi_1 > 0,\ \Phi_2 > 0, \tag{4.1.17}$$

$$\Psi_1^*\Theta_1\Psi_1 < \Theta_1,\ \Psi_2^*\Theta_2\Psi_2 < \Theta_2, \tag{4.1.18}$$

where $\Pi_{1,1} = 2\gamma_1P_1+2\gamma_1Q_1-D_1Q_1-Q_1D_1+R_1+\delta^2S_1+T_1-4\gamma_4^2\eta^2Z_1+L_{f_2}^*G_1L_{f_2}+L_{g_2}^*G_3L_{g_2}$, $\Pi_{1,2} = Q_1$, $\Pi_{1,4} = P_1E_1$, $\Pi_{1,7} = -P_1D_1 + Q_1D_1$, $\Pi_{1,13} = D_1Q_1D_1 - 2\gamma_1Q_1D_1$, $\Pi_{1,14} = 4\gamma_4^2\eta Z_1$, $\Pi_{1,25} = P_1A_1$, $\Pi_{1,28} = P_1B_1$, $\Pi_{2,2} = U_1+X_1+\eta^2Y_1+\eta^4Z_1-N_1-N_1^*$, $\Pi_{2,4} = N_1E_1+N_5^*$, $\Pi_{2,7} = -N_1D_1 - N_2^*$, $\Pi_{2,13} = -Q_1D_1$, $\Pi_{2,25} = N_1A_1 + N_3^*$, $\Pi_{2,28} = N_1B_1 + N_4^*$, $\Pi_{3,3} = -\gamma_4^2X_1$, $\Pi_{4,4} = -(1-\eta_d)\gamma_4^2U_1-N_5E_1-E_1^*N_5^*$, $\Pi_{4,7} = E_1^*N_2^*+N_5D_1$, $\Pi_{4,25} = -N_5A_1-E_1^*N_3^*$, $\Pi_{4,28} = -N_5B_1 - E_1^*N_4^*$, $\Pi_{7,7} = -\gamma_2^2R_1 - N_2D_1 - D_1N_2^*$, $\Pi_{7,13} = -D_1Q_1D_1$, $\Pi_{7,25} = N_2A_1 + D_1N_3^*$, $\Pi_{7,28} = N_2B_1 + D_1N_4^*$, $\Pi_{8,8} = -(1-\tau_d)T_1 + L_{f_2}^*G_2L_{f_2} + L_{g_2}^*G_4L_{g_2}$, $\Pi_{9,9} = T_2-G_1-N_8A_2-A_2^*N_8^*$, $\Pi_{9,12} = -N_8B_2-A_2^*N_9^*$, $\Pi_{9,17} = A_2^*P_2$, $\Pi_{9,18} = A_2^*N_6^*+N_8$, $\Pi_{9,20} =$

$-A_2^*N_{10}^* - N_8E_2$, $\Pi_{9,23} = A_2^*N_7^* + N_8D_2$, $\Pi_{10,10} = -(1-\tau_d)\gamma_3^2T_2 - G_2$, $\Pi_{11,11} = T_3 - G_3$, $\Pi_{12,12} = -(1-\tau_d)\gamma_3^2T_3 - G_4 - N_9B_2 - B_2^*N_9^*$, $\Pi_{12,17} = B_2^*P_2$, $\Pi_{12,18} = B_2^*N_6^* + N_9$, $\Pi_{12,20} = -B_2^*N_{10}^* - N_9E_2$, $\Pi_{12,23} = B_2^*N_7^* + N_9D_2$, $\Pi_{13,13} = 2\gamma_1D_1Q_1D_1 - \gamma_2^2S_1$, $\Pi_{14,14} = -4\gamma_4^2Z_1$, $\Pi_{17,17} = 2\gamma_1P_2 + 2\gamma_1Q_2 - D_2Q_2 - Q_2D_2 + R_2 + \delta^2S_2 + T_2 - 4\gamma_4^2\eta^2Z_2 + L_{f_1}^*G_5L_{f_1} + L_{g_1}^*G_7L_{g_1}$, $\Pi_{17,18} = Q_2$, $\Pi_{17,20} = P_2E_2$, $\Pi_{17,23} = -P_2D_2 + Q_2D_2$, $\Pi_{17,29} = D_2Q_2D_2 - 2\gamma_1Q_2D_2$, $\Pi_{17,30} = 4\gamma_4^2\eta Z_2$, $\Pi_{18,18} = U_2 + X_2 + \eta^2Y_2 + \eta^4Z_2 - N_6 - N_6^*$, $\Pi_{18,20} = N_6E_2 + N_{10}^*$, $\Pi_{18,23} = -N_6D_2 - N_7^*$, $\Pi_{18,29} = -Q_2D_2$, $\Pi_{19,19} = -\gamma_4^2X_2$, $\Pi_{20,20} = -(1-\eta_d)\gamma_4^2U_2 - N_{10}E_2 - E_2^*N_{10}^*$, $\Pi_{20,23} = E_2^*N_7^* + N_{10}D_2$, $\Pi_{23,23} = -\gamma_2^2R_2 - N_7D_2 - D_2N_7^*$, $\Pi_{23,29} = -D_2Q_2D_2$, $\Pi_{24,24} = -(1-\tau_d)T_2 + L_{f_1}^*G_6L_{f_1} + L_{g_1}^*G_8L_{g_1}$, $\Pi_{25,25} = T_5 - G_5 - N_3A_1 - A_1^*N_3^*$, $\Pi_{25,28} = -N_3B_1 - A_1^*N_4^*$, $\Pi_{26,26} = -(1-\tau_d)\gamma_3^2T_5 - G_6$, $\Pi_{27,27} = T_6 - G_7$, $\Pi_{28,28} = -(1-\tau_d)\gamma_3^2T_6 - G_8 - N_4B_1 - B_1^*N_4^*$, $\Pi_{29,29} = 2\gamma_1D_2Q_2D_2 - \gamma_2^2S_2$, $\Pi_{30,30} = -4\gamma_4^2Z_2$,

$$\zeta_1 = \begin{bmatrix} e_1^n - e_6^n \\ e_1^n + e_6^n - 2e_{15}^n \\ e_6^n - e_5^n \\ e_6^n + e_5^n - 2e_{16}^n \end{bmatrix},\ \zeta_2 = \begin{bmatrix} e_{17}^m - e_{22}^m \\ e_{17}^m + e_{22}^m - 2e_{31}^m \\ e_{22}^m - e_{21}^m \\ e_{22}^m + e_{21}^m - 2e_{32}^m \end{bmatrix},\ e_i^p = \begin{bmatrix} 0_{p\times(i-1)p} & I_p & 0_{p\times(32-i)p} \end{bmatrix},\ i = 1,\ldots,32,$$

$$\Phi_1 = \begin{bmatrix} \overline{Y}_1 & M_1 \\ M_1^* & \overline{Y}_1 \end{bmatrix},\ \Phi_2 = \begin{bmatrix} \overline{Y}_2 & M_2 \\ M_2^* & \overline{Y}_2 \end{bmatrix},$$

$$\overline{Y}_1 = \begin{bmatrix} Y_1 & 0 \\ 0 & 3Y_1 \end{bmatrix},\ \overline{Y}_2 = \begin{bmatrix} Y_2 & 0 \\ 0 & 3Y_2 \end{bmatrix},$$

$$\Psi_1 = \begin{bmatrix} I_n + F_k^1 & -F_k^1D_1 \\ 0 & I_n \end{bmatrix},\ \Theta_1 = \begin{bmatrix} P_1 + Q_1 & -Q_1D_1 \\ -D_1Q_1 & D_1Q_1D_1 \end{bmatrix},$$

$$\Psi_2 = \begin{bmatrix} I_m + F_k^2 & -F_k^2D_2 \\ 0 & I_m \end{bmatrix},\ \Theta_2 = \begin{bmatrix} P_2 + Q_2 & -Q_2D_2 \\ -D_2Q_2 & D_2Q_2D_2 \end{bmatrix}.$$

Proof. Consider the following Lyapunov-Krasovskii functional

$$V(t) = V_1(t) + V_2(t) + V_3(t) + V_4(t) + V_5(t) + V_6(t) + V_7(t) + V_8(t) + V_9(t), \qquad (4.1.19)$$

where

$$V_1(t) = \mu^2(t) \begin{bmatrix} \tilde{z}(t) \\ \tilde{w}(t) \end{bmatrix}^* \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix} \begin{bmatrix} \tilde{z}(t) \\ \tilde{w}(t) \end{bmatrix},$$

$$\begin{aligned} V_2(t) &= \mu^2(t) \left(\begin{bmatrix} \tilde{z}(t) \\ \tilde{w}(t) \end{bmatrix}^* - \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \int_{t-\delta}^{t} \begin{bmatrix} \tilde{z}(s) \\ \tilde{w}(s) \end{bmatrix} ds \right)^* \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \\ &\quad \times \left(\begin{bmatrix} \tilde{z}(t) \\ \tilde{w}(t) \end{bmatrix}^* - \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \int_{t-\delta}^{t} \begin{bmatrix} \tilde{z}(s) \\ \tilde{w}(s) \end{bmatrix} ds \right), \end{aligned}$$

$$V_3(t) = \int_{t-\delta}^{t} \mu^2(s) \begin{bmatrix} \tilde{z}(s) \\ \tilde{w}(s) \end{bmatrix}^* \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \begin{bmatrix} \tilde{z}(s) \\ \tilde{w}(s) \end{bmatrix} ds,$$

$$V_4(t) = \delta \int_{-\delta}^{0} \int_{t+\theta}^{t} \mu^2(s) \begin{bmatrix} \tilde{z}(s) \\ \tilde{w}(s) \end{bmatrix}^* \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} \tilde{z}(s) \\ \tilde{w}(s) \end{bmatrix} ds d\theta,$$

$$V_5(t) = \int_{t-\tau(t)}^{t} \mu^2(s) \sigma^*(s) T \sigma(s) ds,$$

$$T = \text{diag}(T_1, T_2, T_3, T_4, T_5, T_6),$$

$$\sigma(s) = \begin{bmatrix} \tilde{z}^*(s) & \tilde{f}_2^*(\tilde{z}(s)) & \tilde{g}_2^*(\tilde{z}(s)) & \tilde{w}^*(s) & \tilde{f}_1^*(\tilde{w}(s)) & \tilde{g}_1^*(\tilde{w}(s)) \end{bmatrix}^*,$$

$$V_6(t) = \int_{t-\eta(t)}^{t} \mu^2(s) \begin{bmatrix} \dot{\tilde{z}}(s) \\ \dot{\tilde{w}}(s) \end{bmatrix}^* \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} \dot{\tilde{z}}(s) \\ \dot{\tilde{w}}(s) \end{bmatrix} ds,$$

$$V_7(t) = \int_{t-\eta}^{t} \mu^2(s) \begin{bmatrix} \dot{\tilde{z}}(s) \\ \dot{\tilde{w}}(s) \end{bmatrix}^* \begin{bmatrix} X_1 & 0 \\ 0 & X_2 \end{bmatrix} \begin{bmatrix} \dot{\tilde{z}}(s) \\ \dot{\tilde{w}}(s) \end{bmatrix} ds,$$

$$V_8(t) = \eta \int_{-\eta}^{0} \int_{t+\theta}^{t} \mu^2(s) \begin{bmatrix} \dot{\tilde{z}}(s) \\ \dot{\tilde{w}}(s) \end{bmatrix}^* \begin{bmatrix} Y_1 & 0 \\ 0 & Y_2 \end{bmatrix} \begin{bmatrix} \dot{\tilde{z}}(s) \\ \dot{\tilde{w}}(s) \end{bmatrix} ds d\theta,$$

$$V_9(t) = 2\eta^2 \int_{-\eta}^{0} \int_{\theta}^{0} \int_{t+\lambda}^{t} \mu^2(s) \begin{bmatrix} \dot{\tilde{z}}(s) \\ \dot{\tilde{w}}(s) \end{bmatrix}^* \begin{bmatrix} Z_1 & 0 \\ 0 & Z_2 \end{bmatrix} \begin{bmatrix} \dot{\tilde{z}}(s) \\ \dot{\tilde{w}}(s) \end{bmatrix} ds d\lambda d\theta.$$

The derivative of $V(t)$ along the trajectories of system (4.1.15) at the points $t \neq t_k$, $k \in \mathbb{Z}^+$, is

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t) + \dot{V}_4(t) + \dot{V}_5(t) + \dot{V}_6(t) + \dot{V}_7(t) + \dot{V}_8(t),$$

where

$$\begin{aligned} \dot{V}_1(t) &= 2\mu(t)\dot{\mu}(t)\left[\tilde{z}^*(t)P_1\tilde{z}(t) + \tilde{w}^*(t)P_2\tilde{w}(t)\right] + \mu^2(t)\left[\dot{\tilde{z}}^*(t)P_1\tilde{z}(t) + \tilde{z}^*(t)P_1\dot{\tilde{z}}(t)\right. \\ &\quad \left.+\dot{\tilde{w}}^*(t)P_2\tilde{w}(t) + \tilde{w}^*(t)P_2\dot{\tilde{w}}(t)\right] \\ &\leq \mu^2(t)\left[2\gamma_1\tilde{z}^*(t)P_1\tilde{z}(t) + 2\gamma_1\tilde{w}^*(t)P_2\tilde{w}(t) + \right. \\ &\quad \left(-D_1\tilde{z}(t-\delta) + A_1\tilde{f}_1(\tilde{w}(t)) + B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) + E_1\dot{\tilde{z}}(t-\eta(t))\right)^* P_1\tilde{z}(t) \\ &\quad +\tilde{z}^*(t)P_1\left(-D_1\tilde{z}(t-\delta) + A_1\tilde{f}_1(\tilde{w}(t)) + B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) + E_1\dot{\tilde{z}}(t-\eta(t))\right)^* \\ &\quad +\left(-D_2\tilde{w}(t-\delta) + A_2\tilde{f}_2(\tilde{z}(t)) + B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) + E_2\dot{\tilde{w}}(t-\eta(t))\right)^* P_2\tilde{w}(t) \\ &\quad \left.+\tilde{w}^*(t)P_2\left(-D_2\tilde{w}(t-\delta) + A_2\tilde{f}_2(\tilde{z}(t)) + B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) + E_2\dot{\tilde{w}}(t-\eta(t))\right)\right], \end{aligned} \tag{4.1.20}$$

$$\begin{aligned} \dot{V}_2(t) &\leq \mu^2(t)\left[2\gamma_1\left(\tilde{z}(t) - D_1\int_{t-\delta}^{t}\tilde{z}(s)ds\right)^* Q_1\left(\tilde{z}(t) - D_1\int_{t-\delta}^{t}\tilde{z}(s)ds\right)\right. \\ &\quad +2\gamma_1\left(\tilde{w}(t) - D_2\int_{t-\delta}^{t}\tilde{w}(s)ds\right)^* Q_2\left(\tilde{w}(t) - D_2\int_{t-\delta}^{t}\tilde{w}(s)ds\right) \\ &\quad +\left(\dot{\tilde{z}}(t) - D_1\tilde{z}(t) + D_1\tilde{z}(t-\delta)\right)^* Q_1\left(\tilde{z}(t) - D_1\int_{t-\delta}^{t}\tilde{z}(s)ds\right) \\ &\quad +\left(\tilde{z}(t) - D_1\int_{t-\delta}^{t}\tilde{z}(s)ds\right)^* Q_1\left(\dot{\tilde{z}}(t) - D_1\tilde{z}(t) + D_1\tilde{z}(t-\delta)\right) \\ &\quad +\left(\dot{\tilde{w}}(t) - D_2\tilde{w}(t) + D_2\tilde{w}(t-\delta)\right)^* Q_2\left(\tilde{w}(t) - D_2\int_{t-\delta}^{t}\tilde{w}(s)ds\right) \\ &\quad \left.+\left(\tilde{w}(t) - D_2\int_{t-\delta}^{t}\tilde{w}(s)ds\right)^* Q_2\left(\dot{\tilde{w}}(t) - D_2\tilde{w}(t) + D_2\tilde{w}(t-\delta)\right)\right] \end{aligned} \tag{4.1.21}$$

$$\begin{aligned} \dot{V}_3(t) &= \mu^2(t)\tilde{z}^*(t)R_1\tilde{z}(t) - \mu^2(t-\delta)\tilde{z}^*(t-\delta)R_1\tilde{z}(t-\delta) \\ &\quad +\mu^2(t)\tilde{w}^*(t)R_2\tilde{w}(t) - \mu^2(t-\delta)\tilde{w}^*(t-\delta)R_2\tilde{w}(t-\delta) \\ &\leq \mu^2(t)\left[\tilde{z}^*(t)R_1\tilde{z}(t) - \gamma_2^2\tilde{z}^*(t-\delta)R_1\tilde{z}(t-\delta) + \tilde{w}^*(t)R_2\tilde{w}(t)\right. \\ &\quad \left.-\gamma_2^2\tilde{w}^*(t-\delta)R_2\tilde{w}(t-\delta)\right], \end{aligned} \tag{4.1.22}$$

$$\begin{aligned}
\dot{V}_4(t) &= \delta^2\mu^2(t)\tilde{z}^*(t)S_1\tilde{z}(t) - \delta\int_{t-\delta}^{t}\mu^2(s)\tilde{z}^*(s)S_1\tilde{z}(s)ds \\
&\quad +\delta^2\mu^2(t)\tilde{w}^*(t)S_2\tilde{w}(t) - \delta\int_{t-\delta}^{t}\mu^2(s)\tilde{w}^*(s)S_2\tilde{w}(s)ds \\
&\leq \mu^2(t)\left[\delta^2\tilde{z}^*(t)S_1\tilde{z}(t) - \gamma_2^2\left(\int_{t-\delta}^{t}\tilde{z}(s)ds\right)^* S_1\left(\int_{t-\delta}^{t}\tilde{z}(s)ds\right)\right. \\
&\quad \left. +\delta^2\tilde{w}^*(t)S_2\tilde{w}(t) - \gamma_2^2\left(\int_{t-\delta}^{t}\tilde{w}(s)ds\right)^* S_2\left(\int_{t-\delta}^{t}\tilde{w}(s)ds\right)\right], \qquad (4.1.23)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_5(t) &= \mu^2(t)\left[\tilde{z}^*(t)T_1\tilde{z}(t) + f_2^*(\tilde{z}(t))T_1f_2(\tilde{z}(t)) + g_2^*(\tilde{z}(t))T_1g_2(\tilde{z}(t)) + \tilde{w}^*(t)T_2\tilde{w}(t)\right. \\
&\quad \left. +f_1^*(\tilde{w}(t))T_2f_2(\tilde{w}(t)) + g_2^*(\tilde{w}(t))T_2g_2(\tilde{w}(t))\right] - (1-\dot{\tau}(t))\mu^2(t-\tau(t)) \\
&\quad \times\left[\tilde{z}^*(t-\tau(t))T_1\ \tilde{z}(t-\tau(t)) + f_2^*(\tilde{z}(t-\tau(t)))T_1f_2(\tilde{z}(t-\tau(t)))\right. \\
&\quad \left. +g_2^*(\tilde{z}(t-\tau(t)))T_1g_2(\tilde{z}(t-\tau(t))) + \tilde{w}^*(t-\tau(t))T_2\tilde{w}(t-\tau(t))\right. \\
&\quad \left. +f_1^*(\tilde{w}(t-\tau(t)))T_2f_1(\tilde{z}(t-\tau(t))) + g_1^*(\tilde{w}(t-\tau(t)))T_2g_1(\tilde{w}(t-\tau(t)))\right] \\
&\leq \mu^2(t)\left[\tilde{z}^*(t)T_1\tilde{z}(t) + f_2^*(\tilde{z}(t))T_1f_2(\tilde{z}(t)) + g_2^*(\tilde{z}(t))T_1g_2(\tilde{z}(t)) + \tilde{w}^*(t)T_2\tilde{w}(t)\right. \\
&\quad \left. +f_1^*(\tilde{w}(t))T_2f_2(\tilde{w}(t)) + g_2^*(\tilde{w}(t))T_2g_2(\tilde{w}(t)) - (1-\tau_d)\gamma_3^2\tilde{z}^*(t-\tau(t))T_1\tilde{z}(t-\tau(t))\right. \\
&\quad \left. -(1-\tau_d)\gamma_3^2f_2^*(\tilde{z}(t-\tau(t)))T_1f_2(\tilde{z}(t-\tau(t)))\right. \\
&\quad \left. -(1-\tau_d)\gamma_3^2g_2^*(\tilde{z}(t-\tau(t)))T_1g_2(\tilde{z}(t-\tau(t)))\right. \\
&\quad \left. -(1-\tau_d)\gamma_3^2\tilde{w}^*(t-\tau(t)))T_2\tilde{w}(t-\tau(t)))\right. \\
&\quad \left. -(1-\tau_d)\gamma_3^2f_1^*(\tilde{w}(t-\tau(t)))T_2f_1(\tilde{w}(t-\tau(t)))\right. \\
&\quad \left. -(1-\tau_d)\gamma_3^2g_1^*(\tilde{w}(t-\tau(t)))T_2g_1(\tilde{w}(t-\tau(t)))\right], \qquad (4.1.24)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_6(t) &\leq \mu^2(t)\left[\dot{\tilde{z}}^*(t)U_1\dot{\tilde{z}}(t) + \dot{\tilde{w}}^*(t)U_2\dot{\tilde{w}}(t) - (1-\eta_d)\gamma_4^2\dot{\tilde{z}}^*(t-\eta(t))U_1\dot{\tilde{z}}(t-\eta(t))\right. \\
&\quad \left. -(1-\eta_d)\gamma_4^2\dot{\tilde{w}}^*(t-\eta(t))U_2\dot{\tilde{w}}(t-\eta(t))\right], \qquad (4.1.25)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_7(t) &\leq \mu^2(t)\left[\dot{\tilde{z}}^*(t)X_1\dot{\tilde{z}}(t) + \dot{\tilde{w}}^*(t)X_2\dot{\tilde{w}}(t) - \gamma_4^2\dot{\tilde{z}}^*(t-\eta)X_1\dot{\tilde{z}}(t-\eta)\right. \\
&\quad \left. -\gamma_4^2\dot{\tilde{w}}^*(t-\eta)X_2\dot{\tilde{w}}(t-\eta)\right] \qquad (4.1.26)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_8(t) &\leq \mu^2(t)\left[\eta^2\dot{\tilde{z}}^*(t)Y_1\dot{\tilde{z}}(t) - \eta\gamma_4^2\int_{t-\eta}^{t}\dot{\tilde{z}}^*(s)Y_1\dot{\tilde{z}}(s)ds\right. \\
&\quad \left. +\eta^2\dot{\tilde{w}}^*(t)Y_2\dot{\tilde{w}}(t) - \eta\gamma_4^2\int_{t-\eta}^{t}\dot{\tilde{w}}^*(s)Y_2\dot{\tilde{w}}(s)ds\right], \qquad (4.1.27)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_9(t) &= \eta^4\mu^2(t)\dot{\tilde{z}}^*(t)Z_1\dot{\tilde{z}}(t) - 2\eta^2\int_{-\eta}^{0}\int_{t+\theta}^{t}\mu^2(s)\dot{\tilde{z}}^*(s)Z_1\dot{\tilde{z}}(s)dsd\theta \\
&\quad +\eta^4\mu^2(t)\dot{\tilde{w}}^*(t)Z_2\dot{\tilde{w}}(t) - 2\eta^2\int_{-\eta}^{0}\int_{t+\theta}^{t}\mu^2(s)\dot{\tilde{w}}^*(s)Z_2\dot{\tilde{w}}(s)dsd\theta \\
&\leq \mu^2(t)\left[\eta^4\dot{\tilde{z}}^*(t)Z_1\dot{\tilde{z}}(t) - 4\gamma_4^2\left(\int_{-\eta}^{0}\int_{t+\theta}^{t}\dot{\tilde{z}}(s)dsd\theta\right)^* Z_1\left(\int_{-\eta}^{0}\int_{t+\theta}^{t}\dot{\tilde{z}}(s)dsd\theta\right)\right. \\
&\quad \left. +\eta^4\dot{\tilde{w}}^*(t)Z_2\dot{\tilde{w}}(t) - 4\gamma_4^2\left(\int_{-\eta}^{0}\int_{t+\theta}^{t}\dot{\tilde{w}}(s)dsd\theta\right)^* Z_2\left(\int_{-\eta}^{0}\int_{t+\theta}^{t}\dot{\tilde{w}}(s)dsd\theta\right)\right]
\end{aligned}$$

$$
\begin{aligned}
&= \mu^2(t)\left(\eta^4\dot{\tilde{z}}^*(t)Z_1\dot{\tilde{z}}(t) - 4\gamma_4^2\left(\int_{-\eta}^{0}\left(\dot{\tilde{z}}(t)-\dot{\tilde{z}}(t+\theta)\right)d\theta\right)^* Z_1\left(\int_{-\eta}^{0}\left(\dot{\tilde{z}}(t)-\dot{\tilde{z}}(t+\theta)\right)d\theta\right)\right.\\
&\quad \left.+\eta^4\dot{\tilde{w}}^*(t)Z_2\dot{\tilde{w}}(t) - 4\gamma_4^2\left(\int_{-\eta}^{0}\left(\dot{\tilde{w}}(t)-\dot{\tilde{w}}(t+\theta)\right)d\theta\right)^* Z_2\left(\int_{-\eta}^{0}\left(\dot{\tilde{w}}(t)-\dot{\tilde{w}}(t+\theta)\right)d\theta\right)\right)\\
&= \mu^2(t)\left(\eta^4\dot{\tilde{z}}^*(t)Z_1\dot{\tilde{z}}(t) - 4\gamma_4^2\left(\eta\tilde{z}(t)-\int_{t-\eta}^{t}\tilde{z}(s)ds\right)^* Z_1\left(\eta\tilde{z}(t)-\int_{t-\eta}^{t}\tilde{z}(s)ds\right)\right.\\
&\quad \left.+\eta^4\dot{\tilde{w}}^*(t)Z_2\dot{\tilde{w}}(t) - 4\gamma_4^2\left(\eta\tilde{w}(t)-\int_{t-\eta}^{t}\tilde{w}(s)ds\right)^* Z_2\left(\eta\tilde{w}(t)-\int_{t-\eta}^{t}\tilde{w}(s)ds\right)\right), \quad (4.1.28)
\end{aligned}
$$

where we used the conditions in (4.1.16) and Assumption 4.2 to obtain the inequalities in (4.1.20)–(4.1.28), Lemma 4.3 for the inequality in (4.1.23), and Lemma 4.4 for the inequality in (4.1.28).

Using Lemmas 4.6–4.7, we have that

$$
\begin{aligned}
-\eta\int_{t-\eta}^{t}\dot{\tilde{z}}^*(s)Y_1\dot{\tilde{z}}(s)ds &= -\eta\int_{t-\eta(t)}^{t}\dot{\tilde{z}}^*(s)Y_1\dot{\tilde{z}}(s)ds - \eta\int_{t-\eta}^{t-\eta(t)}\dot{\tilde{z}}^*(s)Y_1\dot{\tilde{z}}(s)ds\\
&\le -\frac{\eta}{\eta(t)}\begin{bmatrix}\tilde{z}(t)-\tilde{z}(t-\eta(t))\\ \tilde{z}(t)+\tilde{z}(t-\eta(t))-\frac{2}{\eta(t)}\int_{t-\eta(t)}^{t}\tilde{z}(s)ds\end{bmatrix}^*\begin{bmatrix}Y_1 & 0\\ 0 & 3Y_1\end{bmatrix}\\
&\quad\times\begin{bmatrix}\tilde{z}(t)-\tilde{z}(t-\eta(t))\\ \tilde{z}(t)+\tilde{z}(t-\eta(t))-\frac{2}{\eta(t)}\int_{t-\eta(t)}^{t}\tilde{z}(s)ds\end{bmatrix}\\
&\quad -\frac{\eta}{\eta-\eta(t)}\begin{bmatrix}\tilde{z}(t-\eta(t))-\tilde{z}(t-\eta)\\ \tilde{z}(t-\eta(t))+\tilde{z}(t-\eta)-\frac{2}{\eta-\eta(t)}\int_{t-\eta}^{t-\eta(t)}\tilde{z}(s)ds\end{bmatrix}^*\begin{bmatrix}Y_1 & 0\\ 0 & 3Y_1\end{bmatrix}\\
&\quad\times\begin{bmatrix}\tilde{z}(t-\eta(t))-\tilde{z}(t-\eta)\\ \tilde{z}(t-\eta(t))+\tilde{z}(t-\eta)-\frac{2}{\eta-\eta(t)}\int_{t-\eta}^{t-\eta(t)}\tilde{z}(s)ds\end{bmatrix}\\
&= -\xi^*(t)\left[\frac{\eta}{\eta(t)}\begin{bmatrix}e_1^n-e_6^n\\ e_1^n+e_6^n-2e_{15}^n\end{bmatrix}^*\begin{bmatrix}Y_1 & 0\\ 0 & 3Y_1\end{bmatrix}\begin{bmatrix}e_1^n-e_6^n\\ e_1^n+e_6^n-2e_{15}^n\end{bmatrix}\right.\\
&\quad\left.+\frac{\eta}{\eta-\eta(t)}\begin{bmatrix}e_6^n-e_5^n\\ e_6^n+e_5^n-2e_{16}^n\end{bmatrix}^*\begin{bmatrix}Y_1 & 0\\ 0 & 3Y_1\end{bmatrix}\begin{bmatrix}e_6^n-e_5^n\\ e_6^n+e_5^n-2e_{16}^n\end{bmatrix}\right]\xi(t)\\
&\le -\xi^*(t)\begin{bmatrix}e_1^n-e_6^n\\ e_1^n+e_6^n-2e_{15}^n\\ e_6^n-e_5^n\\ e_6^n+e_5^n-2e_{16}^n\end{bmatrix}^*\begin{bmatrix}\overline{Y}_1 & M_1\\ M_1^* & \overline{Y}_1\end{bmatrix}\begin{bmatrix}e_1^n-e_6^n\\ e_1^n+e_6^n-2e_{15}^n\\ e_6^n-e_5^n\\ e_6^n+e_5^n-2e_{16}^n\end{bmatrix}\xi(t),\\
&= -\xi^*(t)\zeta_1^*\Phi_1\zeta_1\xi(t),
\end{aligned}
$$

with the condition that $\Phi_1 = \begin{bmatrix}\overline{Y}_1 & M_1\\ M_1^* & \overline{Y}_1\end{bmatrix} > 0$, which is true by (4.1.17). Analogously, we can prove that

$$
-\eta\int_{t-\eta}^{t}\dot{\tilde{w}}^*(s)Y_2\dot{\tilde{w}}(s)ds \le -\xi^*(t)\zeta_2^*\Phi_2\zeta_2\xi(t),
$$

with the condition that $\Phi_2 > 0$, which is true by (4.1.17).

From Assumption 4.1 we deduce the existence of positive diagonal matrices G_1, G_2, …, G_8, so that

$$
0 \le \tilde{z}^*(t)L_{f_2}^*G_1L_{f_2}\tilde{z}(t) - \tilde{f}_2^*(\tilde{z}(t))G_1\tilde{f}_2(\tilde{z}(t)), \quad (4.1.29)
$$

$$0 \leq \tilde{z}^*(t-\tau(t))L_{f_2}^* G_2 L_{f_2}\tilde{z}(t-\tau(t)) - \tilde{f}_2^*(\tilde{z}(t-\tau(t)))G_2\tilde{f}_2(\tilde{z}(t-\tau(t))), \tag{4.1.30}$$

$$0 \leq \tilde{z}^*(t)L_{g_2}^* G_3 L_{g_2}\tilde{z}(t) - \tilde{g}_2^*(\tilde{z}(t))G_3\tilde{g}_2(\tilde{z}(t)), \tag{4.1.31}$$

$$0 \leq \tilde{z}^*(t-\tau(t))L_{g_2}^* G_4 L_{g_2}\tilde{z}(t-\tau(t)) - \tilde{g}_2^*(\tilde{z}(t-\tau(t)))G_4\tilde{g}_2(\tilde{z}(t-\tau(t))), \tag{4.1.32}$$

$$0 \leq \tilde{w}^*(t)L_{f_1}^* G_5 L_{f_1}\tilde{w}(t) - \tilde{f}_1^*(\tilde{w}(t))G_5\tilde{f}_1(\tilde{w}(t)), \tag{4.1.33}$$

$$0 \leq \tilde{w}^*(t-\tau(t))L_{f_1}^* G_6 L_{f_1}\tilde{w}(t-\tau(t)) - \tilde{f}_1^*(\tilde{w}(t-\tau(t)))G_6\tilde{f}_1(\tilde{w}(t-\tau(t))), \tag{4.1.34}$$

$$0 \leq \tilde{w}^*(t)L_{g_1}^* G_7 L_{g_1}\tilde{w}(t) - \tilde{g}_1^*(\tilde{w}(t))G_7\tilde{g}_1(\tilde{w}(t)), \tag{4.1.35}$$

$$0 \leq \tilde{w}^*(t-\tau(t))L_{g_1}^* G_8 L_{g_1}\tilde{w}(t-\tau(t)) - \tilde{g}_1^*(\tilde{w}(t-\tau(t)))G_8\tilde{g}_1(\tilde{w}(t-\tau(t))). \tag{4.1.36}$$

Also, for any matrices N_1, N_2, …, N_{10}, we have that

$$\begin{aligned}
0 \;=\;& \Big[\dot{\tilde{z}}^*(t)N_1 + \tilde{z}^*(t-\delta)N_2 - \tilde{f}_1^*(\tilde{w}(t))N_3 - \tilde{g}_1^*(\tilde{w}(t-\tau(t)))N_4 - \dot{\tilde{z}}^*(t-\eta(t))N_5\Big] \\
& \times \Big[-\dot{\tilde{z}}(t) - D_1\tilde{z}(t-\delta) + A_1\tilde{f}_1(\tilde{w}(t)) + B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) + E_1\dot{\tilde{z}}(t-\eta(t))\Big] \\
=\;& -\dot{\tilde{z}}^*(t)N_1\dot{\tilde{z}}(t) - \dot{\tilde{z}}^*(t)N_1D_1\tilde{z}(t-\delta) + \dot{\tilde{z}}^*(t)N_1A_1\tilde{f}_1(\tilde{w}(t)) + \dot{\tilde{z}}^*(t)N_1B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) \\
& + \dot{\tilde{z}}^*(t)N_1E_1\dot{\tilde{z}}(t-\eta(t))) - \tilde{z}^*(t-\delta)N_2\dot{\tilde{z}}(t) - \tilde{z}^*(t-\delta)N_2D_1\tilde{z}(t-\delta) + \tilde{z}^*(t-\delta)N_2A_1\tilde{f}_1(\tilde{w}(t)) \\
& + \tilde{z}^*(t-\delta)N_2B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) + \tilde{z}^*(t-\delta)N_2E_1\dot{\tilde{z}}(t-\eta(t))) + \tilde{f}_1^*(\tilde{w}(t))N_3\dot{\tilde{z}}(t) \\
& + \tilde{f}_1^*(\tilde{w}(t))N_3D_1\tilde{z}(t-\delta) - \tilde{f}_1^*(\tilde{w}(t))N_3A_1\tilde{f}_1(\tilde{w}(t)) - \tilde{f}_1^*(\tilde{w}(t))N_3B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) \\
& - \tilde{f}_1^*(\tilde{w}(t))N_3E_1\dot{\tilde{z}}(t-\eta(t)) + \tilde{g}_1^*(\tilde{w}(t-\tau(t)))N_4\dot{\tilde{z}}(t) + \tilde{g}_1^*(\tilde{w}(t-\tau(t)))N_4D_1\tilde{z}(t-\delta) \\
& - \tilde{g}_1^*(\tilde{w}(t-\tau(t)))N_4A_1\tilde{f}_1(\tilde{w}(t)) - \tilde{g}_1^*(\tilde{w}(t-\tau(t)))N_4B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) \\
& - \tilde{g}_1^*(\tilde{w}(t-\tau(t)))N_4E_1\dot{\tilde{z}}(t-\eta(t)) + \dot{\tilde{z}}^*(t-\eta(t))N_5\dot{\tilde{z}}(t) + \dot{\tilde{z}}^*(t-\eta(t))N_5D_1\tilde{z}(t-\delta) \\
& - \dot{\tilde{z}}^*(t-\eta(t))N_5A_1\tilde{f}_1(\tilde{w}(t)) - \dot{\tilde{z}}^*(t-\eta(t))N_5B_1\tilde{g}_1(\tilde{w}(t-\tau(t))) \\
& - \dot{\tilde{z}}^*(t-\eta(t))N_5E_1\dot{\tilde{z}}(t-\eta(t)),
\end{aligned} \tag{4.1.37}$$

$$\begin{aligned}
0 \;=\;& \Big[\dot{\tilde{w}}^*(t)N_6 + \tilde{w}^*(t-\delta)N_7 - \tilde{f}_2^*(\tilde{z}(t))N_8 - \tilde{g}_2^*(\tilde{z}(t-\tau(t)))N_9 - \dot{\tilde{w}}^*(t-\eta(t))N_{10}\Big] \\
& \times \Big[-\dot{\tilde{w}}(t) - D_2\tilde{w}(t-\delta) + A_2\tilde{f}_2(\tilde{z}(t)) + B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) + E_2\dot{\tilde{w}}(t-\eta(t))\Big] \\
=\;& -\dot{\tilde{w}}^*(t)N_6\dot{\tilde{w}}(t) - \dot{\tilde{w}}^*(t)N_6\tilde{w}(t-\delta) + \dot{\tilde{w}}^*(t)N_6A_2\tilde{f}_2(\tilde{z}(t)) + \dot{\tilde{w}}^*(t)N_6B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) \\
& + \dot{\tilde{w}}^*(t)N_6E_2\dot{\tilde{w}}(t-\eta(t)) - \tilde{w}^*(t-\delta)N_7\dot{\tilde{w}}(t) - \tilde{w}^*(t-\delta)N_7D_2\tilde{w}(t-\delta) + \tilde{w}^*(t-\delta)N_7A_2\tilde{f}_2(\tilde{z}(t)) \\
& + \tilde{w}^*(t-\delta)N_7B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) + \tilde{w}^*(t-\delta)N_7E_2\dot{\tilde{w}}(t-\eta(t)) + \tilde{f}_2^*(\tilde{z}(t))N_8\dot{\tilde{w}}(t) \\
& + \tilde{f}_2^*(\tilde{z}(t))N_8D_2\tilde{w}(t-\delta) - \tilde{f}_2^*(\tilde{z}(t))N_8A_2\tilde{f}_2(\tilde{z}(t)) - \tilde{f}_2^*(\tilde{z}(t))N_8B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) \\
& - \tilde{f}_2^*(\tilde{z}(t))N_8E_2\dot{\tilde{w}}(t-\eta(t)) + \tilde{g}_2^*(\tilde{z}(t-\tau(t)))N_9\dot{\tilde{w}}(t) + \tilde{g}_2^*(\tilde{z}(t-\tau(t)))N_9D_2\tilde{w}(t-\delta) \\
& - \tilde{g}_2^*(\tilde{z}(t-\tau(t)))N_9A_2\tilde{f}_2(\tilde{z}(t)) - \tilde{g}_2^*(\tilde{z}(t-\tau(t)))N_9B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) \\
& - \tilde{g}_2^*(\tilde{z}(t-\tau(t)))N_9E_2\dot{\tilde{w}}(t-\eta(t)) + \dot{\tilde{w}}^*(t-\eta(t))N_{10}\dot{\tilde{w}}(t) + \dot{\tilde{w}}^*(t-\eta(t))N_{10}D_2\tilde{w}(t-\delta) \\
& - \dot{\tilde{w}}^*(t-\eta(t))N_{10}A_2\tilde{f}_2(\tilde{z}(t)) - \dot{\tilde{w}}^*(t-\eta(t))N_{10}B_2\tilde{g}_2(\tilde{z}(t-\tau(t))) \\
& - \dot{\tilde{w}}^*(t-\eta(t))N_{10}E_2\dot{\tilde{w}}(t-\eta(t)).
\end{aligned} \tag{4.1.38}$$

Finally, by combining (4.1.20)–(4.1.28), (4.1.29)–(4.1.36) multiplied by $\mu^2(t) > 0$, and the conjugate of (4.1.37)–(4.1.38) added to the initial relations, multiplied by $\mu^2(t) > 0$, we obtain

$$\dot{V}(t) \leq \mu^2(t)\xi^*(t)\left[\Pi - \gamma_4^2\zeta_1^*\Phi_1\zeta_1 - \gamma_4^2\zeta_2^*\Phi_2\zeta_2\right]\xi(t),\ t \neq t_k,\ k \in \mathbb{Z}^+, \tag{4.1.39}$$

where

$$\begin{aligned}\xi(t) \;=\; & \big[\; \tilde{z}^*(t) \quad \dot{\tilde{z}}^*(t) \quad \dot{\tilde{z}}^*(t-\eta) \quad \dot{\tilde{z}}^*(t-\eta(t)) \quad \tilde{z}^*(t-\eta) \quad \tilde{z}^*(t-\eta(t)) \quad \tilde{z}^*(t-\delta) \quad \tilde{z}^*(t-\tau(t)) \\ & \tilde{f}_2^*(\tilde{z}(t)) \quad \tilde{f}_2^*(\tilde{z}(t-\tau(t))) \quad \tilde{g}_2^*(\tilde{z}(t)) \quad \tilde{g}_2^*(\tilde{z}(t-\tau(t))) \quad \int_{t-\delta}^{t} \tilde{z}^*(s)ds \quad \int_{t-\eta}^{t} \tilde{z}^*(s)ds \\ & \tfrac{1}{\eta(t)} \int_{t-\eta(t)}^{t} \tilde{z}^*(s)ds \quad \tfrac{1}{\eta-\eta(t)} \int_{t-\eta}^{t-\eta(t)} \tilde{z}^*(s)ds \\ & \tilde{w}^*(t) \quad \dot{\tilde{w}}^*(t) \quad \dot{\tilde{w}}^*(t-\eta) \quad \dot{\tilde{w}}^*(t-\eta(t)) \quad \tilde{w}^*(t-\eta) \quad \tilde{w}^*(t-\eta(t)) \quad \tilde{w}^*(t-\delta) \quad \tilde{w}^*(t-\tau(t)) \\ & \tilde{f}_1^*(\tilde{w}(t)) \quad \tilde{f}_1^*(\tilde{w}(t-\tau(t))) \quad \tilde{g}_1^*(\tilde{w}(t)) \quad \tilde{g}_1^*(\tilde{w}(t-\tau(t))) \quad \int_{t-\delta}^{t} \tilde{w}^*(s)ds \quad \int_{t-\eta}^{t} \tilde{w}^*(s)ds \\ & \tfrac{1}{\eta(t)} \int_{t-\eta(t)}^{t} \tilde{w}^*(s)ds \quad \tfrac{1}{\eta-\eta(t)} \int_{t-\eta}^{t-\eta(t)} \tilde{w}^*(s)ds \;\big]^*, \end{aligned}$$

and Π is the matrix defined in condition (4.1.17). From that condition we have that $\Pi - \gamma_4^2\zeta_1^*\Phi_1\zeta_1 - \gamma_4^2\zeta_2^*\Phi_2\zeta_2 < 0$, and thus inequality (4.1.39) yields

$$\dot{V}(t) < 0, \ \forall t \in [t_{k-1}, t_k) \cap [T, +\infty), \ k \in \mathbb{Z}^+, \ T \geq 0. \tag{4.1.40}$$

On the other hand, we have that

$$V_1(t_k) = \mu^2(t_k) \begin{bmatrix} \tilde{z}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ \tilde{w}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \end{bmatrix}^* \begin{bmatrix} P_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & P_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{z}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ \tilde{w}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \end{bmatrix},$$

$$V_2(t_k) = \mu^2(t_k) \begin{bmatrix} \tilde{z}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ \tilde{w}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \end{bmatrix}^* \begin{bmatrix} Q_1 & -Q_1D_1 & 0 & 0 \\ -D_1Q_1 & D_1Q_1D_1 & 0 & 0 \\ 0 & 0 & Q_2 & -Q_2D_2 \\ 0 & 0 & -D_2Q_2 & D_2Q_2D_2 \end{bmatrix} \begin{bmatrix} \tilde{z}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ \tilde{w}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \end{bmatrix},$$

which, together with Assumption 4.3 written in the form

$$\tilde{z}(t_k) = (I_n + F_k^1)\tilde{z}(t_k^-) - F_k^1 D_1 \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds,$$

$$\tilde{w}(t_k) = (I_m + F_k^2)\tilde{w}(t_k^-) - F_k^2 D_2 \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds,$$

yield

$$\begin{aligned} V_1(t_k) + V_2(t_k) \;=\; & \mu^2(t_k) \begin{bmatrix} \tilde{z}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ \tilde{w}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \end{bmatrix}^* \begin{bmatrix} P_1+Q_1 & -Q_1D_1 & 0 & 0 \\ -D_1Q_1 & D_1Q_1D_1 & 0 & 0 \\ 0 & 0 & P_2+Q_2 & -Q_2D_2 \\ 0 & 0 & -D_2Q_2 & D_2Q_2D_2 \end{bmatrix} \begin{bmatrix} \tilde{z}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ \tilde{w}(t_k) \\ \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \end{bmatrix} \\ =\; & \mu^2(t_k) \begin{bmatrix} (I_n + F_k^1)\tilde{z}(t_k^-) - F_k^1 D_1 \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ (I_m + F_k^2)\tilde{w}(t_k^-) - F_k^2 D_2 \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \\ \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \end{bmatrix}^* \\ & \times \begin{bmatrix} P_1+Q_1 & -Q_1D_1 & 0 & 0 \\ -D_1Q_1 & D_1Q_1D_1 & 0 & 0 \\ 0 & 0 & P_2+Q_2 & -Q_2D_2 \\ 0 & 0 & -D_2Q_2 & D_2Q_2D_2 \end{bmatrix} \begin{bmatrix} (I_n + F_k^1)\tilde{z}(t_k^-) - F_k^1 D_1 \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ \int_{t_k-\delta}^{t_k} \tilde{z}(s)ds \\ (I_m + F_k^2)\tilde{w}(t_k^-) - F_k^2 D_2 \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \\ \int_{t_k-\delta}^{t_k} \tilde{w}(s)ds \end{bmatrix} \end{aligned}$$

$$
\begin{aligned}
&= \mu^2(t_k)\begin{bmatrix} \tilde{z}(t_k^-) \\ \int_{t_k-\delta}^{t_k}\tilde{z}(s)ds \\ \tilde{w}(t_k^-) \\ \int_{t_k-\delta}^{t_k}\tilde{w}(s)ds \end{bmatrix}^* \begin{bmatrix} I_n+(F_k^1)^* & I_n & 0 & 0 \\ -D_1(F_k^1)^* & 0 & 0 & 0 \\ 0 & 0 & I_m+(F_k^2)^* & I_m \\ 0 & 0 & -D_2(F_k^2)^* & 0 \end{bmatrix} \\
&\quad \times \begin{bmatrix} P_1+Q_1 & -Q_1D_1 & 0 & 0 \\ -D_1Q_1 & D_1Q_1D_1 & 0 & 0 \\ 0 & 0 & P_2+Q_2 & -Q_2D_2 \\ 0 & 0 & -D_2Q_2 & D_2Q_2D_2 \end{bmatrix} \\
&\quad \times \begin{bmatrix} I_n+F_k^1 & -F_k^1D_1 & 0 & 0 \\ 0 & I_n & 0 & 0 \\ 0 & 0 & I_m+F_k^2 & -F_k^2D_2 \\ 0 & 0 & 0 & I_m \end{bmatrix} \begin{bmatrix} \tilde{z}(t_k^-) \\ \int_{t_k-\delta}^{t_k}\tilde{z}(s)ds \\ \tilde{w}(t_k^-) \\ \int_{t_k-\delta}^{t_k}\tilde{w}(s)ds \end{bmatrix} \\
&\leq \mu^2(t_k^-)\begin{bmatrix} \tilde{z}(t_k^-) \\ \int_{t_k-\delta}^{t_k}\tilde{z}(s)ds \\ \tilde{w}(t_k^-) \\ \int_{t_k-\delta}^{t_k}\tilde{w}(s)ds \end{bmatrix}^* \begin{bmatrix} P_1+Q_1 & -Q_1D_1 & 0 & 0 \\ -D_1Q_1 & D_1Q_1D_1 & 0 & 0 \\ 0 & 0 & P_2+Q_2 & -Q_2D_2 \\ 0 & 0 & -D_2Q_2 & D_2Q_2D_2 \end{bmatrix} \begin{bmatrix} \tilde{z}(t_k^-) \\ \int_{t_k-\delta}^{t_k}\tilde{z}(s)ds \\ \tilde{w}(t_k^-) \\ \int_{t_k-\delta}^{t_k}\tilde{w}(s)ds \end{bmatrix} \\
&= V_1(t_k^-)+V_2(t_k^-),
\end{aligned}
$$

where we used condition (4.1.18). Also, we observe that $V_3(t_k) = V_3(t_k^-)$, $V_4(t_k) = V_4(t_k^-)$, $V_5(t_k) = V_5(t_k^-)$, $V_6(t_k) = V_6(t_k^-)$, $V_7(t_k) = V_7(t_k^-)$, $V_8(t_k) = V_8(t_k^-)$, and $V_9(t_k) = V_9(t_k^-)$. Hence, we have that

$$V(t_k) \leq V(t_k^-),\ k \in \mathbb{Z}^+. \tag{4.1.41}$$

From (4.1.40) and (4.1.41), we deduce that $V(t)$ is strictly decreasing for $t \geq T$. This fact, together with the definition of $V(t)$, imply that

$$
\begin{aligned}
\mu^2(t)\lambda_{\min}(P)\left\|\begin{bmatrix}\tilde{z}(t)\\ \tilde{w}(t)\end{bmatrix}\right\|^2 &\leq \mu^2(t)\begin{bmatrix}\tilde{z}(t)\\ \tilde{w}(t)\end{bmatrix}^* P \begin{bmatrix}\tilde{z}(t)\\ \tilde{w}(t)\end{bmatrix} \\
&\leq V(t) \\
&\leq V_0,\ \forall t \geq T,
\end{aligned}
$$

where $V_0 = \max\limits_{0\leq t\leq T} V(t)$. Consequently, we have that

$$\left\|\begin{bmatrix}\tilde{z}(t)\\ \tilde{w}(t)\end{bmatrix}\right\|^2 \leq \frac{V_0}{\mu^2(t)\lambda_{\min}(P)},\ \forall t \geq 0,$$

or, equivalently,

$$\left\|\begin{bmatrix}\tilde{z}(t)\\ \tilde{w}(t)\end{bmatrix}\right\| \leq \frac{M}{\mu(t)},\ \forall t \geq 0,$$

for $M = \sqrt{\frac{V_0}{\lambda_{\min}(P)}}$. Now, Definition 4.1 implies that the origin of the system (4.1.15) is globally μ-stable, thus ending the proof of the theorem. □

Corollary 4.1. *If Assumptions 4.1 and 4.2–4.3 hold, and $\tau(t) \leq \tau$, then the origin of the system* (4.1.15) *is globally exponentially stable if there exist positive definite Hermitian matrices P_1, P_2, Q_1, Q_2, R_1, R_2, S_1, S_2, T_1, T_2, …, T_6, U_1, U_2, X_1, X_2, Y_1, Y_2, Z_1, Z_2, positive diagonal matrices G_1, G_2, …, G_8, and any matrices M_1, M_2, N_1, N_2, …, N_{10}, such that the conditions* (4.1.16)–(4.1.18) *hold, with $\gamma_1 = \varepsilon$, $\gamma_2 = e^{-\varepsilon\delta}$, $\gamma_3 = e^{-\varepsilon\tau}$, $\gamma_4 = e^{-\varepsilon\eta}$, for some positive ε.*

Proof. Let $\mu(t) = e^{\varepsilon t}$. Then, $\forall t \geq 0$, we have

$$\frac{\dot{\mu}(t)}{\mu(t)} = \varepsilon,$$

$$\frac{\inf_{s\in[t-\delta,t]} \mu(s)}{\mu(t)} = e^{-\varepsilon\delta},$$

$$\frac{\mu(t-\tau(t))}{\mu(t)} = e^{-\varepsilon\tau(t)} \geq e^{-\varepsilon\tau},$$

$$\begin{aligned}\frac{\min\{\inf_{s\in[t-\eta,t]} \mu(s),\ \mu(t-\eta(t))\}}{\mu(t)} &= \min\{e^{-\varepsilon\eta}, e^{-\varepsilon\eta(t)}\} \\ &\geq e^{-\varepsilon\eta}.\end{aligned}$$

Now, from Theorem 4.2 and Definition 4.2, we get that the origin of system (4.1.15) is globally exponentially stable. □

Corollary 4.2. *If Assumptions 4.1 and 4.2–4.3 hold, and $\tau(t) \leq \tau_d t$, then the origin of the system* (4.1.15) *is globally power stable if there exist positive definite Hermitian matrices P_1, P_2, Q_1, Q_2, R_1, R_2, S_1, S_2, T_1, T_2, …, T_6, U_1, U_2, X_1, X_2, Y_1, Y_2, Z_1, Z_2, positive diagonal matrices G_1, G_2, …, G_8, and any matrices M_1, M_2, N_1, N_2, …, N_{10}, such that the conditions* (4.1.16)–(4.1.18) *hold, with $\gamma_1 = \varepsilon$, $\gamma_2 = (1-\delta)^\varepsilon$, $\gamma_3 = (1-\tau_d)^\varepsilon$, $\gamma_4 = (1-\eta)^\varepsilon$, for some positive ε.*

Proof. Let $\mu(t) = t^\varepsilon$ for $t \geq 1$. Then, $\forall t \geq 1$, we have

$$\frac{\dot{\mu}(t)}{\mu(t)} = \frac{\varepsilon}{t} \leq \varepsilon,$$

$$\frac{\inf_{s\in[t-\delta,t]} \mu(s)}{\mu(t)} = \left(1-\frac{\delta}{t}\right)^\varepsilon \geq (1-\delta)^\varepsilon,$$

$$\frac{\mu(t-\tau(t))}{\mu(t)} = \left(1-\frac{\tau(t)}{t}\right)^\varepsilon \geq (1-\tau_d)^\varepsilon,$$

$$\frac{\min\{\inf_{s\in[t-\eta,t]} \mu(s),\ \mu(t-\eta(t))\}}{\mu(t)} = \min\left\{\left(1-\frac{\eta}{t}\right)^\varepsilon, \left(1-\frac{\eta(t)}{t}\right)^\varepsilon\right\} \geq (1-\eta)^\varepsilon.$$

From Theorem 4.2 and Definition 4.3, we infer that the origin of system (4.1.15) is globally power stable. □

Corollary 4.3. *If Assumptions 4.1 and 4.2–4.3 hold, and $\tau(t) \leq t - \frac{t}{\ln t}$, then the origin of the system* (4.1.15) *is globally log-stable if there exist positive definite Hermitian matrices P_1, P_2, Q_1, Q_2, R_1, R_2, S_1, S_2, T_1, T_2, …, T_6, U_1, U_2, X_1, X_2, Y_1, Y_2, Z_1, Z_2, positive diagonal matrices G_1, G_2, …, G_8, and any matrices M_1, M_2, N_1, N_2, …, N_{10}, such that the conditions* (4.1.16)–(4.1.18) *hold, with $\gamma_1 = \varepsilon$, $\gamma_2 = 1/2$, $\gamma_3 = 1/2$, $\gamma_4 = 1/2$, for some positive ε.*

Proof. Let $\mu(t) = \ln(\varepsilon t + 1)$. Then

$$\lim_{t\to\infty} \frac{\dot{\mu}(t)}{\mu(t)} = \lim_{t\to\infty} \frac{\varepsilon}{(\varepsilon t+1)\ln(\varepsilon t+1)} = 0,$$

$$\lim_{t\to\infty}\frac{\mu(t-\delta)}{\mu(t)}=\lim_{t\to\infty}\frac{\ln\left(\varepsilon(t-\delta)+1\right)}{\ln\left(\varepsilon t+1\right)}=1,$$

$$\lim_{t\to\infty}\frac{\mu(t-\tau(t))}{\mu(t)}=\lim_{t\to\infty}\frac{\ln\left(\varepsilon(t-\tau(t))+1\right)}{\ln\left(\varepsilon t+1\right)}\geq\lim_{t\to\infty}\frac{\ln\left(\frac{\varepsilon t}{\ln t}+1\right)}{\ln\left(\varepsilon t+1\right)}=1,$$

$$\lim_{t\to\infty}\frac{\mu(t-\eta)}{\mu(t)}=\lim_{t\to\infty}\frac{\ln\left(\varepsilon(t-\eta)+1\right)}{\ln\left(\varepsilon t+1\right)}=1,$$

which means that there exists $T>0$ such that

$$\frac{\dot{\mu}(t)}{\mu(t)}\leq\varepsilon,$$

$$\frac{\inf_{s\in[t-\delta,t]}\mu(s)}{\mu(t)}=\frac{\mu(t-\delta)}{\mu(t)}\geq\frac{1}{2},$$

$$\frac{\mu(t-\tau(t))}{\mu(t)}\geq\frac{1}{2},$$

$$\frac{\min\{\inf_{s\in[t-\eta,t]}\mu(s),\ \mu(t-\eta(t))\}}{\mu(t)}\geq\frac{\mu(t-\eta)}{\mu(t)}\geq\frac{1}{2},$$

$\forall t\geq T$. Theorem 4.2 and Definition 4.4 now give the global log-stability of the origin of system (4.1.15). □

Remark 4.1. In [225], the global asymptotic stability of complex-valued BAM neural networks with constant delays was studied. The existence, uniqueness, and global asymptotic stability analysis for delayed complex-valued Cohen–Grossberg BAM neural networks was discussed in [202]. Also, the existence, uniqueness, and exponential stability analysis for complex-valued memristor-based BAM neural networks with time delays were the focus of [68]. Delay-independent stability criteria for complex-valued BAM neutral-type neural networks with time delays were established in [232]. However, the present paper greatly extends these results by considering neutral-type impulsive complex-valued BAM neural networks with leakage delay and unbounded time-varying delays, and giving delay-dependent sufficient conditions for the global μ-stability of the equilibrium point of such networks.

Remark 4.2. The μ-stability of this type of networks with constant leakage delay, mixed delay, and impulses was analyzed in [39]. Papers [38, 219] give sufficient conditions for the μ-stability of the equilibrium point of complex-valued Hopfield neural networks with unbounded time-varying delays. Paper [62] discusses the same type of stability for complex-valued Hopfield neural networks with leakage delay and unbounded time-varying delays. Our results extend the ones in these papers by considering neutral-type impulsive complex-valued BAM neural networks with leakage delay and unbounded time-varying delays.

Remark 4.3. The real-valued Wirtiger-based integral inequality was used in [222] to study the dynamical properties of complex-valued memristive neural networks, which were split into their real and imaginary parts. Its complex-valued variant was given without proof in [113], and was used for the state estimation of complex-valued neural networks with two additive time-varying delays. In this paper, we provide a proof for the complex-valued Wirtiger-based integral inequality for the first time, and employ it to obtain sufficient stability criteria. Also, we provide a proof for the complex-valued reciprocally convex combination inequality, which, to the best of our knowledge, is used here for the first time in its complex-valued variant.

4.1.2 Numerical examples

In this section, we will prove the efficiency and correctness of the obtained results by a series of examples. The numerical simulations are done in MATLAB, and the LMIs are solved using the effective YALMIP tool.

Example 4.1. Consider the complex-valued bidirectional associative memories given by system (4.1.1), where

$$D_1 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0.1+0.1\imath & -0.2+0.1\imath \\ 0.1-0.1\imath & -0.1 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 0.1+0.1\imath & 0.1\imath \\ -0.1+0.1\imath & 0.2 \end{bmatrix}, \quad E_1 = \begin{bmatrix} 0.1-0.1\imath & 0 \\ 0 & 0.1+0.1\imath \end{bmatrix},$$

$$u_1 = \begin{bmatrix} 2-\imath \\ -2+\imath \end{bmatrix},$$

$$D_2 = \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0.1-\imath & 0.1+0.1\imath \\ 0.2+0.2\imath & 0.2-0.2\imath \end{bmatrix},$$

$$B_2 = \begin{bmatrix} 0.1+0.2\imath & 0.2+0.1\imath \\ 0.3-0.4\imath & -0.3+0.2\imath \end{bmatrix}, \quad E_2 = \begin{bmatrix} 0.2-\imath & 0 \\ 0 & 0.1 \end{bmatrix},$$

$$u_2 = \begin{bmatrix} 2-3\imath \\ -4+\imath \end{bmatrix},$$

$$f_i^1(w) = \frac{1}{1+e^{-w}}, \quad g_i^1(w) = \frac{1-e^{-2w}}{1+e^{-2w}}, \ \forall i = 1, 2,$$

$$f_j^2(z) = \frac{1}{1+e^{-z}}, \quad g_j^2(z) = \frac{1-e^{-2z}}{1+e^{-2z}}, \ \forall j = 1, 2,$$

from which we get that

$$L_{f_1} = L_{f_2} = \begin{bmatrix} 1/4 & 0 \\ 0 & 1/4 \end{bmatrix}, \quad L_{g_1} = L_{g_2} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}.$$

The leakage, time-varying, and neutral-type delays are $\delta = 0.05$, $\tau(t) = 0.3t$, $\eta(t) = 0.4$, so $\tau_d = 0.3$, $\eta = 0.4$, $\eta_d = 0$.

Now, by applying Theorem 4.1, we obtain that system (4.1.1) with the above given parameters has a unique equilibrium point if conditions (4.1.2) and (4.1.3) are satisfied. These conditions can be solved using MATLAB and YALMIP to get

$$P_1 = \begin{bmatrix} 0.4481 & -0.0002-0.0001\imath \\ -0.0002+0.0001\imath & 0.3212 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 0.2495 & 0.0003-0.0001\imath \\ 0.0003+0.0001\imath & 0.3185 \end{bmatrix},$$

$$G_1 = \text{diag}(1.0119, 1.0122), \quad G_2 = \text{diag}(1.0132, 1.0134),$$

$$G_3 = \text{diag}(1.0349, 1.0422), \quad G_4 = \text{diag}(1.0446, 1.0502).$$

We will take $\mu(t) = t$, and so $\gamma_1 = 1$, $\gamma_2 = 0.95$, $\gamma_3 = 0.7$, $\gamma_4 = 0.6$. Applying Theorem 4.2, the equilibrium point of the system (4.1.1) with the same parameters is globally power stable for (the values of the other matrices are not given for brevity reasons):

$$G_1 = \text{diag}(0.2126, 1.5381), \quad G_2 = \text{diag}(0.0204, 0.2485),$$

$$G_3 = \text{diag}(0.0270, 0.2808), \quad G_4 = \text{diag}(0.2575, 0.7659),$$
$$G_5 = \text{diag}(0.3509, 1.6191), \quad G_6 = \text{diag}(0.0130, 0.1347),$$
$$G_7 = \text{diag}(0.0182, 0.1719), \quad G_8 = \text{diag}(0.1362, 0.3814),$$

which satisfy conditions (4.1.16)–(4.1.18).

Example 4.2. Consider now the complex-valued BAM neural network given by system (4.1.1), where

$$D_1 = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}, \quad A_1 = \begin{bmatrix} -0.1 + 0.3\imath & -0.1 + 0.2\imath \\ -0.2 - 0.2\imath & 0.2 + 0.5\imath \end{bmatrix},$$
$$B_1 = \begin{bmatrix} 0.5 + 0.1\imath & 0.1\imath \\ -0.2 + 0.5\imath & 0.1 + 0.5\imath \end{bmatrix}, \; E_1 = \begin{bmatrix} 0.1 + 0.2\imath & 0 \\ 0 & 0.2 + 0.1\imath \end{bmatrix},$$
$$F_k^1 = \begin{bmatrix} -0.4 + 0.2\imath & -0.1 \\ -0.3 + 0.1\imath & -0.2 + 0.1\imath \end{bmatrix}, \; t_k = k, \; k \in \mathbb{Z}^+,$$
$$u_1 = \begin{bmatrix} 2 - 2\imath \\ -2 + 3\imath \end{bmatrix},$$
$$D_2 = \begin{bmatrix} 3 & 0 \\ 0 & 5 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -0.5 + 0.1\imath & -0.5 \\ -0.5 & -0.5 + 0.1\imath \end{bmatrix},$$
$$B_2 = \begin{bmatrix} 0.1 + 0.2\imath & 0.2 + 0.1\imath \\ 0.3 + 0.2\imath & 0.3 + 0.2\imath \end{bmatrix}, \; E_2 = \begin{bmatrix} 0.3 - 0.1\imath & 0 \\ 0 & -0.1 + 0.2\imath \end{bmatrix},$$
$$F_k^2 = \begin{bmatrix} -0.4 - 0.2\imath & -0.1 - 0.2\imath \\ -0.3 + 0.1\imath & -0.2 - 0.1\imath \end{bmatrix}, \; t_k = k, \; k \in \mathbb{Z}^+,$$
$$u_2 = \begin{bmatrix} -3 + 2\imath \\ 3 - \imath \end{bmatrix},$$
$$f_i^1(w) = \frac{1 - e^{-2w}}{1 + e^{-2w}}, \quad g_i^1(w) = \frac{1}{1 + e^{-w}}, \; \forall i = 1, 2,$$
$$f_j^2(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}}, \quad g_j^2(z) = \frac{1}{1 + e^{-z}}, \; \forall j = 1, 2,$$

from which we get that

$$L_{f_1} = L_{f_2} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}, \quad L_{g_1} = L_{g_2} = \begin{bmatrix} 1/4 & 0 \\ 0 & 1/4 \end{bmatrix}.$$

The leakage, neutral-type, and time-varying delays are $\delta = 0.02$, $\tau(t) = 0.5$, $\eta(t) = 0.3$, so $\tau_d = 0$, $\eta = 0.3$, $\eta_d = 0$. We will take $\mu(t) = e^t$, and so $\gamma_1 = 1$, $\gamma_2 = 0.9802$, $\gamma_3 = 1$, $\gamma_4 = 1$.

Applying Theorem 4.2, the equilibrium point of the system (4.1.1) with the above parameters is globally exponentially stable for (the values of the other matrices are not given for brevity reasons):

$$G_1 = \text{diag}(0.2159, 0.7085), \quad G_2 = \text{diag}(0.0082, 0.0888),$$
$$G_3 = \text{diag}(0.1934, 0.7821), \quad G_4 = \text{diag}(0.1297, 0.4889),$$
$$G_5 = \text{diag}(0.2885, 0.9510), \quad G_6 = \text{diag}(0.0087, 0.0606),$$
$$G_7 = \text{diag}(0.7641, 0.8243), \quad G_8 = \text{diag}(0.1962, 0.4878),$$

which satisfy conditions (4.1.16)–(4.1.18).

Chapter 5

Dynamics of quaternion-valued neural networks (QVNNs)

Quaternion-valued neural networks (QVNNs) are natural extensions of the extensively studied real-valued neural networks (RVNNs) and complex-valued neural networks (CVNNs) [79]. While two-dimensional data can be successfully operated by complex-valued neurons, QVNNs are expected to be more adequate to process multidimensional information (e.g., three-dimensional coordinates and color), by means of quaternionic neurons.

Quaternionic multilayer perceptrons have first been introduced by [4], proving their efficiency in quaternionic function interpolation and chaotic time series prediction [5]. Applications of QVNNs also include color image compression [89], color night vision [104] and 3D wind field modeling and forecasting [214]. Compared to RVNNs, [21] showed that QVNNs provide improved performance in polarized signal classification.

Up to this date, few authors have considered the theoretical investigation of dynamical properties of quaternion-valued neural networks, concentrating mainly on global behavior such as global stability.

5.1 Multistability and multiperiodicity in impulsive hybrid QVNNs with mixed delays

It is important to emphasize that in certain applications, such as associative memory storage for pattern recognition [90, 35], neural networks are required to exhibit configurations of several attractors (equilibrium states or periodic orbits), which store information and constitute distributed and parallel neural memory networks. While in mono-stability analysis, sufficient conditions are needed to guarantee the convergence of all trajectories to the unique attractor, in multistability analysis, the networks are allowed to have multiple attractors. With the aim of ensuring recall capability, the investigation of locally asymptotically stable equilibrium states or periodic orbits (existence, number, regions of attraction) is the main focus point in the case of multistable networks.

In the past decade, multistability and multiperiodicity in real-valued neural networks with delays have been extensively studied, showing the coexistence of 2^n attractors [85, 125, 41, 235, 50, 93]. Improved storage capacity guaranteed by a configuration of 3^n exponentially stable equilibrium states has been achieved by [127] and [126], using non-monotonic piecewise linear activation functions.

When it comes to complex-valued neural networks, significantly less multistability results

have been obtained compared to the case of RVNNs. [83] analyzed multistability and multiperiodicity properties in complex-valued neural networks with discrete time-delay and one step piecewise linear activation functions of real-imaginary type, showing the coexistence of 4^n locally exponentially stable equilibrium states/periodic orbits. Increased storage capacity has been achieved in the case of CVNNs without delays by [84]. Employing discontinuous activation functions, [112] obtained sufficient conditions for the coexistence of 9^n locally asymptotically stable equilibrium states in CVNNs with time-varying delays.

We investigate multistability and multiperiodicity properties of impulsive QVNNs with both time-dependent and distributed delays, significantly extending the results obtained by [95, 96, 97]. To the best of our knowledge, these are the first results concerning the coexistence of several attractors in quaternion-valued neural networks. A very general theoretical setting is considered: on one hand, impulsive effects are included to express instantaneous changes that naturally occur in electronic networks, caused by frequency changes, switching phenomena or noise [9]; on the other hand, time delays are incorporated to account for the lags in signal processing and transmission, and the finite switching speed of amplifiers. The dynamical properties of time-delayed systems are known to be much more complex than the behavior of non-delayed dynamical systems, time delays being frequently responsible for unstable or oscillatory behavior. Complementing time-varying delays, distributed delays reflect the whole past history of the variables, proving to be more realistic and more accurate in real world applications than discrete time delays [45].

The presentation in this section follows that in the author's paper [176].

5.1.1 Main results

consider the quaternion-valued Hopfield neural network given by the following system of differential equations

$$\begin{aligned}
\dot{x}_i(t) &= -d_i x_i(t) + \sum_{j=1}^{n} a_{ij} f_j(x_j(t)) + \sum_{j=1}^{n} b_{ij} f_j(x_j(t-\tau_{ij}(t))) \\
&\quad + \sum_{j=1}^{n} c_{ij} f_j\left(\int_0^{\infty} K_{ij}(s) x_j(t-s)ds\right) + u_i(t), \quad t \neq t_k,\ i = \overline{1,n}, \\
\boldsymbol{x}(t_k^-) &= \boldsymbol{x}(t_k), \qquad \boldsymbol{x}(t_k^+) = \boldsymbol{x}(t_k) + J_k(\boldsymbol{x}), \quad t = t_k,\ k \in \mathbb{Z}^+,
\end{aligned} \tag{5.1.1}$$

where $\boldsymbol{x}(t) = (x_1(t), \ldots, x_n(t))^T \in \mathbb{H}^n$ is the state vector at time t, $D = \text{diag}(d_1, d_2, \ldots, d_n) \in \mathbb{R}^{n\times n}$, $d_i > 0$, $\forall i = \overline{1,n}$, is the self-feedback connection weight matrix, $A = (a_{ij})_{1\leq i,j\leq n} \in \mathbb{H}^{n\times n}$ is the connection weight matrix, $B = (b_{ij})_{1\leq i,j\leq n} \in \mathbb{H}^{n\times n}$ is the time-varying delay connection weight matrix, $C = (c_{ij})_{1\leq i,j\leq n} \in \mathbb{H}^{n\times n}$ is the distributed delay connection weight matrix, $f_j : \mathbb{H} \to \mathbb{H}$ are the neuron activation functions, $\forall j = \overline{1,n}$, and $u_i : \mathbb{R} \to \mathbb{H}$ are the external inputs, which can be constants or ω-periodic functions. The sequence of times $\{t_k\}_{k\in\mathbb{Z}^+}$ satisfies $0 = t_0 < t_1 < t_2 < \cdots < \lim_{k\to\infty} t_k = \infty$.

Every quaternion $x \in \mathbb{H}$ can be written as

$$x = x^R + x^I \imath + x^J \jmath + x^K \kappa,$$

where x^R, x^I, x^J, $x^K \in \mathbb{R}$ are the real components of quaternion x, and $\imath$, $\jmath$, κ represent the quaternion imaginary units, which satisfy $\imath^2 = \jmath^2 = \kappa^2 = \imath\jmath\kappa = -1$. The product of two

quaternions $x, y \in \mathbb{H}$ is given by

$$\begin{aligned} xy &= x^R y^R - x^I y^I - x^J y^J - x^K y^K \\ &\quad +(x^R y^I + x^I y^R + x^J y^K - x^K y^J)\imath \\ &\quad +(x^R y^J - x^I y^K + x^J y^R + x^K y^I)\jmath \\ &\quad +(x^R y^K + x^I y^J - x^J y^I + x^K y^R)\kappa, \end{aligned}$$

from which we can see that quaternion multiplication is not commutative, but it can be verified that it is associative. The conjugate of quaternion $x \in \mathbb{H}$ is defined as

$$\overline{x} = x^R - x^I\imath - x^J\jmath - x^K\kappa,$$

and the norm as

$$|x| = \sqrt{x\overline{x}} = \sqrt{(x^R)^2 + (x^I)^2 + (x^J)^2 + (x^K)^2}.$$

We assume that the activation functions f_j can be written in the form

$$f_j(x) = f_j^R(x) + f_j^I(x)\imath + f_j^J(x)\jmath + f_j^K(x)\kappa, \ \forall x \in \mathbb{H},$$

where $f_j^R, f_j^I, f_j^J, f_j^K : \mathbb{H} \to \mathbb{R}, \forall j = \overline{1,n}$.

The initial conditions associated with (5.1.1) have the form

$$\boldsymbol{x}(s) = \boldsymbol{\phi}(s), \qquad s \in (-\infty, 0],$$

where the function $\boldsymbol{\phi} = (\phi_1(\cdot), \phi_2(\cdot), \ldots, \phi_n(\cdot))^T$ is piecewise continuous and bounded with respect to the norm

$$||\boldsymbol{\phi}||_\infty = \max_{i=\overline{1,n}} \left(\sup_{s\in(-\infty,0]} |\phi_i(s)| \right).$$

Denote $\mathcal{M} = \{R, I, J, K\}$. System (5.1.1) can be decomposed in the following way, $\forall P \in \mathcal{M}$,

$$\begin{aligned} \dot{x}_i^P(t) &= -d_i x_i^P(t) + \sum_{j=1}^n \left(a_{ij} f_j(x_j(t))\right)^P + \sum_{j=1}^n \left(b_{ij} f_j(x_j(t-\tau_{ij}(t)))\right)^P \\ &\quad + \sum_{j=1}^n \left(c_{ij} f_j \left(\int_0^\infty K_{ij}(s) x_j(t-s) ds \right) \right)^P + u_i^P(t), \ t \neq t_k, \ i = \overline{1,n}, \\ \boldsymbol{x}^P(t_k^-) &= \boldsymbol{x}^P(t_k), \qquad \boldsymbol{x}^P(t_k^+) = \boldsymbol{x}^P(t_k) + J_k^P(\boldsymbol{x}), \quad t = t_k, \ k \in \mathbb{Z}^+. \end{aligned} \tag{5.1.2}$$

In order to study the multistability and multiperiodicity of the above defined network, we need a series of assumptions.

Assumption 5.1. *The time-varying delays $\tau_{ij} : \mathbb{R} \to \mathbb{R}$ are continuously differentiable (ω-periodic) functions and there exist $\rho_{ij} > 0$ and $\rho'_{ij} \in [0,1)$, such that $\tau_{ij}(t) < \rho_{ij}$ and $\tau'_{ij}(t) \leq \rho'_{ij}$, $\forall t > 0$, $\forall i, j = \overline{1,n}$.*

Assumption 5.2. *The delay kernels $K_{ij} : [0,1) \to [0,1)$ are bounded, piecewise continuous, and satisfy*

$$\int_0^\infty K_{ij}(s) ds = 1 \text{ and } \exists \mu > 0 \text{ such that } \int_0^\infty K_{ij}(s) e^{\mu s} ds < \infty, \ \forall i, j = \overline{1,n}. \tag{5.1.3}$$

Assumption 5.3. *The jump operators $J_k = (J_{k1}, J_{k2}, \dots, J_{kn})^T$ are defined on the following set of functions: $\{\boldsymbol{u} : (-\infty, t_k] \to \mathbb{H}^n | \boldsymbol{u}$ is piecewise continuous on $(-\infty, 0]$, $\boldsymbol{u}$ is left continuous on $[0, t_k]$, with first kind discontinuity at t_l, and $\boldsymbol{u}$ is differentiable on every interval (t_{l-1}, t_l), $1 \le l \le k\}$. We will require that*

$$J_k(\boldsymbol{u}_0) \equiv \mathbf{0} \text{ for any constant function } \boldsymbol{u}_0. \tag{5.1.4}$$

For example, we may consider jump operators of the following forms (or their linear combinations):

(i) $J_k(\boldsymbol{x}) = \int_{t_{k-1}}^{t_k} B_k^1(s)\dot{\boldsymbol{x}}(s)ds$;

(ii) $J_k(\boldsymbol{x}) = \int_{t_{k-1}}^{t_k} B_k^2(s)[\boldsymbol{x}(s) - \boldsymbol{x}(t_k)]ds$;

(iii) $J_k(\boldsymbol{x}) = \sum_{l=0}^{k-1} B_{kl}^3[\boldsymbol{x}(t_l^+) - \boldsymbol{x}(t_l^-)]$,

where $B_k^1 : [t_{k-1}, t_k] \to \mathbb{H}^{n\times n}$ and $B_k^2 : [t_{k-1}, t_k] \to \mathbb{H}^{n\times n}$ are measurable and essentially bounded functions and $B_{kl}^3 \in \mathbb{H}^{n\times n}$.

Denoting $J(1) = (1, \infty)$, $J(-1) = (-\infty, -1)$, and $J_{\mathbb{H}}(x) = J(x^R) + J(x^I)\imath + J(x^J)\jmath + J(x^K)\kappa$, for every $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$, we define the rectangle

$$\Delta_\varepsilon = J_{\mathbb{H}}(\varepsilon_1) \times J_{\mathbb{H}}(\varepsilon_2) \times \cdots \times J_{\mathbb{H}}(\varepsilon_n).$$

For example, $J_{\mathbb{H}}(-1 + 1\imath - 1\jmath + 1\kappa) = (-\infty, -1) + (1, \infty)\imath + (-\infty, -1)\jmath + (1, \infty)\kappa$.

Assumption 5.4. *The components of the activation functions f_j^P are bounded: $|f_j^P(s)| \le 1$, $\forall s \in \mathbb{H}$, $\forall P \in \mathcal{M}$, $\forall j = \overline{1, n}$.*

Assumption 5.5. *There exists $\alpha \in (0, 1)$ such that the functions f_j^P satisfy $f_j^P(s) \ge \alpha$, if $s^P \ge 1$, and $f_j^P(s) \le -\alpha$, if $s^P \le -1$, $\forall s \in \mathbb{H}$, $\forall P \in \mathcal{M}$, $\forall j = \overline{1, n}$.*

Assumption 5.6. *For any $k \in \mathbb{Z}^+$ and $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$, if $\varphi(t) \in \Delta_\varepsilon$ for $t \le t_k$, then $\varphi(t_k) + J_k(\varphi) \in \Delta_\varepsilon$.*

Assumption 5.7. *The activation functions f_j^P are globally Lipschitz continuous. Moreover, β_j^P denotes the Lipschitz constant, i.e., $|f_j^P(u) - f_j^P(v)| \le \beta_j^P|u - v|$, $\forall u, v \in \mathbb{H}$, $\forall P \in \mathcal{M}$, $\forall j = \overline{1, n}$.*

Assumption 5.8. *With the notations from Assumption 5.7, the following inequalities are satisfied:*

$$\sum_{i=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left(\left|a_{ij}^Q\right| + \frac{\left|b_{ij}^Q\right|}{1 - \rho'_{ij}} + \left|c_{ij}^Q\right| \right) \beta_j^P < d_j, \ \forall j = \overline{1, n}.$$

Because $\rho'_{ij} \in [0, 1)$, we can also deduce that the following inequalities hold:

$$\sum_{i=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) \beta_j^P < d_j, \ \forall j = \overline{1, n}.$$

Assumption 5.9. *For any $k \in \mathbb{Z}^+$ and $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$, there exist $\gamma_k \geq 0$ and $\delta_k \geq 0$, such that if $\varphi(t)$, $\psi(t) \in \Delta_\varepsilon$ for $t \leq t_k$, we have*

$$\|J_k(\varphi) - J_k(\psi)\|_1 \leq \gamma_k \|\varphi(t_k) - \psi(t_k)\|_1 + \delta_k \int_{t_{k-1}}^{t_k} \|\varphi(s) - \psi(s)\|_1 \, ds.$$

In particular, if $\psi(t) = \boldsymbol{u}_0 \in \Delta_\varepsilon$, we have

$$\|J_k(\varphi)\|_1 \leq \gamma_k \|\varphi(t_k) - \boldsymbol{u}_0\|_1 + \delta_k \int_{t_{k-1}}^{t_k} \|\varphi(s) - \boldsymbol{u}_0\|_1 \, ds.$$

Assumption 5.10. *The following inequality is satisfied:*

$$\zeta = \sup_{k \in \mathbb{Z}_+^*} \frac{1}{t_k} \sum_{j=1}^{k} \ln\left(1 + \gamma_j + \delta_j \frac{e^{\mu_0(t_j - t_{j-1})} - 1}{\mu_0}\right) < \mu_0,$$

where $\mu_0 \in (0, \mu]$ is chosen such that the following inequality is satisfied:

$$\mu_0 - d_i + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left|a_{ji}^Q\right| \beta_i^P + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left|b_{ji}^Q\right| \beta_i^P \frac{e^{\mu_0 \rho_{ji}}}{1 - \rho'_{ji}}$$

$$+ \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left|c_{ji}^Q\right| \beta_i^P \left(\int_0^\infty K_{ji}(s) e^{\mu_0 s} ds\right) < 0, \ \forall i = \overline{1, n}. \tag{5.1.5}$$

This choice is possible due to the fact that the functions

$$F_i(u) = u - d_i + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left|a_{ji}^Q\right| \beta_i^P + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left|b_{ji}^Q\right| \beta_i^P \frac{e^{u \rho_{ji}}}{1 - \rho'_{ji}}$$

$$+ \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left|c_{ji}^Q\right| \beta_i^P \left(\int_0^\infty K_{ji}(s) e^{us} ds\right), \ \forall i = \overline{1, n}$$

are continuous and satisfy

$$F_i(0) = -d_i + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left(\left|a_{ji}^Q\right| + \frac{\left|b_{ji}^Q\right|}{1 - \rho'_{ji}} + \left|c_{ji}^Q\right|\right) \beta_i^P < 0,$$

based on Assumption 5.8.

In what follows, $PC([0, L], \mathbb{H}^n)$ denotes the space of left continuous functions defined on the interval $[0, L]$, with values in $\mathbb{H}^n$, having discontinuities of the first kind only at the points $\{t_k\}_{k \in \mathbb{Z}^+} \cap [0, L]$.

Assumption 5.11. *The jump operators $J_k = (J_{k1}, J_{k2}, \ldots, J_{kn})^T : PC([0, t_k], \mathbb{H}^n) \to \mathbb{H}^n$ are continuous.*

Assumption 5.12. *Considering $p > 0$ such that $[0, \omega] \cap \{t_k\}_{k \in \mathbb{Z}^+} = \{t_0, t_1, \ldots, t_p\}$, we assume that $t_{k+p} = t_k + \omega$ and $J_{k+p}(\boldsymbol{x}) = J_k(\boldsymbol{x})$, for any $k \in \mathbb{Z}^+$ and any ω-periodic function $\boldsymbol{x}$.*

Assumption 5.13. *The external inputs satisfy:*

$$\begin{aligned}\left|u_i^P(t)\right| \;<\; & \left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right)\alpha - d_i - \sum_{Q\in\mathcal{M}\setminus\{R\}}\left(\left|a_{ii}^Q\right| + \left|b_{ii}^Q\right| + \left|c_{ii}^Q\right|\right) \\ & - \sum_{j\neq i}\sum_{Q\in\mathcal{M}}\left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right|\right),\ \forall P\in\mathcal{M},\ \forall i=\overline{1,n},\ \forall t\in\mathbb{R}. \quad (5.1.6)\end{aligned}$$

Assumption 5.14. *For any $k\in\mathbb{Z}^+$, $P\in\mathcal{M}$, $\varepsilon\in\{\pm 1\pm 1\imath\pm 1\jmath\pm 1\kappa\}^n$, and $i=\overline{1,n}$, there exists $M_{ki}^P>0$ such that, if $\varphi(t)\in\Delta_\varepsilon$ for any $t\leq t_k$, then $0\leq\varepsilon_i^P J_{ki}^P(\varphi)\leq M_{ki}^P$.*

Lemma 5.1. *For any $x,\ y\in\mathbb{H}$, we have that*

$$\begin{aligned}\left|(xy)^R\right| &\leq \left|x^R\right|\left|y^R\right| + \left|x^I\right|\left|y^I\right| + \left|x^J\right|\left|y^J\right| + \left|x^K\right|\left|y^K\right|,\\ \left|(xy)^I\right| &\leq \left|x^R\right|\left|y^I\right| + \left|x^I\right|\left|y^R\right| + \left|x^J\right|\left|y^K\right| + \left|x^K\right|\left|y^J\right|,\\ \left|(xy)^J\right| &\leq \left|x^R\right|\left|y^J\right| + \left|x^I\right|\left|y^K\right| + \left|x^J\right|\left|y^R\right| + \left|x^K\right|\left|y^I\right|,\\ \left|(xy)^K\right| &\leq \left|x^R\right|\left|y^K\right| + \left|x^I\right|\left|y^J\right| + \left|x^J\right|\left|y^I\right| + \left|x^K\right|\left|y^R\right|.\end{aligned}$$

Moreover, if we denote $\mathcal{M}_R=\{(R,R),(I,I),(J,J),(K,K)\}$, $\mathcal{M}_I=\{(R,I),(I,R),(J,K),(K,J)\}$, $\mathcal{M}_J=\{(R,J),(I,K),(J,R),(K,I)\}$, $\mathcal{M}_K=\{(R,K),(I,J),(I,J),(K,R)\}$, the above inequalities can be written compactly as

$$\left|(xy)^P\right|\leq\sum_{(Q,S)\in\mathcal{M}_P}\left|x^Q\right|\left|y^S\right|,\ \forall P\in\mathcal{M}.$$

Proof. Using the product of quaternions,

$$\begin{aligned}xy \;=\; & x^Ry^R - x^Iy^I - x^Jy^J - x^Ky^K\\ & +(x^Ry^I + x^Iy^R + x^Jy^K - x^Ky^J)\imath\\ & +(x^Ry^J - x^Iy^K + x^Jy^R + x^Ky^I)\jmath\\ & +(x^Ry^K + x^Iy^J - x^Jy^I + x^Ky^R)\kappa,\end{aligned}$$

we can easily obtain the inequalities above.

Also, we can write

$$(xy)^P=\sum_{(Q,S)\in\mathcal{M}_P}\eta_{QS}x^Qy^S,\ \forall P\in\mathcal{M},$$

where $\eta_{QS}\in\{\pm 1\}$ is the sign of the product x^Qy^S, $\forall(Q,S)\in\mathcal{M}_P$. From this, we can deduce that

$$\left|(xy)^P\right|\leq\sum_{(Q,S)\in\mathcal{M}_P}\left|x^Q\right|\left|y^S\right|,\ \forall P\in\mathcal{M}.$$

□

5.1.1.1 Multistability analysis

A solution $\boldsymbol{x}:(-\infty,T)\to\mathbb{H}^n$ of (5.1.1) with the initial condition $\boldsymbol{x}(s)=\boldsymbol{\phi}(s)$, $s\in(-\infty,0]$, is piecewise continuous with first kind discontinuity at the points $t_k<T$, left continuous at each $t_k<T$, and differentiable on the open intervals $(t_k,t_{k+1})\subset(0,T)$.

In this subsection, the external inputs are considered to be constants. A steady state $x^u = (x_1^u, x_2^u, \ldots, x_n^u)^T \in \mathbb{H}^n$ of (5.1.1) corresponding to the input $u = (u_1, u_2, \ldots, u_n)^T$ satisfies

$$\begin{aligned} 0 &= -d_i x_i^u + \sum_{j=1}^{n} a_{ij} f_j(x_j^u) + \sum_{j=1}^{n} b_{ij} f_j(x_j^u) \\ &\quad + \sum_{j=1}^{n} c_{ij} f_j\left(x_j^u \int_0^\infty K_{ij}(s)ds\right) + u_i, \quad i = \overline{1,n}, \\ \boldsymbol{x}^u &= \boldsymbol{x}^u + J_k(\boldsymbol{x}^u), \quad k \in \mathbb{Z}^+. \end{aligned}$$

Taking into account the fact that the delay kernels K_{ij} satisfy (5.1.3) and the jump operators J_k satisfy (5.1.4), the above system is equivalent to

$$-d_i x_i^u + \sum_{j=1}^{n} a_{ij} f_j(x_j^u) + \sum_{j=1}^{n} b_{ij} f_j(x_j^u) + \sum_{j=1}^{n} c_{ij} f_j(x_j^u) + u_i = 0, \quad i = \overline{1,n},$$

which, in vector form, can be written as

$$-D\boldsymbol{x}^u + (A + B + C) f(\boldsymbol{x}^u) + u = 0, \tag{5.1.7}$$

where $f(\boldsymbol{x}) = (f_1(x_1), f_2(x_2), \ldots, f_n(x_n))^T$.

The following lemma provides a bound for the set of steady states of (5.1.1) corresponding to an input u:

Lemma 5.2. *If Assumption 5.4 is fulfilled, then, for any input vector $u \in \mathbb{H}^n$, the following statements hold:*

i. There exists at least one steady state of (5.1.1) *(corresponding to u) in the rectangle*

$$\Delta_u = \prod_{i=1}^{n} \left(\left[-M_i^R, M_i^R\right] + \left[-M_i^I, M_i^I\right] \imath + \left[-M_i^J, M_i^J\right] \jmath + \left[-M_i^K, M_i^K\right] \kappa \right)$$

of $\mathbb{H}^n$, where

$$M_i^P = \frac{1}{d_i} \left(\left|u_i^P\right| + \sum_{j=1}^{n} \sum_{Q \in \mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) \right), \ \forall P \in \mathcal{M}, \ \forall i = \overline{1,n}.$$

ii. Every steady state of (5.1.1)*, corresponding to u, belongs to the rectangle Δ_u, defined above.*

Proof. The set of steady states of (5.1.1) corresponding to u are given by equation (5.1.7), which is equivalent to

$$\boldsymbol{x} = D^{-1} \left(u + (A + B + C) f(\boldsymbol{x}) \right).$$

Define the function $h : \mathbb{H}^n \to \mathbb{H}^n$, $h(\boldsymbol{x}) = D^{-1} \left(u + (A + B + C) f(\boldsymbol{x}) \right)$. For any $\boldsymbol{x} \in \mathbb{H}^n$, $P \in \mathcal{M}$, and $i = \overline{1,n}$, based on Assumption 5.4, we have

$$\begin{aligned} \left|h_i^P(\boldsymbol{x})\right| &= \left| \frac{1}{d_i} \left(u_i^P + \sum_{j=1}^{n} \left(\left(a_{ij} f_j(x_j)\right)^P + \left(b_{ij} f_j(x_j)\right)^P + \left(c_{ij} f_j(x_j)\right)^P \right) \right) \right| \\ &\leq \frac{1}{d_i} \left(\left|u_i^P\right| + \sum_{j=1}^{n} \sum_{(Q,S) \in \mathcal{M}_P} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) \left|f_j^S(x_j)\right| \right) \\ &\leq \frac{1}{d_i} \left(\left|u_i^P\right| + \sum_{j=1}^{n} \sum_{Q \in \mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) \right) = M_i^P, \end{aligned}$$

where we used Lemma 5.1 for the inequalities. Therefore, $h(\mathbb{H}^n) \subset \Delta_u$, which proves *ii.* Moreover, one gets that $h(\Delta_u) \subset \Delta_u$, and, as h is a continuous function, Brouwer's fixed point theorem guarantees the existence of at least one steady state of (5.1.1) corresponding to u in Δ_u, so statement *i.* also holds. □

The main result concerning the existence of 16^n steady states for system (5.1.1) is presented in the following.

Theorem 5.1. *Suppose that Assumptions 5.4, 5.5, and 5.13 hold. Then, the following statements hold:*

i. In every rectangle Δ_ε, $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$, there exists at least one steady state of (5.1.1) *corresponding to input u.*

ii. If Assumption 5.6 is fulfilled, then every Δ_ε, $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$, is invariant to the flow of system (5.1.1)*.*

iii. Moreover, if Assumptions 5.7, 5.8, 5.9, and 5.10 hold as well, then the steady state of (5.1.1) *corresponding to the input u, lying in the rectangle Δ_ε, $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$, is unique, it is exponentially stable, and its region of attraction includes Δ_ε.*

Proof. Let u satisfy (5.1.6) and $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$.

i. Consider the function $h : \mathbb{H}^n \to \Delta_u$, defined by

$$h(\boldsymbol{x}) = D^{-1}\left(u + (A + B + C)f(\boldsymbol{x})\right),$$

and the rectangle Δ_u given by Lemma 5.2. For $\boldsymbol{x} \in \overline{\Delta_\varepsilon}$, we have that $\varepsilon_i^P x_i^P \geq 1$, $\forall P \in \mathcal{M}$, $\forall i = \overline{1,n}$, and therefore, based on Assumptions 5.4 and 5.5, we have

$$\begin{aligned}
\varepsilon_i^P h_i^P(\boldsymbol{x}) &= \frac{\varepsilon_i^P}{d_i}\left(\sum_{j=1}^{n}\left((a_{ij}+b_{ij}+c_{ij})f_j(x_j)\right)^P + u_i^P\right) \\
&= \frac{\varepsilon_i^P}{d_i}\left(\sum_{j=1}^{n}\sum_{(Q,S)\in\mathcal{M}_P}\eta_{QS}(a_{ij}^Q+b_{ij}^Q+c_{ij}^Q)f_j^S(x_j) + u_i^P\right) \\
&\geq \frac{1}{d_i}\left(\left(a_{ii}^R+b_{ii}^R+c_{ii}^R\right)\alpha - \sum_{Q\in\mathcal{M}\setminus\{R\}}\left(\left|a_{ii}^Q\right|+\left|b_{ii}^Q\right|+\left|c_{ii}^Q\right|\right)\right. \\
&\quad \left. - \sum_{j\neq i}\sum_{Q\in\mathcal{M}}\left(\left|a_{ij}^Q\right|+\left|b_{ij}^Q\right|+\left|c_{ij}^Q\right|\right) - \left|u_i^P\right|\right) \\
&> 1,
\end{aligned}$$

$\forall P \in \mathcal{M}$, $\forall i = \overline{1,n}$. This means that $h_i(\boldsymbol{x}) \in J_{\mathbb{H}}(\varepsilon_i)$ for any $i = \overline{1,n}$, and, therefore, $h_i(\boldsymbol{x}) \in \Delta_\varepsilon$. We have just proved that $h(\overline{\Delta_\varepsilon}) \subset \Delta_\varepsilon \cap \Delta_u$, and, based on Brouwer's fixed point theorem, we obtain the existence of at least one steady state of (5.1.1) corresponding to the input u in $\Delta_\varepsilon \cap \Delta_u$.

ii. Consider an initial function satisfying $\boldsymbol{\phi}(s) \in \Delta_\varepsilon$ for any $s \leq 0 = t_0$. Assumption 5.6 provides that $\boldsymbol{x}(t_0^+) \in \Delta_\varepsilon$, where $\boldsymbol{x}(t) = \boldsymbol{x}(t;\boldsymbol{\phi},u)$. Hence, there exists $\tau \in (t_0,t_1)$ such that $\boldsymbol{x}(t) \in \Delta_\varepsilon$, for any $t \in (t_0,\tau)$.

Assuming that $\boldsymbol{x}(\tau) \in \partial\Delta_\varepsilon$, there exists $i \in \{1,2,\ldots,n\}$ and $P \in \mathcal{M}$ such that $x_i^P(\tau) = \varepsilon_i^P$. Based on (5.1.6) and Assumptions 5.4 and 5.5, we have

$$\begin{aligned}
\varepsilon_i^P \dot{\boldsymbol{x}}_i^P(\tau) &= \varepsilon_i^P \left(-d_i \varepsilon_i^P + \sum_{j=1}^{n} \left(a_{ij} f_j(x_j(\tau))\right)^P + \sum_{j=1}^{n} \left(b_{ij} f_j(x_j(\tau-\tau_{ij}(\tau)))\right)^P \right. \\
&\quad \left. + \sum_{j=1}^{n} \left(c_{ij} f_j \left(\int_0^\infty K_{ij}(s) x_j(\tau - s) ds \right) \right)^P + u_i^P \right) \\
&\geq -d_i + \left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right) \alpha - \sum_{Q \in \mathcal{M} \setminus \{R\}} \left(\left|a_{ii}^Q\right| + \left|b_{ii}^Q\right| + \left|c_{ii}^Q\right| \right) \\
&\quad - \sum_{j \neq i} \sum_{Q \in \mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) - \left|u_i^P\right| \\
&> 0.
\end{aligned}$$

Therefore, the function $\varepsilon_i^P x_i^P$ is strictly increasing on some small interval $(\tau - \delta, \tau] \subset (t_0, \tau]$. Hence $\varepsilon_i^P x_i^P(t) < \varepsilon_i^P x_i^P(\tau) = \left(\varepsilon_i^P\right)^2 = 1$ for any $t \in (\tau - \delta, \tau)$. This is absurd, as $x_i(t) \in J_{\mathbb{H}}(\varepsilon_i)$ for any $t < \tau$.

It follows that $\boldsymbol{x}(t) \in \Delta_\varepsilon$ for any $t \in (t_0, t_1]$ and, by mathematical induction, it can be similarly shown that $\boldsymbol{x}(t) \in \Delta_\varepsilon$, for any $t \in (t_k, t_{k+1}]$, $k \in \mathbb{Z}^+$.

Hence, the solution $\boldsymbol{x}(t; \boldsymbol{\phi}, u)$, with the initial condition $\boldsymbol{\phi}(s) \in \Delta_\varepsilon$ for any $s \leq 0$, will remain in Δ_ε, for any $t \geq 0$.

iii. For the proof of uniqueness, suppose the contrary, i.e., there exist two steady states $\boldsymbol{x}$, $\boldsymbol{y} \in \Delta_\varepsilon$, $\boldsymbol{x} \neq \boldsymbol{y}$ of (5.1.1). Based on Assumption 5.7, for every $i = \overline{1, n}$ and every $P \in \mathcal{M}$, one has

$$\begin{aligned}
d_i \left| x_i^P - y_i^P \right| &= \left| \sum_{j=1}^{n} \left(a_{ij} (f_j(x_j) - f_j(y_j))\right)^P + \sum_{j=1}^{n} \left(b_{ij} (f_j(x_j) - f_j(y_j))\right)^P \right. \\
&\quad \left. + \sum_{j=1}^{n} \left(c_{ij} (f_j(x_j) - f_j(y_j))\right)^P \right| \\
&\leq \sum_{j=1}^{n} \left(\sum_{(Q,S) \in \mathcal{M}_P} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) \beta_j^S \left| x_j - y_j \right| \right).
\end{aligned}$$

Therefore, using Assumption 5.8, we get

$$\begin{aligned}
\sum_{i=1}^{n} d_i \left| x_i - y_i \right| &\leq \sum_{i=1}^{n} \sum_{P \in \mathcal{M}} d_i \left| x_i^P - y_i^P \right| \\
&\leq \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\sum_{Q \in \mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) \right) \left(\sum_{S \in \mathcal{M}} \beta_j^S \left| x_j - y_j \right| \right) \\
&= \sum_{j=1}^{n} \sum_{i=1}^{n} \sum_{S \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) \beta_j^S \left| x_j - y_j \right| \\
&< \sum_{j=1}^{n} d_j \left| x_j - y_j \right|,
\end{aligned}$$

which is absurd. Therefore, there exists a unique steady state of (5.1.1) corresponding to the input u, lying in the rectangle Δ_ε. It will be denoted by $\boldsymbol{x}^{u,\varepsilon}$.

Let us prove that $\boldsymbol{x}^{u,\varepsilon}$ is exponentially stable and its region of attraction includes Δ_ε. Consider the initial function $\boldsymbol{\phi}(s) \in \Delta_\varepsilon$, for any $s \leq 0$. From *ii.* we get that $\boldsymbol{x}(t;\boldsymbol{\phi},u) \in \Delta_\varepsilon$ for any $t \geq 0$. For $t > 0$, $t \neq t_k$, we have

$$
\begin{aligned}
D^+ \left|x_i(t) - x_i^{u,\varepsilon}\right| &= \sum_{P\in\mathcal{M}} \frac{x_i^P(t) - x_i^{u,\varepsilon,P}}{\left|x_i(t) - x_i^{u,\varepsilon}\right|} \dot{x}_i^P(t) \\
&= \sum_{P\in\mathcal{M}} \frac{x_i^P(t) - x_i^{u,\varepsilon,P}}{\left|x_i(t) - x_i^{u,\varepsilon}\right|} \left(-d_i x_i^P(t) + \sum_{j=1}^n \left(a_{ij} f_j(x_j(t))\right)^P \right. \\
&\quad \left. + \sum_{j=1}^n \left(b_{ij} f_j(x_j(t-\tau_{ij}(t)))\right)^P + \sum_{j=1}^n \left(c_{ij} f_j \left(\int_0^\infty K_{ij}(s) x_j(t-s) ds \right) \right)^P + u_i^P \right) \\
&= \sum_{P\in\mathcal{M}} \frac{x_i^P(t) - x_i^{u,\varepsilon,P}}{\left|x_i(t) - x_i^{u,\varepsilon}\right|} \left(-d_i \left(x_i^P(t) - x_i^{u,\varepsilon,P} \right) + \sum_{j=1}^n \left(a_{ij} \left(f_j(x_j(t)) - f_j(x_j^{u,\varepsilon}) \right) \right)^P \right. \\
&\quad + \sum_{j=1}^n \left(b_{ij} \left(f_j(x_j(t-\tau_{ij}(t))) - f_j(x_j^{u,\varepsilon}) \right) \right)^P \\
&\quad \left. + \sum_{j=1}^n \left(c_{ij} \left(f_j \left(\int_0^\infty K_{ij}(s) x_j(t-s) ds \right) - f_j(x_j^{u,\varepsilon}) \right) \right)^P \right) \\
&= -d_i \left|x_i(t) - x_i^{u,\varepsilon}\right| + \sum_{P\in\mathcal{M}} \frac{x_i^P(t) - x_i^{u,\varepsilon,P}}{\left|x_i(t) - x_i^{u,\varepsilon}\right|} \left(\sum_{j=1}^n \left(a_{ij} \left(f_j(x_j(t)) - f_j(x_j^{u,\varepsilon}) \right) \right)^P \right. \\
&\quad + \sum_{j=1}^n \left(b_{ij} \left(f_j(x_j(t-\tau_{ij}(t))) - f_j(x_j^{u,\varepsilon}) \right) \right)^P \\
&\quad \left. + \sum_{j=1}^n \left(c_{ij} \left(f_j \left(\int_0^\infty K_{ij}(s) x_j(t-s) ds \right) - f_j(x_j^{u,\varepsilon}) \right) \right)^P \right) \\
&\leq -d_i \left|x_i(t) - x_i^{u,\varepsilon}\right| + \sum_{P\in\mathcal{M}} \sum_{j=1}^n \sum_{(Q,S)\in\mathcal{M}_P} \left(\left|a_{ij}^Q\right| \beta_j^S \left|x_j(t) - x_j^{u,\varepsilon}\right| \right. \\
&\quad \left. + \left|b_{ij}^Q\right| \beta_j^S \left|x_j(t-\tau_{ij}(t)) - x_j^{u,\varepsilon}\right| + \left|c_{ij}^Q\right| \beta_j^S \int_0^\infty K_{ij}(s) \left|x_j(t-s) - x_i^{u,\varepsilon}\right| ds \right) \\
&= -d_i \left|x_i(t) - x_i^{u,\varepsilon}\right| + \sum_{j=1}^n \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left(\left|a_{ij}^Q\right| \beta_j^P \left|x_j(t) - x_j^{u,\varepsilon}\right| \right. \\
&\quad \left. + \left|b_{ij}^Q\right| \beta_j^P \left|x_j(t-\tau_{ij}(t)) - x_j^{u,\varepsilon}\right| + \left|c_{ij}^Q\right| \beta_j^P \int_0^\infty K_{ij}(s) \left|x_j(t-s) - x_i^{u,\varepsilon}\right| ds \right).
\end{aligned}
$$

Denoting $w_i(t) = e^{\mu_0 t} \left|x_i(t) - x_i^{u,\varepsilon}\right|$, for $t > 0$, $t \neq t_k$, we obtain

$$
\begin{aligned}
D^+ w_i(t) &= \mu_0 w_i(t) + e^{\mu_0 t} D^+ \left|x_i(t) - x_i^{u,\varepsilon}\right| \\
&\leq (\mu_0 - d_i) w_i(t) + \sum_{j=1}^n \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left(\left|a_{ij}^Q\right| \beta_j^P w_j(t) \right. \\
&\quad \left. + \left|b_{ij}^Q\right| \beta_j^P e^{\mu_0 \rho_{ij}} w_j(t-\tau_{ij}(t)) + \left|c_{ij}^Q\right| \beta_j^P \int_0^\infty K_{ij}(s) e^{\mu_0 s} w_j(t-s) ds \right).
\end{aligned}
$$

Denoting by θ_{ij} the inverse of the strictly increasing function $t - \tau_{ij}(t)$, we consider the function $V : \mathbb{R}_+ \to \mathbb{R}_+$ defined by

$$\begin{aligned} V(t) &= \sum_{i=1}^{n} (w_i(t) \\ &\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|b_{ij}^{Q}\right| \beta_j^P e^{\mu_0\rho_{ij}} \int_{t-\tau_{ij}(t)}^{t} \frac{w_j(s)}{1-\tau'_{ij}(\theta_{ij}(s))} ds \\ &\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|c_{ij}^{Q}\right| \beta_j^P \int_0^{\infty} K_{ij}(s) e^{\mu_0 s} \left(\int_{t-s}^{t} w_j(u) du\right) ds \Bigg). \end{aligned}$$

For $t > 0$, $t \neq t_k$, we obtain

$$\begin{aligned} D^+V(t) &= \sum_{i=1}^{n} \big(D^+ w_i(t) \\ &\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|b_{ij}^{Q}\right| \beta_j^P e^{\mu_0\rho_{ij}} \left(\frac{w_j(t)}{1-\tau'_{ij}(\theta_{ij}(t))} - \frac{(1-\tau'_{ij}(t)) w_j(t-\tau_{ij}(t))}{1-\tau'_{ij}(\theta_{ij}(t-\tau_{ij}(t)))}\right) \\ &\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|c_{ij}^{Q}\right| \beta_j^P \int_0^{\infty} K_{ij}(s) e^{\mu_0 s} (w_j(t) - w_j(t-s)) ds \Bigg) \\ &\leq \sum_{i=1}^{n} \Bigg((\mu_0 - d_i) w_i(t) + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|a_{ij}^{Q}\right| \beta_j^P w_j(t) \\ &\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|b_{ij}^{Q}\right| \beta_j^P \frac{e^{\mu_0\rho_{ij}}}{1-\tau'_{ij}(\theta_{ij}(t))} w_j(t) \\ &\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|c_{ij}^{Q}\right| \beta_j^P \left(\int_0^{\infty} K_{ij}(s) e^{\mu_0 s} ds\right) w_j(t) \Bigg) \\ &\leq \sum_{i=1}^{n} \Bigg(\mu_0 - d_i + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|a_{ji}^{Q}\right| \beta_i^P \\ &\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|b_{ji}^{Q}\right| \beta_i^P \frac{e^{\mu_0\rho_{ji}}}{1-\rho'_{ji}} \\ &\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|c_{ji}^{Q}\right| \beta_i^P \left(\int_0^{\infty} K_{ji}(s) e^{\mu_0 s} ds\right) \Bigg) w_i(t) \\ &< 0. \end{aligned}$$

Hence, the function V is strictly decreasing on every interval (t_k, t_{k+1}), and therefore $V(t) < V(t_k^+)$ for any $(t_k, t_{k+1}]$.

Moreover, from Assumption 5.9, and taking into consideration that $\sum_{i=1}^{n} w_i(t) \leq V(t)$ for any $t \in \mathbb{R}_+$, we have

$$
\begin{aligned}
\sum_{i=1}^{n} w_i(t_k^+) &= e^{\mu_0 t_k} \left\| \boldsymbol{x}(t_k^+) - \boldsymbol{x}^{u,\varepsilon} \right\|_1 \\
&= e^{\mu_0 t_k} \left\| \boldsymbol{x}(t_k) + J_k(\boldsymbol{x}) - \boldsymbol{x}^{u,\varepsilon} \right\|_1 \\
&\leq (1+\gamma_k) \left\| \boldsymbol{x}(t_k) - \boldsymbol{x}^{u,\varepsilon} \right\|_1 + \delta_k \int_{t_{k-1}}^{t_k} e^{\mu_0 t_k} \left\| \boldsymbol{x}(s) - \boldsymbol{x}^{u,\varepsilon} \right\|_1 ds \\
&= (1+\gamma_k) \sum_{i=1}^{n} w_i(t_k) + \delta_k \int_{t_{k-1}}^{t_k} e^{\mu_0 (t_k - s)} \sum_{i=1}^{n} w_i(s) ds \\
&\leq (1+\gamma_k) \sum_{i=1}^{n} w_i(t_k) + \delta_k \int_{t_{k-1}}^{t_k} e^{\mu_0 (t_k - s)} V(s) ds \\
&\leq (1+\gamma_k) \sum_{i=1}^{n} w_i(t_k) + \delta_k V(t_{k-1}^+) \int_{t_{k-1}}^{t_k} e^{\mu_0 (t_k - s)} ds \\
&= (1+\gamma_k) \sum_{i=1}^{n} w_i(t_k) + \delta_k \frac{e^{\mu_0 (t_k - t_{k-1})} - 1}{\mu_0} V(t_{k-1}^+),
\end{aligned}
$$

and hence

$$
\begin{aligned}
V(t_k^+) &= \sum_{i=1}^{n} \big(w_i(t_k^+) \\
&\quad + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left| b_{ij}^Q \right| \beta_j^P e^{\mu_0 \rho_{ij}} \int_{t_k^+ - \tau_{ij}(t_k^+)}^{t_k^+} \frac{w_j(s)}{1 - \tau_{ij}'(\theta_{ij}(s))} ds \\
&\quad + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left| c_{ij}^Q \right| \beta_j^P \int_0^{\infty} K_{ij}(s) e^{\mu_0 s} \left(\int_{t_k^+ - s}^{t_k^+} w_j(u) du \right) ds \Bigg) \\
&\leq (1+\gamma_k) \left(\sum_{i=1}^{n} [w_i(t_k) \right. \\
&\quad + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left| b_{ij}^Q \right| \beta_j^P e^{\mu_0 \rho_{ij}} \int_{t_k - \tau_{ij}(t_k)}^{t_k} \frac{w_j(s)}{1 - \tau_{ij}'(\theta_{ij}(s))} ds \\
&\quad \left. + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left| c_{ij}^Q \right| \beta_j^P \int_0^{\infty} K_{ij}(s) e^{\mu_0 s} \left(\int_{t_k - s}^{t_k} w_j(u) du \right) ds \right] \Bigg) \\
&\quad + \delta_k \frac{e^{\mu_0 (t_k - t_{k-1})} - 1}{\mu_0} V(t_{k-1}^+) \\
&= (1+\gamma_k) V(t_k) + \delta_k \frac{e^{\mu_0 (t_k - t_{k-1})} - 1}{\mu_0} V(t_{k-1}^+) \\
&< \left(1 + \gamma_k + \delta_k \frac{e^{\mu_0 (t_k - t_{k-1})} - 1}{\mu_0} \right) V(t_{k-1}^+).
\end{aligned}
$$

Using Assumption 5.10, we obtain

$$\begin{aligned} V(t_k^+) &< V(0^+)\prod_{j=1}^{k}\left(1+\gamma_j+\delta_j\frac{e^{\mu_0(t_j-t_{j-1})}-1}{\mu_0}\right) \\ &\leq (1+\gamma_0)\,V(0)\prod_{j=1}^{k}\left(1+\gamma_j+\delta_j\frac{e^{\mu_0(t_j-t_{j-1})}-1}{\mu_0}\right) \\ &\leq (1+\gamma_0)\,V(0)e^{\varsigma t_k}. \end{aligned}$$

Moreover, we have

$$\begin{aligned} V(0) &= \sum_{i=1}^{n}\left(|x_i(0)-x_i^{u,\varepsilon}|\right. \\ &\quad +\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\left|b_{ij}^{Q}\right|\beta_j^{P}e^{\mu_0\rho_{ij}}\int_{-\tau_{ij}(0)}^{0}\frac{e^{\mu_0 s}\,|x_i(s)-x_i^{u,\varepsilon}|}{1-\tau_{ij}'(\theta_{ij}(s))}ds \\ &\quad \left.+\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\left|c_{ij}^{Q}\right|\beta_j^{P}\int_0^{\infty}K_{ij}(s)e^{\mu_0 s}\left(\int_{-s}^{0}e^{\mu_0 u}\,|x_i(u)-x_i^{u,\varepsilon}|\,du\right)ds\right) \\ &\leq \|\boldsymbol{\phi}-\boldsymbol{x}^{u,\varepsilon}\|_\infty\left[n+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\beta_j^{P}\left(\left|b_{ij}^{Q}\right|e^{\mu_0\rho_{ij}}\int_{-\tau_{ij}(0)}^{0}\frac{e^{\mu_0 s}}{1-\tau_{ij}'(\theta_{ij}(s))}ds+\right.\right. \\ &\quad \left.\left.+\left|c_{ij}^{Q}\right|\frac{1}{\mu_0}\left(\int_0^{\infty}K_{ij}(s)e^{\mu_0 s}ds-1\right)\right)\right] \\ &= \|\boldsymbol{\phi}-\boldsymbol{x}^{u,\varepsilon}\|_\infty\left[n+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\beta_j^{P}\left(\left|b_{ij}^{Q}\right|e^{\mu_0\rho_{ij}}\int_{0}^{\theta_{ij}(0)}e^{\mu_0(u-\tau_{ij}(u))}du+\right.\right. \\ &\quad \left.\left.+\left|c_{ij}^{Q}\right|\frac{1}{\mu_0}\left(\int_0^{\infty}K_{ij}(s)e^{\mu_0 s}ds-1\right)\right)\right]. \end{aligned}$$

For any $t\in(t_k,t_{k+1}]$, we obtain

$$V(t)<V(t_k^+)<\Lambda\,\|\boldsymbol{\phi}-\boldsymbol{x}^{u,\varepsilon}\|_\infty\,e^{\varsigma t_k}<\Lambda\,\|\boldsymbol{\phi}-\boldsymbol{x}^{u,\varepsilon}\|_\infty\,e^{\varsigma t},$$

where

$$\begin{aligned} \Lambda &= (1+\gamma_0)\left[n+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\beta_j^{P}\left(\left|b_{ij}^{Q}\right|e^{\mu_0\rho_{ij}}\int_{0}^{\theta_{ij}(0)}e^{\mu_0(u-\tau_{ij}(u))}du+\right.\right. \\ &\quad \left.\left.+\left|c_{ij}^{Q}\right|\frac{1}{\mu_0}\left(\int_0^{\infty}K_{ij}(s)e^{\mu_0 s}ds-1\right)\right)\right]. \end{aligned}$$

Finally, it follows that

$$|x_i(t)-x_i^{u,\varepsilon}|=e^{-\mu_0 t}w_i(t)<e^{-\mu_0 t}V(t)<\Lambda\,\|\boldsymbol{\phi}-\boldsymbol{x}^{u,\varepsilon}\|_\infty\,e^{(\varsigma-\mu_0)t},$$

$\forall t>0,\ \forall i=\overline{1,n}$, which, taking into account that $\varsigma<\mu_0$ (by Assumption 5.10), means that $\boldsymbol{x}(t;\boldsymbol{\phi},u)$ converges exponentially to $\boldsymbol{x}^{u,\varepsilon}$. □

5.1.1.2 Multiperiodicity analysis

In this subsection, the external inputs are considered to be ω-periodic functions.

Lemma 5.3. *The function $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))^T$ is an ω-periodic solution of* (5.1.1) *if and only if it is an ω-periodic solution of*

$$x_i(t) = \int_0^\omega G_i\left(t - \left[\frac{t}{\omega}\right]\omega, s\right) H_i(\boldsymbol{x})(s)ds + \sum_{k=0}^{p} G_i\left(t - \left[\frac{t}{\omega}\right]\omega, t_k\right) J_{ki}(x),$$

for any $i = \overline{1,n}$, where

$$H_i(\boldsymbol{x})(s) = \sum_{j=1}^{n} a_{ij} f_j(x_j(s)) + \sum_{j=1}^{n} b_{ij} f_j(x_j(s-\tau_{ij}(s))) + \sum_{j=1}^{n} c_{ij} f_j\left(\int_0^\infty K_{ij}(u)x_j(s-u)du\right) + u_i(s),$$

$$G_i(u,s) = \frac{1}{e^{d_i\omega} - 1}\begin{cases} e^{d_i(\omega+s-u)}, & \text{if } 0 \le s < u \le \omega \\ e^{d_i(s-u)}, & \text{if } 0 \le u \le s \le \omega \end{cases},$$

and $[\cdot]$ denotes the integer part.

Consider the Banach space of piecewise continuous ω-periodic functions

$$X = \{\boldsymbol{x} : \mathbb{R} \to \mathbb{H} \mid \boldsymbol{x} \in PC([0,\omega], \mathbb{H}^n),\ \boldsymbol{x}(t+\omega) = \boldsymbol{x}(t),\ \forall t \in \mathbb{R}\},$$

endowed with the norm

$$\|\boldsymbol{x}\| = \sum_{i=1}^{n} \sup_{t\in[0,\omega]} |x_i(t)|,$$

and define the operator $U : X \to X$, given by $U(\boldsymbol{x}) = (U_1(\boldsymbol{x}), U_2(\boldsymbol{x}), \ldots, U_n(\boldsymbol{x}))^T$, where

$$U_i(\boldsymbol{x})(t) = \int_0^\omega G_i\left(t - \left[\frac{t}{\omega}\right]\omega, s\right) H_i(\boldsymbol{x})(s)ds + \sum_{k=0}^{p} G_i\left(t - \left[\frac{t}{\omega}\right]\omega, t_k\right) J_{ki}(x),\ \forall i = 1, n.$$

It follows from Lemma 5.3 that $\boldsymbol{x}(t)$ is an ω-periodic solution of (5.1.1) if and only if it is a fixed point of the operator U.

For every $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$, we define the closed convex set

$$K_\varepsilon = \{\boldsymbol{x} \in X \mid \varepsilon_i^P x_i^P \ge 1,\ \forall t \in \mathbb{R},\ \forall P \in \mathcal{M},\ \forall i = \overline{1,n}\}.$$

The main result of this subsection will be obtained using the following fixed point theorem:

Theorem 5.2 (Leray–Schauder)**.** *Let X be a Banach space, $K \subset X$ a closed convex subset, $B \subset K$ a bounded set, open in K, and $x_0 \in K$ a fixed element. Assume that the operator $U : \overline{B} \to K$ is completely continuous and satisfies the boundary condition $x \ne (1-\lambda)x_0 + \lambda U(x)$, $\forall x \in \partial B$, $\lambda \in (0,1)$. Then, the operator U has at least one fixed point in $\overline{B}$.*

Moreover, the following compactness criterion will be used (based on Lemma 1.6 from "D. Bainov, P. Simeonov, Impulsive Differential Equations, World Scientific, Singapore, 1995.").

Lemma 5.4 (Compactness criterion)**.** *The set $\mathcal{F} \subset X$ is relatively compact if and only if the following hold:*

1. *$\mathcal{F}$ is bounded, that is, there exists $c > 0$ such that $\|\boldsymbol{x}\| \le c$, for any $\boldsymbol{x} \in \mathcal{F}$;*

2. *$\mathcal{F}$ is quasi-equicontinuous in $[0,\omega]$, i.e., for any $\epsilon > 0$ there exists $\delta > 0$, such that for any $\boldsymbol{x} \in \mathcal{F}$, $l \in \mathbb{Z}^+$, T_1, $T_2 \in (t_{l-1}, t_l] \cap [0,\omega]$, such that $|T_1 - T_2| < \delta$, one has $\|\boldsymbol{x}(T_1) - \boldsymbol{x}(T_2)\|_1 < \epsilon$.*

Lemma 5.5. *Let $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$. If Assumptions 5.2, 5.7, and 5.11 hold, the operator U is continuous on K_ε.*

Proof. Let $\boldsymbol{x}, \boldsymbol{y} \in K_\varepsilon$. Using Assumptions 5.7 and 5.2, and Lemma 5.1, we first evaluate:

$$\begin{aligned}
\left|H_i^P(\boldsymbol{x})(s) - H_i^P(\boldsymbol{y})(s)\right| &\leq \sum_{j=1}^{n} \sum_{(Q,S)\in\mathcal{M}_P} \left|a_{ij}^Q\right| \left|f_j^S(x_j(s)) - f_j^S(y_j(s))\right| \\
&\quad + \sum_{j=1}^{n} \sum_{(Q,S)\in\mathcal{M}_P} \left|b_{ij}^Q\right| \left|f_j^S(x_j(s-\tau_{ij}(s))) - f_j^S(y_j(s-\tau_{ij}(s)))\right| \\
&\quad + \sum_{j=1}^{n} \sum_{(Q,S)\in\mathcal{M}_P} \left|c_{ij}^Q\right| \left| f_j^S\left(\int_0^\infty K_{ij}(u) x_j(s-u)du\right) \right. \\
&\quad \left. - f_j^S\left(\int_0^\infty K_{ij}(u) y_j(s-u)du\right)\right| \\
&\leq \sum_{j=1}^{n} \sum_{(Q,S)\in\mathcal{M}_P} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right|\right) \beta_j^S \left\|\boldsymbol{x} - \boldsymbol{y}\right\|, \ \forall P \in \mathcal{M},
\end{aligned}$$

from which we get that

$$|H_i(\boldsymbol{x})(s) - H_i(\boldsymbol{y})(s)| \leq \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right|\right) \beta_j^P \left\|\boldsymbol{x} - \boldsymbol{y}\right\|.$$

The function $G_i(u,s)$ is positive and it can be easily seen that $\int_0^\omega G_i(u,s)ds = \frac{1}{d_i}$ and $G_i(u,s) \leq \frac{e^{d_i\omega}}{e^{d_i\omega}-1}$, for any u, $s \in [0,\omega]$. Therefore, for any $t \in [0,\omega)$, we obtain

$$\begin{aligned}
|U_i(\boldsymbol{x})(t) - U_i(\boldsymbol{y})(t)| &\leq \int_0^\omega G_i(t,s) \left|H_i(\boldsymbol{x})(s) - H_i(\boldsymbol{y})(s)\right| ds + \sum_{k=0}^{p} G_i(t,t_k) \left|J_{ki}(\boldsymbol{x}) - J_{ki}(\boldsymbol{y})\right| \\
&\leq \frac{1}{d_i} \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right|\right) \beta_j^P \left\|\boldsymbol{x} - \boldsymbol{y}\right\| \\
&\quad + \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} \left|J_{ki}(\boldsymbol{x}) - J_{ki}(\boldsymbol{y})\right|,
\end{aligned}$$

and hence

$$\begin{aligned}
\left\|U(\boldsymbol{x}) - U(\boldsymbol{y})\right\| &\leq \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \frac{1}{d_i} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right|\right) \beta_j^P \left\|\boldsymbol{x} - \boldsymbol{y}\right\| \\
&\quad + \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} \left\|J_k(\boldsymbol{x}) - J_k(\boldsymbol{y})\right\|_1.
\end{aligned}$$

Based on the continuity of the operators J_k given by Assumption 5.11, we obtain that the operator U is continuous on K_ε. □

Lemma 5.6. *If Assumptions 5.4, 5.5, 5.13, and 5.14 hold, for every* $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$, *the operator* U *maps* K_ε *into itself, i.e.,* $U(K_\varepsilon) \subset K_\varepsilon$.

Proof. Let $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$ and $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))^T \in K_\varepsilon$. Based on Assumptions 5.4, 5.5, and 5.13, we have, $\forall P \in \mathcal{M}$, that

$$\begin{aligned}
\varepsilon_i^P H_i^P(\boldsymbol{x})(s) &= \varepsilon_i^P \left(\sum_{j=1}^n \left(a_{ij} f_j(x_j(s))\right)^P + \sum_{j=1}^n \left(b_{ij} f_j(x_j(s-\tau_{ij}(s)))\right)^P \right. \\
&\quad \left. + \sum_{j=1}^n \left(c_{ij} f_j \left(\int_0^\infty K_{ij}(u) x_j(s-u) du \right) \right)^P + u_i^P(s) \right) \\
&= \varepsilon_i^P \left(\sum_{j=1}^n \sum_{(Q,S)\in\mathcal{M}_P} \eta_{QS} \left(a_{ij}^Q f_j^S(x_j(s)) + b_{ij}^Q f_j^S(x_j(s-\tau_{ij}(s))) \right. \right. \\
&\quad \left. \left. + c_{ij}^Q f_j^S \left(\int_0^\infty K_{ij}(u) x_j(s-u) du \right) \right) + u_i^P(s) \right) \\
&\geq \left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right) \alpha - \sum_{Q\in\mathcal{M}\setminus\{R\}} \left(\left|a_{ii}^Q\right| + \left|b_{ii}^Q\right| + \left|c_{ii}^Q\right| \right) \\
&\quad - \sum_{i\neq j} \sum_{Q\in\mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) - \left|u_i^P(s)\right| \\
&\geq d_i.
\end{aligned}$$

Using Assumption 5.14 as well, it follows that

$$\varepsilon_i^P U_i^P(\boldsymbol{x})(t) = \int_0^\omega G_i \left(t - \left[\frac{t}{\omega}\right]\omega, s \right) \varepsilon_i^P H_i^P(\boldsymbol{x})(s) ds + \sum_{k=0}^p G_i \left(t - \left[\frac{t}{\omega}\right]\omega, t_k \right) \varepsilon_i^P J_{ki}^P(x) \geq d_i \frac{1}{d_i} = 1.$$

Hence, $U(K_\varepsilon) \subset K_\varepsilon$. □

Lemma 5.7. *Suppose that Assumptions 5.4, 5.5, 5.13, and 5.14 hold. Let* $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$ *and* $\boldsymbol{x}_\varepsilon \in K_\varepsilon$, *the constant function defined by* $\boldsymbol{x}_\varepsilon(t) = \varepsilon$, *for any* $t \in \mathbb{R}$. *If there exists* $\boldsymbol{x} \in K_\varepsilon$ *and* $\lambda \in (0,1)$ *such that*

$$\boldsymbol{x} = (1-\lambda)\boldsymbol{x}_\varepsilon + \lambda U(\boldsymbol{x}), \tag{5.1.8}$$

then $\|\boldsymbol{x} - \boldsymbol{x}_\varepsilon\| < R$, *where*

$$R = \sum_{i=1}^n \sum_{P\in\mathcal{M}} \left[\frac{(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right)}{d_i} - 2 + \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^p M_{ki}^P \right].$$

Proof. Assume that $\boldsymbol{x} \in K_\varepsilon$ and $\lambda \in (0,1)$ satisfy equation (5.1.8). Hence

$$x_i^P(t) - \varepsilon_i^P = \lambda \left[U_i^P(\boldsymbol{x})(t) - \varepsilon_i^P \right], \ \forall t \in [0,\omega), \ \forall P \in \mathcal{M}, \ \forall i = \overline{1,n}.$$

Since $\boldsymbol{x} \in K_\varepsilon$, it follows that $\varepsilon_i^P x_i^P(t) \geq 1$, for any $t \in [0,\omega)$, and so, $\varepsilon_i^P \left(x_i^P(t) - \varepsilon_i^P \right) \geq 0$, for any $t \in [0,\omega)$. Therefore, for any $t \in [0,\omega)$, $P \in \mathcal{M}$, and $i = \overline{1,n}$, we obtain:

$$\left|x_i^P(t) - \varepsilon_i^P\right| = \left|\varepsilon_i^P \left(x_i^P(t) - \varepsilon_i^P \right)\right| = \varepsilon_i^P \left(x_i^P(t) - \varepsilon_i^P \right) = \varepsilon_i^P \lambda \left[U_i^P(\boldsymbol{x})(t) - \varepsilon_i^P \right] = \lambda \left[\varepsilon_i^P U_i^P(\boldsymbol{x})(t) - 1 \right].$$

Based on Assumptions 5.4, 5.5, and 5.13, we evaluate

$$\begin{aligned}
\varepsilon_i^P H_i^P(\boldsymbol{x})(s) &= \varepsilon_i^P \sum_{j=1}^{n} \left(a_{ij} f_j(x_j(s))\right)^P + \varepsilon_i^P \sum_{j=1}^{n} \left(b_{ij} f_j(x_j(s-\tau_{ij}(s)))\right)^P \\
&\quad + \varepsilon_i^P \sum_{j=1}^{n} \left(c_{ij} f_j\left(\int_0^\infty K_{ij}(u) x_j(s-u)du\right)\right)^P + \varepsilon_i^P u_i^P(s) \\
&\leq \sum_{j=1}^{n} \sum_{Q\in\mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right|\right) + \left|u_i^P(s)\right| \\
&\leq (1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right) - d_i.
\end{aligned}$$

Using Assumption 5.14 as well, for any $t \in [0,\omega)$, we obtain

$$\begin{aligned}
\varepsilon_i^P U_i^P(\boldsymbol{x})(t) &= \int_0^\omega G_i(t,s)\varepsilon_i^P H_i^P(\boldsymbol{x})(s)ds + \sum_{k=0}^{p} G_i(t,t_k)\varepsilon_i^P J_{ki}^P(x) \\
&\leq \left[(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right) - d_i\right] \int_0^\omega G_i(t,s)ds + \sum_{k=0}^{p} G_i(t,t_k) M_{ki}^P \\
&\leq \frac{(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right)}{d_i} - 1 + \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} M_{ki}^P.
\end{aligned}$$

Hence, for any $t \in [0,\omega)$,

$$\left|x_i^P(t) - \varepsilon_i^P\right| = \lambda\left[\varepsilon_i^P U_i^P(\boldsymbol{x})(t) - 1\right] < \frac{(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right)}{d_i} - 2 + \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} M_{ki}^P,$$

and

$$\begin{aligned}
\|\boldsymbol{x} - \boldsymbol{x}_\varepsilon\| &= \sum_{i=1}^{n} \sup_{t\in[0,\omega]} |x_i(t) - \varepsilon_i| \\
&\leq \sum_{i=1}^{n} \sum_{P\in\mathcal{M}} \sup_{t\in[0,\omega]} \left|x_i^P(t) - \varepsilon_i^P\right| \\
&< \sum_{i=1}^{n} \sum_{P\in\mathcal{M}} \left[\frac{(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right)}{d_i} - 2 + \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} M_{ki}^P\right].
\end{aligned}$$

□

Theorem 5.3. *Let $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$. If Assumptions 5.4, 5.5, 5.7, 5.13, and 5.14 hold, the operator $U : \overline{B_\varepsilon} \subset K_\varepsilon \to K_\varepsilon$ has at least one fixed point in $\overline{B_\varepsilon}$, where $B_\varepsilon = \{x \in K_\varepsilon \mid \|\boldsymbol{x} - \boldsymbol{x}_\varepsilon\| < R\}$, with $\boldsymbol{x}_\varepsilon$ and R given by Lemma 5.7.*

Proof. Based on the Leray–Schauder theorem and Lemmas 5.5–5.7, we only need to show that the operator $U : \overline{B_\varepsilon} \subset K_\varepsilon \to K_\varepsilon$ is completely continuous.

Consider $\Omega \subset \overline{B_\varepsilon}$ a bounded set. We will show that $U(\Omega)$ is relatively compact, using Lemma 5.4.

Based on a similar argument as in the proof of Lemma 5.7, for any $\boldsymbol{x} \in \overline{B_\varepsilon}$, we have

$$U_i^P(\boldsymbol{x})(t) \leq \frac{(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right)}{d_i} - 1 + \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} M_{ki}^P,\ \forall t \in [0,\omega),\ \forall i = \overline{1,n},$$

and hence

$$\|U_i(\boldsymbol{x})\| \leq \sum_{i=1}^{n} \sum_{P\in\mathcal{M}} \left[\frac{(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right)}{d_i} - 1 + \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} M_{ki}^P \right].$$

Therefore, the set $U(\Omega)$ is bounded.

To show that $U(\Omega)$ is quasi-equicontinuous in $[0,\omega]$, consider $\boldsymbol{x}\in\Omega$, $l\in\{1,2,\ldots,p+1\}$, and $T_1,\ T_2\in(t_{l-1},t_l]\cap[0,\omega]$, and evaluate:

$$\begin{aligned}
\left|U_i^P(\boldsymbol{x})(T_1) - U_i^P(\boldsymbol{x})(T_2)\right| &\leq \left| \int_0^\omega \left[G_i(T_1,s) - G_i(T_2,s)\right] H_i^P(\boldsymbol{x})(s)ds \right. \\
&\quad \left. + \sum_{k=0}^{p} \left[G_i(T_1,t_k) - G_i(T_2,t_k)\right] J_{ki}^P(x) \right| \\
&\leq \left[(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right) - d_i\right] \int_0^\omega \left|G_i(T_1,s) - G_i(T_2,s)\right| ds \\
&\quad + \sum_{k=0}^{p} M_{ki}^P \left|G_i(T_1,t_k) - G_i(T_2,t_k)\right|.
\end{aligned}$$

On one hand, it can be easily seen that

$$\int_0^\omega \left|G_i(T_1,s) - G_i(T_2,s)\right| ds = \frac{2}{d_i(e^{d_i\omega}-1)} \left[e^{d_i\omega} + 1 - e^{d_i(\omega-|T_1-T_2|)} - e^{d_i|T_1-T_2|}\right].$$

On the other hand, we have

$$\begin{aligned}
\sum_{k=0}^{p} M_{ki}^P \left|G_i(T_1,t_k) - G_i(T_2,t_k)\right| &= \frac{1}{e^{d_i\omega}-1} \left[\sum_{k=0}^{l-1} M_{ki}^P \left| e^{d_i(\omega+t_k-T_1)} - e^{d_i(\omega+t_k-T_2)} \right| \right. \\
&\quad \left. + \sum_{k=l}^{p} M_{ki}^P \left| e^{d_i(t_k-T_1)} - e^{d_i(t_k-T_2)} \right| \right] \\
&\leq \frac{e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} M_{ki}^P e^{d_i t_k} \left| e^{-d_i T_1} - e^{-d_i T_2} \right| \\
&\leq \frac{d_i e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} M_{ki}^P e^{d_i t_k} \left|T_1 - T_2\right|.
\end{aligned}$$

Hence, for any $\boldsymbol{x}\in\Omega$, $l\in\{1,2,\ldots,p+1\}$, and $T_1,\ T_2\in(t_{l-1},t_l]\cap[0,\omega]$, the following estimate holds:

$$\begin{aligned}
\left\|U_i(\boldsymbol{x})(T_1) - U_i(\boldsymbol{x})(T_2)\right\|_1 &\leq \sum_{i=1}^{n} \sum_{P\in\mathcal{M}} \left|U_i^P(\boldsymbol{x})(T_1) - U_i^P(\boldsymbol{x})(T_2)\right| \\
&\leq \sum_{i=1}^{n} \sum_{P\in\mathcal{M}} \frac{2\left[(1+\alpha)\left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right) - d_i\right]}{d_i(e^{d_i\omega}-1)} \left[e^{d_i\omega} + 1 \right. \\
&\quad \left. - e^{d_i(\omega-|T_1-T_2|)} - e^{d_i|T_1-T_2|}\right] \\
&\quad + \sum_{i=1}^{n} \sum_{P\in\mathcal{M}} \left(\frac{d_i e^{d_i\omega}}{e^{d_i\omega}-1} \sum_{k=0}^{p} M_{ki}^P e^{d_i t_k} \right) \left|T_1 - T_2\right|.
\end{aligned}$$

The right hand side of the inequality does not depend on $\boldsymbol{x}$ and converges to 0 as $|T_1 - T_2| \to 0$. Therefore, $U(\Omega)$ is quasi-equicontinuous in $[0, \omega]$, and hence, relatively compact, and the proof is now complete. □

Corollary 5.1. *If Assumptions 5.4, 5.5, 5.7, 5.13, and 5.14 hold, there exist at least* 16^n *periodic solutions of system* (5.1.1)*, that is, at least one periodic solution in every set* $\overline{B_\varepsilon}$*,* $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$.

Lemma 5.8. *Let* $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$*. If Assumptions 5.4, 5.5, 5.7, 5.13, and 5.14 are fulfilled, then the set* Δ_ε *is invariant to the flow of system* (5.1.1)*.*

Proof. Let $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$. Consider an initial function satisfying $\boldsymbol{\phi}(s) \in \Delta_\varepsilon$ for any $s \leq 0$, and let $\boldsymbol{x}(t) = \boldsymbol{x}(t; \boldsymbol{\phi})$ be the solution of system (5.1.1).

Assume that there exists $\tau \in (t_0, t_1]$ such that $\boldsymbol{x}(t) \in \Delta_\varepsilon$, for any $t \in (0, \tau)$, and $\boldsymbol{x}(\tau) \in \partial\Delta_\varepsilon$. Hence, there exists $i \in \{1, 2, \ldots, n\}$ such that $x_i^P(\tau) = \varepsilon_i^P$. Based on Assumptions 5.4, 5.5, and 5.13, we have

$$\begin{aligned}
\varepsilon_i^P \dot{\boldsymbol{x}}_i^P(\tau) &= \varepsilon_i^P \Bigg(-d_i \varepsilon_i^P + \sum_{j=1}^{n} \left(a_{ij} f_j(x_j(\tau))\right)^P + \sum_{j=1}^{n} \left(b_{ij} f_j(x_j(\tau - \tau_{ij}(\tau)))\right)^P \\
&\quad + \sum_{j=1}^{n} \left(c_{ij} f_j \left(\int_0^\infty K_{ij}(s) x_j(\tau - s) ds \right) \right)^P + u_i^P(\tau) \Bigg) \\
&\geq -d_i + \left(a_{ii}^R + b_{ii}^R + c_{ii}^R\right) \alpha - \sum_{Q \in \mathcal{M} \setminus \{R\}} \left(\left|a_{ii}^Q\right| + \left|b_{ii}^Q\right| + \left|c_{ii}^Q\right| \right) \\
&\quad - \sum_{j \neq i} \sum_{Q \in \mathcal{M}} \left(\left|a_{ij}^Q\right| + \left|b_{ij}^Q\right| + \left|c_{ij}^Q\right| \right) - \left|u_i^P(\tau)\right| \\
&> 0,
\end{aligned}$$

Therefore, the function $\varepsilon_i^P x_i^P$ is strictly increasing on some small interval $(\tau - \delta, \tau] \subset (t_0, \tau]$. Hence $\varepsilon_i^P x_i^P(t) < \varepsilon_i^P x_i^P(\tau) = \left(\varepsilon_i^P\right)^2 = 1$ for any $t \in (\tau - \delta, \tau)$. This is absurd, as $x_i(t) \in J_{\mathbb{H}}(\varepsilon_i)$ for any $t < \tau$.

It follows that $\boldsymbol{x}(t) \in \Delta_\varepsilon$ for any $t \in (t_0, t_1]$. Assumption 5.14 guarantees that $\boldsymbol{x}(t_1^+) \in \Delta_\varepsilon$.

By mathematical induction, it can be easily shown that $\boldsymbol{x}(t) \in \Delta_\varepsilon$ for any $t \in (t_{k-1}, t_k]$, and $\boldsymbol{x}(t_k^+) \in \Delta_\varepsilon$, for any $k \in \mathbb{Z}^+$.

Hence, the solution $\boldsymbol{x}(t; \boldsymbol{\phi})$, with the initial condition $\boldsymbol{\phi}(s) \in \Delta_\varepsilon$ for any $s \leq 0$, will remain in Δ_ε for any $t \geq 0$. □

Theorem 5.4. *Assume that Assumptions 5.4, 5.5, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, and 5.14 hold. Then, for every* $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$*, there exists a unique exponentially stable periodic solution in* $\overline{B_\varepsilon}$*, and its region of attraction includes* Δ_ε*.*

Proof. Let $\varepsilon \in \{\pm 1 \pm 1\imath \pm 1\jmath \pm 1\kappa\}^n$. Let $\boldsymbol{x}(t) = \boldsymbol{x}(t; \boldsymbol{\phi})$ and $\boldsymbol{y}(t) = \boldsymbol{y}(t; \boldsymbol{\psi})$ two solutions of system (5.1.1) with the initial functions $\boldsymbol{\phi}(s)$, $\boldsymbol{\psi}(s) \in \Delta_\varepsilon$, for any $s \leq 0$. Based on Lemma 5.8, we get that $\boldsymbol{x}(t; \boldsymbol{\phi})$, $\boldsymbol{y}(t; \boldsymbol{\psi}) \in \Delta_\varepsilon$ for any $t > 0$. For $t > 0$, $t \neq t_k$, we have

$$
\begin{aligned}
D^+ |x_i(t) - y_i(t)| &= \sum_{P\in\mathcal{M}} \frac{x_i^P(t) - y_i^P(t)}{|x_i(t) - y_i(t)|} \left(\dot{x}_i^P(t) - \dot{y}_i^P(t)\right) \\
&= \sum_{P\in\mathcal{M}} \frac{x_i^P(t) - y_i^P(t)}{|x_i(t) - y_i(t)|} \Bigg(-d_i \left(x_i^P(t) - y_i^P(t)\right) + \sum_{j=1}^{n} \left(a_{ij} \left(f_j(x_j(t)) - f_j(y_j(t))\right)\right)^P \\
&\quad + \sum_{j=1}^{n} \left(b_{ij} \left(f_j(x_j(t-\tau_{ij}(t))) - f_j(y_j(t-\tau_{ij}(t)))\right)\right)^P \\
&\quad + \sum_{j=1}^{n} \left(c_{ij} \left(f_j \left(\int_0^\infty K_{ij}(s) x_j(t-s) ds \right) - f_j \left(\int_0^\infty K_{ij}(s) y_j(t-s) ds \right) \right) \right)^P \Bigg) \\
&= -d_i |x_i(t) - y_i(t)| + \sum_{P\in\mathcal{M}} \frac{x_i^P(t) - y_i^P(t)}{|x_i(t) - y_i(t)|} \Bigg(\sum_{j=1}^{n} \left(a_{ij} \left(f_j(x_j(t)) - f_j(y_j(t))\right)\right)^P \\
&\quad + \sum_{j=1}^{n} \left(b_{ij} \left(f_j(x_j(t-\tau_{ij}(t))) - f_j(y_j(t-\tau_{ij}(t)))\right)\right)^P \\
&\quad + \sum_{j=1}^{n} \left(c_{ij} \left(f_j \left(\int_0^\infty K_{ij}(s) x_j(t-s) ds \right) - f_j \left(\int_0^\infty K_{ij}(s) y_j(t-s) ds \right) \right) \right)^P \Bigg) \\
&\leq -d_i |x_i(t) - y_i(t)| + \sum_{P\in\mathcal{M}} \sum_{j=1}^{n} \sum_{(Q,S)\in\mathcal{M}_P} \Big(\left|a_{ij}^Q\right| \beta_j^S |x_j(t) - y_j(t)| \\
&\quad + \left|b_{ij}^Q\right| \beta_j^S |x_j(t-\tau_{ij}(t)) - y_j(t-\tau_{ij}(t))| \\
&\quad + \left|c_{ij}^Q\right| \beta_j^S \int_0^\infty K_{ij}(s) |x_j(t-s) - y_j(t-s)| ds \Big) \\
&= -d_i |x_i(t) - y_i(t)| + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \Big(\left|a_{ij}^Q\right| \beta_j^P |x_j(t) - y_j(t)| \\
&\quad + \left|b_{ij}^Q\right| \beta_j^P |x_j(t-\tau_{ij}(t)) - y_j(t-\tau_{ij}(t))| \\
&\quad + \left|c_{ij}^Q\right| \beta_j^P \int_0^\infty K_{ij}(s) |x_j(t-s) - y_j(t-s)| ds \Big).
\end{aligned}
$$

Denoting $w_i(t) = e^{\mu_0 t} |x_i(t) - y_i(t)|$, for $t > 0$, $t \neq t_k$, we obtain

$$
\begin{aligned}
D^+ w_i(t) &= \mu_0 w_i(t) + e^{\mu_0 t} D^+ |x_i(t) - y_i(t)| \\
&\leq (\mu_0 - d_i) w_i(t) + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \Big(\left|a_{ij}^Q\right| \beta_j^P w_j(t) \\
&\quad + \left|b_{ij}^Q\right| \beta_j^P e^{\mu_0 \rho_{ij}} w_j(t-\tau_{ij}(t)) + \left|c_{ij}^Q\right| \beta_j^P \int_0^\infty K_{ij}(s) e^{\mu_0 s} w_j(t-s) ds \Big).
\end{aligned}
$$

If, like above, θ_{ij} is the inverse of the strictly increasing function $t - \tau_{ij}(t)$, we consider the function $V : \mathbb{R}_+ \to \mathbb{R}_+$ defined by

$$
\begin{aligned}
V(t) &= \sum_{i=1}^{n} (w_i(t) \\
&\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|b_{ij}^{Q}\right| \beta_j^P e^{\mu_0\rho_{ij}} \int_{t-\tau_{ij}(t)}^{t} \frac{w_j(s)}{1-\tau'_{ij}(\theta_{ij}(s))} ds \\
&\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|c_{ij}^{Q}\right| \beta_j^P \int_0^{\infty} K_{ij}(s) e^{\mu_0 s} \left(\int_{t-s}^{t} w_j(u) du \right) ds \Bigg).
\end{aligned}
$$

For $t > 0$, $t \neq t_k$, we obtain

$$
\begin{aligned}
D^+V(t) &= \sum_{i=1}^{n} \big(D^+ w_i(t) \\
&\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|b_{ij}^{Q}\right| \beta_j^P e^{\mu_0\rho_{ij}} \left(\frac{w_j(t)}{1-\tau'_{ij}(\theta_{ij}(t))} - \frac{(1-\tau'_{ij}(t)) w_j(t-\tau_{ij}(t))}{1-\tau'_{ij}(\theta_{ij}(t-\tau_{ij}(t)))} \right) \\
&\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|c_{ij}^{Q}\right| \beta_j^P \int_0^{\infty} K_{ij}(s) e^{\mu_0 s} (w_j(t) - w_j(t-s)) ds \Bigg) \\
&\leq \sum_{i=1}^{n} \Bigg((\mu_0 - d_i) w_i(t) + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|a_{ij}^{Q}\right| \beta_j^P w_j(t) \\
&\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|b_{ij}^{Q}\right| \beta_j^P \frac{e^{\mu_0\rho_{ij}}}{1-\tau'_{ij}(\theta_{ij}(t))} w_j(t) \\
&\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|c_{ij}^{Q}\right| \beta_j^P \left(\int_0^{\infty} K_{ij}(s) e^{\mu_0 s} ds \right) w_j(t) \Bigg) \\
&\leq \sum_{i=1}^{n} \Bigg(\mu_0 - d_i + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|a_{ji}^{Q}\right| \beta_i^P \\
&\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|b_{ji}^{Q}\right| \beta_i^P \frac{e^{\mu_0\rho_{ji}}}{1-\rho'_{ji}} \\
&\quad + \sum_{j=1}^{n} \sum_{P\in\mathcal{M}} \sum_{Q\in\mathcal{M}} \left|c_{ji}^{Q}\right| \beta_i^P \left(\int_0^{\infty} K_{ji}(s) e^{\mu_0 s} ds \right) \Bigg) w_i(t) \\
&< 0.
\end{aligned}
$$

Hence, the function V is strictly decreasing on every interval (t_k, t_{k+1}), and therefore $V(t) < V(t_k^+)$, for any $(t_k, t_{k+1}]$.

Moreover, from Assumption 5.9, and taking into consideration that $\sum_{i=1}^{n} w_i(t) \leq V(t)$, for any $t \in \mathbb{R}_+$, we have

$$
\begin{aligned}
\sum_{i=1}^{n} w_i(t_k^+) &= e^{\mu_0 t_k} \left\| \boldsymbol{x}(t_k^+) - \boldsymbol{y}(t_k^+) \right\|_1 \\
&= e^{\mu_0 t_k} \left\| \boldsymbol{x}(t_k) + J_k(\boldsymbol{x}) - \boldsymbol{y}(t_k) - J_k(\boldsymbol{y}) \right\|_1 \\
&\leq (1+\gamma_k) \left\| \boldsymbol{x}(t_k) - \boldsymbol{y}(t_k) \right\|_1 + \delta_k \int_{t_{k-1}}^{t_k} e^{\mu_0 t_k} \left\| \boldsymbol{x}(s) - \boldsymbol{y}(s) \right\|_1 ds \\
&= (1+\gamma_k) \sum_{i=1}^{n} w_i(t_k) + \delta_k \int_{t_{k-1}}^{t_k} e^{\mu_0 (t_k - s)} \sum_{i=1}^{n} w_i(s) ds \\
&\leq (1+\gamma_k) \sum_{i=1}^{n} w_i(t_k) + \delta_k \int_{t_{k-1}}^{t_k} e^{\mu_0 (t_k - s)} V(s) ds \\
&\leq (1+\gamma_k) \sum_{i=1}^{n} w_i(t_k) + \delta_k V(t_{k-1}^+) \int_{t_{k-1}}^{t_k} e^{\mu_0 (t_k - s)} ds \\
&= (1+\gamma_k) \sum_{i=1}^{n} w_i(t_k) + \delta_k \frac{e^{\mu_0 (t_k - t_{k-1})} - 1}{\mu_0} V(t_{k-1}^+),
\end{aligned}
$$

and hence

$$
\begin{aligned}
V(t_k^+) &= \sum_{i=1}^{n} \big(w_i(t_k^+) \\
&\quad + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left| b_{ij}^Q \right| \beta_j^P e^{\mu_0 \rho_{ij}} \int_{t_k^+ - \tau_{ij}(t_k^+)}^{t_k^+} \frac{w_j(s)}{1 - \tau_{ij}'(\theta_{ij}(s))} ds \\
&\quad + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left| c_{ij}^Q \right| \beta_j^P \int_0^{\infty} K_{ij}(s) e^{\mu_0 s} \left(\int_{t_k^+ - s}^{t_k^+} w_j(u) du \right) ds \Bigg) \\
&\leq (1+\gamma_k) \left(\sum_{i=1}^{n} [w_i(t_k) \right. \\
&\quad + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left| b_{ij}^Q \right| \beta_j^P e^{\mu_0 \rho_{ij}} \int_{t_k - \tau_{ij}(t_k)}^{t_k} \frac{w_j(s)}{1 - \tau_{ij}'(\theta_{ij}(s))} ds \\
&\quad \left. + \sum_{j=1}^{n} \sum_{P \in \mathcal{M}} \sum_{Q \in \mathcal{M}} \left| c_{ij}^Q \right| \beta_j^P \int_0^{\infty} K_{ij}(s) e^{\mu_0 s} \left(\int_{t_k - s}^{t_k} w_j(u) du \right) ds \right] \Bigg) \\
&\quad + \delta_k \frac{e^{\mu_0 (t_k - t_{k-1})} - 1}{\mu_0} V(t_{k-1}^+) \\
&= (1+\gamma_k) V(t_k) + \delta_k \frac{e^{\mu_0 (t_k - t_{k-1})} - 1}{\mu_0} V(t_{k-1}^+) \\
&< \left(1 + \gamma_k + \delta_k \frac{e^{\mu_0 (t_k - t_{k-1})} - 1}{\mu_0} \right) V(t_{k-1}^+).
\end{aligned}
$$

Using Assumption 5.10, we obtain

$$\begin{aligned} V(t_k^+) &< V(0^+)\prod_{j=1}^{k}\left(1+\gamma_j+\delta_j\frac{e^{\mu_0(t_j-t_{j-1})}-1}{\mu_0}\right) \\ &\leq (1+\gamma_0)\,V(0)\prod_{j=1}^{k}\left(1+\gamma_j+\delta_j\frac{e^{\mu_0(t_j-t_{j-1})}-1}{\mu_0}\right) \\ &\leq (1+\gamma_0)\,V(0)e^{\zeta t_k}. \end{aligned}$$

Moreover, we have

$$\begin{aligned} V(0) &= \sum_{i=1}^{n}\left(|x_i(0)-y_i(0)|\right. \\ &\quad +\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\left|b_{ij}^{Q}\right|\beta_j^P e^{\mu_0\rho_{ij}}\int_{-\tau_{ij}(0)}^{0}\frac{e^{\mu_0 s}\,|x_i(s)-y_i(s)|}{1-\tau_{ij}'(\theta_{ij}(s))}ds \\ &\quad \left.+\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\left|c_{ij}^{Q}\right|\beta_j^P\int_0^{\infty}K_{ij}(s)e^{\mu_0 s}\left(\int_{-s}^{0}e^{\mu_0 u}\,|x_i(u)-y_i(u)|\,du\right)ds\right) \\ &\leq \|\boldsymbol{\phi}-\boldsymbol{\psi}\|_\infty\left[n+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\beta_j^P\left(\left|b_{ij}^{Q}\right|e^{\mu_0\rho_{ij}}\int_{-\tau_{ij}(0)}^{0}\frac{e^{\mu_0 s}}{1-\tau_{ij}'(\theta_{ij}(s))}ds+\right.\right. \\ &\quad \left.\left.+\left|c_{ij}^{Q}\right|\frac{1}{\mu_0}\left(\int_0^{\infty}K_{ij}(s)e^{\mu_0 s}ds-1\right)\right)\right] \\ &= \|\boldsymbol{\phi}-\boldsymbol{\psi}\|_\infty\left[n+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\beta_j^P\left(\left|b_{ij}^{Q}\right|e^{\mu_0\rho_{ij}}\int_0^{\theta_{ij}(0)}e^{\mu_0(u-\tau_{ij}(u))}du+\right.\right. \\ &\quad \left.\left.+\left|c_{ij}^{Q}\right|\frac{1}{\mu_0}\left(\int_0^{\infty}K_{ij}(s)e^{\mu_0 s}ds-1\right)\right)\right]. \end{aligned}$$

For any $t\in(t_k,t_{k+1}]$, we obtain

$$V(t)<V(t_k^+)<\Lambda\,\|\boldsymbol{\phi}-\boldsymbol{\psi}\|_\infty e^{\zeta t_k}<\Lambda\,\|\boldsymbol{\phi}-\boldsymbol{\psi}\|_\infty e^{\zeta t},$$

where

$$\begin{aligned} \Lambda &= (1+\gamma_0)\left[n+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{P\in\mathcal{M}}\sum_{Q\in\mathcal{M}}\beta_j^P\left(\left|b_{ij}^{Q}\right|e^{\mu_0\rho_{ij}}\int_0^{\theta_{ij}(0)}e^{\mu_0(u-\tau_{ij}(u))}du+\right.\right. \\ &\quad \left.\left.+\left|c_{ij}^{Q}\right|\frac{1}{\mu_0}\left(\int_0^{\infty}K_{ij}(s)e^{\mu_0 s}ds-1\right)\right)\right]. \end{aligned}$$

Finally, it follows that

$$|x_i(t)-y_i(t)|=e^{-\mu_0 t}w_i(t)<e^{-\mu_0 t}V(t)<\Lambda\,\|\boldsymbol{\phi}-\boldsymbol{\psi}\|_\infty e^{(\zeta-\mu_0)t},$$

$\forall t>0,\ \forall i=\overline{1,n}$. Using Corollary 5.1, and taking into account that $\zeta<\mu_0$ (by Assumption 5.10), it easily follows that there exists a unique periodic solution $\boldsymbol{x}_\varepsilon^\star(t)\in\Delta_\varepsilon$, for any $t\in\mathbb{R}$, which is exponentially stable and its region of attraction includes Δ_ε. □

5.1.2 Numerical examples

Example 5.1. Consider the following impulsive quaternion-valued neural network with one neuron, with mixed delays:

$$\begin{cases} \dot{x}(t) = -d_1 x(t) + a_{11} f_1(x(t)) + b_{11} f_1(x(t-\tau_{11}(t))) + c_{11} f_1\left(\int_0^\infty e^{-s} x(t-s)ds\right) + u_1, & t \neq t_k, \\ x(t_k^-) = x(t_k), \quad x(t_k^+) = x(t_k) + \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} \frac{1}{s+1}[x(s) - x(t_k)]ds, & t = t_k,\ k \in \mathbb{Z}^+, \end{cases} \tag{5.1.9}$$

where

$$d_1 = 2,\ a_{11} = 12 - \imath + 2\jmath - 3\kappa,\ b_{11} = 18 - 4\imath + 5\jmath - 6\kappa,$$
$$c_{11} = 16 - \imath + 2\jmath - 3\kappa,\ u_1 = 2 - 3\imath + 3\jmath - 2\kappa.$$

The activation function components are $f_1^P(s) = \tanh\left(5s^P\right)\tanh\left(10\left(s^P\right)^2 - 1\right)$, $\forall P \in \mathcal{M}$, and $t_k = 2k$. We can see that $\beta_1^P = 0.0009$, that $\left|f_1^P(s)\right| \leq 1$, and that there exists $\alpha = f_1^P(1) = 0.9999$ for which $f_1^P(s) \geq \alpha$, if $s^P \geq 1$, and $f_1^P(s) \leq -\alpha$, if $s^P \leq -1$, $\forall s \in \mathbb{H}$, $\forall P \in \mathcal{M}$, showing that the functions f_1^P satisfy Assumptions 5.4, 5.5, and 5.7. It can be verified that Assumptions 5.8 and 5.13 also hold. The delay kernel is $K_{11}(s) = e^{-s}$, and satisfies (5.1.3) in Assumption 5.2 with $\mu \in (0,1)$. The time-varying delay is $\tau_{11}(t) = \frac{1}{\pi}\sin^2\left(\frac{\pi}{2}t\right)$, which satisfies Assumption 5.1 with $\rho_{11} = \frac{1}{\pi}$, $\rho'_{11} = 0.5$. The jump operators are given by:

$$J_k(x) = \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} \frac{1}{s+1}[x(s) - x(t_k)]ds,$$

and satisfy Assumptions 5.3 and 5.9, with

$$\begin{aligned} \gamma_k &= \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} \frac{1}{s+1} ds = \frac{1}{2}\ln\left(1 + \frac{2}{2k-1}\right) \leq \frac{\ln 3}{2}, \\ \delta_k &= \frac{1}{t_k - t_{k-1}} \cdot \frac{1}{t_{k-1}+1} = \frac{1}{2(2k-1)} \leq \frac{1}{2}. \end{aligned}$$

Considering $x \in \Delta_\varepsilon$ for any $t \leq t_k$, we obtain

$$\begin{aligned} x(t_k) + J_k(x) &= x(t_k) + \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} \frac{1}{s+1}[x(s) - x(t_k)]ds \\ &= \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} \left[\frac{1}{s+1}x(s) + \left(1 - \frac{1}{s+1}\right)x(t_k)\right] ds. \end{aligned}$$

Since $\varepsilon^P x^P(t) > 1$ for any $t \leq t_k$, and any $P \in \mathcal{M}$, it easily follows that

$$\varepsilon^P \left[\frac{1}{s+1}x^P(s) + \left(1 - \frac{1}{s+1}\right)x^P(t_k)\right] > \frac{1}{s+1} + \left(1 - \frac{1}{s+1}\right) = 1,\ \forall s \geq 0,$$

and hence, for any $P \in \mathcal{M}$, we have

$$\varepsilon^P \left[x(t_k) + J_k(x)\right]^P = \frac{1}{t_k - t_{k-1}} \int_{t_{k-1}}^{t_k} \varepsilon^P \left[\frac{1}{s+1}x^P(s) + \left(1 - \frac{1}{s+1}\right)x^P(t_k)\right] ds > 1.$$

Therefore, $x(t_k) + J_k(x) \in \Delta_\varepsilon$, and we obtain that the operators J_k satisfy Assumption 5.6 as well.

Choosing $\mu_0 < 0.8912$, inequality (5.1.5) from Assumption 5.10 is satisfied. Moreover, it can be shown that

$$\zeta = \sup_{k\in\mathbb{Z}_+^*} \frac{1}{t_k}\sum_{j=1}^{k} \ln\left(1+\gamma_j+\delta_j\frac{e^{\mu_0(t_j-t_{j-1})}-1}{\mu_0}\right) = \frac{1}{2}\ln\left(1+\frac{\ln 3}{2}+\frac{e^{2\mu_0}-1}{2\mu_0}\right),$$

and it follows that $\zeta < \mu_0$ whenever $\mu_0 > 0.6361$. Therefore, Assumption 5.10 is satisfied for $\mu_0 \in (0.6361, 0.8912)$.

It follows from Theorem 5.1 that if $\left|u_1^P\right| < 16.9954$ for $P \in \mathcal{M}$, then system (5.1.9) has 16 exponentially stable equilibrium states. For $u_1 = 2 - 3\imath + 3\jmath - 2\kappa$, these steady states are:

No.	Steady state	Domain of attraction
1	$x^{1+\imath+\jmath+\kappa} = 28.5 + 29\imath + 26\jmath + 8.50\kappa$	$\Delta_{1+\imath+\jmath+\kappa} = (1,\infty) + (1,\infty)\imath + (1,\infty)\jmath + (1,\infty)\kappa$
2	$x^{1+\imath+\jmath-\kappa} = 16.5 + 20\imath + 20\jmath - 37.5\kappa$	$\Delta_{1+\imath+\jmath-\kappa} = (1,\infty) + (1,\infty)\imath + (1,\infty)\jmath + (-\infty,-1)\kappa$
3	$x^{1+\imath-\jmath+\kappa} = 37.5 + 17\imath - 20\jmath + 14.5\kappa$	$\Delta_{1+\imath-\jmath+\kappa} = (1,\infty) + (1,\infty)\imath + (-\infty,-1)\jmath + (1,\infty)\kappa$
4	$x^{1+\imath-\jmath-\kappa} = 25.5 + 8\imath - 26\jmath - 31.5\kappa$	$\Delta_{1+\imath-\jmath-\kappa} = (1,\infty) + (1,\infty)\imath + (-\infty,-1)\jmath + (-\infty,-1)\kappa$
5	$x^{1-\imath+\jmath+\kappa} = 22.5 - 17\imath + 38\jmath + 17.5\kappa$	$\Delta_{1-\imath+\jmath+\kappa} = (1,\infty) + (-\infty,-1)\imath + (1,\infty)\jmath + (1,\infty)\kappa$
6	$x^{1-\imath+\jmath-\kappa} = 10.5 - 26\imath + 32\jmath - 28.5\kappa$	$\Delta_{1-\imath+\jmath-\kappa} = (1,\infty) + (-\infty,-1)\imath + (1,\infty)\jmath + (-\infty,-1)\kappa$
7	$x^{1-\imath-\jmath+\kappa} = 31.5 - 29\imath - 8\jmath + 23.5\kappa$	$\Delta_{1-\imath-\jmath+\kappa} = (1,\infty) + (-\infty,-1)\imath + (-\infty,-1)\jmath + (1,\infty)\kappa$
8	$x^{1-\imath-\jmath-\kappa} = 19.5 - 38\imath - 14\jmath - 22.5\kappa$	$\Delta_{1-\imath-\jmath-\kappa} = (1,\infty) + (-\infty,-1)\imath + (-\infty,-1)\jmath + (-\infty,-1)\kappa$
9	$x^{-1+\imath+\jmath+\kappa} = -17.5 + 35\imath + 17\jmath + 20.5\kappa$	$\Delta_{-1+\imath+\jmath+\kappa} = (-\infty,-1) + (1,\infty)\imath + (1,\infty)\jmath + (1,\infty)\kappa$
10	$x^{-1+\imath+\jmath-\kappa} = -29.5 + 26\imath + 11\jmath - 25.5\kappa$	$\Delta_{-1+\imath+\jmath-\kappa} = (-\infty,-1) + (1,\infty)\imath + (1,\infty)\jmath + (-\infty,-1)\kappa$
11	$x^{-1+\imath-\jmath+\kappa} = -8.50 + 23\imath - 29\jmath + 26.5\kappa$	$\Delta_{-1+\imath-\jmath+\kappa} = (-\infty,-1) + (1,\infty)\imath + (-\infty,-1)\jmath + (1,\infty)\kappa$
12	$x^{-1+\imath-\jmath-\kappa} = -20.5 + 14\imath - 35\jmath - 19.5\kappa$	$\Delta_{-1+\imath-\jmath-\kappa} = (-\infty,-1) + (1,\infty)\imath + (-\infty,-1)\jmath + (-\infty,-1)\kappa$
13	$x^{-1-\imath+\jmath+\kappa} = -23.5 - 11\imath + 29\jmath + 29.5\kappa$	$\Delta_{-1-\imath+\jmath+\kappa} = (-\infty,-1) + (-\infty,-1)\imath + (1,\infty)\jmath + (1,\infty)\kappa$
14	$x^{-1-\imath+\jmath-\kappa} = -35.5 - 20\imath + 23\jmath - 16.5\kappa$	$\Delta_{-1-\imath+\jmath-\kappa} = (-\infty,-1) + (-\infty,-1)\imath + (1,\infty)\jmath + (-\infty,-1)\kappa$
15	$x^{-1-\imath-\jmath+\kappa} = -14.5 - 23\imath - 17\jmath + 35.5\kappa$	$\Delta_{-1-\imath-\jmath+\kappa} = (-\infty,-1) + (-\infty,-1)\imath + (-\infty,-1)\jmath + (1,\infty)\kappa$
16	$x^{-1-\imath-\jmath-\kappa} = -26.5 - 32\imath - 23\jmath - 10.5\kappa$	$\Delta_{-1-\imath-\jmath-\kappa} = (-\infty,-1) + (-\infty,-1)\imath + (-\infty,-1)\jmath + (-\infty,-1)\kappa$

Example 5.2. Consider the following impulsive hybrid quaternion-valued network of one neuron with mixed delays:

$$\begin{cases} \dot{x}(t) = -d_1 x(t) + a_{11} f_1(x(t)) + b_{11} f_1(x(t-\tau_{11}(t))) + c_{11} f_1\left(\int_0^\infty e^{-s} x(t-s)ds\right) + u_1(t), \quad t \neq t_k, \\ x^P(t_k^-) = x^P(t_k), \quad x^P(t_k^+) = x^P(t_k) + \arctan(x^P(t_k)) + \int_{t_{k-1}}^{t_k} \frac{x^P(s)}{(x^P(s))^2+1}ds, \ \forall P \in \mathcal{M}, \quad t = t_k, \ k \in \mathbb{Z}^+, \end{cases} \tag{5.1.10}$$

where

$$d_1 = 2, \ a_{11} = 12 - \imath + 2\jmath - 3\kappa, \ b_{11} = 18 - 4\imath + 5\jmath - 6\kappa,$$
$$c_{11} = 16 - \imath + 2\jmath - 3\kappa, \ u_1(t) = 4\sin\left(\frac{\pi}{2}t\right)(1 + \imath + \jmath + \kappa).$$

The external input $u_1(t)$ is a periodic function of period $\omega = 4$. The activation function components are $f_1^P(s) = \tanh\left(5s^P\right)\tanh\left(10\left(s^P\right)^2 - 1\right)$, $\forall P \in \mathcal{M}$, and $t_k = 2k$. The time-varying delay is $\tau_{11}(t) = \frac{1}{\pi}\sin^2\left(\frac{\pi}{2}t\right)$. Like in Example 5.1, it can be verified that Assumptions 5.1, 5.2, 5.3, 5.4, 5.5, 5.7, 5.8, and 5.13 hold with $\beta_1^P = 0.0009$, $\alpha = 0.9999$, $\mu \in (0,1)$, $\rho_{11} = \frac{1}{\pi}$, $\rho'_{11} = 0.5$. Also, Assumption 5.12 holds for $p = 2$.

The jump operators are given by:

$$J_k(x) = \arctan(x^P(t_k)) + \int_{t_{k-1}}^{t_k} \frac{x^P(s)}{(x^P(s))^2+1}ds,$$

and satisfy Assumptions 5.9, 5.11, and 5.14, with $\gamma_k = \delta_k = 0.5$.

Inequality (5.1.5) from Assumption 5.10 is satisfied for any $\mu_0 < 0.8912$. Moreover, it can be shown that

$$\zeta = \sup_{k\in\mathbb{Z}_+^*} \frac{1}{t_k}\sum_{j=1}^{k} \ln\left(1+\gamma_j+\delta_j\frac{e^{\mu_0(t_j-t_{j-1})}-1}{\mu_0}\right) = \frac{1}{2}\ln\left(\frac{3}{2}+\frac{e^{2\mu_0}-1}{2\mu_0}\right),$$

and it follows that $\zeta < \mu_0$ whenever $\mu_0 > 0.6255$. Therefore, Assumption 5.10 is satisfied for $\mu_0 \in (0.6255, 0.8912)$.

It follows from Theorem 5.4 that if $\left|u_1^P(t)\right| < 16.9954$ for $P \in \mathcal{M}$, $\forall t \in \mathbb{R}$, then system (5.1.10) has 16 exponentially stable periodic solutions of period $\omega = 4$, denoted as follows:

No.	Periodic solution	Domain of attraction
1	$x^{1+\imath+\jmath+\kappa}(t)$	$\Delta_{1+\imath+\jmath+\kappa} = (1,\infty)+(1,\infty)\imath+(1,\infty)\jmath+(1,\infty)\kappa$
2	$x^{1+\imath+\jmath-\kappa}(t)$	$\Delta_{1+\imath+\jmath-\kappa} = (1,\infty)+(1,\infty)\imath+(1,\infty)\jmath+(-\infty,-1)\kappa$
3	$x^{1+\imath-\jmath+\kappa}(t)$	$\Delta_{1+\imath-\jmath+\kappa} = (1,\infty)+(1,\infty)\imath+(-\infty,-1)\jmath+(1,\infty)\kappa$
4	$x^{1+\imath-\jmath-\kappa}(t)$	$\Delta_{1+\imath-\jmath-\kappa} = (1,\infty)+(1,\infty)\imath+(-\infty,-1)\jmath+(-\infty,-1)\kappa$
5	$x^{1-\imath+\jmath+\kappa}(t)$	$\Delta_{1-\imath+\jmath+\kappa} = (1,\infty)+(-\infty,-1)\imath+(1,\infty)\jmath+(1,\infty)\kappa$
6	$x^{1-\imath+\jmath-\kappa}(t)$	$\Delta_{1-\imath+\jmath-\kappa} = (1,\infty)+(-\infty,-1)\imath+(1,\infty)\jmath+(-\infty,-1)\kappa$
7	$x^{1-\imath-\jmath+\kappa}(t)$	$\Delta_{1-\imath-\jmath+\kappa} = (1,\infty)+(-\infty,-1)\imath+(-\infty,-1)\jmath+(1,\infty)\kappa$
8	$x^{1-\imath-\jmath-\kappa}(t)$	$\Delta_{1-\imath-\jmath-\kappa} = (1,\infty)+(-\infty,-1)\imath+(-\infty,-1)\jmath+(-\infty,-1)\kappa$
9	$x^{-1+\imath+\jmath+\kappa}(t)$	$\Delta_{-1+\imath+\jmath+\kappa} = (-\infty,-1)+(1,\infty)\imath+(1,\infty)\jmath+(1,\infty)\kappa$
10	$x^{-1+\imath+\jmath-\kappa}(t)$	$\Delta_{-1+\imath+\jmath-\kappa} = (-\infty,-1)+(1,\infty)\imath+(1,\infty)\jmath+(-\infty,-1)\kappa$
11	$x^{-1+\imath-\jmath+\kappa}(t)$	$\Delta_{-1+\imath-\jmath+\kappa} = (-\infty,-1)+(1,\infty)\imath+(-\infty,-1)\jmath+(1,\infty)\kappa$
12	$x^{-1+\imath-\jmath-\kappa}(t)$	$\Delta_{-1+\imath-\jmath-\kappa} = (-\infty,-1)+(1,\infty)\imath+(-\infty,-1)\jmath+(-\infty,-1)\kappa$
13	$x^{-1-\imath+\jmath+\kappa}(t)$	$\Delta_{-1-\imath+\jmath+\kappa} = (-\infty,-1)+(-\infty,-1)\imath+(1,\infty)\jmath+(1,\infty)\kappa$
14	$x^{-1-\imath+\jmath-\kappa}(t)$	$\Delta_{-1-\imath+\jmath-\kappa} = (-\infty,-1)+(-\infty,-1)\imath+(1,\infty)\jmath+(-\infty,-1)\kappa$
15	$x^{-1-\imath-\jmath+\kappa}(t)$	$\Delta_{-1-\imath-\jmath+\kappa} = (-\infty,-1)+(-\infty,-1)\imath+(-\infty,-1)\jmath+(1,\infty)\kappa$
16	$x^{-1-\imath-\jmath-\kappa}(t)$	$\Delta_{-1-\imath-\jmath-\kappa} = (-\infty,-1)+(-\infty,-1)\imath+(-\infty,-1)\jmath+(-\infty,-1)\kappa$

Chapter 6

Dynamics of octonion-valued neural networks (OVNNs)

Neural networks with values in multidimensional domains have attracted the attention of researchers over the last few years. First introduced by [227], complex-valued neural networks (CVNNs) have found numerous applications, which include antenna design, radar imaging, estimation of direction of arrival and beamforming, image processing, communications signal processing, and many others [77, 78]. Quaternion-valued neural networks (QVNNs) were introduced by [4], and have applications in chaotic time-series prediction [7], color image compression [89], color night vision [104], polarized signal classification [22], and 3D wind forecasting [92, 213]. Clifford-valued neural networks (ClVNNs), proposed by [139, 140], and later discussed by [24, 103], have potential applications in high-dimensional data processing. They represent a generalization of the complex- and quaternion-valued neural networks, because complex and quaternion algebras are special cases of the 2^n-dimensional Clifford algebras, where $n \geq 1$.

A different generalization of the complex and quaternion algebras is the octonion algebra. It is an 8-dimensional normed division algebra, which means that a norm and a multiplicative inverse can be defined on it. In fact, it is the only normed division algebra that can be defined over the field of real numbers, besides the complex and quaternion algebras. The octonion algebra is not a special kind of Clifford algebra, because the Clifford algebras are all associative, whereas the octonion algebra is not.

Octonions have applications in physics and geometry [136, 49], and have also been successfully applied in the signal processing domain in the recent years [197]. The signal processing applications include salient object detection [56, 57], hyperspectral fluorescence data fusion [13], L1-norm minimization for octonion signals [224], and the octonion Fourier transform [19]. In physics, octonions were used to reformulate electrodynamics and chromodynamics equations [25, 26, 27, 28], the Maxwell equations [48], the gravitational field equations [47], and the Dirac equation [99].

Taking all the above facts into consideration, octonions may have potential applications in the neural network domain, also. Thus, feedforward octonion-valued neural networks (OVNNs) were first proposed in the author's paper [149]. They may be applied in the signal processing domain, where certain signals can be better represented in the octonion domain. High-dimensional data processing could also benefit from the use of octonion networks. In the same way as complex-valued networks were better than real-valued ones for some applications, and quaternion-valued networks were better than both real- and complex-valued networks in others, octonion-valued networks may outperform all of the above in yet other problems. They

could represent an alternative for 8-dimensional Clifford-valued neural networks, because of the property of being a normed division algebra that the octonion algebra has.

Thus, the multidimensional algebras can constitute a more general framework for neural networks, which could benefit not only from increasing the number of hidden layers and making the architecture ever more complicated, but also from increasing the dimensionality of the data that is being handled by the network. The multidimensional neural networks field is rather new, and it is expected that the future will bring even more applications for this type of networks.

6.1 Octonion-valued feedforward neural networks

We will first give the definition and some properties of the algebra of octonions.

The algebra of octonions is defined as

$$\mathbb{O} := \left\{ x = \sum_{p=0}^{7} [x]_p e_p \,\middle|\, [x]_0, [x]_1, \ldots, [x]_7 \in \mathbb{R} \right\},$$

in which e_p are the octonion units, $0 \leq p \leq 7$.

Octonion addition is defined by $x + y = \sum_{p=0}^{7} ([x]_p + [y]_p) e_p$. Scalar multiplication is given by $\alpha x = \sum_{p=0}^{7} (\alpha [x]_p) e_p$, and octonion multiplication is given by the multiplication of the octonion units:

$\times$	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7
e_0	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7
e_1	e_1	$-e_0$	e_3	$-e_2$	e_5	$-e_4$	$-e_7$	e_6
e_2	e_2	$-e_3$	$-e_0$	e_1	e_6	e_7	$-e_4$	$-e_5$
e_3	e_3	e_2	$-e_1$	$-e_0$	e_7	$-e_6$	e_5	$-e_4$
e_4	e_4	$-e_5$	$-e_6$	$-e_7$	$-e_0$	e_1	e_2	e_3
e_5	e_5	e_4	$-e_7$	e_6	$-e_1$	$-e_0$	$-e_3$	e_2
e_6	e_6	e_7	e_4	$-e_5$	$-e_2$	e_3	$-e_0$	$-e_1$
e_7	e_7	$-e_6$	e_5	e_4	$-e_3$	$-e_2$	e_1	$-e_0$

These operations make $\mathbb{O}$ a real algebra. From the multiplication table, we have that $e_i e_j = -e_j e_i \neq e_j e_i,\ \forall i \neq j,\ 0 < i, j \leq 7$, from which we deduce that $\mathbb{O}$ is not commutative, and that $(e_i e_j) e_k = -e_i (e_j e_k) \neq e_i (e_j e_k)$, for i, j, k distinct, $0 < i, j, k \leq 7$, or $e_i e_j \neq \pm e_k$, which allows us to see that $\mathbb{O}$ is not associative.

The conjugate of an octonion x is defined as $\overline{x} = [x]_0 e_0 - \sum_{p=1}^{7} [x]_p e_p$, its norm as $|x| = \sqrt{x\overline{x}} = \sqrt{\sum_{p=0}^{7} [x]_p^2}$, and its inverse as $x^{-1} = \frac{\overline{x}}{|x|^2}$. We can now see that $\mathbb{O}$ is a normed division algebra, and it can be proved that the only three division algebras that can be defined over the reals are the complex, quaternion, and octonion algebras.

By the Cayley–Dickson construction, the octonion units can be written as

$$e_0 = 1, \quad e_1 = \imath, \quad e_2 = \jmath, \quad e_3 = \imath\jmath, \quad e_4 = \ell, \quad e_5 = \imath\ell, \quad e_6 = \jmath\ell, \quad e_7 = \imath\jmath\ell,$$

where $\mathbb{C} := \{ z = [z]_0 + [z]_1 \imath \,|\, [z]_0, [z]_1 \in \mathbb{R},\ \imath^2 = -1 \}$ is the complex number set. This means that any $x \in \mathbb{O}$ can be written as

$$x = [x']_0 + [x']_1 \jmath + [x']_2 \ell + [x']_3 \jmath\ell,$$

where $[x']_0 = [x]_0 + [x]_1\imath,\ [x']_1 = [x]_2 + [x]_3\imath,\ [x']_2 = [x]_4 + [x]_5\imath,\ [x']_3 = [x]_6 + [x]_7\imath \in \mathbb{C}$. The product of two octonions $x, y \in \mathbb{O}$ written in this way is given by

$$\begin{aligned} xy &= [x']_0[y']_0 - [x']_1[y']_1 - [x']_2[y']_2 - [x']_3[y']_3 \\ &\quad +([x']_0[y']_1 + [x']_1[y']_0 + [x']_2[y']_3 - [x']_3[y']_2)\jmath \\ &\quad +([x']_0[y']_2 - [x']_1[y']_3 + [x']_2[y']_0 + [x']_3[y']_1)\ell \\ &\quad +([x']_0[y']_3 + [x']_1[y']_2 - [x']_2[y']_1 + [x']_3[y']_0)\jmath\ell. \end{aligned}$$

The derivative of an octonion-valued function $x : \mathbb{R}^+ \to \mathbb{O}$ is defined as the octonion formed by the derivatives of each element $[x(t)]_p$ of the octonion $x(t)$ with respect to t: $\dot{x}(t) = \frac{dx(t)}{dt} := \sum_{p=0}^{7} \frac{d([x]_p)}{dt} e_p, \forall t \in \mathbb{R}^+$.

The presentation in this section follows that in the author's paper [149].

6.1.1 Model formulation

In what follows, we will define feedforward neural networks for which the inputs, outputs, weights, and biases are all from $\mathbb{O}$, which means that they are octonions. Let's assume we have a fully connected feedforward neural network with values from $\mathbb{O}$, with L layers, where 1 is the input layer, L is the output layer, and the layers denoted by $\{2, \ldots, L-1\}$ are hidden layers. The error function $E : \mathbb{O}^N \to \mathbb{R}$ for such a network is

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{c} (y_i^L - t_i)\overline{(y_i^L - t_i)}, \tag{6.1.1}$$

where $\overline{y}$ is the conjugate of the octonion y. $\mathbf{y}^L = (y_i^L)_{1 \leq i \leq c} \in \mathbb{O}^c$ represents the vector of outputs of the network, $\mathbf{t} = (t_i)_{1 \leq i \leq c} \in \mathbb{O}^c$ represents the vector of targets of the network, and $\mathbf{w} \in \mathbb{O}^N$ represents the vector of the N weights and biases of the network, all being vectors whose components are octonions.

If we denote by $w_{jk}^l \in \mathbb{O}$ the weight connecting neuron j from layer l with neuron k from layer $l-1$, for all $l \in \{2, \ldots, L\}$, we can define the update step of weight w_{jk}^l in epoch t as being $\Delta w_{jk}^l(t) = w_{jk}^l(t+1) - w_{jk}^l(t)$. With this notation, the gradient descent method has the following update rule for the weight $w_{jk}^l \in \mathbb{O}$: $\Delta w_{jk}^l(t) = -\varepsilon \left(\sum_{a=0}^{7} \frac{\partial E}{\partial [w_{jk}^l]_a}(t) e_a \right)$, where ε is a real number representing the learning rate, and we denoted by $\frac{\partial E}{\partial [w_{jk}^l]_a}(t)$ the partial derivative of the error function E with respect to each element $[w_{jk}^l]_a$ of the octonion $w_{jk}^l \in \mathbb{O}$, where $0 \leq a \leq 7$. Thus, we need to compute the partial derivatives $\frac{\partial E}{\partial [w_{jk}^l]_a}(t)$. For this, we will make the following notations

$$s_j^l = \sum_k w_{jk}^l x_k^{l-1}, \tag{6.1.2}$$

$$y_j^l = G^l(s_j^l), \tag{6.1.3}$$

where equation (6.1.2) shows that the multiplication from the real-valued case is replaced by the octonion multiplication, G^l represents the activation function for the layer $l \in \{2, \ldots, L\}$, $\mathbf{x}^1 = (x_k^1)_{1 \leq k \leq d} \in \mathbb{O}^d$ is the vector of inputs of the network, and we have that $x_k^l := y_k^l$, $\forall l \in \{2, \ldots, L-1\}$, $\forall k$, because x_k^1 are the inputs, y_k^L are the outputs, and $y_k^l = x_k^l$ are the outputs of layer l, which are also inputs to layer $l+1$. The activation function is considered to be defined element-wise. For instance, for the octonion $x = \sum_{a=0}^{7} [x]_a e_a$, an example of activation function is the element-wise hyperbolic tangent function defined by $G\left(\sum_{a=0}^{7} [x]_a e_a\right) = \sum_{a=0}^{7} \left(\tanh [x]_a\right) e_a$.

We will first compute the update rule for the weights between layer $L-1$ and output layer L, i.e. $\Delta w_{jk}^L(t) = -\varepsilon \left(\sum_{a=0}^7 \frac{\partial E}{\partial [w_{jk}^L]_a} e_a \right)$. Using the chain rule, we can write $\forall 0 \leq a \leq 7$:

$$\frac{\partial E}{\partial [w_{jk}^L]_a} = \sum_{b=0}^7 \frac{\partial E}{\partial [s_j^L]_b} \frac{\partial [s_j^L]_b}{\partial [w_{jk}^L]_a}. \tag{6.1.4}$$

To compute $\frac{\partial [s_j^L]_b}{\partial [w_{jk}^L]_a}$, we need an explicit formula for $[s_j^L]_b$, which can be easily deduced from (6.1.2): $[s_j^L]_b = \left[\sum_k w_{jk}^L x_k^{L-1} \right]_b$, $\forall 0 \leq b \leq 7$. Now, we can easily see that

$$\frac{\partial [s_j^L]_b}{\partial [w_{jk}^L]_a} = \frac{\partial \left[\sum_k w_{jk}^L x_k^{L-1} \right]_b}{\partial [w_{jk}^L]_a} = \frac{\partial [w_{jk}^L x_k^{L-1}]_b}{\partial [w_{jk}^L]_a}. \tag{6.1.5}$$

Using the fact that $w_{jk}^L x_k^{L-1} = \sum_{0 \leq c,d \leq 7} [w_{jk}^L]_c [x_k^{L-1}]_d e_c e_d$, equation (6.1.5) can be written as

$$\begin{aligned} \frac{\partial [s_j^L]_b}{\partial [w_{jk}^L]_a} &= \frac{\partial \left[\sum_{0 \leq c,d \leq 7} [w_{jk}^L]_c [x_k^{L-1}]_d e_c e_d \right]_b}{\partial [w_{jk}^L]_a} = \frac{\partial \left(\sum_{\substack{0 \leq c,d \leq 7 \\ \kappa_{c,d} e_c e_d = e_b}} \kappa_{c,d} [w_{jk}^L]_c [x_k^{L-1}]_d \right)}{\partial [w_{jk}^L]_a} \\ &= \kappa_{a,d} [x_k^{L-1}]_d, \ \kappa_{a,d} e_a e_d = e_b, \ \kappa_{a,d} \in \{\pm 1\}. \end{aligned}$$

So, relation (6.1.4) can be written in the form

$$\frac{\partial E}{\partial [w_{jk}^L]_a} = \sum_{\substack{0 \leq b \leq 7 \\ \kappa_{a,d} e_a e_d = e_b}} \frac{\partial E}{\partial [s_j^L]_b} \kappa_{a,d} [x_k^{L-1}]_d. \tag{6.1.6}$$

Next, by denoting $\delta_j^L := \frac{\partial E}{\partial s_j^L}$, we have from the chain rule that $[\delta_j^L]_b = \frac{\partial E}{\partial [s_j^L]_b} = \sum_{0 \leq e \leq 7} \frac{\partial E}{\partial [y_j^L]_e} \frac{\partial [y_j^L]_e}{\partial [s_j^L]_b}$, $\forall 0 \leq b \leq 7$. Taking into account notation (6.1.3), and the expression of the error function given in (6.1.1), we have that

$$[\delta_j^L]_b = \sum_{0 \leq e \leq 7} ([y_j^L]_e - [t_j]_e) \frac{\partial [G^L(s_j^L)]_e}{\partial [s_j^L]_b} = ([y_j^L]_b - [t_j]_b) \frac{\partial [G^L(s_j^L)]_b}{\partial [s_j^L]_b},$$

$\forall 0 \leq b \leq 7$, because $[G^L(s_j^L)]_e$ depends upon $[s_j^L]_b$ only for $e = b$, which means that $\frac{\partial [G^L(s_j^L)]_e}{\partial [s_j^L]_b} = 0, \ \forall e \neq b$. If we denote by $\odot$ the element-wise multiplication of two octonions, the above relation gives

$$\delta_j^L = (y_j^L - t_j) \odot \frac{\partial G^L(s_j^L)}{\partial s_j^L}, \tag{6.1.7}$$

where $\frac{\partial G^L(s_j^L)}{\partial s_j^L}$ represents the octonion of element-wise derivatives of the activation function G^L. For instance, if $x = \sum_{a=0}^7 [x]_a e_a \in \mathbb{O}$, then $\frac{\partial G(x)}{\partial x} = \sum_{a=0}^7 \left(\operatorname{sech}^2 [x]_a \right) e_a$, with the function G defined as in the above example.

Finally, from (6.1.6), we get the expression for the desired update rule in the form: $\Delta w_{jk}^L(t) = -\varepsilon \delta_j^L \overline{x_k^{L-1}}$, where the octonion $\delta_j^L \in \mathbb{O}$ is given by relation (6.1.7).

Now, we will compute the update rule for an arbitrary weight w_{jk}^l, where $l \in \{2, \ldots, L-1\}$. First, we can write that $\Delta w_{jk}^l(t) = -\varepsilon \left(\sum_{a=0}^7 \frac{\partial E}{\partial [w_{jk}^l]_a} e_a \right)$, and then, from the chain rule, we have that

$$\frac{\partial E}{\partial [w_{jk}^l]_a} = \sum_{0 \leq b \leq 7} \frac{\partial E}{\partial [s_j^l]_b} \frac{\partial [s_j^l]_b}{\partial [w_{jk}^l]_a}. \tag{6.1.8}$$

$\forall 0 \leq a \leq 7$. Applying the chain rule again, we obtain that

$$\frac{\partial E}{\partial [s_j^l]_b} = \sum_r \sum_{0 \leq c \leq 7} \frac{\partial E}{\partial [s_r^{l+1}]_c} \frac{\partial [s_r^{l+1}]_c}{\partial [s_j^l]_b}, \tag{6.1.9}$$

$\forall 0 \leq b \leq 7$, where the sum is taken over all neurons r in layer $l+1$ to which neuron j from layer l sends connections. Next, we can write that $\frac{\partial [s_r^{l+1}]_c}{\partial [s_j^l]_b} = \sum_{0 \leq d \leq 7} \frac{\partial [s_r^{l+1}]_c}{\partial [y_j^l]_d} \frac{\partial [y_j^l]_d}{\partial [s_j^l]_b}$, $\forall 0 \leq b, c \leq 7$. Again from (6.1.2), we can compute

$$\frac{\partial [s_r^{l+1}]_c}{\partial [y_j^l]_d} = \frac{\partial \left[\sum_j w_{rj}^{l+1} y_j^l\right]_c}{\partial [y_j^l]_d} = \frac{\partial [w_{rj}^{l+1} y_j^l]_c}{\partial [y_j^l]_d}. \tag{6.1.10}$$

From $w_{rj}^{l+1} y_j^l = \sum_{0 \leq e,f \leq 7} [w_{rj}^{l+1}]_e [y_j^l]_f e_e e_f$, equation (6.1.10) can be written as

$$\begin{aligned} \frac{\partial [s_r^{l+1}]_c}{\partial [y_j^l]_d} &= \frac{\partial \left[\sum_{0 \leq e,f \leq 7} [w_{rj}^{l+1}]_e [y_j^l]_f e_e e_f\right]_c}{\partial [y_j^l]_d} = \frac{\partial (\sum_{\substack{0 \leq e,f \leq 7 \\ \kappa_{e,f} e_e e_f = e_c}} \kappa_{e,f} [w_{rj}^{l+1}]_e [y_j^l]_f)}{\partial [y_j^l]_d} \\ &= \kappa_{e,d} [w_{rj}^{l+1}]_e, \ \kappa_{e,d} e_e e_d = e_c, \ \kappa_{e,d} \in \{\pm 1\}, \end{aligned}$$

and then

$$\frac{\partial [s_r^{l+1}]_c}{\partial [s_j^l]_b} = \sum_{0 \leq d \leq 7} \kappa_{e,d} [w_{rj}^{l+1}]_e \frac{\partial [G^l(s_j^l)]_d}{\partial [s_j^l]_b} = \kappa_{e,b} [w_{rj}^{l+1}]_e \frac{\partial [G^l(s_j^l)]_b}{\partial [s_j^l]_b}, \ \kappa_{e,b} e_e e_b = e_c,$$

$\forall 0 \leq b, c \leq 7$, where again we took into account the fact that $\frac{\partial [G^l(s_j^l)]_d}{\partial [s_j^l]_b} = 0$, $\forall d \neq b$. Now, returning to equation (6.1.9), and putting it all together, we have that

$$\begin{aligned} \frac{\partial E}{\partial [s_j^l]_b} &= \sum_r \sum_{\substack{0 \leq c \leq 7 \\ \kappa_{e,b} e_e e_b = e_c}} \frac{\partial E}{\partial [s_r^{l+1}]_c} \kappa_{e,b} [w_{rj}^{l+1}]_e \frac{\partial [G^l(s_j^l)]_b}{\partial [s_j^l]_b} = \sum_r \left(\sum_{\substack{0 \leq c \leq 7 \\ \kappa_{e,b} e_e e_b = e_c}} \frac{\partial E}{\partial [s_r^{l+1}]_c} \cdot \right. \\ &\left. \cdot \ \kappa_{e,b} [w_{rj}^{l+1}]_e \frac{\partial [G^l(s_j^l)]_b}{\partial [s_j^l]_b} \right) = \sum_r \overline{[w_{rj}^{l+1}} \delta_r^{l+1}]_b \frac{\partial [G^l(s_j^l)]_b}{\partial [s_j^l]_b}, \end{aligned}$$

$\forall 0 \leq b \leq 7$. By denoting $\delta_j^l := \frac{\partial E}{\partial s_j^l}$, we can write the above relation in the form

$$\delta_j^l = \left(\sum_r \overline{w_{rj}^{l+1}} \delta_r^{l+1} \right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}. \tag{6.1.11}$$

Finally, taking into account the fact that $\frac{\partial [s_j^l]_b}{\partial [w_{jk}^l]_a} = \kappa_{a,d} [x_k^{l-1}]_d, \ \kappa_{a,d} e_a e_d = e_b$, relation (6.1.8) becomes $\frac{\partial E}{\partial [w_{jk}^l]_a} = \sum_{\substack{0 \leq b \leq 7 \\ \kappa_{a,d} e_a e_d = e_b}} \frac{\partial E}{\partial [s_j^l]_b} \kappa_{a,d} [x_k^{l-1}]_d = [\delta_j^l \overline{x_k^{l-1}}]_a$, $\forall 0 \leq a \leq 7$. Thus, the update rule for the weight w_{jk}^l can be written in octonion form in the following way: $\Delta w_{jk}^l(t) = -\varepsilon \delta_j^l \overline{x_k^{l-1}}$, which is similar to the formula we obtained for the layer L.

To summarize, we have the following formula for the update rule of the weight w_{jk}^l:

$$\Delta w_{jk}^l(t) = -\varepsilon \delta_j^l \overline{x_k^{l-1}}, \ \forall l \in \{2, \ldots, L\},$$

where

$$\delta_j^l = \begin{cases} \left(\sum_r \overline{w_{rj}^{l+1}} \delta_r^{l+1} \right) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l \leq L-1 \\ (y_j^l - t_j) \odot \frac{\partial G^l(s_j^l)}{\partial s_j^l}, & l = L \end{cases}.$$

6.1.2 Experimental results

6.1.2.1 Synthetic function approximation problem I

The first function we will test the proposed algorithm on is the simple quadratic function $f_1(o_1, o_2) = \frac{1}{6}(o_1^2 + o_2^2)$. This function was used to test the performance of different complex-valued neural network architectures and learning algorithms, for example in [191, 190, 205], and so we decided to test the octonion-valued algorithms on it, also.

For training of the octonion-valued neural network, we generated 1500 octonion training samples with random elements between 0 and 1. The testing set contained 500 samples generated in the same way. The network had 15 neurons on a single hidden layer. The activation function for the hidden layer was the element-wise hyperbolic tangent function given by $G^2\left(\sum_{a=0}^{7}[x]_a e_a\right) = \sum_{a=0}^{7}\left(\tanh[x]_a\right)e_a$, and for the output layer, the activation function was the identity function: $G^3(S) = S$.

The experiment showed that the neural network converges, and the mean squared error (MSE) for the training set was 0.000733 and for the test set was 0.000651. Training was done for 5000 epochs. Although the result is not spectacular, we must take into account the fact that each octonion is formed of 8 real numbers.

6.1.2.2 Synthetic function approximation problem II

A more complicated example, which involves four input variables and the reciprocal of one of the variables, is given by the following function: $f_2(o_1,o_2,o_3,o_4) = \frac{1}{1.5}\left(o_3 + 10o_1o_4 + \frac{o_2^2}{o_1}\right)$, which was used as a benchmark in [191, 192] for complex-valued neural networks, so we used it for octonion-valued neural networks, also. The training and testing sets were randomly generated octonions with elements between 0 and 1, 1500 for the training set, and 500 for the test set. The activation functions were the same as the ones above. The architecture had 15 neurons on a single hidden layer, and the network was trained for 5000 epochs.

In this experiment, the training and testing MSE had similar values, and equal approximately with 0.0071. The performance is worse than the one obtained in the previous experiment, but in this case, the function was more complicated. These results give reasons for hope that in the future these networks can be optimized to perform better on octonion-valued function approximation problems.

6.1.2.3 Linear time series prediction

A possible application of octonion-valued neural networks is in signal processing. A known benchmark proposed in [120], and used in [58, 61, 228] for complex-valued neural networks, is the prediction of the white noise $n(k)$, passed through the stable autoregressive filter given by $y(k) = 1.79y(k-1) - 1.85y(k-2) + 1.27y(k-3) - 0.41y(k-4) + n(k)$. In the octonion setting, the octonion-valued white noise $n(k)$ is given by $n(k) = \sum_{a=0}^{7}[n(k)]_a e_a$, where $[n(k)]_a \sim \mathcal{N}(0,1)$, $\forall 0 \leq a \leq 7$.

The tap input of the filter was 4, so the networks had 4 inputs, 4 hidden neurons on a single hidden layer, and one output. The activation function for the hidden layer was the element-wise hyperbolic tangent function and for the output layer was the identity. Training was done for 5000 epochs with 2500 training samples.

We use a measure of performance called *prediction gain*, defined by $R_p = 10\log_{10}\frac{\sigma_x^2}{\sigma_e^2}$, where σ_x^2 represents the variance of the input signal and σ_e^2 represents the variance of the prediction error. The prediction gain is given in dB. It is obvious that, because of the way it is

defined, a bigger prediction gain means better performance. The network obtained a prediction gain of 0.485.

6.2 Octonion-valued bidirectional associative memories

Bidirectional associative memories represent an extension of the unidirectional Hopfield neural networks, and were first introduced by Kosko in [100]. Since then, they were applied in pattern recognition and automatic control, for example. Complex-valued bidirectional associative memories were introduced in [110], quaternion-valued bidirectional associative memories in [102], and Clifford-valued bidirectional associative memories in [217]. Taking these facts into account, and also the ones mentioned in Section 6.1, we introduce octonion-valued bidirectional associative memories, which could be applied to store octonion patterns and to solve octonion optimization problems.

The presentation in this section follows that in the author's paper [156].

6.2.1 Main results

We introduce octonion-valued bidirectional associative memories for which the states, outputs, and thresholds are all from $\mathbb{O}$.

The following set of differential equations describes this type of networks:

$$\begin{cases} \tau_i \dfrac{dx_i(t)}{dt} = -x_i(t) + \displaystyle\sum_{j=1}^{P} w_{ij} f(y_j(t)) + a_i, \quad \forall i \in \{1, \ldots, N\}, \\ \upsilon_j \dfrac{dy_j(t)}{dt} = -y_j(t) + \displaystyle\sum_{i=1}^{N} w_{ji} f(x_i(t)) + b_j, \quad \forall j \in \{1, \ldots, P\}. \end{cases} \tag{6.2.1}$$

$\tau_i \in \mathbb{R},\ \tau_i > 0$ and $\upsilon_j \in \mathbb{R},\ \upsilon_j > 0$ represent the time constants of neurons x_i and y_j, respectively, $x_i(t),\ y_j(t) \in \mathbb{O}$ are the states of neurons x_i and y_j at time t, respectively, $w_{ij} \in \mathbb{O}$ is the weight connecting neuron x_i to neuron y_j, $f : \mathbb{O} \to \mathbb{O}$ is the nonlinear octonion-valued activation function, and a_i and b_j are the thresholds of neurons x_i and y_j, respectively, $\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, P\}$. The derivative $\frac{dx_i(t)}{dt}$ is considered to be the octonion formed by the derivatives of each element $[x_i(t)]_a$ of the octonion $x_i(t)$ with respect to t:

$$\frac{dx_i(t)}{dt} := \sum_{a=0}^{7} \frac{d([x_i]_a)}{dt} e_a.$$

Thus, the above differential equations have values in $\mathbb{O}$, and the multiplication between the weights and the values of the activation function is the octonion multiplication. Making the notations $u_i(t) := f(x_i(t))$ and $v_j(t) := f(y_j(t))$ for the output of neurons x_i and y_j, respectively, system (6.2.1) can be written as:

$$\begin{cases} \tau_i \dfrac{dx_i(t)}{dt} = -x_i(t) + \displaystyle\sum_{j=1}^{P} w_{ij} v_j(t) + a_i, \ \forall i \in \{1, \ldots, N\}, \\ \upsilon_j \dfrac{dy_j(t)}{dt} = -y_j(t) + \displaystyle\sum_{i=1}^{N} w_{ji} u_i(t) + b_j, \ \forall j \in \{1, \ldots, P\}. \end{cases}$$

We assume that the activation function f is formed of 8 functions $f^a : \mathbb{O} \to \mathbb{R}$, $0 \leq a \leq 7$, thus having the expression:

$$f(x) = \sum_{a=0}^{7} f^a(x) e_a.$$

In order to define an energy function for the network (6.2.1), we need to make a series of assumptions, which will be detailed below.

The first assumption is that the function f is bounded: $\exists M > 0$, $|f(x)| \leq M$, $\forall x \in \mathbb{O}$, and that the functions f^a are continuously differentiable with respect to each $[x]_b$, $\forall 0 \leq b \leq 7$, $\forall 0 \leq a \leq 7$. This allows us to define the 8×8 Jacobian matrix of the function f as

$$\mathbf{Jac}_f(x) = \left(\frac{\partial f^a(x)}{\partial [x]_b} \right)_{\substack{0 \leq a \leq 7 \\ 0 \leq b \leq 7}}.$$

The second assumption is that $\mathbf{Jac}_f(x)$ is symmetric and positive definite, $\forall x \in \mathbb{O}$, and that the function f is injective.

The above assumptions ensure the existence of the inverse function $g : \mathbb{O} \to \mathbb{O}$ of f, $g = f^{-1}$. We can thus write $g(u_i(t)) = x_i(t)$, $\forall i \in \{1, \ldots, N\}$, and $g(v_j(t)) = y_j(t)$, $\forall j \in \{1, \ldots, P\}$. Now, we will construct a function $G : \mathbb{O} \to \mathbb{R}$, which satisfies

$$\frac{\partial G(x)}{\partial [x]_a} = g^a(x), \ \forall 0 \leq a \leq 7, \tag{6.2.2}$$

where $g^a : \mathbb{O} \to \mathbb{R}$ are the component functions of g, i.e.,

$$g(x) = \sum_{a=0}^{7} g^a(x) e_a.$$

It can be proved that the function

$$G(x) = \sum_{a=0}^{7} \int_0^{[x]_a} g^a(y^a) dy,$$

satisfies the conditions (6.2.2), where the octonions y^a have the following form

$$[y^a]_b = \begin{cases} [x]_b, & b < a \\ y & b = a \\ 0, & b > a \end{cases}, \ \forall 0 \leq a, b \leq 7.$$

We can write the conditions (6.2.2) in octonion form as:

$$\frac{\partial G(x)}{\partial x} := \sum_{a=0}^{7} \frac{\partial G(x)}{\partial [x]_a} e_a = \sum_{a=0}^{7} g^a(x) e_a = g(x). \tag{6.2.3}$$

The third assumption is made about the weights of the network, which must satisfy:

$$w_{ji} = \overline{w_{ij}}, \ \forall i \in \{1, \ldots, N\}, \ \forall j \in \{1, \ldots, P\}.$$

At this point, we can define an energy function for network (6.2.1). A function $E : \mathbb{O}^{N+P} \to \mathbb{R}$ is an energy function for the network (6.2.1) if the derivative of E along the trajectories of

the network, denoted by $\frac{dE(\mathbf{u}(t),\mathbf{v}(t))}{dt}$, satisfies the condition $\frac{dE(\mathbf{u}(t),\mathbf{v}(t))}{dt} \leq 0$ and $\frac{dE(\mathbf{u}(t),\mathbf{v}(t))}{dt} = 0 \Leftrightarrow \frac{du_i(t)}{dt} = \frac{dv_j(t)}{dt} = 0, \forall i \in \{1,\ldots,N\}, \forall j \in \{1,\ldots,P\}$.

In the following, we will show that the function $E : \mathbb{O}^{N+P} \to \mathbb{R}$, defined by

$$\begin{aligned} E(\mathbf{u}(t),\mathbf{v}(t)) \;:=\; & -\sum_{i=1}^{N}\sum_{j=1}^{P} \mathrm{Re}\left(\overline{u_i(t)}\,(w_{ij}v_j(t))\right) \\ & +\sum_{i=1}^{N} G(u_i(t)) - \sum_{i=1}^{N} \mathrm{Re}\left(\overline{a_i}u_i(t)\right) \\ & +\sum_{j=1}^{P} G(v_j(t)) - \sum_{j=1}^{P} \mathrm{Re}\left(\overline{b_j}v_j(t)\right), \end{aligned} \tag{6.2.4}$$

is an energy function for the network (6.2.1), if the above assumptions are satisfied, where $\mathrm{Re}(x)$ represents the real part of octonion $x = \sum_{a=0}^{7}[x]_a e_a \in \mathbb{O}$, i.e., $\mathrm{Re}(x) = [x]_0$. The brackets in the first term of E are necessary, because, as noted above, the algebra $\mathbb{O}$ of octonions is not associative. Nonetheless, it can be proved that

$$\mathrm{Re}(x(yz)) = \mathrm{Re}((xy)z),\ \forall x,y,z \in \mathbb{O},$$

so the other choice of brackets would have yielded the same expression.

We start by applying the chain rule:

$$\begin{aligned} \frac{dE(\mathbf{u}(t),\mathbf{v}(t))}{dt} \;=\; & \sum_{i=1}^{N}\sum_{a=0}^{7} \frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial [u_i(t)]_a}\frac{d[u_i(t)]_a}{dt} \\ & +\sum_{j=1}^{P}\sum_{b=0}^{7} \frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial [v_j(t)]_a}\frac{d[v_j(t)]_a}{dt} \\ =\; & \sum_{i=1}^{N} \mathrm{Re}\left(\overline{\left(\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial u_i(t)}\right)}\frac{du_i(t)}{dt}\right) \\ & +\sum_{j=1}^{P} \mathrm{Re}\left(\overline{\left(\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial v_j(t)}\right)}\frac{dv_j(t)}{dt}\right), \end{aligned} \tag{6.2.5}$$

where we denoted by

$$\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial u_i(t)} := \sum_{a=0}^{7}\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial [u_i(t)]_a}e_a,$$

$$\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial v_j(t)} := \sum_{a=0}^{7}\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial [v_j(t)]_a}e_a.$$

Now, we can prove by direct computation that

$$\frac{d\mathrm{Re}(\overline{a}x)}{dx} = a,\ \forall a,x \in \mathbb{O},$$

$$\frac{d\mathrm{Re}(\overline{x}(ay))}{dx} = ay,\ \forall x,a,y \in \mathbb{O},$$

$$\frac{d\text{Re}(\overline{x}(ay))}{dy} = \overline{a}x,\ \forall x, a, y \in \mathbb{O},$$

where the derivatives are defined as in relation (6.2.3). If we also take into account the assumption $w_{ji} = \overline{w_{ij}}$, relation (6.2.3), and the set of equations given by (6.2.1), we can deduce from (6.2.4) the following expressions for the partial derivatives $\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial u_i(t)}$ and $\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial v_j(t)}$:

$$\begin{aligned}
\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial u_i(t)} &= -\sum_{j=1}^{P} w_{ij}v_j(t) + g(u_i(t)) - a_i \\
&= -\left(\sum_{j=1}^{P} w_{ij}v_j(t) - x_i(t) + a_i\right) \\
&= -\tau_i \frac{dx_i(t)}{dt},\ \forall i \in \{1,\ldots,N\},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E(\mathbf{u}(t),\mathbf{v}(t))}{\partial v_j(t)} &= -\sum_{i=1}^{N} \overline{w_{ij}}u_i(t) + g(v_j(t)) - b_j \\
&= -\sum_{i=1}^{N} w_{ji}u_i(t) + g(v_j(t)) - b_j \\
&= -\left(\sum_{i=1}^{N} w_{ji}u_j(t) - y_j(t) + b_j\right) \\
&= -\upsilon_j \frac{dy_j(t)}{dt},\ \forall j \in \{1,\ldots,P\}.
\end{aligned}$$

Plugging these expressions back into relation (6.2.5), we have:

$$\begin{aligned}
\frac{dE(\mathbf{u}(t),\mathbf{v}(t))}{dt} &= \sum_{i=1}^{N} \text{Re}\left(\overline{\left(-\tau_i \frac{dx_i(t)}{dt}\right)}\frac{du_i(t)}{dt}\right) \\
&\quad + \sum_{j=1}^{P} \text{Re}\left(\overline{\left(-\upsilon_j \frac{dy_j(t)}{dt}\right)}\frac{dv_j(t)}{dt}\right) \\
&= -\sum_{i=1}^{N} \tau_i \left[\text{vec}\left(\frac{dx_i(t)}{dt}\right)\right]^T \text{vec}\left(\frac{du_i(t)}{dt}\right) \\
&\quad - \sum_{j=1}^{P} \upsilon_j \left[\text{vec}\left(\frac{dy_j(t)}{dt}\right)\right]^T \text{vec}\left(\frac{dv_j(t)}{dt}\right),
\end{aligned} \tag{6.2.6}$$

where we denoted by vec (x) the vectorization of octonion x, i.e.,

$$\text{vec}(x) := ([x]_0, [x]_1, \ldots, [x]_7)^T,\ \forall x \in \mathbb{O}.$$

We also used the identity

$$\text{Re}(\overline{a}b) = \text{vec}(a)^T \text{vec}(b),\ \forall a, b \in \mathbb{O}.$$

From $g(u_i(t)) = x_i(t)$ and $g(v_j(t)) = y_j(t)$, we can obtain that

$$\text{vec}\left(\frac{dg(u_i(t))}{dt}\right) = \mathbf{Jac}_g(u_i(t))\text{vec}\left(\frac{du_i(t)}{dt}\right),$$

$$\text{vec}\left(\frac{dg(v_j(t))}{dt}\right) = \mathbf{Jac}_g(v_j(t))\text{vec}\left(\frac{dv_j(t)}{dt}\right),$$

$\forall i \in \{1,\dots,N\}, \forall j \in \{1,\dots,P\}$. The second assumption made above tells us that $\mathbf{Jac}_f(x)$ is symmetric and positive definite, and so $\mathbf{Jac}_g(u)$ is also symmetric and positive definite. This means that

$$\left[\text{vec}\left(\frac{du_i(t)}{dt}\right)\right]^T [\mathbf{Jac}_g(u_i(t))]^T \text{vec}\left(\frac{du_i(t)}{dt}\right) \geq 0,$$

$$\left[\text{vec}\left(\frac{dv_j(t)}{dt}\right)\right]^T [\mathbf{Jac}_g(v_j(t))]^T \text{vec}\left(\frac{dv_j(t)}{dt}\right) \geq 0,$$

$\forall i \in \{1,\dots,N\}, \forall j \in \{1,\dots,P\}$, and thus relation (6.2.6) becomes

$$\begin{aligned}
\frac{dE(\mathbf{u}(t),\mathbf{v}(t))}{dt} &= -\sum_{i=1}^{N} \tau_i \left[\text{vec}\left(\frac{dg(u_i(t))}{dt}\right)\right]^T \text{vec}\left(\frac{du_i(t)}{dt}\right) \\
&\quad -\sum_{j=1}^{P} \upsilon_j \left[\text{vec}\left(\frac{dg(v_j(t))}{dt}\right)\right]^T \text{vec}\left(\frac{dv_j(t)}{dt}\right), \\
&= -\sum_{i=1}^{N} \left\{\tau_i \left[\text{vec}\left(\frac{du_i(t)}{dt}\right)\right]^T [\mathbf{Jac}_g(u_i(t))]^T \text{vec}\left(\frac{du_i(t)}{dt}\right)\right\} \\
&\quad -\sum_{j=1}^{P} \left\{\upsilon_j \left[\text{vec}\left(\frac{dv_j(t)}{dt}\right)\right]^T [\mathbf{Jac}_g(v_j(t))]^T \text{vec}\left(\frac{dv_j(t)}{dt}\right)\right\} \\
&\leq 0,
\end{aligned}$$

Equality is attained when $\frac{dE(\mathbf{u}(t),\mathbf{v}(t))}{dt} = 0 \Leftrightarrow \text{vec}\left(\frac{du_i(t)}{dt}\right) = \text{vec}\left(\frac{dv_j(t)}{dt}\right) = 0 \Leftrightarrow \frac{du_i(t)}{dt} = \frac{dv_j(t)}{dt} = 0,\ \forall i \in \{1,\dots,N\},\ \forall j \in \{1,\dots,P\}$, thus ending the proof that E is indeed an energy function for the network (6.2.1).

Possible examples of activation functions, inspired by the ones used in real-valued and complex-valued neural networks, are:

$$f(v) = \frac{v}{1+||v||},\ \forall v \in \mathbb{O},$$

$$f\left(\sum_{a=0}^{7}[v]_a e_a\right) = \sum_{a=0}^{7}(\tanh[v]_a)e_a,\ \forall v \in \mathbb{O}.$$

The first one corresponds to the fully complex activation functions, while the second one corresponds to the split complex activation functions from the complex-valued domain.

6.3 Asymptotic stability for OVNNs with delay

The practical applications of neural networks heavily rely on their dynamical properties. To solve problems of optimization, neural control, and signal processing, neural networks have to

be designed in such way that they exhibit only one globally stable steady state. Thus, sufficient conditions can be determined, which depend on the system parameters, that guarantee the existence of a unique globally stable steady state for a certain neural network.

At the beginning of the 1980's, Hopfield had the idea of introducing an energy function in order to study the dynamics of fully connected recurrent neural networks, see [80, 81]. He showed that combinatorial problems can be solved by using this type of networks. Since then, Hopfield neural networks have been applied to the synthesis of associative memories, image processing, speech processing, control systems, signal processing, pattern matching, etc. Because of the finite switching speed of amplifiers, delays occur in real-life implementations of neural networks, and can cause unstable or oscillatory behavior. This fact lead to the definition of Hopfield neural networks with different types of delays.

Generalizations of the Hopfield neural networks to multidimensional domains appeared over the last few years. Complex-valued Hopfield networks were discussed in [115, 199, 198], quaternion-valued Hopfield networks in [117, 216], and Clifford-valued Hopfield networks in [116, 240]. Taking these facts into account, we introduce octonion-valued Hopfield neural networks with delay, which could be applied to solve octonion optimization problems.

In this and the next sections of this chapter, the following notations will be used: $\mathbb{R}$ is the real number set, $\mathbb{C}$ is the complex number set, and $\mathbb{O}$ is the octonion number set. $\mathbb{R}^N$ (respectively, $\mathbb{C}^N$ and $\mathbb{O}^N$) denotes the set of real-valued (respectively, complex-valued and octonion-valued) N-dimensional vectors, and $\mathbb{R}^{N\times N}$ (respectively, $\mathbb{C}^{N\times N}$ and $\mathbb{O}^{N\times N}$) denotes the set of real-valued (respectively, complex-valued and octonion-valued) matrices of dimension $N \times N$. $|\cdot|$ represents the octonion norm, and $||\cdot||$ stands for the Euclidean vector norm or the matrix Frobenius norm. $A > 0$ ($A < 0$) means that matrix A is positive definite (negative definite). $(\cdot)^T$ is the transpose of a real-valued vector or matrix. $(\cdot)^H$ is the Hermitian transpose or complex conjugate transpose of a complex-valued vector or matrix. $*$ denotes the symmetric terms in a matrix. I_N is the identity matrix of dimension $N \times N$. Finally, $\lambda_{\min}(P)$ denotes the smallest eigenvalue of positive definite matrix P.

The presentation in this section follows that in the author's paper [157].

6.3.1 Main results

We introduce octonion-valued Hopfield neural networks for which the states and weights are from $\mathbb{O}$. The following set of differential equations describes this type of networks:

$$\dot{x}_i(t) = -d_i x_i(t) + \sum_{j=1}^{N} a_{ij} f_j(x_j(t)) + \sum_{j=1}^{N} b_{ij} g_j(x_j(t-\tau)) + u_i, \tag{6.3.1}$$

for $i \in \{1, \ldots, N\}$, where $x_i(t) \in \mathbb{O}$ is the state of neuron i at time t, $d_i \in \mathbb{R}$, $d_i > 0$, is the self-feedback connection weight of neuron i, $a_{ij} \in \mathbb{O}$ is the weight connecting neuron j to neuron i without delay, $b_{ij} \in \mathbb{O}$ is the weight connecting neuron j to neuron i with delay, $f_j : \mathbb{O} \to \mathbb{O}$ is the nonlinear octonion-valued activation function of neuron j without delay, $g_j : \mathbb{O} \to \mathbb{O}$ is the nonlinear octonion-valued activation function of neuron j with delay, $\tau \in \mathbb{R}$ is the delay and we assume $\tau > 0$, and $u_i \in \mathbb{O}$ is the external input of neuron i, $\forall i, j \in \{1, \ldots, N\}$.

The definition of the derivative $\dot{x}_i(t)$ is the one given in Section (6.1), $\forall i \in \{1, \ldots, N\}$. Thus, the above set of differential equations has values in $\mathbb{O}$, and the multiplication between the weights and the values of the activation functions is the octonion multiplication.

We need to make an assumption about the activation functions, in order to study the stability of the above defined network.

Assumption 6.1. *The octonion-valued activation functions f_j and g_j satisfy the following Lipschitz conditions:*

$$||f_j(x) - f_j(x')|| \leq l_j^f ||x - x'||, \ \forall x, x' \in \mathbb{O},$$

$$||g_j(x) - g_j(x')|| \leq l_j^g ||x - x'||, \ \forall x, x' \in \mathbb{O},$$

where $l_j^f > 0$ and $l_j^g > 0$ are the Lipschitz constants, $\forall j \in \{1, \ldots, N\}$. Moreover, we denote $\overline{L_f} = diag(l_1^f I_8, l_2^f I_8, \ldots, l_N^f I_8)$, $\overline{L_g} = diag(l_1^g I_8, l_2^g I_8, \ldots, l_N^g I_8)$.

We will first transform the octonion-valued differential equations (6.3.1) into real-valued ones. To do so, we will detail each equation in (6.3.1) into 8 real-valued equations:

$$\begin{aligned}[\dot{x}_i(t)]_p &= -d_i[x_i(t)]_p + \sum_{j=1}^{N}\sum_{q=0}^{7}[a_{ij}]_{pq}[f_j(x_j(t))]_q \\ &\quad + \sum_{j=1}^{N}\sum_{q=0}^{7}[b_{ij}]_{pq}[g_j(x_j(t-\tau))]_q + [u_i]_p, \end{aligned} \tag{6.3.2}$$

for $0 \leq p \leq 7$, $i \in \{1, \ldots, N\}$, where $[x]_{pq}$ is an element of the matrix $\text{mat}(x)$, defined by

$$\text{mat}(x) := \begin{bmatrix} [x]_0 & -[x]_1 & -[x]_2 & -[x]_3 & -[x]_4 & -[x]_5 & -[x]_6 & -[x]_7 \\ [x]_1 & [x]_0 & -[x]_3 & [x]_2 & -[x]_5 & [x]_4 & [x]_7 & -[x]_6 \\ [x]_2 & [x]_3 & [x]_0 & -[x]_1 & -[x]_6 & -[x]_7 & [x]_4 & [x]_5 \\ [x]_3 & -[x]_2 & [x]_1 & [x]_0 & -[x]_7 & [x]_6 & -[x]_5 & -[x]_4 \\ [x]_4 & [x]_5 & [x]_6 & [x]_7 & [x]_0 & -[x]_1 & -[x]_2 & -[x]_3 \\ [x]_5 & -[x]_4 & [x]_7 & -[x]_6 & [x]_1 & [x]_0 & [x]_3 & -[x]_2 \\ [x]_6 & -[x]_7 & -[x]_4 & [x]_5 & [x]_2 & -[x]_3 & [x]_0 & [x]_1 \\ [x]_7 & [x]_6 & -[x]_5 & -[x]_4 & [x]_3 & [x]_2 & -[x]_1 & [x]_0 \end{bmatrix}.$$

Now, if we denote $\text{vec}(x) = ([x]_0, [x]_1, \ldots, [x]_7)^T$, the equations (6.3.2) can be written as

$$\begin{aligned}\text{vec}(\dot{x}_i(t)) &= -d_i I_8 \text{vec}(x_i(t)) + \sum_{j=1}^{N} \text{mat}(a_{ij})\text{vec}(f_j(x_j(t))) \\ &\quad + \sum_{j=1}^{N} \text{mat}(b_{ij})\text{vec}(g_j(x_j(t-\tau))) + \text{vec}(u_i), \end{aligned} \tag{6.3.3}$$

for $i \in \{1, \ldots, N\}$. Denoting $w(t) = (\text{vec}(x_1(t))^T, \text{vec}(x_2(t))^T, \ldots, \text{vec}(x_N(t))^T)^T$, $\overline{D} = \text{diag}(d_1 I_8, d_2 I_8, \ldots, d_N I_8)$, $\overline{A} = (\text{mat}(a_{ij}))_{1 \leq i,j \leq N}$, $\overline{B} = (\text{mat}(b_{ij}))_{1 \leq i,j \leq N}$,

$$\overline{f}(w(t)) = (\text{vec}(f_1(x_1(t)))^T, \text{vec}(f_2(x_2(t)))^T, \ldots, \text{vec}(f_N(x_N(t)))^T)^T,$$

$$\overline{g}(w(t-\tau)) = (\text{vec}(g_1(x_1(t-\tau)))^T, \text{vec}(g_2(x_2(t-\tau)))^T, \ldots, \text{vec}(g_N(x_N(t-\tau)))^T)^T,$$

$\overline{u} = (\text{vec}(u_1)^T, \text{vec}(u_2)^T, \ldots, \text{vec}(u_N)^T)^T$, with the simplifying notations $w = w(t)$ and $w^\tau = w(t-\tau)$, system (6.3.1) becomes

$$\dot{w} = -\overline{D}w + \overline{A}\,\overline{f}(w) + \overline{B}\,\overline{g}(w^\tau) + \overline{u}. \tag{6.3.4}$$

Remark 6.1. The system (6.3.4) is equivalent with the system (6.3.1), which means that any property proven about system (6.3.4) will also hold for system (6.3.1). Because of this, from now on we will only study the existence, uniqueness, and global asymptotic stability of the equilibrium point of system (6.3.4).

We will also need the following lemmas:

Lemma 6.1. *([55]). If $H(w) : \mathbb{R}^{8N} \to \mathbb{R}^{8N}$ is a continuous map that satisfies the following conditions:*

(i) $H(w)$ is injective on $\mathbb{R}^{8N}$,

(ii) $||H(w)|| \to \infty$ as $||w|| \to \infty$,

then $H(w)$ is a homeomorphism of $\mathbb{R}^{8N}$ onto itself.

Lemma 6.2. *([114]). For any vectors x, $y \in \mathbb{R}^{8N}$, positive definite matrix $P \in \mathbb{R}^{8N\times 8N}$, and real constant $\varepsilon > 0$, the following linear matrix inequality (LMI) holds:*

$$2x^T y \leq \varepsilon x^T P x + \frac{1}{\varepsilon} y^T P^{-1} y.$$

Now, we give an LMI-based sufficient condition for the existence, uniqueness, and global asymptotic stability of the equilibrium point for (6.3.4).

Theorem 6.1. *If Assumption 6.1 holds, then system* (6.3.4) *has a unique equilibrium point which is globally asymptotically stable if there exist real numbers $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$, and positive definite matrix $P \in \mathbb{R}^{8N\times 8N}$ such that the following LMI holds*

$$\begin{bmatrix} P\overline{D} + \overline{D}P - \varepsilon_1 \overline{L_f}^T \overline{L_f} - \varepsilon_2 \overline{L_g}^T \overline{L_g} & P\overline{A} & P\overline{B} \\ * & \varepsilon_1 I_{8N} & 0 \\ * & * & \varepsilon_2 I_{8N} \end{bmatrix} > 0. \tag{6.3.5}$$

Proof. Define the function $H : \mathbb{R}^{8N} \to \mathbb{R}^{8N}$,

$$H(w) = -\overline{D}w + \overline{A}\,\overline{f}(w) + \overline{B}\,\overline{g}(w) + \overline{u}. \tag{6.3.6}$$

We will first prove that H is injective. Assume by contradiction that there exist w, $w' \in \mathbb{R}^{8N}$, $w \neq w'$, such that $H(w) = H(w')$. This equality is equivalent with

$$-\overline{D}(w - w') + \overline{A}(\overline{f}(w) - \overline{f}(w')) + \overline{B}(\overline{g}(w) - \overline{g}(w')) = 0. \tag{6.3.7}$$

By left multiplying this relation by $2(w - w')^T P$, we get that

$$2(w - w')^T P(-\overline{D}(w - w') + \overline{A}(\overline{f}(w) - \overline{f}(w')) + \overline{B}(\overline{g}(w) - \overline{g}(w'))) = 0, \tag{6.3.8}$$

which can be rewritten as

$$\begin{aligned}(w - w')^T(-P\overline{D} - \overline{D}P)(w - w') + 2(w - w')^T P\overline{A}(\overline{f}(w) - \overline{f}(w')) \\ +2(w - w')^T P\overline{B}(\overline{g}(w) - \overline{g}(w')) = 0,\end{aligned} \tag{6.3.9}$$

From Assumption 6.1, we can deduce that

$$(f(w) - f(w'))^T(f(w) - f(w')) \leq (w - w')^T \overline{L_f}^T \overline{L_f}(w - w'), \tag{6.3.10}$$

$$(g(w)-g(w'))^T(g(w)-g(w')) \leq (w-w')^T\overline{L_g}^T\overline{L_g}(w-w'). \tag{6.3.11}$$

Now, taking into account Lemma 6.2 and inequalities (6.3.10) and (6.3.11), we have from (6.3.9) that

$$\begin{aligned}
& (w-w')^T(-P\overline{D}-\overline{D}P)(w-w')+2(w-w')^TP\overline{A}(\overline{f}(w)-\overline{f}(w')) \\
& +2(w-w')^TP\overline{B}(\overline{g}(w)-\overline{g}(w')) \\
\leq\ & (w-w')^T(-P\overline{D}-\overline{D}P)(w-w')+\varepsilon_1(\overline{f}(w)-\overline{f}(w'))^T(\overline{f}(w)-\overline{f}(w')) \\
& +\varepsilon_1^{-1}(w-w')^TP\overline{A}\,\overline{A}^TP(w-w')+\varepsilon_2(\overline{g}(w)-\overline{g}(w'))^T(\overline{g}(w)-\overline{g}(w')) \\
& +\varepsilon_2^{-1}(w-w')^TP\overline{B}\,\overline{B}^TP(w-w') \\
\leq\ & (w-w')^T(-P\overline{D}-\overline{D}P)(w-w')+\varepsilon_1(w-w')^T\overline{L_f}^T\overline{L_f}(w-w') \\
& +\varepsilon_1^{-1}(w-w')^TP\overline{A}\,\overline{A}^TP(w-w')+\varepsilon_2(w-w')^T\overline{L_g}^T\overline{L_g}(w-w') \\
& +\varepsilon_2^{-1}(w-w')^TP\overline{B}\,\overline{B}^TP(w-w') \\
=\ & -(w-w')^T(P\overline{D}+\overline{D}P-\varepsilon_1\overline{L_f}^T\overline{L_f}-\varepsilon_2\overline{L_g}^T\overline{L_g}-\varepsilon_1^{-1}P\overline{A}\,\overline{A}^TP \\
& -\varepsilon_2^{-1}P\overline{B}\,\overline{B}^TP)(w-w').
\end{aligned} \tag{6.3.12}$$

Using Schur's complement, from condition (6.3.5), we get that

$$P\overline{D}+\overline{D}P-\varepsilon_1\overline{L_f}^T\overline{L_f}-\varepsilon_2\overline{L_g}^T\overline{L_g}-\varepsilon_1^{-1}P\overline{A}\,\overline{A}^TP-\varepsilon_2^{-1}P\overline{B}\,\overline{B}^TP>0, \tag{6.3.13}$$

which, plugged back into (6.3.12), finally yields $H(w)-H(w')<0$, which is a contradiction with our initial assumption. We deduce that H is injective.

Next, we prove that $||H(w)|| \to \infty$ as $||w|| \to \infty$. To this end, we deduce from (6.3.13) that there exists a sufficiently small $\varepsilon > 0$, such that $-P\overline{D}-\overline{D}P+\varepsilon_1\overline{L_f}^T\overline{L_f}+\varepsilon_2\overline{L_g}^T\overline{L_g}+\varepsilon_1^{-1}P\overline{A}\,\overline{A}^TP+\varepsilon_2^{-1}P\overline{B}\,\overline{B}^TP<-\varepsilon I_{8N}$. Considering $w'=0$ in (6.3.12), we have

$$\begin{aligned}
2w^TP(H(w)-H(0)) \ \leq\ & w^T(-P\overline{D}-\overline{D}P+\varepsilon_1\overline{L_f}^T\overline{L_f}+\varepsilon_2\overline{L_g}^T\overline{L_g}+\varepsilon_1^{-1}P\overline{A}\,\overline{A}^TP \\
& +\varepsilon_2^{-1}P\overline{B}\,\overline{B}^TP)w<-\varepsilon||w||^2.
\end{aligned} \tag{6.3.14}$$

By applying the Cauchy-Schwarz inequality in relation (6.3.14), we get that

$$2||w||||P||(||H(w)||+||H(0)||)>\varepsilon||w||^2,$$

from which we conclude that $||H(w)|| \to \infty$ when $||w|| \to \infty$.

Now we can use Lemma 6.1 to deduce that H is a homeomorphism of $\mathbb{R}^{8N}$ onto itself. This means that the equation $H(w)=0$ has a unique solution, and so system (6.3.4) also has a unique equilibrium point, which we will denote by $\hat{w}$.

We shift this equilibrium point to the origin, and thus system (6.3.4) is equivalent with

$$\dot{\tilde{w}}=-\overline{D}\tilde{w}+\overline{A}\,\tilde{f}(\tilde{w})+\overline{B}\,\tilde{g}(\tilde{w}^\tau), \tag{6.3.15}$$

where $\tilde{w}=w-\hat{w}$, $\tilde{w}^\tau=w^\tau-\hat{w}$, $\tilde{f}(\tilde{w})=\overline{f}(\tilde{w}+\hat{w})-\overline{f}(\hat{w})$, and $\tilde{g}(\tilde{w}^\tau)=\overline{g}(\tilde{w}^\tau+\hat{w})-\overline{g}(\hat{w})$. Construct the following Lyapunov-Krasovskii functional:

$$V(\tilde{w}(t))=\tilde{w}^T(t)P\tilde{w}(t)+\int_{t-\tau}^{t}\tilde{w}(s)^TQ\tilde{w}(s)ds,$$

where $Q \in \mathbb{R}^{8N\times 8N}$, $Q>0$.

The derivative of $V(\tilde{w}(t))$ with respect to t along the trajectories of system (6.3.15) is computed as

$$\begin{aligned}\dot{V}(\tilde{w}) &= \dot{\tilde{w}}^T P\tilde{w} + \tilde{w}^T P\dot{\tilde{w}} + \tilde{w}^T Q\tilde{w} - \tilde{w}^{\tau T} Q\tilde{w}^\tau \\ &= \tilde{w}^T P(-\overline{D}\tilde{w} + \overline{A}\,\tilde{f}(\tilde{w}) + \overline{B}\,\tilde{g}(\tilde{w}^\tau)) \\ &\quad (-\overline{D}\tilde{w} + \overline{A}\,\tilde{f}(\tilde{w}) + \overline{B}\,\tilde{g}(\tilde{w}^\tau))^T P\tilde{w} + \tilde{w}^T Q\tilde{w} - \tilde{w}^{\tau T} Q\tilde{w}^\tau \\ &= \tilde{w}^T(-P\overline{D} - \overline{D}P)\tilde{w} + \tilde{w}^T P\overline{A}\,\tilde{f}(\tilde{w}) + \tilde{f}^T(\tilde{w})\overline{A}^T P\tilde{w} + \tilde{w}^T P\overline{B}\,\tilde{g}(\tilde{w}^\tau) \\ &\quad +\tilde{g}^T(\tilde{w}^\tau)\overline{B}^T P\tilde{w} + \tilde{w}^T Q\tilde{w} - \tilde{\tilde{w}}^{\tau T} Q\tilde{w}^\tau. \end{aligned} \tag{6.3.16}$$

If we multiply relations (6.3.10) and (6.3.11) by $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$, we obtain

$$0 \leq \varepsilon_1(\tilde{w}^T\overline{L_f}^T\overline{L_f}\tilde{w} - \tilde{f}^T(\tilde{w})\tilde{f}(\tilde{w})), \tag{6.3.17}$$

$$0 \leq \varepsilon_2(\tilde{w}^{\tau T}\overline{L_g}^T\overline{L_g}\tilde{w}^\tau - \tilde{g}^T(\tilde{w}^\tau)\tilde{g}(\tilde{w}^\tau)). \tag{6.3.18}$$

Adding inequalities (6.3.17) and (6.3.18) to (6.3.16), gives

$$\dot{V}(\tilde{w}) \leq \xi^T \Omega \xi, \tag{6.3.19}$$

where

$$\xi = \begin{bmatrix}\tilde{w}^T & \tilde{w}^{\tau T} & \tilde{f}^T(\tilde{w}) & \tilde{g}^T(\tilde{w}^\tau)\end{bmatrix}^T,$$

$$\Omega = \begin{bmatrix} -P\overline{D} - \overline{D}P + Q + \varepsilon_1\overline{L_f}^T\overline{L_f} & 0 & P\overline{A} & P\overline{B} \\ * & -Q + \varepsilon_2\overline{L_g}^T\overline{L_g} & 0 & 0 \\ * & * & -\varepsilon_1 I_{8N} & 0 \\ * & * & * & -\varepsilon_2 I_{8N} \end{bmatrix}.$$

Now, we have $\Omega < 0$ if and only if $Q > \varepsilon_2\overline{L_g}^T\overline{L_g}$ and

$$\begin{bmatrix} -P\overline{D} - D\overline{P} + Q + \varepsilon_1\overline{L_f}^T\overline{L_f} & P\overline{A} & P\overline{B} \\ * & -\varepsilon_1 I_{8N} & 0 \\ * & * & -\varepsilon_2 I_{8N} \end{bmatrix} < 0. \tag{6.3.20}$$

Together, the linear matrix inequalities (6.3.20) and $Q > \varepsilon_2\overline{L_g}^T\overline{L_g}$ are equivalent with condition (6.3.5), which means that (6.3.19) becomes $\dot{V}(\tilde{w}) < 0$, from which we can conclude that the equilibrium point of (6.3.4) is globally asymptotically stable, thus ending the proof of the theorem. □

6.3.2 Numerical example

A numerical example is given to demonstrate the effectiveness of our results.

Example 6.1. Consider the following two-neuron octonion-valued recurrent neural network with time delay:

$$\begin{cases} \dot{x}_1(t) = & -d_1x_1(t) + \sum_{j=1}^2 a_{1j}f_j(x_j(t)) + \sum_{j=1}^2 b_{1j}g_j(x_j(t-\tau)) + u_1, \\ \dot{x}_2(t) = & -d_2x_2(t) + \sum_{j=1}^2 a_{2j}f_j(x_j(t)) + \sum_{j=1}^2 b_{1j}g_j(x_j(t-\tau)) + u_2, \end{cases} \tag{6.3.21}$$

where $d_1 = 50$, $d_2 = 40$,

$$\text{vec}(a_{11}) = (1,1,2,2,1,-1,-1,1)^T,\ \text{vec}(a_{12}) = (2,1,1,-2,2,1,-2,2)^T$$

$$\text{vec}(a_{21}) = (2,-2,2,1,2,-2,1,2)^T,\ \text{vec}(a_{22}) = (1,2,2,-2,1,1,2,-2)^T,$$
$$\text{vec}(b_{11}) = (2,1,2,1,-2,2,-1,2)^T,\ \text{vec}(b_{12}) = (-2,2,-2,2,1,2,-2,2)^T,$$
$$\text{vec}(b_{21}) = (1,-2,2,-2,1,2,2,2)^T,\ \text{vec}(b_{22}) = (1,2,2,1,2,-2,-2,1)^T,$$
$$\text{vec}(u_1) = (10,-20,30,-40,50,-70,80,-90)^T,$$
$$\text{vec}(u_2) = (90,-40,10,-60,30,-80,50,-20)^T,$$
$$f_j([x]_p) = \frac{1}{1+e^{-[x]_p}},\ g_j([x]_p) = \frac{1-e^{-[x]_p}}{1+e^{-[x]_p}},\ p \in \{0,1,\ldots,7\},\ j \in \{1,2\},$$

from which we deduce that $l_1^f = l_2^f = \frac{1}{\sqrt{2}}$ and $l_1^g = l_2^g = 2\sqrt{2}$. The time delay is taken to be $\tau = 0.5$.

By solving the LMI condition (6.3.5) in Theorem 6.1, we get that system (6.3.21) has a unique equilibrium point which is globally asymptotically stable for $\varepsilon_1 = 1.3270$, $\varepsilon_2 = 0.7199$.

6.4 Exponential stability for OVNNs with delay

The presentation in this section follows that in the author's paper [158].

6.4.1 Main results

Assuming that system (6.3.4) has a unique equilibrium point $\hat{w}$, this equilibrium point can be shifted to the origin, and so the system (6.3.4) becomes

$$\dot{\tilde{y}} = -\overline{D}\tilde{y} + \overline{A}\,\tilde{f}(\tilde{y}) + \overline{B}\,\tilde{g}(\tilde{y}^\tau), \tag{6.4.1}$$

where $\tilde{y} = w - \hat{w}$, $\tilde{y}^\tau = w^\tau - \hat{w}$, $\tilde{f}(\tilde{y}) = \overline{f}(\tilde{w} + \hat{w}) - \overline{f}(\hat{w})$, and $\tilde{g}(\tilde{y}^\tau) = \overline{g}(\tilde{w}^\tau + \hat{w}) - \overline{g}(\hat{w})$.

Remark 6.2. Systems (6.4.1) and (6.3.1) are equivalent, meaning that any property that holds for system (6.4.1), will also hold for system (6.3.1). For this reason, we will study the global exponential stability of the origin of system (6.4.1).

We give an LMI-based sufficient condition for the global exponential stability of the origin of (6.4.1).

For this, we will also need the following lemma:

Lemma 6.3. *([66]). For any vector function $y : [a,b] \to \mathbb{R}^{8N}$ and positive definite matrix $M \in \mathbb{R}^{8N\times 8N}$, the following linear matrix inequality (LMI) holds:*

$$\left(\int_a^b y(s)ds\right)^T M \left(\int_a^b y(s)ds\right) \leq (b-a)\int_a^b y^T(s)My(s)ds,$$

where the integrals are well defined.

Theorem 6.2. *If Assumption 6.1 holds, then the origin of system* (6.4.1) *is globally exponentially stable if there exist positive definite matrices P, Q_1, Q_2, Q_3, S_1, S_2, S_3, S_4, positive block-diagonal matrices R_1, R_2, R_3, R_4, all from $\mathbb{R}^{8N\times 8N}$, and $\varepsilon > 0$, such that the following linear matrix inequality (LMI) holds*

$$(\Pi)_{9\times 9} < 0, \tag{6.4.2}$$

where $\Pi_{1,1} = 2\varepsilon P - P\overline{D} - \overline{D}P + Q_1 + \tau S_2 + \tau^{-1}e^{-2\varepsilon\tau}S_1 + \tau\overline{D}S_1\overline{D} + \overline{L_f}^T R_1\overline{L_f} + \overline{L_g}^T R_3\overline{L_g}$, $\Pi_{1,3} = P\overline{A} - \tau^{-1}e^{-2\varepsilon\tau}S_1 - \tau\overline{D}S_1\overline{A}$, $\Pi_{1,6} = P\overline{B} - \tau\overline{D}S_1\overline{B}$, $\Pi_{2,2} = -e^{-2\varepsilon\tau}Q_1 + \tau^{-1}e^{-2\varepsilon\tau}S_1 + \overline{L_f}^T R_2\overline{L_f} + \overline{L_g}^T R_4\overline{L_g}$, $\Pi_{3,3} = Q_2 + \tau S_3 - R_1 + \tau\overline{A}^T S_1\overline{A}$, $\Pi_{3,6} = \tau\overline{A}^T S_1\overline{B}$, $\Pi_{4,4} = -e^{-2\varepsilon\tau}Q_2 - R_2$, $\Pi_{5,5} = Q_3 + \tau S_4 - R_3$, $\Pi_{6,6} = -e^{-2\varepsilon\tau}Q_1 - R_4 + \tau\overline{B}^T S_1\overline{B}$, $\Pi_{7,7} = -\tau^{-1}e^{-2\varepsilon\tau}S_2$, $\Pi_{8,8} = -\tau^{-1}e^{-2\varepsilon\tau}S_3$, $\Pi_{9,9} = -\tau^{-1}e^{-2\varepsilon\tau}S_4$.

Proof. We begin by defining the Lyapunov-Krasovskii functional

$$\begin{aligned}
V(\tilde{y}(t)) &= e^{2\varepsilon t}\tilde{y}^T(t)P\tilde{y}(t) \\
&\quad + \int_{t-\tau}^{t} e^{2\varepsilon s}\tilde{y}^T(s)Q_1\tilde{y}(s)ds \\
&\quad + \int_{t-\tau}^{t} e^{2\varepsilon s}\tilde{f}^T(\tilde{y}(s))Q_2\tilde{f}(\tilde{y}(s))ds \\
&\quad + \int_{t-\tau}^{t} e^{2\varepsilon s}\tilde{g}^T(\tilde{y}(s))Q_3 g(\tilde{y}(s))ds \\
&\quad + \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^T(s)S_1\dot{\tilde{y}}(s)dsd\theta \\
&\quad + \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\tilde{y}^T(s)S_2\tilde{y}(s)dsd\theta \\
&\quad + \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\tilde{f}^T(\tilde{y}(s))S_3\tilde{f}(\tilde{y}(s))dsd\theta \\
&\quad + \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\tilde{g}^T(\tilde{y}(s))S_4 g(\tilde{y}(s))dsd\theta.
\end{aligned}$$

The time derivative of V along the trajectories of system (6.4.1) is

$$\begin{aligned}
\dot{V}(\tilde{y}) &= e^{2\varepsilon t}\Big[2\varepsilon\tilde{y}^T P\tilde{y} + \dot{\tilde{y}}^T P\tilde{y} + \tilde{y}^T P\dot{\tilde{y}} + \tilde{y}^T Q_1\tilde{y} - e^{-2\varepsilon\tau}\tilde{y}^{\tau T}Q_1\tilde{y}^\tau \\
&\quad + \tilde{f}^T(\tilde{y})Q_2\tilde{f}(\tilde{y}) - e^{-2\varepsilon\tau}\tilde{f}^T(\tilde{y}^\tau)Q_2\tilde{f}(\tilde{y}^\tau) + \tilde{g}^T(\tilde{y})Q_3\tilde{g}(\tilde{y}) \\
&\quad - e^{-2\varepsilon\tau}\tilde{g}^T(\tilde{y}^\tau)Q_3\tilde{g}(\tilde{y}^\tau) + \tau\dot{\tilde{y}}^T S_1\dot{\tilde{y}} - \int_{t-\tau}^{t} e^{2\varepsilon(s-t)}\dot{\tilde{y}}^T(s)S_1\dot{\tilde{y}}(s)ds \\
&\quad + \tau\tilde{y}^T S_2\tilde{y} - \int_{t-\tau}^{t} e^{2\varepsilon(s-t)}\tilde{y}^T(s)S_2\tilde{y}(s)ds + \tau\tilde{f}^T(\tilde{y})S_3\tilde{f}(\tilde{y}) \\
&\quad - \int_{t-\tau}^{t} e^{2\varepsilon(s-t)}\tilde{f}^T(\tilde{y}(s))S_3\tilde{f}(\tilde{y}(s))ds + \tau\tilde{g}^T(\tilde{y})S_4\tilde{g}(\tilde{y}) \\
&\quad - \int_{t-\tau}^{t} e^{2\varepsilon(s-t)}\tilde{g}^T(\tilde{y}(s))S_4 g(\tilde{y}(s))ds\Big] \\
&\leq e^{2\varepsilon t}\Big[2\varepsilon\tilde{y}^T P\tilde{y} + (-\overline{D}\tilde{y} + \overline{A}\,\tilde{f}(\tilde{y}) + \overline{B}\,\tilde{g}(\tilde{y}^\tau))^T P\tilde{y} + \tilde{y}^T P(-\overline{D}\tilde{y} \\
&\quad + \overline{A}\,\tilde{f}(\tilde{y}) + \overline{B}\,\tilde{g}(\tilde{y}^\tau)) + \tilde{y}^T Q_1\tilde{y} - e^{-2\varepsilon\tau}\tilde{y}^{\tau T}Q_1\tilde{y}^\tau + \tilde{f}^T(\tilde{y})Q_2\tilde{f}(\tilde{y}) \\
&\quad - e^{-2\varepsilon\tau}\tilde{f}^T(\tilde{y}^\tau)Q_2\tilde{f}(\tilde{y}^\tau) + \tilde{g}^T(\tilde{y})Q_3\tilde{g}(\tilde{y}) - e^{-2\varepsilon\tau}\tilde{g}^T(\tilde{y}^\tau)Q_3\tilde{g}(\tilde{y}^\tau) \\
&\quad + \tau\dot{\tilde{y}}^T S_1\dot{\tilde{y}} - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\dot{\tilde{y}}(s)ds\right)^T S_1\left(\int_{t-\tau}^{t}\dot{\tilde{y}}(s)ds\right) \\
&\quad + \tau\tilde{y}^T S_2\tilde{y} - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\tilde{y}(s)ds\right)^T S_2\left(\int_{t-\tau}^{t}\tilde{y}(s)ds\right)
\end{aligned}$$

$$+\tau\tilde{f}^T(\tilde{y})S_3\tilde{f}(\tilde{y}) - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\tilde{f}(\tilde{y}(s))ds\right)^T S_3\left(\int_{t-\tau}^{t}\tilde{f}(\tilde{y}(s))ds\right)$$
$$+\tau\tilde{g}^T(\tilde{y})S_4\tilde{g}(\tilde{y}) - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\tilde{g}(\tilde{y}(s))ds\right)^T S_4\left(\int_{t-\tau}^{t}\tilde{g}(\tilde{y}(s))ds\right)\Bigg], \quad (6.4.3)$$

where the inequality was deduced using Lemma 6.3.

The Lipschitz conditions in Assumption 6.1 are equivalent with

$$||f_j(x) - f_j(x')|| \leq l_j^f||x - x'|| \Leftrightarrow ||\text{vec}(f_j(x)) - \text{vec}(f_j(x'))|| \leq l_j^f||\text{vec}(x) - \text{vec}(x')||,$$

for $j \in \{1, \dots N\}$, and the analogous ones for the functions g_j. Now, from these inequalities we can deduce that there exist positive block-diagonal matrices $R_1 = \text{diag}(r_1^1 I_8, r_2^1 I_8, \dots, r_N^1 I_8)$, $R_2 = \text{diag}(r_1^2 I_8, r_2^2 I_8, \dots, r_N^2 I_8)$, $R_3 = \text{diag}(r_1^3 I_8, r_2^3 I_8, \dots, r_N^3 I_8)$, $R_4 = \text{diag}(r_1^4 I_8, r_2^4 I_8, \dots, r_N^4 I_8)$, such that

$$0 \leq \tilde{y}^T\overline{L_f}^T R_1\overline{L_f}\tilde{y} - \tilde{f}^T(\tilde{y})R_1\tilde{f}(\tilde{y}),\ 0 \leq \tilde{y}^{\tau T}\overline{L_f}^T R_2\overline{L_f}\tilde{y}^\tau - \tilde{f}^T(\tilde{y}^\tau)R_2\tilde{f}(\tilde{y}^\tau), \quad (6.4.4)$$

$$0 \leq \tilde{y}^T\overline{L_g}^T R_3\overline{L_g}\tilde{y} - \tilde{g}^T(\tilde{y})R_3\tilde{g}(\tilde{y}),\ 0 \leq \tilde{y}^{\tau T}\overline{L_g}^T R_4\overline{L_g}\tilde{y}^\tau - \tilde{g}^T(\tilde{y}^\tau)R_4\tilde{g}(\tilde{y}^\tau). \quad (6.4.5)$$

Adding inequalities (6.4.4) and (6.4.5), with inequality (6.4.3), yields

$$\dot{V}(\tilde{y}) \leq e^{2\varepsilon t}\zeta^T\Pi\zeta, \quad (6.4.6)$$

where

$$\zeta = \Big[\ \tilde{y}^T \quad \tilde{y}^{\tau T} \quad \tilde{f}^T(\tilde{y}) \quad \tilde{f}^T(\tilde{y}^\tau) \quad \tilde{g}^T(\tilde{y})\tilde{g}^T(\tilde{y}^\tau)$$
$$\left(\int_{t-\tau}^{t}\tilde{y}(s)ds\right)^T \quad \left(\int_{t-\tau}^{t}\tilde{f}(\tilde{y}(s))ds\right)^T \quad \left(\int_{t-\tau}^{t}\tilde{g}(\tilde{y}(s))ds\right)^T\ \Big]^T,$$

and Π is defined by (6.4.2). Condition (6.4.2) says that $\Pi < 0$, so we can infer from (6.4.6) that $\dot{V}(\tilde{y}) < 0$, which means that $V(\tilde{y}(t))$ is strictly decreasing for $t \geq 0$. From the definition of $V(\tilde{y}(t))$, we can further deduce that

$$e^{2\varepsilon t}\lambda_{\min}(P)||\tilde{y}(t)||^2 \leq e^{2\varepsilon t}\tilde{y}^T(t)P\tilde{y}(t) \leq V(t) \leq V_0,\ \forall t \geq T,\ T \geq 0,$$

where $V_0 = \max_{0\leq t\leq T} V(t)$. Consequently,

$$||\tilde{y}(t)||^2 \leq \frac{V_0}{e^{2\varepsilon t}\lambda_{\min}(P)} \Leftrightarrow ||\tilde{y}(t)|| \leq Me^{-\varepsilon t},\ \forall t \geq 0,$$

for $M = \sqrt{\frac{V_0}{\lambda_{\min}(P)}}$. Thus, we obtained the global exponential stability for the origin of system (6.4.1). □

6.4.2 Numerical example

We now give a numerical example to prove the correctness of the result derived above.

Example 6.2. Consider the following delayed octonion-valued Hopfield neural network with two neurons:

$$\begin{cases}\dot{x}_1(t) = & -d_1x_1(t) + \sum_{j=1}^{2} a_{1j}f_j(x_j(t)) + \sum_{j=1}^{2} b_{1j}g_j(x_j(t-\tau)) + u_1,\\ \dot{x}_2(t) = & -d_2x_2(t) + \sum_{j=1}^{2} a_{2j}f_j(x_j(t)) + \sum_{j=1}^{2} b_{1j}g_j(x_j(t-\tau)) + u_2,\end{cases} \quad (6.4.7)$$

where $d_1 = 50$, $d_2 = 40$, and

$$\text{vec}(a_{11}) = (1,1,2,2,1,-1,-1,1)^T,\ \text{vec}(a_{12}) = (2,1,1,-2,2,1,-2,2)^T,$$
$$\text{vec}(a_{21}) = (2,-2,2,1,2,-2,1,2)^T,\ \text{vec}(a_{22}) = (1,2,2,-2,1,1,2,-2)^T,$$
$$\text{vec}(b_{11}) = (2,1,2,1,-2,2,-1,2)^T,\ \text{vec}(b_{12}) = (-2,2,-2,2,1,2,-2,2)^T,$$
$$\text{vec}(b_{21}) = (1,-2,2,-2,1,2,2,2)^T,\ \text{vec}(b_{22}) = (1,2,2,1,2,-2,-2,1)^T,$$
$$\text{vec}(u_1) = (10,-20,30,-40,50,-70,80,-90)^T,$$
$$\text{vec}(u_2) = (90,-40,10,-60,30,-80,50,-20)^T,$$
$$f_j([x]_p) = \frac{1}{1+e^{-[x]_p}},\ g_j([x]_p) = \frac{1-e^{-[x]_p}}{1+e^{-[x]_p}},\ p \in \{0,1,\ldots,7\},\ j \in \{1,2\}.$$

We have that $l_1^f = l_2^f = \frac{\sqrt{2}}{2}$ and $l_1^g = l_2^g = 2\sqrt{2}$. Also, the constant delay is $\tau = 0.5$.

The global exponential stability of the equilibrium point of system (6.4.7) is obtained by solving the LMI in condition (6.4.2) in Theorem 6.2, to get $\varepsilon = 3$, $R_1 = \text{diag}(3.1392I_8, 3.0388I_8)$, $R_2 = \text{diag}(0.4777I_8, 0.4685I_8)$, $R_3 = \text{diag}(0.2165I_8, 0.1348I_8)$, $R_4 = \text{diag}(0.0017I_8, 0.0016I_8)$.

6.5 Asymptotic stability of delayed OVNNs with leakage delay

The presentation in this section follows that in the author's paper [159].

6.5.1 Main results

We define delayed octonion-valued Hopfield neural networks with leakage delays, for which the states and weights are from $\mathbb{O}$. This type of network is described by the following set of differential equations:

$$\dot{x}_i(t) = -d_i x_i(t-\delta) + \sum_{j=1}^{N} a_{ij} f_j(x_j(t)) + \sum_{j=1}^{N} b_{ij} g_j(x_j(t-\tau)) + u_i, \tag{6.5.1}$$

for $i \in \{1,\ldots,N\}$, where the notations are the same as the ones in Section 6.3, and $\delta \in \mathbb{R}$, $\delta > 0$ is the leakage delay.

Using the same ideas as in Section 6.3, system (6.5.1) can be transformed into the real-valued one:

$$\dot{y}(t) = -\overline{D}y(t-\delta) + \overline{A}\,\overline{f}(y(t)) + \overline{B}\,\overline{g}(y(t-\tau)) + \overline{u}, \tag{6.5.2}$$

where we denoted $y(t) = \left(\text{vec}(x_i(t))^T\right)_{1\le i\le N}^T$, $\overline{D} = \text{diag}\,(d_i I_8)_{1\le i\le N}$, $\overline{A} = (\text{mat}(a_{ij}))_{1\le i,j\le N}$, $\overline{B} = (\text{mat}(b_{ij}))_{1\le i,j\le N}$, $\overline{f}(y(t)) = \left(\text{vec}(f_j(x_j(t)))^T\right)_{1\le j\le N}^T$, $\overline{g}(w(t-\tau)) = \left(\text{vec}(g_j(x_j(t-\tau)))^T\right)_{1\le j\le N}^T$. By shifting the equilibrium point $\hat{y}$ of (6.5.2) to the origin, we obtain

$$\dot{\tilde{y}}(t) = -\overline{D}\tilde{y}(t-\delta) + \overline{A}\,\tilde{f}(\tilde{y}(t)) + \overline{B}\,\tilde{g}(\tilde{y}(t-\tau)), \tag{6.5.3}$$

where $\tilde{y}(t) = y(t) - \hat{y}$, $\tilde{f}(\tilde{y}(t)) = \overline{f}(\tilde{y}(t)+\hat{y}) - \overline{f}(\hat{y})$, $\tilde{g}(\tilde{y}(t)) = \overline{g}(\tilde{y}(t)+\hat{y}) - \overline{g}(\hat{y})$.

Remark 6.3. Systems (6.5.3) and (6.5.1) are equivalent. This means that any property of system (6.5.3), will also be true for system (6.5.1). Thus, the asymptotic stability of the origin of system (6.5.3) will imply the asymptotic stability of the equilibrium point of (6.5.1).

We give an LMI-based sufficient condition for the asymptotic stability of the origin of (6.5.3).

Theorem 6.3. *If Assumption 6.1 holds, then the origin of system* (6.5.3) *is globally asymptotically stable if there exist positive definite matrices* P_1, P_2, P_3, P_4, P_5^1, P_5^2, P_5^3, P_6, *positive definite block-diagonal matrices* R_1, R_2, R_3, R_4, *all from* $\mathbb{R}^{8N\times 8N}$, *such that the following linear matrix inequality (LMI) holds*

$$(\Pi)_{8\times 8} < 0, \tag{6.5.4}$$

where $\Pi_{1,1} = -\overline{D}P_2 - P_2\overline{D} + P_3 + \delta P_4 + P_5^1 - \tau^{-1}P_6 + \overline{L_f}^T R_1\overline{L_f} + \overline{L_g}^T R_3\overline{L_g}$, $\Pi_{1,2} = -P_1\overline{D}$, $\Pi_{1,3} = \tau^{-1}P_6$, $\Pi_{1,4} = P_1\overline{A} + P_2\overline{A}$, $\Pi_{1,7} = P_1\overline{B} + P_2\overline{B}$, $\Pi_{1,8} = \overline{D}P_2\overline{D}$, $\Pi_{2,2} = -P_3 + \tau\overline{D}P_6\overline{D}$, $\Pi_{2,4} = -\tau\overline{D}P_6\overline{A}$, $\Pi_{2,7} = -\tau\overline{D}P_6\overline{B}$, $\Pi_{3,3} = -P_5^1 - \tau^{-1}P_6 + \overline{L_f}^T R_2\overline{L_f} + \overline{L_g}^T R_4\overline{L_g}$, $\Pi_{4,4} = P_5^2 + \tau\overline{A}^T P_6\overline{A} - R_1$, $\Pi_{4,7} = \tau\overline{A}^T P_6\overline{B}$, $\Pi_{4,8} = -\overline{A}^T P_2\overline{D}$, $\Pi_{5,5} = -P_5^2 - R_2$, $\Pi_{6,6} = P_5^3 - R_3$, $\Pi_{7,7} = -P_5^3 + \tau\overline{B}^T P_6\overline{B} - R_4$, $\Pi_{7,8} = -\overline{B}^T P_2\overline{D}$, $\Pi_{8,8} = -\delta^{-1}P_4$.

Proof. We define the following Lyapunov-Krasovskii functional

$$V(t) = V_1(t) + V_2(t) + V_3(t) + V_4(t) + V_5(t) + V_6(t),$$

where

$$V_1(t) = \tilde{y}^T(t)P_1\tilde{y}(t),$$

$$V_2(t) = \left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right)^T P_2\left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right),$$

$$V_3(t) = \int_{t-\delta}^{t}\tilde{y}^T(s)P_3\tilde{y}(s)ds,$$

$$V_4(t) = \int_{-\delta}^{0}\int_{t+\theta}^{t}\tilde{y}^T(s)P_4\tilde{y}(s)dsd\theta,$$

$$V_5(t) = \int_{t-\tau}^{t}\xi^T(s)P_5\xi(s)ds, \quad P_5 = \text{diag}(P_5^1, P_5^2, P_5^3),$$

$$\xi(s) = \left[\tilde{y}^T(s) \quad \tilde{f}^T(\tilde{y}(s)) \quad \tilde{g}^T(\tilde{y}(s))\right]^T,$$

$$V_6(t) = \int_{-\tau}^{0}\int_{t+\theta}^{t}\dot{\tilde{y}}^T(s)P_6\dot{\tilde{y}}(s)dsd\theta.$$

The derivative of $V(t)$ along the trajectories of system (6.5.3) is

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t) + \dot{V}_4(t) + \dot{V}_5(t) + \dot{V}_6(t),$$

where

$$\dot{V}_1(t) = \dot{\tilde{y}}^T(t)P_1\tilde{y}(t) + \tilde{y}^T(t)P_1\dot{\tilde{y}}(t), \tag{6.5.5}$$

$$\begin{aligned}\dot{V}_2(t) &= \left(\dot{\tilde{y}}(t) - \overline{D}\tilde{y}(t) + \overline{D}\tilde{y}(t-\delta)\right)^T P_2\left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right)\\ &\quad + \left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right)^T P_2\left(\dot{\tilde{y}}(t) - \overline{D}\tilde{y}(t) + \overline{D}\tilde{y}(t-\delta)\right),\end{aligned} \tag{6.5.6}$$

$$\dot{V}_3(t) = \tilde{y}^T(t)P_3\tilde{y}(t) - \tilde{y}^T(t-\delta)P_3\tilde{y}(t-\delta), \tag{6.5.7}$$

$$\begin{aligned}\dot{V}_4(t) &= \delta\tilde{y}^T(t)P_4\tilde{y}(t) - \int_{t-\delta}^{t}\tilde{y}^T(s)P_4\tilde{y}(s)ds \\ &\leq \delta\tilde{y}^T(t)P_4\tilde{y}(t) - \delta^{-1}\left(\int_{t-\delta}^{t}\tilde{y}(s)ds\right)^T P_4\left(\int_{t-\delta}^{t}\tilde{y}(s)ds\right),\end{aligned} \tag{6.5.8}$$

$$\dot{V}_5(t) = \xi^T(t)P_5\xi(t) - \xi^T(t-\tau)P_5\xi(t-\tau), \tag{6.5.9}$$

$$\begin{aligned}\dot{V}_6(t) &= \tau\dot{\tilde{y}}^T(t)P_6\dot{\tilde{y}}(t) - \int_{t-\tau}^{t}\dot{\tilde{y}}^T(s)P_6\dot{\tilde{y}}(s)ds \\ &\leq \tau\dot{\tilde{y}}^T(t)P_6\dot{\tilde{y}}(t) - \tau^{-1}\left(\int_{t-\tau}^{t}\dot{\tilde{y}}(s)ds\right)^T P_6\left(\int_{t-\tau}^{t}\dot{\tilde{y}}(s)ds\right),\end{aligned} \tag{6.5.10}$$

where we have used Lemma 6.3 to obtain the inequalities in (6.5.8) and (6.5.10).

From Assumption 6.1 about the Lipschitz condition, we can deduce that

$$||f_j(x) - f_j(x')|| \leq l_j^f||x - x'|| \Leftrightarrow ||\text{vec}(f_j(x)) - \text{vec}(f_j(x'))|| \leq l_j^f||\text{vec}(x) - \text{vec}(x')||,$$

$\forall j \in \{1, \dots N\}$, $\forall x, x' \in \mathbb{O}$, and analogously for the functions g_j. Thus, there exist positive definite block-diagonal matrices $R_1 = \text{diag}\left(r_j^1 I_8\right)_{1\leq j\leq N}$, $R_2 = \text{diag}\left(r_j^2 I_8\right)_{1\leq j\leq N}$, $R_3 = \text{diag}\left(r_j^3 I_8\right)_{1\leq j\leq N}$, $R_4 = \text{diag}\left(r_j^4 I_8\right)_{1\leq j\leq N}$, such that

$$0 \leq \tilde{y}^T(t)\overline{L_f}^T R_1\overline{L_f}\tilde{y}(t) - \tilde{f}^T(\tilde{y}(t))R_1\tilde{f}(\tilde{y}(t)), \tag{6.5.11}$$

$$0 \leq \tilde{y}^T(t-\tau)\overline{L_f}^T R_2\overline{L_f}\tilde{y}(t-\tau) - \tilde{f}^T(\tilde{y}(t-\tau))R_2\tilde{f}(\tilde{y}(t-\tau)), \tag{6.5.12}$$

$$0 \leq \tilde{y}^T(t)\overline{L_g}^T R_3\overline{L_g}\tilde{y}(t) - \tilde{g}^T(\tilde{y}(t))R_3\tilde{g}(\tilde{y}(t)), \tag{6.5.13}$$

$$0 \leq \tilde{y}^T(t-\tau)\overline{L_g}^T R_4\overline{L_g}\tilde{y}(t-\tau) - \tilde{g}^T(\tilde{y}(t-\tau))R_4\tilde{g}(\tilde{y}(t-\tau)). \tag{6.5.14}$$

Now, adding inequalities (6.5.11)–(6.5.14) to (6.5.5)–(6.5.10), we obtain

$$\dot{V}(t) \leq \zeta^T(t)\Pi\zeta(t),$$

where Π is defined by (6.5.4), and

$$\begin{aligned}\zeta(t) = \Big[\ &\tilde{y}^T(t) \quad \tilde{y}^T(t-\delta) \quad \tilde{y}^T(t-\tau) \quad \tilde{f}^T(\tilde{y}(t)) \quad \tilde{f}^T(\tilde{y}(t-\tau)) \quad \tilde{g}^T(\tilde{y}(t)) \quad \tilde{g}^T(\tilde{y}(t-\tau)) \\ &\left(\textstyle\int_{t-\delta}^{t}\tilde{y}(s)ds\right)^T\ \Big]^T.\end{aligned}$$

From (6.5.4) we have that $\Pi < 0$, which implies that $\dot{V}(t) < 0$, meaning that $V(t)$ is strictly decreasing for $t \geq 0$. It can be further deduced from the definition of $V(t)$ that

$$\lambda_{\text{min}}(P_1)||\tilde{y}(t)||^2 \leq \tilde{y}^T(t)P_1\tilde{y}(t) \leq V(t) \leq V_0,\ \forall t \geq T,\ T \geq 0,$$

where $V_0 = \max\limits_{0\leq t\leq T} V(t)$. Thus,

$$||\tilde{y}(t)||^2 \leq \frac{V_0}{\lambda_{\text{min}}(P_1)} \Leftrightarrow ||\tilde{y}(t)|| \leq M,\ \forall t \geq 0,$$

where $M = \sqrt{\frac{V_0}{\lambda_{\text{min}}(P_1)}}$. The above inequality proves the asymptotic stability of the origin of system (6.5.3), completing the proof of the theorem. □

6.5.2 Numerical example

To assess the effectiveness of the main result, we give a numerical example.

Example 6.3. Consider the following two-neuron delayed octonion-valued Hopfield neural network with leakage delay:

$$\begin{cases} \dot{x}_1(t) = & -d_1 x_1(t-\delta) + \sum_{j=1}^{2} a_{1j} f_j(x_j(t)) + \sum_{j=1}^{2} b_{1j} g_j(x_j(t-\tau)) + u_1, \\ \dot{x}_2(t) = & -d_2 x_2(t-\delta) + \sum_{j=1}^{2} a_{2j} f_j(x_j(t)) + \sum_{j=1}^{2} b_{1j} g_j(x_j(t-\tau)) + u_2, \end{cases} \tag{6.5.15}$$

where $d_1 = 30$, $d_2 = 20$, and

$$\text{vec}(a_{11}) = (1,1,2,2,1,-1,-1,1)^T,\ \text{vec}(a_{12}) = (2,1,1,-2,2,1,-2,2)^T,$$

$$\text{vec}(a_{21}) = (2,-2,2,1,2,-2,1,2)^T,\ \text{vec}(a_{22}) = (1,2,2,-2,1,1,2,-2)^T,$$

$$\text{vec}(b_{11}) = (2,1,2,1,-2,2,-1,2)^T,\ \text{vec}(b_{12}) = (-2,2,-2,2,1,2,-2,2)^T,$$

$$\text{vec}(b_{21}) = (1,-2,2,-2,1,2,2,2)^T,\ \text{vec}(b_{22}) = (1,2,2,1,2,-2,-2,1)^T,$$

$$\text{vec}(u_1) = (10,-20,30,-40,50,-70,80,-90)^T,$$

$$\text{vec}(u_2) = (90,-40,10,-60,30,-80,50,-20)^T,$$

$$f_j\left([x]_p\right) = \frac{1}{1+e^{-[x]_p}},\ g_j\left([x]_p\right) = \frac{1-e^{-[x]_p}}{1+e^{-[x]_p}},\ p \in \{0,1,\ldots,7\},\ j \in \{1,2\},$$

from where we can deduce that $l_1^f = l_2^f = \frac{\sqrt{2}}{2}$ and $l_1^g = l_2^g = \sqrt{2}$. The leakage delay is $\delta = 0.02$ and the time delay is $\tau = 0.7$.

Applying Theorem 6.3, if condition (6.5.4) is satisfied, then the equilibrium point of system (6.5.15) is asymptotically stable. The LMI condition can be solved to yield the matrices $R_1 = \text{diag}(0.0107I_8, 0.0080I_8)$, $R_2 = \text{diag}(0.0005I_8, 0.0003I_8)$, $R_3 = \text{diag}(0.0034I_8, 0.0026I_8)$, $R_4 = \text{diag}(0.0023I_8, 0.0019I_8)$.

6.6 Exponential stability of neutral-type OVNNs with time-varying delays

As previously mentioned in Section 6.3, because of the finite switching speed of amplifiers, delays occur in real-life implementations of neural networks, and can cause unstable or oscillatory behavior. For this reason, we consider both bounded leakage delay, and bounded time-varying delays in the OVNN model in this section. On the other hand, distribution propagation delays may appear as a consequence of a distribution of conduction velocities along the pathways of a neural network implementation, which compelled us to also add continuously distributed delays to our model. Taking these facts into account, we develop exponential stability criteria for OVNNs with leakage delay, time-varying delays, and distributed delays in this section.

The presentation in this section follows that in the author's paper [166].

6.6.1 Main results

we will consider the octonion-valued Hopfield neural networks given by the following system of differential equations

$$\begin{aligned}\dot{x}_i(t) &= -d_i x_i(t-\delta) + \sum_{j=1}^{N} a_{ij} f_j(x_j(t)) + \sum_{j=1}^{N} b_{ij} f_j(x_j(t-\tau_{ij}(t))) \\ &\quad + \sum_{j=1}^{N} c_{ij} \int_{t-\eta_{ij}(t)}^{t} f_j(x_j(s))ds + u_i, \quad i = \overline{1,N}, \end{aligned} \tag{6.6.1}$$

where $x(t) = (x_1(t), \ldots, x_N(t))^T \in \mathbb{O}^N$ is the state vector at time t, $D = \text{diag}(d_1, \ldots, d_N) \in \mathbb{R}^{N\times N}$, $d_i > 0$, $\forall i = \overline{1,N}$, is the self-feedback connection weight matrix, $A = (a_{ij})_{1\leq i,j\leq N} \in \mathbb{O}^{N\times N}$ is the connection weight matrix, $B = (b_{ij})_{1\leq i,j\leq N} \in \mathbb{O}^{N\times N}$ is the time-varying delay connection weight matrix, $C = (c_{ij})_{1\leq i,j\leq N} \in \mathbb{O}^{N\times N}$ is the distributed delay connection weight matrix, $f_j : \mathbb{O} \to \mathbb{O}$ are the neuron activation functions, $\forall j = \overline{1,N}$, and $u = (u_1, \ldots, u_N)^T \in \mathbb{O}^N$ is the external input vector. $\delta > 0$ represents the leakage delay, $\tau_{ij} : \mathbb{R} \to \mathbb{R}$ are the time-varying delays, and $\eta_{ij} : \mathbb{R} \to \mathbb{R}$ are the distributed delays, $\forall i,\ j = \overline{1,N}$.

The derivative is defined as the element-wise derivative of $x_i(t)$ with respect to t: $\dot{x}_i(t) = \sum_{p=0}^{7} \frac{d([x_i(t)]_p)}{dt} e_p$, $\forall i = \overline{1,N}$. The multiplication between the weights and the values of the activation functions in the set of differential equations (6.6.1) is the octonion multiplication, defined above. We assume that the activation functions f_j can be written in the form

$$f_j(x) = [f_j']_0(x) + [f_j']_1(x)\jmath + [f_j']_2(x)\ell + [f_j']_3(x)\jmath\ell,\ \forall x \in \mathbb{O},$$

where $[f_j']_0, [f_j']_1, [f_j']_2, [f_j']_3 : \mathbb{O} \to \mathbb{C}$, $\forall j = \overline{1,N}$.

In order to study the dynamic properties of (6.6.1), we need to make the following assumptions:

Assumption 6.2. *The time-varying delays $\tau_{ij} : \mathbb{R} \to \mathbb{R}$ and the distributed delays $\eta_{ij} : \mathbb{R} \to \mathbb{R}$ are continuously differentiable functions and there exist $\tau,\ \eta > 0$ and $\tau' < 1$, such that $\tau_{ij}(t) < \tau$, $\eta_{ij}(t) < \eta$, $\dot{\tau}_{ij}(t) \leq \tau'$, $\forall t > 0$, $\forall i, j = \overline{1,N}$.*

Assumption 6.3. *The octonion-valued activation functions f_j satisfy the following Lipschitz conditions, $\forall x,\ y \in \mathbb{O}$:*

$$||[f_j']_p(x) - [f_j']_p(y)|| \leq l_j^{[f']_p} ||[x']_p - [y']_p||,$$

where $l_j^{[f']_p} > 0$ are the Lipschitz constants, $\forall p \in \{0,1,2,3\}$, $\forall j = \overline{1,N}$.

Assumption 6.4. *The octonion-valued activation functions f_j satisfy the following Lipschitz conditions, $\forall x,\ y \in \mathbb{O}$:*

$$||f_j(x) - f_j(y)|| \leq l_j^f ||x - y||,$$

where $l_j^f > 0$ are the Lipschitz constants, $\forall j = \overline{1,N}$.

Remark 6.4. Assumptions 6.3 and 6.4 are not equivalent, as it can be easily proved that Assumption 6.3 implies Assumption 6.4, but Assumption 6.4 does not imply Assumption 6.3. Both are useful, however, because there are situations in which the Lipschitz conditions for a function cannot be written in the form given in Assumption 6.3, in which case Assumption 6.4 must be used.

Now, the octonion-valued system (6.6.1) can be transformed into a complex-valued one. As such, each equation in (6.6.1) can be split into the following four complex-valued equations:

$$
\begin{aligned}
[\dot{x}'_i(t)]_0 &= -d_i[x'_i(t-\delta)]_0 \\
&\quad + \sum_{j=1}^{N} \big([a'_{ij}]_0[f'_j]_0(x_j(t)) - [a'_{ij}]_1[f'_j]_1(x_j(t)) - [a'_{ij}]_2[f'_j]_2(x_j(t)) - [a'_{ij}]_3[f'_j]_3(x_j(t))\big) \\
&\quad + \sum_{j=1}^{N} \big([b'_{ij}]_0[f'_j]_0(x_j(t-\tau_{ij}(t))) - [b'_{ij}]_1[f'_j]_1(x_j(t-\tau_{ij}(t))) \\
&\quad - [b'_{ij}]_2[f'_j]_2(x_j(t-\tau_{ij}(t))) - [b'_{ij}]_3[f'_j]_3(x_j(t-\tau_{ij}(t)))\big) \\
&\quad + \sum_{j=1}^{N} \left([c'_{ij}]_0 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_0(x_j(s))ds - [c'_{ij}]_1 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_1(x_j(s))ds \right. \\
&\quad \left. - [c'_{ij}]_2 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_2(x_j(s))ds - [c'_{ij}]_3 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_3(x_j(s))ds \right) \\
&\quad + [u'_i]_0, \quad i = \overline{1,N}, \\
[\dot{x}'_i(t)]_1 &= -d_i[x'_i(t-\delta)]_1 \\
&\quad + \sum_{j=1}^{N} \big([a'_{ij}]_0[f'_j]_1(x_j(t)) + [a'_{ij}]_1[f'_j]_0(x_j(t)) + [a'_{ij}]_2[f'_j]_3(x_j(t)) - [a'_{ij}]_3[f'_j]_2(x_j(t))\big) \\
&\quad + \sum_{j=1}^{N} \big([b'_{ij}]_0[f'_j]_1(x_j(t-\tau_{ij}(t))) + [b'_{ij}]_1[f'_j]_0(x_j(t-\tau_{ij}(t))) \\
&\quad + [b'_{ij}]_2[f'_j]_3(x_j(t-\tau_{ij}(t))) - [b'_{ij}]_3[f'_j]_2(x_j(t-\tau_{ij}(t)))\big) \\
&\quad + \sum_{j=1}^{N} \left([c'_{ij}]_0 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_1(x_j(s))ds + [c'_{ij}]_1 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_0(x_j(s))ds \right. \\
&\quad \left. + [c'_{ij}]_2 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_3(x_j(s))ds - [c'_{ij}]_3 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_2(x_j(s))ds \right) \\
&\quad + [u'_i]_1, \quad i = \overline{1,N}, \\
[\dot{x}'_i(t)]_2 &= -d_i[x'_i(t-\delta)]_2 \\
&\quad + \sum_{j=1}^{N} \big([a'_{ij}]_0[f'_j]_2(x_j(t)) - [a'_{ij}]_1[f'_j]_3(x_j(t)) + [a'_{ij}]_2[f'_j]_0(x_j(t)) + [a'_{ij}]_3[f'_j]_1(x_j(t))\big) \\
&\quad + \sum_{j=1}^{N} \big([b'_{ij}]_0[f'_j]_2(x_j(t-\tau_{ij}(t))) - [b'_{ij}]_1[f'_j]_3(x_j(t-\tau_{ij}(t))) \\
&\quad + [b'_{ij}]_2[f'_j]_0(x_j(t-\tau_{ij}(t))) + [b'_{ij}]_3[f'_j]_1(x_j(t-\tau_{ij}(t)))\big) \\
&\quad + \sum_{j=1}^{N} \left([c'_{ij}]_0 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_2(x_j(s))ds - [c'_{ij}]_1 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_3(x_j(s))ds \right. \\
&\quad \left. + [c'_{ij}]_2 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_0(x_j(s))ds + [c'_{ij}]_3 \int_{t-\eta_{ij}(t)}^{t} [f'_j]_1(x_j(s))ds \right)
\end{aligned}
$$

$$
\begin{aligned}
&+[u_i']_2, \quad i=\overline{1,N},\\
[\dot{x}_i'(t)]_3 &= -d_i[x_i'(t-\delta)]_3\\
&+\sum_{j=1}^{N}\left([a_{ij}']_0[f_j']_3(x_j(t))+[a_{ij}']_1[f_j']_2(x_j(t))-[a_{ij}']_2[f_j']_1(x_j(t))+[a_{ij}']_3[f_j']_0(x_j(t))\right)\\
&+\sum_{j=1}^{N}\left([b_{ij}']_0[f_j']_3(x_j(t-\tau_{ij}(t)))+[b_{ij}']_1[f_j']_2(x_j(t-\tau_{ij}(t)))\right.\\
&\left.-[b_{ij}']_2[f_j']_1(x_j(t-\tau_{ij}(t)))+[b_{ij}']_3[f_j']_0(x_j(t-\tau_{ij}(t)))\right)\\
&+\sum_{j=1}^{N}\left([c_{ij}']_0\int_{t-\eta_{ij}(t)}^{t}[f_j']_3(x_j(s))ds+[c_{ij}']_1\int_{t-\eta_{ij}(t)}^{t}[f_j']_2(x_j(s))ds\right.\\
&\left.-[c_{ij}']_2\int_{t-\eta_{ij}(t)}^{t}[f_j']_1(x_j(s))ds+[c_{ij}']_3\int_{t-\eta_{ij}(t)}^{t}[f_j']_0(x_j(s))ds\right)\\
&+[u_i']_3, \quad i=\overline{1,N}.
\end{aligned}
\tag{6.6.2}
$$

By shifting the equilibrium point $\hat{x}$ of (6.6.2) to the origin, we have that

$$
\begin{aligned}
[\dot{\tilde{x}}'(t)]_0 &= -D[\tilde{x}'(t-\delta)]_0\\
&+\Big([A']_0[\tilde{f}']_0(\tilde{x}(t))-[A']_1[\tilde{f}']_1(\tilde{x}(t))-[A']_2[\tilde{f}']_2(\tilde{x}(t))-[A']_3[\tilde{f}']_3(\tilde{x}(t))\Big)\\
&+\Big([B']_0[\tilde{f}']_0(\tilde{x}(t-\tau(t)))-[B']_1[\tilde{f}']_1(\tilde{x}(t-\tau(t)))\\
&-[B']_2[\tilde{f}']_2(\tilde{x}(t-\tau(t)))-[B']_3[\tilde{f}']_3(\tilde{x}(t-\tau(t)))\Big)\\
&+\left([C']_0\int_{t-\eta(t)}^{t}[\tilde{f}']_0(\tilde{x}(s))ds-[C']_1\int_{t-\eta(t)}^{t}[\tilde{f}']_1(\tilde{x}(s))ds\right.\\
&\left.-[C']_2\int_{t-\eta(t)}^{t}[\tilde{f}']_2(\tilde{x}(s))ds-[C']_3\int_{t-\eta(t)}^{t}[\tilde{f}']_3(\tilde{x}(s))ds\right),\\
[\dot{\tilde{x}}'(t)]_1 &= -D[\tilde{x}'(t-\delta)]_1\\
&+\Big([A']_0[\tilde{f}']_1(\tilde{x}(t))+[A']_1[\tilde{f}']_0(\tilde{x}(t))+[A']_2[\tilde{f}']_3(\tilde{x}(t))-[A']_3[\tilde{f}']_2(\tilde{x}(t))\Big)\\
&+\Big([B']_0[\tilde{f}']_1(\tilde{x}(t-\tau(t)))+[B']_1[\tilde{f}']_0(\tilde{x}(t-\tau(t)))\\
&+[B']_2[\tilde{f}']_3(\tilde{x}(t-\tau(t)))-[B']_3[\tilde{f}']_2(\tilde{x}(t-\tau(t)))\Big)\\
&+\left([C']_0\int_{t-\eta(t)}^{t}[\tilde{f}']_1(\tilde{x}(s))ds+[C']_1\int_{t-\eta(t)}^{t}[\tilde{f}']_0(\tilde{x}(s))ds\right.\\
&\left.+[C']_2\int_{t-\eta(t)}^{t}[\tilde{f}']_3(\tilde{x}(s))ds-[C']_3\int_{t-\eta(t)}^{t}[\tilde{f}']_2(\tilde{x}(s))ds\right),\\
[\dot{\tilde{x}}'(t)]_2 &= -D[\tilde{x}'(t-\delta)]_2\\
&+\Big([A']_0[\tilde{f}']_2(\tilde{x}(t))-[A']_1[\tilde{f}']_3(\tilde{x}(t))+[A']_2[\tilde{f}']_0(\tilde{x}(t))+[A']_3[\tilde{f}']_1(\tilde{x}(t))\Big)\\
&+\Big([B']_0[\tilde{f}']_2(\tilde{x}(t-\tau(t)))-[B']_1[\tilde{f}']_3(\tilde{x}(t-\tau(t)))
\end{aligned}
$$

$$
\begin{aligned}
&+[B']_2[\tilde{f}']_0(\tilde{x}(t-\tau(t))) + [B']_3[\tilde{f}']_1(\tilde{x}(t-\tau(t)))\Big) \\
&+\Big([C']_0\int_{t-\eta(t)}^{t}[\tilde{f}']_2(\tilde{x}(s))ds - [C']_1\int_{t-\eta(t)}^{t}[\tilde{f}']_3(\tilde{x}(s))ds \\
&+[C']_2\int_{t-\eta(t)}^{t}[\tilde{f}']_0(\tilde{x}(s))ds + [C']_3\int_{t-\eta(t)}^{t}[\tilde{f}']_1(\tilde{x}(s))ds\Big),
\end{aligned}
$$

$$
\begin{aligned}
[\dot{\tilde{x}}'(t)]_3 \;=\;& -D[\tilde{x}'(t-\delta)]_3 \\
&+\Big([A']_0[\tilde{f}']_3(\tilde{x}(t)) + [A']_1[\tilde{f}']_2(\tilde{x}(t)) - [A']_2[\tilde{f}']_1(\tilde{x}(t)) + [A']_3[\tilde{f}']_0(\tilde{x}(t))\Big) \\
&+\Big([B']_0[\tilde{f}']_3(\tilde{x}(t-\tau(t))) + [B']_1[\tilde{f}']_2(\tilde{x}(t-\tau(t))) \\
&-[B']_2[\tilde{f}']_1(\tilde{x}(t-\tau(t))) + [B']_3[\tilde{f}']_0(\tilde{x}(t-\tau(t)))\Big) \\
&+\Big([C']_0\int_{t-\eta(t)}^{t}[\tilde{f}']_3(\tilde{x}(s))ds + [C']_1\int_{t-\eta(t)}^{t}[\tilde{f}']_2(\tilde{x}(s))ds \\
&-[C']_2\int_{t-\eta(t)}^{t}[\tilde{f}']_1(\tilde{x}(s))ds + [C']_3\int_{t-\eta(t)}^{t}[\tilde{f}']_0(\tilde{x}(s))ds\Big), \qquad (6.6.3)
\end{aligned}
$$

where $[\tilde{x}'(t)]_p = [x'(t)]_p - [\hat{x}']_p$, $[\tilde{f}']_p(\tilde{x}(t)) = [f']_p(x(t)) - [f']_p(\hat{x})$, $\forall p \in \{0,1,2,3\}$.

By denoting

$$
\tilde{y}(t) = \left([\tilde{x}'(t)]_0^H, [\tilde{x}'(t)]_1^H, [\tilde{x}'(t)]_2^H, [\tilde{x}'(t)]_3^H\right)^H \in \mathbb{C}^{4N},
$$

$$
\overline{D} = \begin{bmatrix} D & 0 & 0 & 0 \\ 0 & D & 0 & 0 \\ 0 & 0 & D & 0 \\ 0 & 0 & 0 & D \end{bmatrix} \in \mathbb{R}^{4N\times 4N},
$$

$$
\overline{A} = \begin{bmatrix} [A']_0 & -[A']_1 & -[A']_2 & -[A']_3 \\ [A']_1 & [A']_0 & -[A']_3 & [A']_2 \\ [A']_2 & [A']_3 & [A']_0 & -[A']_1 \\ [A']_3 & -[A']_2 & [A']_1 & [A']_0 \end{bmatrix} \in \mathbb{C}^{4N\times 4N},
$$

$$
\overline{B} = \begin{bmatrix} [B']_0 & -[B']_1 & -[B']_2 & -[B']_3 \\ [B']_1 & [B']_0 & -[B']_3 & [B']_2 \\ [B']_2 & [B']_3 & [B']_0 & -[B']_1 \\ [B']_3 & -[B']_2 & [B']_1 & [B']_0 \end{bmatrix} \in \mathbb{C}^{4N\times 4N},
$$

$$
\overline{C} = \begin{bmatrix} [C']_0 & -[C']_1 & -[C']_2 & -[C']_3 \\ [C']_1 & [C']_0 & -[C']_3 & [C']_2 \\ [C']_2 & [C']_3 & [C']_0 & -[C']_1 \\ [C']_3 & -[C']_2 & [C']_1 & [C']_0 \end{bmatrix} \in \mathbb{C}^{4N\times 4N},
$$

$$
\tilde{f}(\tilde{y}(t)) = \left([\tilde{f}']_0^H(x(t)), [\tilde{f}']_1^H(x(t)), [\tilde{f}']_2^H(x(t)), [\tilde{f}']_3^H(x(t))\right)^H \in \mathbb{C}^{4N},
$$

system (6.6.3) becomes:

$$
\dot{\tilde{y}}(t) = -\overline{D}\tilde{y}(t-\delta) + \overline{A}\tilde{f}(\tilde{y}(t)) + \overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds. \qquad (6.6.4)
$$

Remark 6.5. Systems (6.6.3), (6.6.4), and (6.6.1) are equivalent. This means that any property of systems (6.6.3), (6.6.4) will also be true for system (6.6.1). Thus, the global exponential stability of the origin of systems (6.6.3), (6.6.4) will imply the global exponential stability of the equilibrium point of (6.6.1).

We will also need the following lemmas:

Lemma 6.4. *([36]) For any vector function $x : [a, b] \to \mathbb{C}^{4N}$ and any positive definite matrix $M \in \mathbb{C}^{4N\times 4N}$, the following linear matrix inequality (LMI) holds:*

$$\left(\int_a^b x(s)ds\right)^H M \left(\int_a^b x(s)ds\right) \leq (b-a)\int_a^b x^H(s)Mx(s)ds,$$

where the integrals are well defined.

Lemma 6.5. *([218]) For any vector function $x : [a, b] \to \mathbb{C}^{4N}$ and any positive definite matrix $M \in \mathbb{C}^{4N\times 4N}$, the following linear matrix inequality (LMI) holds:*

$$\left(\int_a^b\int_\theta^b x(s)dsd\theta\right)^H M \left(\int_a^b\int_\theta^b x(s)dsd\theta\right) \leq \frac{(b-a)^2}{2}\int_a^b\int_\theta^b x^H(s)Mx(s)dsd\theta,$$

where the integrals are well defined.

We will first give an LMI-based sufficient condition for the exponential stability of the origin of (6.6.3).

Theorem 6.4. *If Assumptions 6.2 and 6.3 hold, then the origin of system* (6.6.3) *is globally exponentially stable if there exist positive definite matrices P_p, Q_p, R_p, S_p, T_p^1, T_p^2, U_p, W_p, $Z_p \in \mathbb{C}^{N\times N}$, any matrices N_p^1, N_p^2, N_p^3, N_p^4, $N_p^5 \in \mathbb{C}^{N\times N}$, and positive definite diagonal matrices G_p^1, $G_p^2 \in \mathbb{R}^{N\times N}$, such that the following linear matrix inequalities (LMIs) hold*

$$(\Pi_p)_{19\times 19} < 0, \tag{6.6.5}$$

$\forall p \in \{0, 1, 2, 3\}$, *where* $(\Pi_p)_{1,2} = Q_p$, $(\Pi_p)_{1,3} = -P_pD + Q_pD$, $(\Pi_p)_{1,4} = e^{-2\varepsilon\tau}U_p$, $(\Pi_p)_{1,6:9} = P_p\overline{A}_p$, $(\Pi_p)_{1,10:13} = P_p\overline{B}_p$, $(\Pi_p)_{1,14} = -2\varepsilon Q_pD + DQ_pD$, $(\Pi_p)_{1,15} = 4\tau e^{-2\varepsilon\tau}Z_p$, $(\Pi_p)_{1,16:19} = P_p\overline{C}_p$, $(\Pi_p)_{2,2} = \tau^2U_p + \tau^4Z_p - N_p^1 - (N_p^1)^H$, $(\Pi_p)_{2,3} = -N_p^1D - (N_p^2)^H$, $(\Pi_p)_{2,6:9} = N_p^1\overline{A}_p + (N_p^3)^H\overline{E}_p$, $(\Pi_p)_{2,10:13} = N_p^1\overline{B}_p + (N_p^4)^H\overline{E}_p$, $(\Pi_p)_{2,14} = -Q_pD$, $(\Pi_p)_{2,16:19} = N_p^1\overline{C}_p + (N_p^5)^H\overline{E}_p$, $(\Pi_p)_{3,3} = -e^{-2\varepsilon\delta}R_p - N_p^2D - D(N_p^2)^H$, $(\Pi_p)_{3,6:9} = N_p^2\overline{A}_p + D(N_p^3)^H\overline{E}_p$, $(\Pi_p)_{3,10:13} = N_p^2\overline{B}_p + D(N_p^4)^H\overline{E}_p$, $(\Pi_p)_{3,14} = -DQ_pD$, $(\Pi_p)_{3,16:19} = N_p^2\overline{C}_p + D(N_p^5)^H\overline{E}_p$, $(\Pi_p)_{4,4} = -e^{-2\varepsilon\tau}U_p$, $(\Pi_p)_{5,5} = -(1-\tau')e^{-2\varepsilon\tau}T_p^1 + \overline{L_{[\tilde{f}']_p}}^H G_p^2\overline{L_{[\tilde{f}']_p}}$, $(\Pi_p)_{6:9,6:9} = \overline{E}_p^HT_p^2\overline{E}_p + \eta^2\overline{E}_p^HW_p\overline{E}_p - \overline{E}_p^HG_p^1\overline{E}_p - \overline{E}_p^HN_p^3\left(\overline{A}_p - \overline{A}_p\overline{E}_p^H\overline{E}_p\right) - \left(\overline{A}_p - \overline{A}_p\overline{E}_p^H\overline{E}_p\right)^H(N_p^3)^H\overline{E}_p - \frac{1}{4}\sum_{q=0}^3\overline{E}_q^H\left(N_q^3\overline{A}_q\overline{E}_q^H + \overline{E}_q\overline{A}_q^H(N_q^3)^H\right)\overline{E}_q$, $(\Pi_p)_{6:9,10:13} = -\overline{E}_p^HN_p^3\overline{B}_p - \overline{A}_p^H(N_p^4)^H\overline{E}_p$, $(\Pi_p)_{6:9,16:19} = -\overline{E}_p^HN_p^3\overline{C}_p - \overline{A}^H(N^5)^H\overline{E}_p$, $(\Pi_p)_{10:13,10:13} = -(1-\tau')e^{-2\varepsilon\tau}\overline{E}_p^HT_p^2\overline{E}_p - \overline{E}_p^HG_p^2\overline{E}_p - \overline{E}_p^HN_p^4\left(\overline{B}_p - \overline{B}_p\overline{E}_p^H\overline{E}_p\right) - \left(\overline{B}_p - \overline{B}_p\overline{E}_p^H\overline{E}_p\right)^H(N_p^4)^H\overline{E}_p - \frac{1}{4}\sum_{q=0}^3\overline{E}_q^H\left(N_q^4\overline{B}_q\overline{E}_q^H + \overline{E}_q\overline{B}_q^H(N_q^4)^H\right)\overline{E}_q$, $(\Pi_p)_{10:13,16:19} = -\overline{E}_p^HN_p^4\overline{C}_p - \overline{B}_p^H(N_p^5)^H\overline{E}_p$, $(\Pi_p)_{14,14} = 2\varepsilon DQ_pD - e^{-2\varepsilon\delta}S_p$, $(\Pi_p)_{15,15} = -4e^{-2\varepsilon\tau}Z_p$, $(\Pi_p)_{16:19,16:19} = -e^{-2\varepsilon\eta}\overline{E}_p^HW_p\overline{E}_p - \overline{E}_p^HN_p^5\left(\overline{C}_p - \overline{C}_p\overline{E}_p^H\overline{E}_p\right) - \left(\overline{C}_p - \overline{C}_p\overline{E}_p^H\overline{E}_p\right)^H(N_p^5)^H\overline{E}_p - \frac{1}{4}\sum_{q=0}^3\overline{E}_q^H\left(N_q^5\overline{C}_q\overline{E}_q^H + \overline{E}_q\overline{C}_q^H(N_q^5)^H\right)\overline{E}_q$.

We denoted by $\overline{A}_p$ the $(p+1)$-th row of $\overline{A}$ (and analogously for $\overline{B}_p$ and $\overline{C}_p$) and by $\overline{E}_p$ the $(p+1)$-th row of

$$\overline{E} = \begin{bmatrix} I_N & 0 & 0 & 0 \\ 0 & I_N & 0 & 0 \\ 0 & 0 & I_N & 0 \\ 0 & 0 & 0 & I_N \end{bmatrix} \in \mathbb{R}^{4N\times 4N},$$

$\forall p \in \{0,1,2,3\}$.

Proof. Consider the Lyapunov–Krasovskii functional

$$V(t) = V_1(t) + V_2(t) + V_3(t) + V_4(t) + V_5(t) + V_6(t) + V_7(t) + V_8(t),$$

$$V_1(t) = \sum_{p=0}^{3} e^{2\varepsilon t}[\tilde{x}'(t)]_p^H P_p[\tilde{x}'(t)]_p,$$

$$V_2(t) = \sum_{p=0}^{3} e^{2\varepsilon t}\left([\tilde{x}'(t)]_p - D\int_{t-\delta}^{t}[\tilde{x}'(s)]_p ds\right)^H Q_p\left([\tilde{x}'(t)]_p - D\int_{t-\delta}^{t}[\tilde{x}'(s)]_p ds\right),$$

$$V_3(t) = \sum_{p=0}^{3}\int_{t-\delta}^{t} e^{2\varepsilon s}[\tilde{x}'(s)]_p^H R_p[\tilde{x}'(s)]_p ds,$$

$$V_4(t) = \delta\sum_{p=0}^{3}\int_{-\delta}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}[\tilde{x}'(s)]_p^H S_p[\tilde{x}'(s)]_p ds d\theta,$$

$$V_5(t) = \sum_{p=0}^{3}\int_{t-\tau(t)}^{t} e^{2\varepsilon s}\xi_p^H(s)T_p\xi_p(s)ds, \quad T_p = \text{diag}(T_p^1, T_p^2),$$

$$\xi_p(s) = \left[[\tilde{x}'(s)]_p^H \quad [\tilde{f}']_p^H(\tilde{x}(s))\right]^H,$$

$$V_6(t) = \tau\sum_{p=0}^{3}\int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H U_p[\dot{\tilde{x}}'(s)]_p ds d\theta,$$

$$V_7(t) = \eta\sum_{p=0}^{3}\int_{-\eta}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}[\tilde{f}']_p^H(\tilde{x}(s))W_p[\tilde{f}']_p(\tilde{x}(s))ds d\theta,$$

$$V_8(t) = 2\tau^2\sum_{p=0}^{3}\int_{-\tau}^{0}\int_{\lambda}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H Z_p[\dot{\tilde{x}}'(s)]_p ds d\theta d\lambda.$$

The derivative of V along the trajectories of system (6.6.3) is

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t) + \dot{V}_4(t) + \dot{V}_5(t) + \dot{V}_6(t) + \dot{V}_7(t) + \dot{V}_8(t),$$

$$\begin{aligned}
\dot{V}_1(t) &= \sum_{p=0}^{3} e^{2\varepsilon t}\left(2\varepsilon[\tilde{x}'(t)]_p^H P_p[\tilde{x}'(t)]_p + [\dot{\tilde{x}}'(t)]_p^H P_p[\tilde{x}'(t)]_p + [\tilde{x}'(t)]_p^H P_p[\dot{\tilde{x}}'(t)]_p\right) \\
&= \sum_{p=0}^{3} e^{2\varepsilon t}\Big(2\varepsilon[\tilde{x}'(t)]_p^H P_p[\tilde{x}'(t)]_p + \Big(-D[\tilde{x}'(t-\delta)]_p + \overline{A}_p\tilde{f}(\tilde{y}(t)) + \overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad + \overline{C}_p\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\Big)^H P_p[\tilde{x}'(t)]_p + [\tilde{x}'(t)]_p^H P_p\Big(-D[\tilde{x}'(t-\delta)]_p + \overline{A}_p\tilde{f}(\tilde{y}(t)) \\
&\quad + \overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}_p\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\Big)\Big),
\end{aligned} \tag{6.6.6}$$

$$\begin{aligned}\dot{V}_2(t) &= \sum_{p=0}^{3} e^{2\varepsilon t}\left(2\varepsilon\left([\tilde{x}'(t)]_p - D\int_{t-\delta}^{t}[\tilde{x}'(s)]_p ds\right)^H Q_p\left([\tilde{x}'(t)]_p - D\int_{t-\delta}^{t}[\tilde{x}'(s)]_p ds\right)\right.\\ &\quad + \left([\dot{\tilde{x}}'(t)]_p - D[\tilde{x}'(t)]_p + D[\tilde{x}'(t-\delta)]_p\right)^H Q_p\left([\tilde{x}'(t)]_p - D\int_{t-\delta}^{t}[\tilde{x}'(s)]_p ds\right)\\ &\quad \left. + \left([\tilde{x}'(t)]_p - D\int_{t-\delta}^{t}[\tilde{x}'(s)]_p ds\right)^H Q_p\left([\dot{\tilde{x}}'(t)]_p - D[\tilde{x}'(t)]_p + D[\tilde{x}'(t-\delta)]_p\right)\right),\end{aligned} \tag{6.6.7}$$

$$\dot{V}_3(t) = \sum_{p=0}^{3} e^{2\varepsilon t}\left([\tilde{x}'(t)]_p^H R_p[\tilde{x}'(t)]_p - e^{-2\varepsilon\delta}[\tilde{x}'(t-\delta)]_p^H R_p[\tilde{x}'(t-\delta)]_p\right), \tag{6.6.8}$$

$$\begin{aligned}\dot{V}_4(t) &= \sum_{p=0}^{3}\left(\delta^2 e^{2\varepsilon t}[\tilde{x}'(t)]_p^H S_p[\tilde{x}'(t)]_p - \delta\int_{t-\delta}^{t} e^{2\varepsilon s}[\tilde{x}'(s)]_p^H S_p[\tilde{x}'(s)]_p ds\right)\\ &\leq \sum_{p=0}^{3} e^{2\varepsilon t}\left(\delta^2[\tilde{x}'(t)]_p^H S_p[\tilde{x}'(t)]_p - e^{-2\varepsilon\delta}\left(\int_{t-\delta}^{t}[\tilde{x}'(s)]_p ds\right)^H S_p\left(\int_{t-\delta}^{t}[\tilde{x}'(s)]_p ds\right)\right),\end{aligned} \tag{6.6.9}$$

$$\begin{aligned}\dot{V}_5(t) &= \sum_{p=0}^{3} e^{2\varepsilon t}\left(\xi_p^H(t)T_p\xi_p(t) - (1-\dot{\tau}(t))e^{-2\varepsilon\tau(t)}\xi_p^H(t-\tau(t))T_p\xi_p(t-\tau(t))\right)\\ &\leq \sum_{p=0}^{3} e^{2\varepsilon t}\left(\xi_p^H(t)T_p\xi_p(t) - (1-\tau')e^{-2\varepsilon\tau}\xi_p^H(t-\tau(t))T_p\xi_p(t-\tau(t))\right),\end{aligned} \tag{6.6.10}$$

$$\begin{aligned}\dot{V}_6(t) &= \sum_{p=0}^{3}\left(\tau^2 e^{2\varepsilon t}[\dot{\tilde{x}}'(t)]_p^H U_p[\dot{\tilde{x}}'(t)]_p - \tau\int_{t-\tau}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H U_p[\dot{\tilde{x}}'(s)]_p ds\right)\\ &\leq e^{2\varepsilon t}\left(\tau^2[\dot{\tilde{x}}'(t)]_p^H U_p[\dot{\tilde{x}}'(t)]_p - e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}[\dot{\tilde{x}}'(s)]_p ds\right)^H U_p\left(\int_{t-\tau}^{t}[\dot{\tilde{x}}'(s)]_p ds\right)\right)\\ &= e^{2\varepsilon t}\left(\tau^2[\dot{\tilde{x}}'(t)]_p^H U_p[\dot{\tilde{x}}'(t)]_p\right.\\ &\quad \left. - e^{-2\varepsilon\tau}\left([\tilde{x}'(t)]_p - [\tilde{x}'(t-\tau)]_p\right)^H U_p\left([\tilde{x}'(t)]_p - [\tilde{x}'(t-\tau)]_p\right)\right),\end{aligned} \tag{6.6.11}$$

$$\begin{aligned}\dot{V}_7(t) &= \sum_{p=0}^{3}\left(\eta^2 e^{2\varepsilon t}[\tilde{f}']_p^H(\tilde{x}(t))W_p[\tilde{f}']_p(\tilde{x}(t)) - \eta\int_{t-\eta}^{t} e^{2\varepsilon s}[\tilde{f}']_p^H(\tilde{x}(s))W_p[\tilde{f}']_p(\tilde{x}(s))ds\right)\\ &\leq \sum_{p=0}^{3}\left(\eta^2 e^{2\varepsilon t}[\tilde{f}']_p^H(\tilde{x}(t))W_p[\tilde{f}']_p(\tilde{x}(t)) - \eta(t)\int_{t-\eta(t)}^{t} e^{2\varepsilon s}[\tilde{f}']_p^H(\tilde{x}(s))W_p[\tilde{f}']_p(\tilde{x}(s))ds\right)\\ &\leq \sum_{p=0}^{3} e^{2\varepsilon t}\left(\eta^2[\tilde{f}']_p^H(\tilde{x}(t))W_p[\tilde{f}']_p(\tilde{x}(t))\right.\\ &\quad \left. - e^{-2\varepsilon\eta(t)}\left(\int_{t-\eta(t)}^{t}[\tilde{f}']_p(\tilde{x}(s))ds\right)^H W_p\left(\int_{t-\eta(t)}^{t}[\tilde{f}']_p(\tilde{x}(s))ds\right)\right)\end{aligned}$$

$$\leq \sum_{p=0}^{3} e^{2\varepsilon t}\left(\eta^2[\tilde{f}']_p^H(\tilde{x}(t))W_p[\tilde{f}']_p(\tilde{x}(t))\right.$$
$$\left.-e^{-2\varepsilon\eta}\left(\int_{t-\eta(t)}^{t}[\tilde{f}']_p(\tilde{x}(s))ds\right)^H W_p\left(\int_{t-\eta(t)}^{t}[\tilde{f}']_p(\tilde{x}(s))ds\right)\right), \tag{6.6.12}$$

$$\begin{aligned}
\dot{V}_8(t) &= \sum_{p=0}^{3}\left(\tau^4 e^{2\varepsilon t}[\dot{\tilde{x}}'(t)]_p^H Z_p[\dot{\tilde{x}}'(t)]_p - 2\tau^2\int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H Z_p[\dot{\tilde{x}}'(s)]_p ds d\theta\right)\\
&\leq \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^4[\dot{\tilde{x}}'(t)]_p^H Z_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad\left.-4e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\int_{t+\theta}^{t}[\dot{\tilde{x}}'(s)]_p ds d\theta\right)^H Z_p\left(\int_{-\tau}^{0}\int_{t+\theta}^{t}[\dot{\tilde{x}}'(s)]_p ds d\theta\right)\right)\\
&= \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^4[\dot{\tilde{x}}'(t)]_p^H Z_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad\left.-4e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\left([\tilde{x}'(t)]_p-[\tilde{x}'(t+\theta)]_p\right)d\theta\right)^H Z_p\left(\int_{-\tau}^{0}\left([\tilde{x}'(t)]_p-[\tilde{x}'(t+\theta)]_p\right)d\theta\right)\right)\\
&= \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^4[\dot{\tilde{x}}'(t)]_p^H Z_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad\left.-4e^{-2\varepsilon\tau}\left(\tau[\tilde{x}'(t)]_p-\int_{t-\tau}^{t}[\tilde{x}'(s)]_p ds\right)^H Z_p\left(\tau[\tilde{x}'(t)]_p-\int_{t-\tau}^{t}[\tilde{x}'(s)]_p ds\right)\right),
\end{aligned} \tag{6.6.13}$$

where we have used Assumption 6.2 to obtain the inequalities in (6.6.10) and (6.6.12), Lemma 6.4 to obtain the inequalities in (6.6.9), (6.6.11), and (6.6.12), and Lemma 6.5 to obtain the inequality in (6.6.13).

From Assumption 6.3 about the Lipschitz condition, we can deduce that there exist positive definite diagonal matrices $G_p^1 = \text{diag}\left(r_{j,p}^1\right)_{1\leq j\leq N}$ and $G_p^2 = \text{diag}\left(r_{j,p}^2\right)_{1\leq j\leq N}$, such that

$$0 \leq [\tilde{x}'(t)]_p^H \overline{L_{[\tilde{f}']_p}}^H G_p^1 \overline{L_{[\tilde{f}']_p}}[\tilde{x}'(t)]_p - [\tilde{f}']_p^H(\tilde{x}(t))G_p^1[\tilde{f}']_p(\tilde{x}(t)), \tag{6.6.14}$$

$$0 \leq [\tilde{x}'(t-\tau(t))]_p^H \overline{L_{[\tilde{f}']_p}}^H G_p^2 \overline{L_{[\tilde{f}']_p}}[\tilde{x}'(t-\tau(t))]_p - [\tilde{f}']_p^H(\tilde{x}(t-\tau(t)))G_p^2[\tilde{f}']_p(\tilde{x}(t-\tau(t))), \tag{6.6.15}$$

where we denoted $\overline{L_{[\tilde{f}']_p}} = \text{diag}\left(l_j^{[f']_p}\right)_{1\leq j\leq N}$, and $l_j^{[f']_p}$ are the Lipschitz constants of the functions f_j, $\forall p \in \{0,1,2,3\}$, $\forall j = \overline{1,N}$.

Also, for any matrices N_p^1, N_p^2, N_p^3, N_p^4, $N_p^5 \in \mathbb{C}^{4N\times 4N}$ we have that

$$\begin{aligned}
0 &= \left[[\dot{\tilde{x}}'(t)]_p^H N_p^1 + [\tilde{x}'(t-\delta)]_p^H N_p^2 - [\tilde{f}']_p^H(\tilde{x}(t))N_p^3 - [\tilde{f}']_p^H(\tilde{x}(t-\tau(t)))N_p^4 - \left(\int_{t-\eta(t)}^{t}[\tilde{f}']_p(\tilde{x}(s))ds\right)^H N_p^5\right]\\
&\quad\times\left[-[\dot{\tilde{x}}'(t)]_p - D[\tilde{x}'(t-\delta)]_p + \overline{A}_p\tilde{f}(\tilde{y}(t)) + \overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}_p\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right]\\
&= -[\dot{\tilde{x}}'(t)]_p^H N_p^1[\dot{\tilde{x}}'(t)]_p - [\dot{\tilde{x}}'(t)]_p^H N_p^1 D[\tilde{x}'(t-\delta)]_p + [\dot{\tilde{x}}'(t)]_p^H N_p^1\overline{A}_p\tilde{f}(\tilde{y}(t)) + [\dot{\tilde{x}}'(t)]_p^H N_p^1\overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t)))\\
&\quad+[\dot{\tilde{x}}'(t)]_p^H N_p^1\overline{C}_p\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds - [\tilde{x}'(t-\delta)]_p^H N_p^2[\dot{\tilde{x}}'(t)]_p - [\tilde{x}'(t-\delta)]_p^H N_p^2 D[\tilde{x}'(t-\delta)]_p
\end{aligned}$$

$$
\begin{aligned}
&+[\tilde{x}'(t-\delta)]_p^H N_p^2 \overline{A}_p \tilde{f}(\tilde{y}(t)) + [\tilde{x}'(t-\delta)]_p^H N_p^2 \overline{B}_p \tilde{f}(\tilde{y}(t-\tau(t))) + [\tilde{x}'(t-\delta)]_p^H N_p^2 \overline{C}_p \int_{t-\eta(t)}^{t} \tilde{f}(\tilde{y}(s))ds \\
&+[\tilde{f}']_p^H(\tilde{x}(t)) N_p^3 [\dot{\tilde{x}}'(t)]_p + [\tilde{f}']_p^H(\tilde{x}(t)) N_p^3 D[\tilde{x}'(t-\delta)]_p - [\tilde{f}']_p^H(\tilde{x}(t)) N_p^3 \overline{A}_p \tilde{f}(\tilde{y}(t)) \\
&-[\tilde{f}']_p^H(\tilde{x}(t)) N_p^3 \overline{B}_p \tilde{f}(\tilde{y}(t-\tau(t))) - [\tilde{f}']_p^H(\tilde{x}(t)) N_p^3 \overline{C}_p \int_{t-\eta(t)}^{t} \tilde{f}(\tilde{y}(s))ds \\
&+[\tilde{f}']_p^H(\tilde{x}(t-\tau(t))) N_p^4 [\dot{\tilde{x}}'(t)]_p + [\tilde{f}']_p^H(\tilde{x}(t-\tau(t))) N_p^4 D[\tilde{x}'(t-\delta)]_p - [\tilde{f}']_p^H(\tilde{x}(t-\tau(t))) N_p^4 \overline{A}_p \tilde{f}(\tilde{y}(t)) \\
&-[\tilde{f}']_p^H(\tilde{x}(t-\tau(t))) N_p^4 \overline{B}_p \tilde{f}(\tilde{y}(t-\tau(t))) - [\tilde{f}']_p^H(\tilde{x}(t-\tau(t))) N_p^4 \overline{C}_p \int_{t-\eta(t)}^{t} \tilde{f}(\tilde{y}(s))ds \\
&+\left(\int_{t-\eta(t)}^{t} [\tilde{f}']_p(\tilde{x}(s))ds\right)^H N_p^5 [\dot{\tilde{x}}'(t)]_p + \left(\int_{t-\eta(t)}^{t} [\tilde{f}']_p(\tilde{x}(s))ds\right)^H N_p^5 D[\tilde{x}'(t-\delta)]_p \\
&-\left(\int_{t-\eta(t)}^{t} [\tilde{f}']_p(\tilde{x}(s))ds\right)^H N_p^5 \overline{A}_p \tilde{f}(\tilde{y}(t)) - \left(\int_{t-\eta(t)}^{t} [\tilde{f}']_p(\tilde{x}(s))ds\right)^H N_p^5 \overline{B}_p \tilde{f}(\tilde{y}(t-\tau(t))) \\
&-\left(\int_{t-\eta(t)}^{t} [\tilde{f}']_p(\tilde{x}(s))ds\right)^H N_p^5 \overline{C}_p \int_{t-\eta(t)}^{t} \tilde{f}(\tilde{y}(s))ds. \qquad (6.6.16)
\end{aligned}
$$

Now, taking the conjugate of (6.6.16), adding it to (6.6.16), and then multiplying the result by $e^{2\varepsilon t}$, then multiplying inequalities (6.6.14)–(6.6.15) by $e^{2\varepsilon t}$ and adding them all to (6.6.6)–(6.6.13), we obtain

$$\dot{V}(t) \leq \sum_{p=0}^{3} \zeta_p^H(t) \Pi_p \zeta_p(t),$$

where Π_p are defined by (6.6.5), and

$$
\begin{aligned}
\zeta_p(t) \;=\; \Big[\; & [\tilde{x}'(t)]_p^H \quad [\dot{\tilde{x}}'(t)]_p^H \quad [\tilde{x}'(t-\delta)]_p^H \quad [\tilde{x}'(t-\tau)]_p^H \quad [\tilde{x}'(t-\tau(t))]_p^H \quad \tilde{f}^H(\tilde{y}(t)) \\
& \tilde{f}^H(\tilde{y}(t-\tau(t))) \quad \left(\textstyle\int_{t-\delta}^{t} [\tilde{x}'(s)]_p ds\right)^H \quad \left(\textstyle\int_{t-\tau}^{t} [\tilde{x}'(s)]_p ds\right)^H \quad \left(\textstyle\int_{t-\eta(t)}^{t} \tilde{f}(\tilde{y}(s))ds\right)^H \;\Big]^H,
\end{aligned}
$$

$\forall p \in \{0,1,2,3\}$. From (6.6.5) we have that $\Pi_p < 0$, $\forall p \in \{0,1,2,3\}$, which implies that $\dot{V}(t) < 0$, meaning that $V(t)$ is strictly decreasing for $t \geq 0$. It can be further deduced from the definition of $V(t)$ that

$$\sum_{p=0}^{3} e^{2\varepsilon t} \lambda_{\min}(P_p) ||[\tilde{x}'(t)]_p||^2 \leq \sum_{p=0}^{3} e^{2\varepsilon t} [\tilde{x}'(t)]_p^H P_p [\tilde{x}'(t)]_p \leq V(t) \leq V_0, \; \forall t \geq T, \; T \geq 0,$$

where $V_0 = \max\limits_{0 \leq t \leq T} V(t)$. Thus,

$$\sum_{p=0}^{3} ||[\tilde{x}'(t)]_p||^2 \leq \frac{V_0}{\min_{p\in\{0,1,2,3\}} \lambda_{\min}(P_p)} e^{-2\varepsilon t} \Leftrightarrow \sqrt{\sum_{p=0}^{3} ||[\tilde{x}'(t)]_p||^2} \leq M e^{-\varepsilon t}, \; \forall t \geq 0,$$

where $M = \sqrt{\frac{V_0}{\min_{p\in\{0,1,2,3\}} \lambda_{\min}(P_p)}}$. The above inequality proves the global exponential stability of the origin of system (6.6.3), completing the proof of the theorem. □

Remark 6.6. Although it seems complicated, the LMI sufficient criterion in (6.6.5) is very similar to the ones obtained in the literature for real-valued networks [236]. The computing complexity expressed as the total number of *real* scalar decision variables (NDV) is $76n^2 + 44n$. For comparison, the NDV of Theorem 1 by [236] is $15n^2 + 16n$, which would yield and NDV

of $120n^2 + 128n$ for OVNNs, because they have 8 times the number of real parameters. It can be seen that our result has a lower computational complexity than the one given by [236], for similar real-valued networks.

Now, we give an LMI-based sufficient condition for the exponential stability of the origin of (6.6.4).

Theorem 6.5. *If Assumptions 6.2 and 6.4 hold, then the origin of system* (6.6.4) *is globally exponentially stable if there exist positive definite matrices P, Q, R, S, T^1, T^2, U, W, $Z \in \mathbb{C}^{4N\times 4N}$, any matrices N^1, N^2, N^3, N^4, $N^5 \in \mathbb{C}^{4N\times 4N}$, and positive definite block-diagonal matrices G^1, $G^2 \in \mathbb{R}^{4N\times 4N}$, such that the following linear matrix inequality (LMI) holds*

$$(\Pi)_{10\times 10} < 0, \tag{6.6.17}$$

where

$\Pi_{1,1} = 2\varepsilon P+2\varepsilon Q-\overline{D}Q-Q\overline{D}+R+\delta^2 S+T^1-e^{-2\varepsilon\tau}U-4\tau^2 e^{-2\varepsilon\tau}Z+\overline{L_{\tilde{f}}}^H G^1\overline{L_{\tilde{f}}}$, $\Pi_{1,2} = Q$, $\Pi_{1,3} = -P\overline{D} + Q\overline{D}$, $\Pi_{1,4} = e^{-2\varepsilon\tau}U$, $\Pi_{1,6} = P\overline{A}$, $\Pi_{1,7} = P\overline{B}$, $\Pi_{1,8} = -2\varepsilon Q\overline{D} + \overline{D}Q\overline{D}$, $\Pi_{1,9} = 4\tau e^{-2\varepsilon\tau}Z$, $\Pi_{1,10} = P\overline{C}$, $\Pi_{2,2} = \tau^2 U + \tau^4 Z - N^1 - (N^1)^H$, $\Pi_{2,3} = -N^1\overline{D} - (N^2)^H$, $\Pi_{2,6} = N^1\overline{A} + (N^3)^H$, $\Pi_{2,7} = N^1\overline{B} + (N^4)^H$, $\Pi_{2,8} = -Q\overline{D}$, $\Pi_{2,10} = N^1\overline{C} + (N^5)^H$, $\Pi_{3,3} = -e^{-2\varepsilon\delta}R-N^2\overline{D}-\overline{D}(N^2)^H$,$\Pi_{3,6} = N^2\overline{A}+\overline{D}(N^3)^H$, $\Pi_{3,7} = N^2\overline{B}+\overline{D}(N^4)^H$, $\Pi_{3,8} = -\overline{D}Q\overline{D}$, $\Pi_{3,10} = N^2\overline{C} + \overline{D}(N^5)^H$, $\Pi_{4,4} = -e^{-2\varepsilon\tau}U$, $\Pi_{5,5} = -(1-\tau')e^{-2\varepsilon\tau}T^1 + \overline{L_{\tilde{f}}}^H G^2\overline{L_{\tilde{f}}}$, $\Pi_{6,6} = T^2+\eta^2 W-N^3\overline{A}-\overline{A}^H(N^3)^H-G^1$, $\Pi_{6,7} = -N^3\overline{B}-\overline{A}^H(N^4)^H$, $\Pi_{6,10} = -N^3\overline{C}-\overline{A}^H(N^5)^H$, $\Pi_{7,7} = -(1-\tau')e^{-2\varepsilon\tau}T^2 - N^4\overline{B} - \overline{B}^H(N^4)^H - G^2$, $\Pi_{7,10} = -N^4\overline{C} - \overline{B}^H(N^5)^H$, $\Pi_{8,8} = 2\varepsilon\overline{D}Q\overline{D} - e^{-2\varepsilon\delta}S$, $\Pi_{9,9} = -4e^{-2\varepsilon\tau}Z$, $\Pi_{10,10} = -e^{-2\varepsilon\eta}W - N^5\overline{C} - \overline{C}^H(N^5)^H$.

Proof. Consider the Lyapunov–Krasovskii functional

$$V(t) = V_1(t) + V_2(t) + V_3(t) + V_4(t) + V_5(t) + V_6(t) + V_7(t) + V_8(t),$$

$$V_1(t) = e^{2\varepsilon t}\tilde{y}^H(t)P\tilde{y}(t),$$

$$V_2(t) = e^{2\varepsilon t}\left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right)^H Q\left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right),$$

$$V_3(t) = \int_{t-\delta}^{t} e^{2\varepsilon s}\tilde{y}^H(s)R\tilde{y}(s)ds,$$

$$V_4(t) = \delta\int_{-\delta}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\tilde{y}^H(s)S\tilde{y}(s)dsd\theta,$$

$$V_5(t) = \int_{t-\tau(t)}^{t} e^{2\varepsilon s}\xi^H(s)T\xi(s)ds, \quad T = \operatorname{diag}(T^1, T^2),$$

$$\xi(s) = \left[[\tilde{y}(s)]^H \quad [\tilde{f}]^H(\tilde{y}(s))\right]^H,$$

$$V_6(t) = \tau\int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)U\dot{\tilde{y}}(s)dsd\theta,$$

$$V_7(t) = \eta\int_{-\eta}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\tilde{f}^H(\tilde{y}(s))W\tilde{f}(\tilde{y}(s))dsd\theta,$$

$$V_8(t) = 2\tau^2\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)Z\dot{\tilde{y}}(s)dsd\lambda d\theta.$$

The derivative of V along the trajectories of system (6.6.4) is

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t) + \dot{V}_4(t) + \dot{V}_5(t) + \dot{V}_6(t) + \dot{V}_7(t) + \dot{V}_8(t),$$

$$\begin{aligned}
\dot{V}_1(t) &= e^{2\varepsilon t}\left(2\varepsilon\tilde{y}^H(t)P\tilde{y}(t) + \dot{\tilde{y}}^H(t)P\tilde{y}(t) + \tilde{y}^H(t)P\dot{\tilde{y}}(t)\right) \\
&= e^{2\varepsilon t}\left(2\varepsilon\tilde{y}^H(t)P\tilde{y}(t) + \left(-\overline{D}\tilde{y}(t-\delta) + \overline{A}\tilde{f}(\tilde{y}(t)) + \overline{B}\tilde{f}(\tilde{y}(t-\tau(t)))\right.\right. \\
&\quad \left.+\overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H P\tilde{y}(t) + \tilde{y}^H(t)P\left(-\overline{D}\tilde{y}(t-\delta) + \overline{A}\tilde{f}(\tilde{y}(t))\right. \\
&\quad \left.\left.+\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)\right),
\end{aligned} \tag{6.6.18}$$

$$\begin{aligned}
\dot{V}_2(t) &= e^{2\varepsilon t}\left(2\varepsilon\left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right)^H Q\left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right)\right. \\
&\quad + \left(\dot{\tilde{y}}(t) - \overline{D}\tilde{y}(t) + \overline{D}\tilde{y}(t-\delta)\right)^H Q\left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right) \\
&\quad \left.+\left(\tilde{y}(t) - \overline{D}\int_{t-\delta}^{t}\tilde{y}(s)ds\right)^H Q\left(\dot{\tilde{y}}(t) - \overline{D}\tilde{y}(t) + \overline{D}\tilde{y}(t-\delta)\right)\right),
\end{aligned} \tag{6.6.19}$$

$$\dot{V}_3(t) = e^{2\varepsilon t}\left(\tilde{y}^H(t)R\tilde{y}(t) - e^{-2\varepsilon\delta}\tilde{y}^H(t-\delta)R\tilde{y}(t-\delta)\right), \tag{6.6.20}$$

$$\begin{aligned}
\dot{V}_4(t) &= \delta^2 e^{2\varepsilon t}\tilde{y}^H(t)S\tilde{y}(t) - \delta\int_{t-\delta}^{t}e^{2\varepsilon s}\tilde{y}^H(s)S\tilde{y}(s)ds \\
&\leq e^{2\varepsilon t}\left(\delta^2\tilde{y}^H(t)S\tilde{y}(t) - e^{-2\varepsilon\delta}\left(\int_{t-\delta}^{t}\tilde{y}(s)ds\right)^H S\left(\int_{t-\delta}^{t}\tilde{y}(s)ds\right)\right),
\end{aligned} \tag{6.6.21}$$

$$\begin{aligned}
\dot{V}_5(t) &= e^{2\varepsilon t}\left(\xi^H(t)T\xi(t) - (1-\dot{\tau}(t))e^{-2\varepsilon\tau(t)}\xi^H(t-\tau(t))T\xi(t-\tau(t))\right) \\
&\leq e^{2\varepsilon t}\left(\xi^H(t)T\xi(t) - (1-\tau')e^{-2\varepsilon\tau}\xi^H(t-\tau(t))T\xi(t-\tau(t))\right),
\end{aligned} \tag{6.6.22}$$

$$\begin{aligned}
\dot{V}_6(t) &= \tau^2 e^{2\varepsilon t}\dot{\tilde{y}}^H(t)U\dot{\tilde{y}}(t) - \tau\int_{t-\tau}^{t}e^{2\varepsilon s}\dot{\tilde{y}}^H(s)U\dot{\tilde{y}}(s)ds \\
&\leq e^{2\varepsilon t}\left(\tau^2\dot{\tilde{y}}^H(t)U\dot{\tilde{y}}(t) - e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\dot{\tilde{y}}(s)ds\right)^H U\left(\int_{t-\tau}^{t}\dot{\tilde{y}}(s)ds\right)\right) \\
&= e^{2\varepsilon t}\left(\tau^2\dot{\tilde{y}}^H(t)U\dot{\tilde{y}}(t) - e^{-2\varepsilon\tau}\left(\tilde{y}(t) - \tilde{y}(t-\tau)\right)^H U\left(\tilde{y}(t) - \tilde{y}(t-\tau)\right)\right),
\end{aligned} \tag{6.6.23}$$

$$\begin{aligned}
\dot{V}_7(t) &= \eta^2 e^{2\varepsilon t}\tilde{f}^H(\tilde{y}(t))W\tilde{f}(\tilde{y}(t)) - \eta\int_{t-\eta}^{t}e^{2\varepsilon s}\tilde{f}^H(\tilde{y}(s))W\tilde{f}(\tilde{y}(s))ds \\
&\leq \eta^2 e^{2\varepsilon t}\tilde{f}^H(\tilde{y}(t))W\tilde{f}(\tilde{y}(t)) - \eta(t)\int_{t-\eta(t)}^{t}e^{2\varepsilon s}\tilde{f}^H(\tilde{y}(s))W\tilde{f}(\tilde{y}(s))ds \\
&\leq e^{2\varepsilon t}\left(\eta^2\tilde{f}^H(\tilde{y}(t))W\tilde{f}(\tilde{y}(t)) - e^{-2\varepsilon\eta(t)}\left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H W\left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)\right) \\
&\leq e^{2\varepsilon t}\left(\eta^2\tilde{f}^H(\tilde{y}(t))W\tilde{f}(\tilde{y}(t)) - e^{-2\varepsilon\eta}\left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H W\left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)\right),
\end{aligned} \tag{6.6.24}$$

$$\begin{aligned}
\dot{V}_8(t) &= \tau^4 e^{2\varepsilon t}\dot{\tilde{y}}^H(t)Z\dot{\tilde{y}}(t) - 2\tau^2\int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)Z\dot{\tilde{y}}(s)dsd\theta \\
&\leq e^{2\varepsilon t}\left(\tau^4\dot{\tilde{y}}^H(t)Z\dot{\tilde{y}}(t) - 4e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\int_{t+\theta}^{t}\dot{\tilde{y}}(s)dsd\theta\right)^H Z\left(\int_{-\tau}^{0}\int_{t+\theta}^{t}\dot{\tilde{y}}(s)dsd\theta\right)\right) \\
&= e^{2\varepsilon t}\left(\tau^4\dot{\tilde{y}}^H(t)Z\dot{\tilde{y}}(t) - 4e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\left(\tilde{y}(t)-\tilde{y}(t+\theta)\right)d\theta\right)^H Z\left(\int_{-\tau}^{0}\left(\tilde{y}(t)-\tilde{y}(t+\theta)\right)d\theta\right)\right) \\
&= e^{2\varepsilon t}\left(\tau^4\dot{\tilde{y}}^H(t)Z\dot{\tilde{y}}(t) - 4e^{-2\varepsilon\tau}\left(\tau\tilde{y}(t)-\int_{t-\tau}^{t}\tilde{y}(s)ds\right)^H Z\left(\tau\tilde{y}(t)-\int_{t-\tau}^{t}\tilde{y}(s)ds\right)\right), \quad (6.6.25)
\end{aligned}$$

where we have used Assumption 6.2 to obtain the inequalities in (6.6.22) and (6.6.24), Lemma 6.4 to obtain the inequalities in (6.6.21), (6.6.23), and (6.6.24), and Lemma 6.5 to obtain the inequality in (6.6.25).

From Assumption 6.4 about the Lipschitz condition, we can deduce that there exist positive definite block-diagonal matrices $G^1 = \text{diag}\left(r_j^1 I_4\right)_{1\leq j\leq N}$ and $G^2 = \text{diag}\left(r_j^2 I_4\right)_{1\leq j\leq N}$, such that

$$0 \leq \tilde{y}^H(t)\overline{L_{\tilde{f}}}^H G^1\overline{L_{\tilde{f}}}\tilde{y}(t) - \tilde{f}^H(\tilde{y}(t))G^1\tilde{f}(\tilde{y}(t)), \quad (6.6.26)$$

$$0 \leq \tilde{y}^H(t-\tau(t))\overline{L_{\tilde{f}}}^H G^2\overline{L_{\tilde{f}}}\tilde{y}(t-\tau(t)) - \tilde{f}^H(\tilde{y}(t-\tau(t)))G^2\tilde{f}(\tilde{y}(t-\tau(t))), \quad (6.6.27)$$

where we denoted $\overline{L_{\tilde{f}}} = \text{diag}\left(l_j^f I_4\right)_{1\leq j\leq N}$, and l_j^f are the Lipschitz constants of the functions $f_j, \forall j = \overline{1,N}$.

Also, for any matrices N^1, N^2, N^3, N^4, $N^5 \in \mathbb{C}^{4N\times 4N}$ we have that

$$\begin{aligned}
0 &= \left[\dot{\tilde{y}}^H(t)N^1 + \tilde{y}^H(t-\delta)N^2 - \tilde{f}^H(\tilde{y}(t))N^3 - \tilde{f}(\tilde{y}(t-\tau(t)))N^4 - \left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H N^5\right] \\
&\quad\times\left[-\dot{\tilde{y}}(t) - \overline{D}\tilde{y}(t-\delta) + \overline{A}\tilde{f}(\tilde{y}(t)) + \overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right] \\
&= -\dot{\tilde{y}}^H(t)N^1\dot{\tilde{y}}(t) - \dot{\tilde{y}}^H(t)N^1\overline{D}\tilde{y}(t-\delta) + \dot{\tilde{y}}^H(t)N^1\overline{A}\tilde{f}(\tilde{y}(t)) + \dot{\tilde{y}}^H(t)N^1\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad+\dot{\tilde{y}}^H(t)N^1\overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds - \tilde{y}^H(t-\delta)N^2\dot{\tilde{y}}(t) - \tilde{y}^H(t-\delta)N^2\overline{D}\tilde{y}(t-\delta) \\
&\quad+\tilde{y}^H(t-\delta)N^2\overline{A}\tilde{f}(\tilde{y}(t)) + \tilde{y}^H(t-\delta)N^2\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) + \tilde{y}^H(t-\delta)N^2\overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds \\
&\quad+\tilde{f}^H(\tilde{y}(t))N^3\dot{\tilde{y}}(t) + \tilde{f}^H(\tilde{y}(t))N^3\overline{D}\tilde{y}(t-\delta) - \tilde{f}^H(\tilde{y}(t))N^3\overline{A}\tilde{f}(\tilde{y}(t)) - \tilde{f}^H(\tilde{y}(t))N^3\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad-\tilde{f}^H(\tilde{y}(t))N^3\overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds + \tilde{f}(\tilde{y}(t-\tau(t)))N^4\dot{\tilde{y}}(t) + \tilde{f}(\tilde{y}(t-\tau(t)))N^4\overline{D}\tilde{y}(t-\delta) \\
&\quad-\tilde{f}(\tilde{y}(t-\tau(t)))N^4\overline{A}\tilde{f}(\tilde{y}(t)) - \tilde{f}(\tilde{y}(t-\tau(t)))N^4\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad-\tilde{f}(\tilde{y}(t-\tau(t)))N^4\overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds + \left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H N^5\dot{\tilde{y}}(t) \\
&\quad+\left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H N^5\overline{D}\tilde{y}(t-\delta) - \left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H N^5\overline{A}\tilde{f}(\tilde{y}(t)) \\
&\quad-\left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H N^5\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) - \left(\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds\right)^H N^5\overline{C}\int_{t-\eta(t)}^{t}\tilde{f}(\tilde{y}(s))ds.
\end{aligned} \quad (6.6.28)$$

Now, taking the conjugate of (6.6.28), adding it to (6.6.28), and then multiplying the result by $e^{2\varepsilon t}$, then multiplying inequalities (6.6.26)–(6.6.27) by $e^{2\varepsilon t}$ and adding them all to (6.6.18)–(6.6.25), we obtain

$$\dot{V}(t) \leq \zeta^H(t)\Pi\zeta(t),$$

where Π is defined by (6.6.17), and

$$\begin{aligned}\zeta(t) \;=\; \big[\; & \tilde{y}^H(t) \quad \dot{\tilde{y}}^H(t) \quad \tilde{y}^H(t-\delta) \quad \tilde{y}^H(t-\tau) \quad \tilde{y}^H(t-\tau(t)) \quad \tilde{f}^H(\tilde{y}(t)) \quad \tilde{f}^H(\tilde{y}(t-\tau(t))) \\ & \left(\int_{t-\delta}^t \tilde{y}(s)ds\right)^H \quad \left(\int_{t-\tau}^t \tilde{y}(s)ds\right)^H \quad \left(\int_{t-\eta(t)}^t \tilde{f}(\tilde{y}(s))ds\right)^H \;\big]^H.\end{aligned}$$

From (6.6.17) we have that $\Pi < 0$, which implies that $\dot{V}(t) < 0$, meaning that $V(t)$ is strictly decreasing for $t \geq 0$. It can be further deduced from the definition of $V(t)$ that

$$e^{2\varepsilon t}\lambda_{\min}(P)||\tilde{y}(t)||^2 \leq e^{2\varepsilon t}\tilde{y}^H(t)P\tilde{y}(t) \leq V(t) \leq V_0,\ \forall t \geq T,\ T \geq 0,$$

where $V_0 = \max\limits_{0\leq t\leq T} V(t)$. Thus,

$$||\tilde{y}(t)||^2 \leq \frac{V_0}{\lambda_{\min}(P)}e^{-2\varepsilon t} \Leftrightarrow ||\tilde{y}(t)|| \leq Me^{-\varepsilon t},\ \forall t \geq 0,$$

where $M = \sqrt{\frac{V_0}{\lambda_{\min}(P)}}$. The above inequality proves the global exponential stability of the origin of system (6.6.4), completing the proof of the theorem. □

Remark 6.7. The computation complexity expressed in terms of *real* scalar decision variables (NDV) for the LMI in (6.6.17) is $304n^2 + 44n$, which is a bigger than the NDV for the criteria given in Theorem 6.4. However, taking into account Remark 6.4, it can be seen that Theorem 6.5 is more general, i.e., less conservative, than Theorem 6.4, which means that for any problem for which Theorem 6.4 can be applied, Theorem 6.5 can be applied as well, but not the other way around. This observation is consistent with the ones made by [236], where it is shown that greater complexity amounts to lower conservativeness, and lower complexity translates to greater conservativeness. Which one should be used is dependent on the problem at hand. Nonetheless, both stability criteria can be easily and rapidly solved using the effective YALMIP MATLAB tool.

6.6.2 Numerical examples

In order to prove the effectiveness of the theoretical results, we give two numerical examples.

Example 6.4. Consider the following two-neuron octonion-valued Hopfield neural network with leakage delay and mixed delays:

$$\begin{cases} \dot{x}_1(t) = & -d_1x_1(t-\delta) + \sum_{j=1}^2 a_{1j}f_j(x_j(t)) + \sum_{j=1}^2 b_{1j}f_j(x_j(t-\tau_{1j}(t))) \\ & + \sum_{j=1}^2 c_{1j}\int_{t-\eta_{1j}(t)}^t f_j(x_j(s))ds + u_1, \\ \dot{x}_2(t) = & -d_2x_2(t-\delta) + \sum_{j=1}^2 a_{2j}f_j(x_j(t)) + \sum_{j=1}^2 b_{2j}f_j(x_j(t-\tau_{2j}(t))) \\ & + \sum_{j=1}^2 c_{2j}\int_{t-\eta_{2j}(t)}^t f_j(x_j(s))ds + u_2, \end{cases} \tag{6.6.29}$$

where

$d_1 = 6$, $d_2 = 3$,

$a_{11} = (0.2 + 0.1\imath) + (0.5 - 0.2\imath)\jmath + (0.4 + 0.3\imath)\ell + (0.3 + 0.1\imath)\jmath\ell$,

$a_{12} = (-0.3 + 0.3\imath) + (-0.4 - 0.2\imath)\jmath + (0.3 + 0.1\imath)\ell + (0.2 - 0.1\imath)\jmath\ell$,
$a_{21} = (-0.4 + 0.3\imath) + (-0.3 - 0.2\imath)\jmath + (0.6 + 0.2\imath)\ell + (0.5 + 0.1\imath)\jmath\ell$,
$a_{22} = (-0.5 + 0.2\imath) + (-0.2 - 0.4\imath)\jmath + (0.2 + 0.3\imath)\ell + (0.2 - 0.3\imath)\jmath\ell$,
$b_{11} = (0.6 + 0.4\imath) + (0.9 - 0.5\imath)\jmath + (0.7 + 0.6\imath)\ell + (0.4 - 0.2\imath)\jmath\ell$,
$b_{12} = (0.7 + 0.6\imath) + (0.8 - 0.4\imath)\jmath + (0.9 + 0.4\imath)\ell + (0.3 - 0.2\imath)\jmath\ell$,
$b_{21} = (0.8 + 0.3\imath) + (0.7 - 0.2\imath)\jmath + (0.6 + 0.1\imath)\ell + (0.1 - 0.4\imath)\jmath\ell$,
$b_{22} = (-0.9 + 0.2\imath) + (-0.6 - 0.3\imath)\jmath + (0.8 + 0.5\imath)\ell + (0.2 - 0.4\imath)\jmath\ell$,
$c_{11} = (-0.1 + 0.14\imath) + (-0.13 - 0.16\imath)\jmath + (0.12 + 0.15\imath)\ell + (0.15 - 0.18\imath)\jmath\ell$,
$c_{12} = (0.11 + 0.18\imath) + (0.12 - 0.15\imath) + (0.14 + 0.19\imath) + (0.12 - 0.17\imath)$,
$c_{21} = (0.12 + 0.15\imath) + (0.11 - 0.14\imath)\jmath + (0.13 + 0.17\imath)\ell + (0.14 - 0.12\imath)\jmath\ell$,
$c_{22} = (0.13 + 0.12\imath) + (0.14 - 0.13\imath)\jmath + (0.11 + 0.12\imath)\ell + (0.13 - 0.12\imath)\jmath\ell$,
$u_1 = 10((8 - 7\imath) + (6 - 5\imath)\jmath + (4 - 3\imath)\ell + (2 - 1\imath)\jmath\ell)$,
$u_2 = 10((1 - 2\imath) + (3 - 4\imath)\jmath + (5 - 6\imath)\ell + (7 - 8\imath)\jmath\ell)$,
$f_j\left(\sum_{q=0}^{7}[x]_q e_q\right) = \sum_{q=0}^{7}\frac{0.1}{1+e^{-[x]_q}}e_q$, from which we can deduce that $l_j^{[f']_p} = \frac{0.05}{\sqrt{2}}$, $\forall p \in \{0,1,2,3\}$, $\forall j = \overline{1,2}$, and thus Assumption 6.3 is satisfied. The leakage delay is $\delta = 0.02$, the time delays are $\tau_{1j}(t) = \tau_{2j}(t) = 0.5|\cos t|$, and the distributed delays are $\eta_{1j}(t) = \eta_{2j}(t) = 0.3|\sin\frac{t}{2}|$, $\forall j = \overline{1,2}$, and so $\tau = \tau' = 0.5$ and $\eta = 0.3$, which means that Assumption 6.2 is also satisfied.

It can be verified that the LMIs in (6.6.5) are satisfied for $G_0^1 = \text{diag}(5.3358, 5.1977)$, $G_1^1 = \text{diag}(5.3358, 5.1977)$, $G_2^1 = \text{diag}(5.3358, 5.1977)$, $G_3^1 = \text{diag}(5.3358, 5.1977)$, $G_0^2 = \text{diag}(4.7016, 3.1730)$, $G_1^2 = \text{diag}(4.7015, 3.1730)$, $G_2^2 = \text{diag}(4.7015, 3.1730)$, $G_3^2 = \text{diag}(4.7016, 5.3.1730)$, and $\varepsilon = 0.1$. (For brevity, the values of the other matrices are not given.) Thus, we can deduce from Theorem 6.4 that the equilibrium point of neural network (6.6.29) is globally exponentially stable.

Example 6.5. Consider the same two-neuron octonion-valued Hopfield neural network with leakage delay and mixed delays in (6.6.29), but with the following parameters

$d_1 = 35$, $d_2 = 25$,

$a_{11} = (1 + 3\imath) + (2 + 2\imath)\jmath + (3 - 1\imath)\ell + (-4 + 1\imath)\jmath\ell$, $a_{12} = (4 + 1\imath) + (3 - 2\imath)\jmath + (4 + 1\imath)\ell + (-2 + 3\imath)\jmath\ell$,

$a_{21} = (2-3\imath)+(2+1\imath)\jmath+(4-2\imath)\ell+(1+2\imath)\jmath\ell$, $a_{22} = (1+2\imath)+(3-2\imath)\jmath+(2+1\imath)\ell+(4-2\imath)\jmath\ell$,

$b_{11} = (2 + 1\imath) + (3 + 1\imath)\jmath + (-2 + 3\imath)\ell + (-1 + 4\imath)\jmath\ell$, $b_{12} = (-4 + 2\imath) + (-2 + 3\imath)\jmath + (1 + 2\imath)\ell + (-2 + 2\imath)\jmath\ell$,

$b_{21} = (1 - 4\imath) + (2 - 2\imath)\jmath + (1 + 2\imath)\ell + (3 + 2\imath)\jmath\ell$, $b_{22} = (1 + 2\imath) + (4 + 1\imath)\jmath + (2 - 3\imath)\ell + (-2 + 4\imath)\jmath\ell$,

$c_{11} = (1 + 0.5\imath) + (0.5 - 0.5\imath)\jmath + (1 - 1\imath)\ell + (-1 + 0.5\imath)\jmath\ell$, $c_{12} = (0.5 + 0.5\imath) + (1 - 0.5\imath)\jmath + (1 + 1\imath)\ell + (-0.5 + 0.5\imath)\jmath\ell$,

$c_{21} = (0.5 - 0.5\imath) + (1 + 1\imath)\jmath + (0.5 - 0.5\imath)\ell + (1 + 0.5\imath)\jmath\ell$, $c_{22} = (1 + 0.5\imath) + (0.5 - 0.5\imath)\jmath + (1 + 1\imath)\ell + (0.5 - 0.5\imath)\jmath\ell$,

$u_1 = 30((1 - 2\imath) + (3 - 4\imath)\jmath + (5 - 6\imath)\ell + (7 - 8\imath)\jmath\ell)$, $u_2 = 30((8 - 7\imath) + (6 - 5\imath)\jmath + (4 - 3\imath)\ell + (2 - 1\imath)\jmath\ell)$,

$f_j\left(\sum_{q=0}^{7}[x]_q e_q\right) = \sum_{q=0}^{7} 0.5\frac{1-e^{-[x]_q}}{1+e^{-[x]_q}}e_q$, from which we can deduce that $l_j^f = 0.5\sqrt{2}$, $\forall j = \overline{1,2}$, and thus Assumption 6.4 is satisfied. The leakage delay is $\delta = 0.02$, the time delays are $\tau_{1j}(t) = \tau_{2j}(t) = 0.4|\sin t|$, and the distributed delays are $\eta_{1j}(t) = \eta_{2j}(t) = 0.2|\cos\frac{t}{2}|$, $\forall j = \overline{1,2}$, and so $\tau = \tau' = 0.4$ and $\eta = 0.2$, which means that Assumption 6.2 is also satisfied.

It can be verified that the LMI in (6.6.17) is satisfied for $G^1 = \text{diag}(0.0070217I_4, 0.0068122I_4)$, $G^2 = \text{diag}(0.00079613I_4, 0.00079836I_4)$, and $\varepsilon = 0.1$. (For brevity, the values of the other ma-

trices are not given.) Thus, we can deduce from Theorem 6.5 that the equilibrium point of neural network (6.6.29) with the above parameters is globally exponentially stable.

Remark 6.8. It can be easily verified that Theorem 6.4 cannot be applied for Example 6.5, but Theorem 6.5 can be applied for both examples, which empirically confirms the correctness of the claims made in Remark 6.4 and Remark 6.7.

6.7 Exponential stability of OVNNs with leakage delay and mixed delays

As already mentioned in Section 6.6, time delays are known to appear in practical implementations of neural networks due to the finite switching speed of amplifiers, and can cause instability or chaotic behavior. In neutral-type systems, past derivative information is also considered to influence the present state. These systems more accurately describe the properties of neural reaction processes that naturally occur in the real world. Due to the existence of the neutral-type delays, the study of this type of systems is more complicated than that of the usual time-delayed models. Taking all the above into consideration, we aim to formulate global exponential stability criteria for neutral-type OVNNs with time-varying delays.

The presentation in this section follows that in the author's paper [167].

6.7.1 Main results

We will consider the neutral-type octonion-valued Hopfield neural networks defined by the following system of differential equations:

$$\begin{aligned}\dot{x}_i(t) &= -d_i x_i(t) + \sum_{j=1}^{N} a_{ij} f_j(x_j(t)) + \sum_{j=1}^{N} b_{ij} f_j(x_j(t-\tau_{ij}(t))) \\ &\quad + c_i \dot{x}_i(t-\eta_i(t)) + u_i, \quad i = \overline{1,N}, \end{aligned} \tag{6.7.1}$$

where the notations are the same as the ones in Section 6.6, and $C = \operatorname{diag}(c_1, \ldots, c_N) \in \mathbb{O}^{N\times N}$ is the neutral-type delay connection weight matrix, and $\eta_i : \mathbb{R} \to \mathbb{R}$ are the neutral-type delays, $\forall i,\ j = \overline{1,N}$.

In order to study the stability properties of (6.7.1), we also need the following assumption:

Assumption 6.5. *The time-varying delays $\tau_{ij} : \mathbb{R} \to \mathbb{R}$ and the neutral-type delays $\eta_i : \mathbb{R} \to \mathbb{R}$ are continuously differentiable functions and there exist $\tau,\ \eta > 0$ and $\tau',\ \eta' < 1$, such that $\tau_{ij}(t) < \tau$, $\eta_i(t) < \eta$, $\dot{\tau}_{ij}(t) \le \tau'$, $\dot{\eta}_i(t) \le \eta'$, $\forall t > 0$, $\forall i,\ j = \overline{1,N}$.*

By the Cayley–Dickson construction, system (6.7.1) and by shifting its equilibrium point $\hat{x}$ to the origin, system (6.7.1) can be written more compactly as:

$$\begin{aligned}[\dot{\tilde{x}}'(t)]_0 &= -D[\tilde{x}'(t)]_0 \\ &\quad + \Big([A']_0[\tilde{f}']_0(\tilde{x}(t)) - [A']_1[\tilde{f}']_1(\tilde{x}(t)) - [A']_2[\tilde{f}']_2(\tilde{x}(t)) - [A']_3[\tilde{f}']_3(\tilde{x}(t))\Big) \\ &\quad + \Big([B']_0[\tilde{f}']_0(\tilde{x}(t-\tau(t))) - [B']_1[\tilde{f}']_1(\tilde{x}(t-\tau(t))) \\ &\quad - [B']_2[\tilde{f}']_2(\tilde{x}(t-\tau(t))) - [B']_3[\tilde{f}']_3(\tilde{x}(t-\tau(t)))\Big) \\ &\quad + \big([C']_0[\dot{\tilde{x}}'(t-\eta(t))]_0 - [C']_1[\dot{\tilde{x}}'(t-\eta(t))]_1 \\ &\quad - [C']_2[\dot{\tilde{x}}'(t-\eta(t))]_2 - [C']_3[\dot{\tilde{x}}'(t-\eta(t))]_3\big)\,, \end{aligned}$$

$$
\begin{aligned}
[\dot{\tilde{x}}'(t)]_1 &= -D[\tilde{x}'(t)]_1 \\
&\quad + \Big([A']_0[\tilde{f}']_1(\tilde{x}(t)) + [A']_1[\tilde{f}']_0(\tilde{x}(t)) + [A']_2[\tilde{f}']_3(\tilde{x}(t)) - [A']_3[\tilde{f}']_2(\tilde{x}(t))\Big) \\
&\quad + \Big([B']_0[\tilde{f}']_1(\tilde{x}(t-\tau(t))) + [B']_1[\tilde{f}']_0(\tilde{x}(t-\tau(t))) \\
&\quad + [B']_2[\tilde{f}']_3(\tilde{x}(t-\tau(t))) - [B']_3[\tilde{f}']_2(\tilde{x}(t-\tau(t)))\Big) \\
&\quad + \big([C']_0[\dot{\tilde{x}}'(t-\eta(t))]_1 + [C']_1[\dot{\tilde{x}}'(t-\eta(t))]_0 \\
&\quad + [C']_2[\dot{\tilde{x}}'(t-\eta(t))]_3 - [C']_3[\dot{\tilde{x}}'(t-\eta(t))]_2\big)\,, \\
[\dot{\tilde{x}}'(t)]_2 &= -D[\tilde{x}'(t)]_2 \\
&\quad + \Big([A']_0[\tilde{f}']_2(\tilde{x}(t)) - [A']_1[\tilde{f}']_3(\tilde{x}(t)) + [A']_2[\tilde{f}']_0(\tilde{x}(t)) + [A']_3[\tilde{f}']_1(\tilde{x}(t))\Big) \\
&\quad + \Big([B']_0[\tilde{f}']_2(\tilde{x}(t-\tau(t))) - [B']_1[\tilde{f}']_3(\tilde{x}(t-\tau(t))) \\
&\quad + [B']_2[\tilde{f}']_0(\tilde{x}(t-\tau(t))) + [B']_3[\tilde{f}']_1(\tilde{x}(t-\tau(t)))\Big) \\
&\quad + \big([C']_0[\dot{\tilde{x}}'(t-\eta(t))]_2 - [C']_1[\dot{\tilde{x}}'(t-\eta(t))]_3 \\
&\quad + [C']_2[\dot{\tilde{x}}'(t-\eta(t))]_0 + [C']_3[\dot{\tilde{x}}'(t-\eta(t))]_1\big)\,, \\
[\dot{\tilde{x}}'(t)]_3 &= -D[\tilde{x}'(t)]_3 \\
&\quad + \Big([A']_0[\tilde{f}']_3(\tilde{x}(t)) + [A']_1[\tilde{f}']_2(\tilde{x}(t)) - [A']_2[\tilde{f}']_1(\tilde{x}(t)) + [A']_3[\tilde{f}']_0(\tilde{x}(t))\Big) \\
&\quad + \Big([B']_0[\tilde{f}']_3(\tilde{x}(t-\tau(t))) + [B']_1[\tilde{f}']_2(\tilde{x}(t-\tau(t))) \\
&\quad - [B']_2[\tilde{f}']_1(\tilde{x}(t-\tau(t))) + [B']_3[\tilde{f}']_0(\tilde{x}(t-\tau(t)))\Big) \\
&\quad + \big([C']_0[\dot{\tilde{x}}'(t-\eta(t))]_3 + [C']_1[\dot{\tilde{x}}'(t-\eta(t))]_2 \\
&\quad - [C']_2[\dot{\tilde{x}}'(t-\eta(t))]_1 + [C']_3[\dot{\tilde{x}}'(t-\eta(t))]_0\big)\,, \qquad (6.7.2)
\end{aligned}
$$

where $[\tilde{x}'(t)]_p = [x'(t)]_p - [\hat{x}']_p$, $[\tilde{f}']_p(\tilde{x}(t)) = [f']_p(x(t)) - [f']_p(\hat{x})$, $\forall p \in \{0,1,2,3\}$.

If we denote

$$
\tilde{y}(t) = \big([\tilde{x}'(t)]_0^H, [\tilde{x}'(t)]_1^H, [\tilde{x}'(t)]_2^H, [\tilde{x}'(t)]_3^H\big)^H \in \mathbb{C}^{4N},
$$

$$
\overline{D} = \begin{bmatrix} D & 0 & 0 & 0 \\ 0 & D & 0 & 0 \\ 0 & 0 & D & 0 \\ 0 & 0 & 0 & D \end{bmatrix} \in \mathbb{R}^{4N\times 4N},
$$

$$
\overline{A} = \begin{bmatrix} [A']_0 & -[A']_1 & -[A']_2 & -[A']_3 \\ [A']_1 & [A']_0 & -[A']_3 & [A']_2 \\ [A']_2 & [A']_3 & [A']_0 & -[A']_1 \\ [A']_3 & -[A']_2 & [A']_1 & [A']_0 \end{bmatrix} \in \mathbb{C}^{4N\times 4N},
$$

$$
\overline{B} = \begin{bmatrix} [B']_0 & -[B']_1 & -[B']_2 & -[B']_3 \\ [B']_1 & [B']_0 & -[B']_3 & [B']_2 \\ [B']_2 & [B']_3 & [B']_0 & -[B']_1 \\ [B']_3 & -[B']_2 & [B']_1 & [B']_0 \end{bmatrix} \in \mathbb{C}^{4N\times 4N},
$$

$$\overline{C} = \begin{bmatrix} [C']_0 & -[C']_1 & -[C']_2 & -[C']_3 \\ [C']_1 & [C']_0 & -[C']_3 & [C']_2 \\ [C']_2 & [C']_3 & [C']_0 & -[C']_1 \\ [C']_3 & -[C']_2 & [C']_1 & [C']_0 \end{bmatrix} \in \mathbb{C}^{4N\times 4N},$$

$$\tilde{f}(\tilde{y}(t)) = \left([\tilde{f}']_0^H(x(t)), [\tilde{f}']_1^H(x(t)), [\tilde{f}']_2^H(x(t)), [\tilde{f}']_3^H(x(t))\right)^H \in \mathbb{C}^{4N},$$

then system (6.7.2) becomes:

$$\dot{\tilde{y}}(t) = -\overline{D}\tilde{y}(t) + \overline{A}\tilde{f}(\tilde{y}(t)) + \overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}\dot{\tilde{y}}(t-\eta(t)). \tag{6.7.3}$$

Remark 6.9. Systems (6.7.2), (6.7.3), and (6.7.1) are equivalent. This means that any property of the origin of systems (6.7.2), (6.7.3) will also be true for the equilibrium point of system (6.7.1). Thus, from now on, we will only study the stability properties of the origin of systems (6.7.2), (6.7.3).

The following lemma will also be used to carry out the stability analysis:

Lemma 6.6. *([203]) For any vector function $x : [a, b] \to \mathbb{C}^{4N}$ and any positive definite matrix $M \in \mathbb{C}^{4N\times 4N}$, the following linear matrix inequality (LMI) holds:*

$$\left(\int_a^b\int_\theta^b\int_\lambda^b x(s)dsd\lambda d\theta\right)^H M \left(\int_a^b\int_\theta^b\int_\lambda^b x(s)dsd\lambda d\theta\right) \leq \frac{(b-a)^3}{6}\int_a^b\int_\theta^b\int_\lambda^b x^H(s)Mx(s)dsd\lambda d\theta,$$

where the integrals are well defined.

Theorem 6.6. *If Assumptions 6.5 and 6.3 hold, then the origin of system* (6.7.2) *is globally exponentially stable if there exist positive definite matrices P_p, Q_p^1, Q_p^2, M_p, R_p, S_p, T_p, U_p, W_p, $Z_p \in \mathbb{C}^{N\times N}$, any matrices N_p^1, N_p^2, N_p^3, N_p^4, $N_p^5 \in \mathbb{C}^{N\times N}$, positive definite diagonal matrices G_p^1, $G_p^2 \in \mathbb{R}^{N\times N}$, and a constant $\varepsilon > 0$, such that the following linear matrix inequalities (LMIs) hold*

$$(\Pi_p)_{20\times 20} < 0, \tag{6.7.4}$$

$\forall p \in \{0, 1, 2, 3\}$, *where* $(\Pi_p)_{1,1} = 2\varepsilon P_p - DP_p - P_pD + Q_p^1 + R_p - e^{-2\varepsilon\tau}T_p + \eta^2U_p - 4\tau^2e^{-2\varepsilon\tau}W_p - 9\tau^4e^{-2\varepsilon\tau}Z_p + \overline{L_{[\tilde{f}']_p}}^H G_p^1\overline{L_{[\tilde{f}']_p}} - N_p^2D - D(N_p^2)^H$, $(\Pi_p)_{1,2} = -N_p^2 - D(N_p^1)^H$, $(\Pi_p)_{1,4:7} = P_p\overline{C}_p + N_p^2\overline{C}_p + D(N_p^5)^H\overline{E}_p$, $(\Pi_p)_{1,8} = e^{-2\varepsilon\tau}T_p$, $(\Pi_p)_{1,10:13} = P_p\overline{A}_p + N_p^2\overline{A}_p + D(N_p^3)^H\overline{E}_p$, $(\Pi_p)_{1,14:17} = P_p\overline{B}_p + N_p^2\overline{B}_p + D(N_p^4)^H\overline{E}_p$, $(\Pi_p)_{1,19} = 4\tau e^{-2\varepsilon\tau}W_p$, $(\Pi_p)_{1,20} = 18\tau^2e^{-2\varepsilon\tau}Z_p$, $(\Pi_p)_{2,2} = M_p + S_p + \tau^2T_p + \tau^4W_p + \tau^6Z_p - N_p^1 - (N_p^1)^H$, $(\Pi_p)_{2,4:7} = N_p^1\overline{C}_p + (N_p^5)^H\overline{E}_p$, $(\Pi_p)_{2,10:13} = N_p^1\overline{A}_p + (N_p^3)^H\overline{E}_p$, $(\Pi_p)_{2,14:17} = N_p^1\overline{B}_p + (N_p^4)^H\overline{E}_p$, $(\Pi_p)_{3,3} = -e^{-2\varepsilon\eta}S_p$, $(\Pi_p)_{4:7,4:7} = -(1-\eta')e^{-2\varepsilon\eta}\overline{E}_p^HM_p\overline{E}_p - \overline{E}_p^HN_p^5\left(\overline{C}_p - \overline{C}_p\overline{E}_p^H\overline{E}_p\right) - \left(\overline{C}_p - \overline{C}_p\overline{E}_p^H\overline{E}_p\right)^H(N_p^5)^H\overline{E}_p - \frac{1}{4}\sum_{q=0}^3\overline{E}_q^H\left(N_q^5\overline{C}_q\overline{E}_q^H + \overline{E}_q\overline{C}_q^H(N_q^5)^H\right)\overline{E}_q$, $(\Pi_p)_{4:7,10:13} = -\overline{E}_p^HN_p^5\overline{A}_p - \overline{C}_p^H(N_p^3)^H\overline{E}_p$, $(\Pi_p)_{4:7,14:17} = -\overline{E}_p^HN_p^5\overline{B}_p - \overline{C}_p^H(N_p^4)^H\overline{E}_p$, $(\Pi_p)_{8,8} = -e^{-2\varepsilon\tau}R_p - e^{-2\varepsilon\tau}T_p$, $(\Pi_p)_{9,9} = -(1-\tau')e^{-2\varepsilon\tau}Q_p^1 + \overline{L_{[\tilde{f}']_p}}^H G_p^2\overline{L_{[\tilde{f}']_p}}$, $(\Pi_p)_{10:13,10:13} = \overline{E}_p^HQ_p^2\overline{E}_p - \overline{E}_p^HG_p^1\overline{E}_p - \overline{E}_p^HN_p^3\left(\overline{A}_p - \overline{A}_p\overline{E}_p^H\overline{E}_p\right) - \left(\overline{A}_p - \overline{A}_p\overline{E}_p^H\overline{E}_p\right)^H(N_p^3)^H\overline{E}_p - \frac{1}{4}\sum_{q=0}^3\overline{E}_q^H\left(N_q^3\overline{A}_q\overline{E}_q^H + \overline{E}_q\overline{A}_q^H(N_q^3)^H\right)\overline{E}_q$, $(\Pi_p)_{10:13,14:17} = -\overline{E}_p^HN_p^3\overline{B}_p - \overline{A}_p^H(N_p^4)^H\overline{E}_p$, $(\Pi_p)_{14:17,14:17} = -(1-\tau')e^{-2\varepsilon\tau}\overline{E}_p^HQ_p^2\overline{E}_p - \overline{E}_p^HG_p^2\overline{E}_p - \overline{E}_p^H\left(N_p^4\overline{B}_p - \overline{B}_p\overline{E}_p^H\overline{E}_p\right) - \left(\overline{B}_p - \overline{B}_p\overline{E}_p^H\overline{E}_p\right)^H(N^4)^H\overline{E}_p - \frac{1}{4}\sum_{q=0}^3\overline{E}_q^H\left(N_q^4\overline{B}_q\overline{E}_q^H + \overline{E}_q\overline{B}_q^H(N_q^4)^H\right)\overline{E}_q$, $(\Pi_p)_{18,18} = -e^{-2\varepsilon\eta}U_p$, $(\Pi_p)_{19,19} = -4e^{-2\varepsilon\tau}W_p$, $(\Pi_p)_{20,20} = -36e^{-2\varepsilon\tau}Z_p$.

We denoted by $\overline{A}_p$ the $(p+1)$-th row of $\overline{A}$ (and analogously for $\overline{B}_p$ and $\overline{C}_p$) and by $\overline{E}_p$ the $(p+1)$-th row of

$$\overline{E} = \begin{bmatrix} I_N & 0 & 0 & 0 \\ 0 & I_N & 0 & 0 \\ 0 & 0 & I_N & 0 \\ 0 & 0 & 0 & I_N \end{bmatrix} \in \mathbb{R}^{4N\times 4N},$$

$\forall p \in \{0,1,2,3\}$.

Proof. Consider the Lyapunov–Krasovskii functional

$$V(t) = V_1(t) + V_2(t) + V_3(t) + V_4(t) + V_5(t) + V_6(t) + V_7(t) + V_8(t) + V_9(t),$$

$$V_1(t) = \sum_{p=0}^{3} e^{2\varepsilon t}[\tilde{x}'(t)]_p^H P_p[\tilde{x}'(t)]_p,$$

$$V_2(t) = \sum_{p=0}^{3}\int_{t-\tau(t)}^{t} e^{2\varepsilon s}\xi_p^H(s)Q_p\xi_p(s)ds, \quad Q_p = \operatorname{diag}(Q_p^1, Q_p^2),$$

$$\xi_p(s) = \begin{bmatrix}[\tilde{x}'(s)]_p^H & [\tilde{f}']_p^H(\tilde{x}(s))\end{bmatrix}^H,$$

$$V_3(t) = \sum_{p=0}^{3}\int_{t-\eta(t)}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H M_p[\dot{\tilde{x}}'(s)]_p ds,$$

$$V_4(t) = \sum_{p=0}^{3}\int_{t-\tau}^{t} e^{2\varepsilon s}[\tilde{x}'(s)]_p^H R_p[\tilde{x}'(s)]_p ds,$$

$$V_5(t) = \sum_{p=0}^{3}\int_{t-\eta}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H S_p[\dot{\tilde{x}}'(s)]_p ds,$$

$$V_6(t) = \sum_{p=0}^{3}\tau\int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H T_p[\dot{\tilde{x}}'(s)]_p ds d\theta,$$

$$V_7(t) = \sum_{p=0}^{3}\eta\int_{-\eta}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H U_p[\dot{\tilde{x}}'(s)]_p ds d\theta,$$

$$V_8(t) = \sum_{p=0}^{3} 2\tau^2\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H W_p[\dot{\tilde{x}}'(s)]_p ds d\lambda d\theta,$$

$$V_9(t) = \sum_{p=0}^{3} 6\tau^3\int_{-\tau}^{0}\int_{\theta}^{0}\int_{\lambda}^{0}\int_{t+\mu}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H Z_p[\dot{\tilde{x}}'(s)]_p ds d\mu d\lambda d\theta.$$

The derivative of V along the trajectories of system (6.7.2) is

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t) + \dot{V}_4(t) + \dot{V}_5(t) + \dot{V}_6(t) + \dot{V}_7(t) + \dot{V}_8(t) + \dot{V}_9(t),$$

$$\begin{aligned}\dot{V}_1(t) &= \sum_{p=0}^{3} e^{2\varepsilon t}\left(2\varepsilon[\tilde{x}'(t)]_p^H P_p[\tilde{x}'(t)]_p + [\dot{\tilde{x}}'(t)]_p^H P_p[\tilde{x}'(t)]_p + [\tilde{x}'(t)]_p^H P_p[\dot{\tilde{x}}'(t)]_p\right)\\ &= \sum_{p=0}^{3} e^{2\varepsilon t}\Big(2\varepsilon[\tilde{x}'(t)]_p^H P_p[\tilde{x}'(t)]_p + \Big(-D[\tilde{x}'(t-\delta)]_p + \overline{A}_p\tilde{f}(\tilde{y}(t)) + \overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t)))\\ &\quad +\overline{C}_p\dot{\tilde{y}}(t-\eta(t))\Big)^H P_p[\tilde{x}'(t)]_p + [\tilde{x}'(t)]_p^H P_p\Big(-D[\tilde{x}'(t-\delta)]_p + \overline{A}_p\tilde{f}(\tilde{y}(t))\\ &\quad +\overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}_p\dot{\tilde{y}}(t-\eta(t))\Big)\Big), \end{aligned} \tag{6.7.5}$$

$$\begin{aligned}\dot{V}_2(t) &= \sum_{p=0}^{3} e^{2\varepsilon t}\left(\xi_p^H(t)Q_p\xi_p(t) - (1-\dot{\tau}(t))e^{-2\varepsilon\tau(t)}\xi_p^H(t-\tau(t))Q_p\xi_p(t-\tau(t))\right)\\ &\le \sum_{p=0}^{3} e^{2\varepsilon t}\left(\xi_p^H(t)Q_p\xi_p(t) - (1-\tau')e^{-2\varepsilon\tau}\xi_p^H(t-\tau(t))Q_p\xi_p(t-\tau(t))\right), \end{aligned} \tag{6.7.6}$$

$$\begin{aligned}\dot{V}_3(t) &= \sum_{p=0}^{3} e^{2\varepsilon t}\left([\dot{\tilde{x}}'(t)]_p^H M_p[\dot{\tilde{x}}'(t)]_p - (1-\dot{\eta}(t))e^{-2\varepsilon\eta(t)}[\dot{\tilde{x}}'(t-\eta(t))]_p^H M_p[\dot{\tilde{x}}'(t-\eta(t))]_p\right)\\ &\le \sum_{p=0}^{3} e^{2\varepsilon t}\left([\dot{\tilde{x}}'(t)]_p^H M_p[\dot{\tilde{x}}'(t)]_p - (1-\eta')e^{-2\varepsilon\eta}[\dot{\tilde{x}}'(t-\eta(t))]_p^H M_p[\dot{\tilde{x}}'(t-\eta(t))]_p\right), \end{aligned} \tag{6.7.7}$$

$$\dot{V}_4(t) = \sum_{p=0}^{3} e^{2\varepsilon t}\left([\tilde{x}'(t)]_p^H R_p[\tilde{x}'(t)]_p - e^{-2\varepsilon\tau}[\tilde{x}'(t-\tau)]_p^H R_p[\tilde{x}'(t-\tau)]_p\right), \tag{6.7.8}$$

$$\dot{V}_5(t) = \sum_{p=0}^{3} e^{2\varepsilon t}\left([\dot{\tilde{x}}'(t)]_p^H S_p[\dot{\tilde{x}}'(t)]_p - e^{-2\varepsilon\eta}[\dot{\tilde{x}}'(t-\eta)]_p^H S_p[\dot{\tilde{x}}'(t-\eta)]_p\right), \tag{6.7.9}$$

$$\begin{aligned}\dot{V}_6(t) &= \sum_{p=0}^{3}\left(\tau^2 e^{2\varepsilon t}[\dot{\tilde{x}}'(t)]_p^H T_p[\dot{\tilde{x}}'(t)]_p - \tau\int_{t-\tau}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H T_p[\dot{\tilde{x}}'(s)]_p ds\right)\\ &\le \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^2[\dot{\tilde{x}}'(t)]_p^H T_p[\dot{\tilde{x}}'(t)]_p - e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}[\dot{\tilde{x}}'(s)]_p ds\right)^H T_p\left(\int_{t-\tau}^{t}[\dot{\tilde{x}}'(s)]_p ds\right)\right)\\ &\quad \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^2[\dot{\tilde{x}}'(t)]_p^H T_p[\dot{\tilde{x}}'(t)]_p\right.\\ &= -e^{-2\varepsilon\tau}\left([\tilde{x}'(t)]_p - [\tilde{x}'(t-\tau)]_p\right)^H T_p\left([\tilde{x}'(t)]_p - [\tilde{x}'(t-\tau)]_p\right)\Big), \end{aligned} \tag{6.7.10}$$

$$\begin{aligned}\dot{V}_7(t) &= \sum_{p=0}^{3}\left(\eta^2 e^{2\varepsilon t}[\dot{\tilde{x}}'(t)]_p^H U_p[\dot{\tilde{x}}'(t)]_p - \eta\int_{t-\eta}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H U_p[\dot{\tilde{x}}'(s)]_p ds\right)\\ &\le \sum_{p=0}^{3} e^{2\varepsilon t}\left(\eta^2[\dot{\tilde{x}}'(t)]_p^H U_p[\dot{\tilde{x}}'(t)]_p - e^{-2\varepsilon\eta}\left(\int_{t-\eta}^{t}[\dot{\tilde{x}}'(s)]_p ds\right)^H U_p\left(\int_{t-\eta}^{t}[\dot{\tilde{x}}'(s)]_p ds\right)\right), \end{aligned} \tag{6.7.11}$$

$$
\begin{aligned}
\dot{V}_8(t) &= \sum_{p=0}^{3}\left(\tau^4 e^{2\varepsilon t}[\dot{\tilde{x}}'(t)]_p^H W_p[\dot{\tilde{x}}'(t)]_p - 2\tau^2\int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H W_p[\dot{\tilde{x}}'(s)]_p ds d\theta\right)\\
&\leq \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^4[\dot{\tilde{x}}'(t)]_p^H W_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad \left. -4e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\int_{t+\theta}^{t}[\dot{\tilde{x}}'(s)]_p ds d\theta\right)^H W_p\left(\int_{-\tau}^{0}\int_{t+\theta}^{t}[\dot{\tilde{x}}'(s)]_p ds d\theta\right)\right)\\
&= \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^4[\dot{\tilde{x}}'(t)]_p^H W_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad \left. -4e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}([\tilde{x}'(t)]_p - [\tilde{x}'(t+\theta)]_p)\, d\theta\right)^H W_p\left(\int_{-\tau}^{0}([\tilde{x}'(t)]_p - [\tilde{x}'(t+\theta)]_p)\, d\theta\right)\right)\\
&= \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^4[\dot{\tilde{x}}'(t)]_p^H W_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad \left. -4e^{-2\varepsilon\tau}\left(\tau[\tilde{x}'(t)]_p - \int_{t-\tau}^{t}[\tilde{x}'(s)]_p ds\right)^H W_p\left(\tau[\tilde{x}'(t)]_p - \int_{t-\tau}^{t}[\tilde{x}'(s)]_p ds\right)\right), \quad (6.7.12)
\end{aligned}
$$

$$
\begin{aligned}
\dot{V}_9(t) &= \sum_{p=0}^{3}\left(\tau^6 e^{2\varepsilon t}[\dot{\tilde{x}}'(t)]_p^H Z_p[\dot{\tilde{x}}'(t)]_p - 6\tau^3\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t} e^{2\varepsilon s}[\dot{\tilde{x}}'(s)]_p^H Z_p[\dot{\tilde{x}}'(s)]_p ds d\lambda d\theta\right)\\
&\leq \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^6[\dot{\tilde{x}}'(t)]_p^H Z_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad \left. -36e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t}[\dot{\tilde{x}}'(s)]_p ds d\lambda d\theta\right)^H Z_p\left(\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t}[\dot{\tilde{x}}'(s)]_p ds d\lambda d\theta\right)\right)\\
&= \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^6[\dot{\tilde{x}}'(t)]_p^H Z_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad \left. -36e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\int_{\theta}^{0}\left([\tilde{x}'(t)]_p - [\tilde{x}'(t+\lambda)]_p\right) d\lambda d\theta\right)^H Z_p\left(\int_{-\tau}^{0}\int_{\theta}^{0}\left([\tilde{x}'(t)]_p - [\tilde{x}'(t+\lambda)]_p\right) d\lambda d\theta\right)\right)\\
&= \sum_{p=0}^{3} e^{2\varepsilon t}\left(\tau^6[\dot{\tilde{x}}'(t)]_p^H Z_p[\dot{\tilde{x}}'(t)]_p\right.\\
&\quad \left. -9e^{-2\varepsilon\tau}\left(\tau^2[\tilde{x}'(t)]_p - 2\int_{-\tau}^{0}\int_{t+\theta}^{t}[\tilde{x}'(s)]_p ds\right)^H Z_p\left(\tau^2[\tilde{x}'(t)]_p - 2\int_{-\tau}^{0}\int_{t+\theta}^{t}[\tilde{x}'(s)]_p ds\right)\right), \quad (6.7.13)
\end{aligned}
$$

where we have used Assumption 6.5 to obtain the inequalities in (6.7.6) and (6.7.7), Lemma 6.4 to obtain the inequalities in (6.7.10) and (6.7.11), Lemma 6.5 to obtain the inequality in (6.7.12), and Lemma 6.6 to obtain the inequality in (6.7.13).

From Assumption 6.3 about the Lipschitz condition we can deduce that there exist positive definite diagonal matrices $G_p^1 = \operatorname{diag}\left(r_{j,p}^1\right)_{1\leq j\leq N}$ and $G_p^2 = \operatorname{diag}\left(r_{j,p}^2\right)_{1\leq j\leq N}$, such that

$$
0 \leq [\tilde{x}'(t)]_p^H \overline{L_{[\tilde{f}']_p}}^H G_p^1 \overline{L_{[\tilde{f}']_p}}[\tilde{x}'(t)]_p - [\tilde{f}']_p^H(\tilde{x}(t)) G_p^1 [\tilde{f}']_p(\tilde{x}(t)), \quad (6.7.14)
$$

$$
0 \leq [\tilde{x}'(t-\tau(t))]_p^H \overline{L_{[\tilde{f}']_p}}^H G_p^2 \overline{L_{[\tilde{f}']_p}}[\tilde{x}'(t-\tau(t))]_p - [\tilde{f}']_p^H(\tilde{x}(t-\tau(t))) G_p^2 [\tilde{f}']_p(\tilde{x}(t-\tau(t))), \quad (6.7.15)
$$

where $l_j^{[f']_p}$ are the Lipschitz constants of the functions f_j, $\forall j = \overline{1,N}$, and we denoted $\overline{L_{[\tilde{f}']_p}} = \text{diag}\left(l_j^{[f']_p}\right)_{1\leq j\leq N}$, $\forall p \in \{0,1,2,3\}$.

For any matrices N_p^1, N_p^2, N_p^3, N_p^4, $N_p^5 \in \mathbb{C}^{N\times N}$, we have that

$$\begin{aligned}
0 &= \left[[\dot{\tilde{x}}'(t)]_p^H N_p^1 + [\tilde{x}'(t)]_p^H N_p^2 - [\tilde{f}']_p^H(\tilde{x}(t))N_p^3 - [\tilde{f}']_p^H(\tilde{x}(t-\tau(t)))N_p^4 - [\dot{\tilde{x}}'(t-\eta(t))]_p^H N_p^5\right] \\
&\quad \times \left[-[\dot{\tilde{x}}'(t)]_p - D[\tilde{x}'(t)]_p + \overline{A}_p\tilde{f}(\tilde{y}(t)) + \overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}_p\dot{\tilde{y}}(t-\eta(t))\right] \\
&= -[\dot{\tilde{x}}'(t)]_p^H N_p^1[\dot{\tilde{x}}'(t)]_p - [\dot{\tilde{x}}'(t)]_p^H N_p^1 D[\tilde{x}'(t)]_p + [\dot{\tilde{x}}'(t)]_p^H N_p^1\overline{A}_p\tilde{f}(\tilde{y}(t)) + [\dot{\tilde{x}}'(t)]_p^H N_p^1\overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad +[\dot{\tilde{x}}'(t)]_p^H N_p^1\overline{C}_p\dot{\tilde{y}}(t-\eta(t)) - [\tilde{x}'(t)]_p^H N_p^2[\dot{\tilde{x}}'(t)]_p - [\tilde{x}'(t)]_p^H N_p^2 D[\tilde{x}'(t)]_p \\
&\quad +[\tilde{x}'(t)]_p^H N_p^2\overline{A}_p\tilde{f}(\tilde{y}(t)) + [\tilde{x}'(t)]_p^H N_p^2\overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) + [\tilde{x}'(t)]_p^H N_p^2\overline{C}_p\dot{\tilde{y}}(t-\eta(t)) \\
&\quad +[\tilde{f}']_p^H(\tilde{x}(t))N_p^3[\dot{\tilde{x}}'(t)]_p + [\tilde{f}']_p^H(\tilde{x}(t))N_p^3 D[\tilde{x}'(t)]_p - [\tilde{f}']_p^H(\tilde{x}(t))N_p^3\overline{A}_p\tilde{f}(\tilde{y}(t)) \\
&\quad -[\tilde{f}']_p^H(\tilde{x}(t))N_p^3\overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) - [\tilde{f}']_p^H(\tilde{x}(t))N_p^3\overline{C}_p\dot{\tilde{y}}(t-\eta(t)) \\
&\quad +[\tilde{f}']_p^H(\tilde{x}(t-\tau(t)))N_p^4[\dot{\tilde{x}}'(t)]_p + [\tilde{f}']_p^H(\tilde{x}(t-\tau(t)))N_p^4 D[\tilde{x}'(t)]_p - [\tilde{f}']_p^H(\tilde{x}(t-\tau(t)))N_p^4\overline{A}_p\tilde{f}(\tilde{y}(t)) \\
&\quad -[\tilde{f}']_p^H(\tilde{x}(t-\tau(t)))N_p^4\overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) - [\tilde{f}']_p^H(\tilde{x}(t-\tau(t)))N_p^4\overline{C}_p\dot{\tilde{y}}(t-\eta(t)) \\
&\quad +[\dot{\tilde{x}}'(t-\eta(t))]_p^H N_p^5[\dot{\tilde{x}}'(t)]_p + [\dot{\tilde{x}}'(t-\eta(t))]_p^H N_p^5 D[\tilde{x}'(t)]_p \\
&\quad -[\dot{\tilde{x}}'(t-\eta(t))]_p^H N_p^5\overline{A}_p\tilde{f}(\tilde{y}(t)) - [\dot{\tilde{x}}'(t-\eta(t))]_p^H N_p^5\overline{B}_p\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad -[\dot{\tilde{x}}'(t-\eta(t))]_p^H N_p^5\overline{C}_p\dot{\tilde{y}}(t-\eta(t)).
\end{aligned} \tag{6.7.16}$$

Now, taking the conjugate of (6.7.16), adding it to (6.7.16), and then multiplying the result by $e^{2\varepsilon t}$, then multiplying inequalities (6.7.14)–(6.7.15) by $e^{2\varepsilon t}$, and adding them all to (6.7.5)–(6.7.13), we obtain

$$\dot{V}(t) \leq \sum_{p=0}^{3}\zeta_p^H(t)\Pi_p\zeta_p(t),$$

where

$$\begin{aligned}
\zeta_p(t) &= \Big[\ [\tilde{x}'(t)]_p^H \quad [\dot{\tilde{x}}'(t)]_p^H \quad [\dot{\tilde{x}}'(t-\eta)]_p^H \quad \dot{\tilde{y}}(t-\eta(t)) \quad [\tilde{x}'(t-\tau)]_p^H \quad [\tilde{x}'(t-\tau(t))]_p^H \quad \tilde{f}^H(\tilde{y}(t)) \\
&\quad \tilde{f}^H(\tilde{y}(t-\tau(t))) \quad \left(\int_{t-\eta}^{t}[\dot{\tilde{x}}'(s)]_p ds\right)^H \quad \left(\int_{t-\tau}^{t}[\tilde{x}'(s)]_p ds\right)^H \quad \left(\int_{-\tau}^{0}\int_{t+\theta}^{t}[\tilde{x}'(s)]_p ds\right)^H\ \Big]^H,
\end{aligned}$$

and Π_p are defined by (6.7.4), $\forall p \in \{0,1,2,3\}$. From (6.7.4) we have that $\Pi_p < 0$, $\forall p \in \{0,1,2,3\}$, which implies that $\dot{V}(t) < 0$, meaning that $V(t)$ is strictly decreasing for $t \geq 0$. From the definition of $V(t)$, it can be further deduced that

$$\sum_{p=0}^{3} e^{2\varepsilon t}\lambda_{\min}(P_p)||[\tilde{x}'(t)]_p||^2 \leq \sum_{p=0}^{3} e^{2\varepsilon t}[\tilde{x}'(t)]_p^H P_p[\tilde{x}'(t)]_p \leq V(t) \leq V_0,\ \forall t \geq T,\ T \geq 0,$$

where $V_0 = \max\limits_{0\leq t\leq T} V(t)$. Thus,

$$\sum_{p=0}^{3}||[\tilde{x}'(t)]_p||^2 \leq \frac{V_0}{\min_{p\in\{0,1,2,3\}}\lambda_{\min}(P_p)}e^{-2\varepsilon t} \Leftrightarrow \sqrt{\sum_{p=0}^{3}||[\tilde{x}'(t)]_p||^2} \leq Me^{-\varepsilon t},\ \forall t \geq 0,$$

where $M = \sqrt{\frac{V_0}{\min_{p\in\{0,1,2,3\}}\lambda_{\min}(P_p)}}$, which means that the origin of system (6.7.2) is globally exponentially stable, completing the proof of the theorem. □

Now, we give an LMI-based sufficient condition for the exponential stability of the origin of (6.7.3).

Theorem 6.7. *If Assumptions 6.5 and 6.4 hold, then the origin of system* (6.7.3) *is globally exponentially stable if there exist positive definite matrices* P, Q^1, Q^2, M, R, S, T, U, W, $Z \in \mathbb{C}^{4N\times 4N}$, *any matrices* N^1, N^2, N^3, N^4, $N^5 \in \mathbb{C}^{4N\times 4N}$, *positive definite block-diagonal matrices* G^1, $G^2 \in \mathbb{R}^{4N\times 4N}$, *and a constant* $\varepsilon > 0$, *such that the following linear matrix inequality (LMI) holds*

$$(\Pi)_{11\times 11} < 0, \tag{6.7.17}$$

where $\Pi_{1,1} = 2\varepsilon P - \overline{D}P - P\overline{D} + Q^1 + R - e^{-2\varepsilon\tau}T + \eta^2 U - 4\tau^2 e^{-2\varepsilon\tau}W - 9\tau^4 e^{-2\varepsilon\tau}Z + \overline{L_{\tilde{f}}}^H G^1 \overline{L_{\tilde{f}}} - N^2\overline{D} - \overline{D}(N^2)^H$, $\Pi_{1,2} = -N^2 - \overline{D}(N^1)^H$, $\Pi_{1,4} = P\overline{C} + N^2\overline{C} + \overline{D}(N^5)^H$, $\Pi_{1,5} = e^{-2\varepsilon\tau}T$, $\Pi_{1,7} = P\overline{A} + N^2\overline{A} + \overline{D}(N^3)^H$, $\Pi_{1,8} = P\overline{B} + N^2\overline{B} + \overline{D}(N^4)^H$, $\Pi_{1,10} = 4\tau e^{-2\varepsilon\tau}W$, $\Pi_{1,11} = 18\tau^2 e^{-2\varepsilon\tau}Z$, $\Pi_{2,2} = M + S + \tau^2 T + \tau^4 W + \tau^6 Z - N^1 - (N^1)^H$, $\Pi_{2,4} = N^1\overline{C} + (N^5)^H$, $\Pi_{2,7} = N^1\overline{A} + (N^3)^H$, $\Pi_{2,8} = N^1\overline{B} + (N^4)^H$, $\Pi_{3,3} = -e^{-2\varepsilon\eta}S$, $\Pi_{4,4} = -(1-\eta')e^{-2\varepsilon\eta}M - N^5\overline{C} - \overline{C}^H(N^5)^H$, $\Pi_{4,7} = -N^5\overline{A} - \overline{C}^H(N^3)^H$, $\Pi_{4,8} = -N^5\overline{B} - \overline{C}^H(N^4)^H$, $\Pi_{5,5} = -e^{-2\varepsilon\tau}R - e^{-2\varepsilon\tau}T$, $\Pi_{6,6} = -(1-\tau')e^{-2\varepsilon\tau}Q^1 + \overline{L_{\tilde{f}}}^H G^2 \overline{L_{\tilde{f}}}$, $\Pi_{7,7} = Q^2 - G^1 - N^3\overline{A} - \overline{A}^H(N^3)^H$, $\Pi_{7,8} = -N^3\overline{B} - \overline{A}^H(N^4)^H$, $\Pi_{8,8} = -(1-\tau')e^{-2\varepsilon\tau}Q^2 - G^2 - N^4\overline{B} - \overline{B}^H(N^4)^H$, $\Pi_{9,9} = -e^{-2\varepsilon\eta}U$, $\Pi_{10,10} = -4e^{-2\varepsilon\tau}W$, $\Pi_{11,11} = -36e^{-2\varepsilon\tau}Z$.

Proof. Consider the Lyapunov–Krasovskii functional

$$V(t) = V_1(t) + V_2(t) + V_3(t) + V_4(t) + V_5(t) + V_6(t) + V_7(t) + V_8(t) + V_9(t),$$

$$V_1(t) = e^{2\varepsilon t}\tilde{y}^H(t)P\tilde{y}(t),$$

$$V_2(t) = \int_{t-\tau(t)}^{t} e^{2\varepsilon s}\xi^H(s)Q\xi(s)ds, \quad Q = \operatorname{diag}(Q^1, Q^2),$$

$$\xi(s) = \begin{bmatrix}[\tilde{y}(s)]^H & [\tilde{f}]^H(\tilde{y}(s))\end{bmatrix}^H,$$

$$V_3(t) = \int_{t-\eta(t)}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)M\dot{\tilde{y}}(s)ds,$$

$$V_4(t) = \int_{t-\tau}^{t} e^{2\varepsilon s}\tilde{y}^H(s)R\tilde{y}(s)ds,$$

$$V_5(t) = \int_{t-\eta}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)S\dot{\tilde{y}}(s)ds,$$

$$V_6(t) = \tau\int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)T\dot{\tilde{y}}(s)dsd\theta,$$

$$V_7(t) = \eta\int_{-\eta}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)U\dot{\tilde{y}}(s)dsd\theta,$$

$$V_8(t) = 2\tau^2\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)W\dot{\tilde{y}}(s)dsd\lambda d\theta,$$

$$V_9(t) = 6\tau^3\int_{-\tau}^{0}\int_{\theta}^{0}\int_{\lambda}^{0}\int_{t+\mu}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)Z\dot{\tilde{y}}(s)dsd\mu d\lambda d\theta.$$

The derivative of V along the trajectories of system (6.7.3) is

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t) + \dot{V}_4(t) + \dot{V}_5(t) + \dot{V}_6(t) + \dot{V}_7(t) + \dot{V}_8(t) + \dot{V}_9(t),$$

$$\begin{aligned}
\dot{V}_1(t) &= e^{2\varepsilon t}\left(2\varepsilon\tilde{y}^H(t)P\tilde{y}(t) + \dot{\tilde{y}}^H(t)P\tilde{y}(t) + \tilde{y}^H(t)P\dot{\tilde{y}}(t)\right) \\
&= e^{2\varepsilon t}\Big(2\varepsilon\tilde{y}^H(t)P\tilde{y}(t) + \Big(-\overline{D}\tilde{y}(t) + \overline{A}\tilde{f}(\tilde{y}(t)) + \overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad +\overline{C}\dot{\tilde{y}}(t-\eta(t))\Big)^H P\tilde{y}(t) + \tilde{y}^H(t)P\Big(-\overline{D}\tilde{y}(t) + \overline{A}\tilde{f}(\tilde{y}(t)) \\
&\quad +\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}\dot{\tilde{y}}(t-\eta(t))\Big)\Big), \qquad (6.7.18)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_2(t) &= e^{2\varepsilon t}\left(\xi^H(t)Q\xi(t) - (1-\dot{\tau}(t))e^{-2\varepsilon\tau(t)}\xi^H(t-\tau(t))Q\xi(t-\tau(t))\right) \\
&\leq e^{2\varepsilon t}\left(\xi^H(t)Q\xi(t) - (1-\tau')e^{-2\varepsilon\tau}\xi^H(t-\tau(t))Q\xi(t-\tau(t))\right), \qquad (6.7.19)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_3(t) &= e^{2\varepsilon t}\left(\dot{\tilde{y}}^H(t)M\dot{\tilde{y}}(t) - (1-\dot{\eta}(t))e^{-2\varepsilon\eta(t)}\dot{\tilde{y}}^H(t-\eta(t))M\dot{\tilde{y}}(t-\eta(t))\right) \\
&\leq e^{2\varepsilon t}\left(\dot{\tilde{y}}^H(t)M\dot{\tilde{y}}(t) - (1-\eta')e^{-2\varepsilon\eta}\dot{\tilde{y}}^H(t-\eta(t))M\dot{\tilde{y}}(t-\eta(t))\right), \qquad (6.7.20)
\end{aligned}$$

$$\dot{V}_4(t) = e^{2\varepsilon t}\left(\tilde{y}^H(t)R\tilde{y}(t) - e^{-2\varepsilon\tau}\tilde{y}^H(t-\tau)R\tilde{y}(t-\tau)\right), \qquad (6.7.21)$$

$$\dot{V}_5(t) = e^{2\varepsilon t}\left(\dot{\tilde{y}}^H(t)S\dot{\tilde{y}}(t) - e^{-2\varepsilon\eta}\dot{\tilde{y}}^H(t-\eta)S\dot{\tilde{y}}(t-\eta)\right), \qquad (6.7.22)$$

$$\begin{aligned}
\dot{V}_6(t) &= \tau^2 e^{2\varepsilon t}\dot{\tilde{y}}^H(t)T\dot{\tilde{y}}(t) - \tau\int_{t-\tau}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)T\dot{\tilde{y}}(s)ds \\
&\leq e^{2\varepsilon t}\left(\tau^2\dot{\tilde{y}}^H(t)T\dot{\tilde{y}}(t) - e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\dot{\tilde{y}}(s)ds\right)^H T\left(\int_{t-\tau}^{t}\dot{\tilde{y}}(s)ds\right)\right) \\
&= e^{2\varepsilon t}\left(\tau^2\dot{\tilde{y}}^H(t)T\dot{\tilde{y}}(t) - e^{-2\varepsilon\tau}\left(\tilde{y}(t)-\tilde{y}(t-\tau)\right)^H T\left(\tilde{y}(t)-\tilde{y}(t-\tau)\right)\right), \qquad (6.7.23)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_7(t) &= \eta^2 e^{2\varepsilon t}\dot{\tilde{y}}^H(t)U\dot{\tilde{y}}(t) - \eta\int_{t-\eta}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)U\dot{\tilde{y}}(s)ds \\
&\leq e^{2\varepsilon t}\left(\eta^2\dot{\tilde{y}}^H(t)U\dot{\tilde{y}}(t) - e^{-2\varepsilon\eta}\left(\int_{t-\eta}^{t}\dot{\tilde{y}}(s)ds\right)^H U\left(\int_{t-\eta}^{t}\dot{\tilde{y}}(s)ds\right)\right), \qquad (6.7.24)
\end{aligned}$$

$$\begin{aligned}
\dot{V}_8(t) &= \tau^4 e^{2\varepsilon t}\dot{\tilde{y}}^H(t)W\dot{\tilde{y}}(t) - 2\tau^2\int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)W\dot{\tilde{y}}(s)dsd\theta \\
&\leq e^{2\varepsilon t}\left(\tau^4\dot{\tilde{y}}^H(t)W\dot{\tilde{y}}(t) - 4e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\int_{t+\theta}^{t}\dot{\tilde{y}}(s)dsd\theta\right)^H W\left(\int_{-\tau}^{0}\int_{t+\theta}^{t}\dot{\tilde{y}}(s)dsd\theta\right)\right) \\
&= e^{2\varepsilon t}\left(\tau^4\dot{\tilde{y}}^H(t)W\dot{\tilde{y}}(t) - 4e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\left(\tilde{y}(t)-\tilde{y}(t+\theta)\right)d\theta\right)^H W\left(\int_{-\tau}^{0}\left(\tilde{y}(t)-\tilde{y}(t+\theta)\right)d\theta\right)\right) \\
&= e^{2\varepsilon t}\left(\tau^4\dot{\tilde{y}}^H(t)W\dot{\tilde{y}}(t) - 4e^{-2\varepsilon\tau}\left(\tau\tilde{y}(t) - \int_{t-\tau}^{t}\tilde{y}(s)ds\right)^H W\left(\tau\tilde{y}(t) - \int_{t-\tau}^{t}\tilde{y}(s)ds\right)\right), \qquad (6.7.25)
\end{aligned}$$

$$
\begin{aligned}
\dot{V}_9(t) &= \tau^6 e^{2\varepsilon t}\dot{\tilde{y}}^H(t)Z\dot{\tilde{y}}(t) - 6\tau^3\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t} e^{2\varepsilon s}\dot{\tilde{y}}^H(s)Z\dot{\tilde{y}}(s)dsd\lambda d\theta \\
&\leq e^{2\varepsilon t}\left(\tau^6\dot{\tilde{y}}^H(t)Z\dot{\tilde{y}}(t) - 36e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t}\dot{\tilde{y}}(s)dsd\lambda d\theta\right)^H Z\left(\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t}\dot{\tilde{y}}(s)dsd\lambda d\theta\right)\right) \\
&= e^{2\varepsilon t}\left(\tau^6\dot{\tilde{y}}^H(t)Z\dot{\tilde{y}}(t)\right. \\
&\quad \left. -36e^{-2\varepsilon\tau}\left(\int_{-\tau}^{0}\int_{\theta}^{0}\left(\tilde{y}(t)-\tilde{y}(t+\lambda)\right)d\lambda d\theta\right)^H Z\left(\int_{-\tau}^{0}\int_{\theta}^{0}\left(\tilde{y}(t)-\tilde{y}(t+\lambda)\right)d\lambda d\theta\right)\right) \\
&= e^{2\varepsilon t}\left(\tau^6\dot{\tilde{y}}^H(t)Z\dot{\tilde{y}}(t)\right. \\
&\quad \left. -9e^{-2\varepsilon\tau}\left(\tau^2\tilde{y}(t) - 2\int_{-\tau}^{0}\int_{t+\theta}^{t}\tilde{y}(s)ds\right)^H Z\left(\tau^2\tilde{y}(t) - 2\int_{-\tau}^{0}\int_{t+\theta}^{t}\tilde{y}(s)ds\right)\right), \qquad (6.7.26)
\end{aligned}
$$

where we have used Assumption 6.5 to obtain the inequalities in (6.7.19) and (6.7.20), Lemma 6.4 to obtain the inequalities in (6.7.23) and (6.7.24), Lemma 6.5 to obtain the inequality in (6.7.25), and Lemma 6.6 to obtain the inequality in (6.7.26).

From Assumption 6.4 about the Lipschitz condition, we can deduce that there exist positive definite block-diagonal matrices $G^1 = \text{diag}\left(r_j^1 I_4\right)_{1\leq j\leq N}$ and $G^2 = \text{diag}\left(r_j^2 I_4\right)_{1\leq j\leq N}$, such that

$$0 \leq \tilde{y}^H(t)\overline{L_{\tilde{f}}}^H G^1\overline{L_{\tilde{f}}}\tilde{y}(t) - \tilde{f}^H(\tilde{y}(t))G^1\tilde{f}(\tilde{y}(t)), \qquad (6.7.27)$$

$$0 \leq \tilde{y}^H(t-\tau(t))\overline{L_{\tilde{f}}}^H G^2\overline{L_{\tilde{f}}}\tilde{y}(t-\tau(t)) - \tilde{f}^H(\tilde{y}(t-\tau(t)))G^2\tilde{f}(\tilde{y}(t-\tau(t))), \qquad (6.7.28)$$

where l_j^f are the Lipschitz constants of the functions f_j, $\forall j = \overline{1,N}$, and we denoted $\overline{L_{\tilde{f}}} = \text{diag}\left(l_j^f I_4\right)_{1\leq j\leq N}$.

For any matrices N^1, N^2, N^3, N^4, $N^5 \in \mathbb{C}^{4N\times 4N}$, we have that

$$
\begin{aligned}
0 &= \left[\dot{\tilde{y}}^H(t)N^1 + \tilde{y}^H(t)N^2 - \tilde{f}^H(\tilde{y}(t))N^3 - \tilde{f}(\tilde{y}(t-\tau(t)))N^4 - \dot{\tilde{y}}^H(t-\eta(t))N^5\right] \\
&\quad \times\left[-\dot{\tilde{y}}(t) - \overline{D}\tilde{y}(t) + \overline{A}\tilde{f}(\tilde{y}(t)) + \overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) + \overline{C}\dot{\tilde{y}}(t-\eta(t))\right] \\
&= -\dot{\tilde{y}}^H(t)N^1\dot{\tilde{y}}(t) - \dot{\tilde{y}}^H(t)N^1\overline{D}\tilde{y}(t) + \dot{\tilde{y}}^H(t)N^1\overline{A}\tilde{f}(\tilde{y}(t)) + \dot{\tilde{y}}^H(t)N^1\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad +\dot{\tilde{y}}^H(t)N^1\overline{C}\dot{\tilde{y}}(t-\eta(t)) - \tilde{y}^H(t)N^2\dot{\tilde{y}}(t) - \tilde{y}^H(t)N^2\overline{D}\tilde{y}(t) \\
&\quad +\tilde{y}^H(t)N^2\overline{A}\tilde{f}(\tilde{y}(t)) + \tilde{y}^H(t)N^2\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) + \tilde{y}^H(t)N^2\overline{C}\dot{\tilde{y}}(t-\eta(t)) \\
&\quad +\tilde{f}^H(\tilde{y}(t))N^3\dot{\tilde{y}}(t) + \tilde{f}^H(\tilde{y}(t))N^3\overline{D}\tilde{y}(t) - \tilde{f}^H(\tilde{y}(t))N^3\overline{A}\tilde{f}(\tilde{y}(t)) - \tilde{f}^H(\tilde{y}(t))N^3\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad -\tilde{f}^H(\tilde{y}(t))N^3\overline{C}\dot{\tilde{y}}(t-\eta(t)) + \tilde{f}(\tilde{y}(t-\tau(t)))N^4\dot{\tilde{y}}(t) + \tilde{f}(\tilde{y}(t-\tau(t)))N^4\overline{D}\tilde{y}(t) \\
&\quad -\tilde{f}(\tilde{y}(t-\tau(t)))N^4\overline{A}\tilde{f}(\tilde{y}(t)) - \tilde{f}(\tilde{y}(t-\tau(t)))N^4\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) - \tilde{f}(\tilde{y}(t-\tau(t)))N^4\overline{C}\dot{\tilde{y}}(t-\eta(t)) \\
&\quad +\dot{\tilde{y}}^H(t-\eta(t))N^5\dot{\tilde{y}}(t) + \dot{\tilde{y}}^H(t-\eta(t))N^5\overline{D}\tilde{y}(t) \\
&\quad -\dot{\tilde{y}}^H(t-\eta(t))N^5\overline{A}\tilde{f}(\tilde{y}(t)) - \dot{\tilde{y}}^H(t-\eta(t))N^5\overline{B}\tilde{f}(\tilde{y}(t-\tau(t))) \\
&\quad -\dot{\tilde{y}}^H(t-\eta(t))N^5\dot{\tilde{y}}(t-\eta(t)). \qquad (6.7.29)
\end{aligned}
$$

Now, taking the conjugate of (6.7.29), adding it to (6.7.29), and then multiplying the result by $e^{2\varepsilon t}$, then multiplying inequalities (6.7.27)–(6.7.28) by $e^{2\varepsilon t}$, and adding them all to (6.7.18)–(6.7.26), we obtain

$$\dot{V}(t) \leq \zeta^H(t)\Pi\zeta(t),$$

where

$$
\begin{aligned}
\zeta(t) &= \left[\ \tilde{y}^H(t) \quad \dot{\tilde{y}}^H(t) \quad \dot{\tilde{y}}^H(t-\eta) \quad \dot{\tilde{y}}^H(t-\eta(t)) \quad \tilde{y}^H(t-\tau) \quad \tilde{y}^H(t-\tau(t)) \quad \tilde{f}^H(\tilde{y}(t))\right. \\
&\quad \left. \tilde{f}^H(\tilde{y}(t-\tau(t))) \quad \left(\int_{t-\eta}^{t}\dot{\tilde{y}}(s)ds\right)^H \quad \left(\int_{t-\tau}^{t}\tilde{y}(s)ds\right)^H \quad \left(\int_{-\tau}^{0}\int_{t+\theta}^{t}\tilde{y}(s)ds\right)^H\ \right]^H,
\end{aligned}
$$

and Π is defined by (6.7.17). From (6.7.17) we have that $\Pi < 0$, which implies that $\dot{V}(t) < 0$, meaning that $V(t)$ is strictly decreasing for $t \geq 0$. From the definition of $V(t)$ it can be further deduced that

$$e^{2\varepsilon t}\lambda_{\min}(P)||\tilde{y}(t)||^2 \leq e^{2\varepsilon t}\tilde{y}^H(t)P\tilde{y}(t) \leq V(t) \leq V_0,\ \forall t \geq T,\ T \geq 0,$$

where $V_0 = \max\limits_{0\leq t\leq T} V(t)$. Thus,

$$||\tilde{y}(t)||^2 \leq \frac{V_0}{\lambda_{\min}(P)}e^{-2\varepsilon t} \Leftrightarrow ||\tilde{y}(t)|| \leq Me^{-\varepsilon t},\ \forall t \geq 0,$$

where $M = \sqrt{\frac{V_0}{\lambda_{\min}(P)}}$, which means that the origin of system (6.7.3) is globally exponentially stable, completing the proof of the theorem. □

Remark 6.10. Taking into account Remark 6.4, it can be seen that Theorem 6.6 is not equivalent with Theorem 6.7. Because Assumption 6.3 implies Assumption 6.4, but not the other way around, we can deduce that Theorem 6.7 is more general than Theorem 6.6, and for any example for which Theorem 6.6 can be applied, Theorem 6.7 can be applied as well, but not the other way around.

Remark 6.11. The stability of complex-valued neutral-type neural networks was only discussed in [69] and [232]. The first paper discusses the exponential stability of the periodic solutions, whereas this paper treats global solutions. The second one gives delay-independent criteria for BAM neutral-type neural networks, which are more conservative than the delay-dependent criteria given in this paper. As such, a direct comparison of our results with the ones already existing in the literature is not possible. The particularization of the obtained results for complex-valued neural networks is another novel contribution of the paper. To the best of knowledge, there are no established stability criteria for quaternion-valued neutral-type neural networks.

6.7.2 Numerical examples

In this section, two numerical examples are provided to demonstrate the effectiveness and correctness of the proposed stability criteria.

Example 6.6. Consider the following two-neuron neutral-type OVNN with time-varying delays:

$$\begin{cases} \dot{x}_1(t) = & -d_1x_1(t) + \sum_{j=1}^{2} a_{1j}f_j(x_j(t)) + \sum_{j=1}^{2} b_{1j}f_j(x_j(t-\tau_{1j}(t))) \\ & +c_1\dot{x}_1(t-\eta_1(t)) + u_1, \\ \dot{x}_2(t) = & -d_2x_2(t) + \sum_{j=1}^{2} a_{2j}f_j(x_j(t)) + \sum_{j=1}^{2} b_{2j}f_j(x_j(t-\tau_{2j}(t))) \\ & +c_2\dot{x}_2(t-\eta_2(t)) + u_2, \end{cases} \tag{6.7.30}$$

where

$d_1 = 18$, $d_2 = 15$,
$a_{11} = (0.1 + 0.3\imath) + (0.2 + 0.2\imath)\jmath + (0.3 - 0.1\imath)\ell + (-0.4 + 0.1\imath)\jmath\ell$,
$a_{12} = (0.4 + 0.1\imath) + (0.3 - 0.2\imath)\jmath + (0.4 + 0.1\imath)\ell + (-0.2 + 0.3\imath)\jmath\ell$,
$a_{21} = (0.2 - 0.3\imath) + (0.2 + 0.1\imath)\jmath + (0.4 - 0.2\imath)\ell + (0.1 + 0.2\imath)\jmath\ell$,
$a_{22} = (0.1 + 0.2\imath) + (0.3 - 0.2\imath)\jmath + (0.2 + 0.1\imath)\ell + (0.4 - 0.2\imath)\jmath\ell$,
$b_{11} = (0.2 + 0.1\imath) + (0.3 + 0.1\imath)\jmath + (-0.2 + 0.3\imath)\ell + (-0.1 + 0.4\imath)\jmath\ell$,
$b_{12} = (-0.4 + 0.2\imath) + (-0.2 + 0.3\imath)\jmath + (0.1 + 0.2\imath)\ell + (-0.2 + 0.2\imath)\jmath\ell$,

$b_{21} = (0.1 - 0.4\imath) + (0.2 - 0.2\imath)\jmath + (0.1 + 0.2\imath)\ell + (0.3 + 0.2\imath)\jmath\ell$,
$b_{22} = (0.1 + 0.2\imath) + (0.4 + 0.1\imath)\jmath + (0.2 - 0.3\imath)\ell + (-0.2 + 0.4\imath)\jmath\ell$,
$c_1 = (0.1 + 0.05\imath) + (0.05 - 0.05\imath)\jmath + (0.1 - 0.1\imath)\ell + (-0.1 + 0.05\imath)\jmath\ell$,
$c_2 = (0.1 + 0.05\imath) + (0.05 - 0.05\imath)\jmath + (0.1 + 0.1\imath)\ell + (0.05 - 0.05\imath)\jmath\ell$,
$u_1 = 20((1 - 2\imath) + (3 - 4\imath)\jmath + (5 - 6\imath)\ell + (7 - 8\imath)\jmath\ell)$,
$u_2 = 20((8 - 7\imath) + (6 - 5\imath)\jmath + (4 - 3\imath)\ell + (2 - 1\imath)\jmath\ell)$,
$f_j\left(\sum_{q=0}^{7}[x]_q e_q\right) = \sum_{q=0}^{7} 0.1\frac{1-e^{-[x]_q}}{1+e^{-[x]_q}}e_q$, from which we infer that $l_j^{[f']_p} = \frac{0.1}{\sqrt{2}}$, $\forall p \in \{0,1,2,3\}$, $\forall j = \overline{1,2}$, and so Assumption 6.3 is fulfilled. If the time-varying delays are $\tau_{1j}(t) = \tau_{2j}(t) = 0.3|\sin t|$, $\forall j = \overline{1,2}$, and the neutral-type delays are $\eta_1(t) = \eta_2(t) = 0.1|\cos t|$, so $\tau = \tau' = 0.3$ and $\eta = \eta' = 0.1$, then Assumption 6.5 is also fulfilled.

By solving the LMI conditions in (6.7.4), we deduce from Theorem 6.6 that the equilibrium point of OVNN (6.7.30) is globally exponentially stable, for $G_0^1 = \text{diag}(8.1261, 10.1695)$, $G_0^2 = \text{diag}(7.4576, 9.5724)$, $G_1^1 = \text{diag}(8.1261, 10.1694)$, $G_1^2 = \text{diag}(7.4577, 9.5726)$, $G_2^1 = \text{diag}(8.1262, 10.1695)$, $G_2^2 = \text{diag}(7.4576, 9.5725)$, $G_3^1 = \text{diag}(8.1260, 10.1695)$, $G_3^2 = \text{diag}(7.4576, 9.5725)$, and $\varepsilon = 0.1$. (For brevity, the values of the other matrices are not given.)

Example 6.7. In this example, consider the same neutral-type OVNN with time-varying delays in (6.7.30), but with the following parameters:
$d_1 = 6$, $d_2 = 3$,
$a_{11} = (0.2 + 0.1\imath) + (0.5 - 0.2\imath)\jmath + (0.4 + 0.3\imath)\ell + (0.3 + 0.1\imath)\jmath\ell$,
$a_{12} = (-0.3 + 0.3\imath) + (-0.4 - 0.2\imath)\jmath + (0.3 + 0.1\imath)\ell + (0.2 - 0.1\imath)\jmath\ell$,
$a_{21} = (-0.4 + 0.3\imath) + (-0.3 - 0.2\imath)\jmath + (0.6 + 0.2\imath)\ell + (0.5 + 0.1\imath)\jmath\ell$,
$a_{22} = (-0.5 + 0.2\imath) + (-0.2 - 0.4\imath)\jmath + (0.2 + 0.3\imath)\ell + (0.2 - 0.3\imath)\jmath\ell$,
$b_{11} = (0.6 + 0.4\imath) + (0.9 - 0.5\imath)\jmath + (0.7 + 0.6\imath)\ell + (0.4 - 0.2\imath)\jmath\ell$,
$b_{12} = (0.7 + 0.6\imath) + (0.8 - 0.4\imath)\jmath + (0.9 + 0.4\imath)\ell + (0.3 - 0.2\imath)\jmath\ell$,
$b_{21} = (0.8 + 0.3\imath) + (0.7 - 0.2\imath)\jmath + (0.6 + 0.1\imath)\ell + (0.1 - 0.4\imath)\jmath\ell$,
$b_{22} = (-0.9 + 0.2\imath) + (-0.6 - 0.3\imath)\jmath + (0.8 + 0.5\imath)\ell + (0.2 - 0.4\imath)\jmath\ell$,
$c_1 = (-0.1 + 0.14\imath) + (-0.13 - 0.16\imath)\jmath + (0.12 + 0.15\imath)\ell + (0.15 - 0.18\imath)\jmath\ell$,
$c_2 = (0.13 + 0.12\imath) + (0.14 - 0.13\imath)\jmath + (0.11 + 0.12\imath)\ell + (0.13 - 0.12\imath)\jmath\ell$,
$u_1 = 5((8 - 7\imath) + (6 - 5\imath)\jmath + (4 - 3\imath)\ell + (2 - 1\imath)\jmath\ell)$,
$u_2 = 5((1 - 2\imath) + (3 - 4\imath)\jmath + (5 - 6\imath)\ell + (7 - 8\imath)\jmath\ell)$,
$f_j\left(\sum_{q=0}^{7}[x]_q e_q\right) = \sum_{q=0}^{7}\frac{0.1}{1+e^{-[x]_q}}e_q$, from which $l_j^f = \frac{0.1}{\sqrt{2}}$, $\forall j = \overline{1,2}$, and so Assumption 6.4 is fulfilled. If the time-varying delays are $\tau_{1j}(t) = \tau_{2j}(t) = 0.4|\cos t|$, $\forall j = \overline{1,2}$, and the neutral-type delays are $\eta_1(t) = \eta_2(t) = 0.2|\sin t|$, so $\tau = \tau' = 0.4$ and $\eta = \eta' = 0.2$, then Assumption 6.5 is also fulfilled.

By solving the LMI conditions in (6.7.17), we deduce from Theorem 6.7 that the equilibrium point of OVNN (6.7.30) with the above parameters is globally exponentially stable, for $G^1 = \text{diag}(3.3205, 2.8483)$, $G^2 = \text{diag}(3.3736, 3.4405)$, and $\varepsilon = 0.5$. (For brevity, the values of the other matrices are not given.)

Remark 6.12. It can be easily verified that Theorem 6.6 cannot be applied for Example 6.7, but Theorem 6.7 can be applied for both examples, which empirically confirms the correctness of the claims made in Remark 6.4 and Remark 6.10.

Chapter 7

Dynamics of matrix-valued neural networks (MVNNs)

Multidimensional neural networks have caught the attention of researchers over the last few years. Complex-valued neural networks (CVNNs), first introduced by [227], have numerous applications, including radar imaging, antenna design, image processing, estimation of direction of arrival and beamforming, communications signal processing, and many others [77, 78]. Hyperbolic numbers, which also form a 2-dimensional algebra, represent the basis for hyperbolic-valued neural networks (HVNNs), which are another type of multidimensional networks [103, 134]. First introduced by [4], quaternion-valued neural networks (QVNNs) were applied in color image compression [89], polarized signal classification [22], color night vision [104], chaotic time-series prediction [7], and 3D wind forecasting [92, 213]. Proposed by [139, 140], and later discussed by [24, 103], were Clifford-valued neural networks (ClVNNs), which have potential applications in high-dimensional data processing. These networks represent a generalization of CVNNs, HVNNs, and QVNNs, because complex, hyperbolic, and quaternion algebras are special cases of the 2^n-dimensional Clifford algebras, where $n \geq 1$. Lastly, octonion-valued neural networks (OVNNs), which don't fall into the Clifford neural networks category, were proposed by [149].

On the other hand, the complex, hyperbolic, quaternion, octonion, and Clifford algebras can all be seen as subalgebras of the square matrix algebra, which means that any number belonging to these sets can be represented in matrix form. For instance, the complex number $a + ib$, $i = \sqrt{-1}$, can be represented as $\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$, the hyperbolic number $a + ub$, $u^2 = 1$, $u \neq \pm 1$, as $\begin{pmatrix} a & b \\ b & a \end{pmatrix}$, and the quaternion $a + ib + jc + kd$, $i^2 = j^2 = k^2 = ijk = -1$, as $\begin{pmatrix} a & b & c & d \\ -b & a & -d & c \\ -c & d & a & -b \\ -d & -c & b & a \end{pmatrix}$. This observation means that the matrix algebra represents a generalization of all the above-mentioned algebras, which gave rise to the idea of generalizing CVNNs, HVNNs, QVNNs, OVNNs, and ClVNNs to matrix-valued neural networks (MVNNs), first in their feedforward variant, in [144]. Their degree of generality offers a potential for these neural networks to have many applications in the future at solving problems at which traditional neural networks have failed or performed poorly.

7.1 Matrix-valued Hopfield neural networks

The idea of introducing an energy function in order to study the dynamics of fully connected recurrent neural networks was first proposed by Hopfield at the beginning of the 1980's, see [80, 81, 82, 206]. He showed that combinatorial problems can be solved by using this type of network. Since then, Hopfield neural networks have been applied to the synthesis of associative memories, image processing, speech processing, control, signal processing, pattern matching, etc. Because of the observations made in the beginning of this chapter, we consider an interesting idea to introduce matrix-valued Hopfield neural networks. These networks can be applied to the synthesis of matrix-valued associative memories, and also to image processing and pattern matching, where the data can be treated in matrix form.

The presentation in this section follows that in the author's paper [152].

7.1.1 Main results

Consider the algebra $\mathcal{M}_n$ of square matrices of order n with real entries.

In what follows, we will define Hopfield neural networks for which the states, outputs, weights and thresholds are all from $\mathcal{M}_n$, which means that they are square matrices. The network is described by the set of differential equations

$$\tau_i \frac{dV_i(t)}{dt} = -V_i(t) + \sum_{j=1}^{N} W_{ij} f(V_j(t)) + B_i,\ i \in \{1, \ldots, N\}, \tag{7.1.1}$$

where $\tau_i \in \mathbb{R},\ \tau_i > 0$ is the time constant of neuron i, $V_i(t) \in \mathcal{M}_n$ is the state of neuron i at time t, $W_{ij} \in \mathcal{M}_n$ is the weight connecting neuron j to neuron i, $f : \mathcal{M}_n \to \mathcal{M}_n$ is the nonlinear matrix-valued activation function, and B_i is the threshold of neuron i, $\forall i \in \{1, \ldots, N\}$. The derivative is taken to be the matrix formed by the derivatives of each element $[V_i(t)]_{ab}$ of the matrix $V_i(t)$ with respect to t:

$$\frac{dV_i(t)}{dt} := \left(\frac{d([V_i(t)]_{ab})}{dt} \right)_{1 \le a,b \le n}.$$

If we denote by $X_j(t) := f(V_j(t))$ the output of neuron j, the above set of differential equations can be written as

$$\tau_i \frac{dV_i(t)}{dt} = -V_i(t) + \sum_{j=1}^{N} W_{ij} X_j(t) + B_i,\ i \in \{1, \ldots, N\}.$$

The activation function f is formed of n^2 functions $f^{ab} : \mathcal{M}_n \to \mathbb{R}$, $1 \le a, b \le n$:

$$f(V) = \left(f^{ab}(V) \right)_{1 \le a,b \le n}.$$

In order to study the stability of the above defined network, we need to make a series of assumptions about the activation function.

The first assumption is that the functions f^{ab} are continuously differentiable with respect to each $[V]_{cd}$, $\forall 1 \le c, d \le n$, $\forall 1 \le a, b \le n$, and the function f is bounded: $\exists M > 0$, $||f(V)|| \le M,\ \forall V \in \mathcal{M}_n$, where $||X||$ is the Frobenius norm of matrix X, defined by $||X|| = \sqrt{\text{Tr}(XX^T)}$, and $\text{Tr}(X)$ represents the trace of matrix X. In this setting, the $n^2 \times n^2$ Jacobian matrix of the function f can be defined as

$$\mathbf{Jac}_f(V) = \left(\frac{\partial f^{ab}(V)}{\partial [V]_{cd}}\right)_{\substack{1\leq a,b\leq n\\ 1\leq c,d\leq n}}.$$

The second assumption that we have to make is that f is injective and $\mathbf{Jac}_f(V)$ is symmetric and positive definite, $\forall V \in \mathcal{M}_n$. This, together with the above assumption, assures the existence of the inverse function of f, $g : \mathcal{M}_n \to \mathcal{M}_n$, $g = f^{-1}$. We can thus write $g(X_i(t)) = V_i(t)$, $\forall i \in \{1, \ldots, N\}$. Now, we can define a function $G : \mathcal{M}_n \to \mathbb{R}$,

$$G(X) = \sum_{a,b=1}^{n} \int_0^{[X]_{ab}} g^{ab}(Y^{ab})dy,$$

where $g^{ab} : \mathcal{M}_n \to \mathbb{R}$ are the component functions of g and the matrices Y^{ab} have the following form

$$[Y^{ab}]_{cd} = \begin{cases} [X]_{cd}, & (c,d) < (a,b) \\ y, & (c,d) = (a,b) \\ 0 & (c,d) > (a,b) \end{cases}, \ \forall 1 \leq a, b \leq n.$$

For example, for 2×2 matrices, we have that

$$\begin{aligned} G(X) \ = \ & \int_0^{[X]_{11}} g^{11}\left(\begin{pmatrix} y & 0 \\ 0 & 0 \end{pmatrix}\right) dy + \int_0^{[X]_{12}} g^{12}\left(\begin{pmatrix} [X]_{11} & y \\ 0 & 0 \end{pmatrix}\right) dy \\ & + \int_0^{[X]_{21}} g^{21}\left(\begin{pmatrix} [X]_{11} & [X]_{12} \\ y & 0 \end{pmatrix}\right) dy + \int_0^{[X]_{22}} g^{22}\left(\begin{pmatrix} [X]_{11} & [X]_{12} \\ [X]_{21} & y \end{pmatrix}\right) dy. \end{aligned}$$

This function satisfies

$$\frac{\partial G(X)}{\partial [X]_{ab}} = g^{ab}(X), \ \forall 1 \leq a, b \leq n.$$

The above condition can also be written in matrix form as

$$\frac{\partial G(X)}{\partial X} = g(X). \tag{7.1.2}$$

The last assumption concerns the weights of the network, which must satisfy:

$$W_{ji} = W_{ij}^T, \ \forall i, j \in \{1, \ldots, N\}.$$

Having made all the above assumptions, we can define the energy function $E : \mathcal{M}_n^N \to \mathbb{R}$ of the Hopfield network (7.1.1) as:

$$E(\mathbf{X}(t)) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \mathrm{Tr}(X_i(t)^T W_{ij} X_j(t)) + \sum_{i=1}^{N} G(X_i(t)) - \sum_{i=1}^{N} \mathrm{Tr}(B_i^T X_i(t)). \tag{7.1.3}$$

A function E is an energy function for the Hopfield network (7.1.1) if the derivative of E along the trajectories of network, denoted by $\frac{dE(\mathbf{X}(t))}{dt}$, satisfies the condition $\frac{dE(\mathbf{X}(t))}{dt} \leq 0$ and $\frac{dE(\mathbf{X}(t))}{dt} = 0 \Leftrightarrow \frac{dX_i(t)}{dt} = 0, \ \forall i \in \{1, \ldots, N\}$. We will show that the function E defined in (7.1.3) is indeed an energy function for the network (7.1.1).

For this, we start by applying the chain rule:

$$\begin{aligned}\frac{dE(\mathbf{X}(t))}{dt} &= \sum_{i=1}^{N}\sum_{a,b=1}^{n}\frac{\partial E(\mathbf{X}(t))}{\partial [X_i(t)]_{ab}}\frac{d[X_i(t)]_{ab}}{dt}\\ &= \sum_{i=1}^{N}\mathrm{Tr}\left(\left(\frac{\partial E(\mathbf{X}(t))}{\partial X_i(t)}\right)^T\frac{dX_i(t)}{dt}\right),\end{aligned} \tag{7.1.4}$$

where by $\frac{\partial E(\mathbf{X}(t))}{\partial [X_i(t)]_{ab}}$ we denoted the partial derivative of the function E with respect to each element $[X_i(t)]_{ab}$ of the matrix $X_i(t)$, $\forall 1 \leq a, b \leq n$, $\forall i \in \{1, \ldots, N\}$. Taking (7.1.3) into account, using the fact that

$$\frac{d\mathrm{Tr}(X^TA)}{dX} = \frac{d\mathrm{Tr}(A^TX)}{dX} = A,$$

relation (7.1.2), the assumption $W_{ji} = W_{ij}^T$, and also the set of equations given by (7.1.1), the expression of the partial derivative $\frac{\partial E(\mathbf{X}(t))}{\partial X_i(t)} = \left(\frac{\partial E(\mathbf{X}(t))}{\partial [X_i(t)]_{ab}}\right)_{1\leq a,b\leq n}$ is computed as:

$$\begin{aligned}\frac{\partial E(\mathbf{X}(t))}{\partial X_i(t)} &= -\sum_{j=1}^{N}W_{ij}X_j(t) + g(X_i(t)) - B_i\\ &= -\left(\sum_{j=1}^{N}W_{ij}X_j(t) - V_i(t) + B_i\right)\\ &= -\tau_i\frac{dV_i(t)}{dt},\ \forall i \in \{1, \ldots, N\},\end{aligned}$$

If we denote by $\mathrm{vec}\,(X)$ the vectorization of matrix X, and use the identity $\mathrm{Tr}(A^TB) = \mathrm{vec}(A)^T\mathrm{vec}(B),\ \forall A, B \in \mathcal{M}_n$, we can now write equation (7.1.4) as:

$$\begin{aligned}\frac{dE(\mathbf{X}(t))}{dt} &= \sum_{i=1}^{N}\mathrm{Tr}\left(\left(-\tau_i\frac{dV_i(t)}{dt}\right)^T\frac{dX_i(t)}{dt}\right)\\ &= -\sum_{i=1}^{N}\tau_i\left[\mathrm{vec}\left(\frac{dV_i(t)}{dt}\right)\right]^T\mathrm{vec}\left(\frac{dX_i(t)}{dt}\right)\\ &= -\sum_{i=1}^{N}\tau_i\left[\mathrm{vec}\left(\frac{dX_i(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(X_i(t))]^T\mathrm{vec}\left(\frac{dX_i(t)}{dt}\right)\\ &\leq 0,\end{aligned} \tag{7.1.5}$$

where, from $g(X_i(t)) = V_i(t)$, we obtained that

$$\mathrm{vec}\left(\frac{dg(X_i(t))}{dt}\right) = \mathbf{Jac}_g(X_i(t))\mathrm{vec}\left(\frac{dX_i(t)}{dt}\right),\ \forall i \in \{1, \ldots, N\}.$$

Because $\mathbf{Jac}_f(V)$ is symmetric and positive definite, we deduce that $\mathbf{Jac}_g(X)$ is also symmetric and positive definite, and thus

$$\left[\mathrm{vec}\left(\frac{dX_i(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(X_i(t))]^T\mathrm{vec}\left(\frac{dX_i(t)}{dt}\right) \geq 0,\ \forall i \in \{1, \ldots, N\},$$

which allowed us to write the last inequality in relation (7.1.5). Equality is attained when

$$\frac{dE(\mathbf{X}(t))}{dt} = 0 \Leftrightarrow \text{vec}\left(\frac{dX_i(t)}{dt}\right) = 0 \Leftrightarrow \frac{dX_i(t)}{dt} = 0,\ \forall i \in \{1, \ldots, N\},$$

thus ending the proof that E is indeed an energy function for the network (7.1.1).

We now give two examples of activation functions that satisfy the above assumptions, inspired by the ones used in real-valued and complex-valued neural networks:

$$f(V) = \frac{V}{1 + ||V||},\ \forall V \in \mathcal{M}_n,$$

$$f\left(([V]_{ab})_{1\leq a,b\leq n}\right) = (\tanh[V]_{ab})_{1\leq a,b\leq n},\ \forall V \in \mathcal{M}_n.$$

The first one corresponds to the fully complex activation functions, while the second one corresponds to the split complex activation functions from the complex-valued domain.

7.2 Matrix-valued bidirectional associative memories

First introduced by Kosko in [100], bidirectional associative memories, an extension of the unidirectional Hopfield neural networks, were intensely studied, and have many applications in pattern recognition and automatic control. Because of the fact that complex-valued bidirectional associative memories were introduced in [110], quaternion-valued bidirectional associative memories in [102], and Clifford-valued bidirectional associative memories in [217], we considered an interesting idea to introduce matrix-valued bidirectional associative memories. These networks can be applied to store matrix patterns and to solve difficult matrix optimization problems.

The presentation in this section follows that in the author's paper [172].

7.2.1 Main results

In the following, we will define bidirectional associative memories for which the states, outputs, and thresholds are all from $\mathcal{M}_n$, which means that they are square matrices. The network is described by the set of differential equations

$$\begin{cases} \tau_i \dfrac{dX_i(t)}{dt} = -X_i(t) + \displaystyle\sum_{j=1}^{P} W_{ij} f(Y_j(t)) + A_i, \quad \forall i \in \{1, \ldots, N\}, \\ \upsilon_j \dfrac{dY_j(t)}{dt} = -Y_j(t) + \displaystyle\sum_{i=1}^{N} W_{ji} f(X_i(t)) + B_j, \quad \forall j \in \{1, \ldots, P\}, \end{cases} \tag{7.2.1}$$

where $\tau_i \in \mathbb{R},\ \tau_i > 0$ is the time constant of neuron X_i, $\upsilon_j \in \mathbb{R},\ \upsilon_j > 0$ is the time constant of neuron Y_j, $X_i(t) \in \mathcal{M}_n$ is the state of neuron X_i at time t, $Y_j(t) \in \mathcal{M}_n$ is the state of neuron Y_j at time t, $W_{ij} \in \mathcal{M}_n$ is the weight connecting neuron X_i to neuron Y_j, $f : \mathcal{M}_n \to \mathcal{M}_n$ is the nonlinear matrix-valued activation function, A_i is the threshold of neuron X_i, and B_j is the threshold of neuron Y_j, $\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, P\}$. By denoting $U_i(t) := f(X_i(t))$ the output of neuron X_i and $V_j(t) := f(Y_j(t))$ the output of neuron Y_j, the above set of differential

equations becomes:

$$\begin{cases} \tau_i \dfrac{dX_i(t)}{dt} = -X_i(t) + \displaystyle\sum_{j=1}^{P} W_{ij} V_j(t) + A_i, \ \forall i \in \{1, \dots, N\}, \\ \upsilon_j \dfrac{dY_j(t)}{dt} = -Y_j(t) + \displaystyle\sum_{i=1}^{N} W_{ji} U_i(t) + B_j, \ \forall j \in \{1, \dots, P\}. \end{cases}$$

Studying the stability of the above defined network requires a series of assumptions about the activation function, which we will detail below.

Just like in Section 7.1, in order to study the stability of the above defined network, we need to make a series of assumptions about the activation function. The assumptions will be exactly the same as the ones in Section 7.1.

Taking these assumptions into account, we can define the energy function $E : \mathcal{M}_n^{N+P} \to \mathbb{R}$ of the bidirectional associative memory (7.2.1) as:

$$\begin{aligned} E(\mathbf{U}(t), \mathbf{V}(t)) \ &= \ -\sum_{i=1}^{N}\sum_{j=1}^{P} \mathrm{Tr}(U_i(t)^T W_{ij} V_j(t)) \\ &\quad + \sum_{i=1}^{N} G(U_i(t)) - \sum_{i=1}^{N} \mathrm{Tr}(A_i^T U_i(t)) \\ &\quad + \sum_{j=1}^{P} G(V_j(t)) - \sum_{j=1}^{P} \mathrm{Tr}(B_j^T V_j(t)). \end{aligned} \tag{7.2.2}$$

A function E is an energy function for the network (7.2.1) if the derivative of E along the trajectories of network, denoted by $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt}$, satisfies the condition $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} \leq 0$ and $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} = 0 \Leftrightarrow \frac{dU_i(t)}{dt} = \frac{dV_j(t)}{dt} = 0, \forall i \in \{1, \dots, N\}, \forall j \in \{1, \dots, P\}$. In the following, we will show that the function E defined in (7.2.2) is indeed an energy function for the network (7.2.1).

For this, we start by applying the chain rule:

$$\begin{aligned} \frac{dE(\mathbf{U}(t), \mathbf{V}(t))}{dt} \ &= \ \sum_{i=1}^{N}\sum_{a,b=1}^{n} \frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial [U_i(t)]_{ab}} \frac{d[U_i(t)]_{ab}}{dt} \\ &\quad + \sum_{j=1}^{P}\sum_{a,b=1}^{n} \frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial [V_j(t)]_{ab}} \frac{d[V_j(t)]_{ab}}{dt} \\ &= \ \sum_{i=1}^{N} \mathrm{Tr}\left(\left(\frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial U_i(t)} \right)^T \frac{dU_i(t)}{dt} \right) \\ &\quad + \sum_{j=1}^{P} \mathrm{Tr}\left(\left(\frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial V_j(t)} \right)^T \frac{dV_j(t)}{dt} \right), \end{aligned} \tag{7.2.3}$$

where by $\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [U_i(t)]_{ab}}$ we denoted the partial derivative of the function E with respect to each element $[U_i(t)]_{ab}$ of the matrices $U_i(t)$, $\forall 1 \leq a, b \leq n$, $\forall i \in \{1, \dots, N\}$, and analogously for $\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [V_j(t)]_{ab}}$, $\forall j \in \{1, \dots, P\}$.

For the partial derivatives

$$\frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial U_i(t)} = \left(\frac{\partial E(\mathbf{U}(t), \mathbf{V}(t))}{\partial [U_i(t)]_{ab}} \right)_{1 \leq a,b \leq n}$$

and

$$\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial V_j(t)} = \left(\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [V_j(t)]_{ab}}\right)_{1\leq a,b\leq n},$$

we have from (7.2.2) that

$$\begin{aligned}
\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial U_i(t)} &= -\sum_{j=1}^{P} W_{ij}V_j(t) + g(U_i(t)) - A_i \\
&= -\left(\sum_{j=1}^{P} W_{ij}V_j(t) - X_i(t) + A_i\right) \\
&= -\tau_i \frac{dX_i(t)}{dt},\ \forall i \in \{1,\ldots,N\},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial V_j(t)} &= -\sum_{i=1}^{N} W_{ji}U_i(t) + g(V_j(t)) - B_j \\
&= -\left(\sum_{i=1}^{N} W_{ji}U_j(t) - Y_j(t) + B_j\right) \\
&= -\upsilon_j \frac{dY_j(t)}{dt},\ \forall j \in \{1,\ldots,P\},
\end{aligned}$$

where we used the fact that

$$\frac{d\mathrm{Tr}(X^T A)}{dX} = \frac{d\mathrm{Tr}(A^T X)}{dX} = A,$$

relation (7.1.2), the assumption $W_{ji} = W_{ij}^T$, and also the set of equations given by (7.2.1). Now, equation (7.2.3) becomes:

$$\begin{aligned}
\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} &= \sum_{i=1}^{N} \mathrm{Tr}\left(\left(-\tau_i \frac{dX_i(t)}{dt}\right)^T \frac{dU_i(t)}{dt}\right) \\
&\quad + \sum_{j=1}^{P} \mathrm{Tr}\left(\left(-\upsilon_j \frac{dY_j(t)}{dt}\right)^T \frac{dV_j(t)}{dt}\right) \\
&= -\sum_{i=1}^{N} \tau_i \left[\mathrm{vec}\left(\frac{dX_i(t)}{dt}\right)\right]^T \mathrm{vec}\left(\frac{dU_i(t)}{dt}\right) \\
&\quad - \sum_{j=1}^{P} \upsilon_j \left[\mathrm{vec}\left(\frac{dY_j(t)}{dt}\right)\right]^T \mathrm{vec}\left(\frac{dV_j(t)}{dt}\right) \\
&= -\sum_{i=1}^{N} \left\{\tau_i \left[\mathrm{vec}\left(\frac{dU_i(t)}{dt}\right)\right]^T [\mathbf{Jac}_g(U_i(t))]^T \mathrm{vec}\left(\frac{dU_i(t)}{dt}\right)\right\} \\
&\quad - \sum_{j=1}^{P} \left\{\upsilon_j \left[\mathrm{vec}\left(\frac{dV_j(t)}{dt}\right)\right]^T [\mathbf{Jac}_g(V_j(t))]^T \mathrm{vec}\left(\frac{dV_j(t)}{dt}\right)\right\} \\
&\leq 0, \qquad (7.2.4)
\end{aligned}$$

where we denoted by vec (X) the vectorization of matrix X. We also used the identity

$$\text{Tr}(A^T B) = \text{vec}(A)^T \text{vec}(B),\ \forall A, B \in \mathcal{M}_n,$$

and, from $g(U_i(t)) = X_i(t)$ and $g(V_j(t)) = Y_j(t)$, we obtained that

$$\text{vec}\left(\frac{dg(U_i(t))}{dt}\right) = \textbf{Jac}_g(U_i(t))\text{vec}\left(\frac{dU_i(t)}{dt}\right),$$

$$\text{vec}\left(\frac{dg(V_j(t))}{dt}\right) = \textbf{Jac}_g(V_j(t))\text{vec}\left(\frac{dV_j(t)}{dt}\right),$$

$\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, P\}$. Because $\textbf{Jac}_f(X)$ is symmetric and positive definite, we deduce that $\textbf{Jac}_g(U)$ is also symmetric and positive definite, and thus

$$\left[\text{vec}\left(\frac{dU_i(t)}{dt}\right)\right]^T [\textbf{Jac}_g(U_i(t))]^T \text{vec}\left(\frac{dU_i(t)}{dt}\right) \geq 0,$$

$$\left[\text{vec}\left(\frac{dV_j(t)}{dt}\right)\right]^T [\textbf{Jac}_g(V_j(t))]^T \text{vec}\left(\frac{dV_j(t)}{dt}\right) \geq 0,$$

$\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, P\}$, which allowed us to write the last inequality in relation (7.2.4). Equality is attained when $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} = 0 \Leftrightarrow \text{vec}\left(\frac{dU_i(t)}{dt}\right) = \text{vec}\left(\frac{dV_j(t)}{dt}\right) = 0 \Leftrightarrow \frac{dU_i(t)}{dt} = \frac{dV_j(t)}{dt} = 0,\ \forall i \in \{1, \ldots, N\},\ \forall j \in \{1, \ldots, P\}$, thus ending the proof that E is indeed an energy function for the network (7.2.1).

7.3 Asymptotic stability for MVNNs with delay

We considered an interesting idea to introduce matrix-valued Hopfield neural networks [152], which, as stated earlier, generalize CVNNs, HVNNs, QVNNs, OVNNs, and ClVNNs. These networks can be applied to the synthesis of matrix-valued associative memories, and also to image processing and pattern matching, where the data can be treated in matrix form. Because time delays inherently occur in real life implementations of neural networks, and they can lead to unwanted behavior such as oscillations and chaos, we consider matrix-valued Hopfield neural networks with time delay, and study their dynamics, more precisely the existence, uniqueness, and global asymptotic stability of their equilibrium point.

In this and the following sections, the following notations will be used: with $\mathbb{R}$ we denote the real number set and with $\mathbb{R}^n$ we denote the n dimensional Euclidean space. $\mathcal{M}_n$ denotes the algebra of real square matrices of order n. A^T represents the transpose of matrix A and $*$ denotes the symmetric terms in a matrix. $A > 0$ ($A < 0$) means that matrix A is positive definite (negative definite). I_n represents the identity matrix of order n and 0_n the empty matrix of order n. The vector Euclidean norm or the matrix Frobenius norm are denoted by $||\cdot||$. $\lambda_{\text{min}}(P)$ represents the smallest eigenvalue of matrix P.

The presentation in this section follows that in the author's paper [160].

7.3.1 Main results

In what follows, we will define Hopfield neural networks for which the states, outputs, and weights are all from $\mathcal{M}_n$, which means that they are square matrices. The network is described

by the system of differential equations

$$\dot{X}_i(t) = -d_i X_i(t) + \sum_{j=1}^{N} A_{ij} f_j(X_j(t)) + \sum_{j=1}^{N} B_{ij} g_j(X_j(t-\tau)) + U_i, \qquad (7.3.1)$$

$i \in \{1, \ldots, N\}$, where $X_i(t) \in \mathcal{M}_n$ is the state of neuron i at time t, $d_i \in \mathbb{R}$, $d_i > 0$, is the self-feedback connection weight of neuron i, $A_{ij} \in \mathcal{M}_n$ is the weight connecting neuron j to neuron i without time delay, $B_{ij} \in \mathcal{M}_n$ is the weight connecting neuron j to neuron i with time delay, $f_j : \mathcal{M}_n \to \mathcal{M}_n$ is the nonlinear matrix-valued activation function of neuron j without time delay, $g_j : \mathcal{M}_n \to \mathcal{M}_n$ is the nonlinear matrix-valued activation function of neuron j with time delay, $\tau \in \mathbb{R}$ is the time delay and we assume $\tau > 0$, and U_i is the external input of neuron i, $\forall i, j \in \{1, \ldots, N\}$.

In order to study the stability of the above defined network, we need to make an assumption about the activation functions.

Assumption 7.1. *The matrix-valued activation functions f_j and g_j satisfy the following Lipschitz conditions:*

$$||f_j(X) - f_j(X')|| \leq l_j^f ||X - X'||, \ \forall X, X' \in \mathcal{M}_n,$$

$$||g_j(X) - g_j(X')|| \leq l_j^g ||X - X'||, \ \forall X, X' \in \mathcal{M}_n,$$

where $l_j^f > 0$ and $l_j^g > 0$ are the Lipschitz constants, $\forall j \in \{1, \ldots, N\}$. Furthermore, we denote $\overline{L_f} = diag(l_1^f I_{n^2}, l_2^f I_{n^2}, \ldots, l_N^f I_{n^2})$, $\overline{L_g} = diag(l_1^g I_{n^2}, l_2^g I_{n^2}, \ldots, l_N^g I_{n^2})$.

We will first transform the matrix-valued set of differential equations (7.3.1) into a real-valued one. For this, we will detail each equation in (7.3.1) into n^2 real-valued equations:

$$\begin{aligned}[\dot{X}_i(t)]_{pq} &= -d_i [X_i(t)]_{pq} + \sum_{j=1}^{N} \sum_{r=1}^{n} [A_{ij}]_{pr} f_j^{rq}(X_j(t)) \\ &\quad + \sum_{j=1}^{N} \sum_{r=1}^{n} [B_{ij}]_{pr} g_j^{rq}(X_j(t-\tau)) + [U_i]_{pq}, \end{aligned} \qquad (7.3.2)$$

$1 \leq p, q \leq n$, $i \in \{1, \ldots, N\}$. Now, using the vectorization operation, we can write

$$\begin{aligned}\text{vec}(\dot{X}_i(t)) &= -d_i I_{n^2} \text{vec}(X_i(t)) + \sum_{j=1}^{N} (I_n \otimes A_{ij}) \text{vec}(f_j(X_j(t))) \\ &\quad + \sum_{j=1}^{N} (I_n \otimes B_{ij}) \text{vec}(g_j(X_j(t-\tau))) + \text{vec}(U_i), \end{aligned} \qquad (7.3.3)$$

$i \in \{1, \ldots, N\}$. Finally, if we denote

$$W(t) = (\text{vec}(X_1(t))^T, \text{vec}(X_2(t))^T, \ldots, \text{vec}(X_N(t))^T)^T,$$

$$\overline{D} = \begin{bmatrix} d_1 I_{n^2} & 0 & \cdots & 0 \\ 0 & d_2 I_{n^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & d_N I_{n^2} \end{bmatrix},$$

$$\overline{A} = \begin{bmatrix} I_n \otimes A_{11} & I_n \otimes A_{12} & \cdots & I_n \otimes A_{1N} \\ I_n \otimes A_{21} & I_n \otimes A_{22} & \cdots & I_n \otimes A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ I_n \otimes A_{N1} & I_n \otimes A_{N2} & \cdots & I_n \otimes A_{NN} \end{bmatrix},$$

$$\overline{B} = \begin{bmatrix} I_n \otimes B_{11} & I_n \otimes B_{12} & \cdots & I_n \otimes B_{1N} \\ I_n \otimes B_{21} & I_n \otimes B_{22} & \cdots & I_n \otimes B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ I_n \otimes B_{N1} & I_n \otimes B_{N2} & \cdots & I_n \otimes B_{NN} \end{bmatrix},$$

$$\overline{f}(W(t)) = (\text{vec}(f_1(X_1(t)))^T, \text{vec}(f_2(X_2(t)))^T, \ldots, \text{vec}(f_N(X_N(t)))^T)^T,$$

$$\overline{g}(W(t-\tau)) = (\text{vec}(g_1(X_1(t-\tau)))^T, \text{vec}(g_2(X_2(t-\tau)))^T, \ldots, \text{vec}(g_N(X_N(t-\tau)))^T)^T,$$

$$\overline{U} = (\text{vec}(U_1)^T, \text{vec}(U_2)^T, \ldots, \text{vec}(U_N)^T)^T,$$

with the simplifying notations $W = W(t)$ and $W^\tau = W(t-\tau)$, system (7.3.1) becomes

$$\dot{W} = -\overline{D}W + \overline{A}\,\overline{f}(W) + \overline{B}\,\overline{g}(W^\tau) + \overline{U}. \tag{7.3.4}$$

Remark 7.1. The system (7.3.4) is equivalent with the system (7.3.1), which means that any property proven about system (7.3.4) will also hold for system (7.3.1). For this reason, from now on we will only study the existence, uniqueness and global asymptotic stability of the equilibrium point of system (7.3.4).

Remark 7.2. It can be clearly seen from the above derivation that the matrix-valued recurrent neural network defined in (7.3.1) is not equivalent with an nN-dimensional real-valued recurrent neural network, because, for such a network, the matrices $\overline{A}$ and $\overline{B}$ are general unconstrained matrices, and don't have the particular form given above.

We will also need the following lemmas:

Lemma 7.1. *([55]). If $H : \mathbb{R}^{n^2N} \to \mathbb{R}^{n^2N}$ is a continuous map that satisfies the following conditions:*

(i) H is injective on $\mathbb{R}^{n^2N}$,

(ii) $||H(W)|| \to \infty$ as $||W|| \to \infty$, where $||\cdot||$ represents the Euclidean norm on $\mathbb{R}^{n^2N}$,

then H is a homeomorphism of $\mathbb{R}^{n^2N}$ onto itself.

Lemma 7.2. *([114]). For any vectors x, $y \in \mathbb{R}^{n^2N}$, positive definite matrix $P \in \mathcal{M}_{n^2N}$, and real constant $\varepsilon > 0$, the following linear matrix inequality (LMI) holds:*

$$2x^Ty \leq \varepsilon x^TPx + \frac{1}{\varepsilon}y^TP^{-1}y.$$

Lemma 7.3. *([66]). For any positive definite matrix $M \in \mathcal{M}_{n^2N}$ and vector function $W : [a,b] \to \mathbb{R}^{n^2N}$, such that the integrations concerned are well defined, the following inequality holds:*

$$\left(\int_a^b W(s)ds\right)^T M \left(\int_a^b W(s)ds\right) \leq (b-a)\int_a^b W^T(s)MW(s)ds.$$

7.3.2 Main results

We begin by giving an LMI-based sufficient condition for the existence, uniqueness, and global asymptotic stability of the equilibrium point for (7.3.4).

Theorem 7.1. *If Assumption 7.1 holds, then system* (7.3.4) *has a unique equilibrium point which is globally asymptotically stable if there exist real numbers $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$, and positive definite matrix $P \in \mathcal{M}_{n^2N}$, such that the following LMI holds*

$$\begin{bmatrix} P\overline{D} + \overline{D}P - \varepsilon_1 \overline{L_f}^T \overline{L_f} - \varepsilon_2 \overline{L_g}^T \overline{L_g} & P\overline{A} & P\overline{B} \\ * & \varepsilon_1 I_{n^2N} & 0 \\ * & * & \varepsilon_2 I_{n^2N} \end{bmatrix} > 0. \tag{7.3.5}$$

Proof. Define the function $H : \mathbb{R}^{n^2N} \to \mathbb{R}^{n^2N}$,

$$H(W) = -\overline{D}W + \overline{A}\,\overline{f}(W) + \overline{B}\,\overline{g}(W) + \overline{U}. \tag{7.3.6}$$

First, we will prove that H is injective. We assume by contradiction that there are W, $W' \in \mathbb{R}^{n^2N}$, $W \neq W'$, for which $H(W) = H(W')$. This equality can be written as

$$-\overline{D}(W - W') + \overline{A}(\overline{f}(W) - \overline{f}(W')) + \overline{B}(\overline{g}(W) - \overline{g}(W')) = 0. \tag{7.3.7}$$

If we left multiply this relation by $2(W - W')^T P$, we get that

$$2(W - W')^T P(-\overline{D}(W - W') + \overline{A}(\overline{f}(W) - \overline{f}(W')) + \overline{B}(\overline{g}(W) - \overline{g}(W'))) = 0, \tag{7.3.8}$$

which is equivalent with

$$\begin{aligned} &(W - W')^T(-P\overline{D} - \overline{D}P)(W - W') + 2(W - W')^T P\overline{A}(\overline{f}(W) - \overline{f}(W')) \\ &+2(W - W')^T P\overline{B}(\overline{g}(W) - \overline{g}(W')) = 0, \end{aligned} \tag{7.3.9}$$

From Assumption 7.1, we have that

$$\begin{aligned} &||f_j(X) - f_j(X')|| \leq l_j^f ||X - X'|| \\ \Leftrightarrow\ &||\text{vec}(f_j(X)) - \text{vec}(f_j(X'))|| \leq l_j^f ||\text{vec}(X) - \text{vec}(X')||, \end{aligned}$$

for $j \in \{1, \dots N\}$. Taking into account the above notations, this can be written as

$$(f(W) - f(W'))^T(f(W) - f(W')) \leq (W - W')^T \overline{L_f}^T \overline{L_f}(W - W'). \tag{7.3.10}$$

In the same way, we can deduce that

$$(g(W) - g(W'))^T(g(W) - g(W')) \leq (W - W')^T \overline{L_g}^T \overline{L_g}(W - W'). \tag{7.3.11}$$

Now, we have from (7.3.9) that

$$\begin{aligned} & (W - W')^T(-P\overline{D} - \overline{D}P)(W - W') + 2(W - W')^T P\overline{A}(\overline{f}(W) - \overline{f}(W')) \\ & +2(W - W')^T P\overline{B}(\overline{g}(W) - \overline{g}(W')) \\ \leq\ & (W - W')^T(-P\overline{D} - \overline{D}P)(W - W') + \varepsilon_1(\overline{f}(W) - \overline{f}(W'))^T(\overline{f}(W) - \overline{f}(W')) \\ & +\varepsilon_1^{-1}(W - W')^T P\overline{A}\,\overline{A}^T P(W - W') + \varepsilon_2(\overline{g}(W) - \overline{g}(W'))^T(\overline{g}(W) - \overline{g}(W')) \\ & +\varepsilon_2^{-1}(W - W')^T P\overline{B}\,\overline{B}^T P(W - W') \\ \leq\ & (W - W')^T(-P\overline{D} - \overline{D}P)(W - W') + \varepsilon_1(W - W')^T \overline{L}(W - W') \\ & +\varepsilon_1^{-1}(W - W')^T P\overline{A}\,\overline{A}^T P(W - W') + \varepsilon_2(W - W')^T \overline{M}(W - W') \\ & +\varepsilon_2^{-1}(W - W')^T P\overline{B}\,\overline{B}^T P(W - W') \\ =\ & -(W - W')^T(P\overline{D} + \overline{D}P - \varepsilon_1 \overline{L_f}^T \overline{L_f} - \varepsilon_2 \overline{L_g}^T \overline{L_g} - \varepsilon_1^{-1} P\overline{A}\,\overline{A}^T P \\ & -\varepsilon_2^{-1} P\overline{B}\,\overline{B}^T P)(W - W'), \end{aligned} \tag{7.3.12}$$

where we also took into account Lemma 7.2 and inequalities (7.3.10) and (7.3.11).

From the condition (7.3.5), using Schur's complement, we get that

$$P\overline{D} + \overline{D}P - \varepsilon_1 \overline{L_f}^T \overline{L_f} - \varepsilon_2 \overline{L_g}^T \overline{L_g} - \varepsilon_1^{-1} P\overline{A}\,\overline{A}^T P - \varepsilon_2^{-1} P\overline{B}\,\overline{B}^T P > 0, \tag{7.3.13}$$

which, plugged back into (7.3.12), finally yields

$$H(W) - H(W') < 0,$$

which is an obvious contradiction with our initial assumption. Thus, H is injective.

Next, we will prove that $||H(W)|| \to \infty$ as $||W|| \to \infty$. For this, we deduce from (7.3.13) that there exists a sufficiently small $\varepsilon > 0$, such that

$$-P\overline{D} - \overline{D}P + \varepsilon_1 \overline{L_f}^T \overline{L_f} + \varepsilon_2 \overline{L_g}^T \overline{L_g} + \varepsilon_1^{-1} P\overline{A}\,\overline{A}^T P + \varepsilon_2^{-1} P\overline{B}\,\overline{B}^T P < -\varepsilon I_{n^2N}.$$

Taking $W' = 0$ in (7.3.12), gives

$$\begin{aligned} 2W^T P(H(W) - H(0)) &\leq W^T(-P\overline{D} - \overline{D}P + \varepsilon_1 \overline{L_f}^T \overline{L_f} + \varepsilon_2 \overline{L_g}^T \overline{L_g} + \varepsilon_1^{-1} P\overline{A}\,\overline{A}^T P \\ &\quad + \varepsilon_2^{-1} P\overline{B}\,\overline{B}^T P)W \\ &< -\varepsilon ||W||^2. \end{aligned} \tag{7.3.14}$$

From the Cauchy-Schwarz inequality and relation (7.3.14), we obtain

$$2||W||||P||(||H(W)|| + ||H(0)||) > \varepsilon ||W||^2,$$

which immediately yields that $||H(W)|| \to \infty$ when $||W|| \to \infty$.

Applying Lemma 7.1, we get that H is a homeomorphism of $\mathbb{R}^{n^2N}$ onto itself. This means that the equation $H(W) = 0$ has a unique solution, and so system (7.3.4) also has a unique equilibrium point, which we will denote by $\hat{W}$.

We now shift this equilibrium point to the origin, and the system (7.3.4) becomes

$$\dot{\tilde{W}} = -\overline{D}\tilde{W} + \overline{A}\,\tilde{f}(\tilde{W}) + \overline{B}\,\tilde{g}(\tilde{W}^\tau), \tag{7.3.15}$$

where $\tilde{f}(\tilde{W}) = \overline{f}(\tilde{W} + \hat{W}) - \overline{f}(\hat{W})$ and $\tilde{g}(\tilde{W}^\tau) = \overline{g}(\tilde{W}^\tau + \hat{W}) - \overline{g}(\hat{W})$. Construct the following Lyapunov-Krasovskii functional:

$$V(\tilde{W}(t)) = \tilde{W}^T(t) P \tilde{W}(t) + \int_{t-\tau}^{t} \tilde{W}(s)^T Q \tilde{W}(s) ds,$$

where $Q \in \mathcal{M}_{n^2N}$, $Q > 0$.

The derivative of $V(\tilde{W}(t))$ with respect to t along the trajectories of system (7.3.15) can be computed as

$$\begin{aligned} \dot{V}(\tilde{W}) &= \dot{\tilde{W}}^T P\tilde{W} + \tilde{W}^T P\dot{\tilde{W}} + \tilde{W}^T Q\tilde{W} - \tilde{W}^{\tau T} Q\tilde{W}^\tau \\ &= \tilde{W}^T P(-\overline{D}\tilde{W} + \overline{A}\,\tilde{f}(\tilde{W}) + \overline{B}\,\tilde{g}(\tilde{W}^\tau)) + (-\overline{D}\tilde{W} + \overline{A}\,\tilde{f}(\tilde{W}) + \overline{B}\,\tilde{g}(\tilde{W}^\tau))^T P\tilde{W} \\ &\quad + \tilde{W}^T Q\tilde{W} - \tilde{W}^{\tau T} Q\tilde{W}^\tau \\ &= \tilde{W}^T(-P\overline{D} - \overline{D}P)\tilde{W} + \tilde{W}^T P\overline{A}\,\tilde{f}(\tilde{W}) + \tilde{f}^T(\tilde{W})\overline{A}^T P\tilde{W} + \tilde{W}^T P\overline{B}\,\tilde{g}(\tilde{W}^\tau) \\ &\quad + \tilde{g}^T(\tilde{W}^\tau)\overline{B}^T P\tilde{W} + \tilde{W}^T Q\tilde{W} - \tilde{W}^{\tau T} Q\tilde{W}^\tau. \end{aligned} \tag{7.3.16}$$

Multiplying (7.3.10) and (7.3.11) by $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$, respectively, yields

$$0 \leq \varepsilon_1(\tilde{W}^T\overline{L_f}^T\overline{L_f}\tilde{W} - \tilde{f}^T(\tilde{W})\tilde{f}(\tilde{W})), \tag{7.3.17}$$

$$0 \leq \varepsilon_2(\tilde{W}^{\tau T}\overline{L_g}^T\overline{L_g}\tilde{W}^\tau - \tilde{g}^T(\tilde{W}^\tau)\tilde{g}(\tilde{W}^\tau)). \tag{7.3.18}$$

Together with (7.3.16), inequalities (7.3.17) and (7.3.18) give

$$\dot{V}(\tilde{W}) \leq \xi^T\Omega\xi, \tag{7.3.19}$$

where

$$\xi = \begin{bmatrix}\tilde{W}^T & \tilde{W}^{\tau T} & \tilde{f}^T(\tilde{W}) & \tilde{g}^T(\tilde{W}^\tau)\end{bmatrix}^T,$$

$$\Omega = \begin{bmatrix}\Omega_{11} & 0 & P\overline{A} & P\overline{B} \\ * & -Q+\varepsilon_2\overline{L_g}^T\overline{L_g} & 0 & 0 \\ * & * & -\varepsilon_1 I_{n^2N} & 0 \\ * & * & * & -\varepsilon_2 I_{n^2N}\end{bmatrix},$$

$$\Omega_{11} = -P\overline{D} - \overline{D}P + Q + \varepsilon_1\overline{L_f}^T\overline{L_f}.$$

Now, we have $\Omega < 0$ if and only if $Q > \varepsilon_2\overline{L_g}^T\overline{L_g}$ and

$$\begin{bmatrix}-P\overline{D} - D\overline{P} + Q + \varepsilon_1\overline{L_f}^T\overline{L_f} & P\overline{A} & P\overline{B} \\ * & -\varepsilon_1 I_{n^2N} & 0 \\ * & * & -\varepsilon_2 I_{n^2N}\end{bmatrix} < 0. \tag{7.3.20}$$

Inequality (7.3.20) together with $Q > \varepsilon_2\overline{L_g}^T\overline{L_g}$ are equivalent with condition (7.3.5), which means that (7.3.19) gives

$$\dot{V}(\tilde{W}) < 0,$$

from which we can deduce that the equilibrium point of (7.3.4) is globally asymptotically stable, thus ending the proof of the theorem. □

Theorem 7.2. *If Assumption 7.1 holds, then the equilibrium point of system* (7.3.4) *is globally asymptotically stable if there exist real numbers $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$, and positive definite matrices $P, Q, R, S \in \mathcal{M}_{n^2N}$ such that the following LMI holds*

$$\Pi = \begin{bmatrix}\Pi_{11} & \Pi_{12} & \Pi_{13} & 0 & \Pi_{15} \\ * & \Pi_{22} & 0 & 0 & 0 \\ * & * & \Pi_{33} & 0 & \Pi_{35} \\ * & * & * & \Pi_{44} & 0 \\ * & * & * & * & \Pi_{55}\end{bmatrix} < 0, \tag{7.3.21}$$

where $\Pi_{11} = -P\overline{D} - \overline{D}P + Q + \varepsilon_1\overline{L_f}^T\overline{L_f} + \tau\overline{D}R\overline{D} - \tau^{-1}R$, $\Pi_{12} = \tau^{-1}R$, $\Pi_{13} = P\overline{A} - \tau\overline{D}R\overline{A}$, $\Pi_{15} = P\overline{B} - \tau\overline{D}R\overline{B}$, $\Pi_{22} = -Q + \varepsilon_2\overline{L_g}^T\overline{L_g} - \tau^{-1}R$, $\Pi_{33} = S - \varepsilon_1 I_{n^2N} + \tau\overline{A}^T R\overline{A}$, $\Pi_{35} = \tau\overline{A}^T R\overline{B}$, $\Pi_{44} = -S$, $\Pi_{55} = -\varepsilon_2 I_{n^2N} + \tau\overline{B}^T R\overline{B}$.

Proof. Consider the following Lyapunov-Krasovskii functional

$$\begin{aligned} V(\tilde{W}(t)) &= \tilde{W}^T(t)P\tilde{W}(t) + \int_{t-\tau}^{t}\tilde{W}(s)^TQ\tilde{W}(s)ds + \int_{-\tau}^{0}\int_{t+\theta}^{t}\dot{\tilde{W}}^T(s)R\dot{\tilde{W}}(s)dsd\theta \\ &\quad + \int_{t-\tau}^{t}\tilde{f}^T(\tilde{W}(s))S\tilde{f}(\tilde{W}(s))ds. \end{aligned}$$

Its derivative with respect to t along the trajectories of system (7.3.15) is

$$\begin{aligned}\dot{V}(\tilde{W}) &= \dot{\tilde{W}}^T P\tilde{W} + \tilde{W}^T P\dot{\tilde{W}} + \tilde{W}^T Q\tilde{W} - \tilde{W}^{\tau T} Q\tilde{W}^\tau + \tau\dot{\tilde{W}}^T R\dot{\tilde{W}} \\ &\quad - \int_{t-\tau}^{t} \dot{\tilde{W}}^T(s) R\dot{\tilde{W}}(s)ds + \tilde{f}^T(\tilde{W}) S\tilde{f}(\tilde{W}) - \tilde{f}^T(\tilde{W}^\tau) S\tilde{f}(\tilde{W}^\tau) \\ &\leq \tilde{W}^T P(-\overline{D}\tilde{W} + \overline{A}\,\tilde{f}(\tilde{W}) + \overline{B}\,\tilde{g}(\tilde{W}^\tau)) + (-\overline{D}\tilde{W} + \overline{A}\,\tilde{f}(\tilde{W}) + \overline{B}\,\tilde{g}(\tilde{W}^\tau))^T P\tilde{W} \\ &\quad + \tilde{W}^T Q\tilde{W} - \tilde{W}^{\tau T} Q\tilde{W}^\tau + \tau\dot{\tilde{W}}^T R\dot{\tilde{W}} - \tau^{-1}\left(\int_{t-\tau}^{t} \dot{\tilde{W}}(s)ds\right)^T R\left(\int_{t-\tau}^{t} \dot{\tilde{W}}(s)ds\right) \\ &\quad + \tilde{f}^T(\tilde{W}) S\tilde{f}(\tilde{W}) - \tilde{f}^T(\tilde{W}^\tau) S\tilde{f}(\tilde{W}^\tau), \end{aligned} \tag{7.3.22}$$

where we used Lemma 7.3 to get the inequality

$$\left(\int_{t-\tau}^{t} \dot{\tilde{W}}(s)ds\right)^T R\left(\int_{t-\tau}^{t} \dot{\tilde{W}}(s)ds\right) \leq \tau\int_{t-\tau}^{t} \dot{\tilde{W}}^T(s) R\dot{\tilde{W}}(s)ds.$$

There exist two real numbers $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ so that relations (7.3.17) and (7.3.18) hold, and combining them with (7.3.22) yields

$$\dot{V}(\tilde{W}) \leq \zeta^T \Pi \zeta, \tag{7.3.23}$$

where

$$\zeta = \begin{bmatrix}\tilde{W}^T & \tilde{W}^{\tau T} & \tilde{f}^T(\tilde{W}) & \tilde{f}^T(\tilde{W}^\tau) & \tilde{g}^T(\tilde{W}^\tau)\end{bmatrix}^T,$$

and Π is given by (7.3.21). Also from condition (7.3.21) we have that $\Pi < 0$, so (7.3.23) becomes

$$\dot{V}(\tilde{W}) < 0,$$

and thus the equilibrium point of system (7.3.4) is globally asymptotically stable. □

7.3.3 Numerical examples

In this section, we give two numerical examples to prove the effectiveness of our results.

Example 7.1. Consider the following two-neuron matrix-valued recurrent neural network with time delay:

$$\begin{cases}\dot{X}_1(t) = -d_1 X_1(t) + \sum_{j=1}^{2} A_{1j} f_j(X_j(t)) + \sum_{j=1}^{2} B_{1j} g_j(X_j(t-\tau)) + U_1, \\ \dot{X}_2(t) = -d_2 X_2(t) + \sum_{j=1}^{2} A_{2j} f_j(X_j(t)) + \sum_{j=1}^{2} B_{1j} g_j(X_j(t-\tau)) + U_2,\end{cases} \tag{7.3.24}$$

where $d_1 = d_2 = 20$,

$$A_{11} = \begin{bmatrix}1 & 1\\ 2 & 2\end{bmatrix},\ A_{12} = \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix},\ A_{21} = \begin{bmatrix}1 & 1\\ 2 & 2\end{bmatrix},\ A_{22} = \begin{bmatrix}1 & 2\\ 2 & 1\end{bmatrix},$$

$$B_{11} = \begin{bmatrix}1 & 1\\ 2 & 2\end{bmatrix},\ B_{12} = \begin{bmatrix}1 & 2\\ 2 & 1\end{bmatrix},\ B_{21} = \begin{bmatrix}1 & 2\\ 3 & 2\end{bmatrix},\ B_{22} = \begin{bmatrix}1 & 3\\ 2 & 3\end{bmatrix},$$

$$U_1 = \begin{bmatrix}-14 & 23\\ -45 & 34\end{bmatrix},\ U_2 = \begin{bmatrix}-43 & 33\\ -23 & 13\end{bmatrix},$$

$$f\left(\begin{bmatrix}[X]_{11} & [X]_{12}\\ [X]_{21} & [X]_{22}\end{bmatrix}\right) = \begin{bmatrix}\frac{1}{1+e^{-[X]_{11}}} & \frac{1}{1+e^{-[X]_{12}}}\\ \frac{1}{1+e^{-[X]_{21}}} & \frac{1}{1+e^{-[X]_{22}}}\end{bmatrix},$$

$$g\left(\begin{bmatrix}[X]_{11} & [X]_{12}\\ [X]_{21} & [X]_{22}\end{bmatrix}\right) = 0.5\begin{bmatrix}\frac{1-e^{-[X]_{11}}}{1+e^{-[X]_{11}}} & \frac{1-e^{-[X]_{12}}}{1+e^{-[X]_{12}}}\\ \frac{1-e^{-[X]_{21}}}{1+e^{-[X]_{21}}} & \frac{1-e^{-[X]_{22}}}{1+e^{-[X]_{22}}}\end{bmatrix},$$

from which we get that $l_1 = l_2 = \frac{1}{2}$ and $m_1 = m_2 = 1$. The time delay is $\tau = 0.5$.

Solving the LMI condition (7.3.5) in Theorem 7.1, we obtain that system (7.3.24) has a unique equilibrium point which is globally asymptotically stable for $\varepsilon_1 = 12.4719$, $\varepsilon_2 = 12.1648$ and

$$P = \begin{bmatrix}
0.6951 & -0.0110 & 0 & 0 & -0.0119 & -0.0158 & 0 & 0\\
-0.0110 & 0.6854 & 0 & 0 & -0.0141 & -0.0222 & 0 & 0\\
0 & 0 & 0.6951 & -0.0110 & 0 & 0 & -0.0119 & -0.0158\\
0 & 0 & -0.0110 & 0.6854 & 0 & 0 & -0.0141 & -0.0222\\
-0.0119 & -0.0141 & 0 & 0 & 0.6862 & -0.0205 & 0 & 0\\
-0.0158 & -0.0222 & 0 & 0 & -0.0205 & 0.6729 & 0 & 0\\
0 & 0 & -0.0119 & -0.0141 & 0 & 0 & 0.6862 & -0.0205\\
0 & 0 & -0.0158 & -0.0222 & 0 & 0 & -0.0205 & 0.6729
\end{bmatrix}.$$

Example 7.2. In this example, consider the same two-neuron matrix-valued recurrent neural network in (7.3.24), but with the following parameters: $d_1 = 2.5$, $d_2 = 0.7$,

$$A_{11} = \begin{bmatrix}0.05 & 0.04\\ 0.09 & 0.2\end{bmatrix},\ A_{12} = \begin{bmatrix}0.02 & 0.03\\ 0.09 & 0.18\end{bmatrix},\ A_{21} = \begin{bmatrix}0.01 & 0.03\\ 0.07 & 0.24\end{bmatrix},\ A_{22} = \begin{bmatrix}0.07 & 0.09\\ 0.09 & 0.13\end{bmatrix},$$

$$B_{11} = \begin{bmatrix}0.23 & 0.93\\ 0.03 & 0.5\end{bmatrix},\ B_{12} = \begin{bmatrix}0.18 & 0.87\\ 0.02 & 0.45\end{bmatrix},\ B_{21} = \begin{bmatrix}0.17 & 0.98\\ 0.02 & 0.6\end{bmatrix},\ B_{22} = \begin{bmatrix}0.24 & 0.89\\ 0.04 & 0.54\end{bmatrix},$$

$$U_1 = \begin{bmatrix}-1 & 2\\ -4 & 3\end{bmatrix},\ U_2 = \begin{bmatrix}-4 & 3\\ -2 & 1\end{bmatrix},$$

$$f\left(\begin{bmatrix}[X]_{11} & [X]_{12}\\ [X]_{21} & [X]_{22}\end{bmatrix}\right) = 0.3\begin{bmatrix}\frac{1}{1+e^{-[X]_{11}}} & \frac{1}{1+e^{-[X]_{12}}}\\ \frac{1}{1+e^{-[X]_{21}}} & \frac{1}{1+e^{-[X]_{22}}}\end{bmatrix},$$

$$g\left(\begin{bmatrix}[X]_{11} & [X]_{12}\\ [X]_{21} & [X]_{22}\end{bmatrix}\right) = 0.2\begin{bmatrix}\frac{1-e^{-[X]_{11}}}{1+e^{-[X]_{11}}} & \frac{1-e^{-[X]_{12}}}{1+e^{-[X]_{12}}}\\ \frac{1-e^{-[X]_{21}}}{1+e^{-[X]_{21}}} & \frac{1-e^{-[X]_{22}}}{1+e^{-[X]_{22}}}\end{bmatrix},$$

from which we now have that $l_1 = l_2 = 0.15$ and $m_1 = m_2 = 0.4$. In this case, the time delay is $\tau = 2$.

By solving the LMI in condition (7.3.21) in Theorem 7.2, we get that system (7.3.24) has a unique equilibrium point which is globally asymptotically stable for $\varepsilon_1 = 65.8031$, $\varepsilon_2 = 87.8717$ and

$$P = \begin{bmatrix}
51.7045 & -3.1676 & 0 & 0 & -8.3266 & -5.0198 & 0 & 0\\
-3.1676 & 55.0336 & 0 & 0 & -3.9886 & -3.9673 & 0 & 0\\
0 & 0 & 51.7045 & -3.1676 & 0 & 0 & -8.3266 & -5.0198\\
0 & 0 & -3.1676 & 55.0336 & 0 & 0 & -3.9886 & -3.9673\\
-8.3266 & -3.9886 & 0 & 0 & 36.1497 & -17.6836 & 0 & 0\\
-5.0198 & -3.9673 & 0 & 0 & -17.6836 & 53.5836 & 0 & 0\\
0 & 0 & -8.3266 & -3.9886 & 0 & 0 & 36.1497 & -17.6836\\
0 & 0 & -5.0198 & -3.9673 & 0 & 0 & -17.6836 & 53.5836
\end{bmatrix},$$

$$Q = \begin{bmatrix} 42.6488 & -1.0556 & 0 & 0 & -5.1061 & -3.5766 & 0 & 0 \\ -1.0556 & 43.8133 & 0 & 0 & -2.9332 & -1.9195 & 0 & 0 \\ 0 & 0 & 42.6488 & -1.0556 & 0 & 0 & -5.1061 & -3.5766 \\ 0 & 0 & -1.0556 & 43.8133 & 0 & 0 & -2.9332 & -1.9195 \\ -5.1061 & -2.9332 & 0 & 0 & 22.0132 & -10.0197 & 0 & 0 \\ -3.5766 & -1.9195 & 0 & 0 & -10.0197 & 31.9109 & 0 & 0 \\ 0 & 0 & -5.1061 & -2.9332 & 0 & 0 & 22.0132 & -10.0197 \\ 0 & 0 & -3.5766 & -1.9195 & 0 & 0 & -10.0197 & 31.9109 \end{bmatrix},$$

$$R = \begin{bmatrix} 14.7012 & -1.0320 & 0 & 0 & -5.2659 & -3.0947 & 0 & 0 \\ -1.0320 & 15.7960 & 0 & 0 & -2.4228 & -2.6857 & 0 & 0 \\ 0 & 0 & 14.7012 & -1.0320 & 0 & 0 & -5.2659 & -3.0947 \\ 0 & 0 & -1.0320 & 15.7960 & 0 & 0 & -2.4228 & -2.6857 \\ -5.2659 & -2.4228 & 0 & 0 & 17.3828 & -5.6208 & 0 & 0 \\ -3.0947 & -2.6857 & 0 & 0 & -5.6208 & 23.1786 & 0 & 0 \\ 0 & 0 & -5.2659 & -2.4228 & 0 & 0 & 17.3828 & -5.6208 \\ 0 & 0 & -3.0947 & -2.6857 & 0 & 0 & -5.6208 & 23.1786 \end{bmatrix},$$

$$S = \begin{bmatrix} 34.3922 & -2.3264 & 0 & 0 & -1.4004 & -2.0082 & 0 & 0 \\ -2.3264 & 27.8865 & 0 & 0 & -4.4491 & -6.2563 & 0 & 0 \\ 0 & 0 & 34.3922 & -2.3264 & 0 & 0 & -1.4004 & -2.0082 \\ 0 & 0 & -2.3264 & 27.8865 & 0 & 0 & -4.4491 & -6.2563 \\ -1.4004 & -4.4491 & 0 & 0 & 32.1886 & -4.1373 & 0 & 0 \\ -2.0082 & -6.2563 & 0 & 0 & -4.1373 & 29.4049 & 0 & 0 \\ 0 & 0 & -1.4004 & -4.4491 & 0 & 0 & 32.1886 & -4.1373 \\ 0 & 0 & -2.0082 & -6.2563 & 0 & 0 & -4.1373 & 29.4049 \end{bmatrix}.$$

7.4 Exponential stability for MVNNs with delay

The presentation in this section follows that in the author's paper [161].

7.4.1 Main results

By shifting to the origin the equilibrium point $\hat{Y}$ of (7.3.4), the system (7.3.4) becomes

$$\dot{\tilde{Y}} = -\overline{D}\tilde{Y} + \overline{A}\,\tilde{f}(\tilde{Y}) + \overline{B}\,\tilde{g}(\tilde{Y}^{\tau}), \tag{7.4.1}$$

where $\tilde{f}(\tilde{Y}) = \overline{f}(\tilde{Y} + \hat{Y}) - \overline{f}(\hat{Y})$ and $\tilde{g}(\tilde{Y}^{\tau}) = \overline{g}(\tilde{Y}^{\tau} + \hat{Y}) - \overline{g}(\hat{Y})$.

Remark 7.3. The system (7.4.1) is equivalent with the system (7.3.1), which means that any property proved about system (7.4.1) will also hold for system (7.3.1). For this reason, we will only study stability properties for the origin of system (7.4.1).

We now give a sufficient condition that ensures the global exponential stability of the origin of system (7.4.1).

Theorem 7.3. *If Assumption 7.1 holds, then the origin of system* (7.4.1) *is globally exponentially stable if there are positive definite matrices P, Q_1, Q_2, Q_3, S_1, S_2, S_3, S_4, positive block-diagonal matrices R_1, R_2, R_3, R_4, all from $\mathcal{M}_{n^2N}$, and $\varepsilon > 0$, so that the following linear matrix inequality (LMI) is true*

$$(\Pi)_{9\times 9} < 0, \tag{7.4.2}$$

where $\Pi_{1,1} = 2\varepsilon P - P\overline{D} - \overline{D}P + Q_1 + \tau S_2 - \tau^{-1}e^{-2\varepsilon\tau}S_1 + \tau\overline{D}S_1\overline{D} + \overline{L_f}^T R_1\overline{L_f} + \overline{L_g}^T R_3\overline{L_g}$, $\Pi_{1,2} = \tau^{-1}e^{-2\varepsilon\tau}S_1$, $\Pi_{1,3} = P\overline{A} - \tau\overline{D}S_1\overline{A}$, $\Pi_{1,6} = P\overline{B} - \tau\overline{D}S_1\overline{B}$, $\Pi_{2,2} = -e^{-2\varepsilon\tau}Q_1 - \tau^{-1}e^{-2\varepsilon\tau}S_1 + \overline{L_f}^T R_2\overline{L_f} + \overline{L_g}^T R_4\overline{L_g}$, $\Pi_{3,3} = Q_2 + \tau S_3 - R_1 + \tau\overline{A}^T S_1\overline{A}$, $\Pi_{3,6} = \tau\overline{A}^T S_1\overline{B}$, $\Pi_{4,4} = -e^{-2\varepsilon\tau}Q_2 - R_2$, $\Pi_{5,5} = Q_3 + \tau S_4 - R_3$, $\Pi_{6,6} = -e^{-2\varepsilon\tau}Q_3 - R_4 + \tau\overline{B}^T S_1\overline{B}$, $\Pi_{7,7} = -\tau^{-1}e^{-2\varepsilon\tau}S_2$, $\Pi_{8,8} = -\tau^{-1}e^{-2\varepsilon\tau}S_3$, $\Pi_{9,9} = -\tau^{-1}e^{-2\varepsilon\tau}S_4$.

Proof. Consider the Lyapunov-Krasovskii functional

$$\begin{aligned}
V(\tilde{Y}(t)) &= e^{2\varepsilon t}\tilde{Y}^T(t)P\tilde{Y}(t) + \int_{t-\tau}^{t} e^{2\varepsilon u}\tilde{Y}^T(u)Q_1\tilde{Y}(u)du + \int_{t-\tau}^{t} e^{2\varepsilon u}\tilde{f}^T(\tilde{Y}(u))Q_2\tilde{f}(\tilde{Y}(u))du \\
&\quad + \int_{t-\tau}^{t} e^{2\varepsilon u}\tilde{g}^T(\tilde{Y}(u))Q_3 g(\tilde{Y}(u))du + \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon u}\dot{\tilde{Y}}^T(u)S_1\dot{\tilde{Y}}(u)dud\theta \\
&\quad + \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon u}\tilde{Y}^T(u)S_2\tilde{Y}(u)dud\theta + \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon u}\tilde{f}^T(\tilde{Y}(u))S_3\tilde{f}(\tilde{Y}(u))dud\theta \\
&\quad + \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon u}\tilde{g}^T(\tilde{Y}(u))S_4 g(\tilde{Y}(u))dud\theta.
\end{aligned}$$

Its derivative with respect to t for system (7.4.1) is

$$\begin{aligned}
\dot{V}(\tilde{Y}) &= e^{2\varepsilon t}\Big[2\varepsilon\tilde{Y}^T P\tilde{Y} + \dot{\tilde{Y}}^T P\tilde{Y} + \tilde{Y}^T P\dot{\tilde{Y}} + \tilde{Y}^T Q_1\tilde{Y} - e^{-2\varepsilon\tau}\tilde{Y}^{\tau T}Q_1\tilde{Y}^\tau + \tilde{f}^T(\tilde{Y})Q_2\tilde{f}(\tilde{Y}) \\
&\quad -e^{-2\varepsilon\tau}\tilde{f}^T(\tilde{Y}^\tau)Q_2\tilde{f}(\tilde{Y}^\tau) + \tilde{g}^T(\tilde{Y})Q_3\tilde{g}(\tilde{Y}) - e^{-2\varepsilon\tau}\tilde{g}^T(\tilde{Y}^\tau)Q_3\tilde{g}(\tilde{Y}^\tau) + \tau\dot{\tilde{Y}}^T S_1\dot{\tilde{Y}} \\
&\quad - \int_{t-\tau}^{t} e^{2\varepsilon(u-t)}\dot{\tilde{Y}}^T(u)S_1\dot{\tilde{Y}}(u)du + \tau\tilde{Y}^T S_2\tilde{Y} - \int_{t-\tau}^{t} e^{2\varepsilon(u-t)}\tilde{Y}^T(u)S_2\tilde{Y}(u)du \\
&\quad +\tau\tilde{f}^T(\tilde{Y})S_3\tilde{f}(\tilde{Y}) - \int_{t-\tau}^{t} e^{2\varepsilon(u-t)}\tilde{f}^T(\tilde{Y}(u))S_3\tilde{f}(\tilde{Y}(u))du + \tau\tilde{g}^T(\tilde{Y})S_4\tilde{g}(\tilde{Y}) \\
&\quad - \int_{t-\tau}^{t} e^{2\varepsilon(u-t)}\tilde{g}^T(\tilde{Y}(u))S_4 g(\tilde{Y}(u))du\Big] \\
&\le e^{2\varepsilon t}\Big[2\varepsilon\tilde{Y}^T P\tilde{Y} + (-\overline{D}\tilde{Y} + \overline{A}\,\tilde{f}(\tilde{Y}) + \overline{B}\,\tilde{g}(\tilde{Y}^\tau))^T P\tilde{Y} + \tilde{Y}^T P(-\overline{D}\tilde{Y} + \overline{A}\,\tilde{f}(\tilde{Y}) \\
&\quad +\overline{B}\,\tilde{g}(\tilde{Y}^\tau)) + \tilde{Y}^T Q_1\tilde{Y} - e^{-2\varepsilon\tau}\tilde{Y}^{\tau T}Q_1\tilde{Y}^\tau + \tilde{f}^T(\tilde{Y})Q_2\tilde{f}(\tilde{Y}) \\
&\quad -e^{-2\varepsilon\tau}\tilde{f}^T(\tilde{Y}^\tau)Q_2\tilde{f}(\tilde{Y}^\tau) + \tilde{g}^T(\tilde{Y})Q_3\tilde{g}(\tilde{Y}) - e^{-2\varepsilon\tau}\tilde{g}^T(\tilde{Y}^\tau)Q_3\tilde{g}(\tilde{Y}^\tau) \\
&\quad +\tau\dot{\tilde{Y}}^T S_1\dot{\tilde{Y}} - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\dot{\tilde{Y}}(u)du\right)^T S_1\left(\int_{t-\tau}^{t}\dot{\tilde{Y}}(u)du\right) \\
&\quad +\tau\tilde{Y}^T S_2\tilde{Y} - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\tilde{Y}(u)du\right)^T S_2\left(\int_{t-\tau}^{t}\tilde{Y}(u)du\right) \\
&\quad +\tau\tilde{f}^T(\tilde{Y})S_3\tilde{f}(\tilde{Y}) - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\tilde{f}(\tilde{Y}(u))du\right)^T S_3\left(\int_{t-\tau}^{t}\tilde{f}(\tilde{Y}(u))du\right) \\
&\quad +\tau\tilde{g}^T(\tilde{Y})S_4\tilde{g}(\tilde{Y}) - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\tilde{g}(\tilde{Y}(u))du\right)^T S_4\left(\int_{t-\tau}^{t}\tilde{g}(\tilde{Y}(u))du\right)\Big],
\end{aligned} \tag{7.4.3}$$

where we used Lemma 7.3 for the inequality.

Assumption 7.1 can be written as

$$||f_j(X) - f_j(X')|| \leq l_j^f ||X - X'||$$

$$\Leftrightarrow ||\text{vec}(f_j(X)) - \text{vec}(f_j(X'))|| \leq l_j^f ||\text{vec}(X) - \text{vec}(X')||,$$

for $j \in \{1, \dots N\}$. Taking into account this inequality (and the analogous one for the functions g_j), and the above notations, there exist positive block-diagonal matrices

$$R_1 = \text{diag}(r_1^1 I_{n^2}, r_2^1 I_{n^2}, \dots, r_N^1 I_{n^2}),\ R_2 = \text{diag}(r_1^2 I_{n^2}, r_2^2 I_{n^2}, \dots, r_N^2 I_{n^2}),$$

$$R_3 = \text{diag}(r_1^3 I_{n^2}, r_2^3 I_{n^2}, \dots, r_N^3 I_{n^2}),\ R_4 = \text{diag}(r_1^4 I_{n^2}, r_2^4 I_{n^2}, \dots, r_N^4 I_{n^2}),$$

such that

$$0 \leq \tilde{Y}^T \overline{L_f}^T R_1 \overline{L_f} \tilde{Y} - \tilde{f}^T(\tilde{Y}) R_1 \tilde{f}(\tilde{Y}),\ 0 \leq \tilde{Y}^{\tau T} \overline{L_f}^T R_2 \overline{L_f} \tilde{Y}^\tau - \tilde{f}^T(\tilde{Y}^\tau) R_2 \tilde{f}(\tilde{Y}^\tau), \quad (7.4.4)$$

$$0 \leq \tilde{Y}^T \overline{L_g}^T R_3 \overline{L_g} \tilde{Y} - \tilde{g}^T(\tilde{Y}) R_3 \tilde{g}(\tilde{Y}),\ 0 \leq \tilde{Y}^{\tau T} \overline{L_g}^T R_4 \overline{L_g} \tilde{Y}^\tau - \tilde{g}^T(\tilde{Y}^\tau) R_4 \tilde{g}(\tilde{Y}^\tau). \quad (7.4.5)$$

Combining (7.4.4) and (7.4.5) with (7.4.3), yields

$$\dot{V}(\tilde{Y}) \leq e^{2\varepsilon t} \zeta^T \Pi \zeta, \quad (7.4.6)$$

where

$$\begin{aligned} \zeta \ = \ & \Big[\ \tilde{Y}^T \quad \tilde{Y}^{\tau T} \quad \tilde{f}^T(\tilde{Y}) \quad \tilde{f}^T(\tilde{Y}^\tau) \quad \tilde{g}^T(\tilde{Y}) \tilde{g}^T(\tilde{Y}^\tau) \\ & \Big(\int_{t-\tau}^t \tilde{Y}(u) du \Big)^T \quad \Big(\int_{t-\tau}^t \tilde{f}(\tilde{Y}(u)) du \Big)^T \quad \Big(\int_{t-\tau}^t \tilde{g}(\tilde{Y}(u)) du \Big)^T \ \Big]^T, \end{aligned}$$

and Π is given by (7.4.2). Also from condition (7.4.2) we have that $\Pi < 0$, so (7.4.6) becomes $\dot{V}(\tilde{Y}) < 0$, from which we infer that $V(\tilde{Y}(t))$ is strictly decreasing for $t \geq 0$. This fact, together with the definition of $V(\tilde{Y}(t))$, imply that

$$e^{2\varepsilon t} \lambda_{\min}(P) ||\tilde{Y}(t)||^2 \leq e^{2\varepsilon t} \tilde{Y}^T(t) P \tilde{Y}(t) \leq V(t) \leq V_0,\ \forall t \geq T,\ T \geq 0,$$

where $V_0 = \max\limits_{0 \leq t \leq T} V(t)$. Consequently, we have that

$$||\tilde{Y}(t)||^2 \leq \frac{V_0}{e^{2\varepsilon t} \lambda_{\min}(P)} \Leftrightarrow ||\tilde{Y}(t)|| \leq M e^{-\varepsilon t},\ \forall t \geq 0,$$

for $M = \sqrt{\frac{V_0}{\lambda_{\min}(P)}}$. Thus, we obtained the global exponential stability for the origin of system (7.4.1). □

7.4.2 Numerical example

Next, we give a numerical example to prove the correctness of the above-derived criterion.

Example 7.3. Let us consider the following delayed matrix-valued Hopfield neural network with two neurons:

$$\begin{cases} \dot{X}_1(t) = -d_1 X_1(t) + \sum_{j=1}^2 A_{1j} f_j(X_j(t)) + \sum_{j=1}^2 B_{1j} g_j(X_j(t-\tau)) + U_1, \\ \dot{X}_2(t) = -d_2 X_2(t) + \sum_{j=1}^2 A_{2j} f_j(X_j(t)) + \sum_{j=1}^2 B_{2j} g_j(X_j(t-\tau)) + U_2, \end{cases} \quad (7.4.7)$$

where $d_1 = d_2 = 11$,

$$A_{11} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ A_{12} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},\ A_{21} = \begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix},\ A_{22} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},$$

$$B_{11} = \begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix},\ B_{12} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},\ B_{21} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ B_{22} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},$$

$$U_1 = \begin{bmatrix} 5 & -10 \\ 15 & 10 \end{bmatrix},\ U_2 = \begin{bmatrix} 5 & 15 \\ -10 & -15 \end{bmatrix},$$

$$f\left(([X]_{ab})_{1\le a,b\le 2}\right) = \left(\frac{1}{1+e^{-[X]_{ab}}}\right)_{1\le a,b\le 2},$$

$$g\left(([X]_{ab})_{1\le a,b\le 2}\right) = \left(\frac{1-e^{-[X]_{ab}}}{1+e^{-[X]_{ab}}}\right)_{1\le a,b\le 2},$$

from which we get that $l_1^f = l_2^f = \frac{1}{2}$ and $l_1^g = l_2^g = 1$. The time delay is $\tau = 0.5$.

Solving the LMI condition (7.4.2) in Theorem 7.3, we obtain that the equilibrium point of system (7.4.7) is globally exponentially stable for $\varepsilon = 0.2$, $R_1 = \text{diag}(3.8444I_4, 4.3276I_4)$, $R_2 = \text{diag}(0.4434I_4, 0.3695I_4)$, $R_3 = \text{diag}(1.8953I_4, 1.1550I_4)$, $R_4 = \text{diag}(0.3570I_4, 0.2839I_4)$,

$$P = \begin{bmatrix} 1.2130 & 0.0891 & 0 & 0 & 0.0534 & -0.0414 & 0 & 0 \\ 0.0891 & 1.0538 & 0 & 0 & -0.0239 & 0.0199 & 0 & 0 \\ 0 & 0 & 1.2130 & 0.0891 & 0 & 0 & 0.0534 & -0.0415 \\ 0 & 0 & 0.0891 & 1.0538 & 0 & 0 & -0.0239 & 0.0199 \\ 0.0534 & -0.0239 & 0 & 0 & 0.8634 & -0.1098 & 0 & 0 \\ -0.0414 & 0.0199 & 0 & 0 & -0.1098 & 0.7639 & 0 & 0 \\ 0 & 0 & 0.0534 & -0.0239 & 0 & 0 & 0.8634 & -0.1098 \\ 0 & 0 & -0.0415 & 0.0199 & 0 & 0 & -0.1098 & 0.7639 \end{bmatrix}.$$

(For brevity, the values of the other matrices are not given.)

7.5 Exponential stability of BAM MVNNs with time-varying delays

An extension of the unidirectional Hopfield neural networks, BAM neural networks [100] have many applications in pattern recognition and automatic control. Time delays appear unavoidably in real life implementations of neural networks, which can lead to oscillations and chaos. Complex-valued BAMs were introduced in [110], quaternion-valued BAMs in [102], and Clifford-valued BAMs in [217]. These facts into account, we study the exponential stability of the equilibrium point of matrix-valued BAM neural networks with time-varying delays.

The presentation in this section follows that in the author's paper [162].

7.5.1 Main results

we consider matrix-valued BAM neural networks, for which the states, weights, and outputs are square matrices from $\mathbb{R}^{n\times n}$. This type of network is defined by the following set of differential

equations:

$$\begin{cases} \dot{X}_i(t) = -d_i^1 X_i(t) + \sum_{j=1}^{P} A_{ij}^1 f_j^1(Y_j(t)) + \sum_{j=1}^{P} B_{ij}^1 g_j^1(Y_j(t-\tau(t))) + U_i^1, \forall i \in \{1,\dots,N\}, \\ \dot{Y}_j(t) = -d_j^2 Y_j(t) + \sum_{i=1}^{N} A_{ji}^2 f_i^2(X_i(t)) + \sum_{i=1}^{N} B_{ji}^2 g_i^2(X_i(t-\tau(t))) + U_j^2, \forall j \in \{1,\dots,P\}, \end{cases} \tag{7.5.1}$$

where $X_i(t), Y_j(t) \in \mathbb{R}^{n\times n}$ represent the states of the neurons at time t, $d_i^1, d_j^2 > 0$ represent the self-feedback weights, $A_{ij}^1, A_{ji}^2 \in \mathbb{R}^{n\times n}$ represent the weights without delay, $B_{ij}^1, B_{ji}^2 \in \mathbb{R}^{n\times n}$ represent the weights with delay, $f_j^1, f_i^2 : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ represent the nonlinear matrix-valued activation functions without delay, $g_j^1, g_i^2 : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ represent the nonlinear matrix-valued activation functions with delay, $\tau(t)$ represents the time-varying delay and we assume that $0 < \tau(t) \leq \tau$, and $\dot{\tau}(t) \leq \tau_d < 1$, $\forall t \geq 0$, and $U_i^1, U_j^2 \in \mathbb{R}^{n\times n}$ represent the external inputs, $\forall i \in \{1,\dots,N\}, \forall j \in \{1,\dots,P\}$.

We need to make an assumption about the activation functions in order to study the stability of the above defined network.

Assumption 7.2. *The matrix-valued activation functions f_j^1, f_i^2, g_j^1, g_i^2 satisfy the following Lipschitz conditions, for any X, $X' \in \mathbb{R}^{n\times n}$:*

$$||f_j^1(X) - f_j^1(X')|| \leq l_j^{f^1}||X - X'||, \quad ||f_i^2(X) - f_i^2(X')|| \leq l_i^{f^2}||X - X'||,$$

$$||g_j^1(X) - g_j^1(X')|| \leq l_j^{g^1}||X - X'||, \quad ||g_i^2(X) - g_i^2(X')|| \leq l_i^{g^2}||X - X'||,$$

where $l_j^{f^1}$, $l_i^{f^2}$, $l_j^{g^1}$, $l_i^{g^2} > 0$ are the Lipschitz constants, $\forall i \in \{1,\dots,N\}$, $\forall j \in \{1,\dots,P\}$. Moreover, we denote $\overline{L_{f^1}} = diag\left(l_j^{f^1} I_{n^2}\right)_{1\leq j\leq P}$, $\overline{L_{f^2}} = diag\left(l_i^{f^2} I_{n^2}\right)_{1\leq i\leq N}$, $\overline{L_{g^1}} = diag\left(l_j^{g^1} I_{n^2}\right)_{1\leq j\leq P}$, $\overline{L_{g^2}} = diag\left(l_i^{g^2} I_{n^2}\right)_{1\leq i\leq N}$.

Now, we will transform the matrix-valued differential equations (7.5.1) into real-valued differential equations. We start by expanding each equation in (7.5.1) into n^2 real-valued equations, one corresponding to each entry in the original matrices:

$$\begin{aligned} [\dot{X}_i(t)]_{ab} &= -d_i^1 [X_i(t)]_{ab} + \sum_{j=1}^{P}\sum_{c=1}^{n} [A_{ij}^1]_{ac}[f_j^1]_{cb}(Y_j(t)) \\ &\quad + \sum_{j=1}^{P}\sum_{c=1}^{n} [B_{ij}^1]_{ac}[g_j^1]_{cb}(Y_j(t-\tau(t))) + [U_i^1]_{ab}, \\ [\dot{Y}_j(t)]_{ab} &= -d_j^2 [Y_j(t)]_{ab} + \sum_{i=1}^{N}\sum_{c=1}^{n} [A_{ji}^2]_{ac}[f_i^2]_{cb}(X_i(t)) \\ &\quad + \sum_{i=1}^{N}\sum_{c=1}^{n} [B_{ji}^2]_{ac}[g_i^2]_{cb}(X_i(t-\tau(t))) + [U_j^2]_{ab}, \end{aligned} \tag{7.5.2}$$

for $1 \leq a, b \leq n$, $i \in \{1,\dots,N\}$, $j \in \{1,\dots,P\}$. By using the vectorization operation, the

above differential equations can be written more compactly as:

$$\begin{aligned}
\text{vec}(\dot{X}_i(t)) &= -d_i^1 I_{n^2}\text{vec}(X_i(t)) + \sum_{j=1}^{P}(I_n \otimes A_{ij}^1)\text{vec}(f_j^1(Y_j(t))) \\
&\quad + \sum_{j=1}^{P}(I_n \otimes B_{ij}^1)\text{vec}(g_j^1(Y_j(t-\tau(t)))) + \text{vec}(U_i^1),\ \forall i \in \{1,\dots,N\}, \\
\text{vec}(\dot{Y}_j(t)) &= -d_j^2 I_{n^2}\text{vec}(Y_j(t)) + \sum_{i=1}^{N}(I_n \otimes A_{ji}^2)\text{vec}(f_i^2(X_i(t))) \\
&\quad + \sum_{i=1}^{N}(I_n \otimes B_{ji}^2)\text{vec}(g_i^2(X_i(t-\tau(t)))) + \text{vec}(U_j^2),\ \forall j \in \{1,\dots,P\}, \quad (7.5.3)
\end{aligned}$$

where $A \otimes B$ denotes the Kronecker product of matrices A and B. If we denote $Z(t) = \left(\text{vec}(X_i(t))^T\right)_{1\le i\le N}^T$, $W(t) = \left(\text{vec}(Y_j(t))^T\right)_{1\le j\le P}^T$,

$$\overline{D^1} = \text{diag}\left(d_i^1 I_{n^2}\right)_{1\le i\le N},\ \overline{A^1} = \left(I_n \otimes A_{ij}^1\right)_{\substack{1\le i\le N\\ 1\le j\le P}},\ \overline{B^1} = \left(I_n \otimes B_{ij}^1\right)_{\substack{1\le i\le N\\ 1\le j\le P}},$$

$$\overline{f^1}(W(t)) = \left(\text{vec}(f_j^1(Y_j(t))^T\right)_{1\le j\le P}^T, \overline{g^1}(W(t-\tau(t))) = \left(\text{vec}(g_j^1(Y_j(t-\tau(t))))^T\right)_{1\le j\le P}^T,$$

$$\overline{D^2} = \text{diag}\left(d_j^2 I_{n^2}\right)_{1\le j\le P}, \overline{A^2} = \left(I_n \otimes A_{ji}^2\right)_{\substack{1\le j\le P\\ 1\le i\le N}}, \overline{B^2} = \left(I_n \otimes B_{ji}^2\right)_{\substack{1\le j\le P\\ 1\le i\le N}},$$

$$\overline{f^2}(Z(t)) = \left(\text{vec}(f_i^2(X_i(t)))^T\right)_{1\le i\le N}^T, \overline{g^2}(Z(t-\tau(t))) = \left(\text{vec}(g_i^2(X_i(t-\tau(t))))^T\right)_{1\le i\le N}^T,$$

$$\overline{U^1} = \left(\text{vec}(U_i^1)^T\right)_{1\le i\le N}^T,\ \overline{U^2} = \left(\text{vec}(U_j^2)^T\right)_{1\le j\le P}^T,$$

system (7.5.1) becomes:

$$\begin{cases}
\dot{Z}(t) = -\overline{D^1}Z(t) + \overline{A^1}\,\overline{f^1}(W(t)) + \overline{B^1}\,\overline{g^1}(W(t-\tau(t))) + \overline{U^1} \\
\dot{W}(t) = -\overline{D^2}W(t) + \overline{A^2}\,\overline{f^2}(Z(t)) + \overline{B^2}\,\overline{g^2}(Z(t-\tau(t))) + \overline{U^2}.
\end{cases} \quad (7.5.4)$$

If we assume that $\left(\hat{Z}^T, \hat{W}^T\right)^T$ is the equilibrium point of (7.5.4), we can shift it to the origin, to obtain

$$\begin{cases}
\dot{\tilde{Z}}(t) = -\overline{D^1}\tilde{Z}(t) + \overline{A^1}\,\tilde{f}^1(\tilde{W}(t)) + \overline{B^1}\,\tilde{g}^1(\tilde{W}(t-\tau(t))) \\
\dot{\tilde{W}}(t) = -\overline{D^2}\tilde{W}(t) + \overline{A^2}\,\tilde{f}^2(\tilde{Z}(t)) + \overline{B^2}\,\tilde{g}^2(\tilde{Z}(t-\tau(t))),
\end{cases} \quad (7.5.5)$$

where $\tilde{Z}(t) = Z(t) - \hat{Z}$, $\tilde{W}(t) = W(t) - \hat{W}$, $\tilde{f}^1(\tilde{W}(t)) = \overline{f^1}(W(t)) - \overline{f^1}(\hat{W})$, $\tilde{g}^1(\tilde{W}(t-\tau(t))) = \overline{g^1}(W(t-\tau(t))) - \overline{g^1}(\hat{W})$, $\tilde{f}^2(\tilde{Z}(t)) = \overline{f^2}(Z(t)) - \overline{f^2}(\hat{Z})$, $\tilde{g}^2(\tilde{Z}(t-\tau(t))) = \overline{g^2}(Z(t-\tau(t))) - \overline{g^2}(\hat{Z})$.

Remark 7.4. Because system (7.5.5) is equivalent with system (7.5.1), and so any property that holds for system (7.5.5), will also be true for system (7.5.1), we will only study the exponential stability of the origin of system (7.5.5).

Remark 7.5. A matrix-valued BAM neural network is not equivalent with a general $nN \times nP$-dimensional real-valued BAM neural network, because, for such a network, the matrices $\overline{A^1}$, $\overline{B^1}$, $\overline{A^2}$, $\overline{B^2}$ would be general unconstrained matrices, and wouldn't have the particular form given above.

We give a sufficient criterion which assures the exponential stability of the origin of system (7.5.5).

Theorem 7.4. *If Assumption 7.2 holds, then the origin of system* (7.5.5) *is exponentially stable if there exist positive definite matrices* $P_1^1,\ P_2^1,\ P_1^2, \ldots, P_6^2,\ P_1^3,\ P_2^3$ *of appropriate dimensions, positive definite block-diagonal matrices* $R_1, \ldots, R_8$*, and* $\varepsilon > 0$*, which satisfy the following linear matrix inequality (LMI):*

$$(\Pi)_{14\times 14} < 0, \tag{7.5.6}$$

where $\Pi_{1,1} = 2\varepsilon P_1^1 - 2\overline{D^1}P_1^1 + P_1^2 + \tau\overline{D^1}P_1^3\overline{D^1} - \tau^{-1}e^{-2\varepsilon\tau}P_1^3 + \overline{L_{f^2}}^T R_1\overline{L_{f^2}} + \overline{L_{g^2}}^T R_3\overline{L_{g^2}}$, $\Pi_{1,2} = \tau^{-1}e^{-2\varepsilon\tau}P_1^3$, $\Pi_{1,11} = P_1^1\overline{A^1} - \tau\overline{D^1}P_1^3\overline{A^1}$, $\Pi_{1,14} = P_1^1\overline{B^1} - \tau\overline{D^1}P_1^3\overline{B^1}$, $\Pi_{2,2} = -\tau^{-1}e^{-2\varepsilon\tau}P_1^3$, $\Pi_{3,3} = -e^{-2\varepsilon\tau}(1-\tau_d)P_1^2 + \overline{L_{f^2}}^T R_2\overline{L_{f^2}} + \overline{L_{g^2}}^T R_4\overline{L_{g^2}}$, $\Pi_{4,4} = P_2^2 + \tau\overline{A^2}^T P_2^3\overline{A^2} - R_1$, $\Pi_{4,7} = \tau\overline{A^2}^T P_2^3\overline{B^2}$, $\Pi_{4,8} = \overline{A^2}^T P_2^1 - \tau\overline{A^2}^T P_2^3\overline{D^2}$, $\Pi_{5,5} = -e^{-2\varepsilon\tau}(1-\tau_d)P_2^2 - R_2$, $\Pi_{6,6} = P_3^2 - R_3$, $\Pi_{7,7} = -e^{-2\varepsilon\tau}(1-\tau_d)P_3^2 + \tau\overline{B^2}^T P_2^3\overline{B^2} - R_4$, $\Pi_{7,8} = \overline{B^2}^T P_2^1 - \tau\overline{B^2}^T P_2^3\overline{D^2}$, $\Pi_{8,8} = 2\varepsilon P_2^1 - 2\overline{D^2}P_2^1 + P_4^2 + \tau\overline{D^2}P_2^3\overline{D^2} - \tau^{-1}e^{-2\varepsilon\tau}P_2^3 + \overline{L_{f^1}}^T R_5\overline{L_{f^1}} + \overline{L_{g^1}}^T R_7\overline{L_{g^1}}$, $\Pi_{8,9} = \tau^{-1}e^{-2\varepsilon\tau}P_2^3$, $\Pi_{9,9} = -\tau^{-1}e^{-2\varepsilon\tau}P_2^3$, $\Pi_{10,10} = -e^{-2\varepsilon\tau}(1-\tau_d)P_4^2 + \overline{L_{f^1}}^T R_6\overline{L_{f^1}} + \overline{L_{g^1}}^T R_8\overline{L_{g^1}}$, $\Pi_{11,11} = P_5^2 + \tau\overline{A^1}^T P_1^3\overline{A^1} - R_5$, $\Pi_{11,14} = \tau\overline{A^1}^T P_1^3\overline{B^1}$, $\Pi_{12,12} = -e^{-2\varepsilon\tau}(1-\tau_d)P_5^2 - R_6$, $\Pi_{13,13} = P_6^2 - R_7$, $\Pi_{14,14} = -e^{-2\varepsilon\tau}(1-\tau_d)P_6^2 + \tau\overline{B^1}^T P_1^3\overline{B^1} - R_8$.

Proof. Consider the following Lyapunov-Krasovskii functional

$$V(t) = V_1(t) + V_2(t) + V_3(t),$$

where

$$V_1(t) = e^{2\varepsilon t}\begin{bmatrix}\tilde{Z}(t)\\ \tilde{W}(t)\end{bmatrix}^T P_1 \begin{bmatrix}\tilde{Z}(t)\\ \tilde{W}(t)\end{bmatrix}, \quad P_1 = \text{diag}(P_1^1, P_2^1),$$

$$V_2(t) = \int_{t-\tau(t)}^{t} e^{2\varepsilon s}\xi^T(s)P_2\xi(s)ds, \quad P_2 = \text{diag}(P_1^2, P_2^2, P_3^2, P_4^2, P_5^2, P_6^2),$$

$$\xi(s) = \begin{bmatrix}\tilde{Z}^T(s) & \tilde{f^2}^T(\tilde{Z}(s)) & \tilde{g^2}^T(\tilde{Z}(s)) & \tilde{W}^T(s) & \tilde{f^1}^T(\tilde{W}(s)) & \tilde{g^1}^T(\tilde{W}(s))\end{bmatrix}^T,$$

$$V_3(t) = \int_{-\tau}^{0}\int_{t+\theta}^{t} e^{2\varepsilon s}\begin{bmatrix}\dot{\tilde{Z}}(s)\\ \dot{\tilde{W}}(s)\end{bmatrix}^T P_3 \begin{bmatrix}\dot{\tilde{Z}}(s)\\ \dot{\tilde{W}}(s)\end{bmatrix} dsd\theta, \quad P_3 = \text{diag}(P_1^3, P_2^3).$$

The derivative of $V(t)$ along the trajectories of system (7.5.5) is

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t),$$

where

$$\dot{V}_1(t) = 2e^{2\varepsilon t}\left[\varepsilon\tilde{Z}^T(t)P_1^1\tilde{Z}(t) + \varepsilon\tilde{W}^T(t)P_2^1\tilde{W}(t) + \dot{\tilde{Z}}^T(t)P_1^1\tilde{Z}(t) + \dot{\tilde{W}}^T(t)P_2^1\tilde{W}(t)\right], \tag{7.5.7}$$

$$\begin{aligned}\dot{V}_2(t) &= e^{2\varepsilon t}\left[\xi^T(t)P_2\xi(t) - e^{-2\varepsilon\tau(t)}(1-\dot{\tau}(t))\xi^T(t-\tau(t))P_2\xi(t-\tau(t))\right]\\ &\le e^{2\varepsilon t}\left[\xi^T(t)P_2\xi(t) - e^{-2\varepsilon\tau}(1-\tau_d)\xi^T(t-\tau(t))P_2\xi(t-\tau(t))\right],\end{aligned} \tag{7.5.8}$$

$$\begin{aligned}\dot{V}_3(t) &= e^{2\varepsilon t}\left[\tau\dot{\tilde{Z}}^T(t)P_1^3\dot{\tilde{Z}}(t) - \int_{t-\tau}^{t} e^{2\varepsilon(s-t)}\dot{\tilde{Z}}^T(s)P_1^3\dot{\tilde{Z}}(s)ds\right.\\ &\quad \left.+\tau\dot{\tilde{W}}^T(t)P_2^3\dot{\tilde{W}}(t) - \int_{t-\tau}^{t} e^{2\varepsilon(s-t)}\dot{\tilde{W}}^T(s)P_2^3\dot{\tilde{W}}(s)ds\right]\\ &\le e^{2\varepsilon t}\left[\tau\dot{\tilde{Z}}^T(t)P_1^3\dot{\tilde{Z}}(t) - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\dot{\tilde{Z}}(s)ds\right)^T P_1^3\left(\int_{t-\tau}^{t}\dot{\tilde{Z}}(s)ds\right)\right.\\ &\quad \left.+\tau\dot{\tilde{W}}^T(t)P_2^3\dot{\tilde{W}}(t) - \tau^{-1}e^{-2\varepsilon\tau}\left(\int_{t-\tau}^{t}\dot{\tilde{W}}(s)ds\right)^T P_2^3\left(\int_{t-\tau}^{t}\dot{\tilde{W}}(s)ds\right)\right],\end{aligned} \tag{7.5.9}$$

where, for the last inequality, we used Lemma 7.3.

From Assumption 7.2, we have that

$$||f_j^1(X) - f_j^1(X')|| \leq l_j^{f^1}||X - X'||$$

$$\Leftrightarrow ||\text{vec}(f_j^1(X)) - \text{vec}(f_j^1(X'))|| \leq l_j^{f^1}||\text{vec}(X) - \text{vec}(X')||,$$

and the analogous ones for f_i^2, g_j^1, g_i^2, $\forall i \in \{1, \ldots, N\}$, $\forall j \in \{1, \ldots, P\}$, for any X, $X' \in \mathbb{R}^{n\times n}$, which allow us to deduce the existence of positive definite block-diagonal matrices $R_1, \ldots, R_8$, such that the following inequalities hold:

$$0 \leq \tilde{Z}^T(t)\overline{L_{f^2}}^T R_1 \overline{L_{f^2}}\tilde{Z}(t) - \tilde{f^2}^T(\tilde{Z}(t))R_1\tilde{f^2}(\tilde{Z}(t)), \tag{7.5.10}$$

$$0 \leq \tilde{Z}^T(t-\tau(t))\overline{L_{f^2}}^T R_2 \overline{L_{f^2}}\tilde{Z}(t-\tau(t)) - \tilde{f^2}^T(\tilde{Z}(t-\tau(t)))R_2\tilde{f^2}(\tilde{Z}(t-\tau(t))), \tag{7.5.11}$$

$$0 \leq \tilde{Z}^T(t)\overline{L_{g^2}}^T R_3 \overline{L_{g^2}}\tilde{Z}(t) - \tilde{g^2}^T(\tilde{Z}(t))R_3\tilde{g^2}(\tilde{Z}(t)), \tag{7.5.12}$$

$$0 \leq \tilde{Z}^T(t-\tau(t))\overline{L_{g^2}}^T R_4 \overline{L_{g^2}}\tilde{Z}(t-\tau(t)) - \tilde{g^2}^T(\tilde{Z}(t-\tau(t)))R_4\tilde{g^2}(\tilde{Z}(t-\tau(t))), \tag{7.5.13}$$

$$0 \leq \tilde{W}^T(t)\overline{L_{f^1}}^T R_5 \overline{L_{f^1}}\tilde{W}(t) - \tilde{f^1}^T(\tilde{W}(t))R_5\tilde{f^1}(\tilde{W}(t)), \tag{7.5.14}$$

$$0 \leq \tilde{W}^T(t-\tau(t))\overline{L_{f^1}}^T R_6 \overline{L_{f^1}}\tilde{W}(t-\tau(t)) - \tilde{f^1}^T(\tilde{W}(t-\tau(t)))R_6\tilde{f^1}(\tilde{W}(t-\tau(t))), \tag{7.5.15}$$

$$0 \leq \tilde{W}^T(t)\overline{L_{g^1}}^T R_7 \overline{L_{g^1}}\tilde{W}(t) - \tilde{g^1}^T(\tilde{W}(t))R_7\tilde{g^1}(\tilde{W}(t)), \tag{7.5.16}$$

$$0 \leq \tilde{W}^T(t-\tau(t))\overline{L_{g^1}}^T R_8 \overline{L_{g^1}}\tilde{W}(t-\tau(t)) - \tilde{g^1}^T(\tilde{W}(t-\tau(t)))R_8\tilde{g^1}(\tilde{W}(t-\tau(t))). \tag{7.5.17}$$

Multiplying inequalities (7.5.10)–(7.5.17) by $e^{2\varepsilon t}$, and adding them to (7.5.7)–(7.5.9), we obtain that

$$\dot{V}(t) \leq e^{2\varepsilon t}\zeta^T(t)\Pi\zeta(t),$$

where

$$\zeta(t) = \big[\ \tilde{Z}^T(t) \quad \tilde{Z}^T(t-\tau) \quad \tilde{Z}^T(t-\tau(t)) \quad f^2(\tilde{Z}(t)) \quad f^2(\tilde{Z}(t-\tau(t))) \quad g^2(\tilde{Z}(t)) \quad g^2(\tilde{Z}(t-\tau(t)))$$

$$\tilde{W}^T(t) \quad \tilde{W}^T(t-\tau) \quad \tilde{W}^T(t-\tau(t)) \quad f^1(\tilde{W}(t)) \quad f^1(\tilde{W}(t-\tau(t))) \quad g^1(\tilde{W}(t)) \quad g^1(\tilde{W}(t-\tau(t)))\ \big]^T,$$

and Π is given in (7.5.6). Also from (7.5.6), we have that $\Pi < 0$, thus $\dot{V}(t) < 0$, showing that $V(t)$ is strictly decreasing for $t \geq 0$. From the definition of $V(t)$, we can write the following inequalities:

$$e^{2\varepsilon t}\lambda_{\min}(P_1)\left\|\begin{bmatrix}\tilde{Z}(t)\\ \tilde{W}(t)\end{bmatrix}\right\|^2 \leq e^{2\varepsilon t}\begin{bmatrix}\tilde{Z}(t)\\ \tilde{W}(t)\end{bmatrix}^T P_1\begin{bmatrix}\tilde{Z}(t)\\ \tilde{W}(t)\end{bmatrix} \leq V(t) \leq V_0,\ \forall t \geq T,\ T \geq 0,$$

where $V_0 = \max\limits_{0\leq t\leq T} V(t)$. Consequently, we have that

$$\left\|\begin{bmatrix}\tilde{Z}(t)\\ \tilde{W}(t)\end{bmatrix}\right\|^2 \leq \frac{V_0}{e^{2\varepsilon t}\lambda_{\min}(P_1)} \Leftrightarrow \left\|\begin{bmatrix}\tilde{Z}(t)\\ \tilde{W}(t)\end{bmatrix}\right\| \leq Me^{-\varepsilon t},\ \forall t \geq 0,$$

where $M = \sqrt{\frac{V_0}{\lambda_{\min}(P_1)}}$. Thus, we obtained the exponential stability for the origin of system (7.5.5), ending the proof of the theorem. □

7.5.2 Numerical example

In order to illustrate the correctness of the above result, we give a numerical example.

Example 7.4. Let us consider the following matrix-valued BAM neural network with time-varying delays:

$$\begin{cases} \dot{X}_1(t) = -d_1^1 X_1(t) + A_{11}^1 f_1^1(Y_1(t)) + B_{11}^1 g_1^1(Y_1(t-\tau(t))) + U_1^1, \\ \dot{X}_2(t) = -d_2^1 X_2(t) + A_{21}^1 f_1^1(Y_1(t)) + B_{21}^1 g_1^1(Y_1(t-\tau(t))) + U_2^1, \\ \dot{Y}_1(t) = -d_1^2 Y_1(t) + \sum\limits_{i=1}^{2} A_{1i}^2 f_i^2(X_i(t)) + \sum\limits_{i=1}^{2} B_{1i}^2 g_i^2(X_i(t-\tau(t))) + U_1^2, \end{cases} \tag{7.5.18}$$

where $N = 2$, $P = 1$, $d_1^1 = d_2^1 = d_1^2 = 20$,

$$A_{11}^1 = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ A_{21}^1 = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ A_{11}^2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},\ A_{12}^2 = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},\ B_{11}^1 = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ B_{21}^1 = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix},$$

$$B_{11}^2 = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},\ B_{12}^2 = \begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix},\ U_1^1 = \begin{bmatrix} -14 & -35 \\ 23 & 44 \end{bmatrix},\ U_2^1 = \begin{bmatrix} -43 & -23 \\ 33 & 13 \end{bmatrix},\ U_1^2 = \begin{bmatrix} -14 & -45 \\ 23 & 34 \end{bmatrix},$$

$$f_1^1\left(([X]_{ab})_{1\le a,b\le 2}\right) = f_1^2\left(([X]_{ab})_{1\le a,b\le 2}\right) = f_2^2\left(([X]_{ab})_{1\le a,b\le 2}\right) = \left(\frac{1}{2(1+e^{-[X]_{ab}})}\right)_{1\le a,b\le 2},$$

$$g_1^1\left(([X]_{ab})_{1\le a,b\le 2}\right) = g_1^2\left(([X]_{ab})_{1\le a,b\le 2}\right) = g_2^2\left(([X]_{ab})_{1\le a,b\le 2}\right) = \left(\frac{1-e^{-[X]_{ab}}}{1+e^{-[X]_{ab}}}\right)_{1\le a,b\le 2},$$

from which we get that $l_1^{f^1} = l_1^{f^2} = l_2^{f^2} = \frac{1}{4}$ and $l_1^{g^1} = l_1^{g^2} = l_2^{g^2} = 1$. The time-varying delay is taken to be $\tau(t) = 0.9|\sin t|$, which implies that $\tau = \tau_d = 0.9$.

From Theorem 7.4 we get that the equilibrium point of system (7.5.18) is exponentially stable, if LMI condition (7.5.6) is satisfied. By solving (7.5.6), we obtain $\varepsilon = 0.1$, $R_1 = \text{diag}(2.3413I_4,\ 2.8806I_4)$, $R_2 = \text{diag}(0.1380I_4, 0.0783I_4)$, $R_3 = \text{diag}(1.5670I_4, 1.6963I_4)$, $R_4 = \text{diag}(0.1849I_4, 0.2477I_4)$, $R_5 = 3.5840I_4$, $R_6 = 0.0426I_4$, $R_7 = 1.4531I_4$, $R_8 = 0.2159I_4$. (The values of the other matrices are not given due to space limitations.)

7.6 Dissipativity of impulsive MVNNs with leakage delay and mixed delays

Time delays are known to appear in practical implementations of neural networks due to the finite switching speed of amplifiers, and can cause instability or chaotic behavior. For this reason, we consider both leakage delay, and time-varying delays in our model. Also, the distribution propagation delays may appear as a consequence of a distribution of conduction velocities along the pathways of a neural network implementation, which compelled us to add continuously distributed delays to our model. On the other hand, impulsive effects express instantaneous changes that naturally occur in electronic networks, caused by switching phenomena, frequency changes, or noise. Taking the above analysis into account, the impulsive matrix-valued neural networks with leakage delay and mixed delays will be studied in this section, by giving sufficient conditions for the dissipativity of such a model.

The presentation in this section follows that in the author's paper [174].

7.6.1 Main results

Consider the following impulsive matrix-valued Hopfield neural network with leakage delay and mixed delays:

$$\begin{cases} \dot{X}_i(t) = -d_i X_i(t-\delta) + \sum_{j=1}^N A_{ij} f_j(X_j(t)) + \sum_{j=1}^N B_{ij} f_j(X_j(t-\tau(t))) \\ \qquad\qquad + \sum_{j=1}^N C_{ij} \int_{t-\sigma(t)}^t f_j(X_j(s))ds + U_i(t),\ t \neq t_k,\ t > 0, \\ \Delta X_i(t_k) = X_i(t_k) - X_i(t_k^-) = J_{ki}(X_i(t_k^-), X_{i,t_k^-}),\ k \in \mathbb{Z}^+, \\ Y_i(t) = f_i(X_i(t)), \end{cases} \tag{7.6.1}$$

for $i = 1, \ldots, N$, where $X_i(t) \in \mathcal{M}_n$ is the state of the ith neuron at time t, $d_i \in \mathbb{R}$, $d_i > 0$, is the self-feedback weight of the ith neuron, $A_{ij} \in \mathcal{M}_n$ is the weight without time delay between the ith and jth neurons, $B_{ij} \in \mathcal{M}_n$ is the weight with time delay between the ith and jth neurons, $C_{ij} \in \mathcal{M}_n$ is the distributed delay weight between the ith and jth neurons, $f_j : \mathcal{M}_n \to \mathcal{M}_n$ represent the nonlinear matrix-valued activation functions, $U_i(t) \in \mathcal{M}_n$ is the external input for the ith neuron, $Y_i(t)$ is the output of the ith neuron, and J_{ki} are the impulsive functions, for $i, j = 1, \ldots, N$. $\delta > 0$ is the leakage delay, $\tau : \mathbb{R} \to \mathbb{R}$ is the time-varying delay, and $\sigma : \mathbb{R} \to \mathbb{R}$ is the distributed delay.

In order to study the dissipativity of the proposed model, we need to make the following assumptions:

Assumption 7.3. *The time-varying delays $\tau : \mathbb{R} \to \mathbb{R}$ and the distributed delays $\sigma : \mathbb{R} \to \mathbb{R}$ are continuously differentiable functions and there exist $\tau,\ \sigma > 0$ and $\tau' < 1$, such that $\tau(t) < \tau$, $\sigma(t) < \sigma$, $\dot{\tau}(t) \leq \tau'$, $\forall t > 0$.*

Assumption 7.4. *The matrix-valued activation functions f_j satisfy the following Lipschitz conditions for any $X,\ X' \in \mathcal{M}_n$:*

$$||f_j(X) - f_j(X')|| \leq l_j^f ||X - X'||,$$

where $l_j^f > 0$ are the Lipschitz constants, for $j = 1, \ldots, N$. We will also denote $\overline{L_f} = diag(l_1^f I_{n^2}, l_2^f I_{n^2}, \ldots, l_N^f I_{n^2})$.

Assumption 7.5. *For the impulsive functions J_{ki}, there exist $F_{ki} \in \mathcal{M}_n$, such that*

$$J_{ki}(X_i(t_k^-), X_{i,t_k^-}) = F_{ki}\left(X_i(t_k^-) - d_i \int_{t_k-\delta}^{t_k} X_i(s)ds\right),$$

for $\forall k \in \mathbb{Z}^+$.

We will now transform the matrix-valued system (7.6.1) into a real-valued system. To do so, we expand each equation in (7.6.1) into n^2 real-valued equations:

$$\begin{cases} [\dot{X}_i(t)]_{ab} = -d_i[X_i(t-\delta)]_{ab} + \sum_{j=1}^N \sum_{c=1}^n [A_{ij}]_{ac} f_j^{cb}(X_j(t)) \\ \qquad + \sum_{j=1}^N \sum_{c=1}^n [B_{ij}]_{ac} f_j^{cb}(X_j(t-\tau(t))) + \sum_{j=1}^N \sum_{c=1}^n [C_{ij}]_{ac} \int_{t-\sigma(t)}^t f_j^{cb}(X_j(s))ds \\ \qquad + [U_i(t)]_{ab},\ t \neq t_k,\ t > 0, \\ [\Delta X_i(t_k)]_{ab} = [X_i(t_k)]_{ab} - [X_i(t_k^-)]_{ab} = [J_{ki}(X_i(t_k^-), X_{i,t_k^-})]_{ab},\ k \in \mathbb{Z}^+, \\ [Y_i(t)]_{ab} = f_i^{ab}(X_i(t)), \end{cases} \tag{7.6.2}$$

for $1 \leq a, b \leq n$, $i = 1, \ldots, N$. Using the vectorization operation, the above system becomes

$$\begin{cases} \text{vec}(\dot{X}_i(t)) = -d_i I_{n^2} \text{vec}(X_i(t-\delta)) + \sum_{j=1}^N (I_n \otimes A_{ij}) \text{vec}(f_j(X_j(t))) \\ \qquad\qquad + \sum_{j=1}^N (I_n \otimes B_{ij}) \text{vec}(f_j(X_j(t-\tau(t)))) \\ \qquad\qquad + \sum_{j=1}^N (I_n \otimes C_{ij}) \int_{t-\sigma(t)}^t \text{vec}(f_j(X_j(s)))ds + \text{vec}(U_i(t)),\ t \neq t_k,\ t > 0, \\ \text{vec}(\Delta X_i(t_k)) = \text{vec}(X_i(t_k)) - \text{vec}(X_i(t_k^-)) = \text{vec}(J_{ki}(X_i(t_k^-), X_{i,t_k^-})),\ k \in \mathbb{Z}^+, \\ \text{vec}(Y_i(t)) = \text{vec}(f_i(X_i(t))), \end{cases} \tag{7.6.3}$$

for $i = 1, \ldots, N$. Lastly, by denoting

$$X(t) = (\text{vec}(X_1(t))^T, \text{vec}(X_2(t))^T, \ldots, \text{vec}(X_N(t))^T)^T,$$

$$Y(t) = (\text{vec}(Y_1(t))^T, \text{vec}(Y_2(t))^T, \ldots, \text{vec}(Y_N(t))^T)^T,$$

$$U(t) = (\text{vec}(U_1(t))^T, \text{vec}(U_2(t))^T, \ldots, \text{vec}(U_N(t))^T)^T,$$

$$J_k(X(t_k^-), X_{t_k^-}) = (\text{vec}(J_{k1}(X_1(t_k^-), X_{1,t_k^-}))^T, \ldots, \text{vec}(J_{kN}(X_N(t_k^-), X_{N,t_k^-}))^T)^T,$$

$$\overline{D} = \text{diag}(d_1 I_{n^2}, d_2 I_{n^2}, \ldots, d_N I_{n^2}),$$

$$\overline{A} = \begin{bmatrix} I_n \otimes A_{11} & I_n \otimes A_{12} & \cdots & I_n \otimes A_{1N} \\ I_n \otimes A_{21} & I_n \otimes A_{22} & \cdots & I_n \otimes A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ I_n \otimes A_{N1} & I_n \otimes A_{N2} & \cdots & I_n \otimes A_{NN} \end{bmatrix},$$

$$\overline{B} = \begin{bmatrix} I_n \otimes B_{11} & I_n \otimes B_{12} & \cdots & I_n \otimes B_{1N} \\ I_n \otimes B_{21} & I_n \otimes B_{22} & \cdots & I_n \otimes B_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ I_n \otimes B_{N1} & I_n \otimes B_{N2} & \cdots & I_n \otimes B_{NN} \end{bmatrix},$$

$$\overline{C} = \begin{bmatrix} I_n \otimes C_{11} & I_n \otimes C_{12} & \cdots & I_n \otimes C_{1N} \\ I_n \otimes C_{21} & I_n \otimes C_{22} & \cdots & I_n \otimes C_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ I_n \otimes C_{N1} & I_n \otimes C_{N2} & \cdots & I_n \otimes C_{NN} \end{bmatrix},$$

$$f(X(t)) = (\text{vec}(f_1(X_1(t)))^T, \text{vec}(f_2(X_2(t)))^T, \ldots, \text{vec}(f_N(X_N(t)))^T)^T,$$

system (7.6.1) becomes

$$\begin{cases} \dot{X}(t) = -\overline{D}X(t-\delta) + \overline{A}f(X(t)) + \overline{B}f(X(t-\tau(t))) \\ \qquad\qquad + \overline{C}\int_{t-\sigma(t)}^t f(X(s))ds + U(t),\ t \neq t_k,\ t > 0, \\ \Delta X(t_k) = X(t_k) - X(t_k^-) = J_k(X(t_k^-), X_{t_k^-}),\ k \in \mathbb{Z}^+, \\ Y(t) = f(X(t)). \end{cases} \tag{7.6.4}$$

Remark 7.6. Systems (7.6.4) and (7.6.1) are equivalent, which means that any property we prove for system (7.6.4), will also hold for system (7.6.1). Thus, from now on, we will only discuss system (7.6.4).

To introduce the dissipativity property, we define the energy supply function as follows:

$$G(U, Y, T) = \langle Y, \mathcal{Q}Y \rangle_T + 2\langle Y, \mathcal{S}U \rangle_T + \langle U, \mathcal{R}U \rangle_T,\ \forall T \geq 0,$$

where and $\mathcal{Q}$, $\mathcal{S}$, and $\mathcal{R}$ are real matrices, with $\mathcal{Q}$, $\mathcal{R}$ symmetric, and $\langle A, B \rangle_T = \int_0^T A^T B dt$.

Definition 7.1. The neural network given in (7.6.4) is said to be strictly $(\mathcal{Q},\mathcal{S},\mathcal{R})$-$\gamma$-dissipative if, for some $\gamma>0$, the following inequality holds under zero initial condition:

$$G(U,Y,T)\geq\gamma\langle U,U\rangle_T,\ \forall T\geq 0. \tag{7.6.5}$$

We will also need the following lemmas:

Lemma 7.4. *([66]) The following inequality holds for any positive definite matrix $M\in\mathcal{M}_{n^2N}$ and vector function $X:[a,b]\to\mathbb{R}^{n^2N}$:*

$$\left(\int_a^b\int_\theta^b X(s)dsd\theta\right)^T M\left(\int_a^b\int_\theta^b X(s)dsd\theta\right)\leq\frac{(b-a)^2}{2}\int_a^b\int_\theta^b X^T(s)MX(s)dsd\theta,$$

where the integrals are well defined.

Lemma 7.5. *([239]) The following inequality holds for any positive definite matrix $M\in\mathcal{M}_{n^2N}$ and vector function $X:[a,b]\to\mathbb{R}^{n^2N}$:*

$$\left(\int_a^b\int_\theta^b\int_\chi^b X(s)dsd\chi d\theta\right)^T M\left(\int_a^b\int_\theta^b\int_\chi^b X(s)dsd\chi d\theta\right)\leq\frac{(b-a)^3}{6}\int_a^b\int_\theta^b\int_\chi^b X^T(s)MX(s)dsd\chi d\theta,$$

where the integrals are well defined.

Lemma 7.6. *([138]) For any vectors $X_1,\ X_2\in\mathbb{R}^{n^2N}$, any positive definite matrix $P\in\mathcal{M}_{n^2N}$, any matrix $Q\in\mathcal{M}_{n^2N}$, and any $\alpha\in(0,1)$, such that $\begin{bmatrix}P & Q\\ Q^T & P\end{bmatrix}\geq 0$, the following linear matrix inequality (LMI) holds:*

$$\frac{1}{\alpha}X_1^TPX_1+\frac{1}{1-\alpha}X_2^TPX_2\geq\begin{bmatrix}X_1\\X_2\end{bmatrix}^T\begin{bmatrix}P & Q\\ Q^T & P\end{bmatrix}\begin{bmatrix}X_1\\X_2\end{bmatrix}.$$

Lemma 7.7. *([194]) For any differentiable function $X:[a,b]\to\mathbb{R}^{n^2N}$ and positive definite matrix $P\in\mathcal{M}_{n^2N}$, the following linear matrix inequality (LMI) holds:*

$$(b-a)\int_a^b\dot{X}^T(s)P\dot{X}(s)ds\geq\xi_1^TP\xi_1+3\xi_2^TP\xi_2,$$

where $\xi_1=X(b)-X(a)$, $\xi_2=X(b)+X(a)-\frac{2}{b-a}\int_a^bX(s)ds$.

We now give a sufficient condition which assures the strict $(\mathcal{Q},\mathcal{S},\mathcal{R})$-$\gamma$-dissipativity of neural network (7.6.4).

Theorem 7.5. *If Assumptions 7.3–7.5 hold, then system* (7.6.4) *is strictly $(\mathcal{Q},\mathcal{S},\mathcal{R})$-$\gamma$-dissipative if there exist positive definite matrices P_1, P_2, P_3, P_4, P_5^1, P_5^2, P_6, P_7, P_8, P_9, P_{10}^1, P_{10}^2, any matrices M_1, M_2, M_3, N_1, N_2, N_3, N_4, positive definite diagonal matrices R_1, R_2, and some constant $\gamma>0$, such that the following linear matrix inequalities (LMIs) hold:*

$$\Pi-\zeta^T\Phi\zeta<0,\ \Phi>0, \tag{7.6.6}$$

$$\Psi^T\Theta\Psi<\Theta, \tag{7.6.7}$$

where $\Pi_{1,1}=-\overline{D}P_2-P_2\overline{D}+P_3+\delta^2P_4+P_5^1+P_6-4\tau^2P_8-9\tau^4P_9+M_1+M_1^T+\overline{L_f}^TR_1\overline{L_f}$, $\Pi_{1,2}=P_1+P_2$, $\Pi_{1,3}=P_2\overline{D}$, $\Pi_{1,6}=M_2-M_1^T$, $\Pi_{1,9}=\overline{D}P_2\overline{D}$, $\Pi_{1,10}=4\tau P_8$, $\Pi_{1,11}=$

$M_3 - M_1^T$, $\Pi_{1,15} = 18\tau^2 P_9$, $\Pi_{2,2} = \tau^2 P_7 + \tau^4 P_8 + \tau^6 P_9 + \sigma^2 P_{10}^1 - N_1 - N_1^T$, $\Pi_{2,3} = -N_1\overline{D} - N_2^T$, $\Pi_{2,7} = N_1\overline{A} + N_3^T$, $\Pi_{2,8} = N_1\overline{B} + N_4^T$, $\Pi_{2,9} = -P_2\overline{D}$, $\Pi_{2,12} = N_1\overline{C}$, $\Pi_{2,16} = N_1$, $\Pi_{3,3} = -P_3 - N_2\overline{D} - \overline{D}N_2^T$, $\Pi_{3,7} = N_2\overline{A} + \overline{D}N_3^T$, $\Pi_{3,8} = N_2\overline{B} + \overline{D}N_4^T$, $\Pi_{3,9} = -\overline{D}P_2\overline{D}$, $\Pi_{3,12} = N_2\overline{C}$, $\Pi_{3,16} = N_2$, $\Pi_{4,4} = -P_6$, $\Pi_{5,5} = -(1-\tau')P_5^1 + \overline{L_f}^T R_2 \overline{L_f}$, $\Pi_{6,6} = -M_2 - M_2^T$, $\Pi_{6,11} = -M_3 - M_2^T$, $\Pi_{7,7} = P_5^2 + \sigma^2 P_{10}^2 - R_1 - N_3\overline{A} - \overline{A}^T N_3^T - \mathcal{Q}$, $\Pi_{7,8} = -N_3\overline{B} - \overline{A}^T N_4^T$, $\Pi_{7,11} = -N_3\overline{C}$, $\Pi_{7,16} = -N_3 - \mathcal{S}$, $\Pi_{8,8} = -(1-\tau')P_5^2 - R_2 - N_4\overline{B} - \overline{B}^T N_4^T$, $\Pi_{8,11} = -N_4\overline{C}$, $\Pi_{8,16} = -N_4$, $\Pi_{9,9} = -P_4$, $\Pi_{10,10} = -4P_8$, $\Pi_{11,11} = -P_{10}^1 - M_3 - M_3^T$, $\Pi_{12,12} = -P_{10}^2$, $\Pi_{15,15} = -36P_9$, $\Pi_{16,16} = -(\mathcal{R} - \gamma I_{n^2N})$,

$$\zeta = \begin{bmatrix} e_1 - e_5 \\ e_1 + e_5 - 2e_{13} \\ e_5 - e_4 \\ e_5 + e_4 - 2e_{14} \end{bmatrix},$$

$$e_i = \begin{bmatrix} 0_{n^2N\times(i-1)n^2N} & I_{n^2N} & 0_{n^2N\times(16-i)n^2N} \end{bmatrix},\ i = 1,\dots,16,$$

$$\Phi = \begin{bmatrix} \overline{P_7} & H \\ H^T & \overline{P_7} \end{bmatrix},$$

$$\overline{P_7} = \begin{bmatrix} P_7 & 0 \\ 0 & 3P_7 \end{bmatrix},$$

$$\Psi = \begin{bmatrix} I_{n^2N} + \overline{F_k} & -\overline{F_k}\,\overline{D} \\ 0 & I_{n^2N} \end{bmatrix},\ \Theta = \begin{bmatrix} P_1 + P_2 & -P_2\overline{D} \\ -\overline{D}P_2 & \overline{D}P_2\overline{D} \end{bmatrix}.$$

Proof. Consider the following Lyapunov–Krasovskii functional

$$V(t) = V_1(t) + V_2(t) + V_3(t) + V_4(t) + V_5(t) + V_6(t) + V_7(t) + V_8(t) + V_9(t) + V_{10}(t),$$

where

$$V_1(t) = X^T(t)P_1X(t),$$

$$V_2(t) = \left(X(t) - \overline{D}\int_{t-\delta}^{t} X(s)ds\right)^T P_2 \left(X(t) - \overline{D}\int_{t-\delta}^{t} X(s)ds\right),$$

$$V_3(t) = \int_{t-\delta}^{t} X^T(s)P_3X(s)ds,$$

$$V_4(t) = \delta\int_{-\delta}^{0}\int_{t+\theta}^{t} X^T(s)P_4X(s)ds,$$

$$V_5(t) = \int_{t-\tau(t)}^{t} \xi^T(s)P_5\xi(s)ds, \quad P_5 = \text{diag}(P_5^1, P_5^2),$$

$$\xi(s) = \begin{bmatrix} X^T(s) & f^T(X(s)) \end{bmatrix}^T,$$

$$V_6(t) = \int_{t-\tau}^{t} X^T(s)P_6X(s)ds,$$

$$V_7(t) = \tau\int_{-\tau}^{0}\int_{t+\theta}^{t} \dot{X}^T(s)P_7\dot{X}(s)dsd\theta,$$

$$V_8(t) = 2\tau^2\int_{-\tau}^{0}\int_{\theta}^{0}\int_{t+\lambda}^{t} \dot{X}^T(s)P_8\dot{X}(s)dsd\lambda d\theta,$$

$$V_9(t) = 6\tau^3\int_{-\tau}^{0}\int_{\theta}^{0}\int_{\chi}^{0}\int_{t+\lambda}^{t} \dot{X}^T(s)P_9\dot{X}(s)dsd\lambda d\chi d\theta,$$

$$V_{10}(t) = \sigma \int_{-\sigma}^{0} \int_{t+\theta}^{t} \nu^T(s) P_{10} \nu(s) ds d\theta, \quad P_{10} = \text{diag}(P_{10}^1, P_{10}^2),$$
$$\nu(s) = \left[\dot{X}^T(s) \quad f^T(X(s)) \right]^T.$$

By taking the derivative of V along the trajectories of system (7.6.4), we have:

$$\dot{V}(t) = \dot{V}_1(t) + \dot{V}_2(t) + \dot{V}_3(t) + \dot{V}_4(t) + \dot{V}_5(t) + \dot{V}_6(t) + \dot{V}_7(t) + \dot{V}_8(t) + \dot{V}_9(t) + \dot{V}_{10}(t),$$

where

$$\dot{V}_1(t) = \dot{X}^T(t) P_1 X(t) + X^T(t) P_1 \dot{X}(t), \tag{7.6.8}$$

$$\begin{aligned} \dot{V}_2(t) &= \left(\dot{X}(t) - \overline{D} X(t) + \overline{D} X(t-\delta) \right)^T P_2 \left(X(t) - \overline{D} \int_{t-\delta}^{t} X(s) ds \right) \\ &\quad + \left(X(t) - \overline{D} \int_{t-\delta}^{t} X(s) ds \right)^T P_2 \left(\dot{X}(t) - \overline{D} X(t) + \overline{D} X(t-\delta) \right), \end{aligned} \tag{7.6.9}$$

$$\dot{V}_3(t) = X^T(t) P_3 X(t) - X^T(t-\delta) P_3 X(t-\delta), \tag{7.6.10}$$

$$\begin{aligned} \dot{V}_4(t) &= \delta^2 X^T(t) P_4 X(t) - \delta \int_{t-\delta}^{t} X^T(s) P_4 X(s) ds \\ &\leq \delta^2 X^T(t) P_4 X(t) - \left(\int_{t-\delta}^{t} X(s) ds \right)^T P_4 \left(\int_{t-\delta}^{t} X(s) ds \right), \end{aligned} \tag{7.6.11}$$

$$\begin{aligned} \dot{V}_5(t) &= \xi^T(t) P_5 \xi(t) - (1 - \dot{\tau}(t)) \xi^T(t - \tau(t)) P_5 \xi(t - \tau(t)) \\ &\leq \xi^T(t) P_5 \xi(t) - (1 - \tau') \xi^T(t - \tau(t)) P_5 \xi(t - \tau(t)), \end{aligned} \tag{7.6.12}$$

$$\dot{V}_6(t) = X^T(t) P_6 X(t) - X^T(t-\tau) P_6 X(t-\tau), \tag{7.6.13}$$

$$\dot{V}_7(t) = \tau^2 \dot{X}^T(t) P_7 \dot{X}(t) - \tau \int_{t-\tau}^{t} \dot{X}^T(s) P_7 \dot{X}(s) ds, \tag{7.6.14}$$

$$\begin{aligned} \dot{V}_8(t) &= \tau^4 \dot{X}^T(t) P_8 \dot{X}(t) - 2\tau^2 \int_{-\tau}^{0} \int_{t+\theta}^{t} \dot{X}^T(s) P_8 \dot{X}(s) ds d\theta \\ &\leq \tau^4 \dot{X}^T(t) P_8 \dot{X}(t) - 4 \left(\int_{-\tau}^{0} \int_{t+\theta}^{t} \dot{X}(s) ds d\theta \right)^T P_8 \left(\int_{-\tau}^{0} \int_{t+\theta}^{t} \dot{X}(s) ds d\theta \right) \\ &= \tau^4 \dot{X}^T(t) P_8 \dot{X}(t) - 4 \left(\int_{-\tau}^{0} (X(t) - X(t+\theta)) d\theta \right)^T P_8 \left(\int_{-\tau}^{0} (X(t) - X(t+\theta)) d\theta \right) \\ &= \tau^4 \dot{X}^T(t) P_8 \dot{X}(t) - 4 \left(\tau X(t) - \int_{t-\tau}^{t} X(s) ds \right)^T P_8 \left(\tau X(t) - \int_{t-\tau}^{t} X(s) ds \right), \end{aligned} \tag{7.6.15}$$

$$\dot{V}_9(t) = \tau^6 \dot{X}^T(t) P_9 \dot{X}(t) - 6\tau^3 \int_{-\tau}^{0} \int_{\theta}^{0} \int_{t+\chi}^{t} \dot{X}^T(s) P_9 \dot{X}(s) ds d\chi d\theta$$

$$
\begin{aligned}
&\leq \tau^6 \dot{X}^T(t) P_9 \dot{X}(t) - 36 \left(\int_{-\tau}^{0} \int_{\theta}^{0} \int_{t+\chi}^{t} \dot{X}(s) ds d\chi d\theta \right)^T P_9 \left(\int_{-\tau}^{0} \int_{\theta}^{0} \int_{t+\chi}^{t} \dot{X}(s) ds d\chi d\theta \right) \\
&= \tau^6 \dot{X}^T(t) P_9 \dot{X}(t) \\
&\quad -36 \left(\int_{-\tau}^{0} \int_{\theta}^{0} \left(X(t) - X(t+\chi) \right) d\chi d\theta \right)^T P_9 \left(\int_{-\tau}^{0} \int_{\theta}^{0} \left(X(t) - X(t+\chi) \right) d\chi d\theta \right) \\
&= \tau^6 \dot{X}^T(t) P_9 \dot{X}(t) - 9 \left(\tau^2 X(t) - 2 \int_{-\tau}^{0} \int_{t+\theta}^{t} X(s) ds \right)^T P_9 \left(\tau^2 X(t) - 2 \int_{-\tau}^{0} \int_{t+\theta}^{t} X(s) ds \right),
\end{aligned} \tag{7.6.16}
$$

$$
\begin{aligned}
\dot{V}_{10}(t) &= \sigma^2 \nu^T(t) P_{10} \nu(t) - \sigma \int_{t-\sigma}^{t} \nu^T(s) P_{10} \nu(s) ds \\
&\leq \sigma^2 \nu^T(t) P_{10} \nu(t) - \sigma(t) \int_{t-\sigma(t)}^{t} \nu^T(s) P_{10} \nu(s) ds \\
&\leq \left(\sigma^2 \nu^T(t) P_{10} \nu(t) - \left(\int_{t-\sigma(t)}^{t} \nu(s) ds \right)^T P_{10} \left(\int_{t-\sigma(t)}^{t} \nu(s) ds \right) \right) \\
&\leq \left(\sigma^2 \nu^T(t) P_{10} \nu(t) - \left(\int_{t-\sigma(t)}^{t} \nu(s) ds \right)^T P_{10} \left(\int_{t-\sigma(t)}^{t} \nu(s) ds \right) \right),
\end{aligned} \tag{7.6.17}
$$

where we used Assumption 7.3 to deduce the inequalities in (7.6.12) and (7.6.17), Lemma 7.3 to deduce the inequalities in (7.6.11) and (7.6.17), Lemma 7.4 for the inequality in (7.6.15), and Lemma 7.5 for the inequality in (7.6.16).

Using Lemmas 7.6–7.7, we have that

$$
\begin{aligned}
-\tau \int_{t-\tau}^{t} \dot{X}^T(s) P_7 \dot{X}(s) ds &= -\tau \int_{t-\tau(t)}^{t} \dot{X}^T(s) P_7 \dot{X}(s) ds - \tau \int_{t-\tau}^{t-\tau(t)} \dot{X}^T(s) P_7 \dot{X}(s) ds \\
&\leq -\frac{\tau}{\tau(t)} \begin{bmatrix} X(t) - X(t-\tau(t)) \\ X(t) + X(t-\tau(t)) - \frac{2}{\tau(t)} \int_{t-\tau(t)}^{t} X(s) ds \end{bmatrix}^T \begin{bmatrix} P_7 & 0 \\ 0 & 3P_7 \end{bmatrix} \\
&\quad \times \begin{bmatrix} X(t) - X(t-\tau(t)) \\ X(t) + X(t-\tau(t)) - \frac{2}{\tau(t)} \int_{t-\tau(t)}^{t} X(s) ds \end{bmatrix} \\
&\quad -\frac{\tau}{\tau - \tau(t)} \begin{bmatrix} X(t-\tau(t)) - X(t-\tau) \\ X(t-\tau(t)) + X(t-\tau) - \frac{2}{\tau-\tau(t)} \int_{t-\tau}^{t-\tau(t)} X(s) ds \end{bmatrix}^T \begin{bmatrix} P_7 & 0 \\ 0 & 3P_7 \end{bmatrix} \\
&\quad \times \begin{bmatrix} X(t-\tau(t)) - X(t-\tau) \\ X(t-\tau(t)) + X(t-\tau) - \frac{2}{\tau-\tau(t)} \int_{t-\tau}^{t-\tau(t)} X(s) ds \end{bmatrix} \\
&= -\mu^T(t) \left[\frac{\tau}{\tau(t)} \begin{bmatrix} e_1 - e_5 \\ e_1 + e_5 - 2e_{13} \end{bmatrix}^T \begin{bmatrix} P_7 & 0 \\ 0 & 3P_7 \end{bmatrix} \begin{bmatrix} e_1 - e_5 \\ e_1 + e_5 - 2e_{13} \end{bmatrix} \right. \\
&\quad \left. + \frac{\tau}{\tau - \tau(t)} \begin{bmatrix} e_5 - e_4 \\ e_5 + e_4 - 2e_{14} \end{bmatrix}^T \begin{bmatrix} P_7 & 0 \\ 0 & 3P_7 \end{bmatrix} \begin{bmatrix} e_5 - e_4 \\ e_5 + e_4 - 2e_{14} \end{bmatrix} \right] \mu(t) \\
&\leq -\mu^T(t) \begin{bmatrix} e_1 - e_5 \\ e_1 + e_5 - 2e_{13} \\ e_5 - e_4 \\ e_5 + e_4 - 2e_{14} \end{bmatrix}^T \begin{bmatrix} \overline{P_7} & H \\ H^T & \overline{P_7} \end{bmatrix} \begin{bmatrix} e_1 - e_5 \\ e_1 + e_5 - 2e_{13} \\ e_5 - e_4 \\ e_5 + e_4 - 2e_{14} \end{bmatrix} \mu(t), \\
&= -\mu^T(t) \zeta^T \Phi \zeta \mu(t),
\end{aligned}
$$

with the condition that $\Phi = \begin{bmatrix} \overline{P_7} & H \\ H^T & \overline{P_7} \end{bmatrix} > 0$, which is true by (7.6.7).

Assumption 7.4 can be rewritten as

$$||f_j(X) - f_j(X')|| \leq l_j^f ||X - X'|| \Leftrightarrow ||\text{vec}(f_j(X)) - \text{vec}(f_j(X'))|| \leq l_j^f ||\text{vec}(X) - \text{vec}(X')||,$$

for $j = 1, \dots N$. From this inequality, we can deduce that there exist positive block-diagonal matrices $R_1 = \text{diag}(r_1^1 I_{n^2}, r_2^1 I_{n^2}, \dots, r_N^1 I_{n^2})$, $R_2 = \text{diag}(r_1^2 I_{n^2}, r_2^2 I_{n^2}, \dots, r_N^2 I_{n^2})$, such that

$$0 \leq X^T(t)\overline{L_f}^T R_1 \overline{L_f} X(t) - f^T(X(t)) R_1 f(X(t)), \tag{7.6.18}$$

$$0 \leq X^T(t-\tau(t))\overline{L_f}^T R_2 \overline{L_f} X(t-\tau(t)) - f^T(X(t-\tau(t))) R_2 f(X(t-\tau(t))). \tag{7.6.19}$$

For any matrices M_1, M_2, M_3, from the Leibniz–Newton formula $X(t) - X(t-\sigma(t)) = \int_{t-\sigma(t)}^t \dot{X}(s)ds$, we have that

$$\begin{aligned}
0 &= \left[X(t) - X(t-\sigma(t)) - \int_{t-\sigma(t)}^t \dot{X}(s)ds\right]^T \left[M_1 X(t) + M_2 X(t-\sigma(t)) + M_3 \int_{t-\sigma(t)}^t \dot{X}(s)ds\right] \\
&= X^T(t)M_1X(t) + X^T(t)M_2X(t-\sigma(t)) + X^T(t)M_3 \int_{t-\sigma(t)}^t \dot{X}(s)ds \\
&\quad - X^T(t-\sigma(t))M_1X(t) - X^T(t-\sigma(t))M_2X(t-\sigma(t)) - X^T(t-\sigma(t))M_3 \int_{t-\sigma(t)}^t \dot{X}(s)ds \\
&\quad - \left(\int_{t-\sigma(t)}^t \dot{X}(s)ds\right)^T M_1X(t) - \left(\int_{t-\sigma(t)}^t \dot{X}(s)ds\right)^T M_2X(t-\sigma(t)) \\
&\quad - \left(\int_{t-\sigma(t)}^t \dot{X}(s)ds\right)^T M_3 \int_{t-\sigma(t)}^t \dot{X}(s)ds.
\end{aligned} \tag{7.6.20}$$

Also, for any matrices N_1, N_2, N_3, N_4, we have that

$$\begin{aligned}
0 &= \left[\dot{X}^T(t)N_1 + X^T(t-\delta)N_2 - f^T(X(t))N_3 - f^T(X(t-\tau(t)))N_4\right] \left[-\dot{X}(t) - \overline{D}X(t-\delta)\right. \\
&\quad \left. + \overline{A}f(X(t)) + \overline{B}f(X(t-\tau(t))) + \overline{C}\int_{t-\sigma(t)}^t f(X(s))ds + U(t)\right] \\
&= -\dot{X}^T(t)N_1\dot{X}(t) - \dot{X}^T(t)N_1\overline{D}X(t-\delta) + \dot{X}^T(t)N_1\overline{A}f(X(t)) + \dot{X}^T(t)N_1\overline{B}f(X(t-\tau(t))) \\
&\quad + \dot{X}^T(t)N_1\overline{C}\int_{t-\sigma(t)}^t f(X(s))ds + \dot{X}^T(t)N_1U(t) \\
&\quad - X^T(t-\delta)N_2\dot{X}(t) - X^T(t-\delta)N_2\overline{D}X(t-\delta) + X^T(t-\delta)N_2\overline{A}f(X(t)) \\
&\quad + X^T(t-\delta)N_2\overline{B}f(X(t-\tau(t))) + X^T(t-\delta)N_2\overline{C}\int_{t-\sigma(t)}^t f(X(s))ds + X^T(t-\delta)N_2U(t) \\
&\quad + f^T(X(t))N_3\dot{X}(t) + f^T(X(t))N_3\overline{D}X(t-\delta) - f^T(X(t))N_3\overline{A}f(X(t)) \\
&\quad - f^T(X(t))N_3\overline{B}f(X(t-\tau(t))) - f^T(X(t))N_3\overline{C}\int_{t-\sigma(t)}^t f(X(s))ds - f^T(X(t))N_3U(t) \\
&\quad + f^T(X(t-\tau(t)))N_4\dot{X}(t) + f^T(X(t-\tau(t)))N_4\overline{D}X(t-\delta) - f^T(X(t-\tau(t)))N_4\overline{A}f(X(t)) \\
&\quad - f^T(X(t-\tau(t)))N_4\overline{B}f(X(t-\tau(t))) - f^T(X(t-\tau(t)))N_4\overline{C}\int_{t-\sigma(t)}^t f(X(s))ds \\
&\quad - f^T(X(t-\tau(t)))N_4U(t).
\end{aligned} \tag{7.6.21}$$

Finally, by combining (7.6.8)–(7.6.17), (7.6.18)–(7.6.19), and the transpose of (7.6.20)–(7.6.21) added to the initial relations, we get that

$$\dot{V}(t) - Y^T(t)\mathcal{Q}Y(t) - 2Y^T(t)\mathcal{S}U(t) - U^T(t)(\mathcal{R} - \gamma I_{n^2N})U(t) \leq \mu^T(t)\left(\Pi - \zeta^T\Phi\zeta\right)\mu(t), \tag{7.6.22}$$

for $t \neq t_k,\ k \in \mathbb{Z}^+$, where Π is defined by (7.6.6), and

$$\begin{aligned}\mu(t) \;=\; \Big[\; & X^T(t) \quad \dot{X}^T(t) \quad X^T(t-\delta) \quad X^T(t-\tau) \quad X^T(t-\tau(t)) \quad X^T(t-\sigma(t)) \quad f^T(X(t)) \\ & f^T(X(t-\tau(t))) \quad \Big(\int_{t-\delta}^{t} X(s)ds\Big)^T \quad \Big(\int_{t-\tau}^{t} X(s)ds\Big)^T \quad \Big(\int_{t-\sigma(t)}^{t} \dot{X}(s)ds\Big)^T \\ & \Big(\int_{t-\sigma(t)}^{t} f(X(s))ds\Big)^T \quad \tfrac{1}{\tau(t)}\Big(\int_{t-\tau(t)}^{t} X(s)ds\Big)^T \quad \tfrac{1}{\tau-\tau(t)}\Big(\int_{t-\tau}^{t-\tau(t)} X(s)ds\Big)^T \\ & \Big(\int_{-\tau}^{0}\int_{t+\theta}^{t} X(s)ds\Big)^T \quad U^T(t) \;\Big]^T.\end{aligned}$$

From (7.6.6) we have that $\Pi - \zeta^T\Phi\zeta < 0$, and thus inequality (7.6.22) yields

$$\dot{V}(t) - Y^T(t)\mathcal{Q}Y(t) - 2Y^T(t)\mathcal{S}U(t) - U^T(t)(\mathcal{R} - \gamma I_{n^2N})U(t) < 0,\ \forall t \in [t_{k-1}, t_k),\ k \in \mathbb{Z}^+. \tag{7.6.23}$$

On the other hand, we have that

$$V_1(t_k) = \begin{bmatrix} X(t) \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix}^T \begin{bmatrix} P_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X(t) \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix},$$

$$V_2(t_k) = \begin{bmatrix} X(t) \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix}^T \begin{bmatrix} P_2 & -P_2\overline{D} \\ -\overline{D}P_2 & \overline{D}P_2\overline{D} \end{bmatrix} \begin{bmatrix} X(t) \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix},$$

which, together with Assumption 7.5 vectorized in the form

$$X(t_k) = (I_{n^2N} + \overline{F_k})X(t_k^-) - \overline{F_k}\,\overline{D}\int_{t_k-\delta}^{t_k} X(s)ds,$$

where

$$\overline{F_k} = \begin{bmatrix} I_n \otimes F_{k1} & 0 & \cdots & 0 \\ 0 & I_n \otimes F_{k2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_n \otimes F_{kN} \end{bmatrix},$$

yield

$$
\begin{aligned}
V_1(t_k)+V_2(t_k) &= \begin{bmatrix} X(t_k) \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix}^T \begin{bmatrix} P_1+P_2 & -P_2\overline{D} \\ -\overline{D}P_2 & \overline{D}P_2\overline{D} \end{bmatrix} \begin{bmatrix} X(t_k) \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix} \\
&= \begin{bmatrix} (I_{n^2N}+\overline{F_k})X(t_k^-) - \overline{F_k}\,\overline{D}\int_{t_k-\delta}^{t_k} X(s)ds \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix}^T \\
&\quad \times \begin{bmatrix} P_1+P_2 & -P_2\overline{D} \\ -\overline{D}P_2 & \overline{D}P_2\overline{D} \end{bmatrix} \begin{bmatrix} (I_{n^2N}+\overline{F_k})X(t_k^-) - \overline{F_k}\,\overline{D}\int_{t_k-\delta}^{t_k} X(s)ds \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix} \\
&= \begin{bmatrix} X(t_k^-) \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix}^T \begin{bmatrix} I_{n^2N}+\overline{F_k}^T & 0 \\ -\overline{D}\,\overline{F_k}^T & I_{n^2N} \end{bmatrix} \begin{bmatrix} P_1+P_2 & -P_2\overline{D} \\ -\overline{D}P_2 & \overline{D}P_2\overline{D} \end{bmatrix} \\
&\quad \times \begin{bmatrix} I_{n^2N}+\overline{F_k} & -\overline{F_k}\,\overline{D} \\ 0 & I_{n^2N} \end{bmatrix} \begin{bmatrix} X(t_k^-) \\ \int_{t_k-\delta}^{t_k} X(s)ds \end{bmatrix} \\
&\leq \begin{bmatrix} X(t_k^-) \\ \int_{t_k^--\delta}^{t_k^-} X(s)ds \end{bmatrix}^T \begin{bmatrix} P_1+P_2 & -P_2\overline{D} \\ -\overline{D}P_2 & \overline{D}P_2\overline{D} \end{bmatrix} \begin{bmatrix} X(t_k^-) \\ \int_{t_k^--\delta}^{t_k^-} X(s)ds \end{bmatrix} \\
&= V_1(t_k^-)+V_2(t_k^-),
\end{aligned}
$$

where we used condition (7.6.6) for the inequality. We also observe that $V_3(t_k) = V_3(t_k^-)$, $V_4(t_k) = V_4(t_k^-)$, $V_5(t_k) = V_5(t_k^-)$, $V_6(t_k) = V_6(t_k^-)$, $V_7(t_k) = V_7(t_k^-)$, $V_8(t_k) = V_8(t_k^-)$, $V_9(t_k) = V_9(t_k^-)$, and $V_{10}(t_k) = V_{10}(t_k^-)$. Hence, we have that

$$V(t_k) \leq V(t_k^-),\ k \in \mathbb{Z}^+. \tag{7.6.24}$$

Now, integrating (7.6.23) from 0 to $T \geq 0$, and also taking (7.6.24) into account, we obtain that

$$V(T) - V(0) \leq \int_0^T \left(Y^T(t)\mathcal{Q}Y(t) - 2Y^T(t)\mathcal{S}U(t) - U^T(t)(\mathcal{R}-\gamma I_{n^2N})U(t)\right) dt,$$

which means that inequality (7.6.5) holds under zero initial condition, which implies that neural network (7.6.4) is strictly $(\mathcal{Q},\mathcal{S},\mathcal{R})$-$\gamma$-dissipative, thus ending the proof of the theorem. □

7.6.2 Numerical examples

Next, we give two numerical examples to prove the correctness of the obtained theoretical results.

Example 7.5. Consider the following matrix-valued neural network with two neurons:

$$
\begin{cases}
\dot{X}_i(t) = -d_iX_i(t) + \sum_{j=1}^2 A_{ij}f_j(X_j(t)) + \sum_{j=1}^2 B_{ij}f_j(X_j(t-\tau(t))) \\
\qquad\qquad + \sum_{j=1}^2 C_{ij}\int_{t-\sigma(t)}^t f_j(X_j(s))ds + U_i(t), \\
\Delta X_i(t_k) = X_i(t_k) - X_i(t_k^-) = F_{ki}\left(X_i(t_k^-) - d_i\int_{t_k-\delta}^{t_k} X_i(s)ds\right),\ k \in \mathbb{Z}^+, \\
Y_i(t) = f_i(X_i(t)),
\end{cases} \tag{7.6.25}
$$

for $i = 1, 2$, where $d_1 = d_2 = 20$,

$$A_{11} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ A_{12} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},\ A_{21} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ A_{22} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},$$

$$B_{11} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ B_{12} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},\ B_{21} = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix},\ B_{22} = \begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix},$$

$$C_{11} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ C_{12} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},\ C_{21} = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix},\ C_{22} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},$$

$$F_{k1} = \begin{bmatrix} -0.1 & -0.2 \\ 0.2 & -0.1 \end{bmatrix},\ F_{k2} = \begin{bmatrix} -0.1 & -0.1 \\ -0.1 & -0.1 \end{bmatrix},\ t_k = k,\ k \in \mathbb{Z}^+,$$

$$U_1(t) = \begin{bmatrix} -14 & 23 \\ -45 & 34 \end{bmatrix},\ U_2(t) = \begin{bmatrix} -43 & 33 \\ -23 & 13 \end{bmatrix},\ \forall t > 0,$$

$$f\left(([X]_{ab})_{1\le a,b\le n}\right) = \left(\frac{1}{1+e^{-[X]_{ab}}}\right)_{1\le a,b\le n},$$

from which we have that $l_1^f = l_2^f = \frac{1}{2}$, meaning that Assumption 7.4 is fulfilled. If the leakage delay is $\delta = 0.04$, the time-varying delays are $\tau(t) = 0.1|\cos t|$, and the distributed delays are $\sigma(t) = 0.05|\sin t|$, so $\tau = \tau' = 0.1$ and $\sigma = 0.05$, then Assumption 7.3 is also fulfilled. Assumption 7.5 is clearly fulfilled, from the definition of the network given in (7.6.25).

By taking $\gamma = 0.1$,

$$\mathcal{Q} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$\mathcal{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

$$\mathcal{R} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix},$$

and solving the LMI conditions (7.6.6) and (7.6.7) in Theorem 7.5, we obtain that neural network (7.6.25) is strictly $(\mathcal{Q}, \mathcal{S}, \mathcal{R})$-$\gamma$-dissipative, for $R_1 = \text{diag}(10.3013 I_4,\ 9.3434 I_4)$, $R_2 = \text{diag}(4.6464 I_4, 4.3934 I_4)$. (For brevity, the values of the other matrices are not given.)

Example 7.6. Now, we consider the matrix-valued neural network with two neurons in (7.6.25), where $d_1 = d_2 = 5$,

$$A_{11} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ A_{12} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},\ A_{21} = \begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix},\ A_{22} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},$$

$$B_{11} = \begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix},\ B_{12} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},\ B_{21} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix},\ B_{22} = \begin{bmatrix} 1 & 3 \\ 2 & 3 \end{bmatrix},$$

$$C_{11} = \begin{bmatrix} 1 & 1 \\ 2 & -2 \end{bmatrix},\ C_{12} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},\ C_{21} = \begin{bmatrix} 1 & -1 \\ -2 & 2 \end{bmatrix},\ C_{22} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix},$$

$$F_{k1} = \begin{bmatrix} -0.1 & -0.1 \\ 0.2 & -0.2 \end{bmatrix},\ F_{k2} = \begin{bmatrix} -0.1 & 0.1 \\ -0.1 & -0.1 \end{bmatrix},\ t_k = k,\ k \in \mathbb{Z}^+,$$

$$U_1(t) = \begin{bmatrix} 5 & -10 \\ 15 & 10 \end{bmatrix},\ U_2(t) = \begin{bmatrix} 5 & 15 \\ -10 & -15 \end{bmatrix},\ \forall t > 0,$$

$$f\left(([X]_{ab})_{1\leq a,b\leq n}\right) = \left(\frac{1}{1+e^{-[X]_{ab}}}\right)_{1\leq a,b\leq n},$$

from which we get that $l_1^f = l_2^f = \frac{1}{2}$, and so Assumption 7.4 is fulfilled. If the leakage delay is $\delta = 0.03$, the time-varying delays are $\tau(t) = 0.2|\sin t|$, and the distributed delays are $\sigma(t) = 0.1|\cos t|$, so $\tau = \tau' = 0.2$ and $\sigma = 0.1$, then Assumption 7.3 is also fulfilled. Assumption 7.5 is fulfilled by the definition of the network given in (7.6.25).

Now, taking $\gamma = 0.1$,

$$\mathcal{Q} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$\mathcal{S} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

$$\mathcal{R} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix},$$

and solving the LMI conditions (7.6.6) and (7.6.7) in Theorem 7.5, we get that neural network (7.6.25) is strictly $(\mathcal{Q}, \mathcal{S}, \mathcal{R})$-$\gamma$-dissipative for $R_1 = \text{diag}(19.3159I_4,\ 13.5703I_4)$, $R_2 = \text{diag}(6.6740I_4, 3.5330I_4)$. (For brevity, the values of the other matrices are not given.)

7.7 Lie algebra-valued neural networks

A somewhat different approach of neural networks with values in multidimensional domains, which has no direct connection with Clifford algebras, are vector-valued neural networks, see [128, 129, 130, 131]. These networks process three dimensional input vectors in two ways: one based on the vector product, which has three dimensional vectors as weights, and one which has orthogonal matrices as weights (i.e. matrices that satisfy $AA^T = A^TA = I$). This last variant was further generalized to N-dimensional vectors, and thus the N-dimensional neural networks (see [132, 133]), have N-dimensional vector inputs and outputs, but orthogonal matrix weights. Both three-dimensional and N-dimensional neural networks were used successfully in applications, the first in geometric transformations, and the second in the N-bit parity problem, which was solved using a single N-dimensional neuron.

We proposed a different generalization of real-valued neural networks in multidimensional domains, namely neural networks that have Lie algebraic inputs, outputs, weights and biases, first in the form of feedforward networks, see [146]. Because Lie algebras can have any dimension n, they also represent an alternative to the N-dimensional neural networks that we mentioned earlier. Taking into account the fact that their definition comes from geometry, and that they have numerous applications in physics and engineering (see [87, 46, 189]), and also the fact that they have been successfully used in computer vision over the last few years (for a survey, see [233], and the references thereof), we considered a promising idea to define Hopfield neural networks and bidirectional associative memories with values in Lie algebras. Lie algebra-valued Hopfield neural networks can be applied to image processing and computer vision, where the data can be treated in the form of geometric transformations. Lie algebra-valued bidirectional associative memories can be applied to store Lie-algebraic patterns and to solve difficult optimization problems defined on a Lie group or a Lie algebra.

The presentation in the following two sections in inspired from that in the author's papers [148] and [163], respectively.

7.7.1 Lie algebra-valued Hopfield neural networks

A Lie algebra is a vector space $\mathfrak{g}$ over a field F together with an operation $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ called the Lie bracket, which satisfies the following axioms:

- It is bilinear: $[ax + by, z] = a[x, z] + b[y, z]$, $[x, ay + bz] = a[x, y] + b[x, z]$, $\forall a, b \in F$, $\forall x, y \in \mathfrak{g}$.
- It is skew symmetric: $[x, x] = 0$, which implies $[x, y] = -[y, x]$, $\forall x, y \in \mathfrak{g}$.
- It satisfies the Jacobi identity: $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$, $\forall x, y, z \in \mathfrak{g}$.

Consider the vector space $\mathfrak{so}(n)$ of skew-symmetric square matrices of order n, i.e. square matrices with real components for which $A^T = -A, \forall A \in \mathfrak{so}(n)$. It is easy to verify that the operation given by

$$[A, B] := AB - BA,\ \forall A, B \in \mathfrak{so}(n),$$

satisfies the above axioms, meaning that $\mathfrak{so}(n)$ is a Lie algebra, the operation defined above being its Lie bracket. For the easiness of the exposition, we will work only with this Lie algebra, but the generalization to an arbitrary Lie algebra can be done in a simple way.

In what follows, we will define Hopfield neural networks for which the states, outputs, weights and thresholds are all from $\mathfrak{so}(n)$, which means that they are skew-symmetric square matrices. The network is described by the set of differential equations

$$\tau_i \frac{dV_i(t)}{dt} = -V_i(t) + \sum_{j=1}^{N} W_{ij}^T f(V_j(t)) W_{ij} + B_i,\ i \in \{1, \ldots, N\}, \tag{7.7.1}$$

where $\tau_i \in \mathbb{R},\ \tau_i > 0$ is the time constant of neuron i, $V_i(t) \in \mathfrak{so}(n)$ is the state of neuron i at time t, $W_{ij} \in SO(n)$ is the weight connecting neuron j to neuron i, $f : \mathfrak{so}(n) \to \mathfrak{so}(n)$ is the nonlinear Lie algebra-valued activation function, and B_i is the threshold of neuron i, $\forall i \in \{1, \ldots, N\}$. $SO(n)$ represents the Lie group associated with the Lie algebra $\mathfrak{so}(n)$, and it is known that $W^T X W \in \mathfrak{so}(n), \forall W \in SO(n), \forall X \in \mathfrak{so}(n)$, and so the above expression is well defined. $SO(n)$ is the group of orthogonal square matrices, i.e. square matrices with real components for which $A^T A = A A^T = I_n$ and $\det A = 1, \forall A \in SO(n)$. The derivative is considered to be the matrix formed by the derivatives of each component $[V_i(t)]_{ab}$ of the matrix $V_i(t)$ with respect to t:

$$\frac{dV_i(t)}{dt} := \left(\frac{d([V_i]_{ab})}{dt} \right)_{1 \leq a,b \leq n}.$$

If we denote by $X_j(t) := f(V_j(t))$ the output of neuron j, the above set of differential equations can be written as:

$$\tau_i \frac{dV_i(t)}{dt} = -V_i(t) + \sum_{j=1}^{N} W_{ij}^T X_j(t) W_{ij} + B_i,\ i \in \{1, \ldots, N\}.$$

The activation function is formed of n^2 functions $f^{ab} : \mathfrak{so}(n) \to \mathbb{R}$, $1 \leq a, b \leq n$:

$$f(V) = \left(f^{ab}(V) \right)_{1 \leq a,b \leq n}.$$

In order to study the stability of the above defined network, we need to make a series of assumptions about the activation function. First of all, we need to assume that the functions f^{ab} are continuously differentiable with respect to each $[V]_{cd}$, $\forall 1 \leq c, d \leq n$, $\forall 1 \leq a, b \leq n$, and the function f is bounded: $\exists M > 0$, $||f(V)|| \leq M$, $\forall V \in \mathfrak{so}(n)$, where $||X||$ is the Frobenius norm of matrix X, defined by $||X|| = \sqrt{\text{Tr}(X X^T)}$. Now, the $n^2 \times n^2$ Jacobian matrix of the function f can be defined as

$$\mathbf{Jac}_f(V) = \left(\frac{\partial f^{ab}(V)}{\partial [V]_{cd}} \right)_{\substack{1 \leq a,b \leq n \\ 1 \leq c,d \leq n}}.$$

Another assumption that we have to make about the activation function is that f is injective and $\mathbf{Jac}_f(V)$ is symmetric and positive definite, $\forall V \in \mathfrak{so}(n)$. This, together with the above assumption, assures the existence of the inverse function of f, $g : \mathfrak{so}(n) \to \mathfrak{so}(n)$, $g = f^{-1}$. We can thus write $g(X_i(t)) = V_i(t)$, $\forall i \in \{1, \ldots, N\}$. From the existence of g, we infer the existence of a function $G : \mathfrak{so}(n) \to \mathbb{R}$, which satisfies

$$\frac{\partial G(X)}{\partial [X]_{ab}} = g^{ab}(X),\ \forall 1 \leq a, b \leq n,$$

where $g^{ab} : \mathfrak{so}(n) \to \mathbb{R}$ are the component functions of g. The above condition can also be written in matrix form as

$$\frac{\partial G(X)}{\partial X} = g(X). \tag{7.7.2}$$

The function G is defined by

$$G(X) = \sum_{a,b=1}^{n} \int_0^{[X]_{ab}} g^{ab}(Y^{ab})dy,$$

where the matrix Y^{ab} has the following form

$$[Y^{ab}]_{cd} = \begin{cases} [X]_{cd}, & (c,d) < (a,b) \\ y, & (c,d) = (a,b) \\ 0 & (c,d) > (a,b) \end{cases}, \ \forall 1 \leq a, b \leq n.$$

For example, for 2×2 matrices, we have that

$$\begin{aligned} G(X) &= \int_0^{[X]_{11}} g^{11}\left(\begin{pmatrix} y & 0 \\ 0 & 0 \end{pmatrix}\right) dy + \int_0^{[X]_{12}} g^{12}\left(\begin{pmatrix} [X]_{11} & y \\ 0 & 0 \end{pmatrix}\right) dy \\ &+ \int_0^{[X]_{21}} g^{21}\left(\begin{pmatrix} [X]_{11} & [X]_{12} \\ y & 0 \end{pmatrix}\right) dy + \int_0^{[X]_{22}} g^{22}\left(\begin{pmatrix} [X]_{11} & [X]_{12} \\ [X]_{21} & y \end{pmatrix}\right) dy. \end{aligned}$$

The last assumption concerns the weights of the network, which must satisfy:

$$W_{ji} = W_{ij}^T, \ \forall i, j \in \{1, \ldots, N\}.$$

The network may or may not have self connections, i.e. it is not necessary to assume that $W_{ii} = O_n$, but only that $W_{ii} = W_{ii}^T, \forall i \in \{1, \ldots, N\}$, as the above assumption shows.

In this point, we can define the energy function $E : \mathfrak{so}(n)^N \to \mathbb{R}$ of the Hopfield network (7.7.1) as:

$$\begin{aligned} E(\mathbf{X}(t)) &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \text{Tr}(X_i(t)^T W_{ij}^T X_j(t) W_{ij}) \\ &+ \sum_{i=1}^{N} G(X_i(t)) - \sum_{i=1}^{N} \text{Tr}(B_i^T X_i(t)), \end{aligned} \tag{7.7.3}$$

where $\text{Tr}(X)$ represents the trace of matrix X.

We will show that E is indeed an energy function, i.e. the derivative of the function E along the trajectories of network (7.7.1), denoted by $\frac{dE(\mathbf{X}(t))}{dt}$, satisfies the condition $\frac{dE(\mathbf{X}(t))}{dt} \leq 0$ and $\frac{dE(\mathbf{X}(t))}{dt} = 0 \Leftrightarrow \frac{dX_i(t)}{dt} = 0, \forall i \in \{1, \ldots, N\}$.

For this, we start by applying the chain rule:

$$\begin{aligned} \frac{dE(\mathbf{X}(t))}{dt} &= \sum_{i=1}^{N} \sum_{a,b=1}^{n} \frac{\partial E(\mathbf{X}(t))}{\partial [X_i(t)]_{ab}} \frac{d[X_i(t)]_{ab}}{dt} \\ &= \sum_{i=1}^{N} \text{Tr}\left(\left(\frac{\partial E(\mathbf{X}(t))}{\partial X_i(t)} \right)^T \frac{dX_i(t)}{dt} \right), \end{aligned} \tag{7.7.4}$$

where by $\frac{\partial E(\mathbf{X}(t))}{\partial [X_i(t)]_{ab}}$ we denoted the partial derivative of the function E with respect to each component $[X_i(t)]_{ab}$ of the matrices $X_i(t)$, $\forall 1 \leq a, b \leq n$, $\forall i \in \{1, \ldots, N\}$. For the partial derivative $\frac{\partial E(\mathbf{X}(t))}{\partial X_i(t)} = \left(\frac{\partial E(\mathbf{X}(t))}{\partial [X_i(t)]_{ab}}\right)_{1 \leq a,b \leq n}$, we have from (7.7.3) that

$$\begin{aligned}
\frac{\partial E(\mathbf{X}(t))}{\partial X_i(t)} &= -\sum_{j=1}^{N} W_{ij}^T X_j(t) W_{ij} + g(X_i(t)) - B_i \\
&= -\left(\sum_{j=1}^{N} W_{ij}^T X_j(t) W_{ij} - V_i(t) + B_i\right) \\
&= -\tau_i \frac{dV_i(t)}{dt}, \forall i \in \{1, \ldots, N\},
\end{aligned}$$

where we used the fact that

$$\frac{d\text{Tr}(X^T A)}{dX} = \frac{d\text{Tr}(AX^T)}{dX} = A,$$

relation (7.7.2), the assumption $W_{ji} = W_{ij}^T$, and also the set of equations given by (7.7.1). Now, equation (7.7.4) becomes:

$$\begin{aligned}
\frac{dE(\mathbf{X}(t))}{dt} &= \sum_{i=1}^{N} \text{Tr}\left(\left(-\tau_i \frac{dV_i(t)}{dt}\right)^T \frac{dX_i(t)}{dt}\right) \\
&= -\sum_{i=1}^{N} \tau_i \left[\text{vec}\left(\frac{dV_i(t)}{dt}\right)\right]^T \text{vec}\left(\frac{dX_i(t)}{dt}\right) \\
&= -\sum_{i=1}^{N} \left\{\tau_i \left[\text{vec}\left(\frac{dX_i(t)}{dt}\right)\right]^T [\mathbf{Jac}_g(X_i(t))]^T \text{vec}\left(\frac{dX_i(t)}{dt}\right)\right\} \\
&\leq 0, \qquad (7.7.5)
\end{aligned}$$

where we denoted by vec (X) the vectorization of matrix X. We also used the identity

$$\text{Tr}(A^T B) = \text{vec}(A)^T \text{vec}(B), \forall A, B \in \mathfrak{so}(n),$$

and, from $g(X_i(t)) = V_i(t)$, we obtained that

$$\text{vec}\left(\frac{dg(X_i(t))}{dt}\right) = \mathbf{Jac}_g(X_i(t)) \text{vec}\left(\frac{dX_i(t)}{dt}\right),$$

$\forall i \in \{1, \ldots, N\}$. Because $\mathbf{Jac}_f(V)$ is symmetric and positive definite, we deduce that $\mathbf{Jac}_g(X)$ is also symmetric and positive definite, and thus

$$\left[\text{vec}\left(\frac{dX_i(t)}{dt}\right)\right]^T [\mathbf{Jac}_g(X_i(t))]^T \text{vec}\left(\frac{dX_i(t)}{dt}\right) \geq 0,$$

$\forall i \in \{1, \ldots, N\}$, which allowed us to write the last inequality in relation (7.7.5). Equality is attained when

$$\frac{dE(\mathbf{X}(t))}{dt} = 0 \Leftrightarrow \text{vec}\left(\frac{dX_i(t)}{dt}\right) = 0 \Leftrightarrow \frac{dX_i(t)}{dt} = 0,$$

$\forall i \in \{1, \ldots, N\}$, thus ending the proof that E is indeed an energy function for the network (7.7.1).

We now give two examples of activation functions, inspired by the ones used in real-valued and complex-valued neural networks:

$$f(V) = \frac{V}{1+||V||}, \forall V \in \mathfrak{so}(n),$$

$$f\left(([V]_{ab})_{1\leq a,b\leq n}\right) = (\tanh[V]_{ab})_{1\leq a,b\leq n}, \forall V \in \mathfrak{so}(n).$$

The first one corresponds to the fully complex activation functions, and the second one corresponds to the split complex activation functions from the complex-valued domain. It can be easily verified that these functions satisfy the above assumptions.

7.7.2 Lie algebra-valued bidirectional associative memories

Now, we will define bidirectional associative memories for which the states, outputs, and thresholds are all from $\mathfrak{so}(n)$, which means that they are skew-symmetric square matrices. The network is described by the set of differential equations

$$\begin{cases} \tau_i \dfrac{dX_i(t)}{dt} = -X_i(t) + \displaystyle\sum_{j=1}^{P} W_{ij}^T f(Y_j(t)) W_{ij} + A_i, \quad \forall i \in \{1,\dots,N\}, \\ \upsilon_j \dfrac{dY_j(t)}{dt} = -Y_j(t) + \displaystyle\sum_{i=1}^{N} W_{ji}^T f(X_i(t)) W_{ji} + B_j, \quad \forall j \in \{1,\dots,P\} \end{cases} \tag{7.7.6}$$

where $\tau_i \in \mathbb{R},\ \tau_i > 0$ is the time constant of neuron X_i, $\upsilon_j \in \mathbb{R},\ \upsilon_j > 0$ is the time constant of neuron Y_j, $X_i(t) \in \mathfrak{so}(n)$ is the state of neuron X_i at time t, $Y_j(t) \in \mathfrak{so}(n)$ is the state of neuron Y_j at time t, $W_{ij} \in SO(n)$ is the weight connecting neuron X_i to neuron Y_j, $f : \mathfrak{so}(n) \to \mathfrak{so}(n)$ is the nonlinear Lie algebra-valued activation function, A_i is the threshold of neuron X_i, and B_j is the threshold of neuron Y_j, $\forall i \in \{1,\dots,N\}$, $\forall j \in \{1,\dots,P\}$. If we denote by $U_i(t) := f(X_i(t))$ the output of neuron X_i and by $V_j(t) := f(Y_j(t))$ the output of neuron Y_j, the above set of differential equations can be written as:

$$\begin{cases} \tau_i \dfrac{dX_i(t)}{dt} = -X_i(t) + \displaystyle\sum_{j=1}^{P} W_{ij}^T V_j(t) W_{ij} + A_i, \quad \forall i \in \{1,\dots,N\}, \\ \upsilon_j \dfrac{dY_j(t)}{dt} = -Y_j(t) + \displaystyle\sum_{i=1}^{N} W_{ji}^T U_i(t) W_{ji} + B_j, \quad \forall j \in \{1,\dots,P\}. \end{cases}$$

Just like in Section 7.7.1, in order to study the stability of the above defined network, we need to make a series of assumptions about the activation function. The assumptions will be exactly the same as the ones in Section 7.7.1.

Having made these assumptions, we can define the energy function $E : \mathfrak{so}(n)^{N+P} \to \mathbb{R}$ of the bidirectional associative memory (7.7.6) as:

$$\begin{aligned} E(\mathbf{U}(t),\mathbf{V}(t)) \;=\; & -\sum_{i=1}^{N}\sum_{j=1}^{P} \mathrm{Tr}(U_i(t)^T W_{ij}^T V_j(t) W_{ij}) \\ & + \sum_{i=1}^{N} G(U_i(t)) - \sum_{i=1}^{N} \mathrm{Tr}(A_i^T U_i(t)) \\ & + \sum_{j=1}^{P} G(V_j(t)) - \sum_{j=1}^{P} \mathrm{Tr}(B_j^T V_j(t)). \end{aligned} \tag{7.7.7}$$

A function E is an energy function for the network (7.7.6) if the derivative of E along the trajectories of network, denoted by $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt}$, satisfies the condition $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} \leq 0$ and $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} = 0 \Leftrightarrow \frac{dU_i(t)}{dt} = \frac{dV_j(t)}{dt} = 0, \forall i \in \{1,\ldots,N\}, \forall j \in \{1,\ldots,P\}$. We will show that the function E defined in (7.7.7) is indeed an energy function for the network (7.7.6).

For this, we start by applying the chain rule:

$$\begin{aligned}
\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} &= \sum_{i=1}^{N}\sum_{a,b=1}^{n}\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [U_i(t)]_{ab}}\frac{d[U_i(t)]_{ab}}{dt} \\
&\quad + \sum_{j=1}^{P}\sum_{a,b=1}^{n}\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [V_j(t)]_{ab}}\frac{d[V_j(t)]_{ab}}{dt} \\
&= \sum_{i=1}^{N}\mathrm{Tr}\left(\left(\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial U_i(t)}\right)^T\frac{dU_i(t)}{dt}\right) \\
&\quad + \sum_{j=1}^{P}\mathrm{Tr}\left(\left(\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial V_j(t)}\right)^T\frac{dV_j(t)}{dt}\right), \qquad (7.7.8)
\end{aligned}$$

where by $\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [U_i(t)]_{ab}}$ we denoted the partial derivative of the function E with respect to each element $[U_i(t)]_{ab}$ of the matrices $U_i(t)$, $\forall 1 \leq a,b \leq n$, $\forall i \in \{1,\ldots,N\}$, and analogously for $\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [V_j(t)]_{ab}}$.

For the partial derivatives

$$\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial U_i(t)} = \left(\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [U_i(t)]_{ab}}\right)_{1\leq a,b\leq n}$$

and

$$\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial V_j(t)} = \left(\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial [V_j(t)]_{ab}}\right)_{1\leq a,b\leq n},$$

we have from (7.7.7) that

$$\begin{aligned}
\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial U_i(t)} &= -\sum_{j=1}^{P}W_{ij}^T V_j(t) W_{ij} + g(U_i(t)) - A_i \\
&= -\left(\sum_{j=1}^{P}W_{ij}^T V_j(t) W_{ij} - X_i(t) + A_i\right) \\
&= -\tau_i\frac{dX_i(t)}{dt},\ \forall i \in \{1,\ldots,N\},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E(\mathbf{U}(t),\mathbf{V}(t))}{\partial V_j(t)} &= -\sum_{i=1}^{N}W_{ji}^T U_i(t) W_{ji} + g(V_j(t)) - B_j \\
&= -\left(\sum_{i=1}^{N}W_{ji}^T U_j(t) W_{ji} - Y_j(t) + B_j\right) \\
&= -\upsilon_j\frac{dY_j(t)}{dt},\ \forall j \in \{1,\ldots,P\},
\end{aligned}$$

where we used the facts that

$$\frac{d\text{Tr}(X^TA)}{dX} = A,$$
$$\frac{d\text{Tr}(AXB)}{dX} = A^TB^T,$$

relation (7.7.2), the assumption $W_{ji} = W_{ij}^T$, and also the set of equations given by (7.7.6). Now, equation (7.7.8) becomes:

$$\begin{aligned}
\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} &= \sum_{i=1}^{N}\text{Tr}\left(\left(-\tau_i\frac{dX_i(t)}{dt}\right)^T\frac{dU_i(t)}{dt}\right) \\
&\quad+\sum_{j=1}^{P}\text{Tr}\left(\left(-\upsilon_j\frac{dY_j(t)}{dt}\right)^T\frac{dV_j(t)}{dt}\right) \\
&= -\sum_{i=1}^{N}\tau_i\left[\text{vec}\left(\frac{dX_i(t)}{dt}\right)\right]^T\text{vec}\left(\frac{dU_i(t)}{dt}\right) \\
&\quad-\sum_{j=1}^{P}\upsilon_j\left[\text{vec}\left(\frac{dY_j(t)}{dt}\right)\right]^T\text{vec}\left(\frac{dV_j(t)}{dt}\right) \\
&= -\sum_{i=1}^{N}\left\{\tau_i\left[\text{vec}\left(\frac{dU_i(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(U_i(t))]^T\text{vec}\left(\frac{dU_i(t)}{dt}\right)\right\} \\
&\quad-\sum_{j=1}^{P}\left\{\upsilon_j\left[\text{vec}\left(\frac{dV_j(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(V_j(t))]^T\text{vec}\left(\frac{dV_j(t)}{dt}\right)\right\} \\
&\leq 0, \qquad (7.7.9)
\end{aligned}$$

We also used the identity

$$\text{Tr}(A^TB) = \text{vec}(A)^T\text{vec}(B),\ \forall A,B\in\mathfrak{so}(n),$$

and, from $g(U_i(t)) = X_i(t)$ and $g(V_j(t)) = Y_j(t)$, we obtained that

$$\text{vec}\left(\frac{dg(U_i(t))}{dt}\right) = \mathbf{Jac}_g(U_i(t))\text{vec}\left(\frac{dU_i(t)}{dt}\right),$$
$$\text{vec}\left(\frac{dg(V_j(t))}{dt}\right) = \mathbf{Jac}_g(V_j(t))\text{vec}\left(\frac{dV_j(t)}{dt}\right),$$

$\forall i\in\{1,\ldots,N\}$, $\forall j\in\{1,\ldots,P\}$. Because $\mathbf{Jac}_f(X)$ is symmetric and positive definite, we deduce that $\mathbf{Jac}_g(U)$ is also symmetric and positive definite, and thus

$$\left[\text{vec}\left(\frac{dU_i(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(U_i(t))]^T\text{vec}\left(\frac{dU_i(t)}{dt}\right)\geq 0,$$
$$\left[\text{vec}\left(\frac{dV_j(t)}{dt}\right)\right]^T[\mathbf{Jac}_g(V_j(t))]^T\text{vec}\left(\frac{dV_j(t)}{dt}\right)\geq 0,$$

$\forall i\in\{1,\ldots,N\}$, $\forall j\in\{1,\ldots,P\}$, which allowed us to write the last inequality in relation (7.7.9). Equality is attained when $\frac{dE(\mathbf{U}(t),\mathbf{V}(t))}{dt} = 0 \Leftrightarrow \text{vec}\left(\frac{dU_i(t)}{dt}\right) = \text{vec}\left(\frac{dV_j(t)}{dt}\right) = 0 \Leftrightarrow \frac{dU_i(t)}{dt} = \frac{dV_j(t)}{dt} = 0$, $\forall i\in\{1,\ldots,N\}$, $\forall j\in\{1,\ldots,P\}$, thus ending the proof that E is indeed an energy function for the network (7.7.6).

Chapter 8

Scientific, professional, and academic development plan

8.1 Scientific and professional development plan

The scientific career of the author will continue on the directions opened with the PhD thesis, and continued with the research presented in the current thesis.

The domain of **complex-valued neural networks** saw an increase in interest over the last few years, even more so that when the author began his research in the field. The deep learning paradigm is very popular nowadays, and more algorithms belonging to this paradigm can be extended from the real-valued domain to the complex-valued domain, especially in areas where the data is inherently complex-valued, like signal processing, or image processing in the Fourier domain.

On the other hand, a very active area of research is that dedicated to the study of dynamic properties of recurrent neural networks, specifically Hopfield networks and bidirectional associative memories, with different types of delays, with reaction-diffusion terms, with Markovian jumping parameters, etc. The dynamic properties can be stability, synchronization, dissipativity, passivity, periodicity, etc.

Then comes the domain of **quaternion-valued neural networks**, which also saw an increased interest from researchers in the last few years. These types of networks were used for image processing and recognition with deep learning-type models. The very recent papers leave many open problems and possible future work directions. The experience of the author in working with quaternions, and in adapting neural network models to the quaternion domain is very significant, as there are not many researchers that have this type of experience.

As for complex-valued neural networks, there is a growing corpus of papers dedicated to studying the dynamic properties of Hopfield neural networks and bidirectional associative memories with values in the quaternion domain. The author already has one paper dedicated to this subject, and many ideas of extending results from the real- and complex-valued domains exist, and can be explored in future works.

The **octonion-valued neural networks** were introduced by the author, for the first time in the literature. Some applications are beginning to emerge, which gives hope that in the future, the domain of octonion-valued neural networks can develop similarly with the domains of complex- and quaternion-valued neural networks. Recurrent neural networks with octonion values can also be formulated, and the same dynamic properties can be studied regarding these networks.

The domain of **matrix-valued neural networks** has also been introduced by the author, for

the first time in the literature. The relation with the domain of tensor networks must be further explored, as the two domains are similar, and ideas from the tensor networks domain can be also used in the domain of matrix-valued neural networks for processing higher-dimensional data, and to solve real-world problems, like the ones solved by deep learning models. Again, because they generalize complex-, quaternion-, and octonion-valued neural networks, the same dynamic properties of stability, synchronization, dissipativity, passivity, periodicity, etc. can be studied for models with different types of delays, with reaction-diffusion terms, with Markovian jumping parameters, etc.

On a rather different note, starting very recently, the interest of the author has shifted more to the direction of mainstream **computer vision** done using **deep learning** models. The author has began exploring with students developing their Bachelor's and Master's degrees, different areas related to 3D computer vision, with applications in the domain of **autonomous driving**.

Currently, autonomous driving is one of the most interesting and important applications of artificial intelligence, and it is expected that in the not too distant future, companies will be interested in developing artificial intelligence systems for autonomous driving, but also for meeting the future NCAP vehicle safety requirements. Due to the fact that the automotive industry is very present in Timișoara and Timiș County, it is expected that partnerships will be created between academia and industry, and students who will work in this field for the preparation of their Bachelor's/Master's/Doctoral theses, will be able to be employed as experts by these companies.

As such, developing knowledge and doing research in this domain is of high interest for the automotive industry.

Vehicle environment recognition can be done using cameras, with the specific problems: lane detection, traffic lights detection, semantic segmentation of exterior scenes, depth perception in exterior scenes, 3D traffic participants' detection and tracking (cars, pedestrians, etc.), traffic participants' motion prediction, etc., or using radars/lidars, with the specific problems: semantic segmentation of radar/lidar point clouds, 3D traffic participants' detection and tracking using only radar/lidar point clouds, etc. All these areas were explored by the students of the author, and papers are in development with the most important novel results that were obtain in these ares.

Store theft detection, self-checkout systems, and the complete elimination of checkout in 'grab and go' stores are, in addition to autonomous driving, the other interesting and important current applications of artificial intelligence in the **retail industry**, with a very high potential impact.

The specific problems in this area are: human action recognition, 2D and 3D human pose estimation, 3D hand pose estimation, human-object interaction recognition, 6D object pose estimation, person re-identification in images from multiple cameras, 3D human tracking, semantic segmentation of interior scenes, depth perception in interior scenes, etc. Novel results in these directions were also obtained by the author in collaboration with Bachelor's/Master's/Doctoral students, and the papers presenting them are currently under development.

A very interesting domain, which has captured the interest of the author long ago is that of **reinforcement learning**. This is the subdomain of machine learning which most closely resembles the desiderata of artificial general intelligence, i.e., intelligent agents that interact with an environment, receive feedback from the environment, and take actions based on maximizing a certain reward, which is very similar to the way humans begin to perceive the world as children. Two directions can be followed in this domain: collaborating with colleagues from the Computers and Information Technology Department for applications in robotics, and applying reinforcement learning to the domain of autonomous driving.

The goal of the author in the before-mentioned domains is to build a strong **research team** in the **domain** of **deep learning**, which is extremely popular nowadays in both academia and industry, which will increase the visibility of the Politehnica University of Timișoara among very prestigious and important universities around the world, which already have strong research teams. This goal will certainly be facilitated by the existence of the Master of Machine Learning at the Politehnica University of Timișoara, and the students who graduate could continue their studies at the PhD program, under the supervision of the author, thus gradually building a research team in deep learning.

8.2 Academic development plan

The academic activity of the author started with introducing the *Modeling and Simulation* course at the 4th year Bachelor's Computers – in English. Then came the modernizing of the laboratory at the *Modeling and Simulation* course for both the Computers – in English and Computers – in Romanian specializations. The same modernization followed for the laboratory at the *Graphics and Human-Computer Interaction* course for both the Computers – in English and Computers – in Romanian specializations.

After that, the author completely changed the *Image Processing and Recognition* course and laboratory, which was initially offered as optional to the 1st year Master's of Computer Engineering students. Then, after that master was discontinued, the *Image Processing and Recognition* course replaced the *Modeling and Simulation* course at the 4th year Bachelor's Computers – in English, which was an idea of the author to familiarize the students with the basics of image processing and recognition at the Bachelor's studies.

The most important course of the author is the *Computer Assisted Mathematics* course at the 1st year Bachelor's Computers – in Romanian, which is in the core curriculum of the specialization, and is attended by all the students in the specialization. The course and laboratory were developed by taking into account the latest trends in teaching numerical analysis to computer science and engineering students, with applications specific to their fields. The course and laboratory were very well received by the students, as they had the opportunity to study MATLAB for the first time, which is necessary for late courses involving signal processing.

At the newly introduced Master's of Machine Learning, the author proposed 4 courses: *Deep Learning*, *Reinforcement Learning*, *Autonomous Driving* and the general research course *Research Topics in Machine Learning*. All the courses were developed from scratch by the author, and are for the first time offered to students of the Politehnica University of Timișoara. The aim is to build knowledge in the respective domains for the students in Timișoara at the highest possible international level, thus preparing them for doing competitive research in these domain and working in the industry in these areas. It is expected that the Master's of Machine Learning, for which the author is the program coordinator, will spark the interest of students in the domain of machine learning, and will facilitate the building of a strong deep learning research team at the Politehnica University of Timișoara.

This Master's is also expected to spark a more close relation with the industry players in Timișoara and the Timiș County. Bachelor's and Master's and even Doctoral theses can be developed together with and at the proposal of different companies in the area.

The main academic goal of the author is to continue to develop the courses that he teaches, maintaining them at a competitive level internationally, taking also into account the feedback from the students. Especially for the courses at the Master's of Machine Learning, every year or every few years, the courses must be changed more or less radically, because the respective

domains evolve so fast, that what was state-of-the-art last year or a few years ago, rapidly becomes obsolete, and has to be changed to reflect the most important trends in these domains.

The increasing number of students that seek guidance from the author to develop their Bachelor's and Master's theses shows the increased interest of students in the popular deep learning domain, and gives hope that the goal of building a strong deep learning community in Timișoara is not such a far fetched one, and could put Timișoara and the Politehnica University on the international map of deep learning, both in academic research and in industrial development, and could attract companies to come to Timișoara to establish offices here.

8.3 Research infrastructure

The author contributed in attracting two **sponsorships** from the company VISMA Software for the upgrade of the **Artificial Intelligence** and **Computer Graphics laboratories**, in which the laboratories of the courses taught by the author take place.

The upgrade of the Artificial Intelligence laboratory started with a new, simple, and minimalist room design and furniture, with Scandinavian influences. New equipment has also been provided, including **18** powerful **workstations** with Intel® Core™ i5 processors and video-projection system. The Artificial Intelligence laboratory was ready to receive students with the new look in September 2016.

The following year, in September 2017, the upgraded Computer Graphics laboratory was ready to receive students. The same type of room design and furniture were used for this laboratory, also. The Computer Graphics laboratory was equipped with **18** newly acquired **workstations**, having Intel® Core™ i5 processors.

The preoccupation for attracting these sponsorships proves the author's willingness to provide students with the best conditions to make the first steps in the domain of Artificial Intelligence, and also to lay the foundation of the deep learning research team, the long-standing ambition of the author.

The deep learning domain is a software domain by excellence, and all the needed software tools in order to conduct high-quality research in the field are **freely** available. The only other resource that is needed to conduct research in the domain is computational power, especially represented by high-performance GPUs. In this direction, the author has a constant preoccupation to increase the computational power available in the Department of Computers and Information Technology, and especially to increase the number of available GPUs for doing research in the deep learning domain. This preoccupation materialized in that the author used the financing in the PCD-TC-2017 project to acquire **two high-performance computers**, each having Intel® Core™ i9-7920X X-series Processor, 32 GB of RAM, 1 TB SSD, and 1600 W PSU. Between them, they share **3** NVIDIA GTX 1080Ti and **4** NVIDIA RTX 2080Ti **GPUs**, which were acquired from the project and also from the **ISI grants** of the author, which are provided by the Politehnica University of Timişoara for each ISI journal article written by the author. Currently, there are available funds for buying at least 2 new NVIDIA RTX 3090 GPUs, with plans of increasing this number in the future. The ISI grants will be the most important source of financing the computational infrastructure needed for research, and the author has full commitment to receive as many such grants as possible in the future. The future doctoral students will also receive a **desk** with a personal computer in the office of the author, which will motivate them to be in the same physical space, and will drive the collaboration between them, leading to the creation of a strong and united deep learning research team.

There is also a long-standing commitment of the author to search for **financing opportunities** in the form of National Research Grants, and also in the form of research grants provided

by the companies in Timişoara and the Timiş county interested in the machine learning domain. These will offer the necessary funds for further increasing the computational power available, and also for augmenting the scholarships of the doctoral students provided by the Politehnica University of Timişoara.

Bibliography

[1] N.N. Aizenberg, Y.L. Ivaskiv, and D.A. Pospelov. A certain generalization of threshould functions. *Dokrady Akademii Nauk SSSR*, 196:1287 – 1290, 1971.

[2] M.F. Amin, M. Amin, A.Y.H. Al-Nuaimi, and K. Murase. Wirtinger calculus based gradient descent and levenberg-marquardt learning algorithms in complex-valued neural networks. In B.-L. Lu, L. Zhang, and J. Kwok, editors, *Neural Information Processing*, volume 7062 of *Lecture Notes in Computer Science*, pages 550 – 559. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-24955-6_66.

[3] E. Angiuli, F. Del Frate, B. Polsinelli, and D. Solimini. Towards complex-valued neural algorithms for forest parameters estimation from PolInSAR data. In *IEEE International Geoscience and Remote Sensing Symposium, IGARSS*, volume II, pages 641 – 644. IEEE, July 2008. doi: 10.1109/IGARSS.2008.4779074.

[4] P. Arena, L. Fortuna, L. Occhipinti, and M.G. Xibilia. Neural networks for quaternion-valued function approximation. In *International Symposium on Circuits and Systems (ISCAS)*, volume 6, pages 307 – 310. IEEE, 1994. doi: 10.1109/ISCAS.1994.409587.

[5] P. Arena, S. Baglio, L. Fortuna, and M.G. Xibilia. Chaotic time series prediction via quaternionic multilayer perceptrons. In *International Conference on Systems, Man and Cybernetics*, volume 2, pages 1790 – 1794. IEEE, 1995. doi: 10.1109/ICSMC.1995.538035.

[6] P. Arena, L. Fortuna, G. Muscato, and M.G. Xibilia. Multilayer perceptrons to approximate quaternion valued functions. *Neural Networks*, 10(2):335 – 342, March 1997. doi: 10.1016/S0893-6080(96)00048-2.

[7] P. Arena, L. Fortuna, G. Muscato, and M.G. Xibilia. *Neural Networks in Multidimensional Domains Fundamentals and New Trends in Modelling and Control*, volume 234 of *Lecture Notes in Control and Information Sciences*. Springer London, 1998. doi: 10.1007/BFb0047683.

[8] M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. In *International Conference on Learning Representations*, 2016.

[9] D.D. Bainov and P.S. Simeonov. *Impulsive differential equations*. World Scientific, Singapore, 1995.

[10] P. Baldi and Z. Lu. Complex-valued autoencoders. *Neural Networks*, 33:136 – 147, 2012. doi: doi:10.1016/j.neunet.2012.04.011.

[11] E. Barnard. Optimization for training neural nets. *IEEE Transactions on Neural Networks*, 3(2):232 – 240, March 1992. doi: 10.1109/72.125864.

[12] R. Battiti. First and second-order methods for learning between steepest descent and newton's method. *Neural Computation*, 4(2):141 – 166, March 1992. doi: 10.1162/neco.1992.4.2.141.

[13] S. Bauer and F.P. Leon. Hyperspectral fluorescence data fusion using quaternion and octonion phase. In *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, sep 2016. doi: 10.1109/mfi.2016.7849555.

[14] E.M.L. Beale. A derivation of conjugate gradients. In F. A. Lootsma, editor, *Numerical Methods for Nonlinear Optimization*, pages 39–43. Academic Press, London, 1972.

[15] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1 – 127, 2009. doi: 10.1561/2200000006.

[16] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS 2006*, pages 153–160, 2006.

[17] Y. Bengio, Y. LeCun, and G. Hinton. Deep learning. *Nature*, 521:436 – 444, 2015. doi: doi:10.1038/nature14539.

[18] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[19] Ł. Błaszczyk and K.'M. Snopek. Octonion fourier transform of real-valued functions of three variables - selected properties and examples. *Signal Processing*, 136:29–37, jul 2017. doi: 10.1016/j.sigpro.2016.11.021.

[20] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872 – 1886, 2013. doi: 10.1109/TPAMI.2012.230.

[21] S. Buchholz and N. Le Bihan. Optimal separation of polarized signals by quaternionic neural networks. In *2006 14th European Signal Processing Conference*, pages 1–5. IEEE, 2006.

[22] S. Buchholz and N. Le Bihan. Polarized signal classification by complex and quaternionic multi-layer perceptrons. *International Journal of Neural Systems*, 18(2):75 – 85, April 2008. doi: 10.1142/S0129065708001403.

[23] S. Buchholz and G. Sommer. Quaternionic spinor MLP. In *European Symposium on Artificial Neural Networks*, pages 377 – 382, April 2000.

[24] S. Buchholz and G. Sommer. On Clifford neurons and Clifford multi-layer perceptrons. *Neural Networks*, 21(7):925 – 935, 2008. doi: 10.1016/j.neunet.2008.03.004.

[25] B. C. Chanyal, P. S. Bisht, and O. P. S. Negi. Generalized octonion electrodynamics. *International Journal of Theoretical Physics*, 49(6):1333–1343, apr 2010. doi: 10.1007/s10773-010-0314-5.

[26] B. C. Chanyal, P. S. Bisht, and O. P. S. Negi. Generalized split-octonion electrodynamics. *International Journal of Theoretical Physics*, 50(6):1919–1926, feb 2011. doi: 10.1007/s10773-011-0706-1.

[27] B. C. Chanyal, P. S. Bisht, Tianjun Li, and O. P. S. Negi. Octonion quantum chromodynamics. *International Journal of Theoretical Physics*, 51(11):3410–3422, jun 2012. doi: 10.1007/s10773-012-1222-7.

[28] B.C. Chanyal. Octonion massive electrodynamics. *General Relativity and Gravitation*, 46(1), dec 2013. doi: 10.1007/s10714-013-1646-2.

[29] C. Charalambous. Conjugate gradient algorithm for efficient training of artificial neural networks. *IEE Proceedings G Circuits, Devices and Systems*, 139(3):301 – 310, June 1992.

[30] A. Chaturvedi, R. Sharma, D. Wadekar, A. Bhandwalkar, and S. Shitole. Adaptive parametric estimator for complex valued images. In *International Conference on Technologies for Sustainable Development (ICTSD)*, 2015. doi: 10.1109/ICTSD.2015.7095865.

[31] B. Che Ujang, C.C. Took, and D.P. Mandic. Split quaternion nonlinear adaptive filtering. *Neural Networks*, 23(3):426 – 434, April 2010. doi: 10.1016/j.neunet.2009.10.006.

[32] B. Che Ujang, C.C. Took, and D.P. Mandic. Quaternion-valued nonlinear adaptive filtering. *IEEE Transactions on Neural Networks*, 22(8):1193 – 1206, August 2011. doi: 10.1109/TNN.2011.2157358.

[33] B. Che Ujang, C.C. Took, and D.P. Mandic. On quaternion analyticity: Enabling quaternion-valued nonlinear adaptive filtering. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2117 – 2120. IEEE, March 2012. doi: 10.1109/ICASSP.2012.6288329.

[34] Tianping Chen and Lili Wang. Global μ-stability of delayed neural networks with unbounded time-varying delays. *IEEE Transactions on Neural Networks*, 18(6):1836–1840, nov 2007. doi: 10.1109/TNN.2007.902716. URL `http://dx.doi.org/10.1109/TNN.2007.902716`.

[35] X. Chen, Q. Song, and Z. Li. Design and analysis of quaternion-valued neural networks for associative memories. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(12):2305–2314, 2018. doi: 10.1109/TSMC.2017.2717866.

[36] Xiaofeng Chen and Qiankun Song. Global stability of complex-valued neural networks with both leakage time delay and discrete time delay on time scales. *Neurocomputing*, 121:254–264, dec 2013. doi: 10.1016/j.neucom.2013.04.040.

[37] Xiaofeng Chen and Qiankun Song. Global stability of complex-valued neural networks with both leakage time delay and discrete time delay on time scales. *Neurocomputing*, 121:254–264, dec 2013. doi: 10.1016/j.neucom.2013.04.040. URL `http://dx.doi.org/10.1016/j.neucom.2013.04.040`.

[38] Xiaofeng Chen, Qiankun Song, Xiaohui Liu, and Zhenjiang Zhao. Global μ-stability of complex-valued neural networks with unbounded time-varying delays. *Abstract and Applied Analysis*, 2014:1–9, 2014. doi: 10.1155/2014/263847.

[39] Xiaofeng Chen, Qiankun Song, Yurong Liu, and Zhenjiang Zhao. Global μ-stability of impulsive complex-valued neural networks with leakage delay and mixed delays. *Abstract and Applied Analysis*, 2014:1–14, 2014. doi: 10.1155/2014/397532.

[40] Xiaofeng Chen, Qiankun Song, Yurong Liu, and Zhenjiang Zhao. Global μ-stability of impulsive complex-valued neural networks with leakage delay and mixed delays. *Abstract and Applied Analysis*, 2014:1–14, 2014. doi: 10.1155/2014/397532. URL `http://dx.doi.org/10.1155/2014/397532`.

[41] C.-Y. Cheng and Z. Huang. Non-typical multistability in neural networks with distributed delays. *Neurocomputing*, 121:207–217, 2013. doi: 10.1016/j.neucom.2013.04.022.

[42] L.O. Chua and L. Yang. Cellular neural networks: Theory. *IEEE Transactions on Circuits and Systems*, 35(10):1257–1272, 1988. doi: 10.1109/31.7600. URL `http://dx.doi.org/10.1109/31.7600`.

[43] L.O. Chua and L. Yang. Cellular neural networks: Applications. *IEEE Transactions on Circuits and Systems*, 35(10):1273–1290, 1988. doi: 10.1109/31.7601. URL `http://dx.doi.org/10.1109/31.7601`.

[44] Michael A. Cohen and Stephen Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):815–826, sep 1983. doi: 10.1109/TSMC.1983.6313075. URL `http://dx.doi.org/10.1109/TSMC.1983.6313075`.

[45] J. M. Cushing. *Integrodifferential equations and delay models in population dynamics*, volume 20. Springer Science & Business Media, 2013.

[46] E.A. de Kerf, G.G.A. Bauerle, and A.P.E. ten Kroode. *Lie Algebras: Finite and Infinite Dimensional Lie Algebras and Applications in Physics*. North Holland, 1997.

[47] S. Demir. Hyperbolic octonion formulation of gravitational field equations. *International Journal of Theoretical Physics*, 52(1):105–116, aug 2012. doi: 10.1007/s10773-012-1307-3.

[48] S. Demir and M. Tanişli. Hyperbolic octonion formulation of the fluid maxwell equations. *Journal of the Korean Physical Society*, 68(5):616–623, mar 2016. doi: 10.3938/jkps.68.616.

[49] T. Dray and C.A. Manogue. *The Geometry of the Octonions*. World Scientific, 2015. doi: 10.1142/8456.

[50] Y. Du and R. Xu. Multistability and multiperiodicity for a class of cohen–grossberg BAM neural networks with discontinuous activation functions and time delays. *Neural Processing Letters*, 42(2):417–435, 2015. doi: 10.1007/s11063-014-9364-7.

[51] P. Eichel and R.W. Ives. Compression of complex-valued SAR images. *IEEE Transactions on Image Processing*, 8(10):1483 – 1487, 1999. doi: 10.1109/83.791978.

[52] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? In *Journal of Machine Learning Research*, volume 11, pages 625–660, 2010.

[53] S.E. Fahlman. An empirical study of learning speed in backpropagation networks. Technical Report 1800, Carnegie Mellon University, January 1988. URL `http://repository.cmu.edu/compsci/1800`.

[54] R. Fletcher and M.J.D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163 – 168, August 1963. doi: 10.1093/comjnl/6.2.163.

[55] M. Forti and A. Tesi. New conditions for global stability of neural networks with application to linear and quadratic programming problems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 42(7):354 – 366, 1995. doi: 10.1109/81.401145.

[56] H.-Y. Gao and K.-M. Lam. From quaternion to octonion: Feature-based image saliency detection. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, may 2014. doi: 10.1109/icassp.2014.6854112.

[57] H.-Y. Gao and K.-M. Lam. Salient object detection using octonion with bayesian inference. In *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, oct 2014. doi: 10.1109/icip.2014.7025666.

[58] S.L. Goh and D.P. Mandic. A complex-valued RTRL algorithm for recurrent neural networks. *Neural Computation*, 16(12):2699 – 2713, December 2004. doi: 10.1162/0899766042321779.

[59] S.L. Goh and D.P. Mandic. Nonlinear adaptive prediction of complex-valued signals by complex-valued PRNN. *IEEE Transactions on Signal Processing*, 53(5):1827 – 1836, May 2005. doi: 10.1109/TSP.2005.845462.

[60] S.L. Goh and D.P. Mandic. Stochastic gradient-adaptive complex-valued nonlinear neural adaptive filters with a gradient-adaptive step size. *IEEE Transactions on Neural Networks*, 18(5):1511 – 1516, September 2007. doi: 10.1109/TNN.2007.895828.

[61] S.L. Goh and D.P. Mandic. An augmented CRTRL for complex-valued recurrent neural networks. *Neural Networks*, 20(10):1061 – 1066, December 2007. doi: 10.1016/j.neunet.2007.09.015.

[62] Weiqiang Gong, Jinling Liang, and Jinde Cao. Global μ-stability of complex-valued delayed neural networks with leakage delay. *Neurocomputing*, 168:135–144, nov 2015. doi: 10.1016/j.neucom.2015.06.006.

[63] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Pearson Prentice Hall, 2008.

[64] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[65] I.J. Goodfellow, M. Mirza, A. Courville, and Y. Bengio. Multi-prediction deep boltzmann machines. In *NIPS 2013*, 2013.

[66] K. Gu. An integral inequality in the stability problem of time-delay systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 2805 – 2810, 2000. doi: 10.1109/CDC.2000.914233.

[67] N. Guberman. On complex valued convolutional neural networks. Master's thesis, School of Computer Science and Engineering, The Hebrew University of Jerusalem, 2016. URL `https://arxiv.org/abs/1602.09046`.

[68] Runan Guo, Ziye Zhang, Xiaoping Liu, and Chong Lin. Existence, uniqueness, and exponential stability analysis for complex-valued memristor-based BAM neural networks with time delays. *Applied Mathematics and Computation*, 311:100–117, oct 2017. doi: 10.1016/j.amc.2017.05.021.

[69] Song Guo and Bo Du. Global exponential stability of periodic solution for neutral-type complex-valued neural networks. *Discrete Dynamics in Nature and Society*, 2016:1–10, 2016. doi: 10.1155/2016/1267954.

[70] M.T. Hagan and M.B. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989 – 993, November 1994. doi: 10.1109/72.329697.

[71] R. Hansch and O. Hellwich. Classification of polarimetric SAR data by complex valued neural networks. In *ISPRS Workshop*, 2009.

[72] R. Hansch and O. Hellwich. Complex-valued convolutional neural networks for object detection in PolSAR data. In *European Conference on Synthetic Aperture Radar (EUSAR)*, 2010.

[73] L. Hernandez-Garcia, A.L. Vazquez, and D.B. Rowe. Complex-valued analysis of arterial spin labeling based FMRI signals. *Magnetic Resonance in Medicine*, 62(6):1597 – 1608, 2009. doi: 10.1002/mrm.22106.

[74] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409 – 436, December 1952.

[75] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, jul 2006. doi: 10.1126/science.1127647.

[76] G.E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, July 2006. doi: 10.1162/neco.2006.18.7.1527.

[77] A. Hirose. *Complex-Valued Neural Networks*, volume 400 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-27631-6. doi: 10.1007/978-3-642-27632-3.

[78] A. Hirose. *Complex-Valued Neural Networks: Advances and Applications*. John Wiley & Sons, Inc., 2013. doi: 10.1002/9781118590072.

[79] Akira Hirose. Complex-valued neural networks: the merits and their origins. In *2009 International Joint Conference on Neural Networks*, pages 1237–1244. IEEE, 2009.

[80] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558, Apr 1982. doi: 10.1073/pnas.79.8.2554.

[81] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81(10):3088–3092, May 1984. doi: 10.1073/pnas.81.10.3088.

[82] J.J. Hopfield and D.W. Tank. "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141 – 152, July 1985. doi: 10.1007/BF00339943.

[83] J. Hu and J. Wang. Multistability and multiperiodicity analysis of complex-valued neural networks. In *Advances in Neural Networks – ISNN 2014*, pages 59–68. Springer International Publishing, 2014. doi: 10.1007/978-3-319-12436-0_8.

[84] Yujiao Huang, Huaguang Zhang, and Zhanshan Wang. Multistability of complex-valued recurrent neural networks with real-imaginary-type activation functions. *Applied Mathematics and Computation*, 229:187–200, 2014. doi: 10.1016/j.amc.2013.12.027.

[85] Z. Huang, S. Mohamad, and H. Bin. Multistability of HNNs with almost periodic stimuli and continuously distributed delays. *International Journal of Systems Science*, 40(6): 615–625, 2009. doi: 10.1080/00207720902755754.

[86] Y. Hui and M.R. Smith. MRI reconstruction from truncated data using a complex domain backpropagation neural network. In *Pacific Rim Conference on Communications, Computers, and Signal Processing (PACRIM)*, 1995. doi: 10.1109/PACRIM.1995.519582.

[87] F. Iachello. *Lie Algebras and Applications*, volume 891 of *Lecture Notes in Physics*. Springer Berlin Heidelberg, 2015. doi: 10.1007/978-3-662-44494-8.

[88] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML 2015*, 2015. URL `https://arxiv.org/abs/1502.03167`.

[89] T. Isokawa, T. Kusakabe, N. Matsui, and F. Peper. Quaternion neural network and its application. In V. Palade, R.J. Howlett, and Jai, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 2774 of *Lecture Notes in Computer Science*, pages 318 – 324, 2003. doi: 10.1007/978-3-540-45226-3_44.

[90] T. Isokawa, H. Nishimura, N. Kamiura, and N. Matsui. Associative memory in quaternionic Hopfield neural network. *International Journal of Neural Systems*, 18(02):135–145, 2008. doi: 10.1142/S0129065708001440.

[91] R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295 – 307, 1988. doi: 10.1016/0893-6080(88)90003-2.

[92] C. Jahanchahi, C.C. Took, and D.P. Mandic. On hr calculus, quaternion valued stochastic gradient, and adaptive three dimensional wind forecasting. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1 – 5. IEEE, July 2010. doi: 10.1109/IJCNN.2010.5596629.

[93] Y. Ji and F. Ding. Multiperiodicity and exponential attractivity of neural networks with mixed delays. *Circuits, Systems, and Signal Processing*, 36(6):2558–2573, 2017. doi: 10.1007/s00034-016-0420-6.

[94] E.M. Johansson, F.U. Dowla, and D.M. Goodman. Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method. *International Journal of Neural Systems*, 2(4):291 – 301, 1991. doi: 10.1142/S0129065791000261.

[95] E. Kaslik and S. Sivasundaram. Multiple periodic solutions in impulsive hybrid neural networks with delays. *Applied Mathematics and Computation*, 217(10):4890–4899, 2011. doi: 10.1016/j.amc.2010.11.025.

[96] E. Kaslik and S. Sivasundaram. Impulsive hybrid discrete-time hopfield neural networks with delays and multistability analysis. *Neural Networks*, 24(4):370–377, 2011. doi: 10.1016/j.neunet.2010.12.008.

[97] E. Kaslik and S. Sivasundaram. Multistability in impulsive hybrid hopfield neural networks with distributed delays. *Nonlinear Analysis: Real World Applications*, 12(3): 1640–1649, 2011. doi: 10.1016/j.nonrwa.2010.10.018.

[98] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR 2015*, 2015. doi: https://arxiv.org/abs/1412.6980.

[99] J. Koplinger. Dirac equation on hyperbolic octonions. *Applied Mathematics and Computation*, 182(1):443–446, nov 2006. doi: 10.1016/j.amc.2006.04.005.

[100] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):49–60, January / February 1988. doi: 10.1109/21.87054.

[101] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[102] Y. Kuroe. Models of Clifford recurrent neural networks and their dynamics. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1035 – 1041. IEEE, 2011. doi: 10.1109/IJCNN.2011.6033336.

[103] Y. Kuroe, S. Tanigawa, and H. Iima. Models of Hopfield-type Clifford neural networks and their energy functions – hyperbolic and dual valued networks –. In *International Conference on Neural Information Processing*, number 7062 in Lecture Notes in Computer Science, pages 560 – 569, 2011. doi: 10.1007/978-3-642-24955-6_67.

[104] H. Kusamichi, T. Isokawa, N. Matsui, Y. Ogawa, and K. Maeda. A new scheme for color night vision by quaternion neural network. In *International Conference on Autonomous Robots and Agents*, pages 101 – 106, December 2004.

[105] V. Laparra, M.U. Gutmann, J. Malo, and A. Hyvarinen. Complex-valued independent component analysis of natural images. In *International Conference on Artificial Neural Networks (ICANN)*, 2011.

[106] Y. LeCun and Y. Bengio. *Convolutional networks for images, speech, and time series*, pages 255 – 258. MIT Press, 1995.

[107] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems (NIPS)*, 1989.

[108] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278 – 2324, 1998. doi: 10.1109/5.726791.

[109] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In *International Symposium on Circuits and Systems (ISCAS)*, 2010. doi: 10.1109/ISCAS.2010.5537907.

[110] D.L. Lee and W.J. Wang. A multivalued bidirectional associative memory operating on a complex domain. *Neural Networks*, 11(9):1623 – 1635, December 1998. doi: 10.1016/S0893-6080(98)00078-1.

[111] H. Li, N.M. Correa, P.A. Rodriguez, V.D. Calhoun, and T. Adali. Application of independent component analysis with adaptive density model to complex-valued fMRI data. *IEEE Transactions on Biomedical Engineering*, 58(10):2794 – 2803, 2011. doi: 10.1109/TBME.2011.2159841.

[112] J. Liang, W. Gong, and T. Huang. Multistability of complex-valued neural networks with discontinuous activation functions. *Neural Networks*, 84:125–142, 2016. doi: 10.1016/j.neunet.2016.08.008.

[113] Jing Liang, Kelin Li, Qiankun Song, Zhenjiang Zhao, Yurong Liu, and Fuad E. Alsaadi. State estimation of complex-valued neural networks with two additive time-varying delays. *Neurocomputing*, 309:54–61, oct 2018. doi: 10.1016/j.neucom.2018.05.003.

[114] X. Liao, G. Chen, and E.N. Sanchez. LMI-based approach for asymptotically stability analysis of delayed neural networks. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 49(7):1033 – 1039, 2002. doi: 10.1109/TCSI.2002.800842.

[115] X. Liu and T. Chen. Global exponential stability for complex-valued recurrent neural networks with asynchronous time delays. *IEEE Transactions on Neural Networks and Learning Systems*, 27(3):593 – 606, 2016. doi: 10.1109/TNNLS.2015.2415496.

[116] Y. Liu, P. Xu, J. Lu, and J. Liang. Global stability of Clifford-valued recurrent neural networks with time delays. *Nonlinear Dynamics*, 84(2):767 – 777, 2016. doi: 10.1007/s11071-015-2526-y.

[117] Y. Liu, D. Zhang, J. Lu, and J. Cao. Global μ-stability criteria for quaternion-valued neural networks with unbounded time-varying delays. *Information Sciences*, 2016. doi: 10.1016/j.ins.2016.04.033.

[118] D.G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*, volume 116 of *International Series in Operations Research & Management Science*. Springer, 2008. doi: 10.1007/978-0-387-74503-9.

[119] Danilo P. Mandic and Vanessa Su Lee Goh. *Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models*. Wiley-Blackwell, apr 2009. doi: 10.1002/9780470742624.

[120] D.P. Mandic and J. Chambers. *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. John Wiley & Sons, Inc., New York, NY, USA, 2001. doi: 10.1002/047084535X.

[121] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431 – 441, June 1963. doi: 10.1137/0111030.

[122] M.F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525 – 533, 1993. doi: 10.1016/S0893-6080(05)80056-5.

[123] G. Montavon and K.-R. Müller. Deep boltzmann machines and the centering trick. In G. Montavon, G.B. Orr, and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 621–637. Springer, 2012. doi: 10.1007/978-3-642-35289-8_33.

[124] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[125] X. Nie and J. Cao. Multistability of competitive neural networks with time-varying and distributed delays. *Nonlinear Analysis: Real World Applications*, 10(2):928–942, 2009. doi: 10.1016/j.nonrwa.2007.11.014.

[126] X. Nie and W. X. Zheng. Multistability of neural networks with discontinuous non-monotonic piecewise linear activation functions and time-varying delays. *Neural Networks*, 65:65–79, 2015. doi: 10.1016/j.neunet.2015.01.007.

[127] X. Nie, W. X. Zheng, and J. Cao. Multistability of memristive cohen–grossberg neural networks with non-monotonic piecewise linear activation functions and time-varying delays. *Neural Networks*, 71:27–36, 2015. doi: 10.1016/j.neunet.2015.07.009.

[128] T. Nitta. A back-propagation algorithm for neural networks based on 3d vector product. In *International Joint Conference on Neural Netwoks (IJCNN)*, volume 1, pages 589 – 592. IEEE, 1993. doi: 10.1109/IJCNN.1993.713984.

[129] T. Nitta. An extension of the back-propagation algorithm to three dimensions by vector product. In *International Conference on Tools with Artificial Intelligence (TAI)*, pages 460 – 461. IEEE, 1993. doi: 10.1109/TAI.1993.634002.

[130] T. Nitta. Generalization ability of the three-dimensional back-propagation network. In *International Conference on Neural Networks*, volume 5, pages 2895 – 2900. IEEE, 1994. doi: 10.1109/ICNN.1994.374691.

[131] T. Nitta. Three-dimensional vector valued neural network and its generalization ability. *Neural Information Processing - Letters and Reviews*, 10(10):237 – 242, 2006.

[132] T. Nitta. N-dimensional vector neuron. In *IJCAI Workshop on Complex-Valued Neural Networks and Neuro-Computing: Novel Methods, Applications and Implementations*, pages 2 – 7, 2007.

[133] T. Nitta. *Complex-Valued Neural Networks: Advances and Applications*, chapter N-Dimensional Vector Neuron and its Application to the N-Bit Parity Problem, pages 59 – 74. John Wiley & Sons, Inc., 2013. doi: 10.1002/9781118590072.ch3.

[134] T. Nitta and S. Buchholz. On the decision boundaries of hyperbolic neurons. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2974 – 2980. IEEE, 2008. doi: 10.1109/IJCNN.2008.4634216.

[135] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer New York, 1999. doi: 10.1007/978-0-387-40065-5.

[136] S. Okubo. *Introduction to Octonion and Other Non-Associative Algebras in Physics*. Cambridge University Press, 1995. doi: 10.1017/CBO9780511524479.

[137] A. Pande and V. Goel. Complex-valued neural network in image recognition a study on the effectiveness of radial basis function. *World Academy of Science, Engineering and Technology*, 20:220 – 225, 2007.

[138] PooGyeon Park, Jeong Wan Ko, and Changki Jeong. Reciprocally convex approach to stability of systems with time-varying delays. *Automatica*, 47(1):235–238, jan 2011. doi: 10.1016/j.automatica.2010.10.014.

[139] J.K. Pearson and D.L. Bisset. Back propagation in a Clifford algebra. In *International Conference on Artificial Neural Networks*, volume 2, pages 413 – 416, 1992.

[140] J.K. Pearson and D.L. Bisset. Neural networks in the Clifford domain. In *International Conference on Neural Networks*, volume 3, pages 1465 – 1469. IEEE, 1994. doi: 10.1109/ICNN.1994.374502.

[141] E. Polak and G. Ribiere. Note sur la convergence de méthodes de directions conjuguées. *Revue Française d'Informatique et de Recherche Opérationnelle*, 3(16):35 – 43, 1969.

[142] C.-A. Popa. Enhanced gradient descent algorithms for complex-valued neural networks. In *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 272 – 279. IEEE, September 2014. doi: 10.1109/SYNASC.2014.44.

[143] C.-A. Popa. Scaled conjugate gradient learning for complex-valued neural networks. In R. Matoušek, editor, *Mendel 2015*, volume 378 of *Advances in Intelligent Systems and Computing*, pages 221 – 233. Springer International Publishing, June 2015. doi: 10.1007/978-3-319-19824-8_18.

[144] C.-A. Popa. Matrix-valued neural networks. In R. Matoušek, editor, *Mendel 2015*, volume 378 of *Advances in Intelligent Systems and Computing*, pages 245 – 255, 2015. doi: 10.1007/978-3-319-19824-8_20.

[145] C.-A. Popa. Quasi-newton learning methods for complex-valued neural networks. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2015. doi: 10.1109/IJCNN.2015.7280450.

[146] C.-A. Popa. Lie algebra-valued neural networks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1 – 6. IEEE, July 2015. doi: 10.1109/IJCNN.2015.7280787.

[147] C.-A. Popa. Conjugate gradient algorithms for complex-valued neural networks. In *Neural Information Processing – ICONIP 2015*, pages 412 – 422, 2015. doi: 10.1007/978-3-319-26535-3_47.

[148] C.-A. Popa. Lie algebra-valued hopfield neural networks. In *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE, 2015. doi: 10.1109/SYNASC.2015.41.

[149] C.-A. Popa. Octonion-valued neural networks. In *Artificial Neural Networks and Machine Learning – ICANN 2016*, pages 435 – 443, 2016. doi: 10.1007/978-3-319-44778-0_51.

[150] C.-A. Popa. Levenberg-marquardt learning algorithm for quaternion-valued neural networks. In *18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 272 – 278, 2016. doi: 10.1109/SYNASC.2016.050.

[151] C.-A. Popa. Scaled conjugate gradient learning for quaternion-valued neural networks. In *Neural Information Processing. ICONIP 2016*, pages 243 – 252, 2016. doi: 10.1007/978-3-319-46675-0_27.

[152] C.-A. Popa. Matrix-valued hopfield neural networks. In *Advances in Neural Networks – ISNN 2016*, pages 127–134. Springer International Publishing, 2016. doi: 10.1007/978-3-319-40663-3_15.

[153] C.-A. Popa. Complex-valued convolutional neural networks for real-valued image classification. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 816 – 822. IEEE, may 2017. doi: 10.1109/IJCNN.2017.7965936.

[154] C.-A. Popa. Conjugate gradient algorithms for quaternion-valued neural networks. In R. Matoušek, editor, *Recent Advances in Soft Computing. ICSC-MENDEL 2016*, volume 576 of *Advances in Intelligent Systems and Computing*, pages 176 – 185, 2017. doi: 10.1007/978-3-319-58088-3_17.

[155] C.-A. Popa. Quasi-newton learning methods for quaternion-valued neural networks. In *Advances in Computational Intelligence. IWANN 2017*, pages 362 – 374, 2017. doi: 10.1007/978-3-319-59153-7_32.

[156] C.-A. Popa. Octonion-valued bidirectional associative memories. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017. doi: 10.1109/IJCNN.2017.7965931.

[157] C.-A. Popa. Global asymptotic stability for octonion-valued neural networks with delay. In *Advances in Neural Networks - ISNN 2017*, pages 439–448. Springer International Publishing, 2017. doi: 10.1007/978-3-319-59072-1_52.

[158] C.-A. Popa. Exponential stability for delayed octonion-valued recurrent neural networks. In *Advances in Computational Intelligence*, pages 375–385. Springer International Publishing, 2017. doi: 10.1007/978-3-319-59153-7_33.

[159] C.-A. Popa. Asymptotic stability of delayed octonion-valued neural networks with leakage delay. In *Neural Information Processing (ICONIP 2017)*, pages 728–736. Springer International Publishing, 2017. doi: 10.1007/978-3-319-70090-8_73.

[160] C.-A. Popa. Global asymptotic stability for matrix-valued recurrent neural networks with time delays. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017. doi: 10.1109/IJCNN.2017.7966423.

[161] C.-A. Popa. Global exponential stability for matrix-valued neural networks with time delay. In *Advances in Neural Networks - ISNN 2017*, pages 429–438. Springer International Publishing, 2017. doi: 10.1007/978-3-319-59072-1_51.

[162] C.-A. Popa. Exponential stability of matrix-valued BAM neural networks with time-varying delays. In *Neural Information Processing (ICONIP 2017)*, pages 718–727. Springer International Publishing, 2017. doi: 10.1007/978-3-319-70090-8_72.

[163] C.-A. Popa. Lie algebra-valued bidirectional associative memories. In *Recent Advances in Soft Computing (ICSC-MENDEL 2016*, pages 127–135. Springer International Publishing, 2017. doi: 10.1007/978-3-319-58088-3_12.

[164] C.-A. Popa. Learning algorithms for quaternion-valued neural networks. *Neural Processing Letters*, 47(3):949–973, 2018. doi: 10.1007/s11063-017-9716-1.

[165] C.-A. Popa. Enhanced gradient descent algorithms for quaternion-valued neural networks. In V.E. Balas, L.C. Jain, and M.M. Balas, editors, *Soft Computing Applications. SOFA 2016*, volume 634 of *Advances in Intelligent Systems and Computing*, 2018. doi: 10.1007/978-3-319-62524-9_5.

[166] C.-A. Popa. Global exponential stability of neutral-type octonion-valued neural networks with time-varying delays. *Neurocomputing*, 309:117–133, 2018. doi: 10.1016/j.neucom.2018.05.004.

[167] C.-A. Popa. Global exponential stability of octonion-valued neural networks with leakage delay and mixed delays. *Neural Networks*, 105:277–293, 2018. doi: 10.1016/j.neunet.2018.05.006.

[168] C.-A. Popa. Deep hybrid real-complex-valued convolutional neural networks for image classification. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018. doi: 10.1109/IJCNN.2018.8489274.

[169] C.-A. Popa. Complex-valued stacked denoising autoencoders. In *Advances in Neural Networks – ISNN 2018*, pages 64–71. Springer International Publishing, 2018. doi: 10.1007/978-3-319-92537-0_8.

[170] C.-A. Popa. Complex-valued deep belief networks. In *Advances in Neural Networks – ISNN 2018*, pages 72–78. Springer International Publishing, 2018. doi: 10.1007/978-3-319-92537-0_9.

[171] C.-A. Popa. Complex-valued deep boltzmann machines. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018. doi: 10.1109/IJCNN.2018.8489359.

[172] C.-A. Popa. Matrix-valued bidirectional associative memories. In *Soft Computing Applications (SOFA 2016)*, pages 36–44. Springer International Publishing, 2018. doi: 10.1007/978-3-319-62524-9_4.

[173] C.-A. Popa. Global μ-stability of neutral-type impulsive complex-valued BAM neural networks with leakage delay and unbounded time-varying delays. *Neurocomputing*, 376:73–94, 2020. doi: 10.1016/j.neucom.2019.09.008.

[174] C.-A. Popa. Dissipativity of impulsive matrix-valued neural networks with leakage delay and mixed delays. *Neurocomputing*, 405:85–95, 2020. doi: 10.1016/j.neucom.2020.03.042.

[175] C.-A. Popa and C. Cernăzanu-Glăvan. Fourier transform-based image classification using complex-valued convolutional neural networks. In *Advances in Neural Networks – ISNN 2018*, pages 300–309. Springer International Publishing, 2018. doi: 10.1007/978-3-319-92537-0_35.

[176] C.-A. Popa and E. Kaslik. Multistability and multiperiodicity in impulsive hybrid quaternion-valued neural networks with mixed delays. *Neural Networks*, 99:1–18, 2018. doi: 10.1016/j.neunet.2017.12.006.

[177] M.J.D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12(1):241 – 254, 1977. doi: 10.1007/BF01593790.

[178] C.M. Reeves and R. Fletcher. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149 – 154, 1964. doi: 10.1093/comjnl/7.2.149.

[179] D.P. Reichert and T. Serre. Neuronal synchrony in complex-valued deep networks. In *International Conference on Learning Representations*, 2014.

[180] M. Riedmiller. Advanced supervised learning in multi-layer perceptrons - from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3): 265 – 278, July 1994. doi: 10.1016/0920-5489(94)90017-5.

[181] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE International Conference on Neural Networks*, volume 1, pages 586 – 591. IEEE, March 1993. doi: 10.1109/ICNN.1993.298623.

[182] P.A. Rodriguez, N.M. Correa, T. Adali, and V.D. Calhoun. Quality map thresholding for de-noising of complex-valued FMRI data and its application to ICA of FMRI. In *International Workshop on Machine Learning for Signal Processing*, 2009. doi: 10.1109/MLSP.2009.5306263.

[183] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In *AISTATS 2009*, 2009.

[184] R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. In *AISTATS 2010*, 2010.

[185] R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *ICML 2008*, 2008.

[186] R. Salakhutdinov, J. B. Tenenbaum, and A. Torralba. Learning with hierarchical-deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1958–1971, aug 2013. doi: 10.1109/tpami.2012.269.

[187] S. Samadi, M. Cetin, and M.A. Masnadi-Shirazi. Sparse signal representation for complex-valued imaging. In *Digital Signal Processing Workshop and Signal Processing Education Workshop (DSP/SPE)*, pages 365 – 370, 2009. doi: 10.1109/DSP.2009.4785950.

[188] A.M. Sarroff, V. Shepardson, and M.A. Casey. Learning representations using complex-valued nets. In *International Conference on Learning Representations*, 2016.

[189] D.H. Sattinger and O.L. Weaver. *Lie Groups and Algebras with Applications to Physics, Geometry, and Mechanics*, volume 61 of *Applied Mathematical Sciences*. Springer-Verlag New York, 1986. doi: 10.1007/978-1-4757-1910-9.

[190] R. Savitha, S. Suresh, and N. Sundararajan. A fully complex-valued radial basis function network and its learning algorithm. *International Journal of Neural Systems*, 19(4):253 – 267, August 2009. doi: 10.1142/S0129065709002026.

[191] R. Savitha, S. Suresh, N. Sundararajan, and P. Saratchandran. A new learning algorithm with logarithmic performance index for complex-valued neural networks. *Neurocomputing*, 72(16 - 18):3771 – 3781, October 2009. doi: 10.1016/j.neucom.2009.06.004.

[192] R. Savitha, S. Suresh, and N. Sundararajan. A meta-cognitive learning algorithm for a fully complex-valued relaxation network. *Neural Networks*, 32:209 – 218, August 2012. doi: 10.1016/j.neunet.2012.02.015.

[193] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61: 85 – 117, 2015. doi: doi:10.1016/j.neunet.2014.09.003.

[194] A. Seuret and F. Gouaisbaut. Wirtinger-based integral inequality: Application to time-delay systems. *Automatica*, 49(9):2860–2866, sep 2013. doi: 10.1016/j.automatica.2013.05.030.

[195] D.F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24(111):647 – 656, July 1970. doi: 10.1090/S0025-5718-1970-0274029-X.

[196] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR 2015*, 2015. URL `https://arxiv.org/abs/1409.1556`.

[197] K. M. Snopek. Quaternions and octonions in signal processing – fundamentals and some new results. *Przeglad Telekomunikacyjny + Wiadomosci Telekomunikacyjne*, 6:618 – 622, 2015.

[198] Q. Song and Z. Zhao. Stability criterion of complex-valued neural networks with both leakage delay and time-varying delays on time scales. *Neurocomputing*, 171:179 – 184, 2016. doi: 10.1016/j.neucom.2015.06.032.

[199] Q. Song, H. Yan, Z. Zhao, and Y. Liu. Global exponential stability of complex-valued neural networks with both time-varying delays and impulsive effects. *Neural Networks*, 79:108 – 116, 2016. doi: 10.1016/j.neunet.2016.03.007.

[200] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS 2012*, 2012.

[201] N. Srivastava, R. Salakhutdinov, and G. Hinton. Modeling documents with a deep boltzmann machine. In *UAI 2013*, 2013.

[202] K. Subramanian and P. Muthukumar. Existence, uniqueness, and global asymptotic stability analysis for delayed complex-valued cohen–grossberg BAM neural networks. *Neural Computing and Applications*, 29(9):565–584, sep 2016. doi: 10.1007/s00521-016-2539-6.

[203] K. Subramanian and P. Muthukumar. Global asymptotic stability of complex-valued neural networks with additive time-varying delays. *Cognitive Neurodynamics*, 11(3): 293–306, mar 2017. doi: 10.1007/s11571-017-9429-1.

[204] A.B. Suksmono and A. Hirose. Interferometric SAR image restoration using Monte Carlo Metropolis method. *IEEE Transactions on Signal Processing*, 50(2):290 – 298, 2002. doi: 10.1109/78.978384.

[205] S. Suresh, R. Savitha, and N. Sundararajan. A sequential learning algorithm for complex-valued self-regulating resource allocation network-csran. *IEEE Transactions on Neural Networks*, 22(7):1061 – 1072, July 2011. doi: 10.1109/TNN.2011.2144618.

[206] D.W. Tank and J.J. Hopfield. Simple "neural" optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, 33(5):533 – 541, May 1986. doi: 10.1109/TCS.1986.1085953.

[207] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood and gradient. In *ICML 2008*, 2008.

[208] T. Tollenaere. Supersab: Fast adaptive back propagation with good scaling properties. *Neural Networks*, 3(5):561 – 573, 1990. ISSN 0893-6080. doi: 10.1016/0893-6080(90)90006-7.

[209] C.C. Took and D.P. Mandic. The quaternion lms algorithm for adaptive filtering of hypercomplex processes. *IEEE Transactions on Signal Processing*, 57(4):1316 – 1327, April 2009. doi: 10.1109/TSP.2008.2010600.

[210] C.C. Took and D.P. Mandic. A quaternion widely linear adaptive filter. *IEEE Transactions on Signal Processing*, 58(8):4427 – 4431, August 2010. doi: 10.1109/TSP.2010.2048323.

[211] C.C. Took and D.P. Mandic. Quaternion-valued stochastic gradient-based adaptive iir filtering. *IEEE Transactions on Signal Processing*, 58(7):3895 – 3901, July 2010. doi: 10.1109/TSP.2010.2047719.

[212] C.C. Took, D.P. Mandic, and J. Benesty. Study of the quaternion lms and four-channel lms algorithms. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3109 – 3112. IEEE, April 2009. doi: 10.1109/ICASSP.2009.4960282.

[213] C.C. Took, D.P. Mandic, and K. Aihara. Quaternion-valued short term forecasting of wind profile. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1 – 6. IEEE, July 2010. doi: 10.1109/IJCNN.2010.5596690.

[214] C.C. Took, G. Strbac, K. Aihara, and D.P. Mandic. Quaternion-valued short-term joint forecasting of three-dimensional wind and atmospheric parameters. *Renewable Energy*, 36(6):1754 – 1760, June 2011. doi: 10.1016/j.renene.2010.12.013.

[215] M. Tygert, J. Bruna, S. Chintala, Y. LeCun, S. Piantino, and A. Szlam. A mathematical motivation for complex-valued convolutional networks. *Neural Computation*, 28(5):815 – 825, May 2016. doi: 10.1162/NECO_a_00824.

[216] M.E. Valle. A novel continuous-valued quaternionic Hopfield neural network. In *Brazilian Conference on Intelligent Systems (BRACIS)*, pages 97 – 102. IEEE, October 2014. doi: 10.1109/BRACIS.2014.28.

[217] J.R. Vallejo and E. Bayro-Corrochano. Clifford Hopfield neural networks. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3609 – 3612. IEEE, June 2008. doi: 10.1109/IJCNN.2008.4634314.

[218] G. Velmurugan, R. Rakkiyappan, and S. Lakshmanan. Passivity analysis of memristor-based complex-valued neural networks with time-varying delays. *Neural Processing Letters*, 42(3):517–540, aug 2014. doi: 10.1007/s11063-014-9371-8.

[219] G. Velmurugan, R. Rakkiyappan, and Jinde Cao. Further analysis of global μ-stability of complex-valued neural networks with unbounded time-varying delays. *Neural Networks*, 67:14–27, jul 2015. doi: 10.1016/j.neunet.2015.03.007.

[220] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML 2008*, 2008.

[221] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[222] Jinling Wang, Haijun Jiang, Tianlong Ma, and Cheng Hu. Delay-dependent dynamical analysis of complex-valued memristive neural networks: Continuous-time and discrete-time cases. *Neural Networks*, 101:33–46, may 2018. doi: 10.1016/j.neunet.2018.01.015.

[223] M. Wang, C.C. Took, and D.P. Mandic. A class of fast quaternion valued variable stepsize stochastic gradient learning algorithms for vector sensor processes. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2783 – 2786. IEEE, August 2011. doi: 10.1109/IJCNN.2011.6033585.

[224] R. Wang, G. Xiang, and F. Zhang. L1-norm minimization for octonion signals. In *2016 International Conference on Audio, Language and Image Processing (ICALIP)*. IEEE, jul 2016. doi: 10.1109/icalip.2016.7846602.

[225] Zengyun Wang and Lihong Huang. Global stability analysis for delayed complex-valued BAM neural networks. *Neurocomputing*, 173:2083–2089, jan 2016. doi: 10.1016/j.neucom.2015.09.086.

[226] R.L. Watrous. Learning algorithms for connectionist networks: Applied gradient methods of nonlinear optimization. Technical Reports (CIS) MS-CIS-88-62, University of Pennsylvania, July 1988.

[227] B. Widrow, J. McCool, and M. Ball. The complex LMS algorithm. *Proceedings of the IEEE*, 63(4):719 – 720, April 1975. doi: 10.1109/PROC.1975.9807.

[228] Y. Xia, B. Jelfs, M.M. Van Hulle, J.C. Principe, and D.P. Mandic. An augmented echo state network for nonlinear adaptive filtering of complex noncircular signals. *IEEE Transactions on Neural Networks*, 22(1):74 – 83, January 2011. doi: 10.1109/TNN.2010.2085444.

[229] Y. Xia, C. Jahanchahi, and D.P. Mandic. Quaternion-valued echo state networks. *IEEE Transactions on Neural Networks and Learning Systems*, 26(4):663 – 673, April 2015. doi: 10.1109/TNNLS.2014.2320715.

[230] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine and Learning Algorithms. https://arxiv.org/abs/1708.07747, 2017.

[231] D. Xu, Y. Xia, and D.P. Mandic. Optimization in quaternion dynamic systems: Gradient, hessian, and learning algorithms. *IEEE Transactions on Neural Networks and Learning Systems*, 27(2):249 – 261, February 2016. doi: 10.1109/TNNLS.2015.2440473.

[232] Desheng Xu and Manchun Tan. Delay-independent stability criteria for complex-valued BAM neutral-type neural networks with time delays. *Nonlinear Dynamics*, 89(2):819–832, mar 2017. doi: 10.1007/s11071-017-3486-1.

[233] Q. Xu and D. Ma. Applications of lie groups and lie algebra to computer vision a brief survey. In *International Conference on Systems and Informatics (ICSAI)*, pages 2024 – 2029. IEEE, May 2012. doi: 10.1109/ICSAI.2012.6223449.

[234] M.-C. Yu, Q.-H. Lina, L.-D. Kuang, X.-F. Gong, F. Cong, and V.D. Calhoun. ICA of full complex-valued fMRI data using phase information of spatial maps. *Journal of Neuroscience Methods*, 249:75 – 91, 2015. doi: 10.1016/j.jneumeth.2015.03.036.

[235] Z. Zeng and W. X. Zheng. Multistability of neural networks with time-varying delays and concave-convex characteristics. *IEEE Transactions on Neural Networks and Learning Systems*, 23(2):293–305, 2012. doi: 10.1109/TNNLS.2011.2179311.

[236] Chuan-Ke Zhang, Yong He, Lin Jiang, and Min Wu. Stability analysis for delayed neural networks considering both conservativeness and complexity. *IEEE Transactions on Neural Networks and Learning Systems*, 27(7):1486–1501, jul 2016. doi: 10.1109/tnnls.2015.2449898.

[237] F. Zhang, editor. *The Schur Complement and Its Applications*. Springer US, 2005. doi: 10.1007/b105056.

[238] Huaguang Zhang, Zhanshan Wang, and Derong Liu. A comprehensive review of stability analysis of continuous-time recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 25(7):1229–1262, jul 2014. doi: 10.1109/TNNLS.2014.2317880. URL `http://dx.doi.org/10.1109/TNNLS.2014.2317880`.

[239] Cheng-De Zheng, Yun Gu, Wenlong Liang, and Zhanshan Wang. Novel delay-dependent stability criteria for switched hopfield neural networks of neutral type. *Neurocomputing*, 158:117–126, jun 2015. doi: 10.1016/j.neucom.2015.01.061.

[240] J. Zhu and J. Sun. Global exponential stability of Clifford-valued recurrent neural networks. *Neurocomputing*, 173, Part 3:685 – 689, 2016. doi: 10.1016/j.neucom.2015.08.016.